School of Computing Science

Data Integration Strategies for Informing Computational Design in Synthetic Biology

Göksel Mısırlı

Submitted for the degree of Doctor of Philosophy in the School of Computing Science, Newcastle University

February 2013

© 2013, Göksel Mısırlı

Declaration

I declare that this thesis is my own work unless otherwise stated. No part of this thesis has previously been submitted for a degree or other qualification at Newcastle University or any other institution.

Göksel Mısırlı

February 2013

Publications Arising From This Thesis

1. Misirli, G., J.S. Hallinan, T. Yu, J.R. Lawson, S.M. Wimalaratne, M.T. Cooling, and A. Wipat, Model annotation for synthetic biology: automating model to nucleotide sequence conversion. Bioinformatics, 2011. **27**(7): p. 973-979.

2. Misirli, G., J. Hallinan, J. Weile, S. Cockell, and A. Wipat, BacillOndex: Data integration and visualisation for *Bacillus subtilis*, in *School of Computing Science Technical Report Series 1237*. 2011, School of Computing Science, University of Newcastle upon Tyne.

3. Cooling, M.T., V. Rouilly, G. Misirli, J. Lawson, T. Yu, J. Hallinan, and A. Wipat, Standard virtual biological parts: a repository of modular modeling components for synthetic biology. Bioinformatics, 2010. **26**(7): p. 925-931.

4. Hallinan, J.S., G. Misirli, and A. Wipat, Evolutionary computation for the design of a stochastic switch for synthetic genetic circuits, in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. 2010, IEEE: Buenos Aires, Argentina. p. 768-774.

5. Galdzicki, M., M.L. Wilson, C.A. Rodriguez, L. Adam, A. Adler, J.C. Anderson, J. Beal, D. Chandran, D. Densmore, O.A. Drory, D. Endy, J.H. Gennari, R. Grünberg, T.S. Ham, A. Kuchinsky, M.W. Lux, C. Madsen, G. Misirli, C.J. Myers, J. Peccoud, H. Plahar, M.R. Pocock, N. Roehner, T.F. Smith, G.-B. Stan, A. Villalobos, A. Wipat, and H.M. Sauro, Synthetic Biology Open Language (SBOL) Version 1.0.0, in *BBF RFC #84*. 2011.

6. Swainston, N., D. Waltemath, A. Lister, F. Bergmann, R. Henkel, S. Hoops, M. Hucka, N. Juty, S. Keating, C. Knuepfer, F. Krause, C. Laibe, W. Liebermeister, C. Lloyd, G. Misirli, M. Schulz, M. Taschuk, and N. Le Novère, SBML Level 3 Package Proposal: Annotation. Nature Precedings, 2011.

Abstract

The potential design space for biological systems is complex, vast and multidimensional. Therefore, effective large-scale synthetic biology requires computational design and simulation. By constraining this design space, the time- and cost-efficient design of biological systems can be facilitated. One way in which a tractable design space can be achieved is to use the extensive and growing amount of biological data available to inform the design process. By using existing knowledge design efforts can be focused on biologically plausible areas of design space. However, biological data is large, incomplete, heterogeneous, and noisy. Data must be integrated in a systematic fashion in order to maximise its benefit. To date, data integration has not been widely applied to design in synthetic biology. The aim of this project is to apply data integration techniques to facilitate the efficient design of novel biological systems. The specific focus is on the development and application of integration techniques for the design of genetic regulatory networks in the model bacterium *Bacillus subtilis*.

A dataset was constructed by integrating data from a range of sources in order to capture existing knowledge about *B. subtilis* 168. The dataset is represented as a computationally-accessible, semantically-rich network which includes information concerning biological entities and their relationships. Also included are sequence-based features mined from the *B. subtilis* genome, which are a useful source of parts for synthetic biology. In addition, information about the interactions of these parts has been captured, in order to facilitate the construction of circuits with desired behaviours.

This dataset was also modelled in the form of an ontology, providing a formal specification of parts and their interactions. The ontology is a major step towards the unification of the data required for modelling with a range of part catalogues specifically designed for synthetic biology. The data from the ontology is available to existing reasoners for implicit knowledge extraction. The ontology was applied to the automated identification of promoters, operators and coding sequences. Information from the ontology was also used to generate dynamic models of parts.

The work described here contributed to the development of a formalism called Standard Virtual Parts (SVPs), which aims to represent models of biological parts in a standardised manner. SVPs comprise a mapping between biological parts and modular computational models. A genetic circuit designed at a part-level abstraction can be

investigated in detail by analysing a circuit model composed of SVPs. The ontology was used to construct SVPs in the form of standard Systems Biology Markup Language models. These models are publicly available from a computationally-accessible repository, and include metadata which facilitates the computational composition of SVPs in order to create models of larger biological systems.

To test a genetic circuit *in vitro* or *in vivo*, the genetics elements necessary to encode the enitites in the *in silico* model, and their associated behaviour, must be derived. Ultimately, this process results in the specification for synthesisable DNA sequence. For large models, particularly those that are produced computationally, the transformation process is challenging. To automate this process, a model-to-sequence conversion algorithm was developed. The algorithm was implemented as a Java application called MoSeC. Using MoSeC, both CellML and SBML models built with SVPs can be converted into DNA sequences ready to synthesise.

Selection of the host bacterial cell for a synthetic genetic circuit is very important. In order not to interfere with the existing cellular machinery, orthogonal parts from other species are used since these parts are less likely to have undesired interactions with the host. In order to find orthogonal transcription factors (OTFs), and their target binding sequences, a subset of the data from the integrated *B. subtilis* dataset was used. *B. subtilis* gene regulatory networks were used to re-construct regulatory networks in closely related *Bacillus* species. The system, called BacillusRegNet, stores both experimental data for *B. subtilis* and homology predictions in other species. BacillusRegNet was mined to extract OTFs and their binding sequences, in order to facilitate the engineering of novel regulatory networks in other *Bacillus* species. Although the techniques presented here were demonstrated using *B. subtilis*, they can be applied to any other organism. The approaches and tools developed as part of this project demonstrate the utility of this novel integrated approach to synthetic biology.

To Sümeyye

and to Ediz and my parents.

Acknowledgements

I would like to thank to my supervisors Dr. Jennifer Hallinan and Prof. Anil Wipat for guiding me throughout my research with their invaluable support and advice. I am privileged to have both of them as my supervisors. I also thank Dr. Matthew Pocock for lively and useful discussions for various parts of my research, all of which I cannot even list here. I gratefully acknowledge the funding from EPSRC, NSF and the Newcastle University School of Computing Science.

I also thank to all those who helped me throughout this thesis, particularly Dr. Mike Cooling for his work on the initial formulation of modular models of biological parts and his support regarding the annotation of dynamic models. And thanks to Dr. Jan Baumbach and his team for their support to set up BacillusRegNet, and later for their work to update the system and upload information about additional organisms. I thank Dr. Keith Flanagan and the Newcastle University Bioinformatics Support Unit for providing servers to host my applications. Thanks also to Dr. Jennifer Hallinan, Prof. Anil Wipat, Dr. Matthew Pocock, Dr. Keith Flanagan, Dr. Katherine James, Jennifer Warrender, Sungshic Park, Michael Bell and other members of the writing group for giving feedback about my thesis. In addition, thanks to Dr. Wendy Smith, Dr. Susanne Pohl, Beth Lawry, Matthew Collison, and all present and previous colleagues from the PhD office at the School of Computing Science and the Centre for Bacterial Cell Biology for a friendly environment to carry out my research, and to the members of the Newcastle University's 2009 International Genetically Engineered Machine competition team for a wonderful experience.

I am also grateful to my parents for their encouragement and support throughout my life. And very special thanks go to my wife Sümeyye. Without her support, I could not have started or continued my research. And also thanks to her for raising Ediz, who has been my motivation to submit this thesis.

Contents

Abbreviations

API Application Programming Interface

CAD Computer-Aided Design

CCO Cell Cycle Ontology

CDS Coding Sequence

CO Cell Ontology

COGs Clusters of Orthologous Groups

DL Description Logic

DSL Domain Specific Language

EC Enzyme Commission

FFL Feed-Forward Loop

FTP File Transfer Protocol

GO Gene Ontology

GOA Gene Ontology Annotation

GUI Graphical User Interface

KEGG Kyoto Encyclopedia of Genes and Genomes

KO KEGG Orthology

MIRIAM Minimum Information Requested in the Annotation of Models

OBO Open Biological and Biomedical Ontologies

ODE Ordinary Differential Equation

OTF Orthogonal Transcription Factor

OWA Open World Assumption

OWL Web Ontology Language

OXL Ondex Exchange Language

PMID Pubmed ID

PoPS Polymerases per Second

PRO Protein Ontology

PWM Position Weight Matrix

RBS Ribosome Binding Site

RDF Resource Description Framework

RDFS RDF Schema

REST Representational State Transfer

RiPS Ribosomes per Second

RNAP RNA Polymerase

SBML Systems Biology Markup Language

SBOL Synthetic Biology Open Language

SCS Self-Consistent Set

SD Shine-Dalgarno

SO Sequence Ontology

SOAP Simple Object Access Protocol

SVP Standard Virtual Part

TCS Two-Component System

TF Transcription Factor

TFBS Transcription Factor Binding Site

TSS Transcription Start Site

URI Uniform Resource Identifier

URL Uniform Resource Locater

XML eXtended Markup Language

Chapter 1. Introduction

1.1 Synthetic biology

Synthetic biology has been defined as the application of engineering principles to the design of biological systems [1]. This field has huge potential for developing new approaches to a wide range of problems such as drug and biomolecule production, and bioremediation.

Synthetic biology can be applied to the design and construction of a wide range of novel biological applications, and has huge commercial potential. The global value of synthetic biology reached $1.1 billion in 2010, and it is expected that this value will reach $10.8 billion by 2016 [2]. Potential areas of application include biosensing [3], drug production [4], tissue engineering [5], bioremediation [3, 6], and the production of biofuels [7, 8] and biomaterials [9].

Analogies drawn from engineering, such as modularity, standardisation, and abstraction, can facilitate the assembly of complex biological systems from individual genetic parts with well-defined functions [10]. However, the engineering of biological systems is challenging. These systems are noisy, complex and multidimensional, and can be formed from different types and numbers of biological molecules such as DNA and proteins. Biological parts include promoters, ribosome binding sites (RBSs) and coding sequences (CDSs) [11] (Figure 1.1). The MIT Registry of Standard Biological Parts is an example of a repository of these parts [12], and includes information about over 13,000 parts as of March 2012 [13]. The solution space of possible genetic circuits implementing a desired behaviour can be very large. Moreover, the behaviour of biological systems emerges from a myriad of interactions between biological molecules. Therefore, in order to engineer biologically feasible genetic circuits, domain expertise in different areas of the cell biology of the engineered organisms is essential.

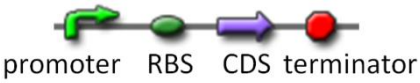


Figure 1.1: A basic genetic circuit composed of promoter, RBS, CDS and terminator parts, where the CDS that encodes for a protein is constitutively expressed.

To date, biological systems have usually been designed manually by a domain

expert who knows the system to be engineered in detail [14]. Most applications are small, and the techniques used to construct these circuits are still *ad hoc* [15]. However, as genetic circuits become larger and more complex, the number of parts required increases, making the manual design of these systems much more difficult. As synthetic biology moves towards tackling real-world problems rather than proof of principle demonstration, large-scale approaches to genetic circuit design and testing, which are beyond the capabilities of individual human experts, are required.

One of the ambitions of synthetic biologists is to apply engineering principles to the construction of large-scale synthetic genetic circuits [16-18]. This goal includes the engineering of entire pathways and eventually the creation of synthetic genomes. A practical example in 2006 was the engineering of the artemisinic acid production pathway using *Saccharomyces cerevisiae* [4]. Artemisinic acid can be transformed into artemisinin, which is a highly effective drug against malaria. This pathway involves 11 genes and converts acetyl-CoA into artemisinic acid [19] (Figure 1.2). However, despite the relatively small size of this pathway, its engineering took 150 person-years of research time and the investment of more than $25 million [20]. In order to construct even larger genetic circuits in a cost- and time-effective manner, computational design and simulation techniques will be essential [20-22]. Technologies such as flow cytometry and microfluidics enable researchers to quantify cellular dynamics, such as the levels of gene expression and the details of interactions between molecules [23-25]. This information allows quantitative models of systems to be built prior to implementation and experimental testing [26]. In addition, DNA synthesis has become a convenient way to create DNA sequences [27]. Therefore, the design of DNA sequences is becoming a bottleneck in the production of new synthetic genetic circuits [20].

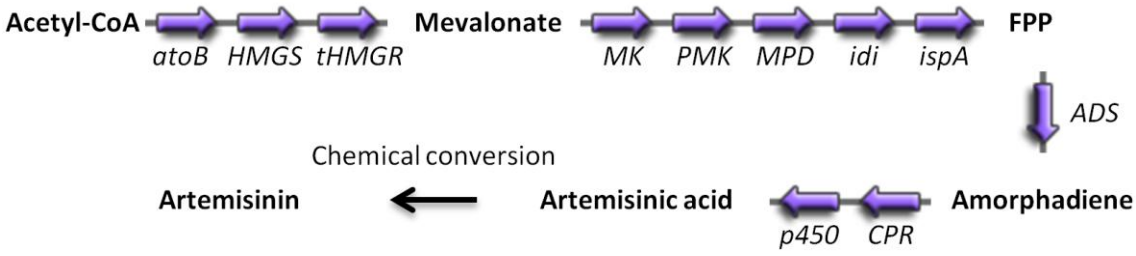


Figure 1.2: Genes involved in the production of artemisinic acid. Acetyl-CoA is initially converted to mevalonate which is then converted to FPP using genes in another operon. The *ADS* gene encodes for a synthase that can convert FPP into amorphadiene, which is then oxidised to produce artemisinic acid. Using a series of chemical conversions, artemisinic acid can be converted into artemisinin. Adapted from [19].

Biological parts studied and characterised in isolation can be combined using a

bottom-up approach in order to construct complex genetic devices and systems [28]. Using this approach well-characterised off-the-shelf parts can be combined to form more complex systems [29]. For small designs, attempting every possible combination of available parts, in order to achieve a desired behaviour may be manageable. However, the number of possible solutions increases exponentially with the number of available parts and the consequent solution space for large circuits may be unfeasibly large for brute-force exploration. Using a library of 36 promoters and genes, the number of circuits that could include up to three genes exceeds 500,000 [30]. Furthermore, not all solutions are biologically plausible. One way to identify such solutions is the use of existing information about biological parts and their relationships.

Top-down design approaches are emerging in order to complement the bottom-up assembly of biological parts [31, 32]. In contrast to the bottom-up approach, the top-down approach involves the decomposing of a system into basic building blocks [31]. The desired phenotype of a biological system is initially specified using a high level definition [15, 33]. For example, a negative autoregulatory circuit can be constructed using a promoter, an RBS and a CDS that encodes for a transcription factor (TF) which deactivates the expression of the promoter. Then, a more detailed specification of the system [32] is created. This finer specification includes the identification of biological parts that meet the desired functionality and constraints [31]. For the above example, CDS parts that encode for TFs and promoter parts that are repressed by corresponding TFs can be identified. Extensive information about parts and their interactions are already available in bioinformatics databases, and can thus be used to guide the computational design of large synthetic biological systems.

The 2012 issue of *Nucleic Acid Research* reports 1380 active bioinformatics databases [34]. These databases provide a variety of information ranging from details of DNA and protein sequences to molecular interactions and biological pathways for many organisms. Identifying relevant sources of information and integrating data about biological parts would be valuable to inform circuit design. However, data integration for synthetic biology has been underutilised. Ontologies [35] can be used to provide explicitly-defined terms for objects from a domain of interest for machine access, and hence can also be used to represent information about biological parts and the relationships between them.

In addition to computational access to information about biological parts, modular parts models can facilitate the construction of models of predictable, large-scale biological systems. Therefore, publicly-accessible repositories of machine-accessible

models of a large number of parts are needed. In addition, simulateble models can act as blueprints for physical genetic circuits, and therefore must be converted into synthesisable DNA sequences in order the test these circuits experimentally [36]. However, the automation of such a conversion has not been accomplished before.

A limited number of model organisms are in wide use in order to test constructed circuits experimentally. Detailed knowledge about the cell biology of many organisms that can be used for potential applications is still not available. The availability of orthogonal parts that are less likely to cause cross-talk between the engineered circuits and host cells would be useful.

1.2 Computational design and simulation of synthetic genetic circuits

In order to construct reliable biological systems, these systems should be modelled, and their behaviour observed *in silico* prior to physical construction and experimental testing [22]. Most genetic circuits developed to date are small, tractable systems that are constructed with a handful of parts [37, 38]. However, it is hard to predict the behaviour even of a simple biological device [11].

A number of CAD tools have been developed that allow genetic circuits to be designed manually [11, 39-47]. These tools usually work with parts from the MIT Registry of Standard Biological Parts [32, 33, 48, 49] and use models of these parts to perform simulations. Although the registry includes parts for regulatory elements [50], regulatory interactions between parts are not captured [51]. Therefore, it is difficult to construct models that adequately represent the systems of interest.

The ability to design genetic circuits in a fully automated manner facilitates the identification of genetic circuits by automating the search of the space of possible solutions. Domain specific languages such as GEC [43] and Eugene [21] have been proposed for the definition of biological circuits at a high level of abstraction. These design specifications can then be implemented using physical parts. However, these ‘programming language’ approaches are restricted to the use of fixed network topologies. In comparison, tools that implement stochastic heuristics such as evolutionary algorithms can design circuits with the same desired behaviour, but with different degrees of connectivity and numbers of parts [52].

Both the manual and automated design of biological systems using computational tools can be facilitated by the use of modular, composable models of parts [22]. Modular modelling in synthetic biology has been demonstrated to be valuable in a

number of studies [11, 30, 41]. However, there is not always a one-to-one mapping between computational parts models and physical parts [30], and the composition of models is application specific. In addition, there are few repositories of modular models. The construction of these models requires the integration of data from several sources such as experimental results, the academic literature and bioinformatics databases [53].

1.3 Integration of data and tools for synthetic biology

Although there is a large amount of biological data available to inform synthetic biology, it is stored in a myriad of biological databases. These databases are often disconnected from each other, and include different types of information. The syntax (or format) of data, and semantics (or underlying meaning) [54] used to represent data, may also be different in different databases. The field of data integration aims to combine heterogeneous databases under a single queryable interface [55] in order to extract the maximum information available from source databases [56]. However, such integrated data must be presented to computational tools in suitable formats.

Data-driven approaches to synthetic biology have recently started to be utilised. The mining of genomes has been proposed in order to identify and categorise existing sequence features for use as basic biological parts [50, 57, 58]. However, sequence-based descriptions of parts alone are not suitable for computational design, since these descriptions do not include information about the potential interactions between parts. In order to analyse a myriad of interactions and how they are linked to sequence features, existing biological knowledge from the literature and public databases should be integrated. The availability of integrated data may facilitate the selection of parts for the rational design of biological systems [49].

Computational approaches in synthetic biology require that information about parts and their interactions must be available in computer-readable formats. Semantic Web languages such as the Resource Description Framework (RDF) [59] have started to make the data on the Internet machine-accessible, and are increasingly being used in the life sciences. Many databases provide their data in RDF as a common data format, which can be queried using standard languages [60]. RDF also facilitates the aggregation of data [61]. Ontologies can also be used to model a domain of interest, providing interoperability across databases and tools [62]. The shared understanding of parts and interactions can therefore be modelled via ontologies and used as a formal specification for information transfer between computer applications.

The importance of these technologies has been recognised by the synthetic biology

community. In order to standardise the exchange of information about parts at the DNA level, a language called Synthetic Biology Open Language (SBOL) has been proposed [63]. Using SBOL, individual parts, and constructs built with these parts, can be computationally represented. SBOL is based on RDF and is backed by an ontology. Several synthetic biology tools, including the MIT Registry have been presented in SBOL format using an RDF triple store [13].

RDF can also be used for the annotation of dynamic models built using formalisms such as CellML [64] or the Systems Biology Markup Language (SBML) [65] [53]. CellML and SBML are widely-accepted systems biology modelling languages that have also been adopted by the synthetic biology community [32]. These XML-based languages provide the syntax for the mathematical representations of biochemical reactions, and can be used to exchange models between computer applications. Such models can be annotated with RDF data to aid both machine and human understanding.

1.4 Motivation for this work

Most genetic circuits are currently designed manually by domain experts, who need to fully understand the system to be engineered [36]. However, in order to design large and complex genetic circuits, computational tools are required [20-22]. An automated design process would include the automation of the production of DNA sequences [14, 66]. However, in order to create biologically feasible designs decisions made during the design process, and the selection of parts should be made in the light of existing biological data [67]. In this thesis, it is demonstrated that novel data integration techniques can be usefully applied to guide the efficient design of biological systems. Here, these techniques are applied in order to facilitate the automation of the design and simulation of biologically plausible genetic circuits in the form of DNA sequences ready for synthesis.

The large-scale engineering of genetic circuits requires a comprehensive understanding of cellular systems. Bioinformatics databases include extensive amounts of information about these systems. However, even for model organisms that are used in synthetic biology, this information is spread throughout different databases. Tools that integrate data from bioinformatics databases to inform synthetic biology are therefore valuable [32].

Therefore, knowledge about many aspects of the cell biology of these organisms must be integrated to give as complete a view as possible. Integrated data can also be used to increase our understanding of non-model organisms, and hence to increase the

number of hosts available for synthetic biology. This approach could be valuable in increasing the number of parts available for synthetic biology [16].

Several computational tools that are relevant to synthetic biology have been developed [22, 32, 50], and various types of novel biological circuit designs have already been described [3, 68-70]. However, the design of large biological systems is currently hindered by the lack of interoperable parts [38, 71]. For example, in order to engineer transcriptional regulatory networks, a small number of well-known TFs are in wide use [38, 69-72]. For large-scale synthetic biology, a wider range of orthogonal TFs that bind to different DNA sequences is needed [49]. The number of parts can be increased by mining sequence features from existing genomes [50, 57, 58]. In addition, biological knowledge about these parts can be integrated and presented to computational tools in order to create complex genetic circuits.

The design space for genetic circuits constructed from biological parts can be very large [21, 30, 73, 74]. However, not all possible solutions are biologically plausible. To make the design of novel biological systems time- and cost-effective, it would be valuable to constrain this design space, for example by reference to existing biological data [75]. The many molecular interactions in a cell can also be analysed for desired and undesired interactions between genetic circuits and host cells [17]. Such an analysis can be based on a detailed map of molecular interactions. Therefore, in addition to sequence-based features, molecular interactions and biological constraints should be catalogued for the rational design of biological systems [76]. A number of possible solutions for achieving a particular target behaviour can be implemented with different numbers and types of biological parts using different network topologies [22, 77-79]. The selection of appropriate parts for these designs can be facilitated by the use of dynamic models of parts [49].

In order to facilitate the use and exchange of models of parts, standard modelling frameworks based on the simple composition of parts are needed [22]. Models of parts should be modular and reusable. These models should also be composable, allowing models of larger biological systems to be constructed from several smaller ones [80]. The process of composition should not be application specific, and models should be represented in standard modelling languages, in order to support the use of, and exchange models between, different design tools. However, mathematical models usually provide the minimum of information required for the simulation of modelled systems - that is, the topology of the circuit and the values of the appropriate parameters [81]. In order to join modular models computationally, mathematical models should be

enriched with computer-readable annotations. For example, modelling entities that represent inputs and outputs should be distinguished from other entities in these models. In addition, models can be annotated with information that allows computers to interpret the DNA sequences and biological reactions that are referred to. When annotated with informative metadata, models of parts can be used to construct larger models that can be converted into DNA sequences encoding the systems being modelled [36].

Although several computational tools have been developed for the design of synthetic biological circuits, these tools generally lack access to a large number of models of physical parts [80]. Quantitative models are based on molecular interactions and biochemical parameters. Therefore, these models must be constructed using existing biological data from bioinformatics databases and the available literature [53]. Large-scale synthetic biology requires the availability of large numbers of parts and data about their interactions. As a result, the manual construction of large models is not feasible, and the process should be automated [14]. In order to facilitate this automation, a wealth of biological knowledge relevant to the modelling of biological systems should be integrated on a large scale [82] and represented in computer-readable formats; for example, using standard knowledge representation languages.

There is a need for the development of data standards for synthetic biology [2]. SBOL aims to represent DNA sequences and their sequence annotations in a standard format [63]. Large amounts of data about genes, gene products and their interactions should also be presented as specifications for computational design tools. Ontologies are widely used in the life sciences and are suitable for modelling biological information as formal specifications. Ontologies can also be helpful for the automated identification of parts in the computational design of large-scale biological systems.

1.5 Contribution of this thesis

In this work, data integration was used to facilitate the computational design of gene regulatory networks for a model organism, *Bacillus subtilis*. Data from a number of sources were initially integrated in the form of a semantically-enriched biological network in order to facilitate computational access to different aspects of the cell biology of *B. subtilis*. The ontological modelling of this network makes data accessible to a wide range of off-the-shelf tools. Using this ontology, information about biological parts and their relationships can be used to identify suitable parts in order to constrain the design space of possible solutions using automated design approaches.

This machine-accessible information about biological parts was used to construct

modular models of biological parts. Initially, a mapping between physical genetic parts and their mathematical models was defined. These models have standard inputs and outputs which are annotated with machine readable information. Therefore, these models are computationally composable and can be used to automate the construction of simulatable models of large-scale biological systems. A computationally-accessible repository of these modular models in standard formats was constructed, and is publicly accessible[1]. These modular models were also annotated with the types and DNA sequences of biological parts in order to automate the conversion of the constructed models into DNA sequences. An algorithm to perform such a model-to-sequence conversion was developed and implemented as a tool, MoSeC [36].

In order to extend the construction of predictable gene regulatory networks to non-model, but closely related, *Bacillus* species, a list of orthogonal TFs that should not interfere with the existing transcriptional machinery of these organisms was identified. Using the information about TFs and their binding sequences from *B. subtilis*, genome-scale gene regulatory networks of 13 *Bacillus* and two *Geobacillus* organisms were constructed. Homology comparisons between TFs from *B. subtilis* and each of these organisms allowed the identification of orthogonal TFs.

We concluded that the use of data integration and the automation of the design and simulation of synthetic genetic circuits can significantly extend the ability of synthetic biology to address large-scale, real-world problems. This work lays the foundation for future investigations into the role of automated data integration, circuit design, simulation and computational reasoning in the large-scale application of synthetic biology.

1.6 Aims and objectives

The main aim of this project is to research and implement novel approaches to the use of data integration in order to constrain the design space of biological systems. A second aim is to implement data integration techniques in order to inform computationally generated biological models which can be used in the design of synthetic genetic circuits.

To achieve these aims, the following objectives were defined:

1. Research and establish a biological knowledge base for *Bacillus subtilis* in the form of a biological network.

[1] http://atgc-eidos.appspot.com

2. Research the use of knowledge representation languages in order to standardise machine access to data for synthetic biology.
3. Research and develop data integration techniques for the annotation of computational models for synthetic biology.
4. Develop a mapping between physical biological parts and their computational models.
5. Create a repository of computationally-accessible, modular models of parts.
6. Apply the outcomes of objectives 1 through 5 to the development of a system for the automated design of regulatory circuits in *B. subtilis.*
7. Extend this research to the identification of orthogonal parts that can be used for the design of regulatory circuits across different *Bacillus* species.

1.7 Thesis structure

This thesis is divided into the following chapters:

- Chapter 2 provides background information and a literature review of synthetic biology, data integration, Semantic Web technologies and modelling of biological systems.
- Chapter 3 describes the application of techniques for the development of a workflow-based data integration pipeline in order to capture the cell biology of *B. subtilis*. Relevant sources of knowledge are identified and integrated, and the pipeline is used to produce a semantically annotated biological network.
- Chapter 4 describes the development of an ontology of parts and their relationships for *B. subtilis*, based upon the network described in Chapter 3. This ontology facilitates the automated querying and reasoning of parts.
- Chapter 5 describes the development and implementation of a mapping between physical biological parts and modular, composable mathematical models of the parts. This mapping permits the assembly of parts into dynamic, simulatable models of biological systems. In addition, this chapter addresses the role of model annotation in the automation of the construction of larger models. A computationally accessible repository of models of parts which can be useful for CAD and automation tools in order to construct novel genetic circuits is presented.
- Chapter 6 describes the need for automation of the conversion of dynamic models into synthesisable DNA sequence specifications. A conversion algorithm is developed and implemented as a tool that generates DNA sequences in standard formats.

- Chapter 7 describes the extension of this project to the construction of genome-wide gene regulatory networks for non-model but closely related *Bacillus* species, using *B. subtilis* as the model organism. A comparative genomics approach is then used to extract transcription factors and their binding sequences as orthogonal parts.
- Chapter 8 reviews the result of this work in the broader context of synthetic biology and describes possible future work.

Chapter 2. Background

2.1 Synthetic biology

Synthetic biology has been an active area of research since the late 1990s, when researchers started to investigate the application of engineering principles to biology. Hartwell and co-workers proposed the development of functional biological modules, drawing analogies from engineering disciplines [83]. By 2005, Endy suggested that engineering principles such as standardisation, decoupling and abstraction would be important for the engineering of biological systems [10].

Synthetic biology has emerged as a distinct field which applies engineering approaches to molecular biology [84]. In classical genetic engineering, molecular biologists focus upon adding or manipulating one or a few genes in an existing biological system [85, 86]. In contrast, synthetic biology places an emphasis on using engineering principles such as modularity, standardisation, and predictive modelling to enable mass production of organisms with novel, desirable characteristics [37].

2.1.1 What is synthetic biology?

As a multidisciplinary field, there are many definitions of synthetic biology in the literature [26, 27]. In most definitions, the emphasis is on the creation of novel, complex biological designs in a reliable and predictable fashion. In a *Nature Biotechnology* article in 2009 [1], experts from different backgrounds varyingly define synthetic biology as involving:

"*the development and application of engineering principles to make the design and construction of complex synthetic biological systems easier and more reliable*" (Christina Smolke).

"*the application of engineering principles towards the construction of novel biological systems*" (Wendell Lim).

"*an engineering discipline for building novel and sophisticated living systems*" (Ron Weiss).

"*the creation of families of genetic 'parts' that behave reliably in designated hosts and have no undesigned interactions*" (Adam Arkin).

2.1.2 Engineering biological systems using biological parts

Genetic circuits can be designed by drawing analogies from engineering disciplines

such as electrical engineering [27]. Electronic systems can be conceptualised with abstract hierarchical layers representing the constituents of these systems. Layers are built on top of each other, with each layer providing a greater level of abstraction than the one below. The underlying elements of these electronic circuits consist of relatively simple components such as diodes and transistors [72]. These elements can be used in combination to create more complex components such as logic gates, which in turn can be combined to form VLSI components such as RAM or CPUs. Finally, abstraction over simple discrete devices is possible through the use of computer networking [87].

The modules that make up a biological system can be considered as analogous to modules that comprise the circuits of an electrical device [83]. There are biological equivalents to logic gates, latches and oscillators which operate using biological molecules [72], such as DNA, RNA and proteins [88] (Figure 2.1). Biochemical reactions, pathways, cells and populations form the top layers in this hierarchy [37].

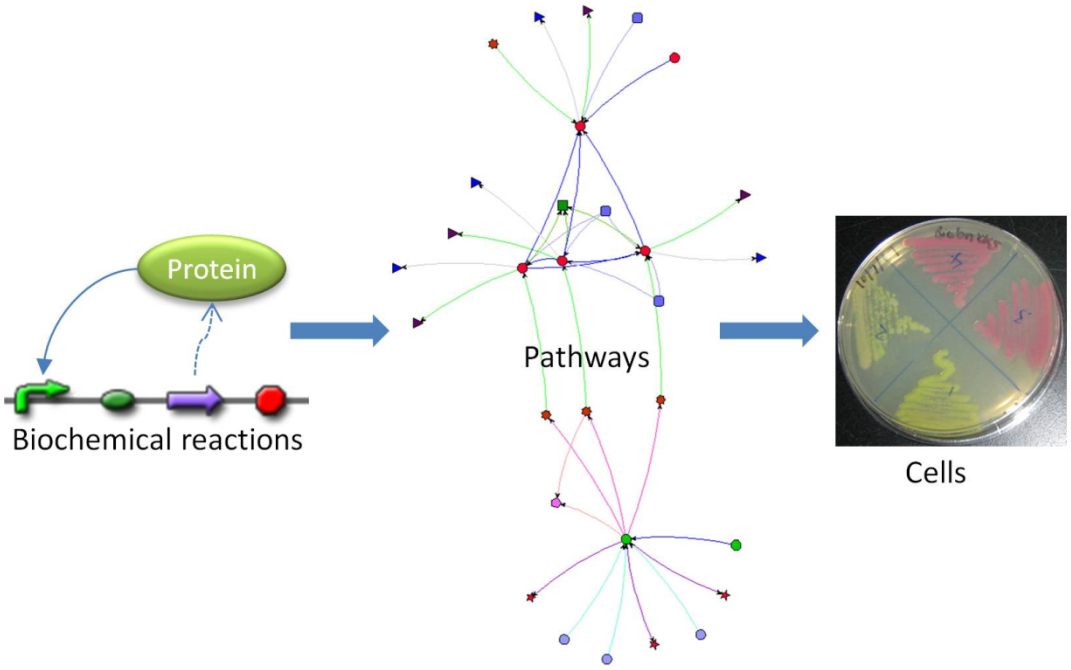


Figure 2.1: A simplified hierarchy for the design of biological circuits. Biological molecules such as DNA and proteins are the underlying elements of genetic circuits. At higher levels are biochemical reactions, pathways and cells.

The modular nature of biological systems means that synthetic biologists can use genetic elements such as promoters, ribosome binding sites (RBSs), coding sequences (CDSs) and regulatory regions as biological parts to build complex systems [11]. Canton and co-workers defined a standard biological part as "a genetically encoded object that performs a biological function and that has been engineered to meet

specified design or performance requirements" [89]. An example of a biological part is the green fluorescent protein (*gfp*) CDS, which can be used on its own as a reporter protein, or can be fused with other genes in order to encode fluorescent fusion proteins [24]. Most proteins have unique domains that can bind to specific biological molecules. [90]. For example, sensory proteins have domains that allow them to recognize a stimulus [91]. The DNA sequences that encode these proteins can be used as modular parts to construct genetic circuits.

The engineering of complex and predictable biological systems requires biological parts with known behaviour, and designs that use these parts interchangeably [37, 90]. In order to standardise the construction of genetic circuits, the BioBricks approach has been proposed [89]. Each BioBrick is a standard biological part that includes flanking DNA sequences which facilitate the combination of BioBricks in a standardised and relatively straightforward manner. The MIT Registry of Standard Biological Parts [12] is a catalogue of BioBricks, and is currently the largest repository of parts for the synthetic biology community [13]. However, compared to actual genomic features [92] the number of parts in the Registry is small. Most parts in the catalogue are for *Escherichia coli*. Another initiative, BIOFAB [93], produces data sheets [89] for a range of biological parts in order to construct predictable genetic circuits.

Standard biological parts can be used to construct complex biological devices and systems. A number of synthetic biology reviews have been published over the last decade [94] which summarise applications and future trends [16, 27, 37, 38, 72, 84, 90, 95]. Applications are currently usually small, involving only a handful of genes. However, even these applications can contribute to libraries that can be used to build larger systems [37]. There are many examples of synthetic genetic circuits including: a genetic circuit that counts [96], the production of the antimalarial drug precursor artemisinic acid [4], bacterial computers [97, 98], a genetic circuit that plays tic-tac-toe through the use of logic gates [99], a heritable genetic memory [100], pattern formation and cellular automata [101], an artificial enzymatic pathway to produce high-yield hydrogen from starch and water [102], the engineering of bacteria in order to seek and destroy the herbicide atrazine [3], and the use of bacteria to repair cracks in concrete [103]. These circuits involved are designed to interact with user-specified signals such as light [104], temperature [105] and metabolites [3]. Using genetic circuits mimicking logic gates, switches, oscillators, feed-forward loops (FFLs) and sensors, inputs are converted to outputs such as the expression of reporter proteins, electrochemical outputs and the production of odours [16, 38, 58] (Figure 2.2). The outputs are controlled by

engineering individual parts at the transcriptional, translational and post-translational levels [27]. Because of their ease of use for constructing a genetic circuit, transcriptional- and translational-level engineering has been the focus of most research to date [27, 37, 58, 90].

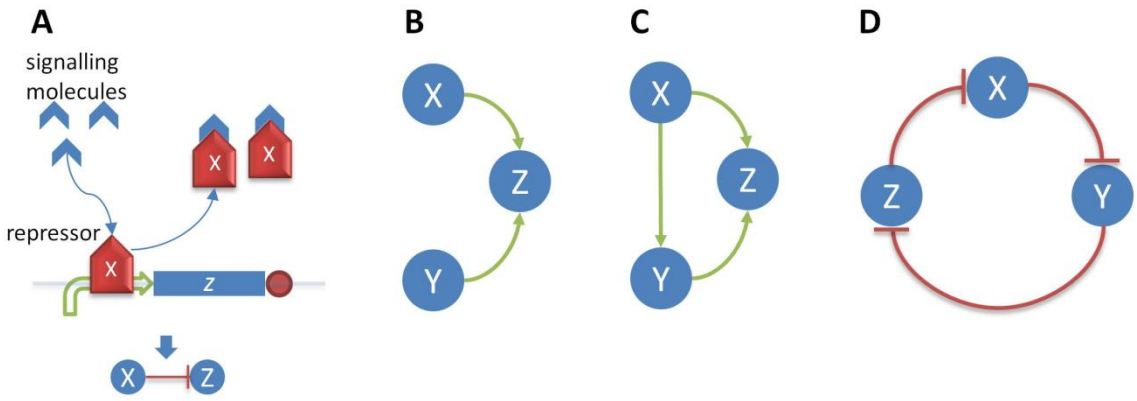


Figure 2.2: Examples of simple genetic circuits. Green arrows represent activation and ⊣ symbols represent repression relationships. **A.** A biological switch acting as an inverter. The *z* gene is not active, and the system's output is OFF under normal conditions due to the transcriptional repression of its promoter by the X transcriptional repressor. Repressors are released via the binding of signalling molecules, switching the system to the ON state. **B.** X and Y are transcriptional activators that are both needed in order to activate the expression of Z. Therefore, the system acts as an AND gate. **C.** This system is an FFL in which X and Y are both needed for the expression of Z, and X also activates the expression of Y. **D.** X, Y and Z transcriptionally repress each other in a cyclic feedback loop. Expression of Z can show oscillatory behaviour.

2.1.2.1 Transcriptional-level engineering

Components of the transcription machinery, such as promoters, RNA polymerases, and regulatory elements can potentially be engineered in order to create useful applications [72]. Promoters with well-characterised transcription rates are important for predicting the level of transcription in a genetic circuit [16]. Transcription rates in turn affect the cellular concentration of proteins via mRNA production, although mRNA levels are not necessarily directly correlated with protein levels [27, 58]. The binding affinity of promoters to their cognate sigma factors is known as their strength, and also affects the rate of mRNA synthesis. Several promoter libraries with different strengths have already been constructed [70, 71, 106]. These promoters and their regulating transcription factors (TFs) can be used to create biological devices such as feedback loops, FFLs, logic gates and switches [37].

There are many existing examples of transcriptional-level engineering. One of the first synthetic genetic circuits, the repressilator, was built using the LacI, CI and TetR proteins transcriptionally repressing one another in a cyclic negative-feedback loop,

resulting in oscillatory behaviour [107] (Figure 2.3). TetR also controlled the expression of a reporter gene which was used to experimentally verify the oscillations. Bistable switches have also been constructed by using two proteins that repress each other transcriptionally [71, 108]. When the concentration of one of these proteins changes, the switch may change its state by repressing the expression of the other protein. These switches require concentrations of TFs to be above certain threshold values for such state changes, and therefore are robust to a low level of transcriptional noise [37]. The decisions made by these switches can persist for generations. In addition, logic gates such as AND and OR gates have been implemented by using combinations of promoters [109]. Such promoters contain more than one operator site in order to induce or repress the activity of promoters [70].

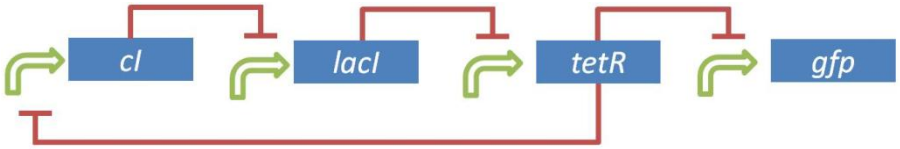


Figure 2.3: The repressilator circuit in which CI represses the expression from the *lacI* gene, whose gene product represses the *tetR* gene [107]. Finally, TetR inhibits both *cI* and *gfp* genes. GFP expression is experimentally measured in order to monitor the state of the circuit.

2.1.2.2 Translation-level engineering

The levels of gene products can also be programmed via the translation process, in which mRNAs are used to produce proteins. Components of the translational machinery such as RBSs, ribosomes and RNAs can therefore also be used as biological parts.

RBSs are the regulatory sequences for the initiation of protein translation and engineered RBSs can be used to control cellular protein levels [110]. Other approaches to translation-level engineering include the use of RNA molecules that are sensitive to external signals [105], riboswitches and small RNAs that provide a fine-tuning mechanism to control the expression of proteins [3, 111], and orthogonal ribosome-mRNA pairs that function independently of the existing cellular machinery [112]. In addition, codons can be optimised to utilise more abundant tRNAs in order to build more proteins or to control the secondary structure of mRNAs [16].

Transcription or translation can also be modulated using spacer sequences. Promoters are context dependent in that, when connected to specific other biological parts such as RBSs and regulatory elements, even a strong promoter may function as a weak promoter. To remove this promoter context dependency, Davis and co-workers

created a promoter library with spacing sequences, which they termed insulations, before and after the core promoter regions [106]. These promoters show less variability in the expression of genes and, therefore, have increased predictability. In addition, spacer sequences between promoters and RBSs can affect the translational efficiency of mRNAs due to changes to RNA secondary structure [113]. Tools that can predict translation rates, such as RBSDesigner [114], have been developed, taking into consideration the nucleotide sequences of these spacer sequences. Spacer sequences perform similar functions to the shims that are used to join electrical or mechanical components [36]. Spacer sequences are therefore referred to as shims in the remainder of this thesis.

2.1.2.3 Post-translational level engineering

The post-translational modification of gene products can also be used to engineer genetic circuits. For example, some TFs need to be activated via phosphorylation in order to participate in gene activation or repression. Therefore, cells can be programmed to change their states based on these modifications. External stimuli such as light and metabolites can also be used to trigger these modifications in order to develop novel applications, including biosensors.

One of the ultimate aims of synthetic biology is to engineer signalling systems by manipulating spatiotemporal intracellular and intercellular interactions [90]. The engineering of cells to sense and respond to signalling molecules has potential benefits in many areas, such as medicine and biotechnology [68]. In signalling systems, environmental inputs detected by cellular receptors are turned into outputs such as gene expression and cell growth.

Two-component systems (TCSs) are a type of signalling system in prokaryotes. These systems allow cells to monitor their environment and respond by converting external signals into gene expression. TCSs comprise a sensor kinase protein that senses a signal, and a response regulator protein that is activated by the kinase to control some biological functionality (Figure 2.4). TCSs can be used as modular biological parts to construct synthetic genetic circuits [91]. These systems may work in functional isolation from one another, and it is possible to connect different inputs to discrete outputs. One of the first examples of the use of TCSs for synthetic biology was a light-activated genetic circuit that projects a pattern of light onto an agar plate using the EnvZ-OmpR kinase and response regulator pair [104]. In this circuit, the OmpR response regulator was used to activate the expression of LacZ which produced a black compound. The

EnvZ kinase was fused to a red light receptor in order to sense light, which disrupted the autophosphorylation of the response regulator, turning off the gene expression. Therefore, when a pattern of light was projected through a mask onto the bacteria, they were not able to produce the black compound in the areas that are not shielded from light, forming a two dimensional image.

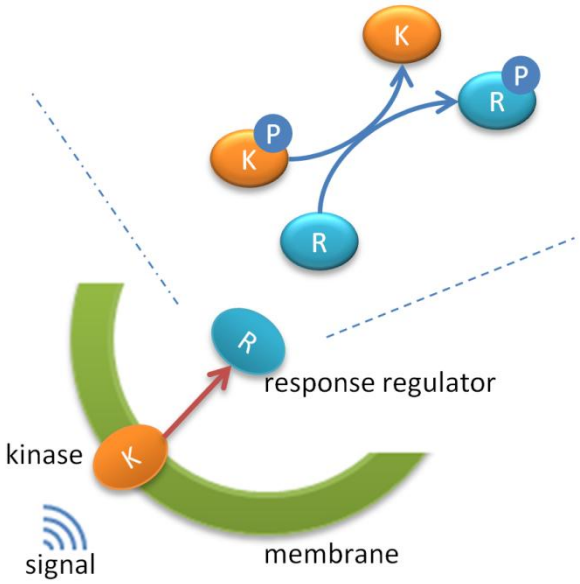


Figure 2.4: An example of a two-component system. Upon sensing the signal, the kinase protein phosphorylates the response regulator, which is commonly a transcription factor. The phosphorylated response regulator becomes active and can therefore regulate the expression of corresponding genes.

2.1.3 Genetic circuit design

Biological parts can be combined using either bottom-up or top-down approaches to create biological devices and systems. The Oxford English Dictionary defines bottom-up as "proceeding from the bottom or beginning of a hierarchy or process upwards". In the context of synthetic biology, the underlying elements of genetic circuits are basic biological parts such as promoters and CDSs. These parts are the components of biological devices that can be joined together to create more complex biological systems. A top-down approach involves "the breaking up of a high-level overview of a system into a number of modules, and then further reduction of the modules into sub-modules until all of the components of the system are specified at an elemental level" [36]. The definition of standard biological parts given above allows the use of abstractions that hide mechanistic details such as the physics of the interactions between biological molecules. Abstraction facilitates the construction of biological systems from modular biological components in a bottom-up fashion [37]. However, constructing

predictable genetic circuits requires a deeper knowledge of the interacting biological components. Therefore, top-down approaches that unravel the interactions between a myriad of components can also be used to understand the behaviour of biological systems built with individual parts [20].

Biological parts are defined as black boxes with standard inputs and outputs. However, unlike electrical systems, the input of a biological system may be different from its output, and the output may then need to be a valid input for the next component in a network [58]. Therefore, a set of standard biological signal carriers have been defined by the synthetic biology community in order to standardise the interfacing of basic biological parts. Generally recognised metrics for signal carriers in synthetic biology include polymerases per second (PoPS) as a measure of transcription rate, and ribosomes per second (RiPS) as a measure of translation rate [10, 11, 41, 58, 89, 115, 116]. PoPS refers to the number of transcription initiations and nonreturn elongations from a promoter. Similarly, RiPS reflect the number of ribosomes passing through a given position on an mRNA molecule. These biological signals create a signal flow between biological parts such as promoters, RBSs, CDSs and transcriptional terminators [11, 41].

Although engineering approaches such as using standard and modular biological parts are useful in order to engineer living cells, the design of genetic circuits using biological parts imposes a number of challenges. For example, issues may arise from the use of particular biological parts in genetic circuit designs. In addition, the design spaces for genetic circuits can be very large and identifying functional solutions can be challenging. These issues are discussed below.

2.1.3.1 Identifying biological parts

Several synthetic biology-oriented computer-aided design (CAD) tools [22, 32, 50] and domain specific languages (DSLs) have been developed, including GEC [43], Eugene [21] and Proto [117]. However, the availability of interoperable and well-characterised parts still remains a major limiting factor in the design of complex biological systems [38, 71]. As a result, the design step is becoming something of a bottleneck in synthetic biology [20]. The effort to answer the question ‘What DNA should we synthesise?’ [118] is hampered by the lack of biological parts. For example, a limited number of well-known TFs, such as LacI, TetR, AraC and LuxR, are in wide use [38, 69-72]. As designs become more complex, the number of parts required also increases. For example, a circuit that can detect multiple inputs and report with multiple outputs

includes more genes, proteins, TFs and other regulatory elements than a simple biosensor [76].

Other aspects of biological systems such as the cellular context, should also be taken into account when designing genetic circuits [20]. Promoters and their regulatory systems should be designed taking into account the circumstances under which the circuits will perform. For example, in *B. subtilis* SigmaA is known as the housekeeping sigma factor, and is active under normal conditions [119]. The alternative sigma factor, SigmaD, is only expressed in motile cells [120]. The chromosomal location of CDSs may also affect the interactions of their encoded molecules, due to differences in diffusion rates and DNA folding [121]. For a TCS to perform, both of its proteins need to be present at the same time in the cell. As a result, they are often located in the same operon and expressed from the same transcript [122]. Such design patterns from nature may be used to inform the structure of a synthetic circuit [30, 90].

In electrical and mechanical circuits, in which circuit architectures do not change in space and time, multiple instances of a component such as a transistor may be used without interference [28]. However, using multiple instances of a biological component such as a promoter may cause cross-talk between subcircuits. Furthermore, adding multiple instances of the same biological part may drain output from other components [79] and create undesired behaviours [115]. Therefore, for the large-scale engineering of genetic circuits, a wide range of parts should be available. However, the design spaces of genetic circuits grow exponentially due to the large numbers of parts required for complex designs.

2.1.3.2 Identifying possible genetic circuit solutions

Computational tools have been developed to explore the solution spaces for the design of biological circuits. Even a biological part with 10 bp has 4^{10} different possible sequences. It is not possible to evaluate every possible combination. Therefore, these tools search the large design spaces for genetic circuits using available data about biological parts. Each nucleotide sequence of a genetic feature can then be mapped to a desired behaviour with a fitness function, creating a fitness landscape for all solutions. The number of parts and their interactions used in a genetic circuit may change the fitness landscape, and hence may affect the performance of such a search process [31].

DSLs and evolutionary algorithm implementations [14] aim to automate the identification of possible solutions for genetic circuit designs. DSLs for biological systems commonly represent a desired genetic circuit using a high-level definition,

without initially defining the parts needed to implement a circuit [123]. For example, without specifying any constraints, a simple '`prom;rbs;pcr;ter`' GEC [43] code represents a device that includes any promoter, RBS, CDS and terminator parts. Therefore, any part from the GEC repository can be substituted in the design, creating a huge solution space even for a relatively simple design. In order to constrain the solution space, the language provides explicit terms, for example specifying that a promoter is negatively regulated by one of the TFs used in the same design. Similarly, the Eugene language [21] focuses on the exploration of combinatorial design space. Eugene allows the permutation of all possible combinations with which a device can be created. This design space is then constrained with rule-based definitions using existing biological knowledge to include or exclude certain parts, and thus to reduce the number of solutions.

The selection of a design algorithm may also increase the solution space. For example, evolutionary algorithm approaches use fitness functions, such as input-output mappings, to identify a circuit with a desired behaviour [14]. In addition to using different parts and parameter values, genetic circuits with different topologies can be created, resulting in diverse solutions.

In order to experimentally test the identified solutions for synthetic genetic circuits, DNA constructs that represent these circuits should be placed in an environment where biological reactions such as transcription and translation can take place. This machinery is usually provided by living cells such as bacteria.

2.1.4 *Bacillus subtilis* as a chassis for synthetic biology

Novel genetic circuits can be placed into bacterial cells in order to produce non-wild type behaviour. The choice of a host cell, also known, by analogy with engineering, as a chassis, [5] is very important, since new circuitry may interact in unanticipated ways with other elements in the cell [115]. To minimise this risk, synthetic biologists tend to use well-known model organisms, such as *Escherichia coli* or *Bacillus subtilis,* or genetically minimised cells, as their chassis. Genetically minimised cells are developed in order to minimise undesired interactions between host cells and the synthetic genetic circuits [38]. In this process, non-essential genes are left out, producing reduced genomes, which have been constructed for different organisms including *E. coli* and *Mycoplasma genitalium* (reviewed in [27]).

2.1.4.1 *Bacillus subtilis*

B. subtilis is a Gram-positive, non-pathogenic, model organism and is generally

considered to be safe [124]. The organism inhabits the soil and may also develop symbiotic relationships with plants [125]. *B. subtilis* is well studied, and its ease of use for genetic manipulation makes it ideal for laboratory studies. This bacterium is widely used in the biotechnology industry to produce vitamins and enzymes [126-128]. *B. subtilis*'s close evolutionary relationship to some pathogens, such as *Bacillus anthracis*, also makes it an ideal choice for the investigation of the genetic bases of a range of diseases [129].

B. subtilis has approximately 4,200 genes [124], of which 271 have been identified, using gene knockouts, as essential [129]. Half of the organism's essential genes have been shown to be involved in information processing [129]. Important genes such as the central controllers of metabolic pathways and various regulators have been identified in a number of studies [130-132]. With an extensive amount of information about *B. subtilis* cell biology already available, this organism is ideal for the development of applications for synthetic biology [133]. Additionally, there is an ongoing minimal genome project underway with the aim of using *B. subtilis* as a chassis [134].

2.1.4.2 *B. subtilis* as a chassis

B. subtilis has many advantages as a chassis for synthetic biology. The organism has a number of natural phenotypes which can be controlled by programmed biological sensors [72, 135]. Cells can be motile; produce chains of non-motile cells and long lasting sessile spores; form biofilms; become genetically competent; and secrete toxins and enzymes to cannibalise their neighbours [131]. The different behaviours of this bacterium may be utilised by synthetic biologists to create new biological applications. For example, an arsenic biosensor was deployed in *B. subtilis* spores which were heated to germinate the spores and therefore to activate the sensor [136]. Engineering biofilm formation may improve the protection of plants against harmful bacteria [133].

Synthetic DNA can exist as plasmids in *B. subtilis* cells, or can be integrated into the chromosome. The latter approach provides more stability for synthetic biology applications, since the genetic material exists as a single copy and is reliably transferred to subsequent generations [125]. As a Gram-positive bacterium, *B. subtilis* has a single membrane, and is therefore ideal for applications involving the secretion of proteins into the environment [125], whereas in Gram-negative bacteria additional steps, such as the lysis of cells, may be required for the secretion of proteins [137].

B. subtilis cells communicate using peptides as signals [138]. This communication system is known as quorum sensing and is used to control the activity of cell

populations [5]. Peptide-based communication is not dependent on specific inducer molecules, and therefore can be used to extend the range of signalling systems available for synthetic biology [139].

The rational engineering of genetic circuits using *B. subtilis* as a chassis requires extensive amounts of knowledge about this organism. A wealth of knowledge about genes, gene products, expression of genes and gene regulatory networks for this organism is already publicly available. However, this knowledge is spread throughout the literature and diverse bioinformatics databases. In order to engineer this organism, knowledge about different aspects of its cell biology should be integrated and analysed.

2.2 Data integration

With the advent of high-throughput technologies such as DNA microarrays and Next Generation sequencing, large amounts of biological data are currently being produced. Instead of sequencing a single gene, researchers often just sequence whole genomes, a task which can now be achieved at relatively low cost [140]. For example, the initial genome sequencing of *B. subtilis* took ten years to complete, involving many researchers and more than forty institutions in Europe, the US and Japan [141]. A similar genome can now be sequenced in less than a day using a single sequencing machine [142]. As the rate of data acquisition grows, the amount of information yielded from a single experiment also increases [143]. It is now possible to examine the expression of genes on a genome-wide scale in a single experiment, providing data for several thousand mRNAs or proteins. Over the last fifty years, huge amounts of biological data have accumulated, the results of which are stored in a myriad of databases. *Nucleic Acid Research* publishes a report every year listing the biological databases available, and 1380 databases are cited in the 2012 issue [34].

2.2.1 What is data integration?

The analysis of large biological datasets can be automated using computational methods and the integration of existing knowledge [144, 145]. Omics technologies, such as genomics and proteomics, each focus on data of a specific type. A combination of omics datasets can therefore provide a comprehensive view of biological systems [38, 146, 147].

Data integration implementations that combine heterogeneous data sources, and provide a single, unified query interface are particularly useful [55]. The integration of data enables researchers to extract the maximum amount of knowledge from the exponentially growing volume of biological data without having to manually access individual databases [56, 148]. Integrated datasets can be used by both humans and computers to analyse biological systems and to develop hypotheses [146]. Data integration has been used in a variety of applications. Examples include the discovery of new drugs [149], the prediction or improvement of functional interactions [150, 151], the identification of essential genes [150] and the inference of the functions of disease-causing genetic variants [152]. Data integration can also be used to guide the efficient design of biological systems.

2.2.2 Data integration for synthetic biology

2.2.2.1 Genome mining for the identification of biological parts

Significant effort has gone into the creation [12] and characterisation [89, 93] of individual parts that can be used to construct genetic circuits. The BioBricks approach is useful for the construction of complex systems in a bottom-up approach. However, information about possible interactions that may emerge from the use of these BioBricks is not available for machine access, and hence this approach is not suitable for the design of predictable biological systems computationally. Considerable research has been undertaken to create parts such as orthogonal, context-independent ribosome-RBS pairs [112] and sigma factor-promoter pairs [153]. However, the number of parts created using experimental approaches is small, since the process requires researchers to experimentally create and test parts, making it time- and resource-intensive, and thus not cost effective.

Data integration approaches are needed to expand the number of parts in catalogues and automate the process of part identification and characterisation [50, 57, 58]. Computational techniques can also be employed to find potential parts by analysing genome sequences. For example, using the information about the model organism *Corynebacterium glutamicum*, the CoryneRegNet system was able to predict promoters and TF binding sequences in closely related organisms [154]. Moreover, a catalogue of transcriptional terminator sequences for *B. subtilis* was constructed based on predictions [155]. These sequences provide a repository of natural biological parts for synthetic biology [57]. The comparative study of biological processes in different organisms will expand the number of parts available for the design of genetic circuits [16]. Therefore, mining sequences from existing genomes and integrating biological knowledge about these sequences can increase the number of parts available, and inform the design of novel genetic circuits using these parts. In addition, information about the interactions between these parts should also be integrated and made available to computational designs tools in order to construct functional genetic circuits.

2.2.2.2 Constraining the design space of genetic circuits

Due to the size and complexity of biological systems, the potential design space for a genetic circuit can be very large. Even for a simple circuit composed of a small number of parts, thousands of different solutions can be created [30]. In biological systems, dynamic interactions between biological molecules such as proteins and DNA give rise to biological functions [83], and thus to the complexity of these systems [90]. Biological

signals are the outcome of interactions between molecules, such as the activation of a protein, inhibition of a promoter or transportation of a compound. Each interaction causes a change in a flux for these signals. Therefore, in order to design predictable applications, parts used in these designs and their possible interactions should be known in detail [20].

As the complexity of genetic circuits increases, it becomes more important that biological interactions are investigated on a global scale [28]. For example, for a TF to function its binding site must also be available, and to reduce the activity of a protein, its antagonist protein may be included in the design of a genetic circuit. Possible solution spaces increase exponentially in direct proportion to the number of available parts and their interactions.

However, not all of potential solutions may be biologically feasible. For example, Densmore and co-workers constructed seven different versions of a genetic device designed to switch a phage from lysogenic to lytic modes [73]. When these constructs were tested experimentally, only one of them functioned correctly. The malfunctioning of the other versions was due to a lack of knowledge about the parts and unknown biological constraints; the regulation of some of the promoters was not correctly captured in the model, some repressors did not work, and some of the parts were toxic to cells. With the addition of biological constraints, the original *in silico* design space of 45 designs was reduced to four designs all of which worked experimentally.

In another study, a combinatorial approach using the Eugene language was applied to create a cell surface display device formed from three protein domains [21]. The first domain was used to expose various peptides to the extracellular environment and the second domain acted as a spacer to connect the first domain to the third which was anchored to the outer membrane of cells. The compilation of the design specified by the language resulted in 540 possible solutions, which were then reduced to three by the addition of biological constraints. In creating synthetic pathways, Hatzimanikatis and co-workers identified almost 75,000 biochemical routes from a chorismate compound to phenyalanine, and more than 350,000 from chorismate to tyrosine [74]. However, using the information about the sequence of biochemical reactions from chorismate to phenyalanine and chorismate to tyrosine, only 7 and 23 solutions respectively were identified as biologically plausible.

Clearly, the large design space of genetic circuits must be constrained in the light of existing biological data [156]. As the number of available parts grows, combinatorially testing every possible solution becomes neither time nor cost effective. Furthermore, for

a computational design, knowledge about biological constraints must be explicitly available to the design tools. In addition to catalogues of parts, computationally-accessible catalogues that hold information about the interactions and constraints of these parts are needed. However, many of the genes, proteins, RNAs and their interactions are not fully characterised, even for model organisms [157, 158]. It may be valuable if data about these parts, and their interactions and constraints are integrated and made available to CAD tools in order to constrain the initial design of genetic circuits [75]. One approach to the constraining of this design space is the use of dynamic models and computational simulation in order to identify solutions with desired behaviour.

2.2.2.3 Data integration for the modelling of biological systems

Large-scale synthetic biology can be facilitated by the modelling of cell-scale molecular interactions [16]. Cellular systems can be reconstructed by collecting the molecular components that comprise these systems and putting these components together in a biochemically consistent way [118]. The behaviour of these systems can be investigated through modelling.

Models are ideally produced using a large amount of existing biological knowledge [53, 77], and they may integrate transcriptional, translational and post-translational level information about biological parts [16]. For example, in order to predict the expression levels of proteins the interactions between promoters and operators, and the TFs that bind to these sequences, should ideally be modelled. However, transcription is not the only factor affecting protein levels [16, 113, 159]. Protein-protein interactions, protein complex formation and disassociation, and other interactions can be included in models in order to achieve a more complete view of cellular function. The concentration of an active protein, and whether it is a monomer, polymer or complexed with other molecules, phosphorylated or unmodified, may be used as the output for a genetic circuit. The optimal design of pathways also requires an extensive amount of information [4]. To engineer a synthetic pathway, important proteins and enzymes with which the target products interact must be identified [50, 160].

2.2.3 Data integration challenges

A large amount of research has been devoted to navigating between biological databases and integrating these data computationally [161]. Data integration in the life sciences is especially challenging for a number of reasons. The amount of biological data is huge and the data are spread over many databases. Mapping of biological concepts across

databases is difficult due to the different conventions, semantics and syntax used by different databases. There is often no standardised way of accessing these databases.

Data are produced by different research communities in separate subfields [59], with different levels of expertise and interests [147]. Consequently, the scope of information and quality of data in different databases may vary. Data may also overlap [162]. An integrated dataset needs to reflect these variations [163] and provide information about data provenance [163].

The mapping of biological concepts from different databases can be very challenging. By inspection it may be possible to identify two biological concepts with similar names; however, to conduct this mapping computationally, these two concepts need to be explicitly defined as being the same. Similarly, researchers should use the same name for the same entity wherever possible. However, this is often not the case [147]. The name used for a particular biological concept may vary between research communities, and hence also between biological databases [60, 164]. Additionally, similar names may not indicate exactly the same biological concepts. In *E.coli*, for example, the *araC* locus encodes the AraC TF. A similar locus in *B. subtilis* also encodes a transcriptional regulator known as AraC or AraR [165]. However, these two TFs are not orthologues, having different functions and belonging to different TF families. Furthermore, the names of biological concepts may change over time. When a gene from *B. subtilis* was found, the gene name given started with the letter 'y', such as *ytxH*. When 407 of such genes were functionally annotated, new names were assigned and these genes are now referred to with the new names [124]. However, some data sources, such as DBTBS [166] which is manually curated from the literature, may still refer to older names that are no longer widely used.

In order to correctly identify biological concepts and overcome inconsistencies in naming schemas, many databases use globally-known accession numbers. For example, BacilluScope [124] is a recent database which includes the latest accession numbers for *B. subtilis*. However, not all databases use accession numbers [166]. Furthermore, even the accession numbers used vary among different databases [55].

Standards for data storage and sharing are very important in an interdisciplinary field such as synthetic biology [147]. However, standards are difficult to establish [162]. In addition to inconsistencies in syntax or the format of data, the semantics or the underlying meaning [54] used to define biological objects and their relationships may also vary. Different researchers may adopt different definitions of terms. For example, the precise definition of a gene is still debatable [167]. To some researchers, a gene is a

transcriptional unit; to others, it also includes upstream and downstream regulatory elements [147]. Relationships such as *is_a*, which is used for various biological concepts, can also have different meanings [168]. These semantic differences may then be reflected in the databases. For example, pathways are annotated and described differently in the Kyoto Encyclopedia of Genes and Genomes (KEGG) and Reactome databases [60].

Additionally, there is no single standard method of access to the biological databases [162]. Most databases allow their data to be viewed online. Database access may be in the form of wikis [126]. Databases may also have application programming interfaces (APIs), and allow their data to be exported in different formats such as tab-delimited, comma-separated or XML [169-171].

Analysis of the huge volume of available biological data may require large computational resources [163]. It may be necessary to select subsets of data for analysis and visualisation purposes. For example, the STRING [171] database is an integrated data source combining several databases. Version 9 of the database contains information about more than 224 million protein interactions for over 1,100[2] organisms. Integrating data from this database may require the use of subsets of the database. Moreover, much biological data may be hierarchically organised, making the use of standard relational databases suboptimal [172]. Representing integrated data can also be challenging [150]. Data integration in the life sciences is an interdisciplinary area, and users and developers are often from different backgrounds. Therefore, visualisation tools should be provided which take the end users into consideration. Graph-based tools such as Cytoscape [173] and Ondex [174] can be used to efficiently visualise biological data.

2.2.4 Data integration methods

The integration of data from several sources in biology starts with the transformation of the data into a predefined data model. Semantically equivalent data are then matched and integrated in a repository [60]. Different data integration methods exist, such as data warehousing, link integration and view integration.

2.2.4.1 Data warehousing

Data warehousing has been widely used for data integration in the life sciences [148, 162, 175]. An example is YeastMine, a data warehouse that provides access to multiple data types from different sources for the model organism *Saccharomyces cerevisiae*

[2] http://string-db.org/ (accessed 11/09/2012)

[148]. In the data warehousing approach, distributed source databases are brought into a single homogeneous database. Data are extracted from the source databases, translated into a data model and loaded into the data warehouse [55] (Figure 2.5).

Initially, a data model (schema) is developed that can accommodate the data from all of the source databases [147]. Data sources are merged and queries are evaluated using this schema [176]. Wrappers are developed to transform data from the source databases into the unified model. Transformed data are then loaded into the warehouse.

This approach eliminates problems such as the unavailability of sources and slow response times [172]. Locally-stored data are more efficient in terms of analysis and visualisation [164], and cleansing or further curation of data is possible [176]. In addition, data can be validated [164] and the results of analyses can be added to the data warehouse [176].

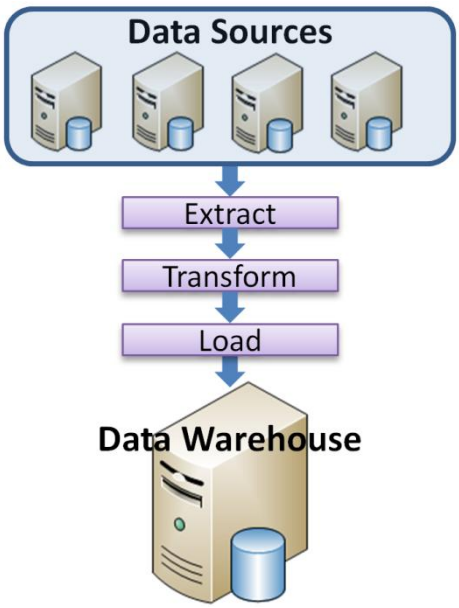


Figure 2.5: Individual data sources are extracted, transformed and loaded into a data warehouse.

Data warehousing provides several challenges. Source databases should be periodically checked, and updates should be synchronized with the information in the warehouse in order to keep it up-to-date [172]. However, source databases may change their modes of data access, as well as their data schemas and types. In these cases, the wrappers need to be refactored in line with any such changes. However, the data models used in warehouses are often difficult to change [162]. Database systems that can be extended flexibly would be useful [177]. In addition, due to the use of different syntaxes and methods used to access data in biological sources, the writing of wrappers can be

challenging.

2.2.4.2 Link integration

In the link integration method, data are linked via the Web [60]. These cross-references can be terms from a controlled vocabulary, or unique accessions [162]. Users follow the links provided by the databases to gain access to data. The integration and interpretation of data is thus left to the user [147]. However, there is no guarantee that specific links will continue to exist, and the linking databases may not be informed of changes from the source databases.

2.2.4.3 View integration

In view integration, data are accessed through portals from several sources [60]. Data are left at the source databases and the emphasis is on query translation rather than data translation [172]. Queries are constructed using a single federated schema and a mediator is used to map this schema to local database schemas [172, 178]. A query processor identifies the source databases from the query in order to create subqueries. The results of these subqueries are then combined and presented to the user [147].

The results of view integration are always generated in real time. Data are therefore always up-to-date, as opposed to the output from data warehousing systems [162]. The maintenance of view integration systems is also easier in comparison to the other methods. However, distributed data approaches rely on querying protocols being agreed between the databases [56]. Furthermore, query performance may not be very efficient [56]. For example, if several databases must be queried, the response time may be slow. In addition, data sources associated with the slowest response times affect the overall performance of the system. One approach which can overcome some of these problems is the use of e-Science technologies such as Web services and workflows. These technologies can facilitate access to and the integration of data distributed in several databases.

e-Science provides an infrastructure for the global and collaborative analysis of large amounts of biological data using computational approaches, and is facilitated by technologies such as Web services and workflows [179, 180]. In these approaches, Web services are used to gain programmatic access to distributed data sources, and workflows define the operation of the large-scale analyses of biological datasets. Workflows also facilitate data integration by enabling the flow of data between different services and data sources [162].

2.2.4.4 Web services

Web services enable interoperability between applications and data sources [181] by providing well-defined inputs and outputs for computational access to data [180]. Although these service-oriented architectures are not themselves data integration mechanisms, they facilitate the integration of data by providing uniform access to data [162]. These services are often the access points of databases to the outside world. Web services are used extensively by life science researchers. Databases such as KEGG [182] and CoryneRegnet [183] have Web service interfaces, enabling the retrieval of data. BioCatalogue[3] stores descriptions of 2287 Web services. These services can be based on a simple object access protocol (SOAP) or representational state transfer (REST).

SOAP-based Web services can be invoked computationally using HTTP, and carry XML-based SOAP messages [184]. In addition to their content, SOAP messages also carry headers that can be used to store additional information, such as user credentials. The details of Web service method calls are available through XML-based Web Services Description Language files [181]. These files are used by client applications to create proxies for remote Web service method calls.

On the other hand, REST-based Web services are simple, and are gaining popularity for data integration [162]. These services can be invoked manually or computationally via HTTP calls such as GET or POST [181]. Using this approach, information about biological parts from the MIT Registry can be accessed using unique URIs for each part[4]. Grunberg also proposed the use of REST services for the integration of data sources and applications for synthetic biology [185]. As the number of parts catalogues increases, REST-based Web services may provide a simple solution for exchanging information about biological parts using standard data formats. However, these services do not offer the advantage of carrying additional headers and they are not capable of being called asynchronously.

2.2.4.5 Workflows

Workflows are used for the enactment of a series of linked processes [162] (Figure 2.6). These pipelines represent complex and compute-intensive tasks in an intuitive manner [186], and facilitate the automated analysis of the exponentially growing amount of data in the life sciences. Scientific workflows can be constructed and shared by researchers.

[3] http://www.biocatalogue.org/ (accessed 27/02/2012)
[4] http://partsregistry.org/Registry_API (accessed 11/09/2012)

For example, the myExperiment[5] Web-based repository is a source of reusable workflows. Tools such as Taverna [187] and Ondex [174] provide the necessary frameworks to execute workflows.

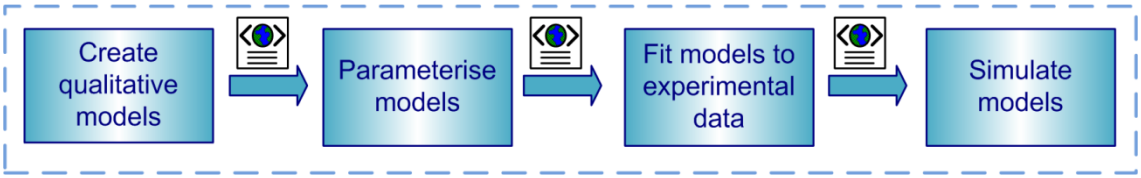


Figure 2.6: An example of a workflow for the modelling of biological systems. Initially, qualitative models are constructed using data from biological databases. These models are then parameterised with reaction kinetics and the resulting models are fitted to the experimental results. Models are then analysed via simulation. These models are expressed using standard modelling languages. Adapted from [186].

In this section, the importance and challenges of data integration, and the need for data integration to inform synthetic biology have been discussed. In addition, data integration methods and some e-Science approaches that facilitate the process of producing integrated datasets have been described. Integrated datasets can be mined to identify, for example, biological parts and constraints. The type of knowledge which can be extracted from an integrated dataset depends, to a large extent, upon the way in which it is represented. There are multiple ways in which biological datasets can be represented, each with its own strengths and weaknesses.

2.2.5 Biological networks

2.2.5.1 What are biological networks?

Biological data can be represented in the form of networks, in which nodes represent biological concepts, such as genes and gene products, and edges represent the relationships between these concepts (Figure 2.7). These qualitative representations are used to capture information about the interactions of biological entities, without including details such as quantitative parameters or reaction stoichiometries [77].

Networks depicting interactions between biological entities are also known as interactomes. Although biochemical parameters are very important to determine how a biological system functions [188], the topology of an interactome may also affect the behaviour of the system [189]. It is known that bacteria can robustly sense signalling molecules such as ligands, and respond to environmental changes. Barkai and Leibler investigated the chemotaxis system of *E.coli* in order to analyse why the tumbling

[5] http://www.myexperiment.org/ (accessed 06/06/2012)

frequency of the bacteria is insensitive to changes in the concentration of ligands or rate constants [190]. The tumbling of the bacteria is controlled by a regulatory protein that is activated by the signalling receptors. The authors concluded that the robustness of the system is partly achieved due to the negative autoregulatory control of the active form of receptors. More examples of networks that show intrinsic robustness are reviewed in [191]. Therefore, the analysis of circuit topologies could be very important in designing reliable biological systems [31]. It has been suggested that recurring motifs can help to create the same response among cells [191].

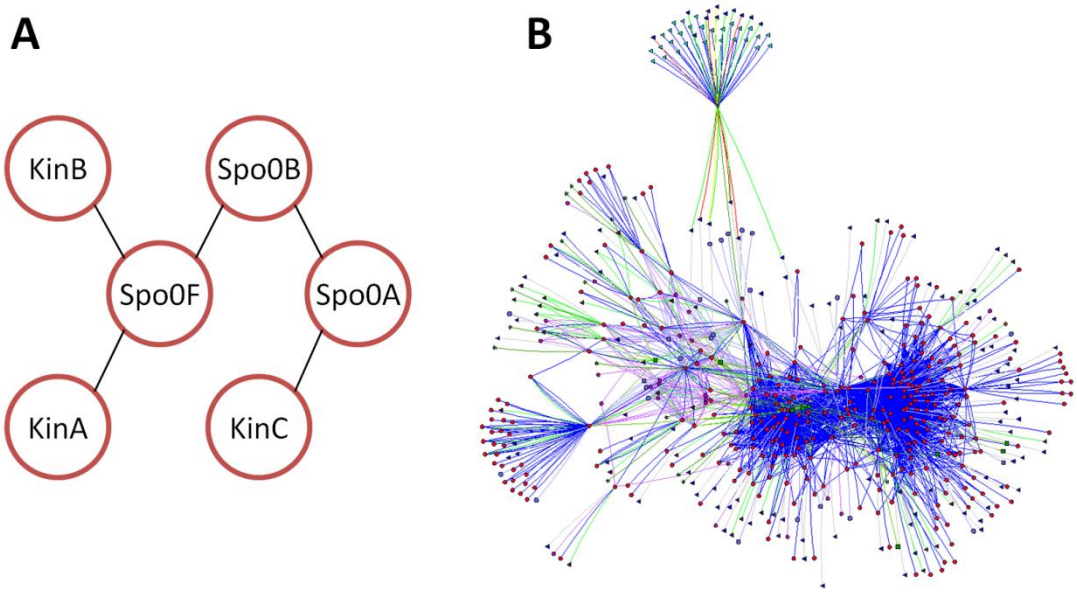


Figure 2.7: Examples of biological networks. Nodes and edges represent biological entities and the relationships between these entities respectively. **A.** A simple network representing the phosphorylation of six sporulation-related proteins from *B. subtilis.* **B.** A more complex network containing information about hundreds of several types of biological entities.

2.2.5.2 Network motifs

Network motifs are subnetworks that are statistically overrepresented in biological networks, as compared to randomised versions of the same biological networks [192]. These randomised networks share the properties of the actual networks, such as the number of nodes and edges [193]. Moreover, each node in a randomised network has the same number of incoming and outgoing edges as the corresponding node in the actual network. However, edges are randomly placed between nodes. Patterns that occur significantly more often in the actual networks than in randomised networks are then identified as network motifs.

The use of recurring circuit elements is shared feature of engineered systems and biological networks. Network motifs appear to perform specific operations such as

signal filtering and autoregulation [194]. Network motifs are therefore an area of interest for synthetic biology [85]. Motifs with known functions, such as negative and positive feedback loops, and FFLs, can be used as modular building blocks for the rational engineering of novel biological networks.

In transcriptional networks, negative feedback is formed from TFs that repress their own genes [195]. The output of such a system can be locked into a steady state, usually accompanied by damped oscillations. The system can also be unstable, resulting in sustained oscillations [85]. Negative feedback tends to be noise resistant [90]. On the other hand, in positive feedback, a TF increases the transcription of its own CDS. Positive feedback increases cell-cell variability and can cause bistability [196].

A FFL is described as "a three-gene pattern that is composed of two input TFs, one of which regulates the other and where both jointly regulate a target gene" [197]. There are eight different FFL types which can be classified as either coherent or incoherent (Figure 2.8). These patterns show useful functionalities such as pulse generation (incoherent type 1) and signal filtering (coherent type 1) [195], and can be used in synthetic constructs in order to achieve the same behaviour with different genes [48].

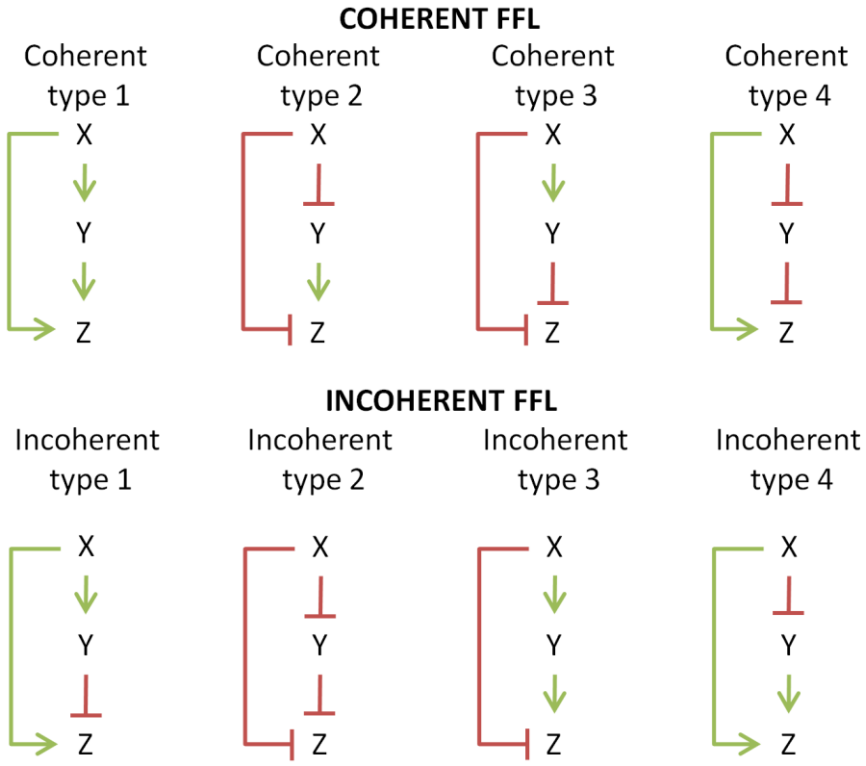


Figure 2.8: Eight types of FFLs (adapted from [195]). The most common types are coherent type 1 and incoherent type 1 FFLs, which can function as persistence detectors and pulse generators respectively. The detailed biological functions of each FFL are reviewed in [197].

Biological networks can be constructed using information from individual data sources about biological molecules such as genes, proteins and metabolites. However, a more comprehensive view of the data can be provided by integrating these data sources [198].

2.2.5.3 Integrating biological networks

Network approaches to integrated data representation can take advantage of existing tools and graph algorithms which offer, for example, search, clustering and pattern recognition algorithms [54]. There are several tools, such as Ondex [174], Cytoscape [173] and Osprey [199], for data integration and network analysis. All of these tools have their own strengths and weaknesses [200]. To facilitate automated analysis and reasoning over a network, a particularly valuable feature is the addition of semantic annotations to nodes and edges. These annotations can facilitate semantic data integration, in which conceptual representations of biological concepts and their relationships are used to eliminate semantic heterogeneities when integrating data from different sources [201, 202]. The Ondex data integration platform is probably the most semantically-aware integration tool currently available.

Ondex[6] is a graph-based data integration tool [174]. It combines semantic data integration and graph-based data analysis methods. The data are represented as networks in which biological concepts such as proteins and CDSs can be nodes, which are termed *concepts*. These concepts can be defined for anything. Semantic relationships, such as `encodes` and `part_of`, which are termed *relations*, annotate edges between nodes (Figure 2.9). Concepts and relations can have attributes that provide additional information.

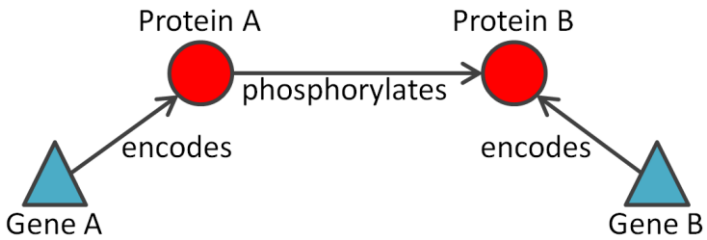


Figure 2.9: A small network of two genes and two proteins. Genes are connected to proteins by the `encodes` relation. The edge between the proteins represents the phosphorylation interaction.

Concepts and relations also have types, called *concept class* and *relation type* respectively. These types are organised hierarchically. For example, the `TF` concept

[6] http://www.ondex.org

class is a specialisation of the `Protein` concept class. The types of relations that a concept can have are described using concept classes and relation types. The metadata in an Ondex network also contains information about types of attributes and data sources. Entities from external data sources can be mapped to a model formed from these types, and can be integrated.

Data are imported through *parsers* which are used to convert data from different formats, such as tab- or comma-separated files, or XML, into the Ondex format, the XML-based Ondex Exchange Language (OXL) [203]. Parsers for a range of data sources such as KEGG, Gene Ontology (GO) [204] annotations and FASTA files are available currently, and new parsers can be written for any data source. Ondex provides the necessary framework with which to map and merge concepts from different data sources. *Mappers* are used to connect concepts based on identifiers such as names or accessions. The connectivities and properties of concepts can be changed using *transformers*, and unwanted concepts or relations can be removed by *filters*. *Exporters* are used to save networks in custom formats. For example, Ondex's OXL exporter is used to export networks as OXL files.

Data transformation and integration in Ondex is carried out using workflows. Workflows are implemented as XML files that include calls to parsers, transformers, mappers, filters and exporters stored in Ondex plugins. Once data are mapped, integrated and cleaned, the resulting network is exported as an integrated dataset. Graphs are used as a way of exchanging data between workflow steps. Each step may add, remove or merge nodes or edges. An example of an Ondex workflow is shown in Figure 2.10. In this example, two data sources are integrated and exported in OXL format.

Ondex has been used for data integration in a range of projects. For example, Cockell and co-workers integrated pharmaceutical characterisation and interaction data to identify possible drug repositioning candidates by identifying semantic motifs involving proteins, diseases and drugs [149]. Integrated networks have also been constructed for the large-scale investigation of various organisms; for example, in order to investigate telomere maintenance in *S. cerevisiae* [205], and to assess the functional association of protein pairs based on different evidence types in *Arabidopsis* [206]. Ondex offers a rich visualisation and querying platform for the manual and computational analysis of these biological networks.

The use of a single data model in a data integration exercise removes ambiguity about the meaning of identifiers. However, different semantics may be used by different

data sources. In order to address semantic heterogeneity and to provide interoperability between computer systems, Semantic Web technologies can be valuable [207].

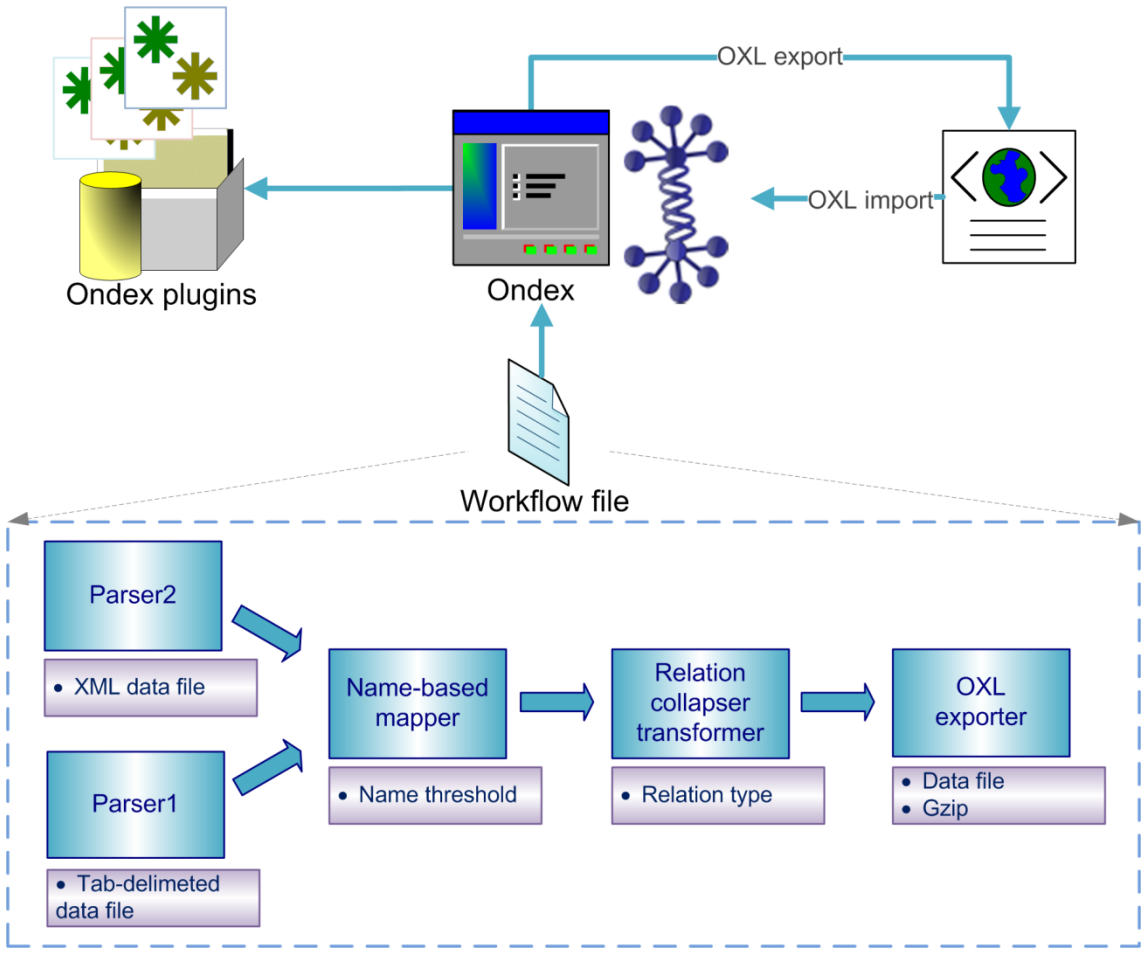


Figure 2.10: An example of an Ondex workflow. The workflow includes calls to two parsers in order to transform data from two data sources. The concepts from these graphs are mapped based on their names. The mapped concepts are then merged using the relation type that is used between the mapped concepts. The resulting graph is exported in OXL format, and can be exchanged with other users.

The Semantic Web started as a collection of languages and technologies to enable data on the Internet to be computer accessible [59]. These languages establish the meaning of terms and their logical connections, in order to provide interoperability between computer systems. Terms, for example, can represent biological concepts such as genes or proteins. The syntax used to identify these concepts is commonly XML. In the Semantic Web layer, the Resource Description Framework (RDF) and the Web Ontology Language (OWL) are two important languages [208]. RDF is suitable for the querying of data, and OWL sits on top of RDF and is useful to express the meaning of data.

2.2.6 Resource Description Framework

Synthetic biology is becoming a field that uses information extensively [49]. As well as storing physical genetic parts in freezers, sequences of parts are stored electronically in databases. However, part catalogues developed by individual research groups are currently disconnected from each other. Moreover, biological parts must be joined together in a biochemically consistent manner in order to form biologically plausible designs. Therefore, in addition to sequence-based descriptions of parts, the many interactions between biological molecules should be analysed and recorded. This information is dispersed among different bioinformatics databases and should be linked to the DNA sequence information of parts. The representation and linking of data can be facilitated by the use of RDF.

2.2.6.1 RDF as a data format

RDF is becoming increasingly popular among bioinformaticians as a common data format for biology. Several databases and knowledge bases such as UniProt [209], YeastHub [172] and Bio2RDF [60] provide their data in RDF format. By associating biological concepts with Uniform Resource Identifiers (URIs), entities from repositories can be linked to each other [59], and different RDF documents can be merged [162]. Therefore, RDF can be used as a way of aggregating information [61].

RDF has a graph-based structure that provides a schema-less model [162]. In these directed graphs, nodes represent resources and values, while the edges represent semantic relationships between nodes [210]. The properties of resources are asserted through statements in the form of *resource-property-value*, also known as *subject-predicate-object*, triples [211]. For example, the <'KinA', 'has function', 'kinase activity'> statement is a triple: in which the subject is 'KinA', the object is 'kinase activity' and the predicate is the 'has function' relationship (Figure 2.11). Values can be either resources or literals. Both resources and properties are identified by unique URIs.

An RDF document can be serialised in several formats such as XML, Turtle and N3 [212]. These syntaxes make an RDF document machine readable. Off-the-shelf RDF tools, APIs and querying languages are available, and allow the efficient storage and retrieval of RDF data using these standard formats.

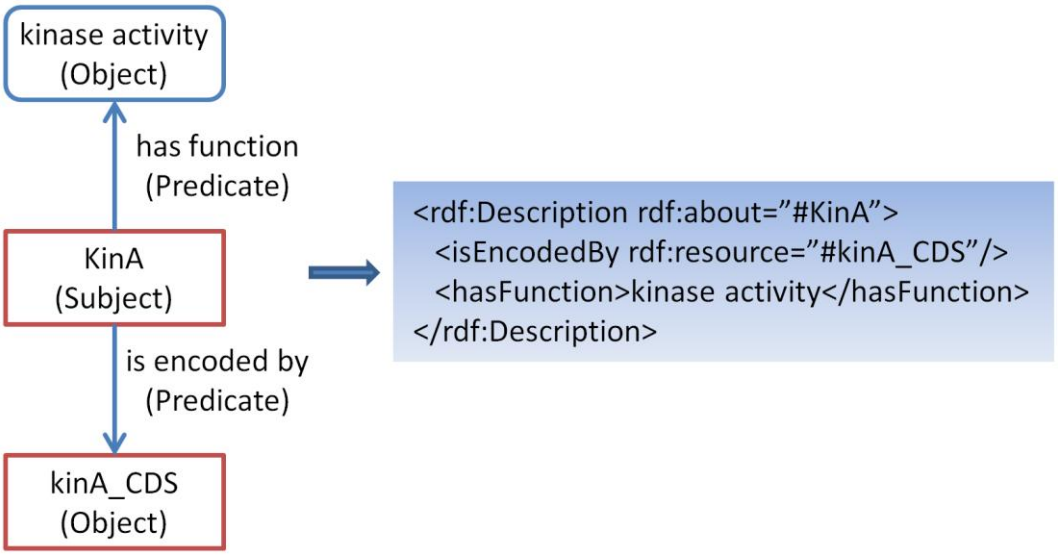


Figure 2.11: An example RDF graph in the form of two subject-predicate-object triples. In the diagram, the blue rounded and red rectangles represent literal values and resources respectively. The graph is directed, and therefore the statements can be read as 'KinA has the function of kinase activity' and 'KinA is encoded by kinA_CDS'. These triples can be written in the form of RDF/XML, as shown on the right-hand side.

2.2.6.2 Storage and retrieval of RDF documents

Data about biological parts and interactions can be accessed via RDF databases and linked using the existing Internet infrastructure [13, 185]. This approach has also previously been proposed for the linking of data in the life sciences. Goble suggested that service providers should publish their data online as RDF and expose RDF endpoints to create a web of biological data that can be crawled and queried in a similar manner to the existing World Wide Web [162]. These endpoints, called triple stores, are database systems that store RDF triples and allow queries to extract knowledge [60, 172]. For example, using this approach a Sesame RDF endpoint has been used to host data from the MIT Registry [13]. Endpoints can be queried using standard languages such as SPARQL [211].

SPARQL is a query language for RDF data [56]. A SPARQL query is a graph pattern used to find matching RDF triples [163]. In these queries, the subject, predicate or object from a triple can be substituted for query variables (Figure 2.12). SPARQL has different types of queries, such as SELECT and CONSTRUCT. In a SELECT query, column names are selected from query variables. For example, the 'KinA hasFunction ?function' triple pattern can be used to retrieve the functions of the 'KinA' subject. SELECT queries return results in the form of tables, while CONSTRUCT queries return RDF triples as an RDF graph. CONSTRUCT queries act as rules, and allow the extraction of information based on patterns from existing data [211].

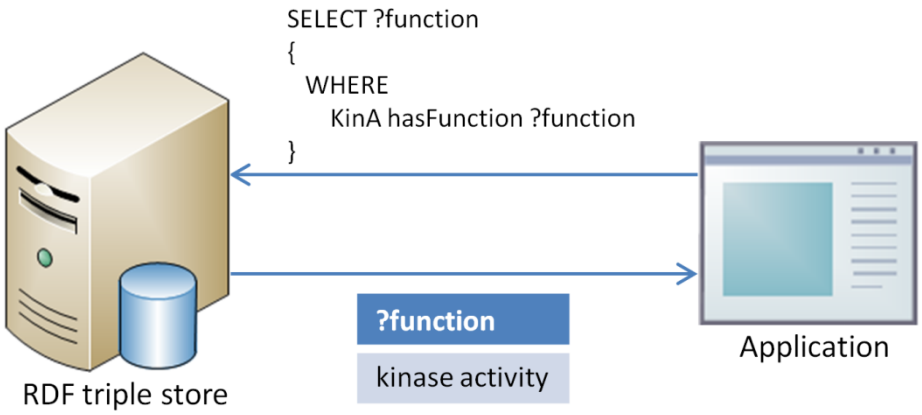


Figure 2.12: An example of a SELECT query. The '?function' query variable is used to find the molecular functions of KinA. The column name in the returned result set is the same query variable.

Although RDF documents can be effectively queried based on their graph structures, The RDF annotation of resources is not expressive enough to facilitate computational access [213]. The meaning of what is being annotated, or how, is not clear [214]. For example, in the <'KinA', 'has function', 'kinase activity'> statement, it is not clear what 'KinA' or 'has function' refers to, and hence these annotations are not machine-accessible. In order to provide machine-understandable annotations, ontology languages have been developed. These languages can be used to define new terms with explicit meanings, as needed [213].

2.2.7 Ontologies

Large-scale synthetic biology can be facilitated by computational design approaches. Therefore, extensive information about parts and molecular interactions must be available to computational tools. The meanings of biological concepts from different data sources must be unified without ambiguity. Ontologies in the life sciences have already been widely used to provide a shared understanding of domains between researchers and computers, and interoperability across databases [62]. Biological concepts and their relationships can be modelled via ontologies, making expert knowledge machine accessible [59, 208].

2.2.7.1 What is an ontology?

Ontologies are used to make knowledge machine-accessible by defining objects from a domain of interest in a manner explicitly intended for use by computers. Gruber defines an ontology as "an explicit and formal specification of a conceptualisation" [35]. Such a conceptualisation uses an abstract and simplified view of a domain being modelled. [60]. Knowledge in ontologies reflects reality as conceived by human brains [215]. This

shared understanding of a domain is captured and used in computer applications, via ontologies, as formal specifications [144]. The representation of the entities of a domain is then consistent and unambiguous [146]. This representation is defined explicitly by a list of terms and the relationships between them [210].

Ontologies are not dictionaries which are prepared for human beings [215]. Terms in an ontology have labels and descriptions, and are identified by unique URIs [59]. The set of properties linked to a term specifies that term's meaning [216]. Terms are structured in a hierarchy, with terms at higher levels having more general meanings [210], while lower level terms are specialisations of upper level terms [147] (Figure 2.13). In addition, because they may have more than one parent, the definitions of child terms are enriched. Unless all of the parents of a term are specified the definition of that term is incomplete [215].

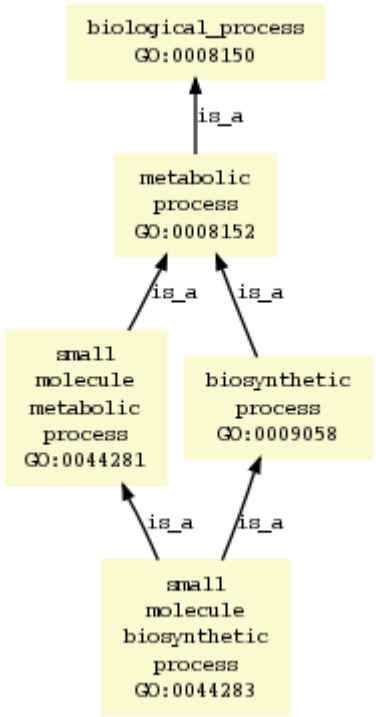


Figure 2.13: An example of a hierarchy of ontology terms. The diagram shows the 'small molecule biosynthesis process' term and its parents from GO[7]. This term represents a biological process and has two parent terms, which in turn have a single, common parent.

The existence of explicitly defined and agreed terms facilitates the removal of semantic heterogeneity [144] and the sharing of information [56]. Controlled vocabularies provide meaning to the text, defining the semantics of data in documents

[7] http://www.ebi.ac.uk/ontology-lookup/browse.do?ontName=GO&termId=GO:0008152 (accessed 31/07/2012)

written using different syntaxes [216]. Information can therefore be organised and processed computationally [56, 62]. Life science researchers have contributed significantly to the development of appropriate ontologies [59].

2.2.7.2 Ontologies for biological domains

Ontologies have been used by bioinformaticians for over a decade. The TAMBIS (Transparent Access to Multiple Bioinformatics Information Sources) ontology, constructed in the late 1990s, was one of the early ontologies to capture biological knowledge [143]. The ontology included terms for biological concepts such as proteins and molecular functions, and aimed to facilitate the integration of data from different data sources by providing unified meanings for biological terms. The Open Biological and Biomedical Ontologies (OBO) foundry coordinates the development of biological ontologies and sets out guidelines for their development [217]. So far, six ontologies, including GO, are listed as OBO ontologies and others currently are being reviewed [208]. In addition, repositories such as the EBI's Ontology Lookup Service provide a common user interface to navigate among and visualise terms from various biological ontologies [208, 216].

Most of these ontologies are based upon controlled vocabularies. For example, GO is the most widely-used ontology in biological research [218], and has been developed to standardise the annotation of genes [204]. This ontology provides terms that are used across different biological domains and has three sub-ontologies detailing cellular components, molecular functions and biological processes. Other examples of ontologies include: the Protein Ontology, for proteins, protein complexes and their relationships [219]; the Cell Cycle Ontology (CCO), for cell cycle processes [146]; the Cell Ontology for cell types [220]; and the Sequence Ontology (SO) for DNA sequence features [221].

Automated reasoning approaches can be applied to OBO ontologies. The phosphatase ontology [144, 222] was developed to classify the phosphatases in a genome. Its classes are defined based on the p-domain composition of proteins. The presence of a particular set of domains in a phosphatase is used for the classification of proteins. Another ontology, CCO [146], classifies proteins using features such as their cellular locations and biological processes. These ontologies are encoded using standard languages.

2.2.7.3 Representing ontologies

The RDF Schema (RDFS) and OWL are examples of two ontology languages [163].

RDFS defines a vocabulary to describe objects in an RDF data model [210]. This language extends RDF in order to define special RDF resources, including `Class` and `subClassOf` resources which are used to describe a hierarchy for other RDF resources [213]. For example, <'KinA', 'rdf:type', 'Protein'> and <'Protein', 'rdfs:subClassOf', 'PhysicalEntity'> triples state that 'KinA' is a kind of 'Protein' and that 'Protein' is a kind of 'PhysicalEntity'. However, RDFS has limitations. For example, it is desirable to define new classes using the conjunction or union of existing classes [213]. A more expressive language is OWL.

OWL is recommended by the World Wide Web Consortium for expressing ontologies [214] and OWL extends RDFS by providing additional terms, such as `cardinality`, `ObjectProperty` and `inverseOf`, as well as supporting the linking of terms via the use of URIs. Hence, an ontology can refer to terms from other ontologies [59].

The basic unit in ontologies is the class [222]. Classes define the types of objects in a domain, and are the representations of things in the world [144]. Classes represent a set of individuals [210, 222]. For example, `Protein` and `Gene` can be the classes of all individual proteins and genes respectively [143]. OWL also defines terms which capture relationships between classes. These relationships are inherited by the instances of an OWL class. Classes that do not have any instances in common are described as disjoint [220], and the individuals of such classes are also disjoint. The OWL syntax is used to define classes, individuals and the relationships between them, using information such as the properties and the specifications of logical relationships [214].

The relationship between two individuals is described using *object* properties. For example, information about a gene encoding a protein can be captured by using the `encodes` relationship between the gene and protein individuals (Figure 2.14). In addition, *datatype* properties are used to describe the relationships between individuals and data values. Object and datatype properties can be applied to classes in the form of restrictions.

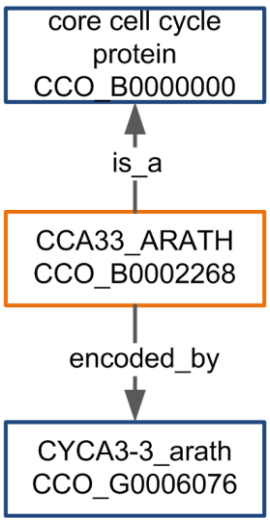


Figure 2.14: An example of relationships between ontology classes. The `CCO_B0002268` class from CCO represents the CCA33_ARATH protein and is linked to the gene class via the `encoded_by` object property. This relationship is inherited by all of the individual instantiations of the protein class. CCA33_ARATH is a core cell cycle protein and this relationship is implemented as subclassing in the ontology.

A restriction is used to create a set of individuals that satisfy a condition [222]. Restrictions can be considered as anonymous super classes, which can be created using existing classes and properties [211]. OWL has four different restriction types:

- Existential restrictions, also known as *SomeValuesFrom* restrictions, describe classes having at least one relationship that restricts the individuals. For example, this relationship can be applied to a class in order to represent animals that eat meat [210]. However, such a condition does not restrict whether or not the class includes animals that eat plants.
- Universal *AllValuesFrom* restrictions specify that, if a relationship exists, this relationship must apply to all individuals of a particular class. Such a relationship cannot have any other value except for the classes specified by the restriction. For example, if a class includes a relationship to state that animals eat meat in the form of the AllValuesFrom restriction, then that class represents all individuals that only eat meat.
- *HasValue* restrictions are used to restrict the individuals of a class to possessing only given data values.
- Cardinality restrictions are used to describe the minimum, maximum and exact cardinality relationships between individuals of a class.

Once an OWL ontology is built using classes, individuals and restrictions, it needs to be serialised in a computer-readable format. OWL ontologies can be exported as RDF [212]. In addition, the Manchester syntax [223] provides a shorthand representation for

OWL axioms. For example, SomeValuesFrom, AllValuesFrom and HasValue restrictions are represented with the `all`, `some` and `value` keywords using this format [211]. Subsumptions in the form of anonymous classes formed by these restrictions are crucial for automated reasoning [146, 222].

2.2.7.4 Reasoning using ontologies

In order to construct biologically functional constructs, biological parts must be selected carefully [16]. This selection is initially guided by the properties of the biological parts, and the identification of desired and undesired interactions between the genetic circuits and their host cells. For large-scale designs that require computational approaches, the identification of parts should be automated. The rich expressivity of ontologies and existing tools and APIs can facilitate the efficient automated querying of data for large-scale synthetic biology.

While a conceptual language such as OWL is used to define the objects and their relationships [60], the use of logics [208] allows reasoning over the data. Ontologies use Description Logics (DL), a knowledge-based representation formalism, for representation and reasoning [146, 224].

Subclassing is central to reasoning [143], since subclasses inherit the relationships of their superclasses. If instances of one class are also instances of another class, the former is said to be the subclass of the latter. In order to facilitate reasoning about inheritance, parent-child relationships are explicitly specified via the *is_a* relationships [215]. In addition, restrictions are added to classes using subclassing. OWL queries can be constructed using the definitions of these restrictions and hence classes with the matching restrictions can be extracted.

In OWL, queries are anonymous classes that define the knowledge asserted by users [146]. Queries run on off-the-shelf reasoners are used to make implicit knowledge explicit. Queries are constructed using conditions that must be fulfilled. A condition that is implemented with a restriction is known as a *necessary condition* [146] and classes with necessary conditions are known as *primitive classes* [225]. For example, the ‘Lion eats some Herbivore’ restriction would be necessary in order to classify lions according to the food consumption pattern of animals.

In order to extract implicit knowledge, conditions that are sufficient must also be used to describe classes. These conditions can be expressed using *Equivalent* classes [213]. A class with both necessary and sufficient conditions is known as *defined* [225]. A similar restriction for the above example can be used to create the Carnivore class.

The ‘Carnivore eats only Herbivore’ class definition would be necessary and sufficient to classify animals that are carnivores.

However, it may not be possible to infer that individuals belong to a class defined with these conditions, even given that the conditions are met. OWL-DL is based upon the open world assumption, in which facts that are not explicitly stated are assumed to be unknown, and hence unless stated there may be additional information [222]. Therefore, the descriptions of such classes must be restricted by using closure axioms [226]. For example, without closure axioms, it is not clear whether lions eat plants, and thus lions cannot be classified as carnivores. In order to classify lions correctly, the ‘Lion eats only Herbivore’ closure axiom must be added to the Lion class.

Classes and their individuals can be viewed using tools such as Protégé[8]. Protégé also allows the use of reasoners. Once a reasoner such as HermiT[9] [227] or Pellet[10] [228] is executed, the ontology loaded into Protégé can be queried using the Manchester syntax for OWL-friendly queries. Furthermore, the Ontology Pre-Processing Language[11] [229] plugin of Protégé can be used to add, update and delete axioms by querying in a similar fashion. The reasoners provide APIs that allow programmatic control over the reasoning process. In addition, an ontology querying language called Terp [230] has been developed. This language is based on the Manchester syntax [223] and SPARQL, and can be used to query ontologies programmatically using Pellet APIs. When an ontology is stored in the RDF/OWL format, it becomes a valid RDF document and therefore can also be queried using SPARQL [146, 213].

Ontologies are also increasingly used in synthetic biology. Recently, a data standard has been proposed in order to exchange information about biological parts [63]. This standard format, called Synthetic Biology Open Language, is backed by an ontology.

2.2.8 The Synthetic Biology Open Language

For the automated design of biological systems from specification to synthesis, a range of tools and services should be integrated. Several tools that address needs in synthetic biology have been developed. Examples include CAD and automated design applications, and tools such as those that can predict quantitative parameters for biological parts, as well as tools which facilitate the manual composition of DNA sequences [22]. Publicly available catalogues of parts have also been developed [12, 13,

[8] http://protege.stanford.edu
[9] http://hermit-reasoner.com
[10] http://clarkparsia.com/pellet
[11] http://oppl2.sourceforge.net

93, 231]. In order for the effective interpretation and electronic exchange of data between these tools, and to access repositories computationally, standard formats are needed. Therefore, DNA sequence specifications of parts and their mathematical models must be represented using accepted data standards.

The Synthetic Biology Open Language (SBOL) is a language which has been developed for the standardised exchange of information about nucleotide sequences and the composition of biological parts and devices [63]. This language specifies a computer-readable format for data that can be exchanged between users, computational design tools and DNA synthesis companies.

SBOL is an RDF-based language. The semantics of the language were built upon OWL, and can be interpreted by any RDF-compliant tool (Figure 2.15) [13]. In the SBOL data model, biological parts are represented by `DnaComponents` which can have multiple types, one of which is selected from the SO. Each `DnaComponent` has a `DnaSequence` that stores the nucleotide sequence for a part. The physical locations of sequence features within `DnaComponents` are represented with `SequenceAnnotations`. Each `DnaComponent` may have more than one `SequenceAnnotation`. The feature described by an annotation is also a type of `DnaComponent`, and a list of `DnaComponents` can explicitly be grouped using a `Collection`.

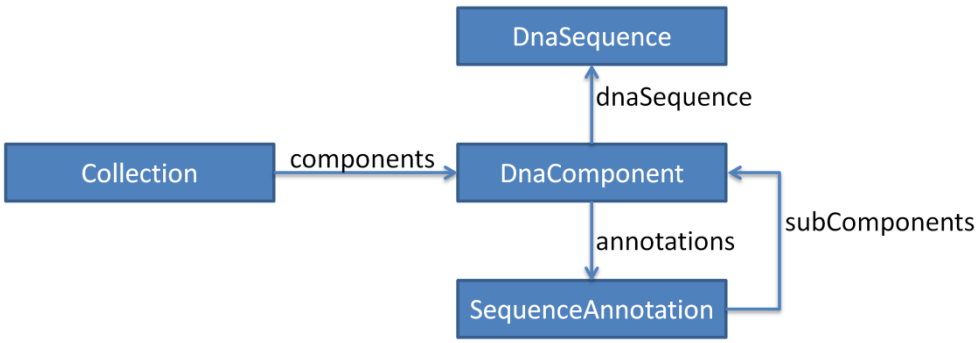


Figure 2.15: The SBOL core data model. `DnaComponents` represent parts. Nucleotide sequences and the physical locations of sequence features can be accessed via `DnaSequences` and `SequenceAnnotations` respectively. `DnaComponents` can be grouped in `Collections`.

Galdzicki and co-workers modelled the information about parts from the MIT Registry [12] using the initial version of SBOL [13]. The data were hosted at an RDF endpoint, enabling the SPARQL querying of parts. However, interactions between parts and physical entities are currently not modelled. Although an extension for modelling has been discussed, there is also no link between dynamic models and SBOL. The use

of semantic annotations can aid to create this mapping.

Throughout this chapter, data integration and tools such as e-Science and Semantic Web technologies which address the integration of large-scale data for synthetic biology have been presented. In addition, SBOL as a data standard for the electronic exchange of information about biological parts has been discussed. However, the analysis of properties and relationships between biological molecules alone is not sufficient to understand the dynamics of biological systems [191]. The topological information stored in biological networks can be used to guide the construction of quantitative models with biochemical parameters for detailed analyses of these systems.

2.3 Quantitative modelling of biological systems

Synthetic biological systems are ideally designed iteratively via modelling, construction, and experimental testing [48]. Mathematical models are used to guide the design of a system and the cycle of design, simulation, testing and modification is continued until the system achieves the desired results [87, 232]. Mathematical analysis and computational simulation is indispensable in order to observe the behaviour of large genetic circuits [53], and hence to optimise biological designs *in silico* prior to wet-lab implementation [22, 24].

Biological system dynamics can be modelled using quantitative mathematical models such as deterministic and stochastic models. Ordinary differential equations (ODEs) can be used to describe changes in gene expression over time [233], and stochastic models and probability theory can help to model heterogeneity in cell populations [84, 234]. In these models, biochemical parameters such as binding affinities or rate constants can be tuned and tested to enhance the behaviour of biological systems [235].

The behaviour of a biological system is the function of interactions between a myriad of molecules [236]. Therefore, in order to engineer predictable synthetic circuits, these interactions should be analysed. Even simple circuits can exhibit non-intuitive behaviour. For large and complex systems quantitative models that support computational simulations are needed [53]. Models should be built at an appropriate level of abstraction in order to obtain insights into the engineered systems [237].

2.3.1 Modelling assumptions

A model is an abstraction of reality [189]. The type of model and the choice of level of abstraction are mainly driven by the nature of the system to be modelled and the questions being asked [22, 189]. While irrelevant details should be left out, fundamental details must be captured in models [22, 37]. However, deciding upon an appropriate level of abstraction at which to model can be challenging, since the resulting models must still capture the behaviour of the systems being modelled. These simplifications allow researchers to concentrate on the problems they are trying to answer [157].

In quantitative biological models, complex cell processes are simplified. For example, the transcription rates of mRNAs and translation rates of proteins can be modelled with simple rate parameters. In this approach, the number of total, free or bound polymerases are not tracked, on the assumption that polymerases exist in excess of required levels [238]. The numbers of ribosomes and the diffusion of mRNAs around

the cell can also be ignored, assuming that ribosomes are not a limiting factor [80] and that mRNAs are uniformly distributed [48].

Furthermore, mathematical transfer functions can be used to map external inputs to measurable experimental outputs. These functions usually have relatively few parameters, simplifying the construction of models, while still providing insights into the behaviour of biological systems [189]. For example, protein concentrations are affected by many factors in a cell, including transcription and translation rates, mRNA and protein degradation rates, mRNA structure formation, CDS length, amino acid composition and start codons [111-113, 159, 238, 239]. Each of these factors can be involved in multiple interactions. In order to model these interactions, a transfer function, such as a Hill equation, can be used to directly map an input, such as the concentration of an inducer molecule, to a reporter, such as GFP [240]. The assumptions chosen are implemented using a suitable modelling formalism.

2.3.2 Modelling formalisms

There are many modelling formalisms, such as Boolean logic [189], stochastic models [234] and differential equations [241]. The choice of a formalism can be based on desired model behaviour: for example, quantitative versus qualitative, discrete versus continuous, or deterministic versus stochastic [157]. Deterministic models, for example, can be used to model the average behaviour of molecules [22]. The change in concentrations is deterministic and there is no randomness.

ODEs are commonly used in deterministic models. These equations are useful to describe a rate of change for a variable and involve only one independent variable [238]. The instantaneous rate of change for the dependent variable is called the *derivative* of that variable. For example, speed is the derivative of position with respect to time, and can be given as the ratio of change in position to change in time. When derivatives are known, ODEs can then be used to calculate the value of the dependent variables.

ODEs can also capture the dynamics of biological systems using two assumptions [156]. In these models, systems are assumed to be continuous and rate equations describe the velocities of reactions. This assumption is valid for systems involving large numbers of molecules. However, when the number of molecules is small, stochastic differential equations might be more appropriate. Secondly, these equations assume homogeneity in space, indicating that molecules are well distributed in the cell [237]. Hence the ODE variables are time, but not space, dependent [22]. Variables such as

protein and mRNA concentrations can be thought of as derivatives of time, and hence can be expressed as ODEs. However, the assumption of homogeneity does not hold for biological reactions which have fast reaction rates, since concentrations of molecules may vary also with respect to space [156]. When the appropriate assumptions are met, however, complex reaction networks can be expressed using differential equations.

ODEs have been used widely for the modelling of biological systems. The designs of early genetic circuits such as the repressilator [107] and the bistable switch [108] were guided by the use of ODE-based modelling. Ellis and co-workers used ODEs to design a genetic timer [71]. Based on predictions from the modelling of the device, the authors used the timer to experimentally control the timing of sedimentation in *S. cerevisiae*. In addition, ODEs have been used to provide insights into natural systems. Voigt and co-workers investigated the role of the SinR protein in controlling population heterogeneity in *B. subtilis* [188]. In this system, the binding of SinR to its antagonist protein and the negative autoregulation of its own gene form a bistable switch architecture. The model predicted that tuning a critical parameter in this system can lead to additional behaviour such as oscillation. ODEs were also used to construct a dynamic model of the competence induction system in *B. subtilis*. ComK, the master regulator of competence, positively regulates the expression of its own gene, and negatively regulates itself by repressing a protein that inhibits the degradation of ComK. The model predicted that these positive and negative feedbacks controlled how cells become competent and return to the vegetative state [196]. The model was also tested experimentally. When the negative feedback was removed, competent cells were unable to return to the vegetative state.

Using ODEs, the process of protein production can be simplified to two abstract reactions representing transcription and translation [41, 238]. In the first reaction, when the promoter is constitutive, mRNA is produced from DNA at a defined production rate which can be given as mRNA molecules per DNA molecule per second. It can be assumed that all of the mRNAs degrade over time at a given rate [41]. Therefore, the rate at which the amount of mRNAs change can be given as:

$$\frac{d[mRNA]}{dt} = k_{transcription} - k_{degradation} * mRNA$$

where $k_{transcription}$ is the transcription rate, $k_{degradation}$ is the degradation rate, and $mRNA$ is either the concentration or number of mRNA molecules.

The transcription rate is not always constant. TFs can bind to regulatory sequences on promoters to activate or inhibit a promoter's activity. Different TFs can bind cooperatively, or can compete with each other for the same binding region, affecting the recruitment of RNA polymerases (RNAPs). Hence the transcription from a promoter can be quantified with the probability that RNAPs bind to the promoter [78, 242]. This probability is called promoter occupancy [238]. A transfer function can be used to model the interactions between activators, repressors, helper molecules and RNAPs [242] using the Hill equation [241]:

$$\frac{TF^n}{TF^n + Km^n}$$

where *TF* is the amount of a particular TF, *Km* is the disassociation constant, and *n* is the Hill coefficient. This function determines the probability of a particular TF being bound to a promoter.

The transcription rate from an inducible promoter can be very low when the promoter is not occupied by an activator. When an activator is bound, the rate of transcription can increase to a maximum level. Therefore, the activity of an inducible promoter is a function of the transcription rate and the probability that the activator is bound [41, 48, 241]:

$$\frac{d[mRNA]}{dt} = k_{transcription} * P_{activator} - k_{degradation} * mRNA$$

where $k_{transcription}$ is the maximum transcription rate that can be reached, and $P_{activator}$ is the promoter occupancy of the activator.

In repressible promoters, transcription occurs when the repressor is not bound [48]. Hence, the transcription rate is modified by the probability that the repressor is not bound:

$$\frac{d[mRNA]}{dt} = k_{transcription} * (1 - P_{repressor}) - k_{degradation} * mRNA$$

where $k_{transcription}$ is the maximum transcription rate that can be reached, and $P_{repressor}$ is the promoter occupancy for the repressor.

The concentration of protein is given by the rate of translation from mRNA

molecules per second. All of the proteins degrade in time at a given degradation rate. Therefore, the rate of change can be represented as protein production minus degradation [41, 189, 241]:

$$\frac{d[Protein]}{dt} = k_{translation} * mRNA - k_{degradation} * Protein$$

where $k_{translation}$ is the translation rate, $k_{degradation}$ is the degradation rate of the protein, and $Protein$ is either the concentration or number of protein molecules.

Although computational tools can have custom GUIs to construct ODEs, models must be in standard formats in order to allow the exchange of models and gain access to repositories of models. Standard languages such as the Systems Biology Markup Language (SBML) [65] and CellML [243] support ODE-based models.

2.3.3 Modelling languages

Modelling languages provide the syntax for the modelling of biochemical reactions. SBML and CellML are two widely-used modelling languages [32]. These formats extend XML [56], allowing the exchange of models between users and tools.

2.3.3.1 The Systems Biology Markup Language

The SBML website lists 232 compatible software packages[12], allowing the import and export of SBML models. A typical SBML model consists of entities (hereafter referred to as 'modelling entities') such as *species* and *reactions* [53]. Species can represent molecules such as mRNAs and proteins. The goal of a model is often to predict the final concentration or number of these molecules. Final amounts of these molecules for a given simulation time are calculated using the information about the biochemical reactions they participate in.

These reactions may be, for example, transcription, translation, degradation and complex formation, and are represented by the SBML's reaction entities. Species are consumed or produced by these reactions, and can also be modifiers, such as enzymes that catalyse biological reactions. For example, in order to model mRNA transcription from an inducible promoter activated by a TF, the 'mRNA' and 'TF' species can be defined. In this example, the 'mRNA' species is produced by a reaction in which 'TF' is the modifier. Using another reaction entity, the degradation of these mRNA molecules, in which the 'mRNA' species is consumed, can be represented. The role of a species is

[12] http://sbml.org/SBML_Software_Guide (accessed 05/04/2012)

explicitly stated in each reaction entity using *speciesReference* entities. In addition, reaction entities use *parameter* entities that specify the velocity of biochemical reactions.

These parameters can be defined locally inside the reactions, or they can be defined globally in an SBML model and can be used by any reaction entity. The latter approach would be useful in order to use, for example, fixed degradation rates for all proteins being modelled. As these parameters can have constant values, they can also be based on *assignment rules.*

These rules, parameters and species can be used in reactions in order to construct mathematical formulae that represent reaction fluxes. For example, the mRNA transcription reaction can be given with the formula:

$$k_{tr} * \frac{TF}{TF + Km}$$

in which the k_{tr} and Km parameter entities represent the transcription rate and disassociation constants, and *TF* is the modifier species. The mRNA species produced in this reaction is not included in the formula, but is explicitly specified as the output via the species reference entities. Similarly the mRNA degradation reaction can be given with the formula:

$$k_{deg} * mRNA$$

in which the k_{deg} parameter represents the mRNA degradation rate and *mRNA* is the mRNA species that is consumed by the degradation reaction. There is no species produced in this reaction.

The sum of all reaction fluxes for a species represents its rate of change with respect to time, and hence can be expressed as an ODE. For a species that is produced by a reaction, the reaction flux contributes as a gain; whereas for a species that is consumed by a reaction, the flux contributes as a loss. For modifier species, this flux has no affect. Therefore, for the above mRNA example, the total reaction flux for the mRNA species is the transcription reaction flux minus the degradation reaction flux, which gives the ODE of the mRNA species as shown below:

$$\frac{d[mRNA]}{dt} = k_{tr} * \frac{TF}{TF + Km} - \mathrm{k_{deg}} * mRNA$$

However, the ODE expression of a species entity is not explicitly captured in models, and is constructed during simulation by SBML-compliant simulator tools, such as COPASI [244] and CellDesigner [45]. In addition, time is not represented as an entity in SBML models. CellML is also similar to SBML in representing these ODEs, although there are some differences.

2.3.3.2 CellML

CellML is also a computational modelling format [243]. However, CellML models do not have specific entities that represent species and reactions, as does SBML. Although in SBML it is not possible to directly infer what species and reaction entities refer to, semantics about their types can be assigned. For example, species and reaction entities represent cellular molecules and biochemical reactions respectively. CellML, on the other hand, uses more abstract terms to model biochemical reactions, cellular molecules and rate parameters.

The basic building blocks in CellML models are *components*, which can be flexibly used to model biological systems. Components include *variable* entities, which can represent simple rate parameters, or can be assigned to mathematical equations as in SBML's reaction entities. However, although mathematical formulae in SBML's reaction entities are used directly to construct ODEs, those in CellML are simple assignment rules. Mathematical equations that represent ODEs can be explicitly specified as differential equations. Therefore, the full ODE expression for a cellular molecule needs to be explicitly written in CellML using all of the relevant reaction fluxes. These ODEs include variables that represent cellular molecules and requires time to be included as a modelling entity. The production of mRNA from an inducible promoter can therefore be given as shown below, using the text-based representation in the Cellular Open Resource [245] modelling tool:

$$ode(mRNA, time) = k_{tr} * TF/(TF + Km) - \mathrm{k_{deg}} * mRNA;$$

where the *mRNA* and *time* variables represent mRNA molecules and time respectively, and the *ode(mRNA,time)* expression specifies that the equation on the right-hand side is an ODE for mRNA molecules. The variables k_{tr}, *TF* and *Km* represent the transcription rate, TF molecules and a disassociation constant respectively.

Using one component, the ODEs for a biological system being modelled can be written and simulations can be performed for the variables represented by these ODEs. However, using this representation, all of the biological reactions in which each cellular

molecule is involved are combined into one ODE expression. As a result, for species that are produced or consumed by similar reactions, the equations representing reaction fluxes are repeated. CellML is flexible in representing reaction fluxes in different CellML components which can then be connected to form these ODEs.

In order to reuse reaction fluxes from other components, CellML uses variable entities. These entities can also be defined as public or private. Moreover, the public variables can be inputs or outputs and are used to connect components via the *connection* entities. In a connection entity, a public output of a component entity can be joined to the public input of another component entity. Therefore, reaction fluxes or other commonly defined variables such as time or volume can be used as inputs in CellML components. The use of connections between CellML components enables the separation of models of systems into modular models.

CellML 1.1 has explicit support for modularity, enabling a model to be composed of smaller models [64]. Models can include import statements specifying the location of models to be included. These import statements also include a list of components to be imported. Such components can then be used in the constructed models as if they had been locally defined. Existing connections between the components from imported models are used directly in the constructed model, and imported components can be connected to other components via the connection entities.

Both the CellML and SBML languages are valuable for the modelling of synthetic genetic circuits. Entities in these models, such as species and reactions in SBML and components in CellML, can be used to represent biological parts and molecular interactions. Moreover, these models can be annotated with machine-accessible information in order to allow the parts and interactions captured in these models to be identified computationally.

2.3.4 Annotation of models

XML-based modelling languages define the syntax used to mathematically describe biochemical reactions between biological molecules. This information is necessary to run simulations [81]. However, in order to provide the necessary biological context, relevant modelling entities should be mapped to their biological counterparts. The use of semantic annotations for this mapping enriches models and allows computational tools to access biological information stored in these models [202].

The availability of access to the semantics of entities in models has many advantages. Models can be exchanged and reused [81]. Larger models can be

constructed from smaller models, and models can be merged and checked for inconsistencies [246]. In addition, annotations can be used as links to access a wealth of information from the bioinformatics databases and literature [247], facilitating the automation of the retrieval and integration of data for models.

The Minimum Information Requested in the Annotation of Models (MIRIAM) standard [248] was proposed in order to standardise the annotation of quantitative models. In this proposal entities in the mathematical models are linked to external information through the use of unique URIs. Another proposal in the MIRIAM project is to embed annotations in quantitative models. Both CellML and SBML models allow RDF annotations to be stored in the XML structure of models [53]. Mappings between annotations and modelling entities are provided by the use of IDs that are defined for metadata.

In CellML, RDF triples in the form of RDF/XML can be directly embedded in the XML structure of the corresponding component entities (Figure 2.16). The mapping between a component and its annotation is achieved by referring to the `cmeta:id` attribute of the component via the `rdf:about` property of the annotation [249].

```
<component xmlns="http://www.cellml.org/cellml/1.0#"
name="BugBuster_Promoter_spaRK" cmeta:id="BugBuster_Promoter_spaRK">
    <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:mts="http://purl.org/modeltosequence/1.0#">
      <rdf:Description rdf:about="#BugBuster_Promoter_spaRK">
        ...
      </rdf:Description>
    </rdf:RDF>
<variable name="localVolume" units="femtoliter"
public_interface="in"/>
...
</component>
```

Figure 2.16: Annotation of a CellML component.

In SBML, modelling entities can have `metaid` attributes that can be referred to by RDF annotations [250]. These annotations are also embedded in the XML structure of the corresponding modelling entities. SBML has a special XML node called `annotation`, which allows any annotation to be stored in the form of valid XML. In the SBML specification, this XML node should contain RDF data called MIRIAM annotations, in order to refer to external terms from controlled vocabularies. This RDF data is constructed using the BioModels qualifiers, which includes terms such as `is` and `hasPart` in order to describe modelling entities. In addition, it has been proposed that other types of annotations that are specific to applications should also be stored inside

the annotation element, albeit in another top-level XML element. This element should be identified with an identifier from a namespace that can be recognised by other applications (Figure 2.17). This scheme allows different applications to add annotations to models without affecting each other.

```
<species id="MyProtein" name="MyProtein" metaid=" MyProtein"
compartment="cell" initialConcentration="0">
        <annotation>
          <MyApp xmlns="http://purl.org/myapp/1.0#">
            <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#" xmlns:myapp="http://purl.org/myapp/1.0#" >
              <rdf:Description rdf:about="#MyProtein">
                ...
              </rdf:Description>
            </rdf:RDF>
          </MyApp>
          <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-
ns#" xmlns:bqbiol="http://biomodels.net/biology-qualifiers/">
            <rdf:Description rdf:about="#MyProtein">
              <bqbiol:is>
                ...
              </bqbiol:is>
            </rdf:Description>
          </rdf:RDF>
        </annotation>
      </species>
```

Figure 2.17: Example of an annotation for an SBML species modelling entity. The MyApp top-level element is used to store the application-specific data in the form of RDF. The second top-level element can be processed by SBML tools and contains references to external terms.

Chapter 3. Data Integration for Synthetic Biology

Synthetic biologists aim to rationally design biological circuits. This process is greatly simplified when the biology of the engineered organisms is well understood. In this chapter, we discuss the rational design of genetic circuits for *Bacillus subtilis*. Knowledge about the biology of *B. subtilis* is distributed among a variety of databases with different data types, formats and domain models. To capture a fuller picture of *B. subtilis* cell biology at a systems level, these data require integration. In this chapter, research underpinning approaches to the integration of *B. subtilis* data to inform the design of synthetic genetic circuits is presented.

3.1 Introduction

A wide range of synthetic genetic circuits has been designed. Most of the applications developed are small systems built manually from a few genes [37, 38]. Currently, large-scale projects are not cost- and time-effective [20]. Furthermore, techniques to construct, analyse and model genetic circuits are still carried out in a predominantly *ad hoc* fashion [15]. There is an emerging consensus in the synthetic biology community that the delivery of fast, affordable and predictable designs requires the use of computational design and simulation [20-22].

The information contained in biological databases is invaluable for the development of design strategies [118], the construction of individual parts, and the creation and validation of complex, predictable designs [28]. Most databases use disparate data types, formats and conventions, and utilise different types of biological domain models [164]. These data sources must be integrated to optimise their use for the rational design of large and complex synthetic biological systems. Furthermore, in order to promote integration the data should be presented in a computationally amenable format using standardised data formats where possible.

The availability of integrated biological datasets promises to enhance tool support for synthetic biology. Computational design tools, genetic circuit design algorithms, and domain specific languages (DSLs) for specifying genetic circuits have been developed (reviewed in [32]). Computer-aided design (CAD) tools such as Promot [251] and TinkerCell [252] allow users to select parts with which to sketch genetic designs.

Evolutionary algorithms can also be used to find different solutions to the problem of designing genetic circuits by, for example, using input-output mappings [18]. DSLs [123] such as GEC [43], Eugene [21] and Proto [117] have been developed to specify circuit designs. The tools that support these languages implement designs using part catalogues, but the solution space explored by these design processes is potentially very large [21, 73]. Each solution in a given solution space may combine different numbers of genetic parts, logic gates, and network topologies to achieve a desired behaviour [77-79]. As the complexity of applications and the ambition of researchers to create novel applications increase, the number of possible solutions grows exponentially due to the variety of parts and interactions required [76]. However, not all solutions are equally biologically plausible or implementable [74]. One way to constrain the design space is to incorporate existing biological knowledge from integrated data sources, to identify the biologically plausible regions of solution spaces [75].

Integrated biological knowledge bases can also provide a useful source of new parts for synthetic biology. CAD tools require access to catalogues that store diverse types of computationally amenable parts to enable the creation of large-scale designs [21, 43, 117]. However, the number of parts characterised experimentally so far is low compared to the huge number of available annotated genomic sequences such as promoters, operators, coding sequences (CDSs) and terminators. In March 2012, the NCBI's GenBank sequence database included more than 149 million sequence entries [92] and UniProt had more than 20 million entries [209]. However, the BioBricks Parts Registry [12], the largest synthetic biology parts catalogue developed to date, contained only around 13,000 parts as of March 2012 [13]. This relatively small number limits the complexity of designs [20].

One way to overcome the lack of availability of parts is to use data integration. The large amounts of knowledge stored in databases can be integrated to identify and catalogue biological parts [57]. This integrated data can be used to map parts mined from genomes onto existing knowledge, concerning for example biological processes, molecular functions and enzymatic reactions. Such mapping helps in the understanding how these parts function, and aids in designing genetic circuits with desired behaviours [16, 50].

Bottom-up approaches can be used to create [12, 112, 253] and characterise parts [89]. Top-down approaches are also needed to investigate the dynamics of these parts [28] and the cellular context in which these parts work and interact at a systems level [254]. Such approaches have been deemed crucial for large-scale synthetic biology [28].

However, the lack of available data about biological interactions and relevant parts is currently a bottleneck in constraining solution spaces for genetic circuit design [76].

Interactions between physical elements can include transcription factor (TF)-DNA binding, enzyme-compound specificity, protein-protein and protein-compound interactions [76]. These interactions may occur in specific cellular compartments or pathways [255]. Therefore, knowledge about regulatory systems, biological pathways, genomic information, encoded products such as proteins and RNAs and their interactions must all be integrated to understand the systems as a whole [28, 255, 256].

Even for *B. subtilis*, the model Gram-positive bacterium, interactions between physical elements are not fully understood. Obtaining biochemical parameters for interactions and biochemical reactions has been a challenge so far [257]. However, qualitative models such as graph-based biological networks can capture the relationships between biological concepts and still provide insights into cellular systems [130, 238, 258, 259]. In simple, manual designs interactions between individual parts are specified by a domain expert. For a computationally automated design, this knowledge must be machine-accessible. The use of semantically defined labels to represent the relationships between biological concepts is needed to make these biological networks computationally accessible. In addition, these networks, when combined with quantitative parameters, can be transformed into simulatable models computationally. The creation of these models, such as those based on ordinary differential equations, from computationally amenable networks is discussed in Chapter 5.

The integration of data sources for *B. subtilis*, and the analysis of the resulting integrated data requires new computational tools and approaches. Ondex is a system that combines semantic data integration with graph-based analysis techniques [174]. Therefore, Ondex is designed to model and integrate biological data as networks and to provide tools to visualise and analyse the resulting integrated networks. In these networks, nodes and edges represent the biological concepts, such as genes and gene products, and the relationships between these entities respectively. In Ondex terminology nodes representing entities are termed *concepts* and edges representing relationships are termed *relations*. Ondex provides *parsers* to transform data from databases into the Ondex graph-based data model and allows custom parsers to be developed. Data sources are initially transformed into a pre-defined data model that is backed by an ontology, making the networks machine-accessible. Ondex also provides a framework that allows computational analysis, merge and cleansing. *Mappers, filters,*

transformers, and exporters are used respectively to link biological concepts, clean unwanted data, alter graphs, and export networks in custom formats. Data integration is carried out by XML-based workflows that include calls to parsers, mappers, transformers, filters, and exporters.

The aim of the work described here was to develop a data model for a data warehouse that would act as a knowledge base for *B. subtilis* to support the synthetic biology design process and in particular facilitate automated computational analysis in addition to manual analysis. The result was a knowledge base termed BacillOndex, which was designed to support the rational design of genetic circuits for *B. subtilis*. BacillOndex provides a single repository containing information about a wide range of *B. subtilis* cell biology. The Ondex system was chosen since it provides a convenient framework for data integration, with a range of parsers and a lightweight approach to data integration using a graph-based approach. The use of well defined metadata hierarchies to describe the metadata associated with the graph components provides sufficient semantic richness to support graph-based reasoning with requiring the heavy-weight reasoners for a full OWL-DL type implementation.

The research described here proceeded in two phases. Firstly, a data model for the graph was defined and then data sources were identified and parsers were developed to populate this model with data from these sources. Ondex workflows were defined to facilitate the integration process in the execution of the parsers, data integration and automated data shaping activities.

The following sections describe this semantically defined data model and the workflows constructed for the mapping of data sources to this model. Then an integration workflow is discussed that combines the resulting network representations of data from the transformation workflows. Finally, the BacillOndex dataset, with examples given of biological concepts and relationships, is presented.

3.2 The development of a semantically defined data model for the BacillOndex integrated dataset

Humans bring large amounts of background domain knowledge to the manual analysis of molecular interaction data [260]. This knowledge is often documented in the form of free-flowing text, diagrams, and tables in manuscripts. Automated reasoning over such data sources is challenging and prone to error [261]. Knowledge must be converted into a standard, unambiguous format in order for computers to be able to interpret and reason over semantically-enriched datasets.

BacillOndex is represented as a semantically rich network. This representation allows integrated data from public databases to be machine-accessible in order to guide the understanding and large-scale engineering of the organism. BacillOndex also enables the extraction of parts and analysis of the interactions between parts on a global scale to support large-scale design in synthetic biology. In this network, entities such as sequence features and gene products are represented as concepts. These concepts can be created for any type of entity, and relationships such as `encodes`, `has_function` and `part_of` are represented as relations. Concepts and relations have additional annotations, including semantically defined labels. These labels are drawn from the Ondex ontology, to allow the information to be accessible to machines.

Although the Ondex ontology includes a large number of types of biological concepts, the encoding of proteins and RNAs, and transcriptional relationships can be modelled at the gene level using Ondex. However, biological parts are associated with fine-grained elements of genes, such as promoters and CDSs, in synthetic biology. Therefore, concept types representing sequence features, including promoters, operators, CDSs, ribosome binding sites (RBSs), shims, terminators and operons, were added to the Ondex ontology in order to construct BacillOndex. In addition, the regulatory relationships of proteins, in the form of network motifs such as feed-forward loops (FFLs), can be used to construct biological devices, and hence FFL concepts were included in the metadata. As Gene Ontology (GO) terms are useful to annotate gene products, the Clusters of Orthologous Groups[13] (COG) numbers can also be used to classify proteins and also find orthologous parts from other species. Therefore, concept types for COG numbers and their categories were also included.

The BacillOndex data model was designed to encompass a wide range of concepts from a variety of source databases (Figure 3.2). Biological concepts such as `Protein` and `CDS` are modelled as concept classes as required by the Ondex system (Section 2.5.1.3). Instances of concept classes inherit the relationships defined in the model. For example, the relationships between proteins and CDSs are represented by `is_encoded_by`, while those between TFs and proteins are annotated as `is_equivalent_to`. Proteins, TFs and enzymes can share properties such as names, but are represented by different concepts. For example, the Spo0A protein concept has a phosphorylation protein-protein relationship. Explicit transcriptional regulation information is accessible through the Spo0A TF concept which is linked to the Spo0A

[13] http://www.ncbi.nlm.nih.gov/COG/

protein via the `is_equivalent_to` relation (Figure 3.1). This approach helps to encapsulate role-specific attributes.

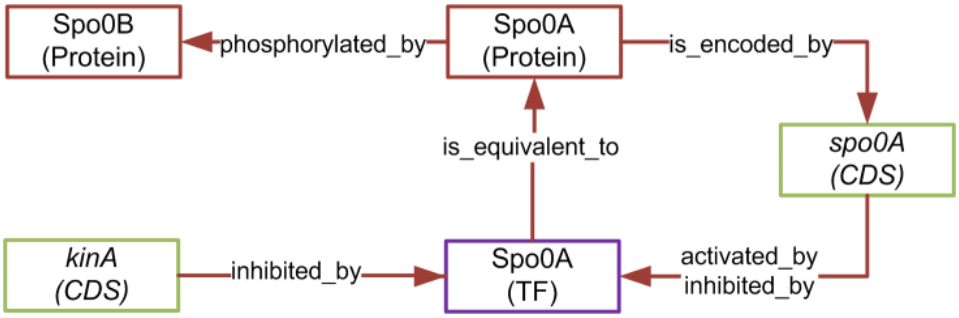


Figure 3.1: A small network of the Spo0A protein. The relationships that model the protein-protein interactions are accessible via the protein concept. The Spo0A TF concept is used for the modelling of the gene regulatory network of Spo0A. The TF inhibits the expression of both *spo0A* and *kinA*, and activates the expression of *spo0A*.

The BacillOndex dataset includes multiple types of concept:

- CDS, promoter, operator, operon, terminator, shim and RBS concepts are derived from sequence-based features.
- Protein, RNA, enzyme, TF and protein complex concepts represent gene products and their aggregates.
- COG class, COG class category, KEGG orthologs enzyme (KOEN), KEGG orthologs gene (KOGE) and KEGG orthologs protein (KOPR) concepts classify other concepts using orthology terms.
- Cellular component, molecular function, biological process and enzyme classification concepts are used for location-, function- and biological process-based classifications.
- Reaction, pathway and compound concepts form the core of the pathways in the network.
- Concepts for FFLs and microarray experiments are also included.

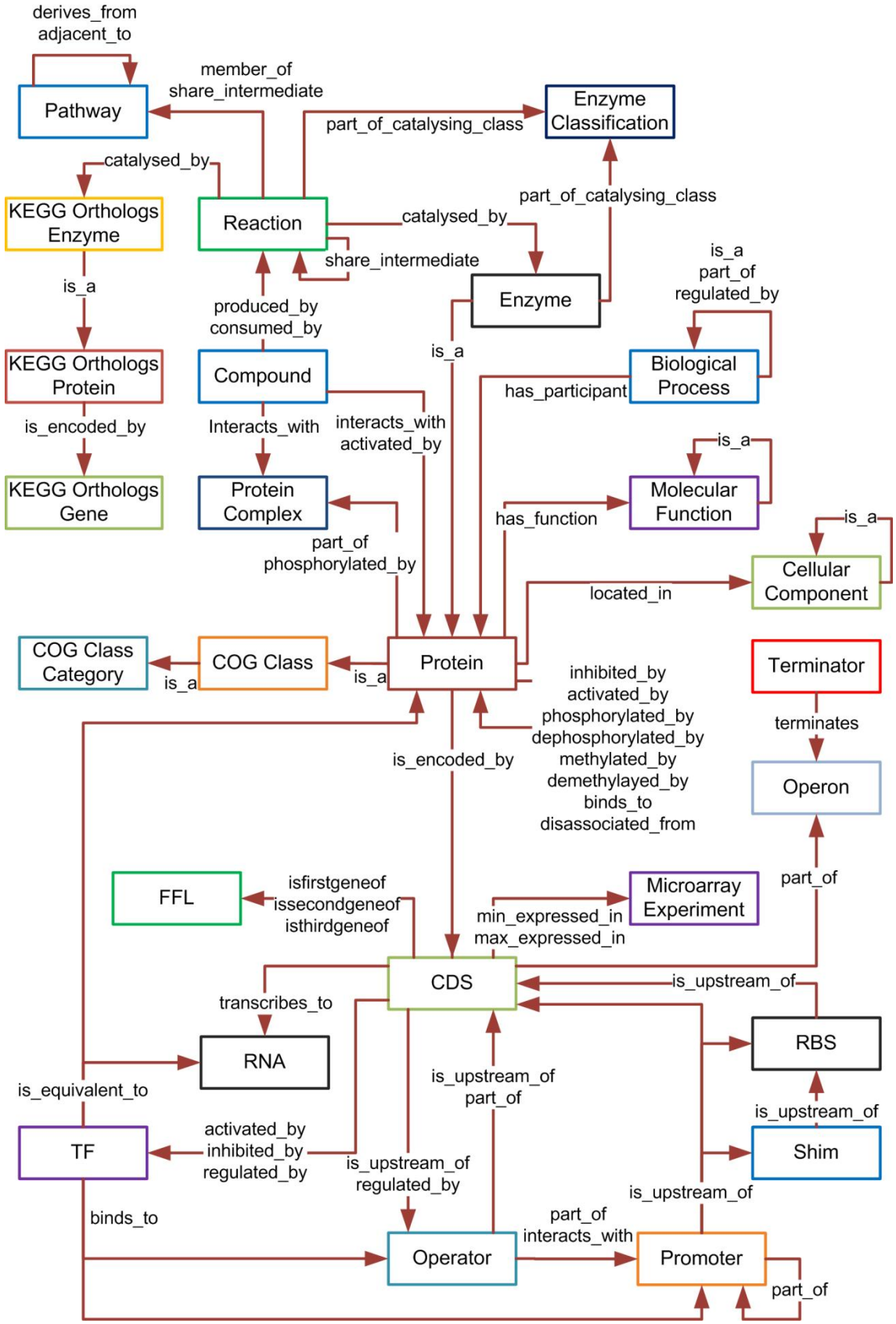


Figure 3.2: Data model for the integrated dataset, showing the concepts and relations between them. Boxes represent the types of biological concepts. For example, `Protein` represents the type of protein concepts in the network. Representing the relations between types of biological concepts, edges are directional and include definitions of the relations. For example, the `is_encoded_by` relation between the Protein and CDS concepts indicates protein and CDS concepts where these proteins are encoded by the

CDSs connected by the edges.

3.3 Identifying data sources

Biological knowledge for the rational design of genetic circuits for *B. subtilis* is provided by different databases. Even genome-specific databases do not provide all of the available information. DBTBS [166] is a database of transcriptional regulators for *B. subtilis* and is a useful resource for identifying relevant binding sites, promoters and sigma factors. The BacilluScope database [124] provides the latest information and sequences about *B. subtilis* genes and gene products. The Kyoto Encyclopedia of Genes and Genomes (KEGG) [169] provides useful information to identify reactions and compounds involved in *B. subtilis* pathways. The STRING [171] database provides possible protein-protein interactions and functional associations. Combining data from these sources may give a better understanding of biological interactions and hence help in building robust genetic circuits. This data integration enables the creation of catalogues of parts and their interactions that occur naturally. Information from multiple databases can be combined to infer new information [147]. For example, the integration of gene expression data and gene regulatory networks can be used to estimate promoter activities, where neither gene expression data nor gene regulatory networks alone are sufficient [57]. In this project, data were integrated from a range of resources including BacilluScope, KEGG, DBTBS, STRING, and GO [204] terms and the associated GO annotations (GOA) [260] for *B. subtilis* (Table 3.1).

Table 3.1: Data sources used to construct BacillOndex.

Source	Data type
BacilluScope [124]	Sequence, annotations
DBTBS [166]	The transcriptional regulatory network
STRING [171]	Physical and functional protein-protein interactions
GO [204]	Terms for cellular components, molecular functions and biological processes
GOA [260]	GO annotations
KEGG [169]	Metabolic pathways
KEGG EXPRESSION [262]	Microarray gene expression

3.3.1 BacilluScope

BacilluScope[14] is a database that contains up-to-date *B. subtilis* records. *B. subtilis* was initially sequenced in 1996 [141, 263]. However, due to large number of research

[14] https://www.genoscope.cns.fr/agc/microscope/mage/viewer.php?S_id=843

groups involved in the project, different areas of the final sequence were of different quality [124]. Several of the genes, that had been sequenced prior to the commencement of the project, were not processed. The *B. subtilis* Consortium therefore re-sequenced and re-annotated the whole genome in 2009 [124].

The sequences, their annotations, and the results of comparative analyses are stored in BacilluScope. Annotations include genome locations, compositional features of CDSs and proteins, product types, biological roles and processes, and updated gene names. For example, 407 genes whose names start with 'y', indicating unknown function, were renamed based upon experimental evidence from the literature. In addition, proteins were clustered using the Clusters of Orthologous Groups (COG) numbers and the functional categories of these numbers [264]. The COG numbers are used to assign orthology relationships, and are useful in transferring interactions between organisms [265]. For example, an association between a protein pair is assumed to exist between orthologous protein pairs, identified by COG numbers, in different species. Each COG number is assigned to a functional category using a one letter code. The COGs database lists 5665 COG numbers and 25 functional categories[15].

3.3.2 DBTBS

DBTBS[16] is a reference database for information about transcriptional regulation in *B. subtilis* [166]. The database can be used to infer the gene regulatory networks of *B. subtilis* at a detailed level. The database is manually-curated with experimentally-validated information about TFs, their binding sequences and motifs, and information about the genes they regulate. Promoter, binding site and terminator sequences are provided, with references to the literature. Binding sites are linked to individual promoters to indicate how they may affect promoter expression, and promoters are annotated with their sigma factors. In addition, the database includes operon membership predictions.

3.3.3 STRING

The STRING[17] database is an integrated database that includes known and putative protein-protein interactions such as binding, gene fusion, co-occurrence and co-expression [171]. In addition to physical protein interactions, information about indirect functional interactions such as genetic interactions and co-citation, which can also

[15] http://www.ncbi.nlm.nih.gov/COG/ (accessed 01/03/2012)
[16] http://dbtbs.hgc.jp/
[17] http://string.embl.de/

provide insights into cellular systems, is stored [171]. Examples of interactions from the STRING database include:

- Co-expression: Genes that show similar gene expression patterns across a variety of conditions are considered to be functionally related [266].
- Co-occurrence: The patterns of absence and presence of genes across different species are used to infer functional relationships [267].
- Gene fusion: Genes may encode for a single fusion protein. Proteins which are fused in one organism, but not in others, may have a functional linkage [266].
- Neighbourhood: Genes that are located close together in prokaryotic genomes may have a functional linkage [266].
- Text mining: Most biological information is still stored as text [174]. Text mining methods can be used to infer relationships between genes and gene products [206]. For example, the co-occurrence of gene names in paper abstracts can be indicative of functional relationships [171].

The data are integrated from high-throughput experimental datasets and from databases such as MINT [268], Reactome [269], BioGRID [270], and KEGG [169]. Orthology information imported from the COGs database [264] is used to infer the existence of protein-protein associations between organisms [265]. Text-mining techniques are used to infer functional interactions from PubMed abstracts by searching for co-occurring gene names [171]. In addition, systematic genome comparisons are used to infer gene fusion, neighbourhood and co-occurrence associations [171]. The co-expression of genes is inferred from diverse microarray experimental data and transferred across species [265]. STRING interactions are weighted against a common reference set [265].

3.3.4 The Kyoto Encyclopedia of Genes and Genomes

The Kyoto Encyclopedia of Genes and Genomes (KEGG)[18] is a large database which provides data about pathways and biochemical reactions. The system consists of different databases such as KEGG PATHWAY, KEGG ORTHOLOGY, KEGG GENES, KEGG LIGAND and KEGG EXPRESSION [169]. These databases include information about catalytic enzymes, product and substrate compounds, and physical protein-protein interactions, including phosphorylation, methylation and protein complex formation. In addition, the KEGG EXPRESSION[19] database stores

[18] http://www.genome.jp/kegg/
[19] http://www.genome.jp/kegg/expression/

information about genes and their expression values from microarray gene expression profiles [262].

The KEGG database stores information about various different organisms. Genes are identified by their locus tags and Enzyme Commission (EC) [271] numbers are used to refer to enzymes. To identify orthologous genes, KEGG uses the KEGG Orthology (KO) identifiers which are constructed from COG numbers and manual classifications [169]. These IDs identify orthologous genes that are involved in the same pathway.

3.3.5 The Gene Ontology Annotation database

The GO has been developed in order to standardise the annotation of proteins. The assignments include information about the cellular locations, biological processes and molecular functions of proteins. Accordingly, proteins stored in the UniProt [209] database have been annotated with standard GO terms, both manually and computationally [204]. These annotations are stored in the Gene Ontology Annotation (GOA) database[20] [260, 272]. The associations between proteins and GO terms are available as tab-delimited gene association files. Entries in these files include cross-references to databases such as UniProt, InterPro [218] and the Enzyme nomenclature database [271].

The data model defined in Section 3.2 semantically describes the relationships between various biological concepts. Therefore, data from these identified sources can be mapped to this model and semantically equivalent concepts can then be merged using computational approaches. Workflows are suitable for automating this data integration process.

3.4 Data integration

Biological data from these databases, such as that concerning protein-protein interactions and metabolic pathways, can readily be represented as networks. However, each data source may provide different types and detail of biological information. For computational approaches, the mapping of biological concepts followed by data cleansing and integration operations must be explicitly defined. Workflows are useful to define the analysis of biological datasets in a number of operational steps [180] and can also be used to automate the process of integrating biological networks.

In this project data integration was facilitated by workflows using Ondex. Data from each source were retrieved and mapped into the data model described in Section 3.2,

20 http://www.ebi.ac.uk/GOA/

producing network representations of data. These networks were then integrated. The workflows include calls to custom parsers and transformers, available in Ondex plugins. New parsers were developed for the BacilluScope, DBTBS, STRING and KEGG EXPRESSION databases (Appendix A), using the Java programming language. Existing Ondex parsers were used to convert data from the GO terms and annotations databases and from KEGG. When the same concept was retrieved from several different databases, the concepts were merged using Ondex's existing name- and accession-based mappers.

3.4.1 Data transformation

The first step in the integration of the data sources is to convert each data source into the Ondex data format, OXL, which is an XML representation of concepts and relations. Ondex uses networks in order to exchange information between workflow steps. In each step biological concepts from different networks can be mapped or merged to produce integrated views of the data. However, in order to map two concepts from different networks, these concepts must be defined so as to be semantically the same. Therefore, before integration, the data from each source were initially mapped to the model described in Section 3.2 and converted into the OXL format using Ondex's OXL exporter. OXL files were produced for BacilluScope, DBTBS, STRING, GO terms, GO annotations, KEGG and KEGG EXPRESSION data. Semantically equivalent concepts of networks included in these files were then merged to produce the integrated dataset (Figure 3.3). Table 3.2 shows the list of concept classes, the number of concept instances for each concept class and the data sources from which these classes were derived.

Table 3.2: The list of concept classes, the number of their instances in BacillOndex and the data sources from which the classes are derived. Some of the concepts were derived computationally. For example, RBS and shim concepts were harvested from the genome, FFLs were searched for using the graph analysis of the network, and gene expression values from the microarray experiments were normalised computationally.

Concept class	Number of instances	Data source
CDS	4,516	BacilluScope, KEGG, DBTBS, STRING, GOA
Promoter	955	DBTBS, computational
Operator	772	DBTBS
Operon	1,131	DBTBS
RBS	458	Computational
Shim	287	Computational
Terminator	1,117	DBTBS
Protein	4,599	BacilluScope, KEGG, DBTBS, STRING, GOA
RNA	179	BacilluScope
TF	369	BacilluScope, DBTBS, STRING
Enzyme	690	KEGG
Protein Complex	19	KEGG
Pathway	270	KEGG
Reaction	3,963	KEGG
Compound	2,817	KEGG
EC	2,300	KEGG
KOEN	1,533	KEGG
KOPR	1,533	KEGG
KOGE	1,533	KEGG
Cellular component	87	GO, GOA
Biological Process	559	GO, GOA
Molecular Function	974	GO, GOA
COG Class	1,741	BacilluScope
COG Class Category	23	BacilluScope
FFL	539	Computational
Microarray Experiment	79	KEGG EXPRESSION, computational

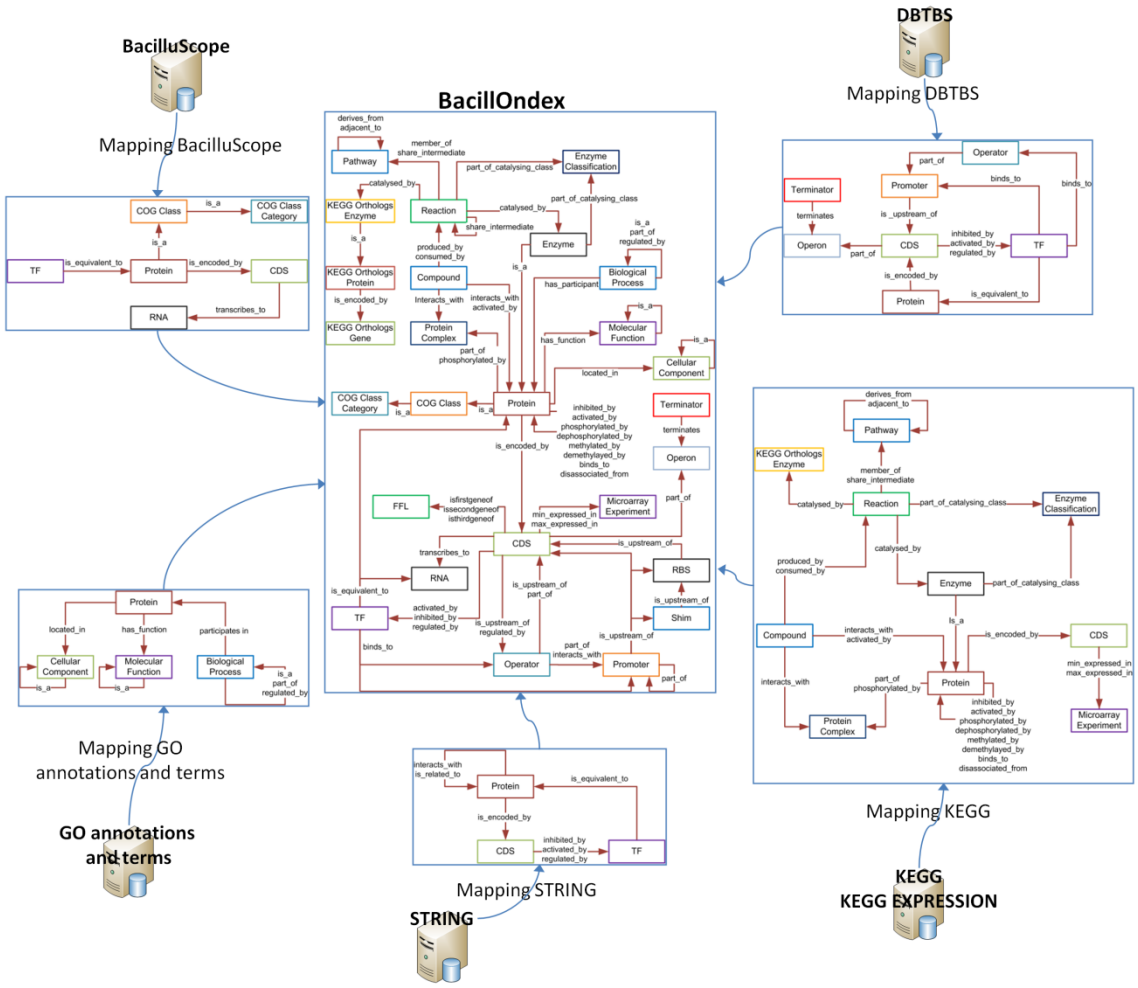


Figure 3.3: Transformation and integration of data to construct the integrated BacillOndex dataset. Data from BacilluScope, DBTBS, STRING, GO annotations and terms, KEGG and KEGG EXPRESSION were mapped to the BacillOndex data model and integrated using workflows.

The data transformation process for each data source used to construct BacillOndex is discussed below.

3.4.1.1 Mapping BacilluScope to BacillOndex

BacilluScope has up-to-date records from the latest sequencing of the *B. subtilis* genome, including annotations about gene products, updated names and genome positions for CDSs, and information about new open reading frames. This information is available as text files. The BacilluScope workflow takes the 'Tab Delimited' and 'COG automatic classification' files from BacilluScope and converts data into the Ondex format. Information from BacilluScope was used to create the `CDS`, `RNA`, `Protein`, `TF`, `COG Class` and `COG Class Category` concepts (Figure 3.4). CDS and protein compositional features such as GC content and molecular weight were added as concept attributes for CDSs and proteins respectively. Protein concepts were annotated with information concerning their description, type, localisation, role, biological process

and product type. TF concepts were created for proteins with a 'Transcription regulation' entry from the tab-delimited data file. In addition, RNA concepts for rRNAs, tRNAs and other miscellaneous RNAs were included.

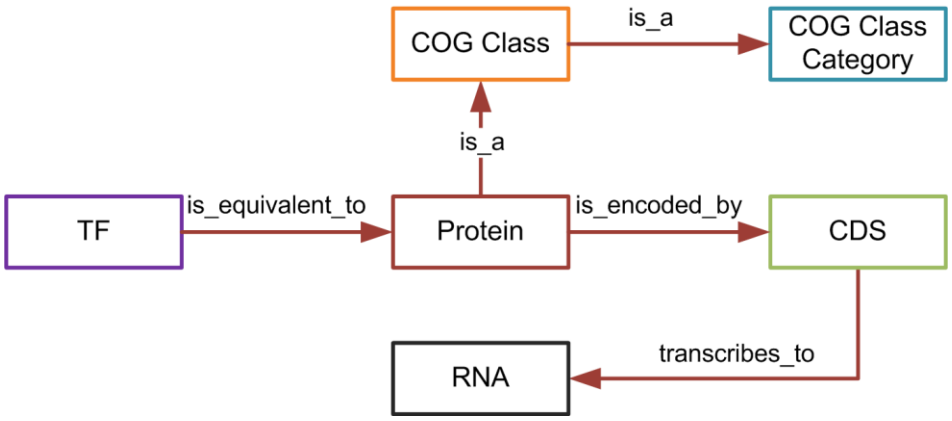


Figure 3.4: The BacilluScope data model in BacillOndex. CDSs and gene products are annotated with information from BacilluScope. This model is a subset of the semantically defined data model described in Section 3.2.

For each COG cluster, a COG class concept was created. The single letter COG class category, and its description and process definition were stored as `COG Class Category` concepts. The weight score that determines the likelihood that a protein belongs to a COG class was stored as an attribute of the relationships between the `Protein` and `COG Class` concepts.

Finally, synonyms for genes were eliminated in the workflow by comparing each synonym with the synonyms of other genes. If a synonym is used as a preferred name for another gene, the synonym was discarded, and if used as a synonym for another gene, the latter was ignored.

The amino acid and nucleotide sequences of CDSs that encode proteins and RNAs were downloaded as FASTA files from BacilluScope. A workflow was created to convert each of these FASTA files to OXL files. These workflows use Ondex's FASTA parser and serialise the output using the OXL exporter to create CDS and Protein concepts with nucleotide and amino acid sequences as their attributes.

3.4.1.2 Mapping DBTBS to BacillOndex

Data from DBTBS were used to extract information about the gene regulatory networks of *B. subtilis*. After parsing the file from DBTBS using the *Dbtbs* parser, the DBTBS workflow performs a name-based mapping to link synonymous CDS concepts. Synonyms that are used as preferred names for other CDSs are removed from the relevant CDS concepts. CDSs that cannot be mapped to any concepts in BacillOndex

were manually provided with additional synonyms as input to the workflow (Section A.2).

The concepts created from DBTBS included `TF`, `Promoter`, `CDS`, `Protein`, `Operator`, `Operon` and `Terminator` (Figure 3.5). In the modelling of biological concepts from DBTBS, TFs bind to operators, which are part of promoters, CDSs are part of operons and terminators terminate operons. Relations were created between TFs and CDSs to indicate the type of regulation: either activation or inhibition. Sequence annotations were stored as attributes, as were nucleotide sequences in sequence-based features such as promoters, operators and terminators.

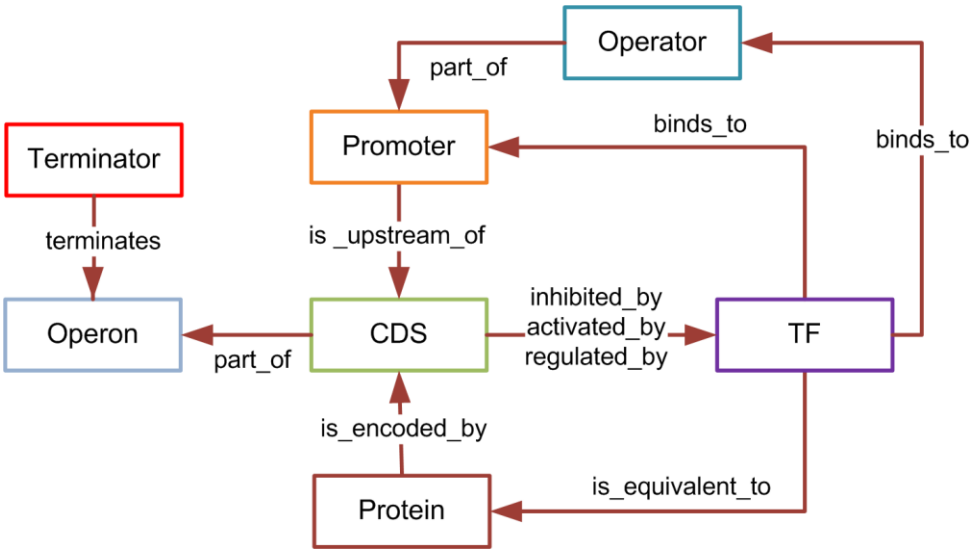


Figure 3.5: Part of the the DBTBS data model in BacillOndex showing a detailed view of the representations of gene regulatory networks of *B. subtilis*, including TFs, their binding sites and the CDSs they regulate.

Sigma factors can also be considered as biological parts in order to programme the expression of genes [153]. Although the DNA binding of TFs affects the efficiency of transcription, the initiation of transcription is largely determined by sigma factors, which function as transient subunits of RNA polymerases (RNAPs) [119]. When bound to sigma factors, RNAPs can initiate the expression from promoters recognised by the corresponding sigma factors which are also represented as TFs binding to promoters in the BacillOndex data model. In addition, some tRNAs have antitermination effect, enabling the full synthesis of the transcripts [273]. For example, TrnJ-Gly controls the expression of the *glyQ* gene and is therefore represented as a TF.

3.4.1.3 Mapping STRING to BacillOndex

The STRING workflow uses the *String* parser to represent protein-protein relationships

from the STRING database as Ondex concepts and relations. An association between two proteins is termed a *link* in STRING. Links can include scores for indirect functional associations such as neighbourhood, gene fusion, co-occurrence, co-expression and text-mining. In addition, STRING computes a 'combined' score for each link using these scores. Furthermore, the interaction types for direct protein-protein interactions such as post-translational modification and gene expression are also available and are termed *actions*. These protein actions and links were downloaded from the STRING website as tab-delimited text files, and records specific to *B. subtilis* were extracted prior to the integration process. Only actions for protein-protein binding, protein post-translational modification, expression and activation were incorporated into BacillOndex.

The STRING database, at the time of downloading[21], contained 463,060 protein links, which were filtered to extract those with high scores. The thresholds used for gene fusion, co-expression, co-occurrence and combined scores were 400, 500, 700 and 900 respectively (Figure 3.6). These thresholds were chosen in order to select the top 5% relationships with the highest scores (Table 3.3). The data from STRING contained 6,070 gene fusion links, 870 of which were over the threshold score of 400 and were imported into BacillOndex. Out of 199,757 co-occurrence links, 9,868 links with a threshold of 500 co-occurrence score, and out of 8,700 co-expression links, 1,012 links with a threshold of 700 co-expression score were selected. In addition, 8,480 links with combined scores over the threshold of 900 were incorporated into BacillOndex.

A STRING protein link may have more than one type of functional association. For example, the KinC and Spo0F protein pair has a protein link with a co-occurrence score of 739 and a combined score of 929. As a result, 20,230 selected links specific to *B. subtilis* could be mapped to 17,390 unique links. The STRING database contains two records for each relationship; one in each direction. In the workflow these duplicate relationships were removed. Therefore, 8,695 unique protein links were imported from the STRING database.

[21] Version 8.2 (accessed 08/01/2010)

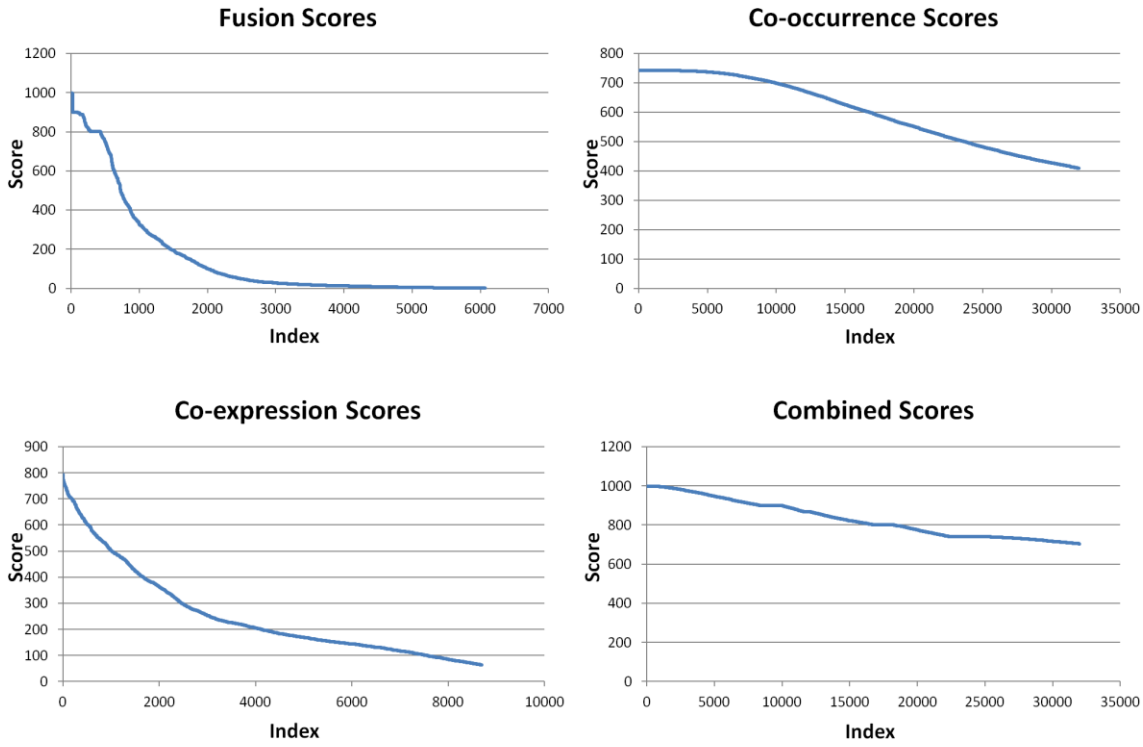


Figure 3.6: Gene fusion, co-expression, co-occurrence and combined scores sorted from the highest to the lowest from the STRING database. The X axis shows the scores and the Y axis shows the index of a score. The highest 32,000 co-expression, co-occurrence and combined scores, and the highest 6,070 fusion scores are shown. 870 links with a fusion score of 400 or more, 9,868 links with a co-occurrence score of 700 or more, 1,012 links with a co-expression score of 500 or more, and 8480 links with a combined score of 900 or more were chosen for the BacillOndex dataset. In total, 8,695 unique relations were created from 17,390 paired links.

Table 3.3: Filtering of protein links from STRING.

Protein link	Threshold	Included links	Total links
Fusion	400	870	6,070
Co-occurrence	700	9,868	199,757
Co-expression	500	1,012	8,700
Combined	900	8,480	463,060

The concepts created from the STRING database include `CDS`, `Protein` and `TF` (Figure 3.7). The regulation of the transcriptional expression of a protein by another protein is represented as a protein action, with the type of 'Expression'. This relationship was used to create a CDS concept for the former protein and a TF concept for the latter. An `inhibited_by` or `activated_by` relation was created between a CDS and a TF concept if the action is annotated as 'inhibition' or 'activation' respectively. Otherwise the default `regulated_by` relation was used.

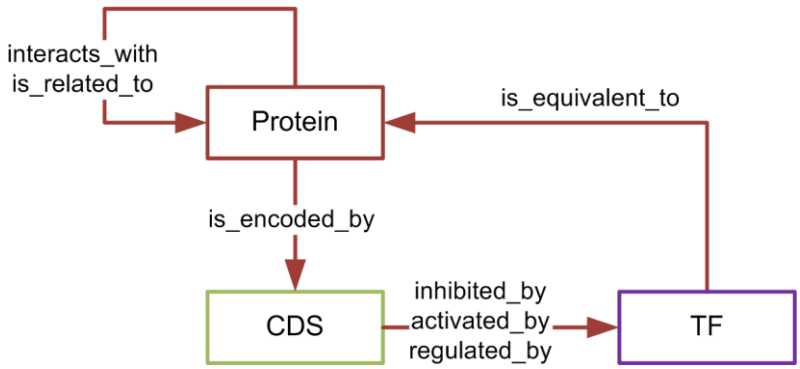


Figure 3.7: The STRING data model in BacillOndex. The model includes direct and indirect protein-protein associations, and the gene regulatory network.

Other protein actions and links in STRING are represented by the `interacts_with` and `is_related_to` relations in BacillOndex respectively. The type of protein action, such as 'ptmod' or 'binding', was recorded as an attribute of the `interacts_with` relation. The score for these type of protein actions was recorded as the attribute value for the corresponding BacillOndex relation. Protein link scores that include neighbourhood, fusion, co-occurence, co-expression, experimental, database, text mining, and combined score were stored as attribute values of the corresponding protein link attribute of the `is_related_to` relations.

3.4.1.4 Mapping the Gene Ontology database to BacillOndex

GO annotations specific to *B. subtilis* were extracted from the entire set of UniProt GO annotations[22] and were converted using Ondex's existing GO annotation parser into the Ondex format in a workflow. Concepts created for the GO annotations include `Cellular_Component`, `Molecular_Function`, `Biological_Process` and `Protein`. Proteins were assigned with `located_in`, `has_function` and `has_participant` relations between the protein concepts and the GO annotation concepts (Figure 3.8).

GO annotations are associated with proteins using locus tags or textual protein names. The preferred names by which the proteins are referred to, and synonyms including the locus tags from the GO annotations, are modelled as the Ondex terms' preferred names and synonyms in the workflow. Using protein names to identify annotations requires that each step in integrating the data from other sources with the GO annotations must be carried out based on names. To simplify the process, names which are not preferred names or locus tags were removed from the data produced from

[22] ftp://ftp.ebi.ac.uk/pub/databases/GO/goa/UNIPROT/gene_association.goa_uniprot.gz (accessed 02/10/2010)

GO annotations. Each GO annotation for a protein is stored in one line in a text-based GOA UniProt file. This structure means that there is more than one line per protein, and hence more than one concept needed to be created for each protein. To produce only one concept per protein, concepts were merged using name-based mapping. GO annotations only include the GO terms' IDs, and hence concepts do not include any GO term label.

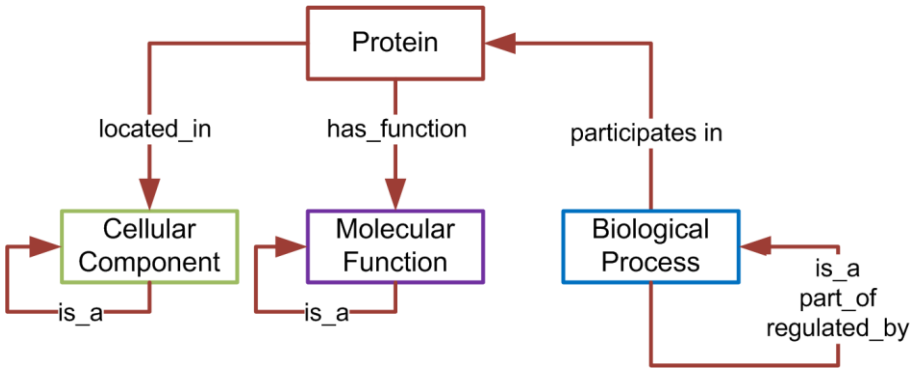


Figure 3.8: The data model for GO annotations and terms. Proteins are annotated with their cellular locations, molecular functions and biological processes. GO terms are represented as a hierarchy of parent and child relationships.

GO terms were thus integrated with the GO annotations. Initially, GO terms were downloaded in OBO format and converted into the Ondex format using Ondex's GO parser. This process results in `Cellular_Component`, `Molecular_Function` and `Biological_Process` concepts being created. GO terms are structured in a hierarchy where child terms are linked to the parent terms with the `is_a` relationship. These relationships were modelled in BacillOndex with the `is_a` relation between the GO concepts.

Another workflow was used to integrate GO annotations and GO terms. Since GO related concepts all have GO term IDs as their accession identifiers, the concepts were mapped to each other using accession-based mapping and merged using the relation collapser. However, the number of GO terms is high making the network hard to browse. To simplify the network, unconnected nodes were filtered out, removing GO terms that are not linked to any other concept. The preferred names of the GO annotation concepts were updated with human-readable GO term labels.

3.4.1.5 Mapping KEGG to BacillOndex

KEGG was used to incorporate information about metabolic pathways, reactions, compounds and enzymes into BacillOndex. Ondex's KEGG parser requires that content

from the KEGG GENES, PATHWAY, BRITE and LIGAND databases is downloaded as archived files to a local computer. However, these files contain gigabytes of data for the entire database content for all organisms from KEGG. The Ondex website already contains OXL files produced from the KEGG database for several organisms, including *B. subtilis*, which were created by running the KEGG parser against the KEGG archive files. Therefore, rather than re-executing the KEGG parser to transform data from KEGG into the Ondex format, the OXL file for *B. subtilis* was downloaded from the Ondex website[23]. Amino acid and nucleotide sequences were removed from the concepts and replaced with the latest sequences from BacilluScope in the final dataset. Locus tags were converted to accession IDs for CDS and protein concepts using the `Name To Accession Converter` transformer.

Concepts from KEGG include `CDS`, `Protein`, `Protein complex`, `Pathway`, `Enzyme`, `Compound`, `Reaction`, `Enzyme Classification`, `KEGG Orthologs Enzyme`, `KEGG Orthologs Protein` and `KEGG Ortholog CDS` (Figure 3.9). The model from KEGG includes the basic components of metabolic pathways and their relationships. For example, reactions are members of pathways. Compounds are produced or consumed by reactions and may interact with protein complexes and proteins. Proteins may also interact with other proteins or protein complexes. Enzymes are part of catalysing Enzyme Classification classes, which have Enzyme Commission (EC) [271] numbers specifying the chemical reactions catalysed. Thus enzymes that catalyze the same reaction can be referred to using the same EC number [274]. Reactions are therefore also part of these classifications.

Pathways often interact [275] with each other, forming a network. For example, a compound produced in one pathway can be consumed by reactions in other pathways. These links between pathways are modelled with the `adjacent_to` relation in BacillOndex. KEGG uses reference pathways that are identified by KEGG orthology numbers to construct organism-specific pathways [276]. In the dataset, the link between a pathway specific to *B. subtilis* and a reference KEGG pathway is modelled via the `derived_from` relation. Information derived from KEGG was linked to the gene regulatory networks of *B. subtilis* in BacillOndex using concepts such as CDSs and proteins, allowing the extraction of information to optimise these pathways via gene expression.

[23] http://www.ondex.org/doc.html

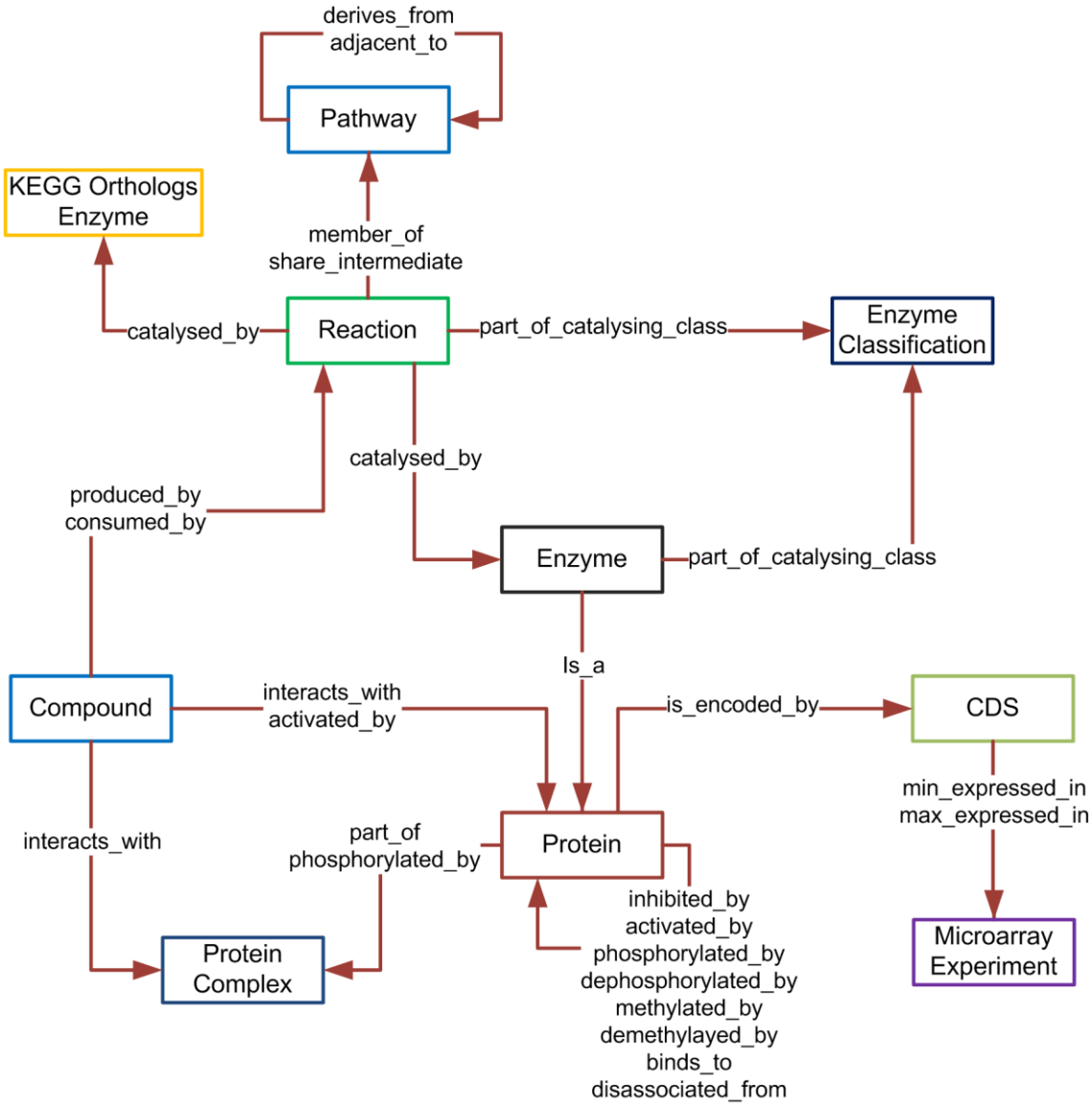


Figure 3.9: The data model for the KEGG database, modelling the network of *B. subtilis* pathways and their components, such as reactions and compounds. Gene expression data are linked to their genomic context.

Promoter parts are used to control the levels of gene expression in genetic circuits, and can be specified with minimum and maximum strengths in order to create predictable circuits [71]. Promoters from DBTBS can also be used as biological parts and their strength can be identified from microarray experiments. Therefore, *B. subtilis* microarray data from the KEGG EXPRESSION database was used to find the relative minimum and maximum expression values of genes across different experiments. These values were recorded as attributes on the CDS concepts, which were then linked to microarray experiment concepts. The expression values were normalised according to the algorithm developed by Dawes and Glassey [277].

The algorithm identifies a self-consistent set of genes (SCS) by comparing the

rankings of each gene's contribution to total expression in every experiment. The genes whose rankings do not change by more than a specified amount across all experiments are identified as the SCS. Genes with consistently high or low expression values across all experiments may introduce a bias [277]. Therefore, the algorithm excludes a specified percentage of the genes that have the lowest and highest gene expression values across all experiments. By using 10 as the percentage of genes to exclude and 1,000 as the threshold, 11 CDSs were identified as the SCS. These CDSs were then used to normalise expression values across experiments.

Controls and targets are considered as separate experiments when applying the algorithm, and hence the microarray dataset contained 158 columns, from 79 experiments. Background values were extracted from both control and signal values prior to normalisation. The number of rows is equal to the number of genes in the dataset, and each gene's contribution to total expression for a single experiment was calculated by dividing each value by the column total. The average contribution of each CDS was calculated by first adding all the contributions for that CDS and then averaging. A subset of the contribution matrix was extracted by excluding the previously specified CDSs with the least and most contributions, and the final matrix was used to identify the SCS CDSs (Table 3.4). The CDSs whose rankings among control and target values do not change more than the threshold between experiments were selected as SCS CDSs. The algorithm was iterated until there was no change in the SCS between iterations. The CDSs in the SCS list were used to normalise the data by dividing each value in the original data by the summed values of the SCS CDSs. Since the resulting values were small, each value was multiplied by 1000 to improve readability. Minimum and maximum values for each CDS across all experiments were identified after normalisation.

The concepts created from the KEGG EXPRESSION database are `CDS` and `MicroarrayExperiment` (Figure 3.9). The normalised and non-normalised values for the minimum and maximum expression levels were stored as attributes of the CDS concepts. CDS concepts are connected to the experiments in which the CDSs are expressed at minimum or maximum level by `min_expressed_in` and `max_expressed_in` relations respectively. The experiments are annotated with the accession, PMID, description of the experiment and details of the control and target experiments. Each control and target experiment annotation includes information about the strain, medium, temperature, and concentration of the bacteria.

Table 3.4: SCS CDSs identified across the microarray experiments. Each expression value in a single experiment was divided by the total expression values of these CDSs from the experiment to normalise the data.

Locus tag	Name
BSU15650	*yloB*
BSU00930	*cysE*
BSU32720	*yurZ*
BSU28350	*ysnB*
BSU10200	*yhfE*
BSU28640	*pheS*
BSU21730	*ypmS*
BSU40970	*parA*
BSU16040	*rplS*
BSU40500	*rplI*
BSU28360	*ysnA*

The gene expression file[24] that contains expression data for *B. subtilis* and other species was downloaded from the KEGG EXPRESSION[25] database. However, there are additional experimental data that are not provided as part of this file, and these were downloaded individually[26] (Appendix B) using the website since the downloadable files in the FTP site do not include headers for the experiments. For example, to gain both the header and the values for experiment ex0000258, data from the corresponding KEGG URL[27] was saved as a single text file. All the files were saved to the directory specified by the parser's 'KeggExpressionFolder' parameter. The mapping file for the locus tags and ORF IDs was also downloaded from KEGG[28]. After the data had been normalised and converted into the Ondex format, accession IDs were created for CDS and protein concepts from the locus tags using the `Name To Accession Converter` transformer.

3.4.2 Data integration

To integrate all of the data sources, an integration workflow was constructed. This workflow defines the operational steps in order to automate the integration of data to construct the BacillOndex dataset. Each step in the workflow takes two data sources as input. The input data sources were mapped based on accession identifiers. For example, the *spo0A* CDS concept from BacilluScope is mapped to the *spo0A* CDS concept from STRING using the BSU24220 locus tag. Name-based mapping was used when

[24] ftp://ftp.genome.jp/pub/db/community/expression/expression
[25] ftp://ftp.genome.jp/pub/db/community/expression/
[26] http://www.genome.jp/kegg-bin/get_htext?htext=Exp_DB&hier=1
[27] http://www.genome.jp/dbget-bin/www_bget?ex:ex0000258+withDATA
[28] ftp://ftp.genome.jp/pub/kegg/genes/organisms/bsu/bsu_subtilist-bsu.list

accession identifiers were not present. DBTBS uses human-readable labels for CDSs and proteins. Therefore, the data from DBTBS can only be mapped by using name-based mapping. The mapping process creates a relationship between the mapped concepts using the `is_equivalent_to` relation type. Once the mapping is complete, concepts linked by the `is_equivalent_to` relation are merged using the relation collapser transformer. The new concepts inherit all of the attributes and relationships from their parents. The new concepts' preferred names were set to the human-readable labels from BacilluScope for CDSs and gene products.

BacilluScope concepts were annotated with sequence information (Figure 3.10). Protein and RNA-encoding CDSs were annotated with their nucleotide sequences, and proteins were further annotated with amino acid sequences. The output was then integrated with DBTBS, STRING, GO annotations and terms, and KEGG and KEGG EXPRESSION data.

The resulting network includes extensive information about molecular interactions, some of which may form network motifs that can be used to achieve a desired behaviour based on the connectivity of biological molecules [191]. Therefore, a network motif search was applied to the network. In addition, new promoter concepts were added by creating composite promoters from the core promoter regions and their surrounding operators. RBS and shim concepts were extracted and added to the network. Finally, all sequence-based features were annotated with genome positions.

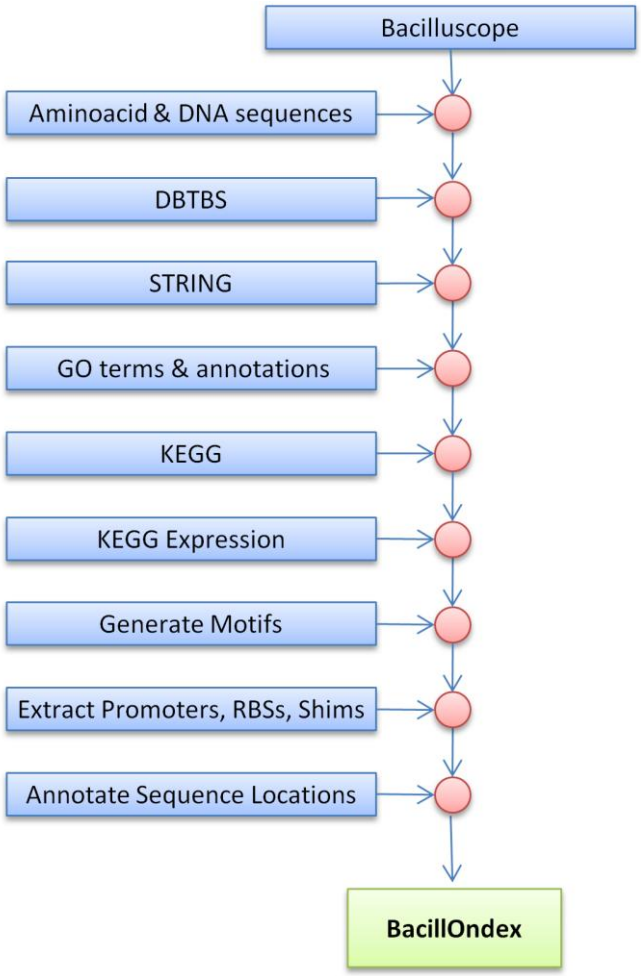


Figure 3.10: Data integration workflow. Rectangles represent the data sources converted into the Ondex format by the workflows. Each circle shows a workflow step, which includes the integration of its two input sources. In the final steps, motifs are generated, positions of sequence-based features on the genome are annotated, and composite promoters, RBSs and shims are extracted from the integrated data.

3.4.2.1 Extracting network motifs

In transcriptional regulatory networks, the most widely recurring network motifs are positive and negative autoregulatory genes and FFLs [195]. These motifs show distinct behavioural features such as bistability or signal filtering, and can therefore be used as modular building blocks for the design of biological systems. In order to identify these motifs from the network, a workflow step that calls the *Network Motif Generator* transformer was executed. This transformer was also developed as part of this project using the Java programming language. Although the network motif search was efficiently carried out using BacillOndex, there might be performance problems for larger networks. This probability would require further research.

Using the transcriptional regulatory network, positive and negative autoregulatory CDSs and FFLs were identified. The fact that TFs regulate operons rather than individual CDSs was taken into account when finding the motifs. For example, Spo0A

does not directly regulate the expression of SinR. Instead, Spo0A binds to the *sinI* promoter. Because both SinR and SinI are translated from the same transcript, Spo0A also regulates the expression of SinR. Using graph analysis, FFL concepts for eight different FFL types (Figure 2.8) were created, and the participating CDS concepts were linked to them. These relationships explicitly show the functional order of the genes as the first, second and third gene in FFLs in which the first TF regulates the second TF, and with the second TF, regulates a target gene [197]. The network motif types in which CDSs participate, and information about whether these CDSs are positively or negatively autoregulatory, were recorded as attributes on the CDS concepts.

3.4.2.2 Annotating genome positions

The genome positions of biological parts are important in demonstrating their provenance. Such information is also useful to annotate biological parts for other sequence features that are included. In addition, sequence features that are close in the genome can be combined to create biological devices. For example, promoters and their upstream and downstream operators are candidates to create composite promoters.

Information about CDSs in BacillOndex includes their start and end positions from the BacilluScope database. Information about the genome positions of other sequence-based features, such as promoters and operators also exists, but comes from DBTBS which was constructed before the *B. subtilis* genome was re-sequenced and re-annotated. Therefore, the positional information for sequence-based features from DBTBS was re-calculated using information from BacilluScope.

The start and end locations of sequence-based features such as CDSs, promoters, terminators and operators were added using the *Sequence Location Annotater* transformer. The locations of sequence feature entities such as promoters, terminators and operators were annotated using the locations of the CDSs as reference. For each CDS, promoters that are upstream of the CDS and operators related to the promoters were searched for, both upstream and downstream of the CDS and in the CDS itself. The terminator sequences were searched for among all genes that are part of operons terminated by these terminators. The number of upstream and downstream base pairs used for each search varied, starting from 500 bp to 2000 bp, 8000 bp and 32000 bp, until the sequence was found. The default, reverse, complementary, and reverse and complementary sequences were searched. Unless the strand information is specified, start and end positions also specify the start of a sequence feature. For example, if the start is smaller than the end, then the feature is on the reverse strand.

3.4.2.3 Extracting composite promoters

Promoters with known TF binding sites are needed to engineer genetic circuits at the transcriptional level [27, 37, 58, 90]. Early applications of synthetic biology, such as the repressilator [107] and the bistable switch [108], used inducible and repressible promoters to rewire gene regulatory networks. To increase the diversity of available promoters, TF binding sites were combined experimentally with sigma factor-dependent promoters to create combinatorially regulated composite promoters [70, 71, 240, 278]. These composite promoters have been used to create new devices such as feedback loops, FFLs, logic gates and switches [37]. The relative locations of operators within the promoters [70] and the spacer sequences used between operators and core promoters [106] may change the rate of expression for these composite promoters. A library of combinatorial promoters from the *B. subtilis* genome with known regulatory strengths would be a valuable tool for synthetic biologists using this organism.

The core promoters from BacillOndex were combined with upstream, downstream and overlapping operators to create a combinatorial promoter library. BacillOndex integrates only the core sequences of promoters to which RNA polymerases bind, and the binding sequences of TFs. Only a few of these core promoters include binding sites. Other core promoters are suitable for the control of sigma factor-dependent expression. With the addition of information about the genome positions of promoters and operators, sequences that represent inducible or repressible promoters were extracted from the *B. subtilis* genome using the `Sequence Relation Updater` transformer.

Initially, the binding sites were linked to promoters with the `part_of` relation. However, not all binding sites are part of promoter regions. Although operators are sometimes included in core promoter sequences, this is not usually the case. For example, the protein RocR binds to a terminator sequence at the end of the *rocG* CDS. Extracting the *rocG* promoter using this relation would result in retrieving the whole sequence from the start of the promoter to the end of the binding site, including the *rocG* CDS. To correctly capture relationships between promoters and operators, the relationships were annotated as `interacts_with`. The `part_of` relations were reserved for cases where the full operator sequences are included in the core promoter sequences. The `part_of` relations between promoters and operators that are not upstream of the CDSs were deleted. Instead, operators that are in the middle of CDSs were linked to CDSs with the `part_of` relation. CDSs that are upstream of operators were linked to operators with the `is_upstream_of` relation.

Regions containing core promoters and all of their associated operators were

extracted as potential new promoters. Only the core promoters whose sequence positions and their operators' sequence positions are identified were used. The new promoters were annotated with their start and end positions. Core promoters and the operators that are part of the extracted promoters were linked to the new promoters with the `part_of` relation. In cases where an operator is also part of a core promoter, the operator was also linked to the core promoter with the `part_of` relation.

Figure 3.11 shows an example of the creation of a composite promoter. The 'AbrB-P2' promoter includes a binding site for the AbrB TF in its core promoter region. Another operator for the Spo0A TF is in its upstream sequence. The extracted promoter sequence starts from the left side of the Spo0A operator and continues to the end of the 'AbrB-P2' promoter. The second promoter, 'AbrB-P1', starts from the middle of the 'AbrB-P2' promoter and also includes the operator site for AbrB. No promoter needs to be extracted in the latter case, since the core promoter already includes the operator.

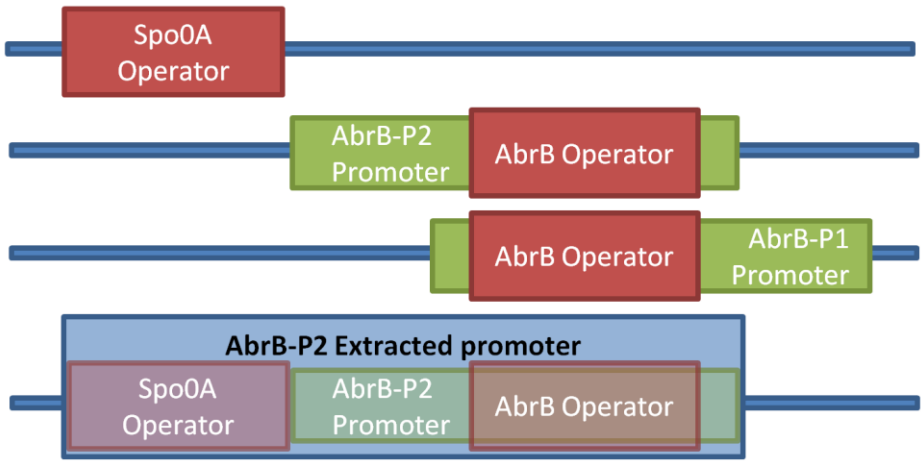


Figure 3.11: The AbrB promoter with associated operators. The first promoter, 'AbrB-P2', contains an AbrB operator and is downstream of a Spo0A operator. The second promoter starts from the middle of the first and includes the same AbrB operator. The third promoter is extracted by combining the Spo0A operator and the 'AbrB-P2' promoter.

The relationships between the promoters and their operators are shown in Figure 3.12. The AbrB operator is part of the two core promoters and the new extracted promoter. The AbrB-P2 core promoter and the Spo0A operator are also part of the new promoter.

Sometimes the operator and core promoter sequences overlap, in which cases the overlapping sequences were combined to create a composite promoter. For example, the beginning of the *rocR* promoter and the end of the RocR operator share some nucleotides (Figure 3.13). These two sequences were combined to create a RocR

repressible promoter (Figure 3.14). Therefore, the new promoter can be used as a simple inverter logic gate for the RocR input.

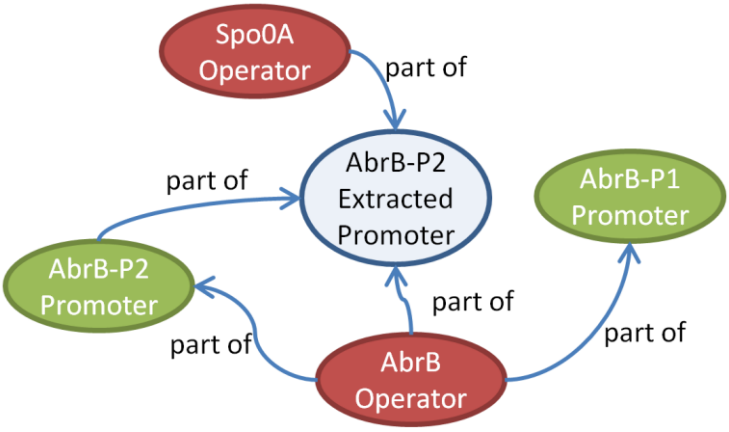


Figure 3.12: Relationships between the *abrB* promoters and operators. The AbrB operator is part of the sequence of all three promoters. The 'AbrB-P2' promoter and the Spo0A operator are part of the newly extracted promoter.

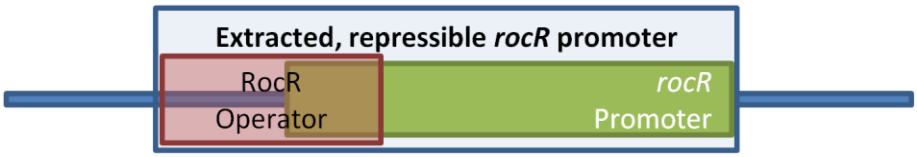


Figure 3.13: The extracted RocR repressible promoter. The *rocR* core promoter is combined with the upstream RocR operator to create a RocR repressible composite promoter.

```
Core promoter      :      AAAATTCTTTTGCATATCCTCTCCGTTTTTTTATAAAATAGAAGCAATA
Operator           : AGTGCAAAATTCTTTTGCATA
Extracted promoter: AGTGCAAAATTCTTTTGCATATCCTCTCCGTTTTTTTATAAAATAGAAGCAATA
```

Figure 3.14: The aligned sequences of *rocR* promoters and the operator.

3.4.2.4 Extracting RBSs and shims

RBSs affect the amount of proteins produced translationally. Therefore, RBS sequences identified from the genome can also be used as biological parts. In addition, the spacer sequences between promoters and these RBSs can be important to determine the rate of translation from RBSs due to secondary structure formation [113], and hence should also be identified.

RBSs were extracted from the network using the *Sequence Extractor* transformer. In *B. subtilis*, RBSs are usually located 4-13 nucleotides upstream of the start codon, with a consensus sequence of AAGGAGGT [124]. However, the RBS may be further

upstream. For example, the 20 bp preceding the *amhX* CDS is given with the AG**AAGGAGGT**ATTTCTTACC sequence. The consensus sequence marked in bold starts 18 bases upstream of the CDS. Furthermore, the upstream bases may also affect the translation efficiency of an RBS [113]. Therefore, 20 bases preceding the CDSs were used to identify putative RBS regions. In the workflow, when a promoter and a CDS were separated by more than 20 bases, the last 20 bases of this sequence, immediately upstream of the CDS start codon were extracted as the potential RBS. The beginning of this sequence between the promoter and the RBS, was extracted as a potential shim. The promoter was connected to the shim, the shim was connected to the RBS and the RBS was connected to the CDS with the `is_upstream_of` relation. Sometimes, the sequence between a promoter and a CDS could be less than 20 bases, in which cases, sequences more than 13 bases were also recorded as RBSs.

The spacing between most promoters and CDSs is more than 20 bases. However, some promoters extend to the start of the CDS, and some also include the start codon. In these cases, the 20 bases upstream of a CDS were used to create an RBS concept and no shim concept was created (Figure 3.15). If a sequence between a TSS and the start codon was less than 20 bases, the sequence between a TSS and a CDS was taken as the RBS and again no shim was created. The promoter sequences were modified to finish where the RBSs start. The original sequence annotations in all cases were kept as attributes of the concepts.

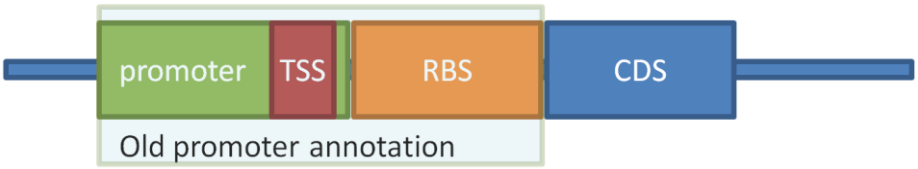


Figure 3.15: Extraction of RBSs that are annotated as part of promoters.

3.5 BacillOndex integrated dataset

The completed Ondex dataset includes 33,043 concepts and 94,774 relations between them. In the dataset, concepts are semantically connected, and the latest annotations for the genome were combined with data from KEGG, gene regulatory networks, gene expression data and protein-protein interactions. The concepts and relations were linked to the literature and other public databases using PMID and accession numbers where possible.

Ondex was used to visualise the dataset. A clustered view of the concepts and their

relations is displayed when the data are first loaded (Figure 3.16). Metadata describing concept types and their relationships is accessible through the Ondex metadata viewer. Different search, filtering and layout options can be used to visualise the data. Figure 3.17 shows a sample view for 'spo0A' as the query CDS and its first level neighbours. Triangles represent the CDSs and stars represent the TFs. Activation and inhibition relations are coloured red and green respectively. Regulation interactions are coloured yellow when the regulation type is not known.

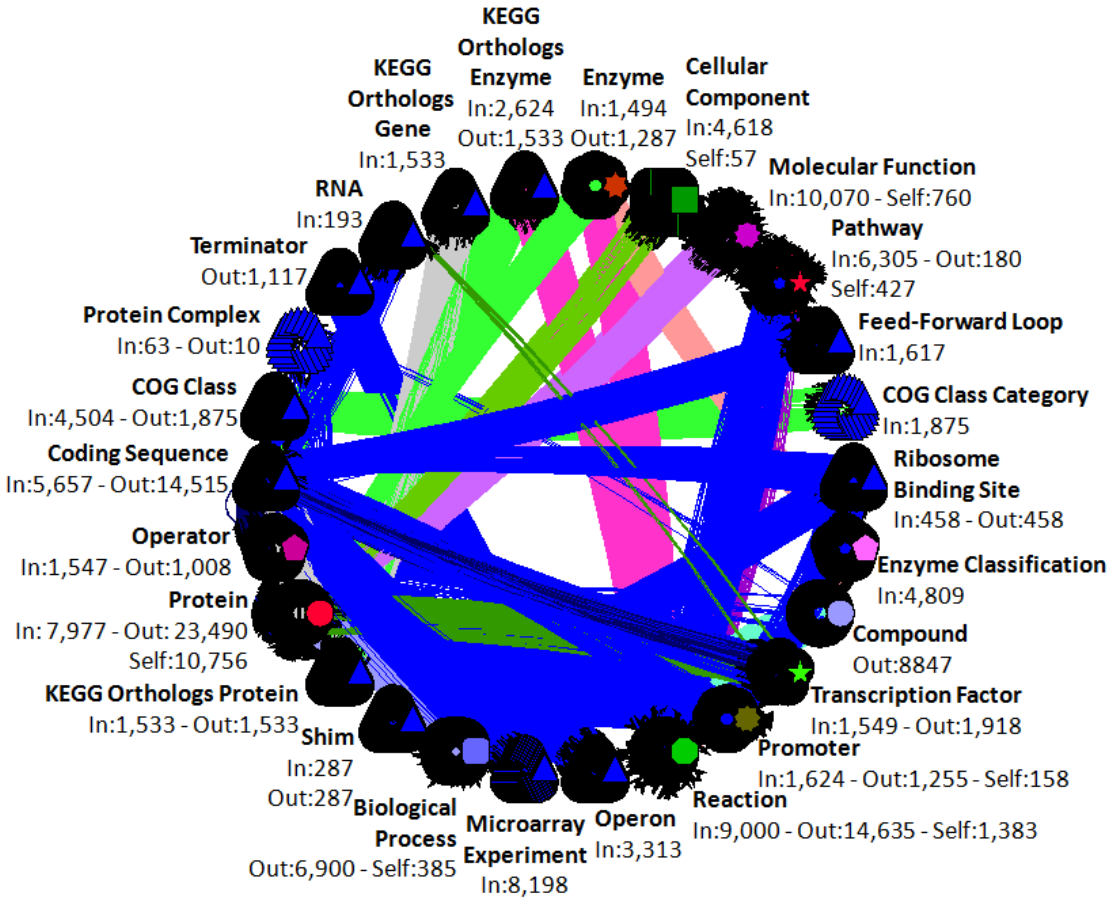


Figure 3.16: Clustered view of *B. subtilis* concepts in Ondex. Each circle represents a cluster of concepts from a concept class and the lines represent the relations between the concepts. Clusters are labelled with names, and numbers of incoming and outgoing relations which are respectively shown with the 'In' and 'Out' labels. The 'Self' label is used to indicate the number of relations within a cluster.

Concepts and relations in the network have attributes. Concepts have concept classes as their types and data source attributes indicating their data source(s). For example, the data source 'BacilluScope:DBTBS' indicates that the concept was derived by integrating data from the BacilluScope and DBTBS data sources. Most of the concepts in the network have preferred names, synonyms and accession IDs. Concepts derived from sequence-based features such as promoters, operators, CDSs, terminators, shims

and RBSs have nucleotide sequences and genome locations recorded as concept attributes[29]. A concept or relation derived from multiple sources may have additional attributes or relations inherited from these sources.

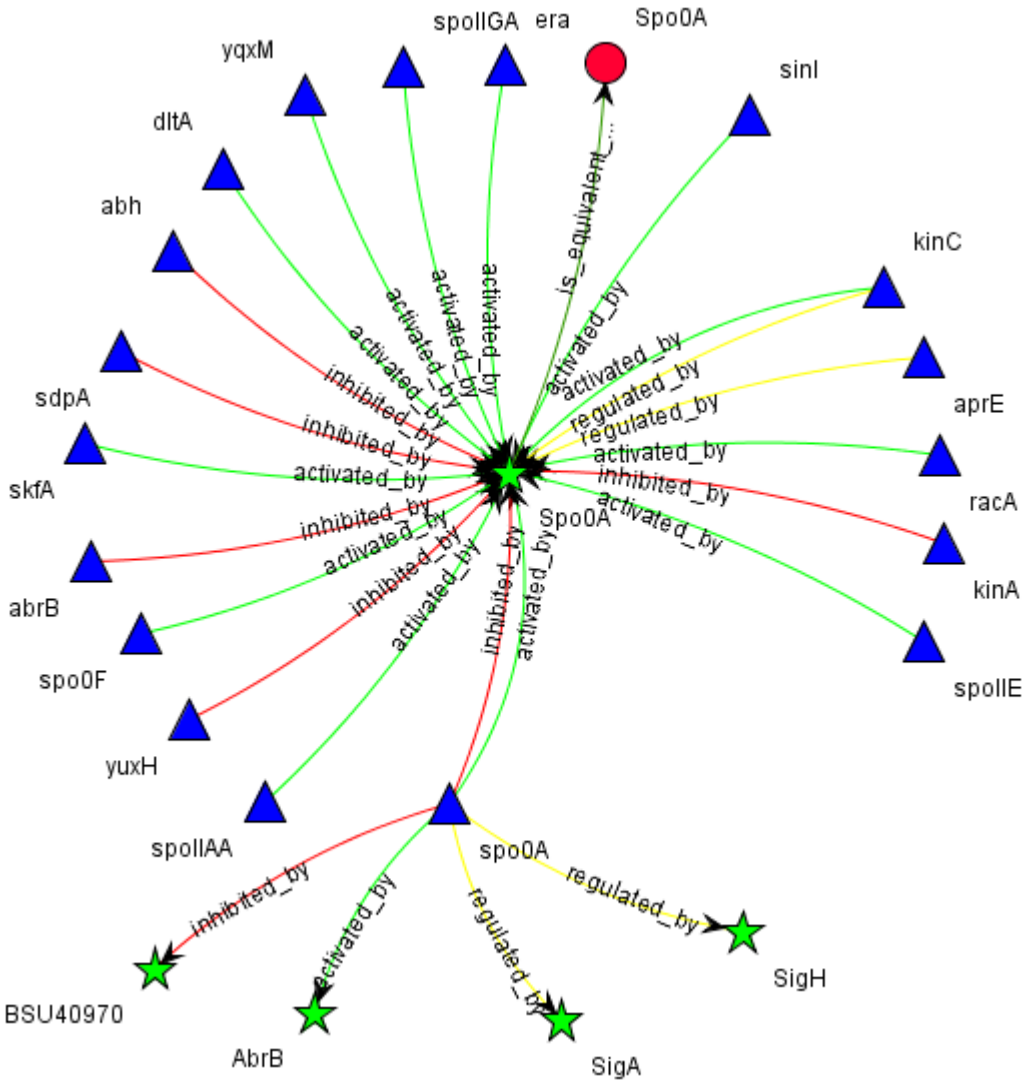


Figure 3.17: An Ondex subnetwork, including 'spo0A' as the query CDS with its first level neighbours. Triangles show the CDSs and stars show the TFs. Relations coloured red, green and yellow represent inhibition, activation and unspecified regulation respectively. The Spo0A TF regulates its own expression both positively and negatively.

3.5.1 Sequence-based features

A CDS concept has the attributes of nucleotide sequence, CDS length, reading frame, start and end position on the genome, normalised/unnormalised min/max gene expression values, KEGG URL, pathway tag (used to group the concepts according to the pathways they are involved in) and whether the CDS is positively or negatively

[29] These attributes are not recorded for operons.

autoregulated (Figure 3.18).

The integrated dataset includes 955 promoters, of which 158 were computationally inferred from core promoters and their surrounding operator regions. These promoters, including operator sites, can be inducible, repressible or combinatorial, allowing regulation by more than one TF. Core promoter concepts and operators are therefore part of these composite promoters. Core promoters and operators are also connected to each other via the `interacts_with` relation. For example, Figure 3.19 shows the repressible *rocR* promoter and its relationship to the negatively autoregulated *rocR* CDS. Both the core *rocR* promoter and the RocR operator are part of the 'rocR_full' composite promoter.

Figure 3.18: The attributes of *spo0A* CDS in BacillOndex.

In the network, 596 promoters, including the newly created composite promoters, are mapped to the *B. subtilis* genome. Typical bacterial promoters are around 100 bp in length [70]. In BacillOndex, the length of the 438 core promoters from the annotated promoter set range from 38 bp to 86 bp, and the composite promoters range from 45 bp (*spoVD* promoter) to 415 bp (*epr* promoter). Figure 3.20 shows the distribution of promoter lengths. Promoter attributes also include PMID and the annotated nucleotide sequence that has the -10 , -35 and TSS regions marked (Figure 3.21).

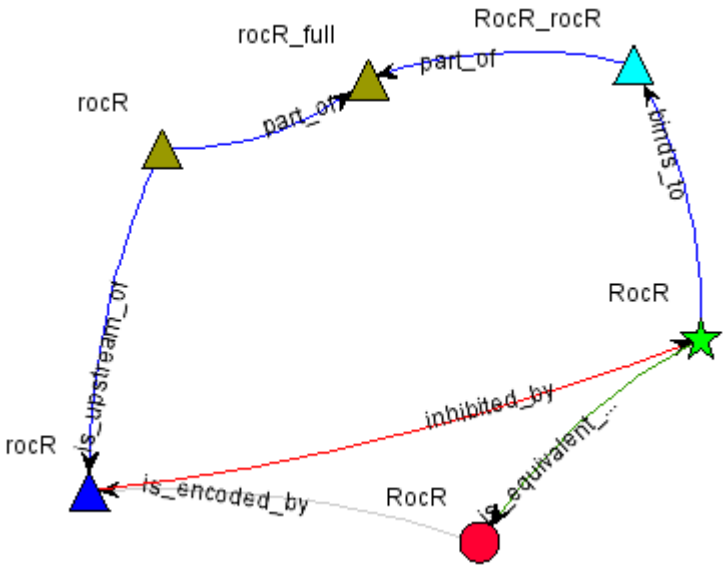


Figure 3.19: *rocR* promoters and the negative autoregulation of the *rocR* CDS. The *rocR* promoter and the 'RocR_rocR' operator are part of the composite 'rocR_full' promoter. The RocR TF binds to the operator to inhibit the expression of the *rocR* CDS. The TF is equivalent to the RocR protein encoded by the CDS.

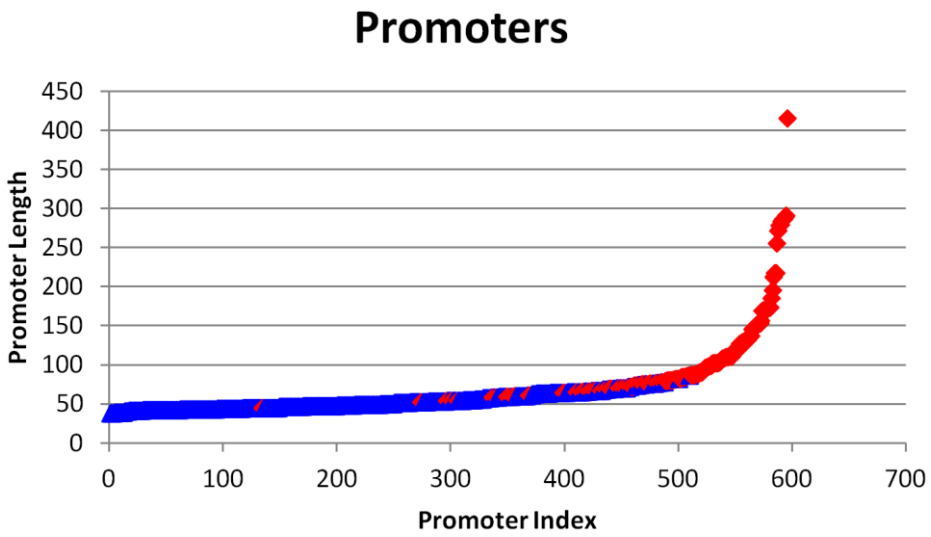


Figure 3.20: The distribution of promoter lengths for 596 promoters with genome positions identified. The lengths range from 38 bp to 415 bp. Red diamond and blue triangle shapes represent composite and core promoter sequences respectively.

A total of 458 RBS concepts were created, 345 of which are exactly 20 bases long and were used to create a consensus sequence. A FASTA file containing the RBS sequences was used as an input to the MEME tool [279] to identify the consensus sequence. The RBS consensus motif [AT][ATG]AAAGGAGG[ATG][AGT][AT][AT][ATC] was constructed (Figure 3.22), which includes the previously reported AAGGAGGT *B. subtilis* consensus sequence [124]. The first and last eight bases of the RBSs were also searched for a motif. However, the MEME tool could not find any motif for these areas. In addition, the integrated dataset includes 287 transcriptional shims for spacing promoters and RBSs, which were harvested upstream from RBSs and downstream from promoters.

Figure 3.21: The details in BacillOndex of the *ccdA* promoter, its RBS and the spacer sequence between the RBS and the promoter.

The other sequence-based concepts in the integrated dataset are 772 operators, 1,117 terminators and 1,131 operons. CDSs expressed from the same transcriptional unit are annotated as `part of` the same operon concepts which are `terminated by` the

terminator concepts; the latter are annotated with Gibbs free energy attributes [238] to indicate terminator strength. The nucleotide sequences of the operators are the TF binding sequences. Operators are further annotated with the attributes PMID, the identifiers of the sigma factor(s) that work with the operators and the regulation type: 'positive' or 'negative'.

Figure 3.22: The sequence logo for the consensus motif [AT][ATG]AAAGGAGG [ATG][AGT][AT][AT][ATC], produced using 345 RBSs with 20 bases.

3.5.2 Other biological molecules

In the network, 369 TF concepts `bind to` operator or promoter concepts. TF concept attributes include the TF domain, the TF family and binding motif. For example, Figure 3.23 shows the attributes of the Spo0A TF concept. The data source attribute of the concept indicates that the TF was derived from the BacilluScope, DBTBS and STRING databases. Its binding motif is TGTCGAA, it has 'response regulator' and 'Spo0A_C' domains and it is a member of the family of 'LuxR' type TFs. A TF concept may be connected to a protein or an RNA concept by the `is_equivalent_to` relation. When a TF concept is a sigma factor, the TF is annotated as binding to a promoter. In this case, a binding motif attribute represents the consensus sequence of promoters recognised by that sigma factor. TFs can activate or inhibit gene expression. This information is accessible through the regulation type attribute of the operator concept to which TFs bind. Alternatively, relationships exist between TFs and CDSs showing the type of regulation as activation or inhibition. When this relationship is not known explicitly, the `regulated_by` relation is used.

In the network there are 4,599 protein concepts whose attributes include amino acid sequence, compositional features such as molecular weight and isoelectric point, and various classifications such as product types, biological roles and processes. For example, the ComA protein has the 'Control' biological process classification, 'Inner membrane-associated' location, 'regulator' product type and 'Transcriptional regulation' role classification attributes. Most of the relationships in the network belong to proteins. Some of these relationships are also used to classify the proteins. For

example, the ComA protein is a COG0784 concept from the COG Class concept class (Figure 3.24). This relationship is represented with the `is_a` relation. The COG Class concepts are further classified using COG categories with single letter codes where each letter is assigned to a Biological Process or a Molecular Function. In the example, COG0784 is classified as COG class category concept 'T', which represents the molecules participating in cellular processes and signalling.

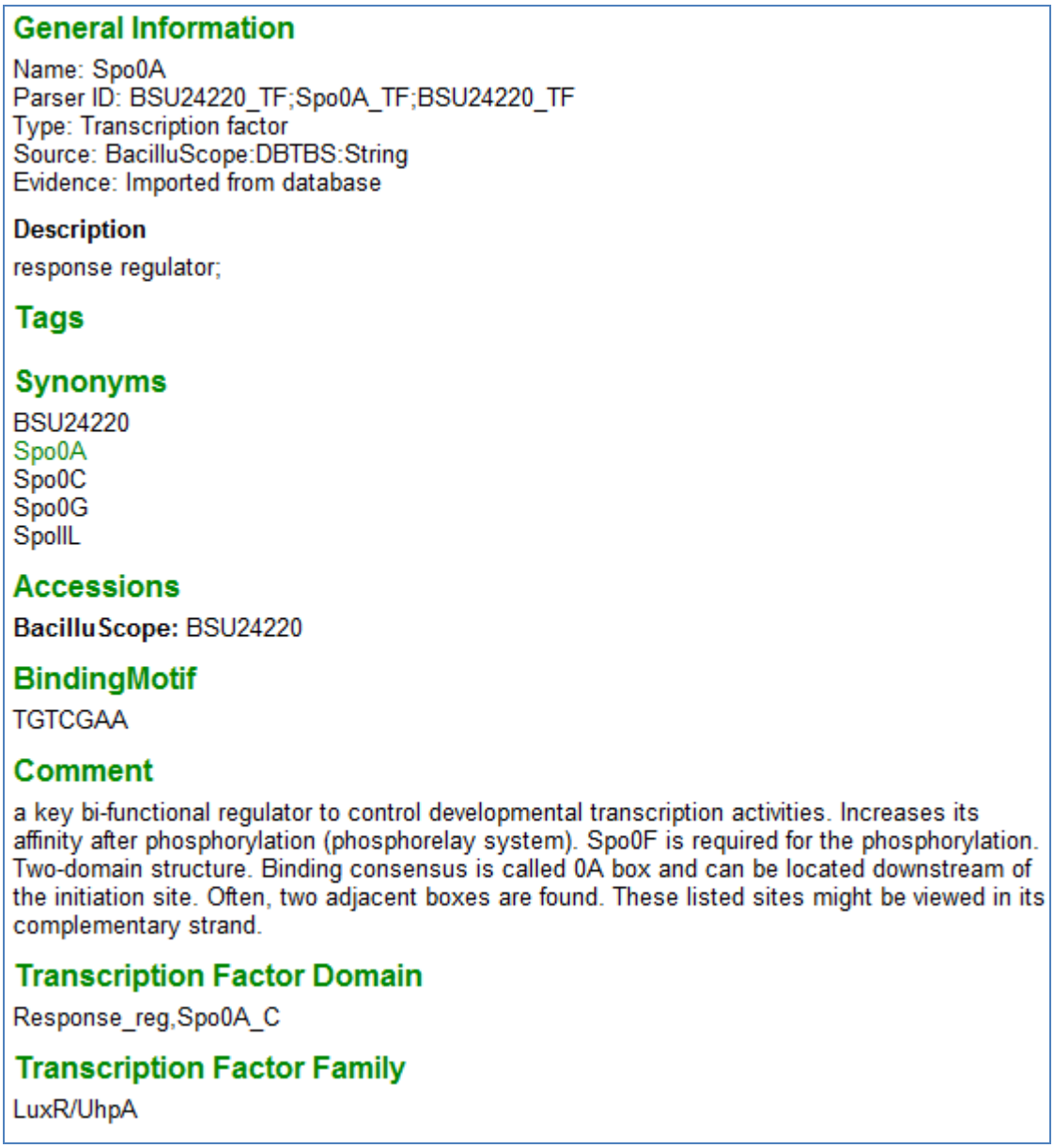


Figure 3.23: The details of the Spo0A TF concept. The TF is a LuxR-type response regulator, which also includes Spo0C and Spo0G synonyms. The Spo0A binding motif is TGTCGAA.

The integrated dataset includes 87 Cellular Compartment, 974 Molecular Function and 559 Biological Process concepts derived from the GO annotations. The GO-based concepts have human-readable labels from the ontology as concept names. Protein

concepts have functions from the Molecular Function concepts, participate in Biological Process concepts and are located in Cellular Component concepts (Figure 3.24). Protein concepts can also include `phophorylated_by`, `dephosphorylated_by`, `binds_to`, `disassociates_from`, `methylated_by` and `demethylated_by` relations to proteins. In this example, ComA is activated by ComP via phosphorylation (Figure 3.24). ComA is also related to DegQ via interactions such as those derived from text mining, and physically binds to RapC. Proteins can be `part of` protein complexes, which can also activate or phosphorylate proteins.

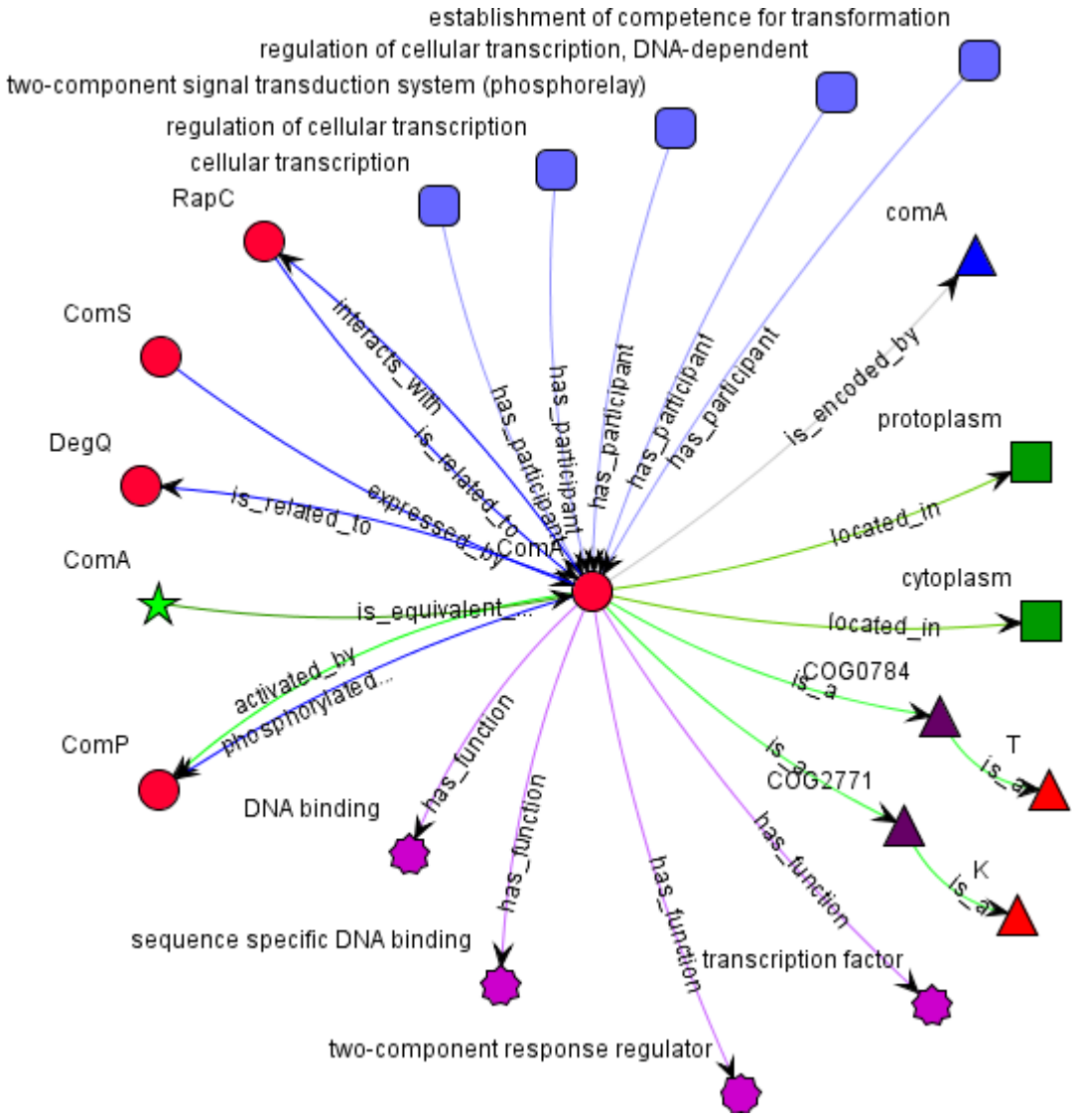


Figure 3.24: The ComA protein and its relationships to other concepts. The protein participates in biological processes such as ‘regulation of cellular transcription’, and ‘two component signal transduction system’. It is located in the cytoplasm and the protoplasm. It has ‘DNA binding’ and ‘two component response regulator’ functions. It is phosphorylated by ComP and binds to RapC. It is a COG0784 (‘CheY-like receiver’) and a COG2771 (‘DNA-binding HTH domain-containing proteins’) type protein.

In the network, 690 enzymes are also proteins catalysing a total of 3,963 reactions. Both enzyme and reaction concepts are connected to any of 2,300 enzyme classification concepts by the 'catalyzing class' relation. The same classification concept can be connected to more than one enzyme or reaction concept. For example, the YhfS, YusK and MmgA enzyme concepts are `part of catalyzing class` 'EC:2.3.1.9'. Reaction concepts can also be connected to any of 1,533 KEGG Orthologs Enzyme concepts by the `catalyzed_by` relation. These classification concepts have COG numbers and GO term IDs as their accession identifiers, and are annotated with EC numbers and KEGG URLs. Reaction concepts also have biochemical reaction annotations as free text. For example, the compounds 'Acetyl coenzyme A' (C00024) and 'Butanoyl-CoA' (C00136) are `produced by`, and compounds 'Coenzyme A' (C00010) and '3-Ketohexanoyl-CoA' (C0526) are `consumed by` the reaction 'C00024 + C00136 => C00010 + C0526' (RN:R01177) (Figure 3.25). Reactions that share compounds are connected to each other by the `share_intermediate` relation. For example, the reaction R01177 is connected to R04748 by this relation.

There are 2,817 compound concepts `produced by` or `consumed by` reactions. Compound concepts are annotated with their chemical formulae and their accessions to compound databases. Compounds can be connected to protein complexes by the `interacts_with` relation. Such complexes are usually formed to transport these compounds, which can also be `activated by` or `interacts with` proteins to indicate how these compounds are transported to the cell.

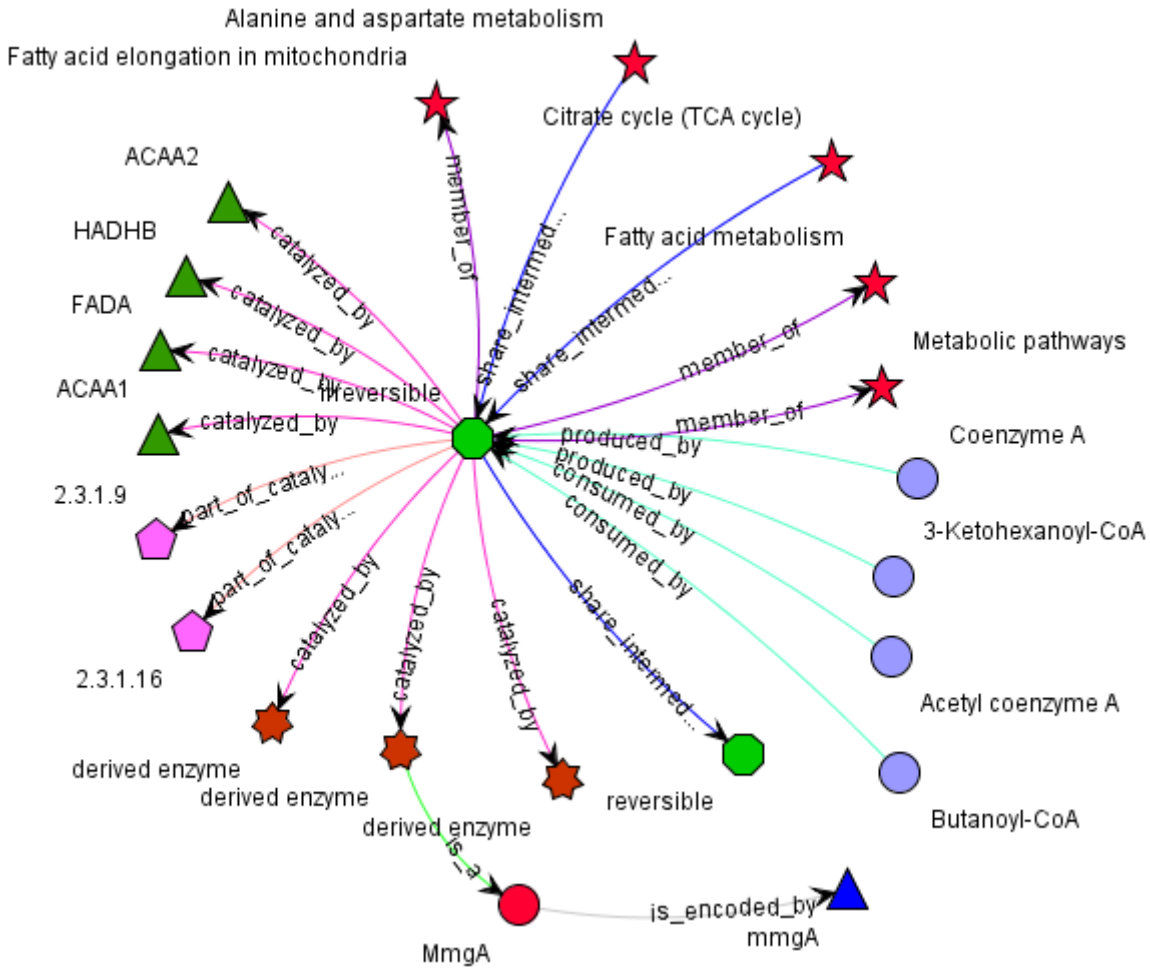


Figure 3.25: The reaction R01177 concept and its relationships to other concepts. This reaction is a member of pathways such as 'Citrate cycle' and 'Fatty acid metabolism' and is catalysed by three enzymes, one of which is MmgA encoded by the *mmgA* CDS. The reaction is part of the catalysing classes 'EC 2.3.1.16' and 'EC 2.3.1.9'. The catalyzing enzymes have also categories which are KOEN concepts, represented in this figure by green triangles.

3.5.3 Pathways

Enzyme, reaction, compound, enzyme classification and KOEN concepts are tagged with the pathways in which they participate. There are 270 pathway concepts in the network, such as 'Fatty acid metabolism', 'Citrate cycle', and 'Glycolysis'. Reactions are connected to pathways by the `member_of` relation. A pathway can also be connected to reaction concepts that are not members by a `share_intermediate` relation indicating that compounds produced or consumed by reactions from other pathways are used. For example, the 'Citrate cycle' pathway has a reaction member that uses the compound 'Acetyl coenzyme A', which is consumed by a reaction that is not a member of the pathway. Hence the pathway is connected to the reaction by a `share_intermediate` relation. Different pathways are also connected by `adjacent_to` and `derives_from` relations.

3.5.4 Network motifs

In the network, 51 negative and 19 positive autoregulatory CDS concepts were identified and 539 FFLs were extracted. Most of these FFLs are composed of sigma factors, and for some the nature of the regulation between genes and their products is unknown. Forty-four FFL concepts of eight different FFL types involve CDSs that are not sigma factors with known regulation types (Table 3.5). *B. subtilis* has more repressors than activators. Accordingly, negative autoregulatory genes and FFLs with repressors may occur more often than positive autoregulatory genes and FFLs with activators.

Table 3.5: Number of FFLs with known regulation types and involving no known sigma factor in BacillOndex.

FFL concept type	Number of instances
Coherent type 1	5
Coherent type 2	7
Coherent type 3	5
Coherent type 4	7
Incoherent type 1	3
Incoherent type 2	14
Incoherent type 3	5
Incoherent type 4	2

Of the 44 FFLs whose regulation is known, the one designated Incoherent type 2 (I2-FFL) employs three repressors and occurs most frequently (14 times). In *B. subtilis*, I2-FFLs are employed to make critical cell fate decisions, such as to sporulate, become competent or motile, produce extracellular degradative enzymes or form biofilms [131, 241]. These FFLs are highly interconnected. For example, 10 of the I2-FFLs are formed from 10 CDSs which regulate the *sdpA*, *comK*, *sboA*, *rok*, *epr*, *aprE* CDSs (Figure 3.26). All of these CDSs encode important proteins that control cell-fate decisions. *sdpA* is part of the *sdpABC* operon responsible for the production of the Sdp cannibalism protein[280], *rok* and *comK* play important roles in competence [281], *epr* has a role in swarming [282], *sboA* encodes an antibiotic in defence against other bacteria [283] and *aprE* is involved in scavenging [131]. The intracellular concentration of one of the most important proteins, SinR, is central in cell fate decisions in these I2-FFLs [18]. This protein is a master regulator of sporulation, biofilm and flagellum formation [284, 285].

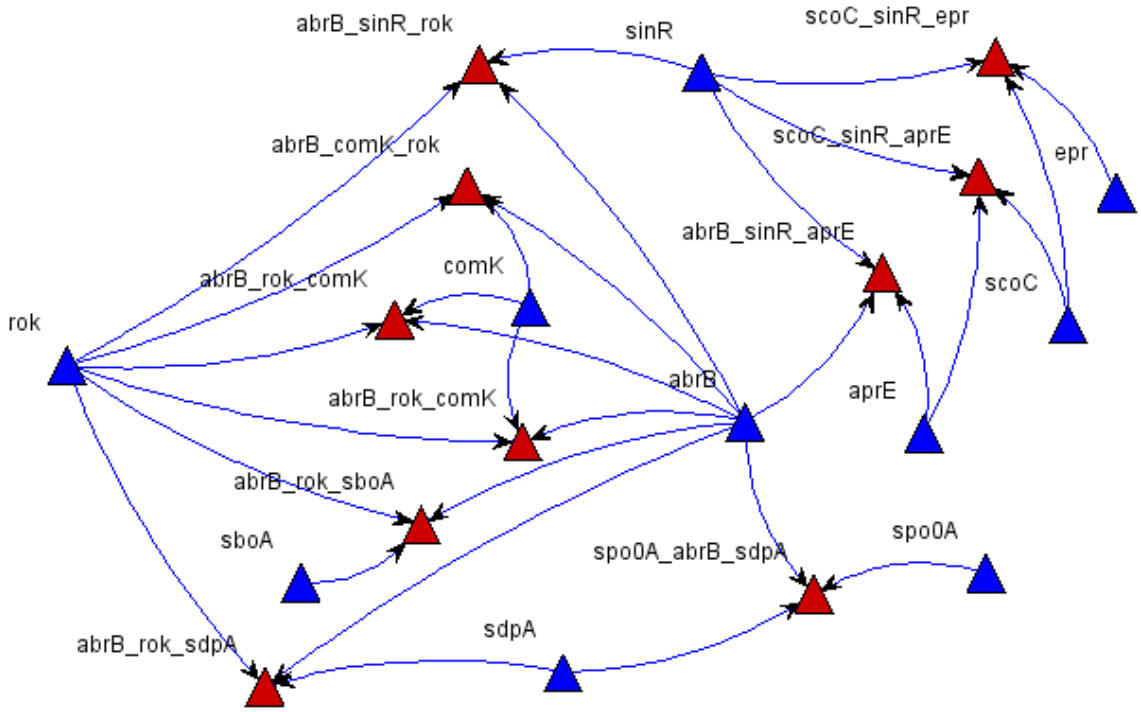


Figure 3.26: I2-FFLs involved in phenotype decisions.

3.6 Discussion

Various CAD tools [45, 251, 252] and design languages [21, 43, 117, 123] have been developed for synthetic biology. These tools use computational representations of biological parts to model or assemble biological circuits [33, 43]. Parts are imported, for example, from the BioBrick Parts Registry [12], the largest synthetic biology parts catalogue developed to date [13]. Thousands of students and researchers across the world have submitted constructs to the registry. However, compared to the range of existing DNA features in living systems, the number of parts in the Registry is very small. In March 2012, the Registry listed only nine promoters, four RBSs and 17 CDSs for *B. subtilis*[30]. Extensive information about DNA sequences that can be used as parts exists, but is distributed throughout the literature and numerous databases [16, 57, 150] making its manual extraction labour- and time-intensive. For large-scale synthetic biology, information about parts must be integrated and presented to computational design tools. In addition, in order to construct predictable circuits rationally, molecular interactions between these parts must also be captured.

3.6.1 Biological parts

The BacillOndex integrated dataset presented here includes information about

[30] http://partsregistry.org/Bacillus_subtilis. There may be a few other parts that are not listed on this page (accessed 13/03/2012).

thousands of sequence features that can be used as modular building blocks for synthetic genetic circuits. These basic biological parts [20] include CDSs, RBSs, promoters, operators, shims and terminators. Integrating data about these parts increases the understanding of these features, and hence facilitates the rational design of biological systems. In addition, these parts are mapped to the genome with their start and end positions. This location information indicates an exact origin and provides provenance about parts.

One of the most important biological part types for the design of genetic circuits is the promoter. These parts are used to transcriptionally control the amount of proteins. Therefore, various different libraries of promoters with known strengths have been developed [70, 71, 106]. However, these promoters are responsive to a few well-known TFs. For large-scale synthetic biology, promoters that can be regulated by a range of TFs are needed. The integrated dataset has 955 promoter concepts, 536 of which have been mapped to the genome with start and end positions, and are therefore immediately usable.

These promoter concepts include not only core promoter sequences representing where RNAP binds, but also promoters that have been combined with operators. These composite promoters can act as inverters, amplifiers, two-input logic gates such as NOR, NAND, AND, and multiple-input logic gates. When combined with promoters, operators enable the creation of new regulatory interactions. However, even for an expert, combining a promoter with an operator may be a difficult task, since the spacing between these sequences may change the functionality of a circuit [106]. These composite promoters offer a set of already working promoters with added operators from the *B. subtilis* genome.

Although much research has gone into creating promoter libraries [70, 71, 106], sequences of operators that can be used as modular parts are difficult to find. The integrated dataset is also a catalogue of operators. These parts combinatorially increase the possible number of new promoters, and hence of new regulations. For example, Cox and co-workers created a promoter library with 288 promoters using the AraC, LacI, LuxI and TetR operators combinatorially [70]. Therefore, this integrated dataset is a useful source of operators that can be used to construct logic gates such as AND and OR, when combined with promoter sequences.

Information about operator sequences can also be used to select suitable promoters for the design of genetic circuits. Using the integrated dataset, information about the connectivity of TFs to promoter regions and the location of their binding sites could be

combined with rules such as those relating to the effect of the relative location of binding site for a TF on gene expression [70]. For example, operators in the middle of promoters can be used to create a list of strongly repressed promoters for *B. subtilis* [90].

Most promoters from existing libraries are annotated with relative strengths [70, 71, 106]. The promoters in this integrated dataset are connected to CDS concepts which are annotated with normalised gene expression values. Gene expression rates with high values may suggest that the promoter preceding the CDS is a strong promoter. By mapping these expression values to transcription rates, the promoters can also be used in modelling.

RBSs can also be used to control the amount of proteins produced by genetic circuits [110]. The integrated dataset contains data on 458 RBSs with information about their relationships to preceding promoters and following CDSs. Although the dataset does not include data on the rate of translation for RBS sequences, tools such as RBSCalculator [110] and RBSDesigner [114] use RBS sequences as inputs and give estimated translation rates as outputs. Hence, the RBSs in the repository can be used in different designs.

In addition, spacer sequences were harvested from the *B. subtilis* genome for the separation of promoters and RBSs at the transcriptional level. These sequences are important since they may affect the translational efficiency of mRNAs due to secondary structure formation [113]. However, the shims described in the integrated dataset have yet to be experimentally tested to verify that they are functionally inert.

The parts mentioned so far are useful in order to control the expression of gene products, whose interactions determine the phenotype of the cell. CDS parts should also be catalogued for the functional properties of the gene products encoded. However, the process of going from a function to a DNA sequence has not been standardised [50]. The functional descriptions of parts in the BioBricks Parts Registry were made computationally, using the category classifications [13]. However, further readily available information such as GO Biological Process and Molecular Function annotations is desirable for part selection.

Machine-accessible data provided in the integrated dataset allows the automated extraction of parts for genetic circuit design using standard GO annotations. For example, CDSs that encode for kinases and response regulators can be identified using these annotations. In addition, cellular location annotations can be useful to identify proteins that occupy certain cellular locations. For example, to find a part that encodes a

protein located in the cell membrane, the specific Cellular Component GO term can be used. COG and KEGG orthology concepts can further be used for similar searches, or to find orthologous parts from other species. Standard EC number classifications are included for reactions and can be used to find parts that encode the desired enzymes. In addition to concepts derived from sequence features, the integrated dataset stores information about relationships between these features and gene products.

3.6.2 Interactions

Significant effort has gone into building catalogues of parts for synthetic biology [12, 13, 70, 71, 106], but these catalogues do not store information about how parts interact. Such knowledge is essential in synthetic biology. For example, the design of signalling systems requires information about how modifications such as phopshorylation are transferred between molecules [68]. The transport of an essential compound into the cell may require that the DNA features encoding the components of the transporter are also included in the target chassis [50]. The engineering of a pathway may require that the relationships between compounds, enzymes and reactions in the pathway are known [160]. The integrated dataset captures all of this information semantically, allowing the manual or computational mining of the knowledge. Computational approaches can therefore be used to automate the analysis of interactions for thousands of biological concepts and their relationships.

Information about interactions allows the global impact of a DNA feature at the phenotypic level to be investigated [16], facilitating the creation of predictable, robust and controllable genetic circuits. Such approaches are valuable to identify biological parts correctly and therefore to constrain the space of possible solutions for a biologically plausible genetic circuit design.

For example, a TF alone is not very useful unless the operator to which it binds is known. The integrated dataset does not only have operator and TF concepts but also relationship information through the `binds_to` relation specifying which TF binds to a particular operator. In addition, the information about the CDS required to encode a TF is recorded in the repository. Hence the integrated dataset is a valuable source of TF-operator pairs, with 369 TF and 772 operator concepts in the current version. These operator-TF pairs facilitate the rewiring of genetic circuits. In addition, some of these TFs are response regulators. For such a two-component quorum sensing system to work, both the sensory and regulatory proteins must exist together and the response regulator must be phosphorylated by the kinase [286]. In the integrated dataset, the

kinase and the response regulator proteins are connected to each other via the phosphorylation relationship. Once the proteins are identified, CDSs that encode them can be retrieved using the `is_encoded_by` relation between the proteins and the CDSs. Information about these relationships is essential in order to build predictive models.

Modelling is one of the most important tools in synthetic biology [53]. Dynamic models are built by first creating static models of the components in the model [77]. The integrated dataset is a static model of *B. subtilis*-specific sequence features, gene products and aggregates, and can be used to analyse *B. subtilis* parts and interactions on a large scale. A model typically includes representations of proteins and their interactions with other proteins, DNA features and external signals [22, 48]. Using relationships identified from the integrated dataset, an initial model for a given circuit or pathway can be constructed manually or computationally. When this knowledge is combined with the appropriate constraints and kinetic parameters, dynamic models can be generated.

In addition, the integrated dataset includes data about the functional interactions of the proteins. Genes encoding interacting proteins tend to be clustered on the chromosome, and positional information can be used to predict functional interactions [121, 127]. It has been shown in several organisms that chromosomal gene order affects the translation of proteins due to the very low diffusion rates of mRNAs [121]. Scores for phenomena such as gene fusion [266], co-expression [266] and co-occurrence [267] can be considered when constructing of genetic circuits, placing genes closer or further apart in order to increase or decrease any possible interactions. The design of biological circuits can also benefit from the use of network motifs, which may be used to provide the same response among cells [191].

3.6.3 Network motifs

Libraries of reusable high-level devices constructed from standard biological parts may be valuable for the engineering of complex biological systems. It is suggested that such modules can form the foundation for a set of design patterns for synthetic biology [90, 287]. Network motifs such as the FFLs and autoregulation genes identified in the integrated dataset can be used as building blocks with which to generate desired behaviours. For example, positive or negative autoregulatory genes can be used to create stochastic [196] or fast-response behaviours [195] respectively. For a genetic circuit that requires stable oscillations, for example, a positive feedback loop coupled

with negative feedback can be used [90, 241]. A biological device may be desired which creates a pulse for a particular signal or filters out low level environmental signals. Modules such as Incoherent type 1 FFLs as pulse generators or Coherent type 1 FFLs to filter noisy signals are already identified in the integrated dataset [195].

In addition, these network motifs can be analysed to investigate how the connection of modular building blocks gives rise to biological functions in *B. subtilis*. For example, from the integrated dataset, it can be seen that the organism uses I2-FFLs frequently for cell fate decisions. This special FFL motif appears to work as NAND or NOR gates, which are at the core of many electronic circuit designs. The use of repressors ensures that binding sites are always occupied when the repressors are present, and reduces the chance of other TFs binding to the DNA by not exposing the DNA site [241]. As *B. subtilis* has various phenotypes, this choice may be the reason that repressors are used more than the activators.

The BacillOndex integrated dataset was primarily developed to make *B. subtilis* data from public databases machine-accessible for large-scale genomic engineering. The dataset can also be used as a source of data for other applications. The semantically well-structured representation of data allows computational tools to extract useful information by traversing the graph. For example, network motifs in the repository were identified by investigating the connectivity of concepts; composite promoters, RBSs, and shim concepts were created. However, such approaches are dependent on the Ondex APIs. The dataset is also useful for any researcher working on this model organism. Custom views of the dataset can be exported and exchanged with other researchers using Ondex and its XML-based OXL format.

Although BacillOndex is an integrated dataset for *B. subtilis*, it also provides a set of tools and algorithms to facilitate the building of integrated networks for other species. All the databases used in this project, except DBTBS, include data for many other species[31]. Therefore, parsers and transformers from the plugin that has been developed for data integration can be configured with appropriate parameters in order to construct automated data integration workflows for other organisms. Compiling data for other species would increase the number of parts available for the design of large-scale biological systems [16]. These parts can then be used for *B. subtilis*.

This chapter demonstrates how data integration can be applied to extract extensive information about biological parts from existing genomes in order to construct complex

[31] Starting from June 2011, the KEGG's public download links were removed and license fees are charged to use its data.

biological systems. Information about these features is also linked semantically to functional annotations and molecular interactions between gene products, their aggregates and biological compounds for machine access, facilitating the rational design of these systems using computational approaches. In addition, the data model and the workflow-based data integration approach presented here can also be usefully applied to other organisms.

3.6.4 Conclusion

This chapter has described the process of integrating data to produce an integrated dataset for *B. subtilis* using public databases. Ondex was used as the integration platform. The dataset includes 33,043 concepts of 26 types and 94,774 relations between these concepts. Examples include 4,516 CDSs, 955 promoters, 772 operators, 458 RBSs, 1,117 terminators and 287 shims. Mining of the genome was used to increase the number of parts for synthetic biology. In addition, this integrated dataset enables tools to access information about parts and their interactions in order to construct biologically plausible biological systems. The dataset was developed as a semantically enriched biological network in the form of an XML-based OXL file of 82MB size. This network can be used to extract information to inform the design of synthetic genetic circuits, but is more suited for manual browsing and analysis using the Ondex tool. A formal representation of the data for computational access to scale and to automate the design process is needed. The dataset can be exported in standard formats such as RDF and OWL to make the knowledge accessible to a wide range of off-the-shelf tools. This process is discussed in Chapter 4.

Chapter 4. Ontologies for Synthetic Biology

The large-scale engineering of genetic circuits may be facilitated by automated design. To enable this automation, extensive knowledge about biological parts and their associated biology must be made available in a computationally-amenable format. However, data standards in synthetic biology are still not adequate for the representation of this knowledge. Ontologies are suitable for modelling a domain of interest as a formal specification for computers [144], and can therefore be used to represent available domain knowledge about biological parts for machine access. In this chapter, the modelling and use of ontologies for synthetic biology built on existing biological knowledge is demonstrated. The reasoning capabilities of ontologies are applied to automate the classification of information about biological parts for large-scale synthetic biology.

4.1 Introduction

In Chapter 3, the design and development of the BacillOndex integrated dataset was described. BacillOndex includes information about many aspects of *B. subtilis* molecular biology, including sequence features, protein-protein interactions, transcriptional regulations, gene expression and pathways. To construct the dataset, the six most relevant data sources containing data about *B. subtilis* were integrated into a semantically-annotated biological network using the Ondex data integration system. The resulting dataset can be visualised, queried and searched for biological concepts and their relationships.

For the large-scale engineering of genetic circuit designs, biological knowledge needs to be in standard formats that can be accessed and processed by computers. Computer-aided design (CAD) tools have already been developed by several groups to automate the design of synthetic biological systems [21, 43, 117]. For complex designs, these tools should ideally have access to computationally amenable catalogues of parts that provide large numbers of diverse types of parts [20, 118]. Ontologies are already widely used in the life sciences for this purpose, and may also be used for synthetic biology.

An ontology is defined as “an explicit specification of a conceptualisation” [35].

Therefore, a common conceptualisation of parts and their relationships can be modelled using ontologies. This modelling would enable shared understanding between computer applications. Moreover, for a fully automated design, knowledge about biological constraints in such machine accessible formats can be used to restrict the design space to possible solutions and to design biologically plausible systems [21].

Synthetic genetic circuits can be designed using a combination of top-down and bottom-up approaches in which the designs are specified using high-level specifications which are then implemented with well-defined biological parts [31, 32]. However, the design of complex and large-scale biological systems requires automation [14]. Therefore, for a computational design, domain knowledge that can facilitate mappings between these specifications and individual parts must be computationally available, providing the relationships between biological parts. Ontologies can act as mediators between the design tools and available domain knowledge to choose suitable parts in order to contruct biologically plausible systems.

The availability of machine-accessible data about parts and their interactions from the wealth of information can also facilitate the automation of large-scale model construction for synthetic biology. Such models are valuable for the computational analysis of designed circuits, particularly if the models are simulatable.

Although there is a large, and still growing, amount of data available to inform synthetic biology, most of this data is still implicit. Ontologies can be used to infer knowledge from known facts and to validate hypotheses [146]. The extraction of implicit knowledge can be used to systematically select parts that are required for a desired functionality. For example, promoter parts can be identified for the control of a desired system of regulation, and parts that encode for a targeted enzyme can be searched for computationally. The use of standard ontologies means that off-the-shelf reasoning tools can be used for the classification of information and hence for the automated identification of parts for synthetic biology.

Semantic Web technologies have become increasingly popular for modelling, accessing and exchanging data in the life sciences [59, 208]. Several databases provide data in the Resource Description Framework (RDF) format [60]. These databases use standard terms from biological ontologies [204, 221] for the annotation of biological concepts and their interactions. Furthermore, off-the-shelf tools that support Semantic Web technologies are used for the storage [172], querying [59] of, and reasoning with, biological data [146, 222, 288].

These technologies are also increasingly being used within the synthetic biology

community. Tool and part catalogue developers, representatives from industry and academics have agreed on a format for the electronic exchange of information about biological parts. This format is called the Synthetic Biology Open Language (SBOL), and is based on an ontology [63]. Currently, the core data model is small, focusing upon the exchange of sequence-based information; however, extensions for purposes such as modelling and the representation of experimental results are under discussion. Existing data in SBOL format about BioBricks from the Parts Registry [12] have been presented using an RDF triple store, enabling SPARQL querying of the parts [13].

Modelling BacillOndex with an ontology would enable computational access to information about parts and biological constraints from this integrated dataset in order to automate the identification of parts for the design of synthetic genetic circuits. BacillOndex is built upon Ondex, a data integration tool which uses semantically defined labels to represent biological concepts and their relationships. These labels form an ontology. However, this ontology is not accessible via standard ontology languages, and is therefore not available to the wide range of computational tools that implement these languages, including automated reasoners. The Ondex ontology is not rich enough to represent all the conceptual relationships typically supported by automated ontological reasoning. For example, although biological concepts are modelled in Ondex using a hierarchy, multiple inheritance is not available. In addition, computational access to the dataset can only be achieved through the use of Ondex APIs programmatically. The dataset is more suited to analysis by humans. The use of standard ontology languages would make the dataset available to a wide range of existing tools.

This chapter describes the development of an ontology for synthetic biology, SynthBiOnt, using the BacillOndex integrated dataset. This ontology is a computationally amenable, semantically-rigorous catalogue of parts and their interactions. In SynthBiOnt, biological concepts and their relationships are expressed with strictly defined terms in order to enable the automation of data extraction and classification for synthetic biology. The biological concepts, their relationships and attributes from BacillOndex were initially converted into RDF triples, which were used to build the ontology using Web Ontology Language (OWL) axioms. SynthBiOnt is in the RDF/OWL format and can be queried using languages such as SPARQL [59] and Terp [230]. SynthBiOnt incorporates terms from the Sequence Ontology (SO) and Gene Ontology (GO), the standards in their areas, and so to make the data available to the wider community. SynthBiOnt also conforms to the SBOL standard, which supports the

standard querying and exchange of data among the synthetic biology community. This chapter also demonstrates the application of automated reasoning to the systematic identification of parts for synthetic biology.

4.2 From biological networks to semantically-rich triple stores

BacillOndex was converted to RDF format using an Ondex workflow, organised as follows. The dataset was read into Ondex using the OXL parser and exported as an RDF graph using the Ondex RDF exporter. The graph was saved as a single RDF file which contains triples of concepts, their relations and their attributes, together with the Ondex metadata, including annotations regarding concept classes, relation types and the relation hierarchy (Figure 4.1).

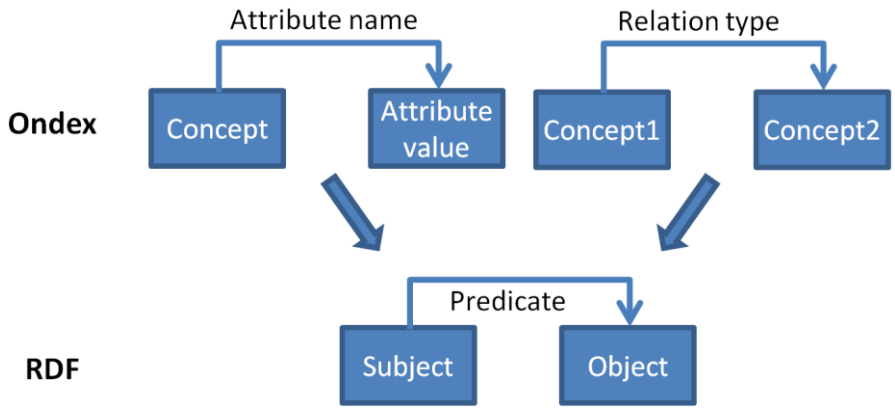


Figure 4.1: The way in which data from BacillOndex were converted to RDF. Example patterns include concept-attribute-value and concept-relation-concept triples.

Each concept in BacillOndex was represented as an RDF resource with a unique URI. The relations linking concepts are represented by triples in which the subject, the predicate and the object are, respectively, the concept, the relation type and the concept with which it interacts. The attributes of a concept are also represented as triples in which the subject, the predicate and the object are, respectively, the concept, its attribute type and the attribute value. The attribute values are represented by RDF literals. Figure 4.2 shows the RDF representation of a promoter with the ID of 3256 as an example. Attributes of the promoter, such as 'BEGIN', 'END', and 'NA', are shown with their values. The promoter is upstream of two other sequence features identified by the IDs 3256 and 30843. These relationships are represented with two different RDF statements, which include the promoter as the resource, the `upstream` predicate, and the URIs of these features as objects: '3256 upstream 28023' and '3256 upstream 30843'.

The semantics of the RDF graph are provided by RDFS statements. Concept classes

can be subclasses of other concept classes reflecting the hierarchy of the Ondex metadata. Relation types are modelled as RDF properties. These properties can have parent properties, as they do in the Ondex metadata.

```
<rdf:Description rdf:about="http://www.bacillondex.org/concept/3256">
  <j.0:annotatedsequence>TTT[AACGAAA]GTTCCATCTGATTTAA[CAAAAGAT]AAAAC/A/G
     TCACATATTA</j.0:annotatedsequence>
  <j.0:StartOfBindingSite>-39</j.0:StartOfBindingSite>
  <j.0:PMID>6205155</j.0:PMID>
  <j.3:elementOf rdf:resource="http://www.bacillondex.org/cv/DBTBS"/>
  <rdf:type rdf:resource="http://www.bacillondex.org/conceptClass/Promoter"/>
  <j.0:BEGIN rdf:datatype="http://www.w3.org/2001/XMLSchema#int">53083
</j.0:BEGIN>
  <j.3:conceptName>sspF</j.3:conceptName>
  <j.2:upstream rdf:resource="http://www.bacillondex.org/concept/30843"/>
  <j.3:pid>sspF_PR</j.3:pid>
  <j.0:EndOfBindingSite>+12</j.0:EndOfBindingSite>
  <j.0:NA>TTTAACGAAAGTTCCATCTGATTTAACAAAAGATAAAACAGTCACATATTA</j.0:NA>
  <j.2:upstream rdf:resource="http://www.bacillondex.org/concept/28023"/>
  <j.3:evidence rdf:resource="http://www.bacillondex.org/evidenceType/IMPD"/>
  <j.0:END rdf:datatype="http://www.w3.org/2001/XMLSchema#int">53133</j.0:END>
</rdf:Description>
```

Figure 4.2: The RDF representation of a promoter concept. Attributes and relations of the concept are represented with RDF statements, in which the promoter is the subject. Attribute names and values are represented as predicates and literals respectively. Relation names and concepts that are linked to the promoter are represented as predicates and resources respectively. These concepts are accessible through their unique URIs in these statements.

The RDF file can be loaded into a triple store backend for querying purposes. SPARQL queries can then be constructed using the concept classes, relation types, attribute names, and Ondex core properties. The relevant namespace declarations for these resources are shown in Table 4.1. For example, the query in Figure 4.3 retrieves the URI and name of all promoters that have operator sites.

Table 4.1: Namespaces used in the RDF triple store.

Category	Namespace
Concept Classes	http://www.bacillondex.com/conceptClass/
Relation Types	http://www.bacillondex.com/relationType/
Attribute Names	http://www.bacillondex.com/attributeName/
Ondex Core Properties	http://ondex.sourceforge.net/ondex-core#

```
PREFIX att: <http://www.bacillondex.org/attributeName/>
PREFIX cc: <http://www.bacillondex.org/conceptClass/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rt: <http://www.bacillondex.org/relationType/>
PREFIX core: <http://ondex.sourceforge.net/ondex-core#>

select distinct ?Promoter ?PromoterName
where{
   ?Promoter rdf:type cc:Promoter;
      core:conceptName  ?PromoterName.
   ?Operator rdf:type  cc:Operator;
      rt:part_of ?Promoter.
}
```

Figure 4.3: The SPARQL query to retrieve all promoters that have operators. The promoters that have `cc:Promoter` as the `rdf:type` are filtered out. Only the promoters that have the object `cc:Operator` for the `rt:part_of` predicate are returned.

4.3 From triple stores to ontologies

RDFS provides the semantics to model the types of resources in the BacillOndex RDF triple store, and the hierarchy between them. However, an ontological modelling of these resources using OWL offers more formal and powerful representations, making computational access easier. For example, terms provided in OWL [24], such as restrictions that specify the values of properties, makes knowledge more explicit for machine access [16], and hence allows better reasoning capabilities compared to reasoning using RDFS. In addition, an ontology can be stored in RDF format to take advantage of existing RDF tools and query languages [213]. Furthermore, such ontological representation of data allows the use of off-the-shelf reasoners [227, 228] to infer implicit knowledge.

Because of these factors, the BacillOndex RDF graph was converted into OWL/RDF format in order to formally model the *B. subtilis* domain knowledge as an ontology. The resources that represent concept classes and their concepts from BacillOndex were modelled as OWL classes in the ontology. The relations and attributes of concepts were modelled as subclass restrictions on these classes (Figure 4.4). The ontology was mapped to external resources such as SO and SBOL. In addition, a subset of GO was included to represent the full class hierarchy of the terms that have *B. subtilis* annotations.

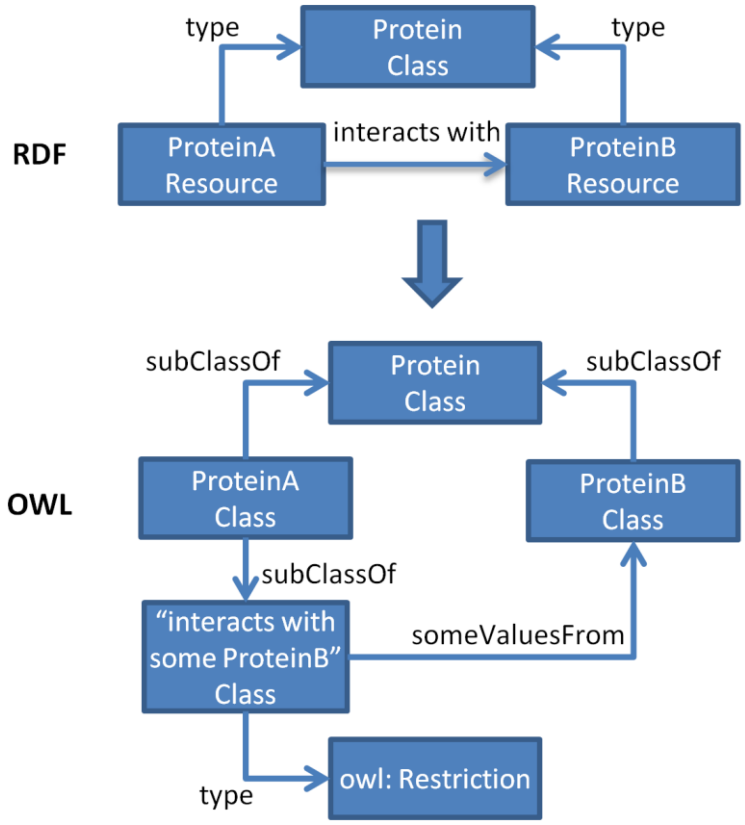


Figure 4.4: Example of the transformation of a relationship between two proteins into the OWL format. The 'Protein' concept class and the interacting proteins are represented with OWL classes. The interaction relationship is placed as an OWL restriction on the first protein class.

4.3.1 Classes vs. individuals

When building an ontology, it is important to decide which concepts should be modelled as classes and which as individuals in order to allow automated reasoning to infer sound statements [208]. By using appropriate subsumption (`is_a` relationships) between classes by grouping individuals into classes that have common behaviour, the usefulness of the results of reasoning across the data can be improved.

The entities in BacillOndex, such as coding sequences (CDSs) and proteins, do not represent individual molecules but groups of molecules that exist in all cells. When expressing this knowledge, details about specific cells or molecules are not referred to [208]. Such molecules can be modelled with classes that represent all of their individuals [146]. This approach has also been previously applied to the modelling of knowledge in OBO ontologies or in biomedical knowledge bases that are annotated using the classes from OBO ontologies [146, 208, 289]. Accordingly, concepts from BacillOndex were modelled as classes, and, hence, the attributes and relations of a concept are inherited by all its individuals. For example, the 'Spo0A' protein concept in BacillOndex represents a class to which all individual Spo0A proteins belong. By

relating the ‘Spo0A’ class to the ‘Spo0B’ protein class by using the ‘is phosphorylated by’ restriction, all of the ‘Spo0A’ individuals inherit this relationship. Hence, SynthBiOnt models general features of such a protein, but does not describe the specific properties of its individuals.

4.3.2 Modelling the SynthBiOnt ontology

The BacillOndex dataset includes 26 concept classes (Table 3.2), including `CDS`, `Protein`, `Enzyme`, `TF` and `Pathway`, which were modelled as OWL classes in SynthBiOnt. These classes have subclasses which represent corresponding concepts from BacillOndex. For example, an OWL class that is created for a protein concept from BacillOndex is a subclass of the `Protein` class. Example classes include the ‘spo0A’ CDS, ‘Spo0A’ protein, ‘Spo0A’ transcription factor (TF), ‘histidine kinase’ molecular function, the ‘pVeg’ promoter, and the Spo0A’s ‘kinA’ operator classes. These classes have unique identifiers as in the OBO ontologies [62]. SynthBiOnt uses the base URI http://www.bacillondex.org to construct these IDs. For example, the class for the ‘cold shock protein CspB’ is identified with the URI http://www.bacillondex.org#10016.

OWL object properties were used to model the relationships (Figure 3.2), between the classes in SynthBiOnt. These properties can also be identified with unique URIs consisting of the base ontology URI and the relation type ID from BacillOndex. For example, the `binds_to` relation is associated with the http://www.bacillondex.org#bi_to URI. Object properties are annotated with their names and descriptions from BacillOndex, using the standard `rdfs:label` and `rdfs:comment` annotation properties. In addition, `evidence` and `elementOf` object properties were added to model the evidence type and data source relationships respectively.

Relationships between classes were captured using OWL property restrictions. These restrictions were implemented using the `owl:Restriction` class, which is a special class that allows one to define classes by using existing properties and classes [211]. For example, the ‘Spo0A’ protein is represented by a single class, which has a restriction which state that all of the the individuals of the ‘Spo0A’ protein class are encoded by individuals of the ‘spo0A’ CDS class.

In SynthbiOnt, restrictions are usually expressed using the OWL’s `someValuesFrom` restriction [225]. This restriction type states that instances of this class are in a given relation to at least one instance of a particular class, but does not

rule out that same individual being in the same relationship to instances of other classes. For example, a restriction can be used to say that the 'Spo0A' TF binds to the 'kinA' operator. SynthBiOnt represents this restriction as 'binds to some kinA operator' on the 'Spo0A' class. The statement does not imply whether or not there are other operators that the TF binds to. This open world assumption (OWA) facilitates the modelling of biological concepts without making overly restrictive or specific claims [222].

OWL reasoners allow the classification of individuals using inverse object properties [290]. Inverse relationships between individuals are two-way relationships, and can be deduced even when an inverse property is not explicitly stated for the individuals in a particular ontology. For example, for a given statement:

'ChildA_individual has_parent ParentA_individual'

, the inverse statement:

'ParentA_individual has_child ChildA_individual'

can be deduced. However, the same approach does not work for the classification of classes. Given that

'a_daughter has_parent a_mother'

restriction, it is possible to infer that every daughter has a mother. However, the inverse 'has_child' relationship is not enough to classify all mothers as having daughters as they may only have sons. Hence, some inverse object properties, such as `encodes`, `has_part` and `bound_by` were added to the SynthBiOnt ontology, and classes were explicitly annotated with restrictions that use these properties.

The `is_a` relation from BacillOndex was used to implement subclassing between the classes in the ontology. This relation type is used to define one biological concept as the subtype of another [219]. For example, the 'NarK is_a COG0477'[32] triple from BacillOndex is modelled as 'NarK is a subclass of COG0477' in SynthBiOnt, to state that every instance of 'NarK' is also an instant of 'COG0477'.

Another relation type, `is_equivalent_to`, is used to assert that two concepts from BacillOndex refer to the same thing. This relation was modelled using OWL's `equivalentClass` axioms. For example, the protein NsrR and the concept TF are defined as same in SynthBiOnt. The classes that are linked with this property are regarded as one class, and all of the restrictions that apply to one apply to all. Queries can refer to any of these classes, taking advantage of implicit semantic collapsing. For example, rather than querying for CDSs that encode for proteins that act as TFs, the

[32] COG0477: Permeases of the major facilitator superfamily.

query can be simplified to retrieving the CDSs that encode TFs.

Attributes of biological concepts were modelled using OWL's `hasValue` restrictions (Figure 4.5). Moreover, some concept attributes were modelled as annotations, using OWL's annotation properties. These attributes were selected from those that are not likely to be used for reasoning purposes. For example, the values corresponding to the `title` and `conceptName` attributes are stored as `rdfs:label` annotations. `Description`, `annotation`, `comment` and `conceptDescription` values are stored as `rdfs:comment` annotations.

```
Class: 26996
    SubClassOf:
        conceptAccession value "BSU24220",
        ProductType value "r : regulator",
        LOC value "2 : Cytoplasmic",
        RoleClassification value "3.5.2 : Transcription regulation",
        BioProcess value "16.12 : Sense"
```

Figure 4.5: Some of the restrictions created for the 'Spo0A' protein class in the Manchester OWL syntax.

4.3.3 Reasoning

SynthBiOnt can be used for the classification of information about biological parts for synthetic biology using automated reasoning. Although text descriptions of classes from the ontology can be used by humans to assign class memberships manually, in order to use computers this process requires the logical definition of classes [291], which can be described expressing *necessary*, or *necessary and sufficient* conditions [222]. Necessary conditions are those that are required to be fulfilled by individuals of these classes. For example, having the DNA binding molecular function can be considered as a necessary condition to classify proteins as TFs. These conditions were implemented in the SynthBiOnt classes using restrictions acting as superclasses.

However, necessary conditions are not sufficient for the classification of information. For example, proteins such as nucleases can also bind to DNA, and hence the DNA binding condition would not be sufficient to classify TFs. On the other hand, necessary and sufficient conditions, such as a restriction about having a transcriptional regulator molecular function for a TF, can be used to assign class memberships directly, and hence to extract implicit knowledge [144, 222]. When implemented via `equivalentClass` axioms [213], such restrictions become necessary and sufficient conditions [146]. Examples of these equivalent classes were defined for operators, promoters and CDSs, which are explained in Section 4.4.1. Therefore, when reasoners

are executed, classes with necessary conditions can be linked to these newly defined classes through subclassing.

Criteria described in defined classes are used by reasoners to categorise classes. However, due to the OWL's OWA, not all classes can be classified. For example, the classification of concepts such as promoters in terms of their compositional features requires that the set of features is explicitly specified and that these compositional features can be distinguished from each other [222]. Therefore, in order to classify a promoter with only one TF binding site, in addition to the necessary condition to have an operator, the sufficient condition that this operator is the only binding site must be included. Such a sufficient condition can be provided by closure axioms, which are used to indicate that no other information would be available except what is provided [225]. These closure axioms were added to the ontology using OWL's universal `allValuesFrom` (only) restrictions [288, 292].

4.3.3.1 Enabling the automated classification of promoter parts

When designing synthetic circuits, promoters can be used as logic gates such as one-input inverters (repressible), amplifiers (inducible), or two-input logic gates like AND and OR [69, 293-295]. Therefore, the automated classification of promoters requires information about their operators and whether they are regulated positively or negatively. SynthBiOnt already provides the relationships between promoter and operator classes using the existential restrictions on `has_part` object properties. However, closure axioms must be added to provide the sufficient conditions.

Figure 4.6 shows two promoters, Promoter1 with one operator and Promoter2 with two operators. In a classical database approach that uses the closed world assumption, according to which "a statement is true when its negation cannot be proven" [296], the first promoter would have only one operator. However, in the OWA model which is used by OWL, the 'Promoter1 has_part some Operator1' axiom does not imply that the promoter has *only* Operator1, unless this fact is explicitly specified. For example, Operator2 may be part of Promoter1 and the two promoters may even be the same. Furthermore, the axiom 'Promoter2 has_part Operator2' does not explicitly state whether Promoter2 has two operators, since it has not been specified that Operator1 is different from Operator2. A closure axiom 'has_part only Operator1' can be added to specify that Promoter1 has only Operator1. However, this restriction does not state the number of constituents of the promoter. Information about the operators is particularly important in order to classify a promoter as constitutive, regulated by one TF or acting

as a multi-input logic gate.

```
Class: Promoter1
     SubClassOf:
          has_part some Operator1,
```

```
Class:Promoter2
     SubClassOf:
          has_part some Operator1,
          has_part some Operator2,
```

Figure 4.6: Two promoter classes with necessary conditions. The first promoter has the Operator1 operator, and the second promoter has both Operator1 and Operator2 operators.

The number of operators for a promoter was made explicit in the ontology using cardinality restrictions (see Section 2.2.7.3) to facilitate reasoning about the number of a promoter's inputs. In OWL, these restrictions are used to describe the minimum, maximum, or exact number of relationships for a class. Figure 4.7 shows Promoter2 and its cardinality restrictions. The promoter has precisely one Operator1 and one Operator2. In addition, the universal 'has_part only (Operator1 or Operator2)' closure axiom is added to specify that the promoter can only have Operator1 or Operator2, which are explicitly defined as two disjoint operators. Therefore, reasoners can now infer that 'Promoter2' has exactly two distinct operators.

```
Class: Operator1
    DisjointWith:
       Operator2
Class: Operator2
    DisjointWith:
        Operator1
Class: Promoter2
    SubClassOf:
       has_part exactly 1 Operator1,
       has_part exactly 1 Operator2,
       has_part only (Operator1 or Operator2),
```

Figure 4.7: Closure axioms and disjointness statements are added to enable reasoners to infer that Promoter2 has two operators.

In SynthBiOnt, disjointness axioms (see Section 2.3.2.3) were added to operators that are part of the same promoters. Disjointness was also defined between the promoter and operator superclasses. By adding disjointness on these superclasses, all of the operator subclasses are made disjoint from all of the promoter subclasses. Otherwise, it would not be possible to distinguish the core promoter part of a composite promoter from its operators using the OWA model.

4.3.3.2 Systematic identification of CDS parts

The classification of CDSs in SynthBiOnt is achieved by defining classes that refer to

GO terms. CDSs generally form the core of genetic circuits, encoding the functional products. The proteins encoded by CDSs have relationships with GO classes representing biological functions, cellular compartments and biological processes. Therefore, a relationship between the protein product of a CDS and a GO class is sufficient to classify the CDS. For example, to classify the CDSs encoding proteins that have the 'receptor activity' molecular function, the `ReceptorEncodingCDS` class can be defined to describe this relationship. In addition, the SBOL ontology provides terms to model biological parts for computational access and these terms can be applied to standardise the querying of sequence features from SynthBiOnt.

4.3.4 Mapping SynthBiOnt to the SBOL ontology

Sequence-based features and their part-whole hierarchy of part composition were also expressed using the SBOL data model in SynthBiOnt. In the SynthBiOnt ontology, promoters, CDSs, terminators, shims, ribosome binding sites (RBSs) and operators are basic biological parts and have corresponding OWL classes, with specified nucleotide sequence restrictions. These sequence features can be modelled with the `DnaComponent` class of SBOL. In SynthBiOnt, some sequence-based features such as operator sites are modelled, via SBOL annotations, as part of other features. For example, a promoter with two operator sites can be modelled as a `DnaComponent` with two `annotations` that have operators as `subComponents`. Sequence annotations in SBOL include the start and end positions of sequence features. Although such information does not exist directly in the ontology, it can be inferred from the chromosomal start and end positions. In addition, SBOL's `DnaSequence` class is used to represent nucleotide sequences. Although SBOL provides terms to describe the relationships between sequence features and their sequence annotations, information about these sequence features is represented with RDF resources that represent the individuals of sequence features [13]. Therefore, individuals representing sequence features were created and mapped to the SBOL data model in SynthBiOnt.

Rule-based mapping is one way of presenting the data from SynthBiOnt in SBOL format. Example rules include "If a SynthBiOnt class inherits from Promoter, then it is associated with a DnaComponent individual in the SBOL representation" and "If a Promoter has Operator parts, then the DnaComponent associated with this Promoter has Operator annotations that are also DnaComponents" (Figure 4.8).

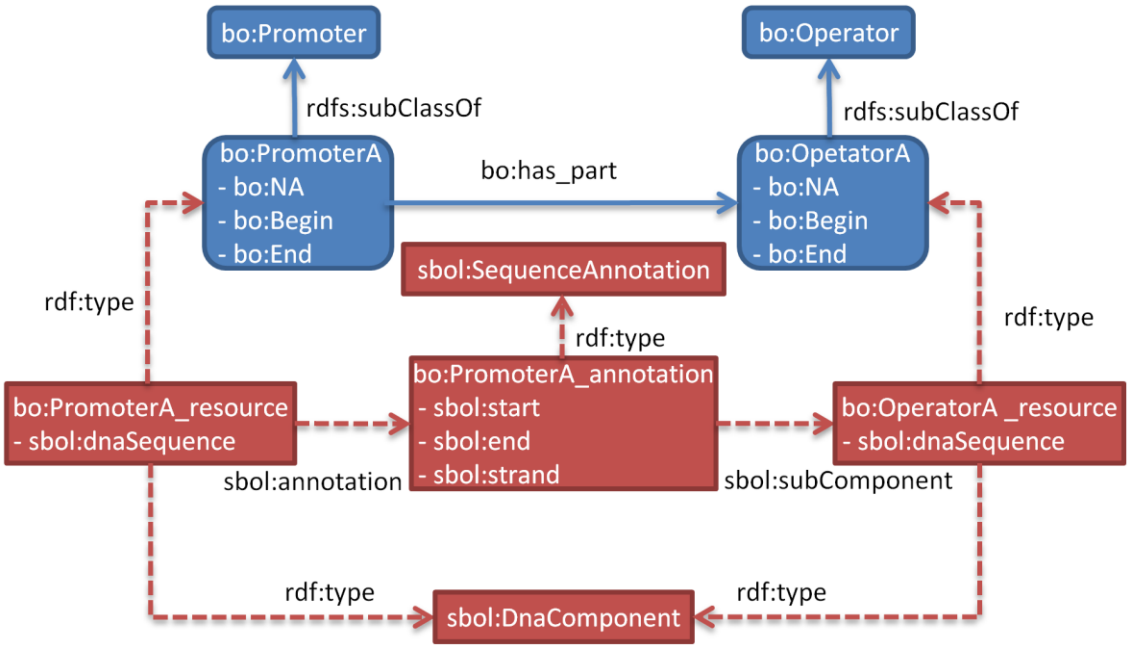


Figure 4.8: A representation of the mapping for a simple promoter that contains an operator. Blue boxes with round corners and straight lines represent ontology classes and their relationships. Red boxes and dashed lines represent SBOL resources and their relationships added.

Classes were mapped using five rules:

- If a class is a subclass of `Promoter`, `RBS`, `Operator`, `Shim`, `Terminator` or `CDS`, then there is an individual resource that has the type `sbol:DnaComponent`. Another type is the class from SynthBiOnt, for which the individual is created.
- The `sbol:dnaSequence` property identifies the resource that includes the nucleotide sequence of an SBOL individual using the `sbol:nucleotides` property. The sequence is extracted from the restriction on the `NA` property of the class from SynthBiOnt.
- A sequence feature individual that is part of a `DnaComponent` individual is also a type of `sbol:DnaComponent`.
- If a sequence feature class includes another sequence feature class, then the individual of the former has an `sbol:Annotation` resource describing the start and end locations of the annotation. The annotation resource's `sbol:subComponent` property identifies the individual of the latter class.
- The `start`, `end` and `strand` properties of an `sbol:Annotation` resource are inferred from the genome positions of the parent and child classes.

The rule-based mapping was implemented using SPARQL's CONSTRUCT queries [211].

4.4 SynthBiOnt: An ontology for synthetic biology

SynthBiOnt captures major aspects of the cell biology of *Bacillus subtilis* in a computationally-amenable form. SynthBiOnt was built using the BacillOndex dataset, which includes information from BacilluScope, DBTBS, KEGG, KEGG Expression, STRING, GO and GOA. The ontology was constructed using OWL semantics. The precise meaning defined for the relationships between OWL classes makes SynthBiOnt computationally amenable. The rich expressivity of OWL enables the construction of complex computational queries and automated reasoning across the integrated data.

SynthBiOnt includes 42259 OWL classes, with 41 object, 21 datatype and 26 annotation properties. There are 269,726 `SubClassesOf`, 386 `EquivalentClass`, 169 `DisjointClass`, and 274,003 `AnnotationAssertion` axioms. As the ontology conforms to the RDF and OWL standards, it can be manipulated using existing ontology editors such as Protégé[33], and information can be extracted using reasoners such as Pellet [228] and HermiT [227]. The ontology is also available as an RDF repository[34] to allow the querying of information using standard SPARQL queries. The base URI of the ontology is http://www.bacillondex.org. Table 4.2 shows the namespaces and prefixes used in the ontology.

Table 4.2: Namespaces and their prefixes used in the ontology.

Namespace prefix	Namespace
bo	http://www.bacillondex.org#
so	http://purl.org/obo/owl/SO#
go	http://purl.org/obo/owl/GO#
sbol	http://sbols.org/sbol.owl#
bods	http://www.bacillondex.org/cv#
boet	http://www.bacillondex.org/evidenceType#
owl	http://www.w3.org/2002/07/owl#
rdf	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#

The types of biological concepts, such as protein, CDS and pathway, in the ontology are the first level superclasses that are subclasses of `owl:Thing` in the ontology (Figure 4.9). The subclasses of these superclasses in the ontology represent biological concepts. For example, the CDS class for *ycbF*, `bo:28534` is shown in Figure 4.10. This class is a subclass of `bo:CDS` and other restriction classes. The superclasses that represent

[33] http://protege.stanford.edu
[34] http://sbol.ncl.ac.uk:8081/openrdf-workbench

sequence-based biological concept types, such as `bo:Promoter` and `bo:CDS`, are linked to terms from SO with the `is_a` relationship (Figure 4.11). The sequence classes are also represented using the SBOL model (Section 4.4.2). Therefore, an SBOL-aware tool can use the classes and their relationships to access nucleotide sequences or their compositional features.

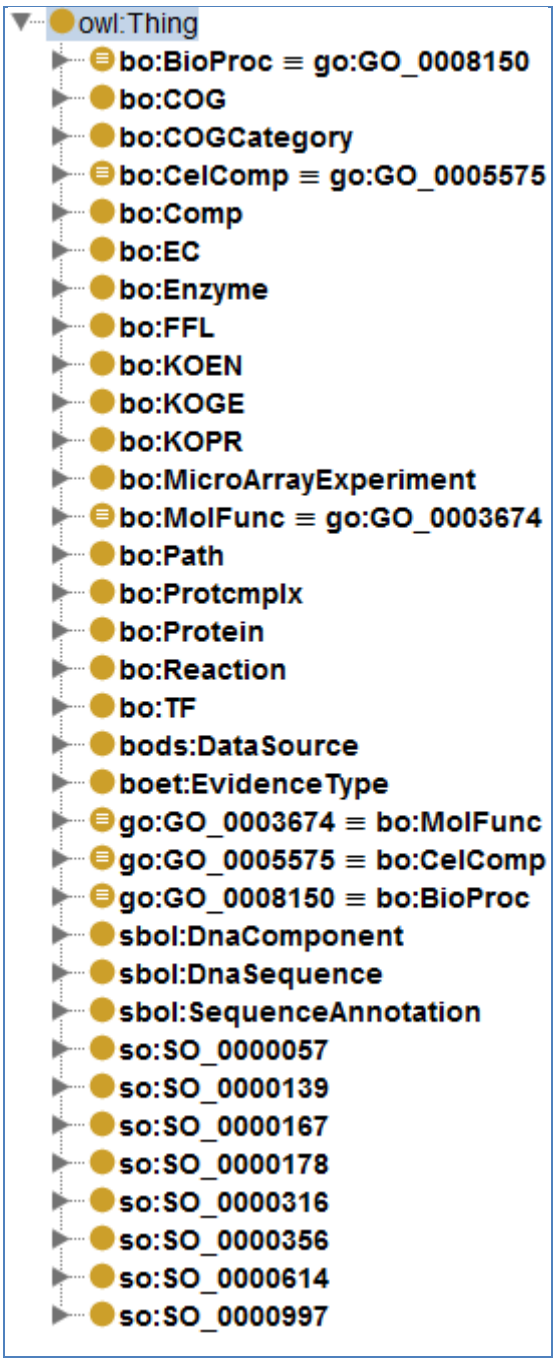


Figure 4.9: Classes that represent the types of biological concepts, and classes from GO, SO and SBOL ontologies included in SynthBiOnt.

Other molecules such as proteins, TFs, RNAs, enzymes, protein complexes, and compounds are also modelled as OWL classes. Classes representing proteins that act as

TFs are linked to the corresponding TF classes using the `owl:equivalentClass` property. Therefore, such protein and TF classes can be used interchangeably. This technique, known as semantic collapsing, can also be applied to RNAs that act as TFs. Enzymes are special proteins that catalyse reactions, and are therefore modelled as subclasses of the corresponding `Protein` classes.

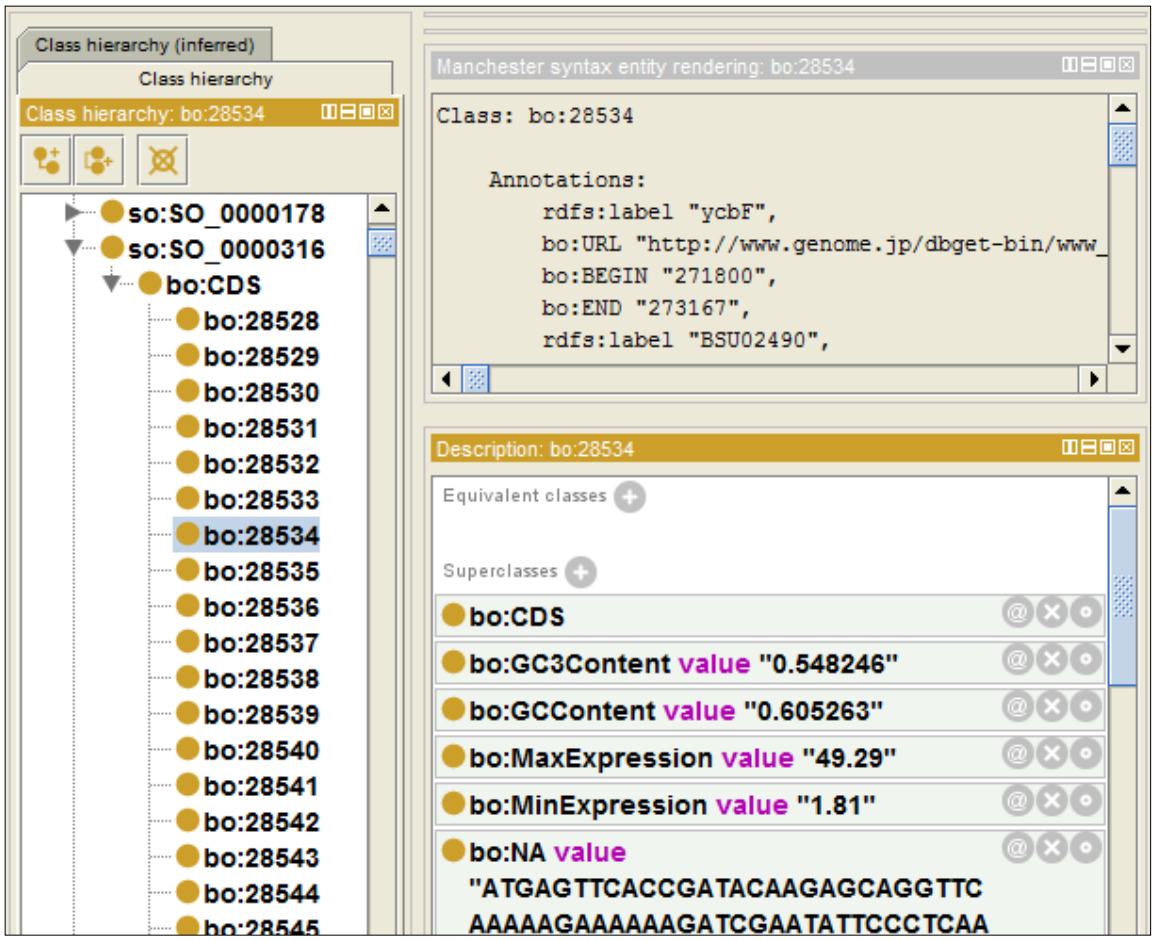


Figure 4.10: The `bo:28534` CDS class from SynthBiOnt in the Protégé ontology editor. This class is a subclass of `bo:CDS` and other restriction classes, used to model the common properties of its individuals. In addition, the class includes annotation properties which are shown in the Manchester OWL syntax.

Relationships between these molecules are also modelled in the ontology. Such relationships include the way in which proteins interact with each other, protein complexes are formed, enzymes interact with compounds, compounds are transported into cells and TFs bind to DNA sequences, all of which are captured with OWL semantics. These computationally amenable relationships between classes enable the computational reconstruction and analysis of networks. Figure 4.12 shows a subset of information about the relationships of the MntR protein as represented using the ontology classes. The information includes the molecular functions of the protein,

where it is located, a biological process that the protein participates in, the encoding CDS and its binding sequences.

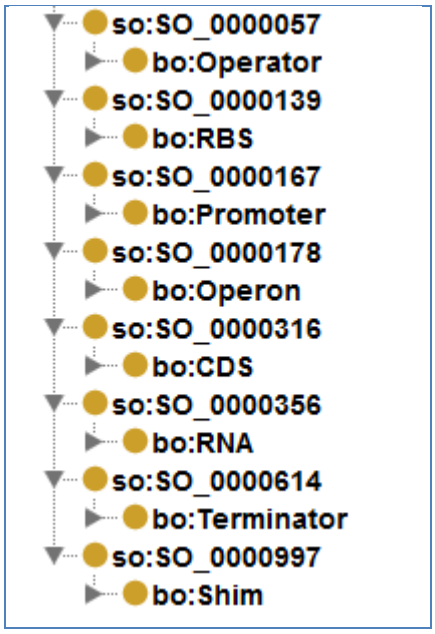


Figure 4.11: Sequence features from SynthBiOnt and their mapping to SO. Sequence-based superclasses are subclasses of the SO terms.

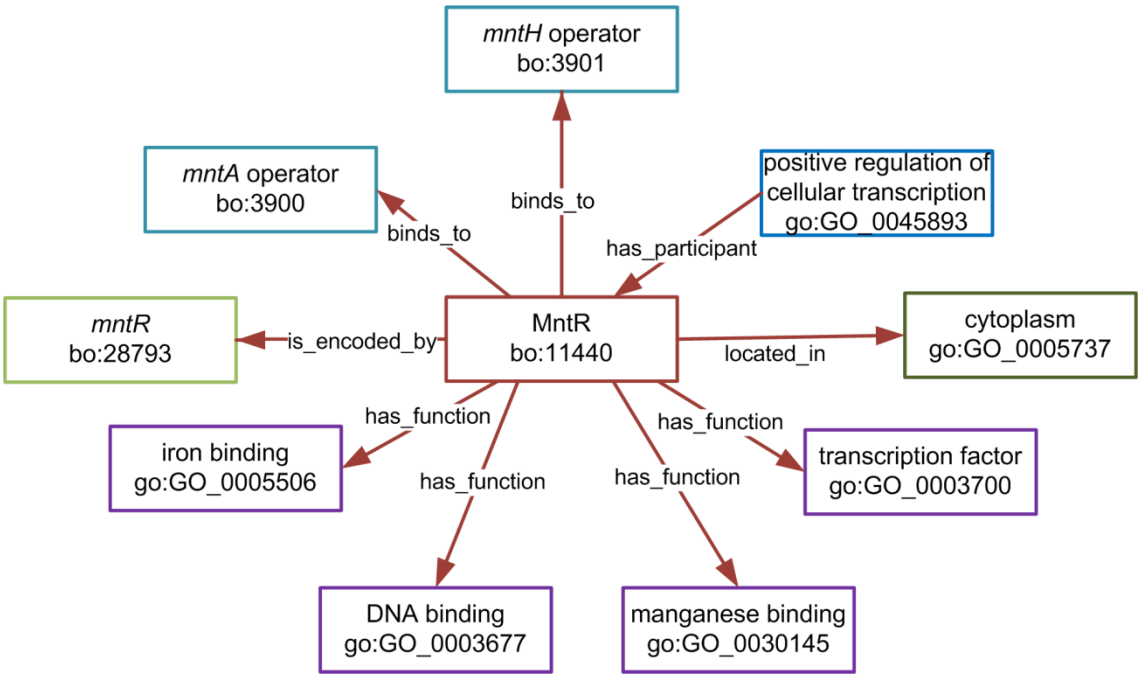


Figure 4.12: Example of the MntR protein's relationships in SynthBiOnt. The diagram shows a subset of the relationships that are modelled as restrictions on object properties, such as `is_encoded_by` and `has_function`. The information includes the molecular functions of the protein, where it is located, a biological process that the protein participates in, the encoding CDS and its binding sequences.

Individuals are not included in the ontology. However, members of specific classes

must meet necessary conditions, which are represented in the form of existential someValuesFrom (some) restrictions. For example, the ‘bo:encodes some bo:9793’ restriction on the bo:31594 CDS class indicates that all of the individuals of this class encode the protein identified by bo:9793. In addition, the hasValue (value) restrictions represent the values that individuals can have for the specified datatype properties such as GC content, TF family and product type. For example, the ‘bo:GCContent value "0.525229"’ restriction on the bo:31594 CDS class restricts all individuals to have 0.525229 as the value of the GCContent property. These object and datatype properties provide exact meanings for the relationships between classes, facilitating computational access. The classes also contain human-readable annotation properties such as name, description, data source and evidence type.

Classes that are used to classify or place restrictions on classes representing physical entities include enzyme classifications, KEGG orthologs enzymes, molecular functions, biological processes, cellular compartments, and COG classes and categories. SynthBiOnt also includes classes for reactions, pathways, microarray experiments, feed-forward loops (FFLs), data sources and evidence types.

The classes MolFunc, CelComp and BioProcess are equivalent classes to the classes of the GO’s molecular_function (GO_0003674), biological_process (GO_0008150), and cellular_component (GO_0005575). In addition to the GO classes that have annotations for *B.subtilis*, the ontology includes the full hierarchy of these terms. Hence, the ontology can be queried using more generic parent terms. For example, GO_0006402 and GO:0016075 refer to mRNA and rRNA degradation biological processes. By using the parent term GO_0006401, all the proteins that are involved in RNA breakdown can be retrieved or, by using a higher parent term GO_0008152, the proteins involved in metabolism can be retrieved (Figure 4.13). SynthBiOnt can be used to computationally answer different types of questions.

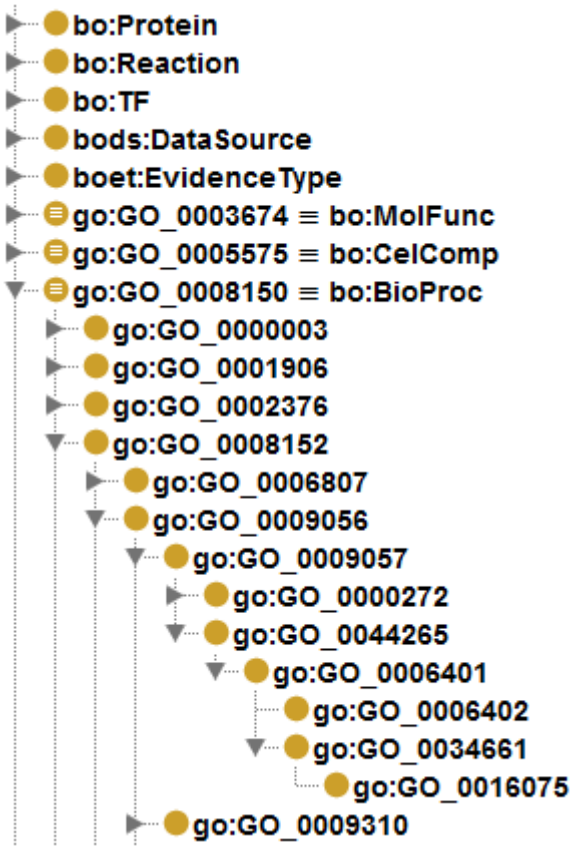


Figure 4.13: The GO class hierarchy was added for GO classes that have annotations for *B. subtilis*. Therefore, parent terms can also be used to retrieve information about child terms. For example, the figure displays the hierarchy for the `GO_0016075` 'rRNA degradation' biological process class. The parent term `GO_0006401` can be used to refer to all biological processes related to RNA degradation.

The scope of ontologies can be identified with a set of questions that are called *competency questions* [290]. Whether the ontologies contain enough detailed information can then be tested using these questions, which do not have to be exhaustive and can be written informally. SynthBiOnt was designed to answer competency questions that are of interest in the design of synthetic biological systems. Examples of competency questions and their corresponding answers, as revealed by the ontology, are listed below:

- Which parts are SigmaA type promoters?

  The SigmaA sigma factor is the `TF` with the `conceptAccession` of 'BSU25200' and `binds to Promoters`, which can be identified as SigmaA type promoters.

- Which promoters are constitutive?

  SigmaA `Promoters` that do not have any `Operators` can be constitutive promoters.

- Which parts can be used as inducible promoters?

Operators have regulation type restrictions to indicate whether they are used positively or negatively in regulating gene expression. A Promoter with one Operator part that has the 'Positive' regulation type restriction is an inducible promoter.

- Which parts are SigmaA type inducible promoters?

  Promoters that are both subclasses of SigmaA and inducible type promoters are candidate promoters.

- Which parts are regulated by the MntR TF?

  MntR binds to the *mntA* and *mntH* Operators.

- What are the nucleotide sequences that the Spo0A TF binds to?

  Operators that are bound by the Spo0A TF have restrictions on the nucleotide sequence property.

- Which parts encode for the two-component systems (TCSs)?

  These parts are CDSs encoding for Proteins that have functions of 'kinase activity' and 'response regulator activity'. The GO classes with the conceptAccessions of GO_0000155 and GO_0000156 respectively represent these functions.

- Which parts can be used to upregulate the production of ammonium?

  The Compound 'Ammonia' with the conceptAccession of 'C00014' is produced by the Reaction 'RN:R00131', which consumes the Compound 'Carbamide' (C00086). Carbamide is produced by a Reaction that is catalysed by an Enzyme, which is a subclass of a Protein encoded by the *argI* CDS with the conceptAccession of 'BSU40320'.

- Which pathways should be targeted for the over-production of ammonium?

  Ammonium is produced by Reactions that are member of 'Arginine and proline metabolism' and 'Purine metabolism' Pathways.

- How can the Spo0A protein, the master regulator of sporulation, be phosphorylated to trigger sporulation?

  Spo0A is phosphorylated by the KinC and Spo0B Proteins. The Spo0B protein is phosphorylated by Spo0F Protein which is further phosphorylated by the KinA and KinB Proteins.

- What are the possible NAND gate promoters?

  NAND gate promoters can be searched for in the list of Promoters that have two Operator parts with 'Negative' regulation type restrictions.

- Which parts should be upregulated to increase Mannose compound transport to the cells?

  The 'D-Mannose 6-phosphate' compound with the `conceptAccession` of 'C00275' `interacts with` a `ProteinComplex`. ManP and LevF `Proteins` are `part of` this complex.

4.4.1 Applying SynthBiOnt to the automated classification of information for synthetic biology

SynthBiOnt is based on Description Logic (DL) and hence can be used for automated reasoning [144]. Several different types of information were derived from the ontology, using an automated reasoning approach. When the ontology is submitted to a reasoner the class hierarchy between the defined and primitive classes is computed. Defined classes that provide both necessary and sufficient conditions were used to classify operator, promoter and CDS parts. SynthBiOnt can also be queried directly using SPARQL or OWL- friendly queries [230].

The classification of operators includes information about whether an operator is involved in negative or positive regulation. In SynthBiOnt, operator classes have `hasValue` restrictions on the 'regulator type' (RT) property. To classify the binding sites for inhibitors or activators, operators that have an RT value of 'Negative' or 'Positive' were selected respectively (Figure 4.14). Only the operators with known nucleotide sequences were included. In total, 333 repressor (Table D-17) and 222 activator sites (Table D-18) were identified.

Promoters were classified according to their regulation types. Classes were defined for inducible, repressible and constitutive promoters. Promoters that act as two-input logic gates were also identified. In addition, a range of classes were defined to classify promoters according to their sigma factors.

Promoters that act as one-input amplifiers [80] were represented with the `InduciblePromoter` defined class. These promoters were identified as having one binding site for an activator TF, which is modelled with a cardinality restriction. Another cardinality restriction was added to specify that there is *only* one binding site (Figure 4.16). In total, 51 promoters (Table D-1) were identified as `InduciblePromoter`. A subset of these promoters is shown in Figure 4.15.

```
Class: NegativelyRegulatedOperator
    Annotations:
        rdfs:label "Negatively regulated operator",
        rdfs:comment "Operator site for a repressor"
    EquivalentTo:
        Operator
         and (NA some rdf:PlainLiteral)
         and (RT value "Negative")
    SubClassOf:
        Operator

Class: PositivelyRegulatedOperator
    Annotations:
        rdfs:label "Positively regulated operator",
        rdfs:comment "Operator site for an activator"
    EquivalentTo:
        Operator
         and (NA some rdf:PlainLiteral)
         and (RT value "Positive")
    SubClassOf:
        Operator
```

Figure 4.14: The class definitions for operator classification in the Manchester OWL syntax. Operators with known sequences are classified according to their regulator type restrictions.

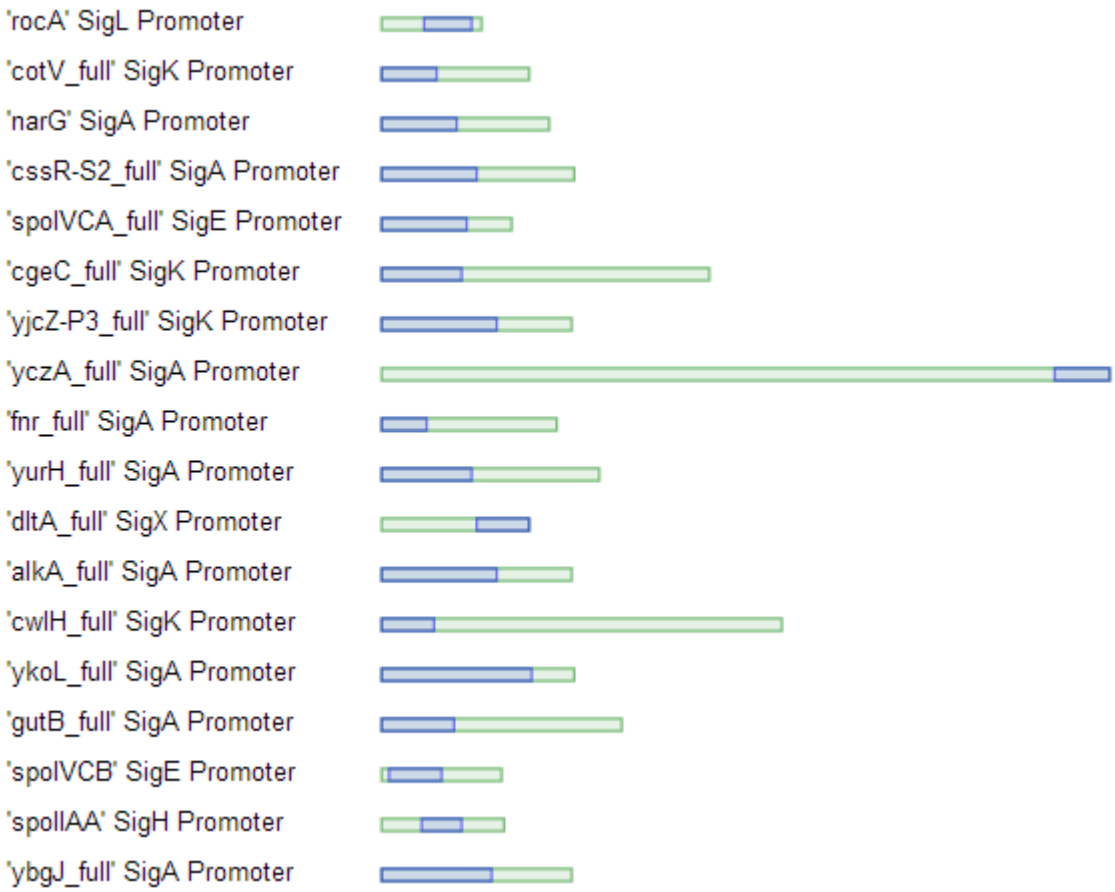


Figure 4.15: Some of the inducible promoters classified. The outer green rectangles and inner blue rectangles represent the promoters and TF binding sites respectively. The length of a box is proportional to the corresponding promoter's sequence length.

Similarly, 85 promoters (Table D-2) that act as one-input inverters were classified as `RepressiblePromoter` (Figure 4.17). This class definition specifies that a repressible promoter has only one binding site which is for a repressor TF.

```
Class: InduciblePromoter
    Annotations:
        rdfs:label "Inducible promoter",
        rdfs:comment "Promoter with an activator site"
    EquivalentTo:
        Promoter
         and (has_part exactly 1 Operator)
         and (has_part exactly 1 PositivelyRegulatedOperator)
    SubClassOf:
        Promoter
```

Figure 4.16: The inducible promoter class definition. Promoters with one operator for an activator are classified as inducible promoters.

```
Class: RepressiblePromoter
    Annotations:
        rdfs:label "Repressible promoter",
        rdfs:comment "Promoter with a repressor site"
    EquivalentTo:
        Promoter
         and (has_part exactly 1 NegativelyRegulatedOperator)
         and (has_part exactly 1 Operator)
    SubClassOf:
        Promoter
```

Figure 4.17: The repressible promoter class definition. Promoters with one operator for a repressor are classified as repressible promoters.

The `InduciblePromoterWith2Operators` class was defined to classify 15 promoters (Table D-3) with two binding sites that are only for activators (Figure 4.18). Similarly, the `RepressiblePromoterWith2Operators` class was defined to classify 25 promoters (Table D-4) with two binding sites that are only for repressors (Figure 4.19). In addition, the `RepressibleInduciblePromoterWith2Operators` class was defined for promoters with two binding sites, one for a repressor and one for an activator (Figure 4.20). However, promoters from the ontology could not be classified using this class definition.

```
   Promoter
      and (has_part exactly 2 Operator)
      and (has_part exactly 2 PositivelyRegulatedOperator)
```

Figure 4.18: The OWL expression for the `InduciblePromoterWith2Operators` class which is used to classify promoters with two activator sites only.

```
Promoter
    and (has_part exactly 2 Operator)
    and (has_part exactly 2 NegativelyRegulatedOperator)
```

Figure 4.19: The OWL expression for the `RepressiblePromoterWith2Operators` class which is used to classify promoters with two repressor sites only.

```
Promoter
    and (has_part exactly 2 Operator)
    and (has_part exactly 1 NegativelyRegulatedOperator)
    and (has_part exactly 1 PositivelyRegulatedOperator)
```

Figure 4.20: The OWL expression to classify promoters with one activator and one repressor site.

The positions of the binding sites on promoters and the cooperativity in TF binding results in different logic gate behaviours. A promoter with two activator sites can function as an AND or an OR gate [78, 189, 297]. Conversely, a promoter with two repressor sites can function as a NAND or a NOR gate [78, 293, 297]. The `ANDGatePromoter` and `ORGatePromoter` classes in the ontology represent promoters that work as AND and OR gates respectively, and are subclasses of the `InduciblePromoterWith2Operators` class (Figure 4.21). Similarly, `NANDGatePromoter` and `NORGatePromoter` classes in the ontology represent promoters that work as NAND and NOR gates respectively, and are subclasses of the `RepressiblePromoterWith2Operators` class (Figure 4.21). The `ANDNGatePromoter` class represents promoters that work as ANDN gates [293, 297]. This class is a subclass of the `RepressibleInduciblePromoterWith2Operators` class (Figure 4.21).

Promoters were also classified based on their sigma factors. For example, `SigAPromoter` is a promoter to which the TF with the accession 'BSU25200' binds. Such a promoter may be a core SigA promoter or a composite promoter that includes a core SigA promoter (Figure 4.22). Similarly, classes were defined for other sigma factors. In total, 465 SigA, 67 SigB, 33 SigD, 97 SigE, 30 SigF, 63 SigG, 31 SigH, 1 SigI, 71 SigK, 10 SigL, 8 SigM, 0 SigV, 39 SigW, 16 SigX, 2 SigY, 0 SigZ, 1 YlaC type promoters were classified (Table 4.3).

Twenty-one promoters could not be classified since no sigma factor data were available in BacillOndex for these promoters.

```
Class: ANDGatePromoter
    SubClassOf:
        InduciblePromoterWith2Operators

Class: ORGatePromoter
    SubClassOf:
        InduciblePromoterWith2Operators

Class: NORGatePromoter
    SubClassOf:
        RepressiblePromoterWith2Operators

Class: NANDGatePromoter
     SubClassOf:
        RepressiblePromoterWith2Operators

Class: ANDNGatePromoter
     SubClassOf:
        RepressibleInduciblePromoterWith2Operators
```

Figure 4.21: Two-input logic gate promoter definitions.

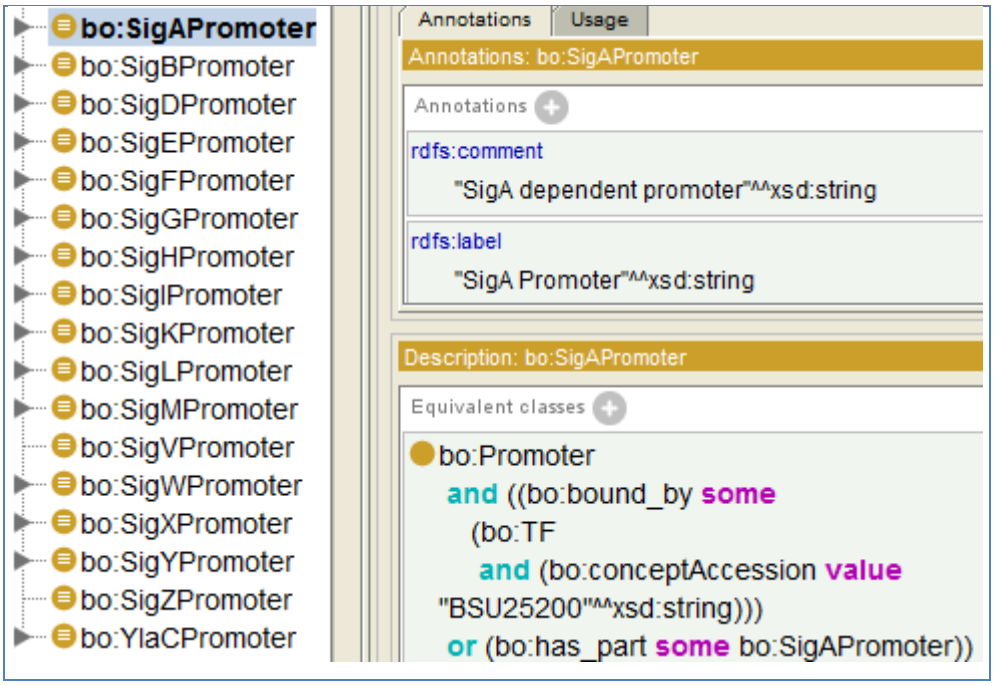


Figure 4.22: Classification of promoters based on sigma factors. The right pane displays the definition of the `SigAPromoter` class used for the classification of SigA promoters.

Table 4.3: Number of sigma factors classified. Full lists of these sigma factors can be found in Appendix D as tables which are specified in the 'Table reference' column below.

Sigma factor	Count	Table reference
SigA	465	Table D-6
SigB	67	Table D-7
SigD	33	Table D-8
SigE	97	Table D-9
SigF	30	Table D-10
SigG	63	Table D-11
SigH	31	Table D-12
SigI	1	Table D-16
SigK	71	Table D-13
SigL	10	Table D-16
SigM	8	Table D-16
SigW	39	Table D-14
SigX	16	Table D-15
SigY	2	Table D-16
YlaC	1	Table D-16

Constitutive promoters are only dependent on RNA polymerases, and their definition does not rely upon information about transcriptional regulation. In *B. subtilis*, SigA promoters without any TF binding sites are constitutive promoters. To classify constitutive SigA promoters, the `ConstitutiveSigAPromoter` class was defined as a subclass of both `SigAPromoter` and `ConstitutivePromoter` classes. In addition, a restriction class was added to specify that these promoters cannot have any operators (Figure 4.23). As a result, 311 constitutive promoters were identified.

```
Class: ConstitutivePromoter
    Annotations:
        rdfs:comment "Constitutive Promoter",
        rdfs:label "Constitutive Promoter"
    SubClassOf:
        Promoter

Class: ConstitutiveSigAPromoter
    Annotations:
        rdfs:comment "Constitutive SigA Promoter",
        rdfs:label "Constitutive SigA Promoter"
    EquivalentTo:
        SigAPromoter
         and (not (has_part some Operator))
    SubClassOf:
        SigAPromoter,
        ConstitutivePromoter
```

Figure 4.23: Constitutive promoter definitions in SynthBiOnt. Constitutive SigA promoters are SigA promoters that do not include any operator binding sites.

SynthBiOnt includes a variety of biological concepts, attributes and relationships that can be used to automate the identification of CDS parts. COG numbers, and the GO molecular function, biological process and cellular location terms can be used to classify gene products, and hence to find the encoding CDSs. Furthermore, classes such as `RNA`, `TF` and `Enzyme` provide more explicit meanings for the roles of gene products. The restrictions on the protein classes based on properties such as biological process and role classification can also be used to find the encoding CDSs.

Many synthetic biology projects focus upon the use of regulators such as transcriptional activators and repressors [70, 107, 297-299], and sensory systems such as TCSs [91, 104, 122, 253, 286]. Similar to TFs, TCSs have the potential to be used as modular biological parts [91]. These systems can be introduced into other host cells that do not have any analogous systems, hopefully providing well-isolated circuits. To contribute to the list of available CDS parts in these categories, CDSs were classified as transcriptional activator-, transcriptional repressor-, kinase- and response regulator-encoding CDSs.

In order to classify CDS parts that encode for transcriptional repressors, the `RepressorEncodingCDS` class was defined as a `CDS` that `encodes` for a `Protein` which `binds to` at least one repressor site (Figure 4.24). These repressor sites have known nucleotide sequences. Therefore, reliable parts that encode and provide binding sites can be retrieved as pairs. Similarly, the `ActivatorEncodingCDS` class was defined as a `CDS` that `encodes` for a `Protein` which `binds to` at least one activator site (Figure 4.25). Using the reasoners, 44 activator- and 55 repressor-encoding CDSs were identified. In the ontology, the binding restrictions are modelled for TF classes. However, because the TF classes are linked to their protein counterparts using the `owl:equivalentClass` axioms, TF and Protein classes can be used interchangeably in the class definitions.

```
CDS and
     (encodes some (Protein and
        (bi_to some NegativelyRegulatedOperator)))
```

Figure 4.24: The OWL expression for the `RepressorEncodingCDS` defined class. A `CDS` that `encodes` for a `Protein` binding to at least one repressor site is classified as `RepressorEncodingCDS`.

```
CDS and
     (encodes some (Protein and
        (bi_to some PositivelyRegulatedOperator)))
```

Figure 4.25: The OWL expression for the `ActivatorEncodingCDS` defined class. A `CDS` that `encodes` for a `Protein` binding to at least one `activator site` is classified as `ActivatorEncodingCDS`.

CDSs that encode TCS kinase and response regulators are classified based on the relevant GO terms. A `CDS` that `encodes` a `Protein` which `has` the `GO_0000155` molecular function ('two-component system sensor activity') is classified as a `KinaseEncodingCDS` class (Figure 4.26). Similarly, a `CDS` that `encodes` a `Protein` which `has` the `GO_0000156` molecular function ('two-component response regulator activity') is classified as `ResponseRegulatorEncodingCDS` class. In total, 40 kinase- and 38 response regulator-encoding CDSs were identified.

```
CDS and
     (encodes some  (Protein and
        (has_function some go:GO_0000155)))
```

Figure 4.26: The OWL expression for the `KinaseEncodingCDS` defined class. A `CDS` that `encodes` for a `Protein` that has function `go:GO_0000155` is classified as `KinaseEncodingCDS`.

In these examples, SynthBiOnt was used to classify biological parts for the design of transcriptional regulatory networks. Queries were constructed in the form of OWL classes, which provide logical definitions of their subclasses that represent biological parts. Subclass membership was inferred by existing reasoners using the definitions provided in these queries. As a result, operators that can be used as binding sites for repressors or activators were identified. Promoter classes were assigned different types of class membership based on information about the type of transcriptional regulation and sigma factors. Examples for classification also included the identification of CDSs that encode for activators, repressors, kinases and response regulators. This list is not exhaustive and can be extended. Although SynthBiOnt is computationally-amenable, the representation of information about biological parts in widely-accepted formats would increase the use of the ontology.

4.4.2 Incorporating SBOL into SynthBiOnt

Various tools that can be used to facilitate the large-scale design of biological systems have been developed [22, 32, 50]. However, in order to enable the electronic exchange of data between these tools, part catalogues and synthesis companies, biological parts

and genetic circuits constructed from these parts must be represented in standard formats. SBOL was developed to address this issue [63]. The core data model is implemented using OWL, and hence can be incorporated into SynthBiOnt. This approach would make SynthBiOnt accessible by a range of tools that can understand the SBOL format.

Therefore, to enable the electronic exchange of information between tools concerning sequence features and details about the way in which they can be combined, the sequence features were also represented in the SBOL format. Promoters, CDSs, RBSs, terminators, shims and operators, together with their relevant properties, were mapped to the SBOL's data model using rule-based mapping (Appendix C). SynthBiOnt contains 7,754 standard `DnaComponent` parts that can be exchanged using SBOL.

For each OWL class representing a sequence feature, an individual of type `sbol:DnaComponent` was created (Figure C-1). The SBOL model enforces the rule that each `DnaComponent` resource must be a type of sequence feature from SO. SO-based superclasses included in SynthBiOnt are used to infer these types. The names, descriptions and nucleotide sequences of the resources were extracted from the OWL classes and stored using the `rdfs:label`, `rdfs:comment` and `sbol:nucleotides` properties respectively.

Relationships of type `has_part` were used to create SBOL sequence annotations. Differences between the genome positions of sequence features linked by a specific relationship were used to calculate the `sbol:bioStart` and `sbol:bioEnd` properties of the sequence annotations. The `sbol:subComponent` property was used to identify the sequence feature resources used for annotation.

Figure 4.27 shows an example of a promoter in SBOL format. The resource is of both types `sbol:DnaComponent` and `so:SO_0000167` ('promoter sequence'), which are mandatory in the SBOL model. In addition, the resource is also of type `bo:2685` from SynthBiOnt. The promoter has an annotation identified by the `sbol:annotation` property.

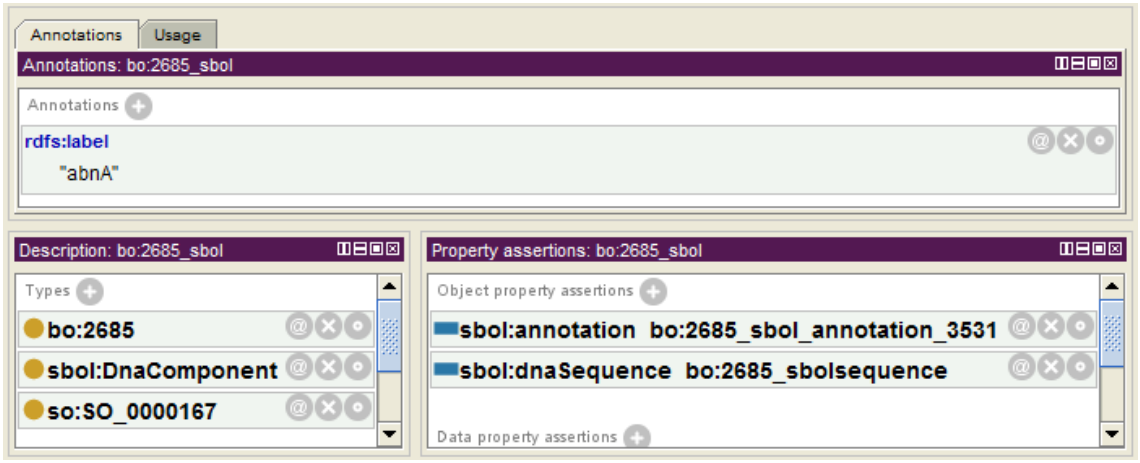


Figure 4.27: An SBOL promoter resource, a type of `sbol:DnaComponent`, `so:SO_0000167` and `bo:2685`.

The sequence annotation resource has data triples indicating the start, end and strand of the annotated sequence feature (Figure 4.28). In addition, the sequence feature `bo:3531_sbol` is accessible by the `sbol:subComponent` property. This feature is an operator of types `sbol:DnaComponent`, `so:SO_0000057` and `bo:3531`.

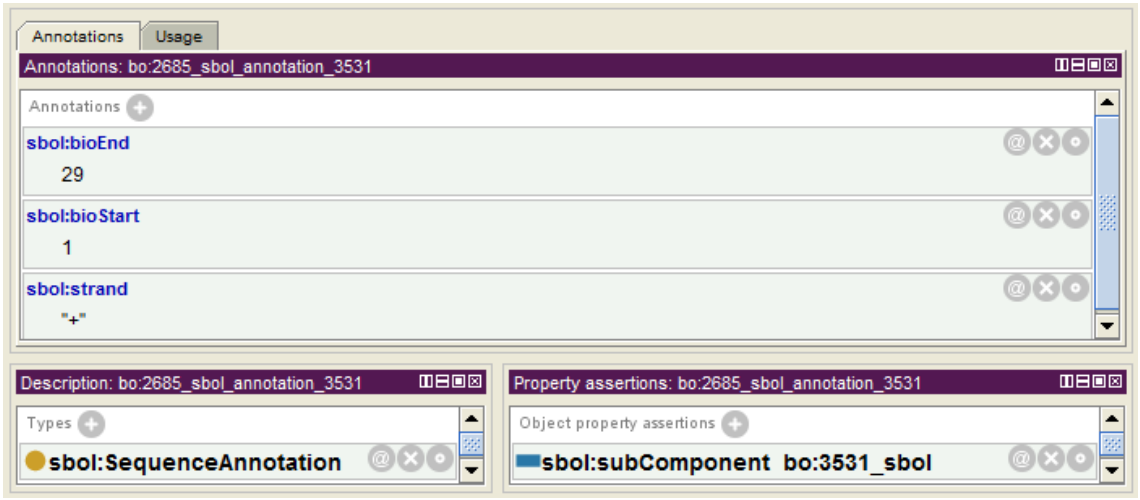


Figure 4.28: The SBOL sequence annotation for the promoter example in Figure 4.27. The promoter is annotated with an operator sequence feature. The start, end and strand of the annotation are also recorded.

In addition, the `sbol:dnaSequence` property of the promoter identifies the nucleotide sequence. The sequence can be accessed by the `sbol:nucleotides` property of this resource. This property is of type `sbol:DnaSequence`.

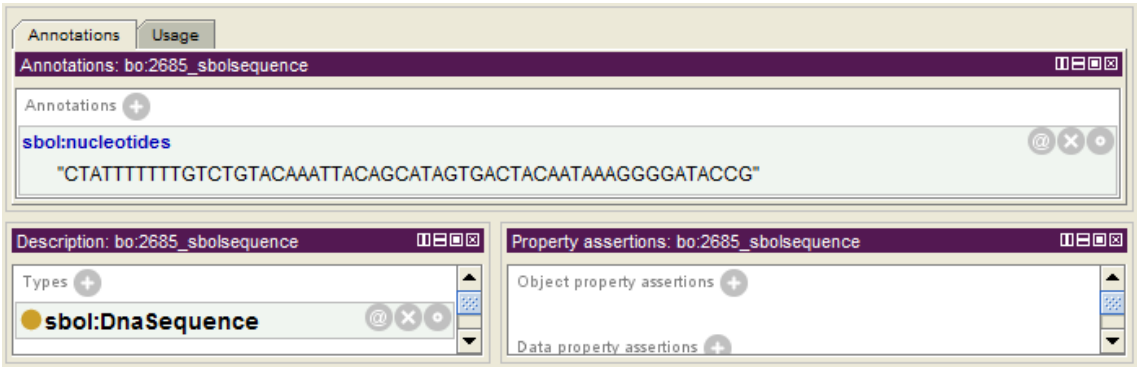


Figure 4.29: The DNA sequence resource of the example promoter. Nucleotide information is given by the `sbol:nucleotides` property.

4.4.3 Querying SynthBiOnt

In addition to its use in defining classes for automated reasoning, SynthBiOnt can be queried computationally or manually. For example, the Terp query language can be used to write OWL-friendly queries that use the SPARQL syntax [230]. These queries are similar to OWL class definitions written in the Manchester syntax [223]. For example, the Terp query in Figure 4.30 retrieves the operators to which repressors bind. A similar class definition was used to classify repressor sites (Figure 4.14). Initially, the Pellet reasoner is run on the ontology. Subsequently, the Terp query is executed programmatically to get the result set.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX bo: <http://www.bacillondex.org#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>

SELECT distinct ?operator
WHERE {
      ?operator rdfs:subClassOf ( bo:Operator and ( bo:RT value
"Negative") and (bo:NA some rdf:PlainLiteral)) .
}
```

Figure 4.30: The Terp query that selects the operators for inhibitors. An anonymous class, which defines these operators, is specified with the `rdfs:subClassOf` property in the query and resembles the corresponding defined class in the Manchester syntax. As with SPARQL queries, the prefixes are included at the top.

The RDF graph of the ontology was stored in a publicly accessible Sesame[35] RDF endpoint. This framework allows the querying of the ontology using SPARQL queries both manually and computationally. Classes in the ontology are referred to using their unique URIs. When queries are submitted manually, the results display URIs as HTML links to the triples about the resources. This section describes several examples of

[35] http://www.openrdf.org/

SPARQL queries used to extract information about biological parts from SynthBiOnt.

The query in Figure 4.31 is the SPARQL equivalent of the Terp query in Figure 4.30. SPARQL queries can be run directly on the triple store without the need to run any reasoner; however, such queries can become complex due to the explicit use of the RDF graph.

```
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  bo:   <http://www.bacillondex.org#>
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
SELECT DISTINCT  ?Operator
WHERE
{
    ?Operator
       rdfs:subClassOf         bo:Operator ;
       rdfs:subClassOf         ?NegativelyRegulatedRestriction .
    ?NegativelyRegulatedRestriction
       rdf:type                owl:Restriction ;
       owl:onProperty          bo:RT ;
       owl:hasValue            ?RegulationType .
    FILTER ( ?RegulationType = "Negative" )
}
```

Figure 4.31: The SPARQL query that selects operators for inhibitors. SPARQL queries can be more complex than OWL-like queries.

Figure 4.32 shows a query to retrieve the nucleotide sequence of the 'cadA' `DnaComponent`. Initially, the `DNAComponent` is filtered by name. The `sbol:dnaSequence` property is used to identify the `DnaSequence` resource that holds the nucleotide sequence.

```
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  bo:   <http://www.bacillondex.org#>
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX  sbol: <http://sbols.org/sbol.owl#>

SELECT DISTINCT ?DnaComponent ?Name ?NA
WHERE
{
    ?DnaComponent rdf:type          sbol:DnaComponent ;
                  sbol:dnaSequence  ?DnaSequence ;
                  rdfs:label        ?Name.
     ?DnaSequence rdf:type          sbol:DnaSequence;
                  sbol:nucleotides  ?NA .
    FILTER (?Name="cadA")
}
```

Figure 4.32: An example of a SPARQL query to get the nucleotide sequence of the 'cadA' `DnaComponent`.

The relationships of the 'cadA' `DnaComponent` can also be investigated using SPARQL. The example in Figure 4.33 uses CONSTRUCT queries to get the results in the form of subject-predicate-object triples. In the triples, the subject is the URI that identifies the queried DnaComponent, the predicate is the object property upon which the restriction is placed, and the object is a URI of the classes that are the values of the `someValuesFrom` restrictions. As a result, the relationships such as the encoded protein and the inhibiting TF are returned (Figure 4.34).

```
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  bo:   <http://www.bacillondex.org#>
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX  sbol: <http://sbols.org/sbol.owl#>
PREFIX  so:<http://purl.org/obo/owl/SO#>

CONSTRUCT
{
    ?DnaComponent  ?Relation ?Concept .
}
WHERE
{
    ?DnaComponent rdf:type sbol:DnaComponent ;
                  rdf:type so:SO_0000316 ;
                  rdf:type ?RestrictionClass ;
                  rdfs:label ?Name.
    ?RestrictionClass ?r ?o .
    ?o            rdf:type owl:Restriction;
                  owl:someValuesFrom  ?Concept;
                  owl:onProperty ?Relation .

   FILTER ((?Name="cadA") &&
      (regex(str(?BacillOndexClass),"http://www.bacillondex.org" ))
   )
}
group by ?DnaComponent  ?Relation ?Concept
```

Figure 4.33: The SPARQL CONSTRUCT query to retrieve the relationships involving the 'cadA' `DnaComponent`. The type information of the resource from the ontology is used to retrieve the relationships using the RDF graph of the ontology.

Subject	Predicate	Object
bo:28826_sbol	bo:in_by	bo:6925
bo:28826_sbol	bo:min_expressed_in	bo:27039
bo:28826_sbol	bo:max_expressed_in	bo:27057
bo:28826_sbol	bo:encodes	bo:10997

Figure 4.34: Relationships of the 'cadA' `DnaComponent` to classes from SynthBiOnt.

4.5 Discussion

The ontology discussed in this chapter represents domain knowledge about *B. subtilis* in a computationally-amenable form. A range of biological concepts and their relationships were included. Automated queries that were not possible using the individual data sources, can be implemented using this ontology. In addition, finding or classifying information can be automated using standard knowledge representation languages such as OWL and RDF. Such automation was demonstrated to classify information about hundreds of biological parts such as promoters, operators and CDSs. This approach is useful for large-scale synthetic biology which requires computational access to information.

As the amount of data and number of tools available for synthetic biology increases, it becomes increasingly important for standards to be adopted by the community [63, 300]. It is desirable that the catalogues do not duplicate data; instead, they should be linked to each other [185]. When the same data are stored in different catalogues, the meaning of data should be specified semantically using existing standards. Techniques and standards from Semantic Web technologies can be borrowed to address issues of the representation of data and communication between repositories [13]. In this chapter, these technologies and automated reasoning techniques [213] including RDF, OWL, SPARQL and DLs were used for these purposes. Such Semantic Web technologies enable the exchange and linking of data via unique URIs. In addition, tools that support these standards can be of benefit in using SynthBiOnt; for example, Protégé to view the ontology, a Sesame RDF endpoint to store and query the ontology, and HermiT and Pellet reasoners to automate the classification of information.

Developing data standards is not a trivial exercise. Standards should be easy to use, involve the community and be supported by tools [301]. SBOL is still in the early stages of becoming a data exchange language. Applications that use SBOL can widen its usage within the synthetic biology community as an adopted language. This language is important in representing information about biological parts computationally, and hence it can facilitate the effective reuse in automated approaches of several different types of tools that have already been developed for synthetic biology, as well as information from part catalogues.

SynthBiOnt was mapped to SBOL as well as other external standards such as the SO and GO. The use of the SO and SBOL provides standardised access to sequence-based classes. GO is one of the most well-known biological ontologies and uses

standard terminology [302]. The use of widely accepted ontology terms facilitates the mapping of data models from different repositories for the purposes of data integration [303]. Therefore, these terms may facilitate federated queries to be run on different repositories by providing knowledge that can be reused. With initiatives for standards continuing, this method may help in unifying the modelling of data for synthetic biology.

SynthBiOnt models in detail small DNA fragments such as promoters and CDSs, and their relationships, in order to automate the extraction of parts for synthetic biology. Existing ontologies such as Protein Ontology [219], Phosphatase Ontology [144] and Cell Cycle Ontology [146] focus upon the modelling of gene products which are then mapped to their encoding genes. The UniProt protein database includes millions of protein annotations using GO terms [217]. However, in synthetic biology applications, smaller DNA fragments with defined nucleotide sequences are often used, rather than genes. A desired biological function has to be linked to individual sequence features. The ontology described in this chapter enables the automated analysis of sequence features in detail, not just at the gene level but using smaller DNA components, as well as their global impact on whole cells.

The current SBOL model focuses upon the exchange of sequence-based descriptions of parts [63]. However, for the automated design of large genetic circuits, details of the interactions between these parts must also be made accessible to computational tools [21]. In SynthBiOnt, biological constraints are encoded through the use of machine-accessible OWL restrictions, which are also available to reasoners for the automated classification of parts according to their relationships and biological properties.

4.5.1 Automated reasoning

SynthBiOnt captures domain knowledge about *B. subtilis* using DL, and can be submitted to existing reasoners. Automated reasoning was used here to automatically classify sequence features such as promoters, operators and CDSs. Hence, the ontology is also a model for how automated reasoning can be used to inform synthetic biology.

The classification of promoters from the *B. subtilis* genome is valuable, for example, in creating catalogues of promoters that already exist and are proven to work. By looking at the relationships of promoters and operators, a human domain expert can deduce whether a promoter is inducible or repressible. SynthBiOnt makes knowledge about such relationships machine accessible, permitting the automation of the large-scale classification of promoters. Using existential and universal restrictions on

promoter classes, promoters were classified using data about their associated operators and sigma factors. For example, constitutive, inducible, repressible and two-input promoters were identified. Two-input promoters can be further mined to search for promoters that can be used as logic gates, such as NAND, NOR, AND and OR gates. Due to the unknown binding cooperativity of TFs, the exact logic output will still have to be experimentally validated [90, 189]. However, rules such as how close operator sites are to each other can be used to infer whether or not TFs interact when they bind to promoters [278].

Although the identification of regulatory elements has been the focus of the work reported in this chapter, there is also a need to develop strategies to classify CDSs [50]. SynthBiOnt can be used to identify CDS parts using several different types of queries. In this chapter, such a classification was demonstrated using knowledge about the products encoded by CDSs. For example, kinase and response regulator CDSs were classified using the GO terms assigned to the corresponding proteins.

It is known that existing reasoners have performance problems with large ontologies [146]. SynthBiOnt includes thousands of axioms. The performance of reasoners decreases as the complexity of the ontology increases, due to the number of classes and relationships involved in queries. Therefore, choosing a subset of the ontology that is relevant to a reasoning task will reduce the time required for classification. In addition, the performance of reasoners may differ depending on the number of axioms and queries involved.

SynthBiOnt has also been stored in an RDF endpoint, allowing SPARQL queries to be run effectively. However, a precise RDF graph structure must be used. On the other hand, writing OWL-friendly queries using Terp [230] or the Manchester syntax [223] is simpler. Once the Pellet reasoner is executed on the ontology, responses to subsequent Terp queries are given quickly since the reasoner would only be executed once. Although nested Terp queries that include cardinalities could not be executed, most of the queries deriving class definitions were executed successfully. However, the Pellet APIs are currently required in order to computationally execute Terp queries. Therefore, a Web service that takes Terp queries and runs on the reasoner that is already applied to the ontology would be desirable for querying the ontology.

Some of the properties in the ontology were recorded as human-readable annotations. This process resulted in thousands of annotations in order to reduce the number of axioms available to reasoners and to increase their performance. However, these properties can still be queried using SPARQL. In addition, approaches such as the

Ontology Pre-Processing Language [229] or rule-based queries, as demonstrated in this chapter, can be used to extract new axioms from annotations to make the data available to reasoners.

Extensive knowledge about *B. subtilis* is computationally available with the SynthBiOnt ontology presented in this chapter. Information about biological parts and molecular interactions can therefore be used for the automated design of complex and large-scale biological systems. The process of choosing a biological part in order to construct functional genetic circuits should also be automated in order to fully automate the design of these systems. SynthBiOnt can facilitate this automation for large-scale synthetic biology by providing access to extensive information about parts in standard formats. This ontology is encoded in OWL and accessible using the RDF syntax. Hence, existing off-the-shelf ontology tools, such as reasoners, and querying languages, including SPARQL and OWL-friendly Terp, can facilitate the extraction of large-scale information for synthetic biology. Therefore, this ontology can provide a link between the existing design tools and available domain knowledge for large-scale synthetic biology.

4.5.2 Conclusion

This chapter has described an OWL-based ontology, SynthBiOnt, which contains 42,259 classes, 269,729 subclass axioms, 21 data, 26 annotation, and 41 object properties. The ontology has been used to automatically classify classes using off-the-shelf reasoning tools. For example, 465 SigmaA promoters were identified, and 38 CDSs were classified as kinase encoding. The ontology is SBOL compliant, allowing querying using the SBOL model. The SBOL mapping of the 7,754 ontology classes was implemented using rule-based SPARQL queries. In addition to basic information about these sequence features that can be used as biological parts, SynthBiOnt provides information about the functional annotations and molecular interactions of gene products using the standard OWL language. This extensive information and its availability in standard formats, as demonstrated here, is essential to effectively automate large-scale synthetic biology. Although the ontology provides machine-accessible data, the ontological form of the data is not directly usable by CAD tools which use computational models. Therefore, parts and their interactions from the ontology should be represented using standard modelling languages such as SBML and CellML. In addition, the information from the ontology can be used to computationally annotate models for human and machine access.

Chapter 5. Composable Modular Models for Synthetic Biology

Modelling and computational simulation are crucial for the large-scale engineering of biological circuits since they allow the system under design to be simulated prior to implementation *in vivo*. To support automated, model-driven design it is desirable that *in silico* models are modular, composable and in standard formats. The synthetic biology design process typically involves the composition of genetic circuits from individual parts. At the most basic level these parts are representations of genetic level features such as promoters, ribosome binding sites (RBSs), and coding sequences (CDSs). However, it is also desirable to model the biological molecules and behaviour that arise when these parts are combined *in vivo*. Modular models allow these models of the parts to be composed and their associated systems to be simulated, facilitating the process of model centred design. The availability of databases of modular models then becomes desirable to support software tools in the model-driven design process. This chapter presents an approach for composable and modular models for synthetic biology, termed virtual parts, and describes the development of a publicly available database of these models for computational design tools.

5.1 Introduction

One ambition of synthetic biology is the large-scale engineering of biological systems [16-18]. This goal moves beyond conventional genetic engineering by aiming to design and build entire pathways and genomes [27]. Therefore, it is necessary to bring biological engineering up to a genomic scale and build on developments in high-throughput biology. However, as the complexity and size of designs increases, the manual design of genetic circuits becomes more challenging and the need for the inclusion of automated strategies and the development of new computational tools becomes important [36]. Computational designs are highly desirable for designing complex systems to improve the success of predicting the actual behaviour of the *in silico* system once implemented *in vivo* [20].

The mapping between the desired properties of a large complex system and the genetic parts necessary to encode the system *in vivo* are difficult, if not impossible for a human to do manually. In particular, dynamic models of the system to be built will help

optimise the design of the system by predicting the behaviour of the system before implementation *in vivo*. These models seek to capture *in silico* representations of the biological entities involved in the system behaviour, the levels of those entities and the relationships between the entities. As such, models can potentially act as a blueprint for the design of the system to be implemented.

Model-driven design has been used extensively in a number of different areas such as the software, automotive and aerospace sectors [304]. Models can be constructed to represent the relationships between elements in a system being modelled and consequently can be used to derive the process by which the system is generated [305]. In this approach, models are transformed into platform-specific artifacts. For example, in embedded systems, a model that represents a system defined with a set of requirements is used to automatically generate the hardware implementation of the system [306]. Similar to genetic circuits, these systems are also built with individual components such as transistors and logic gates, and the output of a component should be compatible with the input of the next component in order to allow the flow of data among the components. These components can be represented with transfer functions that produce outputs for given inputs using a set of equations. The integration of a set of equations from these transfer functions allows models to be produced that can be simulated in order to choose among and verify designs. The manual creation of simple electronic circuits was long ago replaced with the use of computer-aided design tools and design automation. As a result, electronic design automation has become an industry in which very-large scale integrated circuits can be constructed from well-defined components and subsystems that are electrically and physically correct [307].

Computational design tools that allow genetic circuits to be designed and simulated also have been developed [11, 39-47]. These tools often use libraries of mathematical models of biological parts in order to aid the user in building complex and predictable designs [33, 43]. However, there is a lack of modular and reusable models of biological parts in standard formats [80]. The availability of databases of such models would provide a useful computational resource to support tools for the design of synthetic genetic circuits. These databases could typically store information about the connectivity of parts, such as how they function together and regulate each other, in order to constrain the possible solution space for computational designs. By combining such information with quantitative parameters, the dynamics of systems can be simulated [53, 77].

Modular modelling approaches for synthetic biology have been demonstrated in

previous studies [11, 30, 41]. However, there is not always a one-to-one mapping between these models and the biological parts used to build actual systems *in vivo* [30]. Although these approaches define a formalism to represent models of parts, the number of models available is small. Whilst libraries of models are available for systems biology [46] there are very few available for synthetic biology applications. There is a need for computationally accessible and searchable libraries of models of parts for genetic circuit design.

Furthermore, in these modular modelling approaches, models represent mathematical representations of biological reactions; however, they do not contain information about the biological context they represent. Moreover, the composition of models is often application specific. Models can be annotated with machine-accessible metadata in order to add biological context and enable computational model composition using other tools.

Modular models can also be based on standard formats with levels of abstractions suitable to represent parts and their interactions. The Systems Biology Modelling Language (SBML) and CellML are two modelling languages widely-recognised in the synthetic biology community [32]. Both languages have XML-based formats that allow the electronic exchange of models between computer applications and support the writing of ordinary differential equations (ODEs) for biological reactions. These languages represent physical entities and their relationships coupled with kinetic parameters, allowing the simulation of the encoded systems in models.

This chapter describes the development of a modelling system for biological systems design, called standard virtual parts (SVPs). SVPs are reusable, modular models of physical biological parts. The use of annotations that aid in the computational composition of SVPs in order to construct large and complex models is also described. A publicly and programmatically accessible repository called BacilloBricks was developed to provide access to this library of SVPs. The repository also stores models of interactions which can be used to combine the models of individual parts in order to create simulatable models.

SVPs were initially developed using the CellML modelling language in collaborative work between Dr. Mike Cooling, Dr. Jennifer Hallinan, and Prof. Anil Wipat, in which I was also involved. In the work presented here, I have extended the SVP concept to use SBML and also defined additional types of models of biological parts. For example, I defined SVPs for operators, spacer sequences and promoters that can act as two-input logic gates. SVPs were initially described to represent biological

parts or any biochemical reactions associated with those parts. In the work presented here, one SVP is used for each biological part, encapsulating biochemical reactions that are specific to the biological parts being modelled in order to facilitate the composition of SVPs. Moreover, I developed a repository of SVPs that can be used by CAD tools, and annotated these SVPs in order to facilitate the computational composition of SVPs.

5.2 SVPs: A method for composable modular models

5.2.1 Standard virtual parts

SVPs are abstract models that are designed to represent basic, physical biological parts such as promoters, RBSs, and CDSs *in silico* [80]. Intracellular events such as biochemical processes and gene products that influence the behaviour of a genetic circuit are also represented using SVPs. SVPs have standard inputs and outputs that allow the construction of models of larger systems to be built from basic biological parts. These interfaces are based on widely-accepted biological signals such as polymerases per second (PoPS) and ribosomes per second (RiPS) [116] as described in previous modular modelling approaches in synthetic biology [11, 41]. Sharing these common signals makes the models composable [32]. SVPs were initially defined for promoters, RBSs and CDSs [80].

A promoter SVP is formulated with the equation $k * c_1(V)$ [80]. In this formula, k is the rate of transcription, measured in PoPS. Details about the background transcription machinery of the cell are abstracted. The second part of the formula, which converts the PoPS signal into concentration rate in nanomolar per second, is [80]:

$$c_1(V) = \frac{10^9}{V * A * 10^{-15}}$$

where *A* is the Avogadro's number and *V* is the volume of the cellular compartment in femtolitres, where transcription takes place. For a whole *B. subtilis* cell, which is one femtolitre, this conversion factor becomes 1.66.

Not all promoters have constant transcription rates. This rate can be changed by the binding of an activator or repressor. Hill equations can be used to model the probability of whether or not the promoter is free, for maximal transcription [241]. Therefore, the rate of transcription for an inducible promoter formulated using Hill equations is (see Section 2.3.2):

$$k * \frac{Activator^n}{Activator^n + Km^n}$$

where *Activator* is the concentration of an inducer, *Km* is the disassociation constant, and *n* is the Hill coefficient. Similarly, the rate of transcription for a repressible promoter is:

$$k * \frac{Km^n}{Repressor^n + Km^n}$$

where *Repressor* is the concentration of an inhibitor.

SVPs also exist for mRNAs [80]. These SVPs do not include any parameterisation; however, they are used to transform PoPS signals into mRNA signals by using an ODE expression, in which the mRNAs are derivatives of PoPS.

RBSs convert mRNAs into RiPS signals. The total RiPS is the multiplication of translation rate, measured in RiPS, from an RBS by the concentration of total mRNAs [80]:

$$R = k_{tl} * mRNA * c_2(V)$$

where k_{tl} is the translation rate and $c_2(V)$ is a conversion factor, *V* is the volume of the cellular compartment where translation takes place, and *R* is given as attomoles per second.

The CDS formulation uses the RiPS from an RBS to create a flux of protein production in nanomolar per second [80]:

$$J = \frac{R}{c_2(V)}$$

5.2.1.1 Biological reactions

Biological reactions can also be represented within modular models. These reactions are often represented using mass-action kinetics. For example, a reaction that includes the phosphorylation of a protein by another molecule can be modelled as [80]:

$$k_f * Protein * Inducer$$

where k_f is the reaction rate, *Protein* is the concentration of the protein and *Inducer* is the concentration of the molecule that transfers its phosphate. This reaction flux, when simulated, is represented by a gain to the total fluxes for the protein molecule. The dephosphorylation reaction can be represented as:

$$k_b * Protein{\sim}P$$

where k_b is the rate of dephosphorylation and $Protein{\sim}P$ is the concentration of the phosphorylated form of the protein, and it is assumed that dephosphorylation is not affected by other molecules. Similarly, the degradation of a protein or an RNA substrate can be represented as:

$$k_{deg} * Substrate$$

where k_{deg} is the rate of degradation. This reaction flux, when simulated, contributes a loss to the total fluxes for the substrate.

The mathematical formulations presented here can be instantiated to create SVPs by providing quantitative parameters. SVPs can be constructed for individual biological parts such as CDSs, promoters and RBSs, enabling the construction of models of larger systems built from these biological parts. SVPs could also be extended to other types of parts, such as operators, spacer sequences and complex promoters that are regulated by more than one transcription factor (TF).

5.2.1.2 Complex SVPs for two-input logic gate promoters

Whilst SVPs can be defined for basic biological parts, which are often the most widely used unit of composition, especially in the bottom up approach to synthetic biology, there is often a need to compose systems from units that are already combinations of basic biological parts. One example of this requirement is the need to simulate the design of bacterial logic gates prior to their implementation *in silico*. A number of studies in synthetic biology have used logic gates, such as AND and OR, in order to build genetic circuits [66, 99]. A set of SVPs was defined for logic gates in order to explore the utility of SVPs in representing more complex parts and to provide a tool kit for bacterial logic gate design.

Most computational approaches in synthetic biology use single-input promoters [43, 66, 123]. Bacterial promoters that are controlled by two TFs also display logic-gate behaviour [78, 189, 293, 297]. These promoters can be used effectively to create diverse

applications. NAND and NOR gates in particular are very important, since either of these gates can be combined to implement any computational operation [69]. With the rational design of combinatorial promoters [70, 71], it is important to have models of two-input promoters in order to predict the behaviour of constructed circuits.

A single bacterial promoter that is controlled by two TFs can behave as an AND, OR, ANDN, NAND or NOR gate. Table 5.1 shows a truth table for these gates. For an AND gate to be active, two inputs must exist together. On the other hand, a NAND gate is active when two inputs do not exist together. An OR gate is active when either of the inputs exist; conversely, a NOR gate is active when neither input is present. ANDN is only active when only the first input is present.

Table 5.1: Truth table for promoter logic gates.

Input1	**Input2**	**AND**	**OR**	**ANDN**	**NAND**	**NOR**
0	0	OFF	OFF	OFF	ON	ON
0	1	OFF	ON	OFF	ON	OFF
1	0	OFF	ON	ON	ON	OFF
1	1	ON	ON	OFF	OFF	OFF

The behaviour of logic gate promoters arises from the cooperativity or competition between TFs that bind to promoters [90]. For the modelling of these promoters, it is assumed that TFs bind to DNA with rapid equilibrium. This assumption allows a promoter's occupancy to be represented by the concentration of TFs, which thus simplifies the modelling [238]. Hence, using the binding strengths given by the disassociation constants, and the cooperativity between the TFs, the binding probability of RNAPs to a promoter can be computed [78]. It has been shown that, according to the nature of the spacing sequence between two operators, their binding affinities and the promoter's strength, the overall design can function, for example, as an AND or OR gate [189].

An AND gate promoter (Figure 5.1) can be defined as a weak promoter with two weak upstream activator sites that are next to each other [78, 189, 297]. The promoter is weak, and therefore the gate is not active when the TFs are not present. Both operators are also weak and the TFs cannot bind to operators; however, when both TFs are present, they can bind cooperatively to be able to activate a weak AND gate promoter. The transfer function for the promoter occupancy of such a promoter can be given as:

$$A1\ AND\ A2 = P(A1) * P(A2) = \frac{\frac{A1}{K1}}{1+\frac{A1}{K1}} * \frac{\frac{A2}{K2}}{1+\frac{A2}{K2}} = \frac{\frac{A1}{K1} * \frac{A2}{K2}}{1+\frac{A1}{K1}+\frac{A2}{K2}+\frac{A1}{K1} * \frac{A2}{K2}}$$

where *A1* and *A2* are the activating TFs, and *K1* and *K2* are the corresponding disassociation parameters.

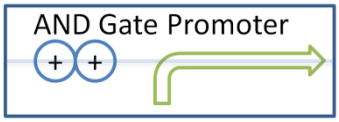


Figure 5.1: An AND gate promoter with two activator sites next to each other. Both the operators and the promoter are weak. The promoter is weak and therefore is not active when the TFs are not present. In addition, the promoter can be activated by two TFs, which can only bind due to the cooperative interaction between the TFs.

A NAND gate promoter is defined as a strong promoter with two weak repressor sites that are next to each other [78, 293, 297]. The binding sites overlap with the core promoter sequences. When the TFs are not present, the gate is active since the promoter is strong and does not require activation for transcription. Similar to an AND gate promoter, each TF cannot bind to the promoter without the help of the cooperative interaction between the TFs. Therefore, the TFs bind cooperatively to be able to repress the activity of a NAND gate promoter. The transfer function for the promoter occupancy can be given as:

$$\overline{R1\ AND\ R2} = 1 - \big(P(R1) *\ P(R2)\big) = \frac{1+\frac{R1}{K1}+\frac{R2}{K2}}{1+\frac{R1}{K1}+\frac{R2}{K2}+\frac{R1}{K1} * \frac{R2}{K2}}$$

where *R1* and *R2* are the inhibiting TFs, and *K1* and *K2* are the corresponding disassociation parameters.

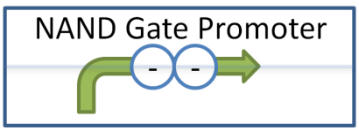


Figure 5.2: A NAND gate promoter with two inhibitor sites next to each other. Operators are weak; however, the promoter is strong.

An OR gate promoter (Figure 5.3) is defined as a weak promoter with two strong, upstream activator sites that are not next to each other [78, 189, 297]. The promoter is

weak and, hence not active when the TFs are not present. Either of the bindings of an activator is sufficient to activate the weak promoter. The transfer function for the promoter occupancy can be given as:

$$A1\ OR\ A2 = P(A1) + P(A2) - P(A1) * P(A2) = \frac{\frac{A1}{K1} + \frac{A2}{K2} + \frac{A1}{K1} * \frac{A2}{K2}}{1 + \frac{A1}{K1} + \frac{A2}{K2} + \frac{A1}{K1} * \frac{A2}{K2}}$$

where *A1* and *A2* are the activating TFs, and *K1* and *K2* are the corresponding disassociation parameters.

Figure 5.3: An OR gate promoter with two activator sites to which the TFs bind independently. Operators are strong; however, the promoter is weak.

A NOR gate promoter (Figure 5.4) can be defined as a strong promoter with two strong repressor sites that are not next to each other [78, 293, 297]. The binding sites overlap with the core promoter sequences. Similar to a NAND gate, when neither TF is present, the gate is active since the promoter is strong and does not require any activation for transcription. Either of the bindings of a repressor is sufficient to inhibit the strong promoter. The transfer function for the promoter occupancy can be given as:

$$\overline{R1\ OR\ R2} = 1 - \big(P(R1) + P(R2) - P(R1) * P(R2)\big) = \frac{1}{1 + \frac{R1}{K1} + \frac{R2}{K2} + \frac{R1}{K1} * \frac{R2}{K2}}$$

where *R1* and *R2* are the inhibiting TFs, and *K1* and *K2* are the corresponding disassociation parameters.

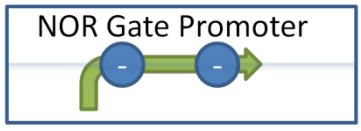


Figure 5.4: A NOR gate promoter with two repressor sites to which the TFs bind independently. Both the operators and the promoter are strong.

An ANDN gate promoter (Figure 5.5) can be defined as a weak promoter with a strong activator and strong repressor sites [293, 297]. The activator site is upstream and

the repressor site overlaps with the core promoter sequence. The promoter is active when the activator is bound and the repressor is not bound. The transfer function for the promoter occupancy can be given as:

$$A\ AND\ \overline{R} = P(A) * P(\overline{R}) = \frac{\frac{A}{K_A}}{1 + \frac{A}{K_A} + \frac{R}{K_R} + \frac{A}{K_A} * \frac{R}{K_R}}$$

where A and R are the activating and inhibiting TFs respectively, and K_A and K_R are the corresponding disassociation parameters.

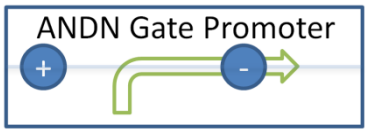


Figure 5.5: An ANDN gate promoter with one activator and one inhibitor site. Operators are strong; however, the promoter is weak.

5.2.1.3 Modelling of operators

As operators can be used as parts of promoters to create logic-gate like functions, they can also be used to regulate the transcription as individual parts by combining them with promoters. For example, Bagh and his co-workers constructed a promoter which can be induced by arabinose, IPTG and ATC [240]. In their design, the AraC operator was added to the front of the LacI repressible promoter, which was followed by the TetR operator.

Operators can be modelled with their operator occupancies for the corresponding activators or inhibitors. For an operator bound by a TF, the operator occupancy can be given as:

$$O(TF) = \frac{TF}{TF + K_D}$$

where K_D is the disassociation constant for the TF. Therefore, the output of an operator bound by an activator is:

$$Input * O(Activator) = Input * \frac{TF}{TF + K_D}$$

where *Input* is the preceeding PoPS signal or 1 when an operator is the only component upstream of a promoter. Similarly, the output of an operator bound by a repressor is:

$$Input * \big(1 - O(Repressor)\big) = Input * \frac{K_D}{TF + K_D}$$

PoPS is currently not modelled at the RBS and CDS level. It is assumed that PoPS from a promoter, or an operator which can modulate the output of a promoter is directly used to produce mRNAs. However, SVPs can be extended to incorporate factors that affect the elongation of transcription, such as the length of a CDS or binding sequences as parts of CDSs [11], and hence affecting the final amounts of PoPS.

5.2.1.4 Modelling of spacer sequences

Spacer sequences, termed shims in this work, can also be used as modular parts for the fine tuning of input biological signals such as PoPS or mRNA. For example, a shim sequence that is used to alter the spacing between a promoter and an RBS sequence may have an effect at the translational level. These shims may change the folding of an mRNA to increase or decrease the probability that the RBS sequence is open for ribosomes to bind to [113], hence influencing the number of mRNAs that can be used to produce the RiPS signal. Therefore, the output of a shim can be given as:

$$k * Input$$

where *k* is the conversion parameter that increases or reduces the amount of a signal, and *Input* is the modulated biological signal.

The modular nature of SVPs allows new templates to be defined with the desired levels of abstraction. In this section, the list of SVPs has been extended to include operators and shims. Promoters representing two-input logic gates have also been defined. The abstraction used for the modelling of these promoters takes their relationships with binding TFs into account. Different types of logic gates were modelled based on physical constraints such as the closeness and strength of operators and promoters. In the next section, SVPs are extended to encapsulate biological reactions in order to create a one-to-one mapping between SVPs and corresponding sequence features.

5.3 Mapping SVPs to genetic elements

SVPs, on their own, cannot always be simulated. They must be joined appropriately to construct larger models that can be simulated in order to investigate the behaviour of biological systems under design. For a computational composition of these models, the inputs and outputs of SVPs must be clearly specified. It would be ideal if each sequence feature had a corresponding dynamic model, and these models could be easily joined together based on defined inputs and outputs. For example, for a simple genetic circuit that includes a promoter, an RBS and a CDS, models of these biological parts should be easily composable. However, with the current definition of SVPs [80], this process is difficult.

Although SVPs represent models of physical biological parts, which are sequence features such as promoters, RBSs and CDSs, there are other SVPs that cannot be mapped to sequence features. Such modelling entities are used to combine SVPs which represent sequence features, in order to construct simulatable models. For example, for most proteins or RNAs, modelling entities which represent degradation must be added. Proteins are also represented by SVPs. Such elements are added each time the SVPs representing biological parts are joined. However, for a computational composition of these models, a computer application has to recognise all of these individual modelling entities and different types of inputs and outputs, such as the negative flux from a protein degradation process which requires a protein input. Therefore, wherever possible, these recurring elements can be encapsulated as part of the SVPs representing the biological parts. This approach would enable the one-to-one mapping of biological parts to their dynamic models, facilitating the composition of SVPs. In addition, by using a single model to represent a part, models of biological parts can be distinguished from models of their interactions. This separation would enable the mapping of physical interactions between molecules to models of interactions between SVPs. Hence, the process of constructing larger models can be simplified to joining SVPs together by adding models of interactions.

In order to map a physical part to its behaviour in a single model, SVPs are constructed as coarse-grained models with inputs and outputs. Modelling entities representing reactions that are specific to an SVP are encapsulated in the model in order to simplify the connection of SVPs. Recurring modelling entities such as protein translation and degradation elements are glued together inside the SVPs and thus do not need to be added when SVPs are joined together.

CDSs and the behaviour of their encoded products were modelled using FunctionalPart SVPs. An SVP of this type has a RiPS input that can be connected to models of RBSs. The output is given with the default and, if exists, modified forms of gene products. For example, for a protein that can be phosphorylated, both its unmodified and phosphorylated forms are the output for the SVP. Additional inputs can also be defined. For a protein that is induced by an environmental factor, the concentration of the inducer molecule can be used as the input, assuming that the interaction between the inducer and the protein can be abstracted using a Hill-like equation, and hence it can be encapsulated inside the SVPs.

These SVPs encapsulate the modelling entities for gene products and their production. The interaction of gene products with environmental factors and biochemical reactions such as degradation and deactivation, which are abstracted so as not to be dependent on other parts, are modelled as internal interactions. The example in Figure 5.6 represents an SVP for a CDS and its encoded protein product. The RiPS signal is converted to a protein flux through a translation reaction. The concentration of the protein is represented by a modelling entity, such as a *species* in SBML. The protein is induced by a small molecule which is an input to the SVP. The reaction, which models the activation of the protein, links its unmodified and activated forms. The flux from this reaction is a gain for the activated form of the protein while it is a loss for the unmodified protein. Both forms of the protein are also linked through the deactivation interaction. In the example, the degradation of both forms of the protein is also encapsulated as part of the SVP. Interactions of the protein with other molecules can therefore be linked directly to this SVP.

Such interactions are not part of any SVPs, which only include modelling entities that are specific to a sequence feature, and if exist its gene product such as a protein. These interactions are also represented by models, which are hereafter referred to as models of interactions, and are used to join SVPs in order to create simulatable models.

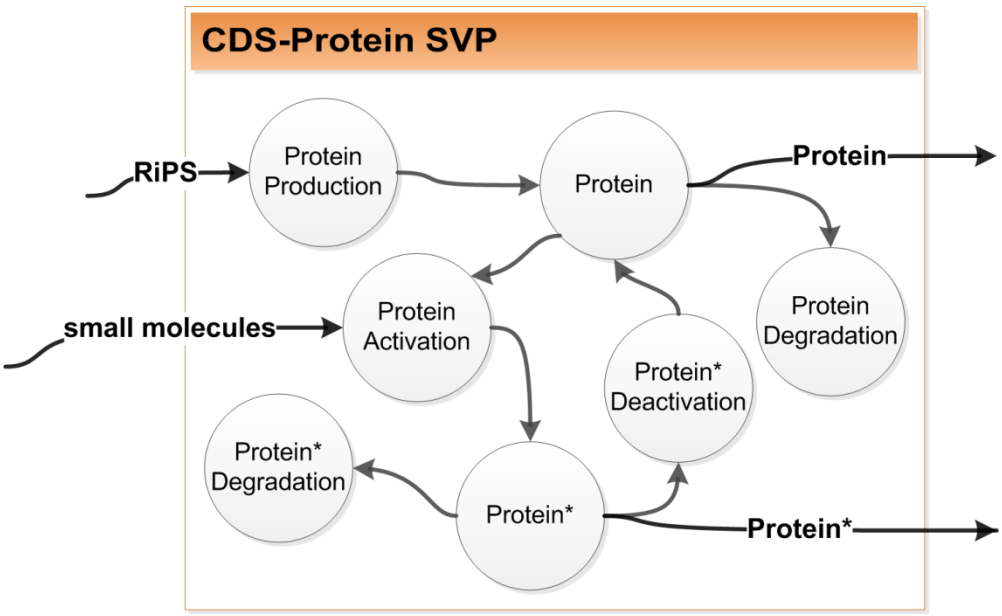


Figure 5.6: An example of an SVP for a CDS and its encoded protein. Protein* represents the activated form of the protein in the SVP. The SVP includes the modelling entities for protein production, and biochemical reactions such as activation, degradation, and deactivation.

Promoter SVPs can be constructed to encapsulate mRNA production. However, the PoPS output of a promoter can be regulated by using other parts [11]. For example, regulatory regions may amplify or reduce PoPS signals. Therefore, a promoter SVP is simply a PoPS generator (Figure 5.7), and mRNA SVPs must be connected to the PoPS output of a relevant SVP. This approach decouples the use of promoters and other parts such as operators that affect the final magnitude of PoPS signals. Moreover, relationships between promoters and TFs are represented as models of interactions. These models have PoPS inputs and outputs which are calculated using the PoPS inputs and the promoter occupancies of TFs. This approach also provides more flexibility in the design of genetic circuits, since the variances of a TF can be used to control the activity of the same target promoter. Hence, these interaction models can be used based on TFs. A similar approach was also used for operator SVPs. Hence, models of interactions for operator and TF SVPs represent the operator occupancies of TFs.

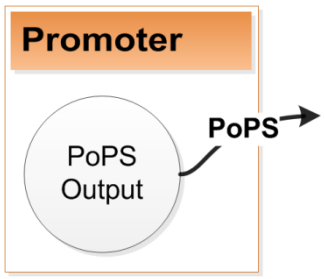


Figure 5.7: Promoter SVPs are PoPS generators.

An mRNA SVP includes modelling entities for mRNA production, degradation and the mRNA itself (Figure 5.8). Although this SVP does not have an equivalent physical part at the DNA level, it is required as a glue to connect the modelling of transcription and translation by transforming PoPS into the RiPS signal.

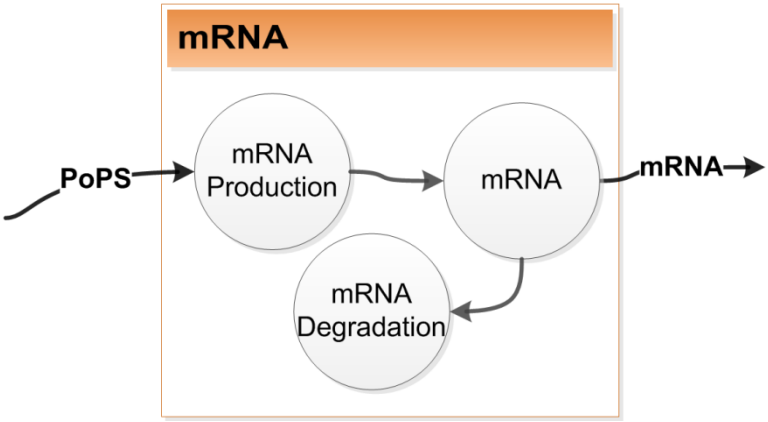


Figure 5.8: mRNA SVP that contains modelling entities for mRNA production, degradation and mRNA.

The definition of inputs and outputs, as being demonstrated in this section, enables the composition of models that encapsulate detailed biochemical reactions.

5.4 Manually composing virtual systems from SVPs

SVPs are specified with inputs and outputs which refer to widely-accepted biological signals in synthetic biology, such as PoPS and RiPS. Therefore, as biological parts that send and receive these signals can be combined based on the exchanges of these signals, SVPs are composable using these inputs and outputs. Promoter SVPs have PoPS outputs which are converted into mRNA (Figure 5.9). RBS SVPs have mRNA as the input and produce RiPS as output which is then used as input for proteins. Each molecular form of a protein is represented as output. These defined signals specify how SVPs can be composed together without handling fine-grained internal modelling entities and their relationships.

Figure 5.9: Interfacing SVPs with common signal carriers.

Operator and promoter SVPs can be joined to amplify or reduce PoPS signals. It is

assumed that these parts do not overlap, and that their respective activators or repressors act independently [242]. This assumption allows the use of operator parts freely regardless of upstream and downstream sequences [241] (Figure 5.10). In cases where operators overlap or TFs bind cooperatively, operators should be modelled as part of promoters in order to reflect the combined transfer functions.

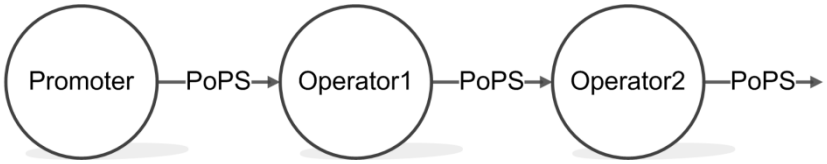


Figure 5.10: An example of an interfacing between a promoter and two operators. Operators independently modulate the PoPS signal.

SVPs are formulated for the computational simulation of biological systems composed from individual physical parts. In order to construct simulatable models; these SVPs must be available for use with computational tools. Therefore, instances of SVPs must be serialised into formats that can be read by computers. SBML is a widely-adopted machine-readable modelling language [53, 308]. SBML has also been used by tools developed for genetic circuit design [11, 30, 33, 41, 47]. In addition, tools such as COPASI [244] can simulate SBML models both deterministically and stochastically. Therefore, SVPs were also created in the form of SBML models, building on the previous work with CellML [80].

5.4.1 Modular modelling in SBML

Modular SBML models must be joined together using their inputs and outputs, such as proteins, mRNAs, PoPS and RiPS, in order to create simulatable models These models consists of a number of entities hereafter referred to as ‘modelling entities’. In SBML, mathematical formulations of SVPs can be represented using *reaction* or *assignment rule* entities. Assignment rules are used to describe model parameters using mathematical equations. On the other hand, reactions represent biochemical reactions that can change the quantity of cellular molecules which are represented with species entities, and can also refer to assignment rules. Inputs and outputs in the form of species entities that represent the same cellular molecules such as proteins in different SVPs must be connected in order to add their reaction fluxes so that the final concentrations can be calculated. For example, for a TF that needs to be phosphorylated by a kinase, two different models can exist: while the first one includes the production of the TF and has the TF as output, the second model can include a model fragment representing the

phosphorylation of the TF, and hence has the TF as input. Therefore, these two models must be connected in order to predict amounts of phosphorylated TFs. In SBML, species are defined globally and can be used directly in modelling entities such as reactions or rules. Moreover, reaction fluxes for each species are collected implicitly. Therefore, when referred globally across SVPs, species can be used without any further need to connect these entities explicitly. Similarly, reactions that represent the interactions of species can be included directly without any further amendments to the models.

PoPS and RiPS inputs and outputs represent rates of transcription and translation respectively and have mathematical formulations. In SVPs, physical polymerases and ribosome molecules are not modelled, since it is assumed that the amounts of these molecules are not limiting factors in transcription and translation respectively [80, 238]. Instead, these rates are used to calculate the amounts of mRNAs and proteins which are species in SVPs. Therefore, inputs and outputs whose definitions are based on PoPS and RiPS were modelled using assignment rules.

These assignment rules must be joined appropriately. For example, FunctionalPart SVPs are formed of species, reactions, and assignment rules. Assignment rules specify the RiPS signals that must be transformed into molecules such as proteins or RNAs. Species such as external inducer molecules and corresponding proteins with their different molecular forms can be inputs or outputs for these SVPs. Reactions then represent the biochemical reactions of these species. The connection of species and reactions are implicitly handled as described; however, the RiPS inputs of these SVPs must be explicitly connected.

5.4.2 Composition of models in SBML

To demonstrate SVPs and their interfacing, the construction of a subtilin receiver device was used as a use case. This device was submitted by the Newcastle University's 2008 International Genetically Engineered Machine competition team to the Parts Registry[36], and in this system, the lantibiotic (a class of small peptide antibiotic) subtilin is sensed by a two-component system (TCS) [309, 310]. This TCS comprises the kinase protein SpaK and the regulatory protein SpaR. CDSs encoding these proteins are located in the same operon and transcribed by a constitutive promoter. Each CDS has its own RBS.

[36] http://2008.igem.org/Team:Newcastle_University

5.4.2.1 Conceptual modelling of the subtilin receiver device at a systems level using SBML

In order to demonstrate the utility of SVPs the modelling of the system using a standard SBML approach is presented first. In the next section we then show how the system can be composed from SVPs.

The *spaRK* operon comprises the phosphate-mediated signalling system required to form an activated SpaR protein (Figure 5.11). Upon sensing subtilin, SpaK activates SpaR via phosphorylation. The activated SpaR then activates the downstream *pSpaS* promoter to express the green fluorescent protein (GFP) as a reporter protein. Both SpaK and SpaR are autodephosphorylated at a constant rate. The reactions that abstract the activation and deactivation of SpaK and SpaR proteins are given below:

SpaK activation: Subtilin + SpaK → SpaK_Active + Subtilin

SpaK deactivation: SpaK_Active → SpaK

SpaR activation: SpaK_Active + SpaR → SpaK + SpaR_Active

SpaR deactivation: SpaR_Active → SpaR

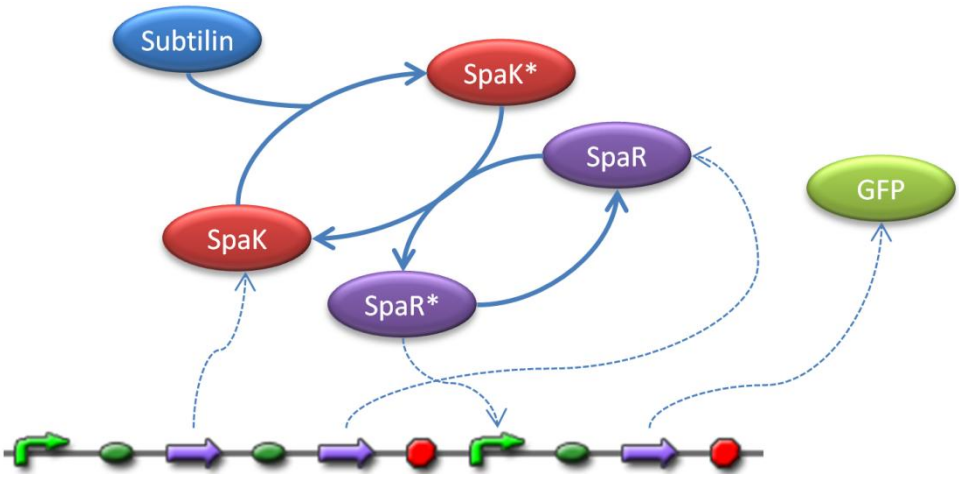


Figure 5.11: The subtilin receiver device. Adapted from [80].

The computational model of the subtilin receiver device includes modelling entities for the activation and deactivation of SpaK and SpaR proteins, the transcription of mRNAs from the two promoters, and the translation of proteins. Modelling entities representing the degradation of all of the proteins and mRNAs in the device are also added to the model. In the constructed model, the PoPS signal from the *pSpaRK* promoter is converted to mRNAs. It is assumed that the mRNA transcription is uniform for both CDSs. Therefore, both RBSs use the same mRNA output from the *pSpaRK* promoter to produce RiPS signals which are then converted to the SpaK and SpaR

proteins. The activation of SpaR is modelled via an interaction between the SpaK and SpaR SVPs. Similarly, the activation of the *pSpaS* promoter by the activated SpaR is modelled via an interaction between the corresponding SVPs. This interaction has a PoPS output which is converted to mRNA, which is the input to the third RBS in order to produce GFP.

5.4.2.2 Subtilin receiver with SVPs

The process of constructing the subtilin receiver from SVPs is now considered. Although the composition rules of SVPs are explained here using a manual approach, SVPs are designed to be constructed computationally. An application programming interface (API) that provides methods for each of these steps is also available and is presented in Section 5.7.4.

1. **Create an empty model and add the models of SVPs.**

In order to construct a larger model, SVPs are initially placed in an empty SBML container model. For the subtilin receiver device, these SVPs are the 'pspaRK' and 'pspaS' promoter SVPs; three SVPs representing the RBSs for *spaK*, *spaR* and *gfp* CDSs; and three FunctionalPart SVPs for SpaR, SpaK and GFP proteins and their encoding CDSs (Figure 5.12).

2. **Add mRNA SVPs.**

For each RBS SVP, an mRNA SVP is added. For operons that include more than one RBS, the same mRNA SVP can be used for all RBS SVPs representing the RBS parts in the operon. Therefore, for the subtilin receiver device one mRNA SVP is used for SVPs representing the RBSs for *spaK* and *spaR* CDSs, and another one is used for the RBS preceding the *gfp* CDS.

3. **Join SVPs by connecting their inputs and outputs**

The the same types of inputs and outputs of SVPs are connected according to the rules below:

- Connect the PoPS output of a promoter SVP to the PoPS input of an mRNA SVP.
- Connect the mRNA output of an mRNA SVP to the mRNA input of an RBS SVP.
- Connect the RiPS output of an RBS to the RiPS input of a FunctionalPart SVP.

These connections can be implemented by using assignment rules in SBML. In this process, an assignment rule is specified for a PoPS, RiPS or mRNA input of an SVP. For example, the '`SpaK_RiPSInput = 1 * SpaK_RBS_RiPSOutput`' assignment

rule can be constructed, in which the ‘SpaK_RiPSInput’ RiPS input of the ‘SpaK’ SVP is defined to be the ‘SpaK_RBS_RiPSOutput’ RiPS output from the ‘SpaK_RBS’ RBS SVP. Using the same approach, PoPS and mRNA inputs and outputs between different SVPs can also be connected.

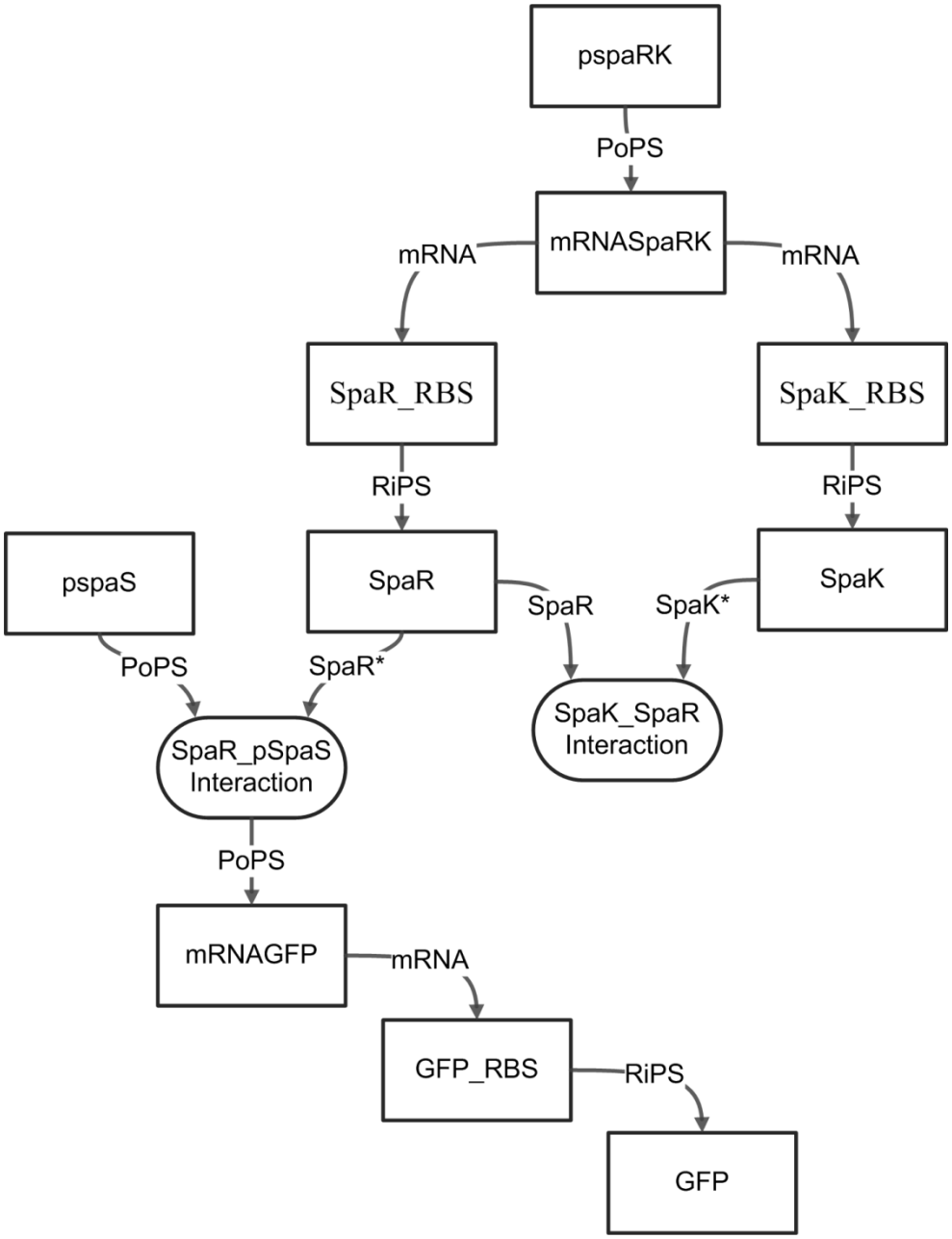


Figure 5.12: Relationships between the SVPs of the subtilin receiver model. Curved rectangles represent models of interactions and other boxes represent SVPs. Each SVP has internal modelling entities that are encapsulated. SVPs are connected with signal carriers such as PoPS and RiPS.

4. Join SVPs by adding models of interactions.

Models of interactions between parts must also be added. If the interacting parts are proteins, the interaction model can be directly added to the model since the connections

for reactions in SBML are implicit. Therefore, in the subtilin receiver model, the interaction between the SpaK and SpaR proteins can be added directly to the container model without any explicit connection. However, for a TF-promoter or a TF-operator interaction, the PoPS output of the DNA SVP must be connected to the PoPS input of the interaction model. Such a model also has a PoPS output which can then be joined to an mRNA SVP. These connections can also be defined using assignment rules as demonstrated above. For the interaction between the 'pspaS' promoter and 'SpaR' SVPs, the PoPS output of the 'pspaS' SVP must be connected to the PoPS input of the model of the interaction. Moreover, the PoPS output from this interaction model should be connected to the corresponding mRNA SVP.

This section has described the composition of SVPs implemented using SBML. These models include modelling entities that can be mapped to commonly-accepted biological signal carriers. The construction of larger models requires the addition of modelling entities in order to join the SVPs together by linking these inputs and outputs. The composition of SVPs was then demonstrated for the construction of the subtilin receiver device. Currently this process is performed manually. Therefore, the models should be annotated with metadata in order to enable the computational composition of SVPs.

5.5 Enabling automated model composition via input/output annotations

SVPs can be manually joined together to create models of systems. However, this process is time consuming and error-prone when carried out manually. Ideally, SVPs need to be joined together computationally to enable large-scale, model-driven, design. SVPs are designed to be used by CAD tools. However, these models only contain information about the mathematical representation of biological concepts and their relationships necessary for computational simulation. Information about how to join the parts is not included.

The ordering of the biological parts in a genetic circuit is crucial. It is obvious to an expert that a CDS should be placed after an RBS, but to enable computational design the flow of signals between biological parts must be defined in a computationally accessible format.

In order to standardise the interfacing of these parts, signal carriers have been defined. SVPs also have inputs and outputs that represent biological signal carriers. Therefore, these interfaces can be annotated with machine-readable metadata in order to

enable the computational composition of SVPs. Each SVP may contain more than one modelling entity. Therefore, entities that can be mapped to signal carriers must be distinguished from other entities.

Metadata about the inputs and outputs of SVPs should be available in their XML forms in order to aid the design of complex circuits using computational design tools. RDF documents can be serialised in an XML format, and RDF has also been accepted as the standard annotation language for both SBML and CellML models [53]. Therefore, annotations in the form of RDF/XML documents were added to the SVPs.

Each SVP is annotated with a list of inputs and outputs that can be mapped to relevant modelling entities. These annotations are given with resource-property-value RDF triples, in each of which the resource is the SVP being annotated, the property is whether the interface is an input or output, and the value is the modelling entity being referred to by the annotation. Properties belong to a special namespace[37] and are prefixed with `mts`. The URI for the namespace does not exist physically; however, it is used to construct unique URIs. Input and output properties are specified with `mts:Input` and `mts:Output` URIs.

The modelling entities referred to are annotated with additional information. These annotations are also in the RDF/XML form and are stored in the XML structure of the corresponding modelling entities. This additional information is given with RDF properties. The `mts:InterfaceType` property can either have ‘Input’ or ‘Output’ as its value, and is used to indicate whether the entity acts as input or output. The `mts:SignalType` property specifies the type of signal carrier that is represented by the modelling entity. The values for this property include ‘PoPS’, ‘mRNA’, ‘RiPS’, ‘Species’, ‘EnvironmentConstant’ or 'Volume'. This property enables the composition of models based on biological signal carriers. `mts:MolecularForm` is an additional property that specifies the molecular forms of the species. Values for this property are ‘Default’, ‘Phosphorylated’, ‘Dimer’, and ‘Tetramer’, and this list can be extended. Species that represent physical molecules can be linked to these molecules with the `mts:Species` property.

5.5.1 Annotation of promoter SVPs

Promoters are PoPS generators. The PoPS output of a promoter SVP is represented by a modelling entity. Therefore, the SVP annotation includes a statement about this entity which contains further information about the promoter. This entity is referred to with

[37] http://purl.org/modeltosequence/1.0#

the `mts:Output` property (Figure 5.13).

```
<rdf:Description rdf:about="#PspaRK">
    <mts:Output>PspaRK_PoPS_Output</mts:Output>
</rdf:Description>
```

Figure 5.13: An example of a promoter SVP annotation. The promoter SVP lists one output that can be linked to the relevant modelling entity.

The entity that corresponds to the statement is further annotated with additional information. The `SignalType` and `InterfaceType` properties are given as 'PoPS' and 'Output' respectively. Therefore, this entity can be used as a PoPS source to connect to an mRNA SVP.

```
<rdf:Description rdf:about="#PspaRK_PoPS_Output">
  <mts:InterfaceType>Output</mts:InterfaceType>
  <mts:SignalType>PoPS</mts:SignalType>
</rdf:Description>
```

Figure 5.14: Annotation of a promoter modelling entity. `SignalType` and `InterfaceType` properties are given as 'PoPS' and 'Output' respectively.

5.5.2 Annotation of SVPs representing CDSs and encoded proteins

The annotation of an SVP representing a CDS that encodes for a protein may include additional entries. Such an SVP has a RiPS input and a species output. Molecules such as external inducers can also be inputs. Therefore, the SVP annotation includes a list of inputs and outputs. Figure 5.15 shows the annotation for the 'SpaK' SVP. The annotation includes two entries for the modelling entities that represent the RiPS and subtilin input of the SVP. Protein species in default and phosphorylated forms are annotated as output.

```
<rdf:Description rdf:about="#SpaK">
   <mts:Input>SpaKProductionRiPSInput</mts:Input>
   <mts:Input>Subtilin</mts:Input>
   <mts:Output>SpaK</mts:Output>
   <mts:Output>SpaKPhosphorylated</mts:Output>
</rdf:Description>
```

Figure 5.15: Example of a protein SVP annotation. Two entries are listed as input and output respectively.

Each of the modelling entities listed in the annotation is also annotated with `SignalType` and `InterfaceType` properties. Modelling entities that represent the RiPS input are given with 'RiPS' signal type and 'Input' interface type statements.

Species that represent the encoded proteins are given ‘Species’ and ‘Output’ values for these properties respectively. In order to distinguish between proteins with different molecular forms, each species entity includes a statement given with the `mts:MolecularForm` property. Furthermore, these species are linked to the proteins they represent with the `mts:Species` property. Environmental constants such as inducers are given with ‘Input’ and ‘EnvironmentConstant’ values for the interface type and signal type properties respectively. Similarly, the molecules they represent are annotated with the `mts:Species` property. Figure 5.14 shows the annotation of the four modelling entities given in Figure 5.15. Entities for RiPS and subtilin, and the default and phosphorylated forms of the SpaK protein are annotated.

```
Subtilin:
<rdf:Description rdf:about="#Subtilin">
   <mts:InterfaceType>Input</mts:InterfaceType>
   <mts:SignalType>EnvironmentConstant</mts:SignalType>
   <mts:Species>Subtilin</mts:Species>
   <mts:MolecularForm>Default</mts:MolecularForm>
</rdf:Description>

SpaK:
<rdf:Description rdf:about="#SpaK">
   <mts:InterfaceType>Output</mts:InterfaceType>
   <mts:SignalType>Species</mts:SignalType>
   <mts:Species>SpaK</mts:Species>
   <mts:MolecularForm>Default</mts:MolecularForm>
</rdf:Description>

SpaKPhosphorylated
<rdf:Description rdf:about="#SpaKPhosphorylated">
   <mts:InterfaceType>Output</mts:InterfaceType>
   <mts:SignalType>Species</mts:SignalType>
   <mts:Species>SpaK</mts:Species>
   <mts:MolecularForm>Phosphorylated</mts:MolecularForm>
</rdf:Description>

SpaKProductionRiPSInput:
<rdf:Description rdf:about="#SpaKProductionRiPSInput">
   <mts:SignalType>RiPS</mts:SignalType>
   <mts:InterfaceType>Input</mts:InterfaceType>
</rdf:Description>
```

Figure 5.16: An example of the annotation of modelling entities for a protein. Modelling entities representing RiPS and subtilin, and the default and phosphorylated forms of the SpaK protein are annotated.

The annotation of other parts such as RBSs, operators and shims is performed similarly. RBSs have mRNA input and RiPS output, and operators have PoPS input and output. Shims can have PoPS, mRNA or RiPS inputs. The output of shims is of the same type as their input. Modelling entities for these parts can also be annotated as

demonstrated.

The approach presented here can be used to construct modular models and specify rules for the composition of such models, and enable the creation of larger models. However, there is a lack of repositories available for modular models. Data integration techniques could be used to take advantage of extensive amounts of biological knowledge in order to build and parameterise reusable modular models.

5.6 Using integrated datasets to construct SVPs

One approach to expand the number of parts available for synthetic biology is the use of existing biological information which is already publicly available [58]. In addition to mining sequence-based parts, the creation of *in silico* models of these parts allows designs to be tested *in silico* through simulation.

5.6.1 Constructing SVPs

Using the information from the ontology presented in Chapter 4, SVPs were constructed for promoters and CDSs encoding TFs. Information about relationships such as the TF regulation of promoters and the TF binding of operators from the ontology were represented as models of interactions that can be used to join these SVPs together. Sequences of shims, RBSs and terminators were also extracted from the ontology as basic biological parts. All of this information about parts and their interactions can be used to construct novel regulatory networks.

Some of these TFs are response regulators that are part of TCSs. Such TFs must be activated by kinase proteins via phosphorylation interactions. Therefore, SVPs were also constructed for kinase proteins. Phosphorylation interactions between these proteins were represented as modular models.

A physical entity in the ontology may have modified versions in nature. However, the same OWL class is used to represent all different versions. For example, a protein class that has a `phosphorylated_by` restriction represents both the unmodified and the phosphorylated forms of a protein. Therefore, different molecular forms of proteins were incorporated into SVPs based on the types of their interactions.

In addition, superclasses from the ontology are used as subtypes in order to classify SVPs. For example, the ontology includes constitutive, repressible, and inducible promoters. Promoters are also classified based on sigma factors. These classifications are available as metadata in order to filter SVPs according to their functional roles. Furthermore, the ontology is used to link SVPs with additional information, including

Gene Ontology (GO) terms and the Clusters of Orthologous Groups (COG) numbers. These explicit terms enable the filtering of SVPs computationally.

5.6.2 Model parameterisation

In addition to deriving new SVPs, the integrated knowledge from the ontology was used to parameterise the strengths of promoters represented as SVPs. Promoters are characterised with transcription rates. Although information about transcription rates does not exist directly in the ontology, maximum and minimum values of normalised gene expressions are available. It has been known that prokaryotic transcription rates vary between 0.0001 and 1 mRNA per second, and 80 bp can be transcribed per second [241]. Transcription rates in the literature have also been reported within this range [107, 116, 188, 311].

In order to construct promoter SVPs with initial values, maximum normalised gene expression values were mapped into a range between 0.0001 and 1. The new rates represent relative transcription rates and can be used in simulations. The formula used for the mapping is:

$$tr_{promoter} = tr_{min} + (geneExpressionValue - minValue) * \frac{tr_{max} - tr_{min}}{maxValue - minValue}$$

where tr_{min} is 0.0001, tr_{max} is 1, and *geneExpressionValue* is the maximum gene expression value of the downstream CDS. The *maxValue* (5,963.24) and *minValue* (2.45) parameters are the maximum and minimum values among all normalised gene expression values. For example, using the formula, a gene expression value of 2,000 is mapped to 0.3351 as the transcription rate (Figure 5.17).

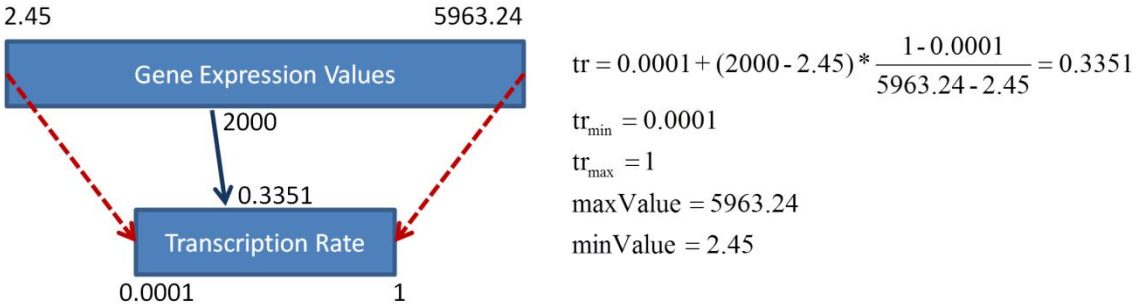


Figure 5.17: An example of the mapping of a gene expression value to the transcription rate. The normalised gene expression with the value of 2,000 is calculated as 0.3351.

Nominal values were used for other kinetic parameters. Although biochemical parameters are also important for simulations [191], obtaining these parameters has

been difficult [257], and for most cases these parameters are unknown. Therefore, quantitative parameters were selected from a range of values reported in the literature. The half-lives of mRNA and protein molecules are assumed to be 2 and 10 minutes respectively [107]. Using the formula `log(2)/half-life` [241], the corresponding rates are calculated as 0.0058 and 0.0012 per second for mRNAs and proteins respectively. 1000 nM for a TF in a bacterial cell is a high concentration, and Km disassociation constants can change between 1 nM and 10000 nM [78]. Therefore, these constants were initially assigned as 500 nM. Phosphorylation and dephosphorylation rates were selected as 0.0001 per nM per second and 0.1 per second respectively [312].

In this section, data integration is used as a source of information to mine SVPs which were constructed for promoters, RBSs, operators, functional parts, shims and terminators. In addition, the interactions captured in the integrated data were used to create models of interactions to glue these SVPs together in order to create larger models. The SVPs and models of interactions were constructed using initial variables. Moreover, promoter SVPs were quantified using the normalised gene expression values from the integrated data. These SVPs are stored in a computationally accessible repository.

5.7 BacilloBricks: A repository of SVPs for *B. subtilis*

SVPs that were built manually and created computationally from the integrated data were stored in a publicly accessible repository called BacilloBricks. The repository includes 2,996 SVPs and 699 interactions between them. Parts are provided for promoters, operators, RBSs, terminators, shims and CDSs (Table 5.2). The CDSs and their encoded products are modelled as functional parts. Currently, the SVPs in the repository are provided in the SBML exchange format.

Table 5.2: Types of SVPs and the number of models for each type.

SVP type	Number
Promoter	452
Operator	554
RBS	461
Shim	291
Terminator	1,117
FunctionalPart	121

The repository can be accessed online as a Web application[38] (Figure 5.18), which

[38] http://atgc-eidos.appspot.com

was developed as part of this project using the Java programming language. The website can be browsed manually and is also programmatically accessible. The menu provides access to parts and interactions with each page displaying 50 items. Parts are listed with their IDs, types and names. In the part view, the side bar includes links which enable the filtering of parts by type. The ID of a part is linked to the detail page for the part, providing additional information.

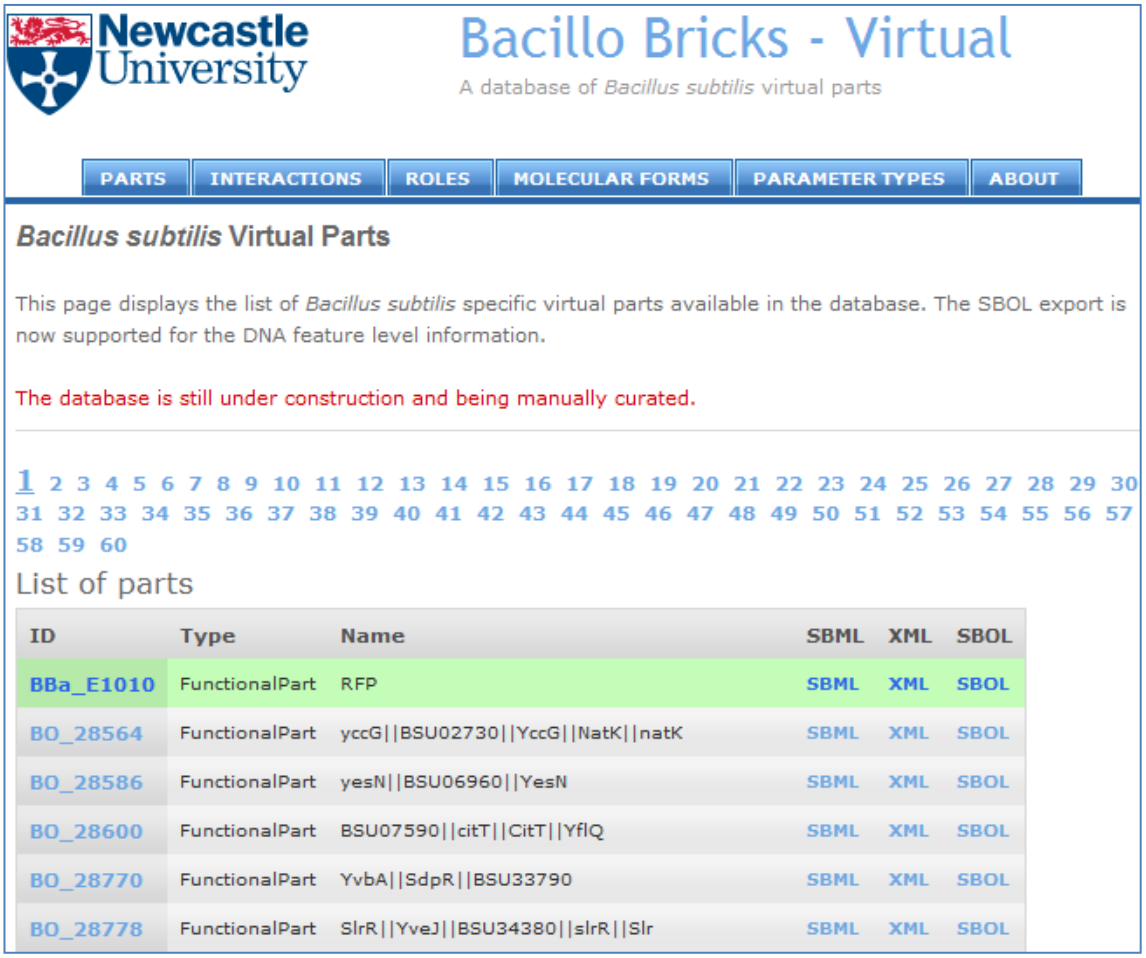


Figure 5.18: The BacilloBricks website showing the interface which allows parts to be browsed manually.

5.7.1 Standard virtual parts

SVPs have unique IDs. In the case of SVPs which are retrieved using ontology mediated data mining, the IDs are derived from the OWL classes that represent corresponding sequence features. These IDs can be used as cross references to access more information about the parts from the ontology. For example, the ontology includes additional information about SVPs such as the genome positions of sequence features and the amino acid sequences of proteins, and this information can be queried using SPARQL. Currently, IDs for manually created SVPs are given manually. IDs form

unique URIs which allow access to pages containing the details of SVPs. The details of an SVP include its ID, name, description, type, sequence, source organism and whether the SVP is manually or computationally created (Figure 5.19).

SVPs can be used to construct models with the desired behaviour. However, trying combinations of SVPs from the repository to build larger models is not time- or cost-effective. SVPs should therefore be selected according to their functions in order to plug them into a larger model. SVPs have additional types in addition to basic types such as promoters and RBSs (Table 5.3). These subtypes are populated from the superclasses in the ontology presented in Chapter 4, and are suitable for the filtering of parts.

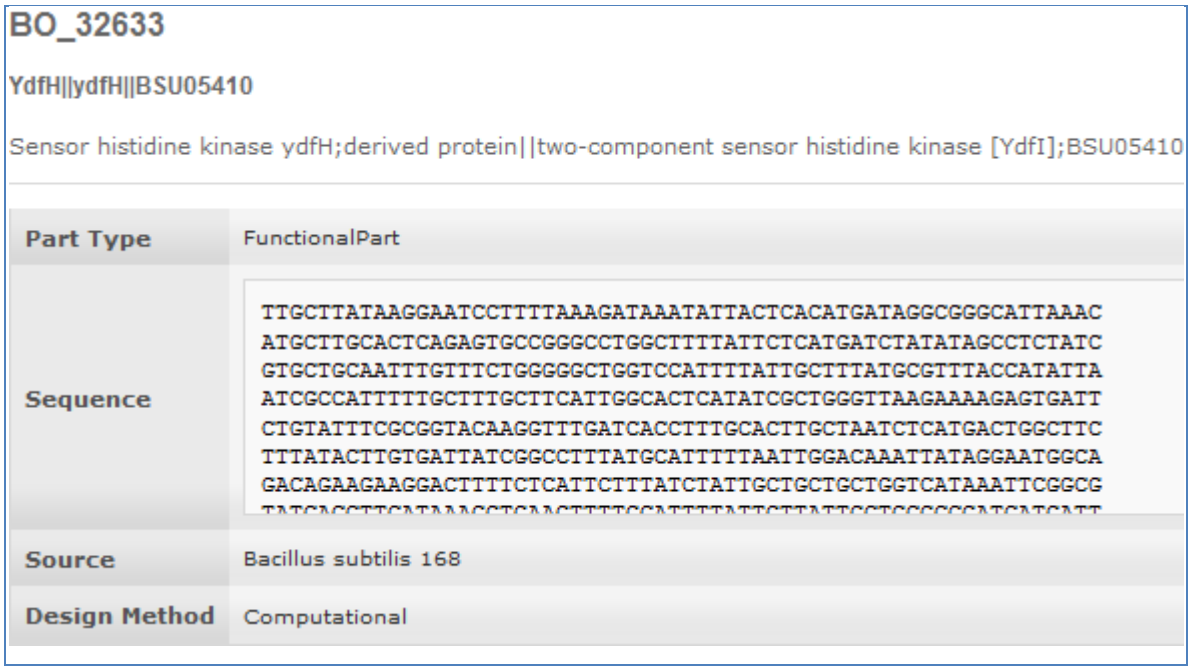


Figure 5.19: Details of the BO_32633 SVP.

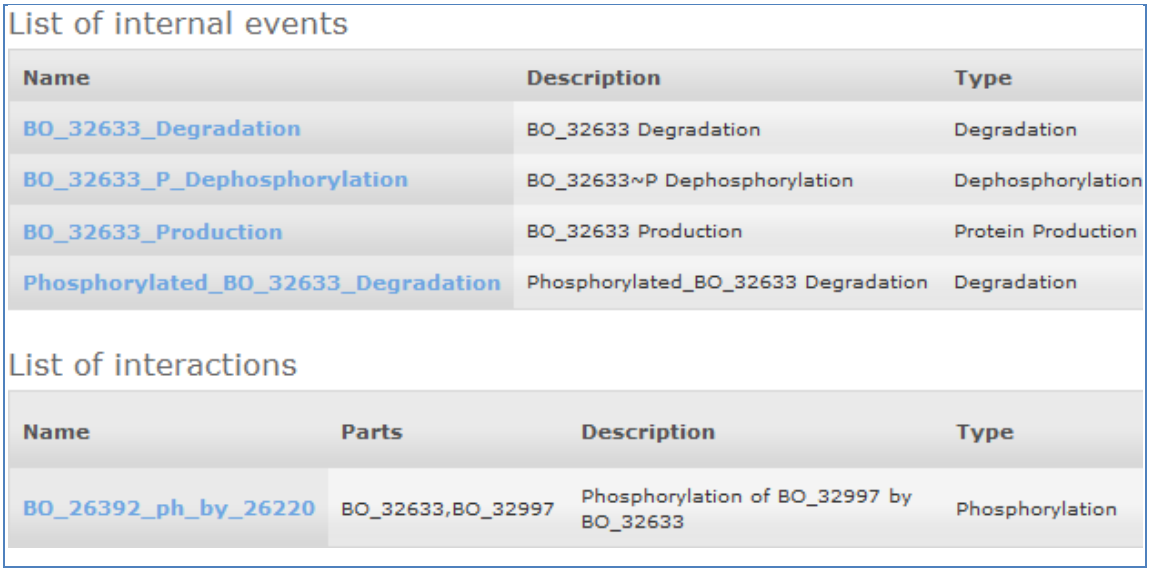
SVPs can have more than one subtype. For example, an SVP may be both a BO_SigAPromoter and a BO_InduciblePromoter, which indicates that the part modelled is an inducible SigA promoter. Similarly operators can be filtered according to the type of regulation of their TFs. FunctionalPart SVPs are of types defined by COG classification which are also linked to the ontology. SVPs also have additional properties for accession, PMIDs, and the data sources from which they are retrieved. For FunctionalPart SVPs, properties also exist for their cellular locations and molecular functions in the form of GO terms. Links to these terms from the repository provide access to all parts having the same terms.

Table 5.3: Examples of subtypes of SVPs and number of corresponding models in the repository.

SVP subtype	Type	Number
Inducible Promoter	Promoter	51
Repressible Promoter	Promoter	86
Constitutive SigA promoter	Promoter	310
Negatively Regulated Operator	Operator	333
Positively Regulated Operator	Operator	221
Transcription Factor	FunctionalPart	94

5.7.2 Models of interactions

SVPs include biochemical reactions, such as degradation, that are not dependent on other parts. Such reactions are listed as internal events and are encapsulated in the models of the SVPs. Interactions of parts with other parts are also stored in the BacilloBricks repository. Such interactions, however, must be added when interacting parts are included, in order to create simulatable models. The interactions of a part are displayed with details such as the participating parts, interaction types and descriptions, and are linked to their details pages (Figure 5.20).


List of internal events

Name	Description	Type
BO_32633_Degradation	BO_32633 Degradation	Degradation
BO_32633_P_Dephosphorylation	BO_32633~P Dephosphorylation	Dephosphorylation
BO_32633_Production	BO_32633 Production	Protein Production
Phosphorylated_BO_32633_Degradation	Phosphorylated_BO_32633 Degradation	Degradation

List of interactions

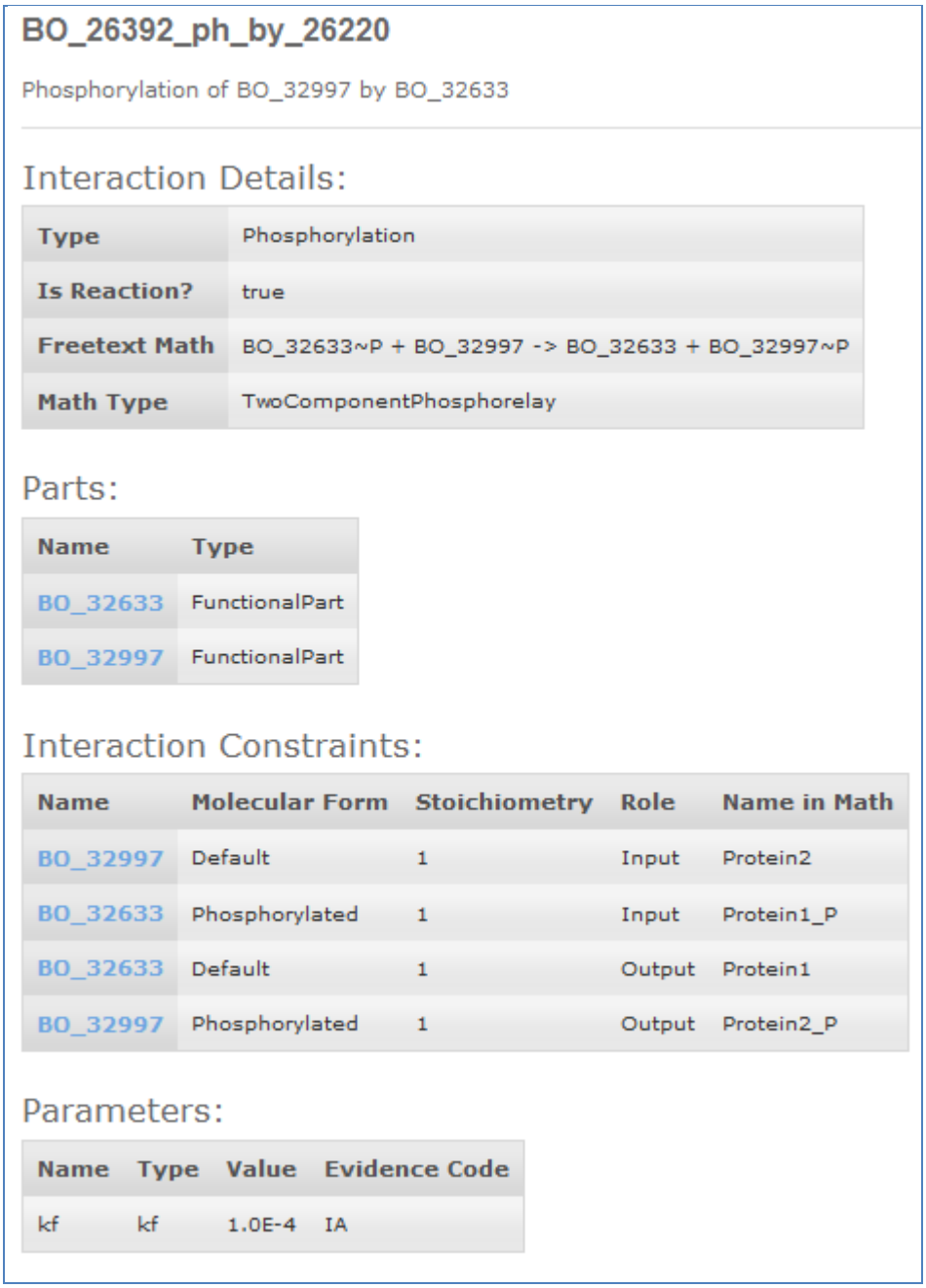
Name	Parts	Description	Type
BO_26392_ph_by_26220	BO_32633,BO_32997	Phosphorylation of BO_32997 by BO_32633	Phosphorylation

Figure 5.20: Interactions of the BO_32633 part.

The details page of an interaction includes information about the interaction's unique ID, description, a mathematical representation of the reaction and its type (Figure 5.21). Additional information includes links to the participating parts, reaction stoichiometries, the molecular forms of the parts involved and kinetic parameters.

Types of interaction types in the repository currently include transcriptional activation, transcriptional repression, phosphorylation, dephosphorylation,

transcriptional activation by an operator, and transcriptional repression by an operator (Table 5.4).



BO_26392_ph_by_26220

Phosphorylation of BO_32997 by BO_32633

Interaction Details:

Type	Phosphorylation
Is Reaction?	true
Freetext Math	BO_32633~P + BO_32997 -> BO_32633 + BO_32997~P
Math Type	TwoComponentPhosphorelay

Parts:

Name	Type
BO_32633	FunctionalPart
BO_32997	FunctionalPart

Interaction Constraints:

Name	Molecular Form	Stoichiometry	Role	Name in Math
BO_32997	Default	1	Input	Protein2
BO_32633	Phosphorylated	1	Input	Protein1_P
BO_32633	Default	1	Output	Protein1
BO_32997	Phosphorylated	1	Output	Protein2_P

Parameters:

Name	Type	Value	Evidence Code
kf	kf	1.0E-4	IA

Figure 5.21: An example of an interaction details page. Information includes the name, type, and reaction formula of the interaction, participating parts, interaction constraints and kinetic parameters.

Table 5.4: Examples of types of interactions. The second column shows the number of interactions for the corresponding types.

Interaction type	Number
transcriptional activation	44
transcriptional repression	85
phosphorylation	27
transcriptional activation by an operator	196
transcriptional repression by an operator	333

5.7.3 Computational access to the repository

The website is also able to act as a REST-based Web service. The flattened URLs may be used to serve XML documents that can be accessed computationally. The URLs can be constructed by adding '/xml' to the end of human-accessible URLs. In addition, computational models of parts and their interactions are available as SBML documents. The parts are also available in the SBOL format to conform to the standardisation efforts in synthetic biology. The Web service interface is explained below:

- /parts/page/[PAGE_NUMBER]/xml: Retrieves parts for a given page number.
  Example: /parts/page/1/xml.
- /parts/[PART_TYPE]/page/[PAGE_NUMBER]/xml: Retrieves parts for a given page number and part type.
  Example: /parts/Promoter/page/1/xml.
- /parts/[PART_TYPE]/[ROLE]/[ROLE_VALUE]/page/[PAGE_NUMBER]/xml: Retrieves parts for a given property name and value pair. Part type and page number are also passed parameters.

  Examples:

  /parts/Promoter/type/BO_RepressiblePromoter/page/1/xml: Retrieves repressible promoters.

  /parts/FunctionalPart/located_in/GO_0005886/page/1/xml: Retrieves parts that are located in the cell membrane.

  /parts/FunctionalPart/has_function/GO_0046983/page/1/xml: Retrieves parts that have dimerisation activity.

- /parts/summary/xml: Retrieves summary information such as the page count.
- /parts/[PART_TYPE]/summary/xml: Retrieves summary information for a given part type.
- /parts/[PART_TYPE]/[ROLE]/[ROLE_VALUE]/summary/xml: Retrieves summary information for a given property name, value, and part type.

- /part/[PART_ID]/xml: Retrieves the details of the part given with an ID.
  Example: /part/BO_28564/xml.
- /part/[PART_ID]/interactions/xml: Retrieves the interactions of a part given with an ID.
  Example: /part/BO_28770/interactions/xml.
- /part/[PART_ID]/sbml: Retrieves the SBML model associated with the part.
  Example: /part/BO_28770/sbml.
- /interaction/[INTERACTION_ID] /sbml: Retrieves the SBML model of the interaction given with an ID.
  Example: /interaction/BO_28770_bi_to_BO_4242/sbml.
- /part/[PART_ID]/sbol: Retrieves the sequence information of a part given with an ID in SBOL format.

Parts and interactions can also be filtered by the use of flattened URLs. These REST-based URLs provide identifiers for resources in the system and can easily be indexed by search engines. Basic types can be used to filter SVPs. For example, the /parts/Promoter relative URL lists the promoter parts. Properties of parts can also be used in a similar manner to filter parts. For example, the BO_SigAPromoter[39] subtype can be used to retrieve a list of SigA promoters. Similarly, parts can be filtered by COG number and GO term properties. For example, the GO_0000155[40] and GO_0000166[41] terms can be used to list kinase and response regulator SVPs respectively. The website can also be accessed computationally in a similar manner.

5.7.4 The BacilloBricks API

An API that enables programmatic access to BacilloBricks via a Web service was also developed. The API, called JParts, returns Java objects in response to Web service calls. Due to the huge number of SVPs in the database, it is not possible to retrieve all of the records for a particular request at once. Therefore, similar to the approach used in the BacilloBricks website and Web service, all of the SVPs can be retrieved 50 records at a time using the API. SVPs are returned with their details such as name, type, and DNA sequence. SVPs can be filtered using their types and subtypes (Table 5.5). For example, in order to retrieve the first 50 SVPs that represent SigA promoters, '`GetParts(1,"Promoter","type" ,"BO_SigAPromoter")`' method call can be used, in which `Promoter` is the type of a part and `BO_SigAPromoter` is the type of a

[39] http://atgc-eidos.appspot.com/parts/Promoter/type/BO_SigAPromoter
[40] http://atgc-eidos.appspot.com/parts/FunctionalPart/has_function/GO_0000155
[41] http://atgc-eidos.appspot.com/parts/FunctionalPart/has_function/GO_0000166

promoter. Moreover, SVPs can also be searched for according to their functional roles. For example, protein SVPs with associated GO classes can be used to find parts based on their molecular functions and cellular locations. Parts can also be retrieved by their IDs using the API.

A list of interactions for a part can be retrieved in a different method call. Additionally, SBML models of parts and interactions can be retrieved to construct models of biological systems. Models are returned using the SBMLDocument objects of the SBML's jSBML library [313] which provides Java objects for SBML entities. JParts also provides utility methods to create a container model into which SVPs can be placed and their interactions, to join SVPs, and to add mRNA SVPs. Figure 5.22 shows an example of the use of the API. The PartHandler class is used to access the models of parts and interactions. The ModelBuilder class is used to add these SVPs into a given SBML model and to join them using the defined inputs and outputs of SVPs. In the example a genetic circuit is constructed using a promoter, an RBS and the *spaK* CDS. SVPs of these parts are joined computationally in order to construct the model of this circuit.

```
PartsHandler partsHandler = new PartsHandler(serverURL);
//Create an SBML container model
SBMLDocument sbmlContainer = sbmlHandler.GetSBMLTemplateModel("SpaKProtein");
ModelBuilder modelBuilder = new ModelBuilder(sbmlContainer);
//Retrieve the models for the promoter, RBS, SpaK and mRNA SVPs
SBMLDocument pspaRK = partsHandler.GetModel(partsHandler.GetPart("PspaRK"));
SBMLDocument rbsSpaK=partsHandler.GetModel(partsHandler.GetPart("RBS_SpaK"));
SBMLDocument proteinSpaK=partsHandler.GetModel(partsHandler.GetPart("SpaK"));
SBMLDocument mRNASpaRK = partsHandler.GetModel(partsHandler.GetPart("mRNA"));
//Add the promoter SVP
modelBuilder.Add(pspaRK);
//Add the mRNA SVP and connect to the promoter SVP
modelBuilder.Link(pspaRK, mRNASpaRK);
modelBuilder.Add(mRNASpaRK);
//Add the RBS SVP and connect to the mRNA SVP
modelBuilder.Link(mRNASpaRK,rbsSpaK);
modelBuilder.Add(rbsSpaK);
//Add the SpaK SVP and connect to the RBS SVP
modelBuilder.Link(rbsSpaK,proteinSpaK);
modelBuilder.Add(proteinSpaK);
//Get the XML content of the SBML model
String sbmlOutput = modelBuilder.GetSBMLOutput();
```

Figure 5.22: An example use of the API in order to construct a simple promoter-RBS-CDS genetic circuit, using the 'PspaRK', 'RBS_SpaK' and 'SpaK' SVPs. The API includes methods to retrieve and join SVPs in order to create simulatable models.

Table 5.5: Examples for retrieving parts programmatically.

API Examples
`GetParts("SpaK")`: Gets the part with the ID of 'SpaK'
`GetParts(1)`: Gets all parts
`GetParts(1,"Promoter")`: Gets the promoters
`GetParts(1,"Promoter","type","BO_SigAPromoter")`: Gets the SigA promoters
`GetParts (1,"FunctionalPart","has_function","GO_0000155")`: Gets the parts with kinase activity

5.8 Discussion

The approach represented in this chapter demonstrates the advantages of standardising models for synthetic genetic circuits [11, 41]. Computational simulation is valuable in order to predict the behaviour of genetic circuits constructed with biological parts, and hence models of these parts should also be available. Moreover, the use of computers and automation has become a necessity in designing larger circuits than those that are currently available. Therefore, models should be in standard formats allowing the syntax and information stored in these models to be understood by different design tools. However, currently, there is no standard for the modelling of synthetic biological systems [22], although modular modelling approaches have previously been presented [11, 30, 41]. In this work, these approaches have been extended in order to construct modular models in standard formats which are also computationally composable. In addition, the SVP repository provides a useful resource for CAD and automation tools to use a wide range of SVPs for the purposes of designing complex biological systems. These approaches are discussed below.

5.8.1 Composition of models

SVPs include information about biochemical reactions that are specific to the parts being modelled, providing a one-to-one mapping between biological parts and their dynamic models. These models can be combined in order to produce simulatable models that can represent biological systems. Therefore, SVPs can be reused in the modelling of different biological systems, reducing the cost of creating new models of parts for every modelling task. In addition to the use of standard SBML, widely-adopted biological signals such as PoPS and RiPS [116] are standard inputs and outputs in these models. These interfaces make the models composable [32].

Combining models manually is a difficult process, which requires the investigation of the XML structure of dynamic models, and hence requires automation for the

efficient modelling of large biological systems. However, models are mathematical representations that usually include information about how to simulate the behaviour of a system. The semantic annotation of modelling entities can aid in the understanding of models by computational tools [202].

Model annotation as demonstrated here can facilitate the computational composition of these modular models. In this work, inputs and outputs of SVPs were annotated with machine-level information. Therefore, the process of model composition can be automated, facilitating the efficient construction of simulatable models for large-scale biological systems. In addition, these annotations were embedded in the XML structure of SVPs and can be exchanged between tools with no loss of information. These SVPs are stored in a publicly accessible catalogue and can be used by tools that adopt the standards used here, such as SBML and widely-accepted biological signals that are used as the inputs and outputs of SVPs.

5.8.2 A repository for modular models

Although a number of CAD tools have already been developed for synthetic biology [32], these tools often lack access to models of biological parts [49]. The repository described in this chapter is accessible computationally via a Web service. Although SVPs are available in standard SBML format and include machine-readable RDF annotations, programmatic access to SVPs by design tools requires the development of applications that can retrieve SVPs from the repository and join them together computationally. Thus, an API that enables programmatic access to the Web service was also developed. Design tools can hence directly access the repository in order to retrieve models of hundreds of parts using the API. Moreover, the API includes methods to add SVPs into an SBML model and join them using a simple programming interface. Therefore, this API is useful in providing easy access to the SVPs in order to construct models of biological systems.

This repository was populated using a data mining approach mediated by the ontology presented in Chapter 4. This approach demonstrates how existing biological data can be integrated and used as a basis upon which to construct dynamic models for synthetic biology. Using the knowledge encoded in the ontology, parts and their relationships were mapped to modelling entities such as species and reactions. Furthermore, data integration techniques can be used to parameterise models. For example, the normalised maximum gene expression values were mapped to relative transcription rates in order to create initial kinetic parameters for promoter SVPs. The

representation of biological knowledge about biological parts and their relationships in the form of simulatable models can be used to identify functional solutions with desired behaviour.

5.8.3 Constraining the design space

These mathematical models are suitable to facilitate the model-driven design of large-scale biological systems. Most genetic circuits are still constructed manually. Densmore compared this situation to the early 1960s when integrated circuits were also constructed manually [314]. The use of CAD tools and electronic design automation has now enabled the creation of very-large scale designs. As a result, researchers are able to integrate millions of transistors into a single chip. The availability of modular, composable models of biological parts in standard formats is similarly important to automate the searching of design spaces in order to find solutions for large-scale biological systems. Model construction is not a trivial task. This process is carried out using extensive information from available biological databases and experimental results [53]. Therefore, decoupling the creation of CAD and automation tools, and the creation of repositories of modular models would be useful to tool developers.

Composable SVPs as presented here can be used by CAD and genetic design automation tools in a bottom-up approach in order to design more complex systems. Automation tools can also take different approaches. For example, DSLs [123] are suitable for specifying a target design using a high-level description. However, designs should then be mapped to individual parts and simulated by integrating a set of equations for each part [15]. SVPs are suitable for the representation of biological functions that can emerge due to the use of biological parts and the interactions between them, and hence can be used for the verification of solutions via computational simulation.

The solution space for genetic circuits that can be constructed using parts from the repository can be very large. However, not all parts interact with each other, and hence not all designs are biologically plausible [73]. Without considering such constraints, the possible solution space for biological systems would grow exponentially [30]. The repository also provides information about interactions. Using this information, interacting partners can be identified for a target part in order to constrain the design space. These interactions are also represented as models with inputs and outputs which facilitate combining SVPs correctly in order to create simulatable models.

One of the problems with computational design in synthetic biology is the

automated selection of DNA sequences [49]. Although SVPs link physical parts to their functions, the simulation of models of circuits in order to choose circuit components can be computer-intensive. These SVPs can also be searched for using the programmatic access to the model repository. For example, the Web service interface of the repository uses metadata about parts such as their types, subtypes and properties in order to search for SVPs. These subtypes and properties are the explicit terms from GO, COG classification and the ontology presented in Chapter 4. Combination of these terms can be used to identify the most relevant parts, and hence to reduce the space of possible solutions computationally.

The modular modelling approach and the catalogue of models of biological parts presented in this chapter facilitates the large-scale engineering of complex biological systems using computational methods. The models are specified with defined levels of abstraction in the form of mathematical formulae, which is important in standardising the modelling of biological parts for synthetic biology. These modular models can be accessed and searched for computationally from the catalogue presented here. In addition, these models include machine-level information that enables their computational composition, which is particularly important when constructing models of large systems that are not feasible to do so manually.

5.8.4 Conclusion

In this chapter, a modelling approach for modular, composable models of biological parts and a catalogue of these models for *B. subtilis* have been presented. The catalogue includes 2,996 parts and 699 interactions. Rate parameters of 452 promoters from the catalogue were estimated. The catalogue is accessible as a human-browsable website, and as a REST-based Web service for computational tools. The Web service provides data and modular models for the parts and their interactions. A Java API is also available as a wrapper to the Web service in order to provide programmatic access. In addition, DNA-sequence information about biological parts can be accessed in SBOL format. SVPs have been constructed using widely-accepted standards such as SBML to represent models, information about biological signals including PoPS and RiPS to facilitate the composition of models, and RDF for the annotation of models. These modular, composable SVPs are suitable for the model-driven design of biological systems in order to facilitate the automation of genetic design. While models of large-scale genetic circuits can be constructed using SVPs in a bottom-up approach, high-level specifications of genetic designs can also be mapped to SVPs. Therefore, design

automation and CAD tools can take the advantage of readily available modular models to create complex and novel biological systems. Moreover, the automation of the conversion of the constructed models into DNA sequences could also be useful in fully automating the design of biological systems. This process is discussed in Chapter 6.

Chapter 6. Model Annotation for Synthetic Biology: Automating Model to Nucleotide Sequence Conversion

Modelling is essential for the large-scale engineering of complex and predictable genetic circuits. Dynamic models allow the behaviour of putative circuits to be observed *in silico* prior to implementation potentially saving time and resources when constructing synthetic systems. Since dynamic models capture information about the entities in a design and the relationships between these entities, in principle they can be used to derive the genetic features necessary to implement a design in-vivo. However, models that are produced computationally [22, 30, 41] can be very large. Manually deriving the DNA sequences from these models is error-prone, time-consuming and very difficult. It is therefore desirable that the process of converting dynamic models into the DNA sequences of these genetic elements should be automated. Currently, such automation is hampered by the lack of available metadata in dynamic models. This chapter presents an algorithm to convert dynamic models into DNA sequences and an annotation scheme to support this automation.

6.1 Introduction

The ultimate aim in the design of a genetic circuit is usually the production of a DNA sequence or sequences as a specification ready to construct *in vivo* [22]. In principle, it should be possible to derive the genetic level design in terms of the required genetic features and their sequence for the system directly from the model, allowing model driven design for synthetic biology. However, large-scale designs that are built by computational tools may involve large models. Therefore, as the size and complexity of such models increases, it becomes highly desirable to automate their conversion into synthesisable DNA sequences efficiently and reliably [36].

However, this conversion process is not straightforward. Firstly, standard models do not contain enough metadata to derive the type, sequence and ordering of genetic level parts, and model annotation strategies are aimed at systems biology. Annotation of models for synthetic biology needs to be subtlety different. A typical dynamic model

includes entities that represent physical molecules and the biochemical reactions they participate in [53]. However, not all elements in a computational model will have physical representations at the DNA sequence level. For example, models include entities that represent proteins, protein and RNA degradation, environmental inputs, and chassis-related factors. The automation process should be able to distinguish which entities in a model map directly to the sequence features such as promoters and coding sequences (CDSs), as well as their relationships, necessary to encode the systems modelled. Secondly, deriving the ordering for genetic features from a model is non-trivial especially for complex designs. New algorithms are required to identify the parts and the order in which these genetic parts must be assembled in the form of a DNA sequence or ordered set of genetic features.

Both issues were addressed in the work presented in this chapter. Initially, computationally-amenable metadata about the modelling entities, which facilitates the automated derivation of a DNA sequence represented by the entirety of a dynamic model, has been identified. This annotation approach is demonstrated using Standard Virtual Parts (SVPs), presented in Chapter 5. Next, an algorithm for the conversion of dynamic models of biological systems into DNA sequences is presented. The algorithm is implemented using a Java application called MoSeC. This tool can accept CellML [64] and the Systems Biology Markup Language (SBML) [65] models built with SVPs and convert them into DNA sequences in the form of standard GenBank or EMBL formats.

6.2 Model annotation for the automation of the conversion process

In order to automate the conversion process, models can be annotated with computer-readable metadata. These annotations should include the types and sequences of physical genetic parts in order to construct and validate the final DNA sequences of the genetic circuits represented by the models. Information about the ordering of genetic parts to derive these sequences should also be available. In the previous chapter, the annotation of SVPs for the automated composition of models was demonstrated using RDF. In this section, these annotations are extended to automate the model-to-sequence conversion process. The issues addressed by these annotations are discussed below.

6.2.1 Genomic context

In order to design biologically plausible genetic circuits, structural and functional

constraints must be taken into account. For example, rules can be defined such as the fact that promoters must be upstream of ribosome binding sites (RBSs). However, a modelling entity for a physical part may not be directly connected to an entity representing the downstream physical part. These entities can be joined together by additional abstract elements. These relationships between modelling entities may convey the ordering of physical parts in a genetic circuit. Therefore, *cis* interactions that include sequence-based features must be distinguished from other interactions [36].

In addition, models may include many other abstract elements in order to simulate the behaviour of the modelled circuits. Although these elements are required for models to operate mathematically, they do not represent physical entities [36]. Therefore, elements that represent physical entities and their interactions must be explicitly specified by computer-readable annotations.

6.2.2 Transcriptional and translational flux

In order to derive DNA sequences from dynamic models, transcriptional and translational fluxes between the modelling entities should be analysed [36]. These fluxes specify how biological parts such as promoters, RBSs and CDSs can be combined to construct functional genetic circuits in which fluxes of polymerases can be converted into proteins. The SVPs presented in Chapter 5 were annotated with machine-readable information in order to enable their composition into models of larger systems. Using these annotations, SVPs can be joined together based on the principle of exchanging standard biological signals such as polymerases per second (PoPS) and ribosomes per second (RiPS) [89, 116]. The inputs and outputs of SVPs also reflect these signals, and hence fluxes at the transcriptional and translational levels can be tracked in a model built with SVPs. Therefore, the connection of SVPs can be used to derive the ordering of physical parts at the DNA level.

One of the challenges for the automation of model-to-sequence conversion is the availability of metadata in dynamic models. This metadata should also be drawn up according to existing standards [36]. For example, Semantic Web technologies such as ontologies and Resource Description Framework (RDF) can be used for the annotation of dynamic SBML and CellML models [248]. In these annotations, species and reactions in SBML and components in CellML models can be marked up with RDF statements that are backed by terms from ontologies.

6.2.3 A controlled vocabulary

Ontologies are formal and explicit specifications of domains of interests [35]. They

provide a shared understanding of a domain via explicitly defined terms and their relationships. Such terms are ideal for adding biological context to mathematical descriptions in quantitative models. These models are not designed to store huge amounts of information about the biological descriptions. However, the use of ontology terms facilitates the integration of a wealth of information from a myriad of databases using computational approaches [247]. Ontology terms are also useful for identifying modelling entities that represent sequence features.

The Minimum Information Requested in the Annotation of Models (MIRIAM) proposal states that annotations for a model should also be kept within quantitative models, and that entities in the mathematical models should be linked to external information via the use of unique URIs [248]. RDF has been recognised as a standard annotation language in both CellML and SBML models [53]. Therefore, modelling entities can be linked to explicit terms from ontologies in the form of RDF triples [315]. Subjects of these triples are the annotated entities and the values are the unique URIs that represent ontology terms. A property such as 'is a' or 'has a' is used to qualify the relationships between annotated entities and ontology terms.

The Gene Ontology (GO) [204] includes terms that can be used to describe biochemical reactions. In addition, the Sequence Ontology (SO) offers a wide range of terms for sequence-based features [221]. By combining the URLs of these ontologies with the selected ontology terms, unique URIs can be constructed for the purpose of annotating modelling entities [248]. For example, the SO term `SO_0000167` describes promoter sequences. Combining this identifier with the URL of SO (http://purl.org/obo/owl/SO) produces a perennial identifier (http://purl.org/obo/owl/SO#SO_0000167) that can be used for the annotation of entities representing promoter parts.

6.2.3.1 Terms

Currently, SVPs exist for basic biological parts. In order to represent these parts, a model-to-sequence ontology with a set of classes was defined using the Web Ontology Language (OWL). These classes represent promoters, Shine-Dalgarno (SD) sequences, CDSs and shims, and are subclasses of the corresponding SO terms (Table 6.1). The `Signal_Carrier` class was defined to distinguish modelling entities that are not at the DNA level, but which are required to join entities representing physical parts. For example, mRNA entities are required to join the transcription and translation reactions by converting PoPS to RiPS. In addition, modelling entities can be used to collect fluxes

for these signals [316]. Such entities are also represented as signal carriers. Finally, the `Chassis` class is used to represent the host to be engineered. This class includes cell specific features that are required for model-to-sequence conversion and is linked to the Cell Ontology (CO) [317].

Table 6.1: The OWL classes defined in order to identify modelling entities for the model-to-sequence conversion process.

Name	Title	Description
Promoter	Promoter Standard Virtual Part	Defines promoter entities in a model. Subclass of 'promoter' from SO (`SO_0000167`).
Shine_Dalgarno_Sequence	Shine-Dalgarno Sequence Standard Virtual Part	Defines SD sequence entities in a model. Subclass of 'Shine_Dalgarno_Sequence' from SO (`SO_0000552`).
CDS	CDS Standard Virtual Part	Defines CDS entities in a model. Subclass of 'CDS' from SO (`SO_0000316`).
Shim	Spacing Sequence Standard Virtual Part	Defines shim entities in a model. Subclass of 'gene_fragment' from SO (`SO_0000997`).
Signal_Carrier	Signal Carrier Virtual Part	Defines the entities in a model that do not correspond to genetic features at the DNA level but are required for carrying or converting signals such as PoPS or RiPS.
Chassis	Chassis Virtual Part	Defines the chassis used in the model. Subclass of 'experimentally modified cell' from CO (`CL_0000578`).

6.2.3.2 Qualifiers

Annotations for the model-to-sequence conversion process are stored in the form of RDF documents. An RDF document is linked to the corresponding modelling entity via the `rdf:about` property, and includes RDF statements required for the conversion. In order to facilitate the conversion of models into DNA sequences, a standard set of qualifiers has been described for the annotation of SVPs (Table 6.2). In addition to the types of parts annotated in a model, the individual sequences of physical parts must also be available in order to derive the DNA sequence for the entirety of a model. The `IsDNABased` qualifier shows whether or not a part has an associated DNA sequence. If this attribute is set to TRUE, the DNA sequence of the part must be provided by either the `Sequence` or `SequenceURI` qualifiers. The `Sequence` qualifier allows the

nucleotide sequence of a physical part to be included directly with the SVP part definition. However, as the size of the constructed models becomes larger, including the DNA sequence for every CDS or promoter may substantially increase the size of models. In this case, the latter qualifier, `SequenceURI`, can be used store the URI of the FASTA sequence for a specific modelling entity. The IDs of modelling entities are unique, and hence may consist of long string of alphanumeric characters. For visualisation purposes, the `VisualName` qualifier is used to store a human-readable name. The `TerminatorSequence` qualifier is used to store the sequence of a terminator that can be used at the end of transcriptional units.

CellML 1.1 supports explicit modularity via the file structure of quantitative models. Files can import other models and reuse the mathematical definitions provided by external files. SVPs can also be constructed using files that act as templates which have mathematical formulae. SVPs are parameterised with kinetic values and are linked to these templates [80]. However, not all imported files are templates. SVPs can also be imported by models, and any model that contains components and connections can be used for the construction of larger models. Therefore, the `IsDNABasedPartTemplate` qualifier is used to label templates for SVPs that represent sequence-based features. For other templates, the `IsTemplate` qualifier is used. These qualifiers are identified with a namespace[42] prefixed with `mts`.

Table 6.2: Qualifiers required for model-to-sequence conversion.

Name	Description
Sequence	Nucleotide sequence of a physical DNA. Required for DNA-based parts if 'SequenceURI' is not provided.
SequenceURI	URI that points to a remote location that contains the nucleotide sequence in FASTA format for a DNA-based part. Required if 'Sequence' is not provided.
VisualName	Name of a model constituent to appear when visualised. Optional.
IsDNABased	Shows whether or not a part is DNA based.
IsTemplate	Shows that the modelling entity is used as a template for SVPs. For CellML 1.1 models only.
IsDNABasedPartTemplate	Shows that the model entity is used as a template for DNA-based SVPs. For CellML 1.1 models only.
TerminatorSequence	A terminator sequence used to terminate each operon construct. Chassis specific.

[42] http://purl.org/modeltosequence/1.0#

6.2.4 Annotation of SVPs

In this section, entities in SVPs in the form of CellML or SBML models are annotated with metadata to facilitate the conversion of models into DNA sequences, extending the annotations described for the composition of models. RDF annotations embedded in models are mapped to relevant modelling entities. The annotations described here are demonstrated using the model of the subtilin receiver device presented in Chapter 5. In the subtilin receiver model, expressions of the SpaK and SpaR proteins are driven by a constitutive promoter. On sensing subtilin, SpaK is phosphorylated and activates SpaR, which in turn regulates another promoter to express the green fluorescent protein (GFP). The model includes modelling entities that represent promoters, RBSs, shims and CDSs. These entities are annotated using the terms and qualifiers described earlier (Section 6.2.3).

Figure 6.1 shows the annotation of the *pspaRK* promoter SVP. The type of SVP is specified with `mts:promoter` using the `rdf:type` property. The sequence associated with the SVP and the name used when visualised are specified by `mts:Sequence` and `mts:VisualName` properties, respectively.

```
<rdf:Description rdf:about="#BugBuster_Promoter_spaRK">
   <rdf:type rdf:resource="mts:promoter"/>
   <mts:Sequence>gcatgaaataaa...tataatatttg</mts:Sequence>
   <mts:VisualName>pspaRK</mts:VisualName>
   <mts:IsDNABased>true</mts:IsDNABased>
</rdf:Description>
```

Figure 6.1: Annotation of the *pspaRK* promoter SVP.

Similarly, SVPs for shims, SD sequences and CDSs are specified with the `mts:shim`, `mts:CDS` and `mts:Shine_Dalgarno_Sequence` terms respectively. The types of modelling entities in CellML and SBML models that are not mapped to sequence features but are used for the exchange of biological signals such as PoPS, RiPS and mRNA are specified with the `mts:SignalCarrier` term (Figure 6.2).

```
<rdf:Description rdf:about="#RNA">
   <rdf:type rdf:resource="mts:Signal_Carrier"/>
</rdf:Description>
```

Figure 6.2: Annotation of a signal carrier modelling entity.

A modelling entity that represent the chassis is specified with the `mts:Chassis` term via the `rdf:type` property. A terminator sequence that can be used to end

transcription is also specified as part of the chassis annotation (Figure 6.3).

```
<rdf:Description rdf:about="#Chassis_Bacillus">
   <rdf:type rdf:resource="mts:Chassis"/>
   <mts:TerminatorSequence>taaaggac...tgttaa</mts:TerminatorSequence>
</rdf:Description>
```

Figure 6.3: Annotation of the chassis.

6.2.4.1 Annotating CellML models

CellML is an XML-based model exchange language, and annotations in the form of RDF/XML can be stored directly in the XML structure of CellML component entities [64]. Therefore, RDF annotations for promoter, shim, CDS and SD sequence SVPs are placed inside CellML components and linked to the `cmeta:id` attributes of these components.

Other CellML components that are used for the exchange of biological signals, such as PoPS, RiPS and mRNA, are given with `mts:SignalCarrier`. Since the components that represent mRNAs are regarded as signal carriers, components that collect the reaction fluxes [316] for mRNA production and degradation are also annotated as signal carriers in order to create a flow of biological signals via the annotations.

A CellML 1.0 model is formed from a single physical model file. The metadata required for the conversion process is also stored in the same file. However, CellML 1.1 allows individual models to be imported [64]. Therefore, CellML components can be distributed across multiple model files. Using this approach, SVPs can be constructed using templates and can be included in larger models [80]. However, the metadata required for the conversion should also be distributed between SVPs and templates.

SVP templates have common metadata that can be used by all inheriting SVPs. Currently, this information includes the type of SVPs that can be generated from a template, and whether or not the template is used for the construction of SVPs (Figure 6.4). Therefore, templates provide both mathematical formulae and metadata that are shared by the same type of SVP. SVPs are models of physical parts, and have part-specific metadata, such as DNA sequences. However, this metadata should be combined with information stored in the SVP templates for the model-to-sequence conversion.

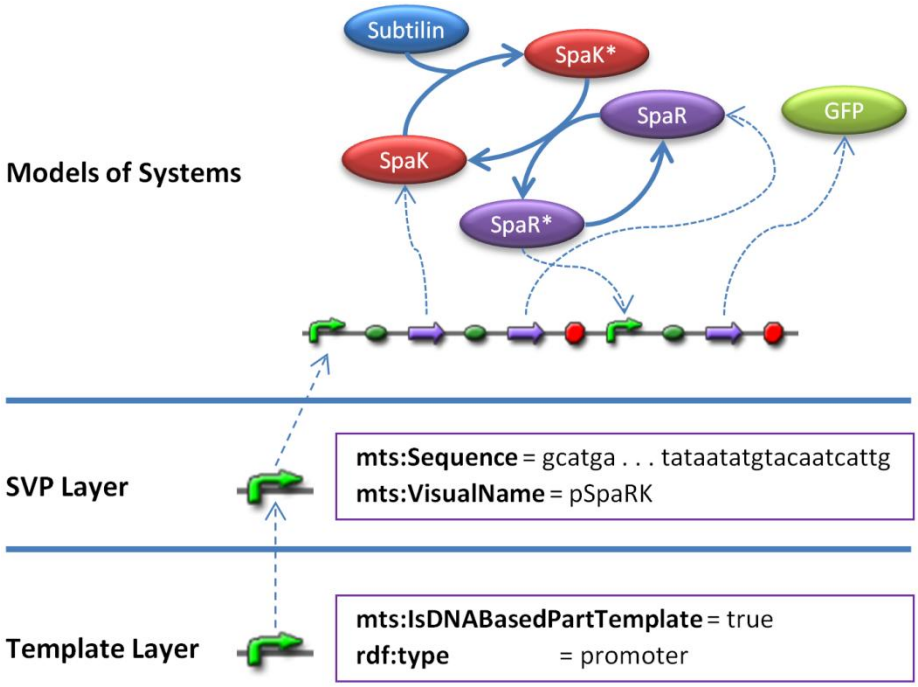


Figure 6.4: Model annotation with SVPs for CellML 1.1 models. Metadata is split between SVPs and their templates. In this example, the *pspaRK* promoter SVP template is annotated with `rdf:type` and `mts:IsDNABasedPartTemplate` properties. The promoter SVP is annotated with `mts:Sequence` and `mts:VisualName` properties.

6.2.4.2 Annotating SBML models

SBML models constructed using SVPs are currently formed from single files. Therefore, the annotation of SVPs is not layered. Each SVP contains all of the required metadata for the conversion process. In Section 3 of the SBML specification [250], it is advised that application specific annotations are placed into a single top-level element inside the annotation tag of the relevant modelling entity and identified with a namespace specific to the application. Therefore, RDF annotations for the model-to-sequence conversion are placed into the `ModelToSequence` top-level elements inside the annotation tags of modelling entities such as species and reactions. In addition, chassis-related metadata is provided via the compartments in SBML using the annotation tag of a compartment modelling entity that represents the cell. In order to conform to the MIRIAM standards, models are also annotated appropriately as suggested by the SBML specification.

6.2.4.3 Constructing MIRIAM compliant models

In order to increase the ability of computational tools to understand mathematical models, entities representing physical molecules and biochemical reactions should be annotated with a corresponding biological context [202]. For example, species modelling entities should be linked to entries from the UniProt [209] protein database.

This mapping can be achieved via MIRIAM URIs [248] and can be used to access a wealth of biological information. The use of these URIs and BioModels qualifiers is a standard way of annotating SBML models, and these annotations are called controlled annotations [250]. Modelling tools such as COPASI [244] can also read these annotations.

Therefore, the protein species in the subtilin receiver model were annotated with a set of protein IDs from UniProt. For the SpaR protein, this MIRIAM URI is `urn:miriam:uniprot:P33112`, in which P33112 is the UniProt ID of the SpaR protein (Figure 6.5). The modelling entity for the phosphorylated form of SpaR also uses the same ID. However, the information about the molecular form of the protein can be retrieved via the `mts:MolecularForm` property as described in Chapter 5.

```
<rdf:Description rdf:about="#SpaR">
   <bqbiol:is>
      <rdf:Bag>
         <rdf:li rdf:resource="urn:miriam:uniprot:P33112"/>
      </rdf:Bag>
   </bqbiol:is>
</rdf:Description>
```

Figure 6.5: The MIRIAM annotation of the species modelling entity representing the SpaR protein. The species is linked to the UniProt entry P33112.

Similarly, biochemical reactions should also be annotated. The reactions represented in the model are phosphorylation, dephosphorylation, RNA degradation, protein degradation, transcription and translation, and they are linked to GO terms using MIRIAM URIs (Table 6.3).

Table 6.3: Mapping of modelling entities that represent biochemical reactions and their GO term annotations.

Reaction	GO term
Phosphorylation	GO:0016310
Dephosphorylation	GO:0016311
RNA degradation	GO:0006401
Protein degradation	GO:0009056
Transcription	GO:0009299
Translation	GO:0006412

6.2.4.4 Using integrated datasets to annotate SVPs

The subtilin receiver model presented in this chapter was constructed with SVPs that are pre-annotated manually. However, as the number of models increases, it becomes

difficult to manually annotate individual models. The annotation of SVPs should therefore be automated. This process requires the availability of information about the biological context represented by modelling entities in computer-readable formats. As data integration techniques can be used to create models of parts, the same techniques can also be used to increase the understanding of models for computational tools.

A catalogue of SVPs was presented in Chapter 5. These SVPs were annotated with metadata in order to facilitate the construction of larger models computationally, and were created using the information from the ontology presented in Chapter 4. This ontology includes a variety of information about parts, such as their types and nucleotide sequences that encode them. This information has been used to annotate SVPs for model-to-sequence conversion using the terms and qualifiers as described in this chapter. The MIRIAM annotations describing biochemical reactions have also been added to SVPs. Therefore, SVPs from the catalogue can be combined to create models that can be used as input to computational tools capable of deriving DNA sequences represented by these models.

Computational models can be constructed from biological networks by adding quantitative parameters [77]. These models can also be visualised using networks to analyse the modelling entities and their relationships. This representation allows the use of graph analysis techniques to investigate the connectivity of modelling entities computationally. The fluxes between transcription and translation events can therefore be tracked in graph representations of models in order to derive the ordering of parts.

6.3 Conversions of CellML and SBML models to graphs

Graphs have been used to model the topology of various biological data such as protein-protein interactions, transcriptional regulatory networks and gene expression [157]. Both CellML and SBML models include modelling entities that represent biological molecules and their biochemical reactions, and which can be represented as networks. These models support ordinary differential equations (ODEs) to describe physical molecules. Each reaction flux contributes as a gain or a loss in order to construct an ODE for a biological molecule. Therefore, these models are networks of interconnected ODEs formed of nodes in the form of reaction fluxes, participating molecules and mathematical rules. In order to follow the flow of information, these models should be represented appropriately as networks.

6.3.1 SBML models as graphs

A typical SBML model is formed of a list of biochemical reactions [65]. These reactions have substrates and products that are modelled with species. Depending on whether a species is a substrate or a product, the reaction flux contributes as a loss or gain for the ODE expression of the species. Species can also have modifier roles in which they catalyse reactions. Such reaction fluxes do not affect the final concentration of species. The complete ODEs for species are not visible in an SBML model, since ODEs are constructed implicitly during the simulations. In addition, in their mathematical formulations, reaction modelling entities can refer to globally defined assignment rules that are constructed for species, kinetic parameters or other rules.

Therefore, an SBML model is a network of nodes and edges, in which nodes represent species, reactions and rules, and edges represent the relationships between these nodes. The following rules are applied to represent SBML models as networks for model-to-sequence conversion [36]:

- Assignment rules, species and reactions are represented as nodes.
- If one assignment rule is used in another, an edge is created from the former to the latter.
- If a species is used in an assignment rule, an edge is created from the species to the assignment rule.
- If an SBML assignment rule is used in a reaction, an edge is created from the rule to the reaction.
- If a species is a reactant or modifier of a reaction, an edge is created from the species to the reaction.
- If a species is a product of a reaction, an edge is created from the reaction to the species.

6.3.2 CellML models as graphs

The representations of CellML models as graphs are simpler than is the case with SBML models. CellML models include two basic modelling entities called components and connections [64]. Components have mathematical formulations in the form of ODEs or simple algebraic rules. Parameters used in these formulations are represented by variable modelling entities. Variables are encapsulated inside components and can be public or private. They can also be defined as being an input or an output. Public variables are used to join components via connection entities. In SBML, species modelling entities act as a common interface to collect reaction fluxes in order to build

ODEs. In CellML, these reaction fluxes have to be collected explicitly. Therefore, in order to reuse reaction fluxes from different components, an interface component is used for each cellular molecule modelled [316]. Each reaction flux is represented by a variable in these interface components, and an ODE is explicitly written for a variable that represents a species output.

Therefore, a CellML model is a network of nodes and edges in which nodes and edges are components and connections respectively. However, the situation becomes more complex with CellML 1.1 models. CellML 1.1 provides a hierarchy of models for a particular system that is to be modelled [64]. The main model file is linked to individual models and imports the components from these individual models. Each import is specified with the URI of the model, together with a list of components to import. As the new connections can be defined once the components are imported, existing connections of components are implicitly included from the imported files. Therefore, the following rules are applied to represent CellML models as networks for model-to-sequence conversion:

- Components are represented as nodes.
- Connections are represented as edges.
- When the output of a component is an input to another component, an edge is created from the former to the latter.
- Connections of components can also be defined in the imported files without the need to redefine them. For such a connection, if both components are used in the final model, the connection is represented as an edge between the two components.

In the previous section, an annotation scheme that can be used to store types and DNA sequences of parts in dynamic models has been described. The graph analysis of models as described in this section, along with the annotations, can be used to investigate the relationships of entities in order to derive DNA sequences for models. However, for the purposes of a computational approach, the steps required for this process must be explicitly specified. This issue is addressed in the next section.

6.4 An algorithm for graph to sequence conversion

This section describes a model-to-sequence conversion algorithm that can be used to convert models into DNA sequences. Each model includes modelling entities for a genetic circuit composed of a set of transcriptional units. Models are constructed with SVPs, and hence relevant entities can be mapped to physical parts such as promoters and RBSs. The ordering of parts is determined using the graph analysis of modelling

entities. Information flows between these entities, such as in the conversion of PoPS to RiPS via mRNAs, form the basis for the derivation of DNA sequences for each transcriptional unit (Figure 6.6). The final sequence that represents an entire model is then formed from the combination of the sequences of these transcriptional units.

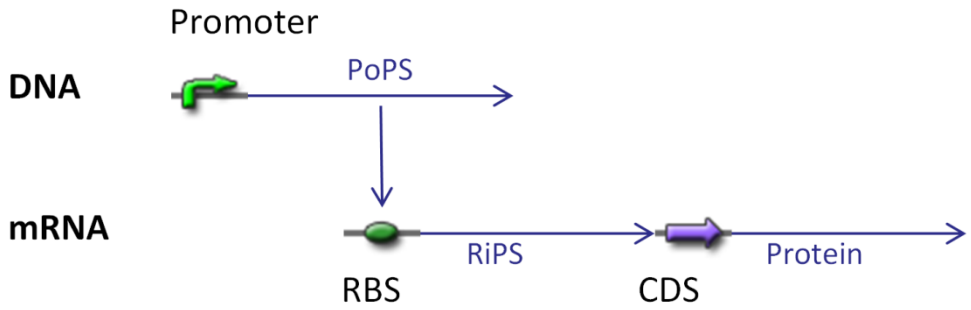


Figure 6.6: The flow of information at the DNA and RNA level. At the DNA level, PoPS is converted to mRNA. At the mRNA level, RBSs convert mRNAs into RiPS as input for CDSs. Used from [36].

The conversion algorithm includes the following steps:

1. **Convert the XML representation of a model to a graph representation.**

Initially, the model is represented as a graph according to rules specified in the previous section. When models are constructed with SVPs, edges between entities that are for physical parts represent biological signals such as PoPS, RiPS and mRNA. Although the number of nodes and edges and the connectivity of nodes in a model can differ between CellML and SBML models, the flow of information between physical part entities remains similar. Therefore, the same algorithm is applicable to both CellML and SBML models.

2. **Identify nodes that correspond to DNA-based parts.**

Modelling entities that are mapped to physical parts are identified via computer-readable annotations that include types of parts and nucleotide sequences. This information is used to derive the DNA sequences of the modelled systems.

3. **Remove all non-*cis* interactions.**

A dynamic model includes many modelling entities that are not mapped to physical parts. Although entities such as activation or degradation are necessary for the simulation of the model, these entities are omitted from the graph since they are not necessary when deriving the sequence information. The remaining subgraph is composed of nodes that represent physical parts. Entities that are annotated as signal carriers are used to join physical parts. The relationships of these entities count as *cis* interactions, and hence are also kept in the final graph. Examples include RNA-based

entities.

4. **Identify the start node.**

Start nodes should initially be identified in order to derive the ordering of parts by following biological signals. As annotations can be used to identify these nodes, the topology of the graph can also reveal the start nodes. For example, nodes without incoming edges from other nodes are candidate start nodes.

5. **Join the DNA-based parts in the direction specified by the edges.**

Beginning from the start nodes, DNA-based parts are joined in order to create a final DNA sequence. The direction of the flow of PoPS and RiPS are determined by following the directed edges in the graphs.

6. **Identify the subgraphs of physical components joined by mRNAs.**

Operon structures consist of more than one RBS-CDS pairs which can be joined by shims. It is assumed that the rate of transcription is uniform throughout an operon. Therefore, after mRNA modelling entities, several branches of subgraphs can be formed. In these subgraphs mRNAs are converted into proteins.

7. **Join the branches to form the operon structures.**

These branches should be joined to form operon structures. Currently, this is done by taking the left branch first. However, design patterns from nature could be investigated in order to derive more optimal rules for this process. For example, in *B. subtilis*, the histidine kinases precede the response regulators in NarL type two-component systems (TCSs), whereas in TCSs from OmpR families, the response regulators are situated first [318]. Therefore, future designs could be optimised in the light of existing biological knowledge.

8. **Add terminators to the end of each transcriptional unit.**

Each transcriptional unit must end with a terminator sequence. Although terminators are not modelled as part of SVPs, the sequences of terminators must be accessible to the conversion tool. The selection of terminator sequences can be conducted via annotations that include terminator sequences or by retrieving them directly from part databases.

9. **Concatenate the sequences of the transcriptional units to form a linear DNA structure.**

Transcriptional units are then joined together to form the final DNA sequence which provides the genetic information that encodes the modelled dynamic behaviour.

The previous section has described how a graph representation of models built with SVPs could be constructed for the purposes of deriving the ordering of parts. Using the algorithm presented in this section, these graphs can be converted into DNA sequences.

SVPs are also annotated with machine-readable metadata that can facilitate the automation of this algorithm. The implementation of this conversion is described in the next section, using a model-to-sequence conversion tool.

6.5 MoSeC: A model to sequence conversion tool

MoSeC is a tool that implements the model-to-sequence conversion algorithm presented in the previous section, in order to convert computational models into synthesisable DNA sequences. The output of the tool is in the form of widely-accepted EMBL or GenBank-formatted files. The algorithm is applicable to both CellML and SBML models built with SVPs. In order to perform this task, modelling entities such as species and reactions in SBML or components in CellML are marked up with metadata. Annotations give semantic meaning to the modelling entities and facilitate computational access for the conversion. MIRIAM compliant models annotated with the additional minimum information, required for the conversion, are used as input to MoSeC.

6.5.1 Minimum information required for this automation

In order to generate a DNA sequence for a model, SVPs representing parts of a transcriptional unit such as promoters, SD sequences and CDSs must be included. In addition, these SVPs must be annotated with the minimum amount of information required for the conversion process. SVPs that correspond to physical parts must be marked up with the `mts:IsDNABased` property, and their types and sequences must be provided by the `rdf:type` and `mts:Sequence` properties respectively. Signal carrier types of parts such as mRNAs must also be identified. For CellML 1.1 models, templates of SVPs for physical parts must be marked up with the `mts:IsDNABasedTemplate` property. In addition, terminator sequences must be available via the annotation of the chassis modelling entity using the `mts:TerminatorSequence` property.

6.5.2 Implementing the conversion algorithm

The conversion of models using MoSeC was applied to the CellML version 1 and 1.1, and SBML version 2 models of the subtilin receiver (Figure 6.7). The models were annotated with the information required for the automation of the conversion. MoSeC was used to generate EMBL/GenBank-formatted DNA sequences for the device. Models are represented as graphs, and the non-*cis* interactions are removed. DNA-based parts are then joined and the final DNA sequences are produced.

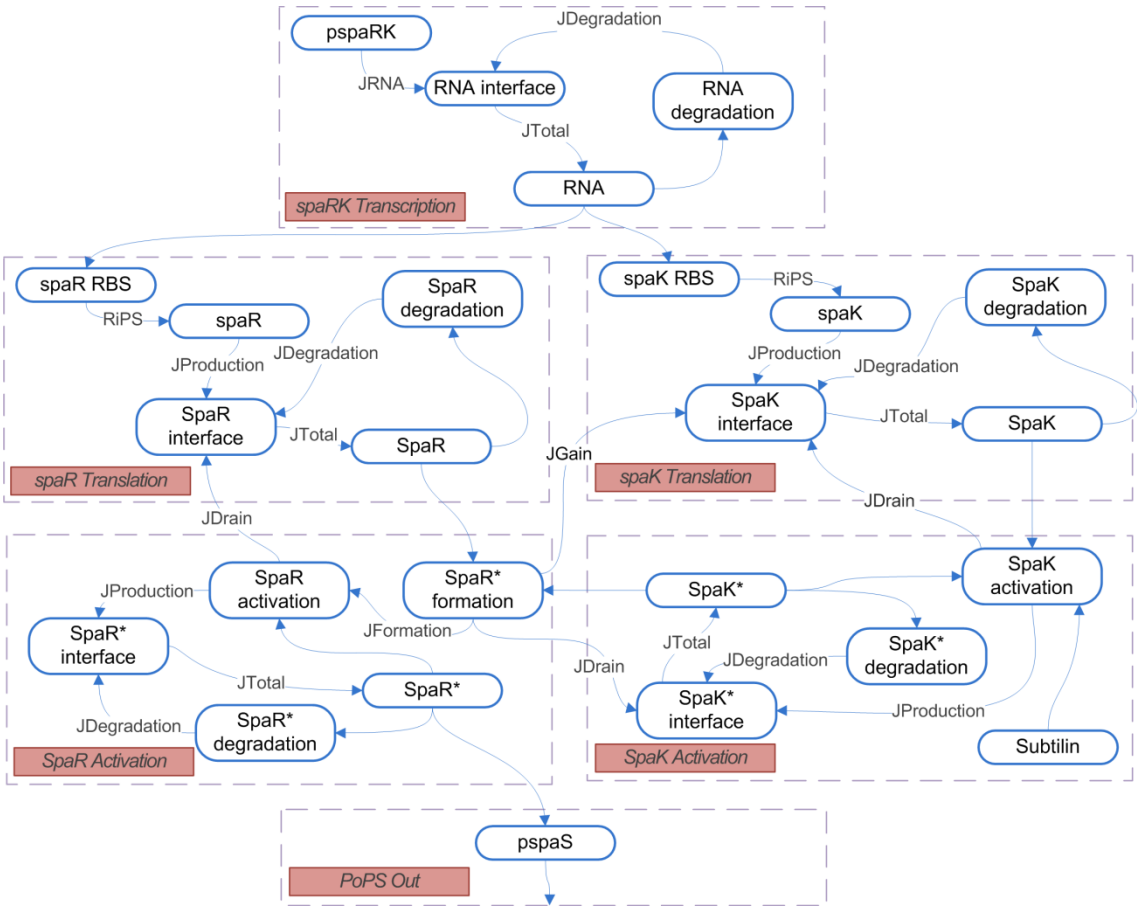


Figure 6.7: An overview of the CellML model of the subtilin receiver. In the figure, transcription from the *pspaRK* promoter, SpaR and SpaK translation, and SpaR and SpaK activation components are grouped together. Edges show the reaction fluxes exchanged between the components. The output of the device is a PoPS signal that can be connected to express, for example, GFP.

6.5.2.1 Producing the graph representation of a model

MoSeC initially converts XML-based CellML and SBML models into graphs. The tool also provides a visualisation platform for these graphs. In MoSeC, circles represent the modelling entities and the lines between the circles represent the relationships of the corresponding modelling entities. Nodes are labeled using annotations via the `VisualName` property. If a modelling entity is not annotated with this property, its ID in the model is used as its visual name. Edges are directed and are constructed according to the rules described in Section 6.3. Edges are also labeled. For CellML models, labels have the *Output variable-Input variable* format. The names of output and input variables that are used to connect components are joined together to produce the edge labels. For SBML models, labels have the *Entity1-Entity2* format in which Entity1 is the modelling entity that is used by the Entity2 modelling entity.

For dynamic models that have all of the annotations in a single file, this conversion process is straightforward. However, the use of template files present in CellML 1.1

models requires the collation of annotations for an SVP. Templates store the types of SVPs that can be instantiated, and SVPs represent physical parts and thus carry the DNA sequence annotations. MoSeC combines all of the annotations for an SVP and renders the SVP as if loading from a single file. In this approach, template files are omitted from the graph representation in order to simplify the conversion. In addition, annotations applied to the SVPs are read by MoSeC in such a way that the models in the upper level in the hierarchy overwrite the annotations coming from lower levels.

The availability of computer-readable information about SVPs for physical parts enables MoSeC to identify these entities. Such SVPs are displayed with their corresponding part icons. Figure 6.8 shows the graph representation of the CellML model in MoSeC. In this graph, nodes for two promoters and three RBSs can be identified visually. These nodes are joined to other nodes and edges. The directions of the edges show the flow of information. Starting from a promoter, this flow branches after the first mRNA and joins again via the activation of the SpaR protein by SpaK. The activated SpaR is connected to a second promoter to drive the expression of GFP.

The graph-based representation of the SBML model is also very similar (Figure 6.9). The graph includes fewer nodes when compared with the CellML version of the graph due to the implicit derivation of ODEs for species in SBML. However, the same conversion algorithm is applicable to both CellML and SBML models, since the flow of information can be tracked in a similar manner in both graphs.

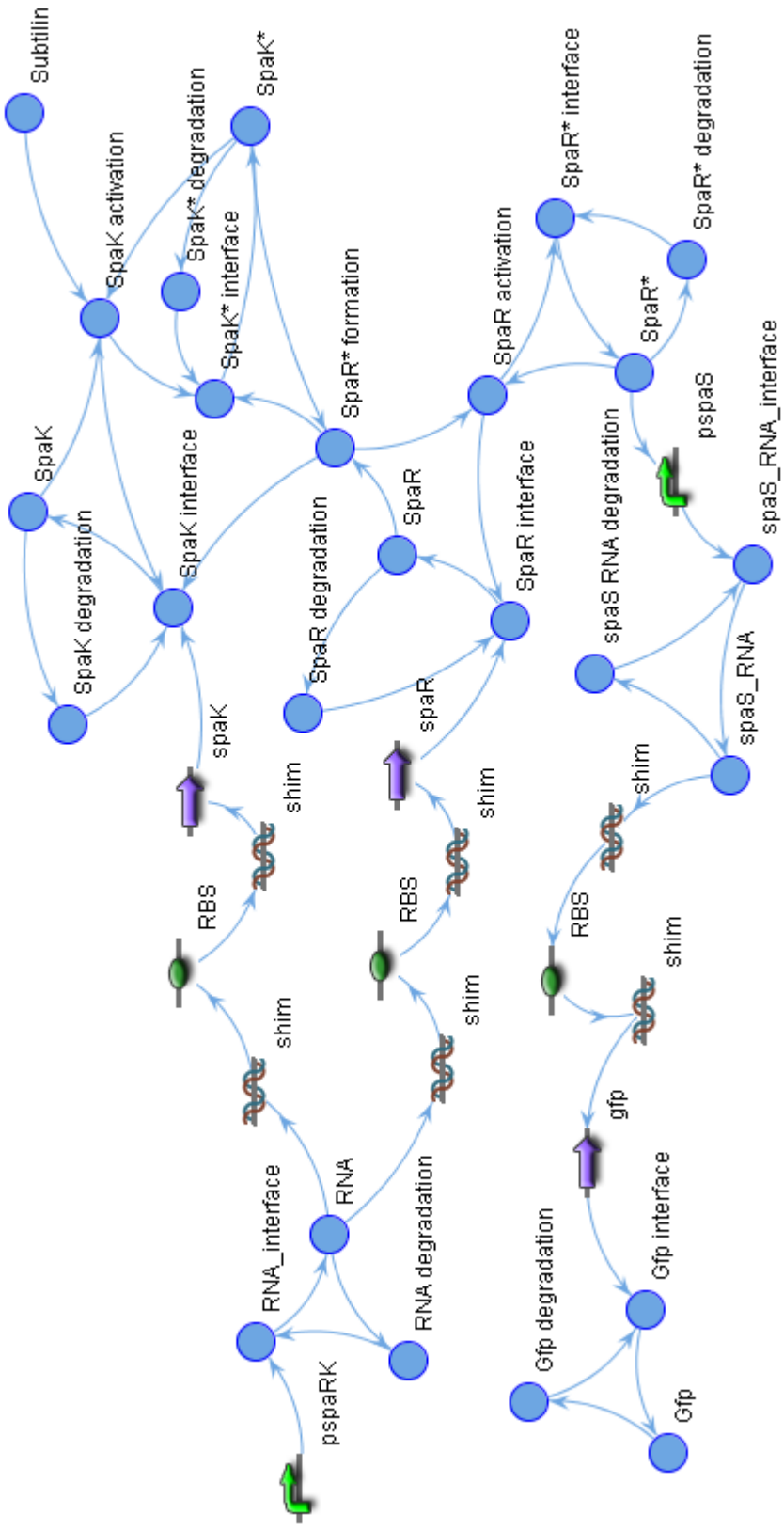


Figure 6.8: The graph-based representation of the CellML model of the subtilin receiver device in MoSeC. Physical entities such as promoters, RBSs, shims and CDSs are displayed with corresponding icons. SpaK* and SpaR* represent the activated SpaK and SpaR respectively.

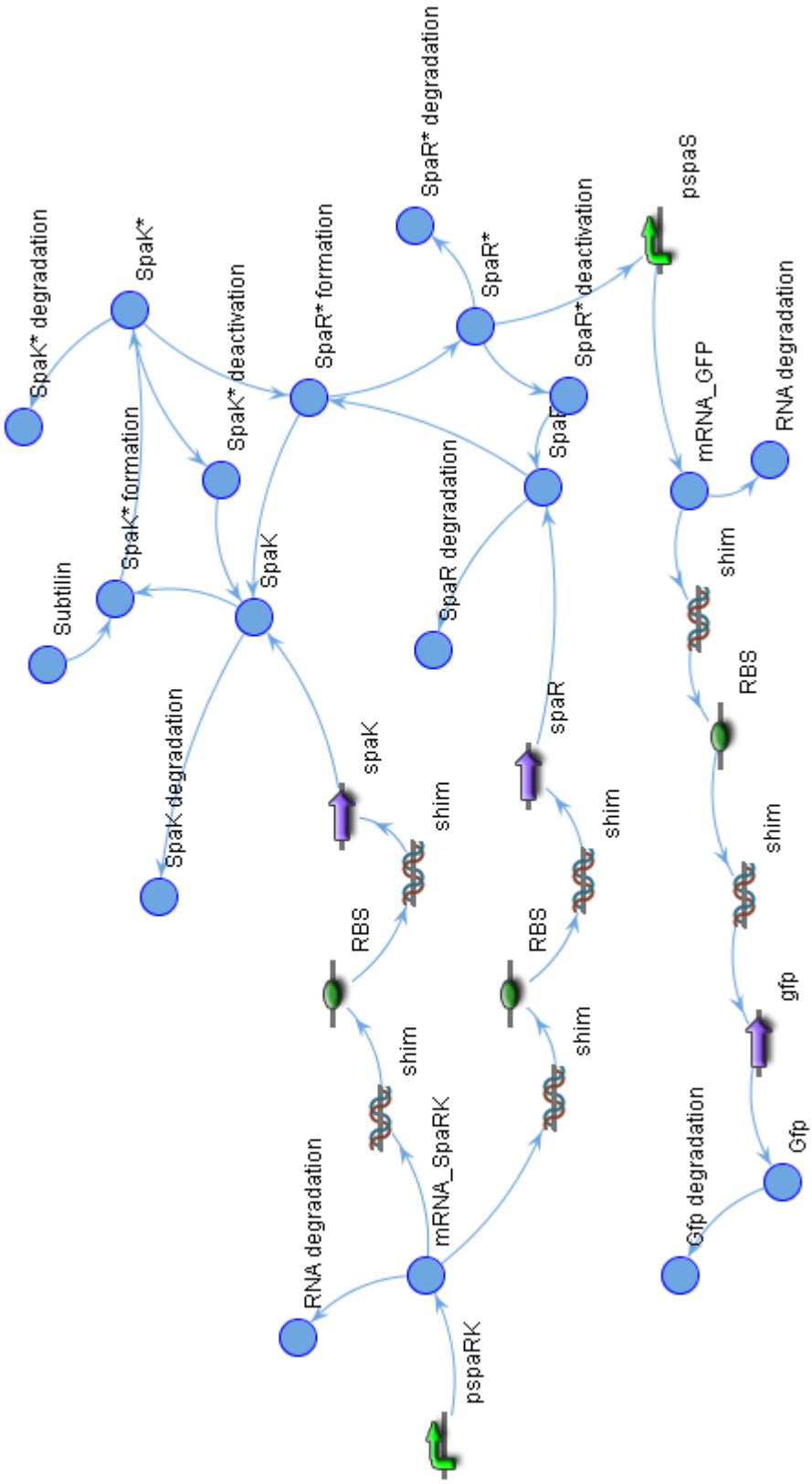


Figure 6.9: The graph representation of the SBML model of the subtilin receiver device in MoSeC. SpaK* and SpaR* represent the activated SpaK and SpaR respectively.

6.5.2.2 Removing non-*cis* interactions

Using the available annotations, MoSeC identifies nodes that represent physical entities. Interactions between these entities count as *cis* interactions. All other non-*cis* interactions can be removed (Figure 6.10). Interactions between RNA-based components also count as *cis* interactions, and so are not removed. The corresponding edges are used to combine the nodes that represent physical items between transcription and translation events. Unconnected nodes are omitted from the graph.

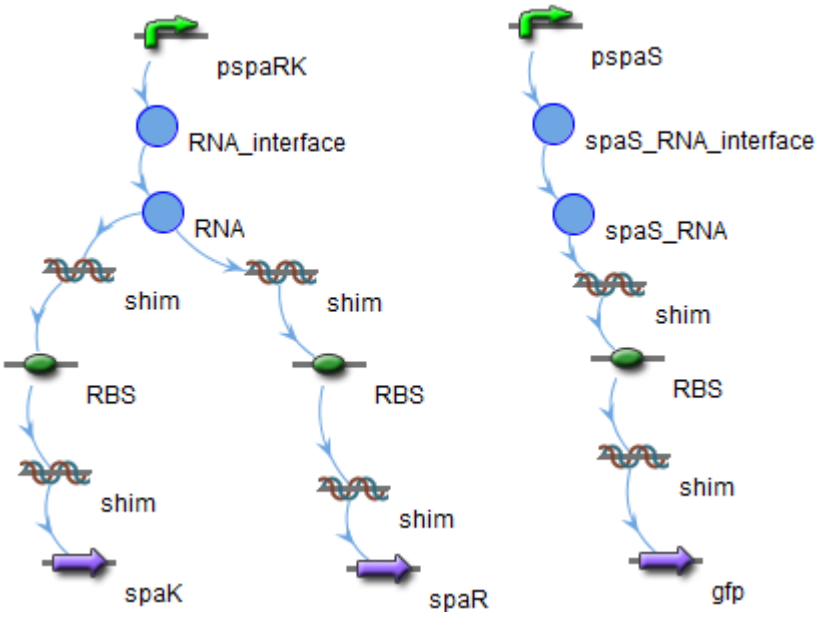


Figure 6.10: Non-*cis* interactions are removed. A subgraph of nodes that represent physical entities and other signal carriers is retained.

6.5.2.3 Joining DNA-based parts

Once the graph is simplified to a network of nodes for physical entities and their relationships, the 'start' nodes can easily be identified. For the subtilin receiver model, these are the 'pspaRK' and 'pspaS' nodes, which represent promoters, and are considered as roots. Starting from these roots, any node that does not represent a physical entity is omitted (Figure 6.11). Child nodes are connected to the parent of the removed node. This process is repeated until all nodes that do not represent physical entities are removed.

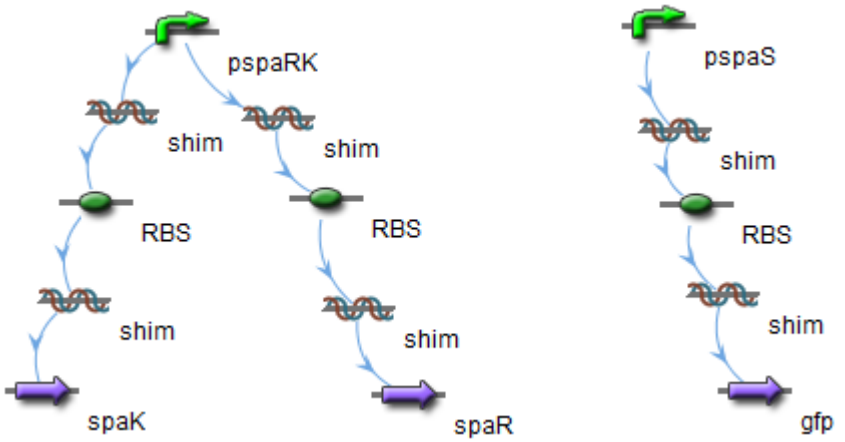


Figure 6.11: Starting from the root nodes, DNA-based entities are joined together.

6.5.2.4 Constructing the DNA sequences

The second subgraph, in which ‘pspaS’ is the root, is also a linked list of nodes and can therefore be used to directly derive the ordering of parts (Figure 6.11). However, the subgraph with ‘pspaRK’ as the root has two branches and represents an operon. These branches should be joined to form an operon structure (Figure 6.12). Both transcriptional units are then combined with terminator sequences.

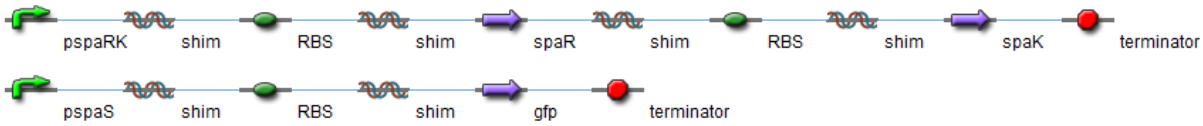


Figure 6.12: Transcriptional units are constructed, and terminator sequences are added.

The final DNA sequence is produced by concatenating the sequences of these transcriptional units (Figure 6.13). The sequence can then be exported in either GenBank or EMBL formats. In addition to automating the conversion of models into DNA sequences, MoSeC is also a lightweight visualisation tool that can be used to analyse the connectivity of modelling entities in CellML and SBML models.

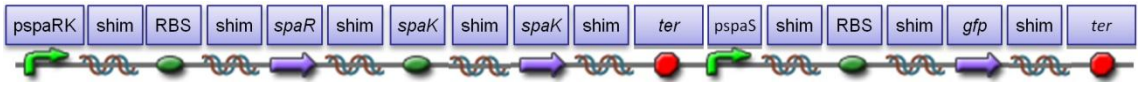


Figure 6.13: The sequences are concatenated in order to produce the final DNA sequence.

6.5.3 MoSeC as a model visualisation tool

MoSeC uses graphs to represent models in order to take advantage of the graph analysis techniques. Once a model is visualised, different layout options can be selected to achieve different views of the model (Figure 6.14). Single or multiple nodes can be

selected and moved inside the viewer. The toolbar and the menu items provide links to a variety of functionalities.

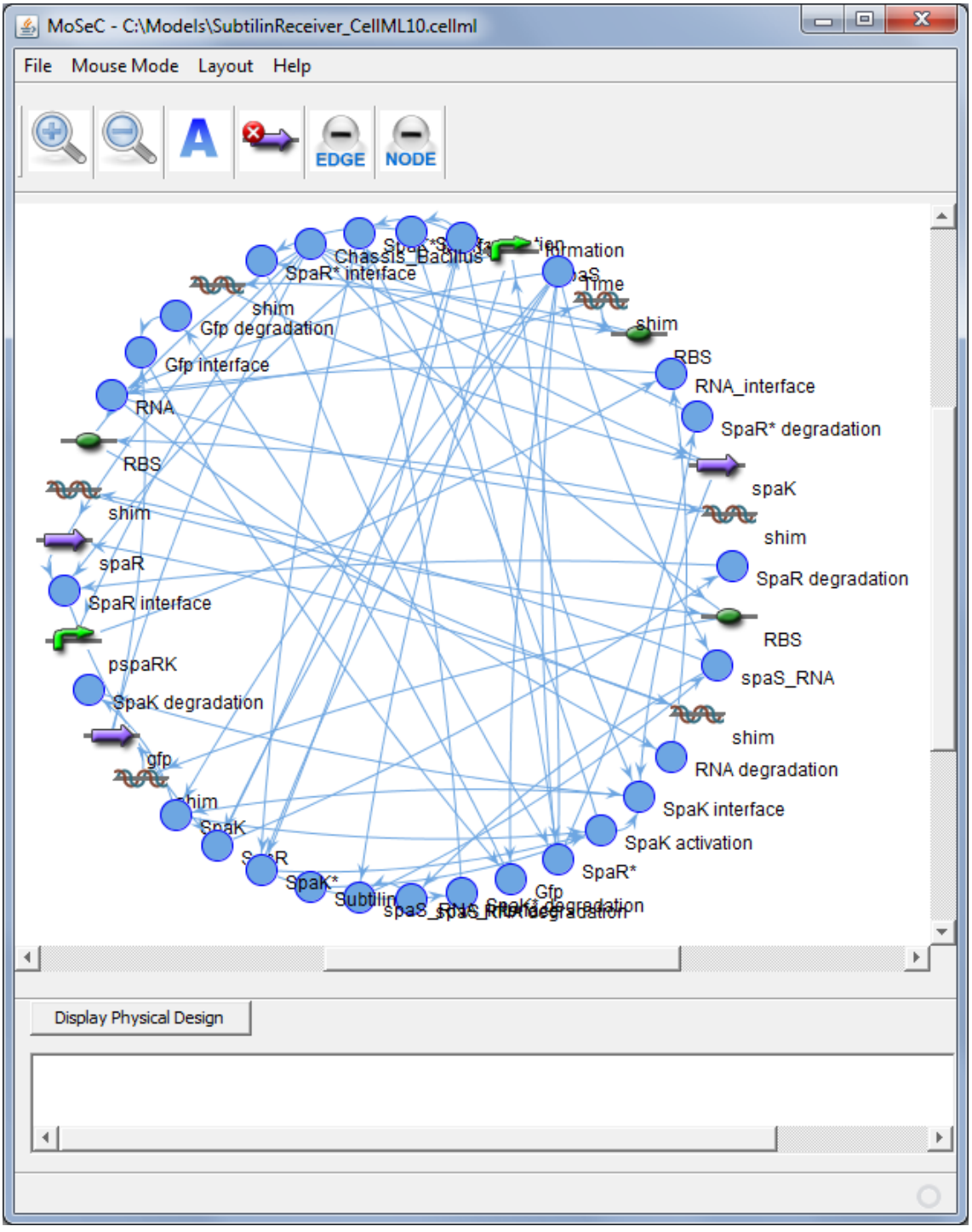


Figure 6.14: The CellML 1.1 version of the subtilin receiver visualised in MoSeC using the ‘circle’ layout. Other layout options are accessible from the ‘Layout’ menu item. The toolbar below the menu provides quick access to functions for customising the visibility of nodes, edges and edge labels, showing the corresponding icons for nodes representing physical items, and zooming in and out of the view. The ‘Display Physical Design’ button converts the model into a DNA sequence which can then be exported as an EMBL or GenBank file via the menu items.

Using the options provided, subgraphs of the analysed graph can be hidden. For example, nodes can be searched for by name and removed from the graph. This functionality is especially useful for CellML models, where components that provide constants, or act as time or volume providers have connections to a large number of components, making the manual analysis of models difficult. Therefore, such components can be hidden for simplicity. Similarly, edges can also be searched for by name in order to hide connected nodes. Labels of edges can also be hidden. MoSeC also provides the ability to pan and zoom the view of loaded graphs.

6.6 Discussion

Computational design and simulation has become increasingly important in synthetic biology as technology advances and the scale of projects consequently increases. Several CAD tools [32] have been developed to aid in the design of novel genetic circuits. However, the manual design of large biological circuits remains challenging [36]. The domain specific languages (DSLs) [21, 43, 117] developed provide abstract designs that can be implemented by swapping parts in and out in a fixed network topology. Possible solutions are then modelled and simulated to choose suitable designs. In addition, techniques such as evolutionary algorithms [14] can be used to design circuits with desired outcomes without the need to specify the order of components. Such approaches can result in novel systems that may not be possible to achieve via the manual specification of types and ordering of genetic components.

In order to experimentally test the designs, models that meet a particular target behaviour will ultimately be used to specify the DNA sequences of the genetic parts that encode the system. However, models are only mathematical representations of biological systems. In addition to the syntax provided by the modelling languages to represent modelling entities, the semantic annotation of these entities can allow computational tools to understand and process models. Therefore, for the automation of the model-to-sequence conversion, informative metadata is essential [36].

Although models provide mapping between DNA sequences and their functions [49], in a computational approach this mapping should be specified explicitly in the form of annotations. In this chapter, SVPs have been annotated with DNA sequence information. Therefore, models include both mathematical descriptions and the DNA sequences that represent them. Currently, terms are used to describe types of parts, and qualifiers represent the information necessary for the conversion.

These annotations are generic and can therefore be applied to model annotation in

synthetic biology, in which modelling can be used as a tool to construct biological systems. Models of biological parts can be included in several larger models in order to predict the behaviour of different systems. Therefore, annotations for the reuse and exchange of these models in synthetic biology would be valuable. Model annotation approaches from systems biology are also useful; however, the mapping of biological parts to modelling entities and specifying their types or sequences has not been the focus in systems biology models [319]. Therefore, the approaches presented here can be used as a basis for model annotation in synthetic biology, although annotations are used here to automate the conversion of models into DNA sequences.

In order to perform the conversion of models into DNA sequences, a novel algorithm has been developed. Although DNA sequences of genetic circuits can be derived by manually ordering the parts of a system, computational approaches have the ability to systematically try different types of parts and orderings in parallel. A large number of models for target behaviour can therefore be simulated and the most appropriate ones can be chosen computationally [30]. The algorithm presented here explicitly defines the conversion of SBML and CellML models into graphs in order to derive the ordering of parts at the DNA level. In this algorithm, the relationships of modelling entities are investigated using graph analysis approaches. The analysis of transcriptional and translational fluxes is used to derive the ordering of individual DNA sequences stored as annotations in SVPs. The availability of DNA sequences of parts and information about their ordering enables the specification of the DNA sequence represented by the entirety of a model to be produced in an automated fashion.

The algorithm described in this chapter was implemented in a tool called MoSeC, which can process CellML and SBML models constructed with SVPs and then produce DNA sequences in the form of GenBank or EMBL formats. This process is facilitated by the availability of machine-readable metadata in SVPs. The automation of the design of biological systems has already attracted interest [314, 320]. MoSeC and the annotation of models support this automation by generating DNA sequences from dynamic models, ready to order for synthesis. The catalogue presented in the previous chapter includes SVPs that are annotated to automate the construction of larger models computationally. These SVPs also include metadata required for model-to-sequence conversion. Therefore, the design of dynamic models and derivation of DNA sequences from the constructed models can be automated using computational approaches.

In addition, the use of standard formats enables the exchange and reuse of the designs of genetic circuits. DNA sequences of most published genetic circuits are not

available. In a letter to *Nature*, it was suggested that the DNA sequences of these circuits should also be published in the form of GenBank files [321]. MoSeC may be useful for the construction of such DNA sequences, especially for models generated by computational designs, and their subsequent publication and analysis in standard formats.

MoSeC is also useful for the visualisation of CellML and SBML models. Modelling entities and the relationships between them are displayed in the form of graphs. Entities that are mapped to physical parts are displayed with their corresponding icons, increasing the understanding of parts and their relationships. CellML 1.1 models can be formed from many individual models [64]. Even for small models, understanding the connections between the imported files and components can be tedious. Although a simple CellML viewer exists, the tool renders components and all of the encapsulated variables; even simple models are displayed with a large number of nodes and edges [322]. MoSeC only displays components and their connections, and hence can be used as a lightweight visualisation tool. In addition, different layout options allow the investigation of the connectivity of modelling entities.

The approach presented in this chapter is a step towards the automated design of biological systems. Deriving DNA sequences from models, especially from those that are produced computationally, is non-trivial, and therefore should be automated. This automation was presented using modular models of biological parts that can also be joined together computationally in order to construct models of large biological systems. Therefore, model-driven design approaches can be used to automate the searching of design spaces for possible solutions and deriving DNA sequences ready to encode complex biological systems for large-scale synthetic biology. These DNA sequences are produced in standard formats, which are also essential, to be used as input to other computational tools for the exchange of information in the automation of the design and synthesis of DNA sequences for biological systems. Moreover, the model annotation presented here can be generically applied in synthetic biology for the mapping of biological parts to their dynamic models.

Chapter 7. Identification of Orthogonal Parts for Engineering Regulatory Pathways in *B. subtilis*

7.1 Introduction

The type of bacterial hosts that can be used as the chassis for synthetic biology is currently limited to well-known model organisms [136]. As the field of synthetic biology matures, and more diverse applications emerge, there is a need to expand the range of available chassis. Furthermore, to construct predictable biological systems, the availability of orthogonal parts which minimise crosstalk between host systems and the introduced synthetic genetic circuits are desirable [58]. Although genome sequences are available for many organisms, little is known about the gene regulatory networks for most non-model organisms.

Synthetic genetic regulatory circuits can be used to programme the timing and level of gene expression. Components of the transcriptional machinery of cells, such as promoters, transcription factor (TF) binding sequences, and RNA polymerases (RNAPs), can be used to control the rate of transcription, and hence control the production rates of proteins via mRNA production [10]. Moreover, the use of these circuits with signal transduction systems allows cells to change their states based on external stimuli. Genetic regulatory circuits are increasingly being used in synthetic biology for a variety of uses. Genetic devices that were built with these circuits, such as logic gates [69, 70], oscillators [107, 323], bistable switches [71, 108] and biosensors [3, 68], can be used to engineer more complex biological systems.

One of the biggest challenges in synthetic biology is finding well-defined regulatory parts to use in engineering these circuits. As a result, a lack of interoperable parts is restricting the design of complex genetic circuits [38, 71]. Current designs use a limited number of TFs [38, 69-72]. For large and complex designs, a diverse range of TFs and their target binding sequences should be available.

A further challenge in the design of synthetic genetic circuits is to find orthogonal parts that do not introduce undesired behaviours. It would be ideal that modular parts which are designed, tested and characterised in isolation should also have the same

input-output behaviour when connected to a larger system [79]. However, possible interactions with existing circuitry may introduce crosstalk [51]. To minimise undesired interactions between host and synthetic networks, minimal genomes have been developed [27, 38, 86], and parts that would have the same behaviour under different conditions have been designed. For example, a library of promoters which show less variability, hence increasing the predictability of behaviour in different contexts, has been constructed [106]. Another approach is to use orthogonal parts that would work independently of the host systems.

The engineering of orthogonal parts has been applied to the control of expression of genes in synthetic gene regulatory networks. Rackham and Chin engineered ribosomes in order to produce orthogonal ribosome-mRNA pairs that function independently of the existing translational machinery of cells in *E.coli* [112]. These ribosomes only bind to mRNAs that have the corresponding binding sequences without affecting the translation of wild-type mRNAs, and orthogonal mRNAs are not processed by the wild-type ribosomes. A similar approach was applied to control the expression of genes transcriptionally. T7 RNAP, derived from bacteriophage T7, and its cognate promoters were used as orthogonal parts to construct genetic circuits in *E. coli* [153, 324]. Moreover, orthogonal TFs (OTFs) were developed by engineering variants of TFs that can bind to non-native DNA sequences or that can be activated by different chemicals [325]. These OTFs can be used to control the expression of multiple genes in parallel for the development of complex regulatory circuits in model organisms. As Rao pointed out, the number of TFs used in various genetic circuits even for the Gram-negative model bacterium *E.coli*, which has been probably the most commonly used bacteria for the design of synthetic genetic circuits, is roughly half a dozen [325]. For other organisms such as *B. subtilis* the situation is even worse, and usually promoters responsive to a few TFs, such as LacI, XlyR and AraR, are used even though *B. subtilis* is the Gram-positive model organism. The situation is even more complicated for organisms that are less well studied.

There is an increasing need to engineer in chassis that are not model organisms. Such organisms can have different features such as being able thrive in a wide range of conditions, at different PH levels and in different temperatures and pressures [326], allowing the development of specific biological applications. Although *B. subtilis* is a key industrial bacterium used extensively in synthetic biology [136], other *Bacillus* species can also be useful as host organisms in engineering new biological systems. *B. amyloliquefaciens* is known to promote plant growth [327], and *B. megaterium* is used

in industry, for example to produce vitamin B12 [328]. *B. licheniformis* is also an industrially-used organism in the production of antibiotics and enzymes [329]. There are also *Bacillus*-related species such as *Geobacillus* organisms, some of which were initially classified under the genus *Bacillus* [326]. These species can be thermophilic, and hence can be used in high temperature environments, for example to metabolise hydrocarbons in oil fields [330]. *Geobacillus* species can also be tolerant to high-ethanol levels, and can potentially be used in ethanol and biofuel production, with other potential industrial applications including the production of enzymes and bacteriocins and the degradation of herbicides [330]. In order to create novel regulatory networks, the transcriptional control machineries of these species must be well understood.

Although genomes of many non-model organisms have been sequenced, information about accurate regulatory networks of these organisms is not typically available. There are around 110 sequenced *Bacillus* strains in NCBI[43]. The NCBI's taxonomy browser[44] lists over 15,000 *Bacillus* strains. However, detailed information about TFs, their binding sequences and promoters is not well understood for non-model *Bacillus* species. It is not possible to carry out wet-lab based experiments for all of these species. Therefore, in addition to experimentally creating orthogonal parts, computational approaches are needed in order to identify orthogonal parts from closely related organisms.

Despite the lack of knowledge of non-model organisms, comparative genomic approaches can be used to infer the transcriptional regulatory networks of non-model *Bacillus* species. It is known that taxonomical distance correlates with the similarity of gene regulatory networks and, hence, the comparison of genomes in order to identify conserved genes has been used in the prediction of gene regulatory networks [331]. Although this information is useful to transfer the relationships between TFs and target genes to other organisms, finer details such the sequences of TF binding sites (TFBSs) should be available in order to facilitate the engineering of regulatory networks. Moreover, the use of TFBSs may increase the quality of prediction and may help in inferring regulatory networks that could not have been predicted using the previous approach [331, 332]. Searching for these sequences is mostly facilitated by position weight matrices (PWMs) [297], which can represent the TF binding motifs and can be used as input to the motif finding tools such as PoSSumSearch [333] and MAST [279]. Furthermore, applications such as RegNet [154], FITBAR [334] and RegPredict [335]

[43] ftp://ftp.ncbi.nih.gov/genbank/genomes/Bacteria/ (accessed 08/05/2012)
[44] http://www.ncbi.nlm.nih.gov/Taxonomy/Browser/wwwtax.cgi?id=1386 (accessed 08/05/2012)

provide platforms in order to perform TFBS predictions, by utilising existing methods and tools, and allow Web access to results.

RegNet reconstructs prokaryotic transcriptional regulatory networks on a genome-wide scale by combining TFBS searches with the orthology detections. In addition, it improves the construction of PWMs in predicting TFBSs by using reverse strands of binding sequences. Binding motifs are also optimised by readjusting the binding sequences [154]. In this process, the position of the predicted binding sequences can be modified using a sliding window. Moreover, predictions are given with corresponding p-values indicating their statistical significance. This tool can be used to infer gene regulatory networks in non-model *Bacillus* species.

7.1.1 The RegNet system

The RegNet [154] system uses model organisms as a source to predict the regulatory networks of their close relatives, and was initially produced for the *Corynebacteria*. The experimentally-verified gene regulatory networks of *Corynebacterium glutamicum* ATCC 13032 was used to predict TFs and their binding sequences in other *corynebacterium* species. This system was then extended to include *E.coli* [336] and the *mycobacteria* [183]. The results were stored in publicly-accessible CoryneRegNet [154, 183, 336-338] and MycoRegNet [339] databases.

For each species, RegNet integrates a number of different types of data, such as the nucleotide and amino acid sequences of coding sequences (CDSs) and proteins, operon structures and known gene regulation relationships (Figure 7.1). The system initially detects orthologous genes, and searches upstream of these genes to find binding sequences using the information from model organisms. The list of predicted interactions is then imported to a database back-end. All-versus-all BLAST is run to detect sequence similarities. The results are used as an input to detect protein homologies by using a protein clustering algorithm [183, 337]. If the binding sites are conserved and their regulated genes are homologous, the predicted interactions are more accurate [154]. Therefore, interactions that have both homologous TFs and target gene pairs are searched for in terms of binding sequences in the relevant species.

The binding sequences that are modelled with PWMs are searched for on a genome-wide scale in target organisms using the PoSSumSearch tool [333]. As a result, a list of binding sequences is produced for conserved TFs and the conserved target genes between the model organisms being used and their closely related species. During the transfer it is also assumed that the role of an interaction is also conserved, irrespective

of whether the regulation is positive or negative [339].

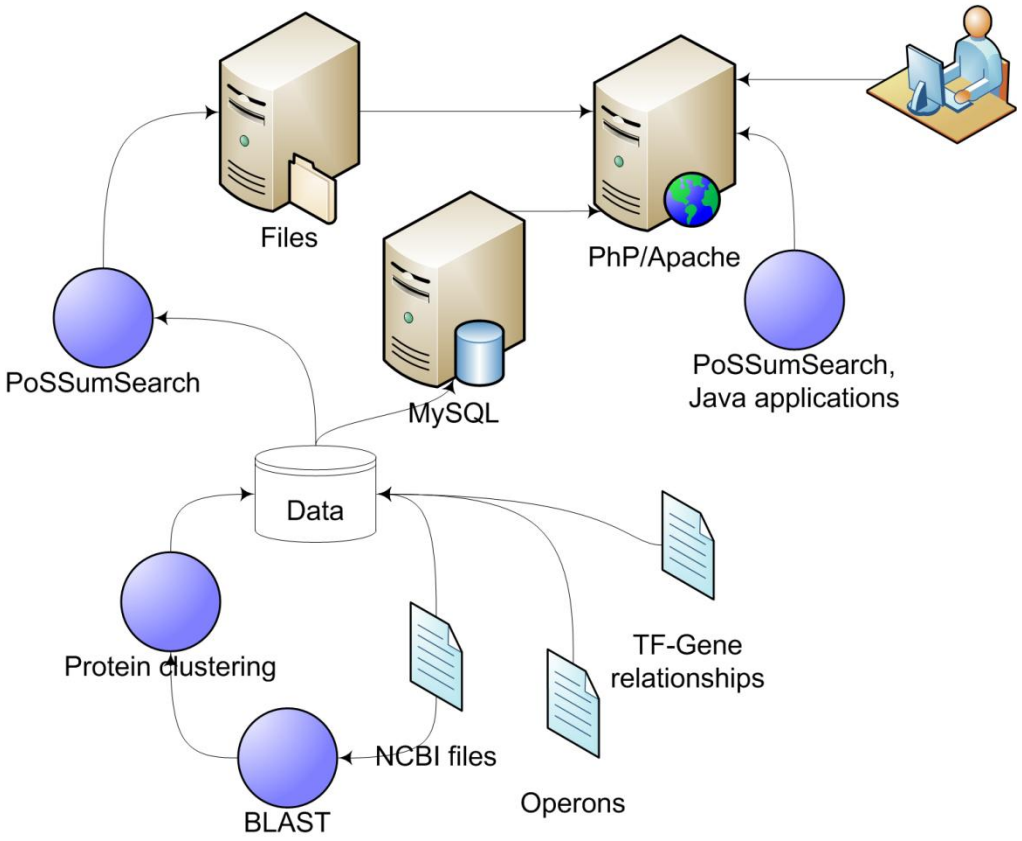


Figure 7.1: The components of the RegNet system. NCBI files, known gene regulation relationships, operons are integrated. Protein homologies are detected by BLAST and using a protein clustering algorithm. PoSSumSearch is used to search for binding sequences. The results can then be accessed via a Web interface.

The approach used in RegNet can be extended for *B. subtilis* in order to construct genome-wide gene regulatory networks for *Bacillus* species, and hence to increase the number of genomes available to synthetic biologists. *B. subtilis* is a model organism studied in a large number of publications. It has the advantage of large volumes of publicly available and computationally accessible data of appropriate levels of quality and accuracy, and wide coverage of its genes. In addition, there are strong similarities between *B. subtilis* and closely related species. Hence *B. subtilis* can be used as a template organism in RegNet in order to transfer experimentally-known information about its gene regulatory networks to its close relatives. Homology information for proteins is also available in RegNet. Therefore, the genome-wide comparison of TFs between model organisms and non-model organisms can be used to find TFs that do not exist in these non-model organisms. These TFs and their binding sequences can then be used as orthogonal parts in order to develop applications specific to non-model organisms.

The work described here demonstrates an approach for identifying regulatory parts and their corresponding binding sites in *B. subtilis* that can be used to construct regulatory genetic circuits in non-model organisms. OTFs and their target binding sites were extracted from *B. subtilis* regulatory networks using a comparative genomics approach based on prediction. A gene regulatory prediction tool has been developed based on the RegNet system to allow OTFs to be identified for a range of non-model organisms. The tool also provides useful approach for the computational transfer of regulatory networks from model to non-model organisms for systems biology studies. Whilst the research here is described for members of the genus *Bacillus*, the general approach is more widely applicable for identifying orthogonal regulatory parts for engineering regulatory systems in a range of bacterial hosts.

7.2 Inferring the gene regulatory networks for *Bacillus* species

In order to construct genome-wide gene regulatory networks of non-model *Bacillus* species, the RegNet [154] system was employed. The DBTBS database [166] provides TFs and their binding sites for *B. subtilis*. The latest annotations of these TFs are stored in BacilluScope [124]. The information from these databases was integrated in the BacillOndex dataset presented in Chapter 3. In this chapter, a subset of this dataset containing the details of the gene regulatory networks of *B. subtilis* was used to construct a data file in a format as required by the RegNet system.

Data about the gene regulation relationships was created as a tab-delimited file. The file contains a row for each TF and its regulated CDS. Data from BacillOndex is mapped to RegNet's format, as shown in Table 7.1. For example, TF family names and accessions representing the locus tags are used for the mapping. If a CDS concept in the BacillOndex dataset has positive or negative autoregulation recorded as an attribute, the TF's auto field is populated respectively. The CDS gene module is taken from the biological role classification attribute of the corresponding CDS concept. The characters 'A' and 'R' are used to represent the role of the TF binding as activator and repressor respectively. The PMIDs and binding sequences are recorded as semi-colon separated lists. For TFs that are sigma factors, the binding sequences represent the core promoter sequences where the RNAPs bind. These TFs are identified with the 'Is CDS Sigma factor' field. Since the RegNet system only deals with proteins, genes that encode RNAs were excluded.

Although the resulting file included the binding sequences for *B. subtilis*, these sequences needed to be processed prior to be used in RegNet. These sequences may

include additional upstream and downstream sequences that are not part of the actual binding sites. As a result, whole sequences may not be conserved. Although most of the binding sequences have annotations indicating the actual binding sequences, not all of them are annotated. However, PWMs, which are used to search for TFBSs in target organisms, require all the binding sequences for a particular TF to be the same length. Therefore, binding sites should be aligned and the additional bases from upstream and downstream of the binding sites should be excluded. This process was carried out by using the MEME tool [279].

Table 7.1: List of gene regulation fields required for the RegNet system and the mapping from the BacillOndex dataset.

Regnet field	Values mapped from BacillOndex
CDS	CDS accession
CDS gene name	CDS name
CDS gene module	Role classification
TF Family	TF family name such as ArsR, GntR and sigma factor
Auto	`Empty` : if not auto regulatory `-` : if negative auto regulatory `+` : if positive auto regulatory
Role	Role of the binding `Empty` : If not known `A` : If activator `R` : If repressor
Target gene	Target CDS accession
Target gene name	Target CDS name
Target gene module	Role classification of the target CDS
Motif known	`known` : If the binding motif is known `-` : If the binding motif is not known
Evidence	`Experimental` : To indicate that experimental information is known
PubmedIDs	Semi colon separated list of PMIDs
Binding motif	Semi colon separated list of TF binding sequences
Is CDS Sigma factor	`+` : If the CDS is encoding for a sigma factor

The parameters of the MEME tool were adjusted to find the most accurate binding motif for each TF. Each binding sequence of a TF was required to have the binding motif, which was also searched for in reverse and complementary sequences. The maximum length of a binding motif was set to the length of the binding sequence with the smallest sequence length, and the minimum length was set to 90% of the smallest sequence length for the TF. MEME was run for every TF having more than one binding sequence available. The resulting sequences that form the binding motifs were used as the new binding sequences in genome-wide searches (Figure 7.2).

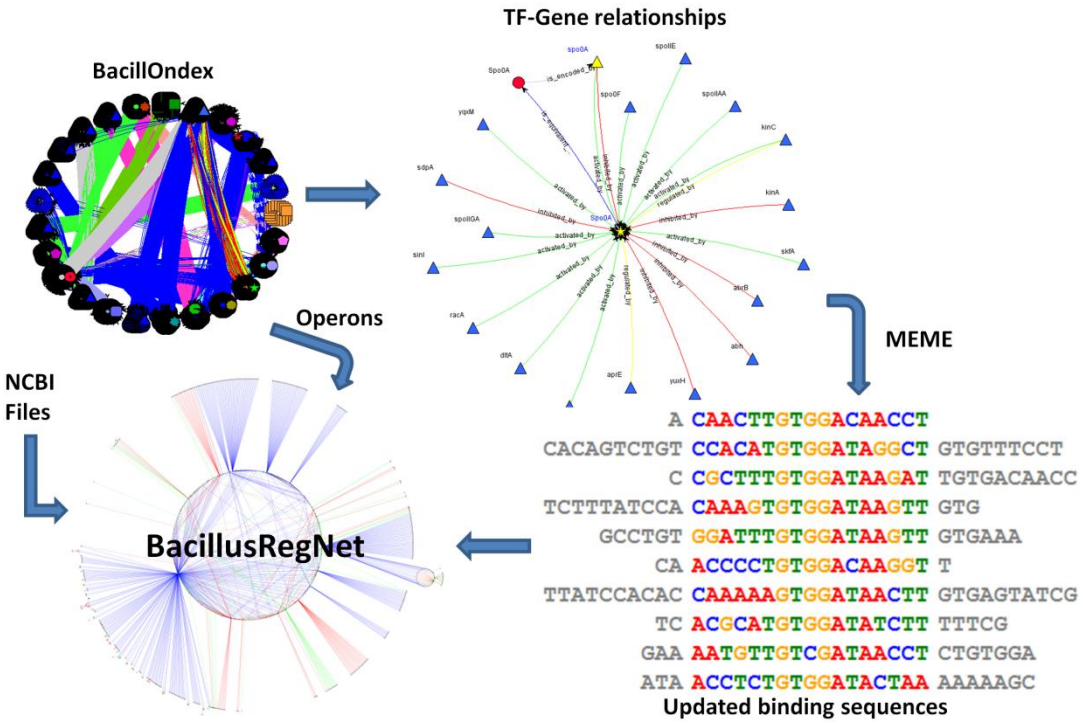


Figure 7.2: The binding sequences for each TF from the BacillOndex dataset were adjusted to have the same length using the MEME tool in order to create PWMs. These matrices are used to search for binding sequences in other organisms.

For example, in the system, eight binding sequences exist for the RocR TF, ranging from 21 bp to 29 bp in length (Figure 7.3). Therefore, the motif was searched for with 19 and 21 as the minimum and maximum length respectively. As a result, the motif was constructed using the 19 bp sequences. The sequence logo for the updated sequences is shown in Figure 7.4.

Figure 7.3: The search for a binding motif for the RocR TF. The coloured bases show the sequences that form the binding motif.

In addition to information about the gene regulatory networks of *B. subtilis*, additional input files for each organism were used. FASTA files for the nucleotide sequences and GenBank files for all organisms were downloaded from NCBI[45]. For

[45] ftp://ftp.ncbi.nih.gov/genomes/Bacteria

each non-model organism operon predictions were downloaded from Microbes Online [340] and converted into the RegNet format, using a utility provided in RegNet. For *B. subtilis*, this information was extracted from the BacillOndex dataset. These files were then used in RegNet in order to carry out predictions.

Figure 7.4: The sequence logo constructed for the binding motif of the RocR TF.

This section has described the RegNet system for *Bacillus* species using information about *B. subtilis* from the BacillOndex dataset, which was presented in Chapter 3. The system uses the experimental TF binding and promoter sequences for *B. subtilis* to predict similar sequences in close *Bacillus* species. These experimental sequences are modelled by PWMs in RegNet for genome-wide searches. Therefore, binding sequences and promoters with different lengths were adjusted using MEME to have the same length. Updated sequences were used to build the BacillusRegNet system which stores experimental and predicted gene regulation relationships for *Bacillus* species.

7.3 BacillusRegNet: A platform for the analysis and transfer of *Bacillus* gene regulatory networks

This section describes the development of BacillusRegNet[46] as a platform used to analyse the gene regulatory networks of *B. subtilis* 168 and its 15 relatives[47] including 13 *Bacillus* and two *Geobacillus* organisms. The list includes strains from *B. amyloliquefaciens*, *B. licheniformis*, *B. pumilus*, *B. megaterium*, *B. halodurans*, *B. anthracis*¸ *B. clausii*, *B. thuringiensis*, *B. cytotoxicus*, *B. weihenstephanensis*, *B. cereus*, *B. tusciae*, *G. kaustophilus* and *G. thermodenitrificans* organisms. Some of these organisms include plasmids which are also available in BacillusRegNet. The system is able to construct genome-wide gene regulatory networks for these non-model organisms using the well-understood TFs, and their target genes and binding sequences from *B.*

[46] http://bacillus.ncl.ac.uk

[47] The system was initially built for *B. subtilis*, *B. amyloliquefaciens*, *G. kaustophilus*, and two *B. anthracis* strains. The data files prepared for *B. subtilis* 168 were later used to build the latest version of BacillusRegNet with the latest software and additional 11 organisms in collaboration with the developers of CoryneRegNet.

subtilis 168 (Figure 7.5). The system includes predictions for nucleotide sequences of TF binding sites and promoters in target organisms, and homology information about CDSs and proteins.

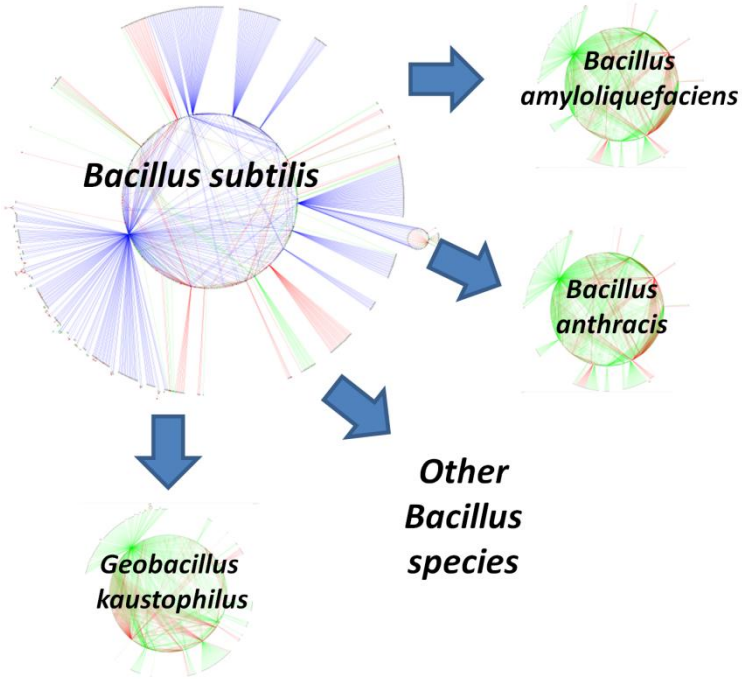


Figure 7.5: BacillusRegnet is able to construct putative genome-wide gene regulatory networks for closely related *Bacillus* species using the data from *B. subtilis* 168.

7.3.1 Genome-wide construction of gene regulatory networks

BacillusRegNet provides two databases, with each having a separate Web user interface: Experimental and Predicted. Figure 7.6 shows the summary from these two databases. The database of experimental data contains information only about known regulation relationships and information derived from GenBank files and nucleotide sequences. The experimental data include 69,388 genes and proteins for *Bacillus* species, and information about the gene regulatory networks of *B. subtilis* 168 for 140 TFs, 1,148 binding sequences and 1,250 regulatory relationships. The database for the predicted data stores predictions based on the experimental gene regulation relationships, in addition to containing all the data from the database of experimental data. The system was able to predict additional 696 TFs, 7,856 binding sequences and 14,540 regulatory relationships for 15 non-model organisms. For all organisms in the system, the number of activations and repressions classified are 11,239 and 4,551 respectively.

Experimental

Element	Number
Genes	69388
Proteins	69388
Regulations	1250
Regulators	140
Regulated genes	787
Binding motifs	1148
Position weight matrices	91
Protein clusters	7081
Genomes	26

Predicted

Element	Number
Genes	69388
Proteins	69388
Regulations	15790
Regulators	836
Regulated genes	9349
Binding motifs	9004
Position weight matrices	784
Protein clusters	7081
Genomes	26

Figure 7.6: Summary of the databases containing experimental and predicted data. 'Experimental' data include information about the gene regulatory networks of *B. subtilis* 168 only. The 'predicted' database includes everything from the experimental data and the predictions for 15 other non-model organisms in the system.

The predictions for each of these 15 non-model organisms are also available from BacillusRegNet. Table 7.2 presents the summary of these predictions. Each row contains information such as the number of predicted regulators, binding sequences and regulated genes, together with the number of proteins stored in the system. The list of genes, detailed information about genes and proteins, and operon structures for each organism are accessible from the BacillusRegNet website.

Table 7.2: Number of predicted regulators, binding sequences, regulation relationships and regulated genes are listed for each organism. Genome sizes for the organisms were taken from the NCBI's GenBank database.

	Proteins	**Regulation relationships**	**Regulators**	**Binding sequences**	**Regulated genes**	**Genome size (Mb)**
B. subtilis 168	4,175	1,250	140	1,148	787	4,215
B. amyloliquefaciens (FZB42)	3,693	1,611	75	875	870	3,918
B. halodurans (C-125)	4,065	845	43	443	498	4,202
B. anthracis (A0248)	5,040	862	43	466	528	5,227
B. anthracis (Sterne)	5,289	945	45	559	494	5,228
B. licheniformis (ATCC 14580)	4,173	989	72	989	567	4,222
B. clausii (KSM-K16)	4,096	983	43	455	566	4,303
B. thuringiensis (Al Hakam)	4,736	1,043	47	499	623	5,257
B. cytotoxicus (NVH 391-98)	3,833	821	36	359	496	4,087
B. pumilus (SAFR-032)	3,679	1,377	61	668	741	3,704
B. weihenstephanensis (KBAB4)	5,155	1,056	46	502	626	5,262
B. cereus (B4264)	5,398	901	43	517	529	5,419
B. tusciae (DSM 2912)	3,150	284	14	121	249	3,384
B. megaterium (DSM 319)	5,100	1,287	52	768	732	5,097
G. kaustophilus (HTA426)	3,497	787	37	378	492	3,544
G. thermodenitrificans (NG80-2)	3,392	748	38	326	480	3,550

7.3.2 Analysis of the gene regulatory networks using BacillusRegNet

The system provides both text- and graph-based data visualisation using html views and GraphVis respectively. The data can be searched using gene and protein identifiers, and the results are displayed in various categories. Details for each gene include information about its protein product, a list of TFs that regulate the gene and if the gene is encoding for a TF, then genes that are regulated by it. Other details include homologous genes and proteins, gene attributes, a PWM and the sequence logo used to depict the binding motif. GraphVis can be used to visualise both the experimental and predicted gene regulation relationships.

Figure 7.7 shows an example of predicted gene regulation relationships using BacillusRegNet. In the figure, the transcriptional network of the *spo0A* gene is visualised with GraphVis for *B. subtilis* 168 and the target organism *B. licheniformis* (ATCC 14580). The inhibition of Spo0A on *kinA* and *yuxH* genes and activation on *spoIIE*, *spoIIAA*, *spoIIGA*, *dltA* and *kinC* can be found in the target organism. However, the inhibition for *sdpA* and activation for *skfA* could not be found, since there are no homologues of these genes and their encoded proteins. Although there are homologues of *era* and *yqxM* in the target organism, the positive regulation of these genes by Spo0A

could not be identified. The system was able to predict the *sigA* and *sigH* promoters of the *spo0A* gene. Additionally, activation relationships for *lchAA*, *lchAB*, *lchAC* and *dhbF* were predicted. The sequence logo constructed from predictions for *B. licheniformis* (ATCC 14580) is similar to that of *B. subtilis* 168.

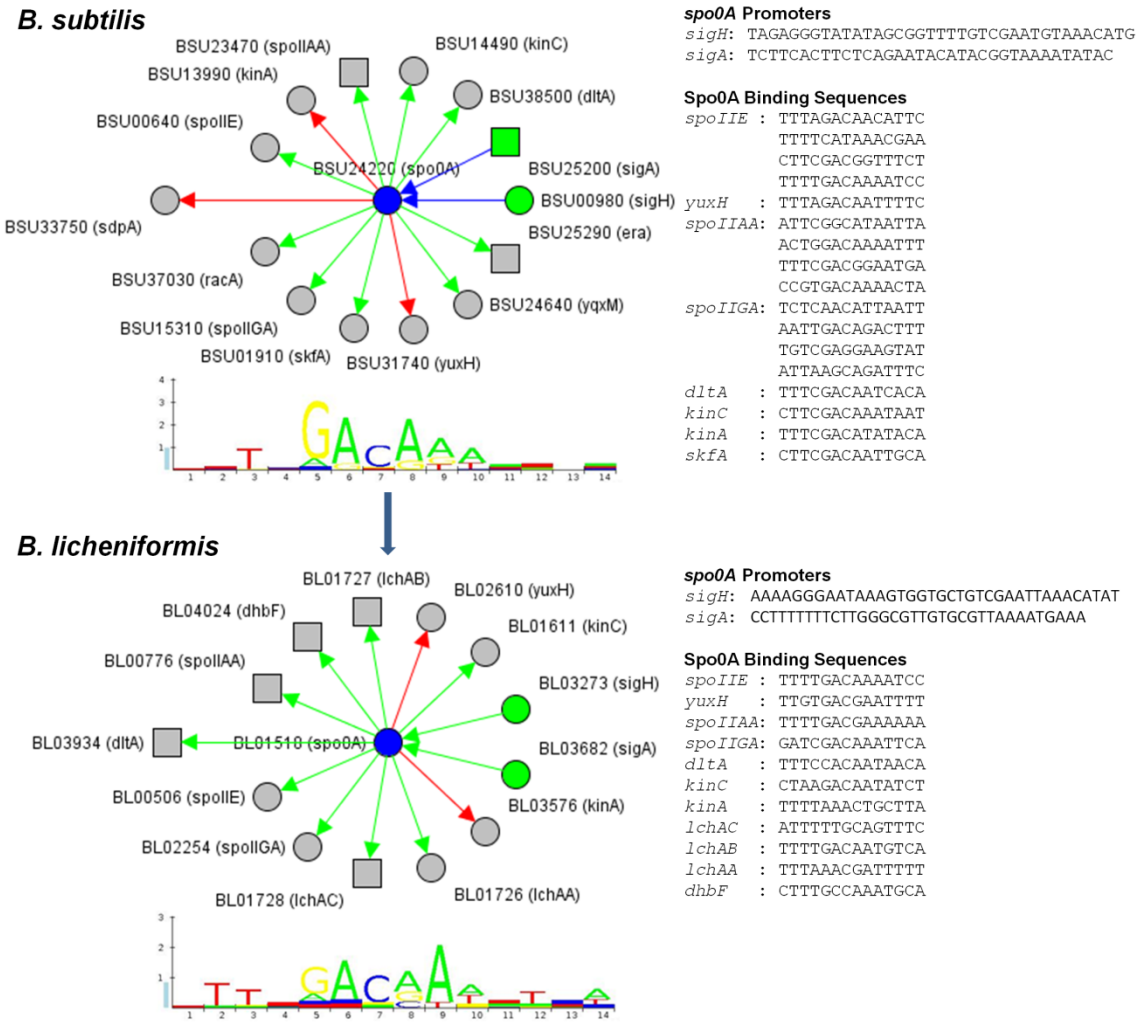


Figure 7.7: The transcriptional network of the *spo0A* gene for *B. subtilis* 168 and *B. licheniformis* (ATCC 14580). Red and green lines represent the inhibition and activation interactions, respectively. Similarly, red and green shapes represent the activators and inhibitors. Blue lines for the model organism show the sigma factor regulation relationships. Squares and circles are used to represent whether or not the genes are part of operons and preceeded by TF binding sequences respectively. Sequence logos that depict the binding motifs are shown below each network. The sequences on the right-hand side are the known and predicted promoter and TF binding sequences for *B. subtilis* 168 and *B. licheniformis* (ATCC 14580) in BacillusRegNet respectively.

TF binding sequence predictions can also be conducted manually. This method allows the choice of user-defined cut-off values for the predictions of gene regulation relationships. Binding sequences can be searched for in two ways. In the first option,

upstream sequences of all genes for a target organism are searched to find the genes that are regulated by the TF encoded by the queried gene. In the second option, TFs that regulate the queried gene are listed. In both cases, upstream sequences of the genes are searched for. After running a transcription factor binding site (TFBS) search, both the source genes and the target genes can be laid out in the GraphVis view with the relationships added for homologue proteins between the target and the source organisms. Additionally, TFBS predictions can also be achieved by submitting the binding sequences to be searched for as FASTA files through the TFBSScan section of the website.

B. subtilis 168 core promoters that include the binding sites for RNAPs are uploaded to BacillusRegNet as binding sequences of the relevant sigma factors. Therefore, the system also predicts promoters in closely related *Bacillus* species. Figure 7.8 shows the SigA sequence logos for *B. subtilis* 168 and *B. amyloliquefaciens* (FZB42) aligned with the consensus sequence TTGACA-N_{14}-tgnTATAAT [341]. Compared to 317 experimentally known *sigA* promoters in *B. subtilis* 168, 574 *sigA* promoters were predicted for *B. amyloliquefaciens* (FZB42). As can be seen, the predicted sequence logo is also similar to the consensus sequence. The *sigA* gene is known as *rpoD* in *B. amyloliquefaciens* (FZB42). This information is available in the list of *B. subtilis* 168 *sigA* homologues.

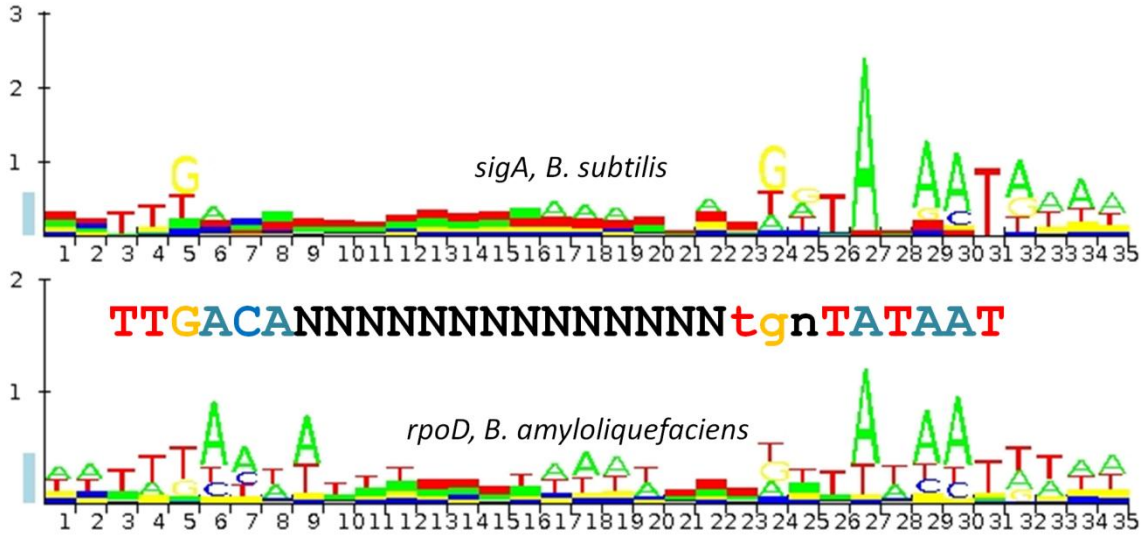


Figure 7.8: The sequence logos of *sigA* promoters for *B. subtilis* 168 and *B. amyloliquefaciens* (FZB42). The logo at the top is drawn using 317 core promoter sequences that includes the binding sequences for RNAPs from *B. subtilis* 168. The logo at the bottom was constructed from 574 predicted *B. amyloliquefaciens* (FZB42) core promoters. The middle sequence shows the consensus *sigA* promoter sequence, aligned with two sequence logos.

Genome-wide statistics can also be accessed from the website. The statistics provided include the distribution of binding sites from the genes' start locations, quantities of regulators and regulation types, and the distribution of the number of co-regulating TFs. ATGC content for the genome, and coding and non-coding regions are also available for each organism in the system. Figure 7.9 shows the distribution of TFBS distances from the start of genes for *G. kaustophilus* (HTA426). The number of repressors and activators predicted for G. *kaustophilus* (HTA426) is 17 and 15. Additional five TFs have dual roles. However, compared to 552 activation relationships, only 235 repressions were predicted. These activators and repressor sites tend to be between 0 and +100 relative to the gene start locations.

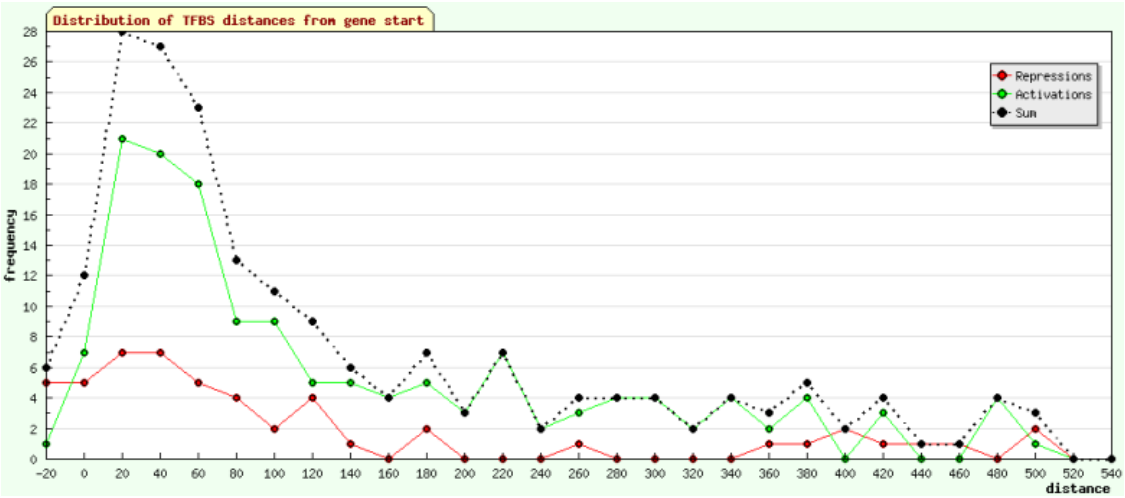


Figure 7.9: An example of the distribution of TF binding site distances from gene start for *G. kaustophilus* (HTA426). Red and green lines show the distribution of repressions and activations respectively. The distribution for all regulations is represented with the dashed black lines.

BacillusRegNet was built as two different Web applications, with database back-ends. The first application is used only to access to experimental data, while the second application can be used to gain access to both experimental and predicted data about gene regulatory networks. Currently, the system contains 16 species including *B. subtilis* 168 as the model organism. Genome-wide predictions for 15 non-model organisms can be accessed from the website. The system can also be used to identify proteins that exist in some species but not in target organisms in order to find OTFs for these targets.

7.4 A comparative genomics approach for the identification of orthogonal TFs

Homology information of TFs available in BacillusRegNet can be used to identify orthogonal parts. The system currently does not offer a Web interface to query the OTFs directly. Therefore, putative OTFs were extracted by querying the database directly. TFs from *B. subtilis* 168 that do not have homologous proteins in the other species in the database were identified as orthogonal parts for that species. Only the OTFs with known binding sequences were retrieved.

In total, 39 TFs from *B. subtilis* 168 were identified as being orthogonal for use in one or more of 13 *Bacillus* and two *Geobacillus* species. Table 7.3 shows a list of these TFs and organisms in which they can be used as orthogonal parts. The first two columns represent the CDSs that encode for the identified TFs and the first row includes the list of organisms. TFs that are orthogonal in any of these organisms are marked with ticks in the corresponding columns and rows. The number of orthogonal parts that can be used for each species ranges from three for *B. amyloliquefaciens* (FZB42) to 24 for *B. tusciae* (DSM 2912).

Table 7.3: List of 39 OTFs identified for 13 *Bacillus* and two *Geobacillus* organisms. Only the TFs with known binding sequences are included. The CDSs on the left-hand side, given with the locus tags and labels, are all from *B. subtilis* 168 and encode for TFs that can be used as orthogonal parts in closely related organisms. A CDS with a tick sign can be used as an orthogonal part for the organism in the corresponding column.

	B. subtilis 168 TFs	*B. amyloliquefaciens* (FZB42)	*B. halodurans* (C-125)	*B. anthracis* (A0248)	*B. anthracis* (Sterne)	*B. licheniformis* (ATCC 14580)	*B. clausii* (KSM-K16)	*B. thuringiensis* (Al Hakam)	*B. cytotoxicus* (NVH 391-98)	*B. pumilus* (SAFR-032)	*B. weihenstephanensis* (KBAB4)	*B. cereus* (B4264)	*B. tusciae* (DSM 2912)	*B. megaterium* (DSM 319)	*G. kaustophilus* (HTA426)	*G. thermodenitrificans* (NG80-2)
BSU01810	*adaA*												√			
BSU33970	*araR*												√			
BSU25810	*arsR*	√				√		√		√				√		
BSU26580	*bltR*						√						√		√	√
BSU24020	*bmrR*	√					√						√		√	√
BSU29740	*ccpA*												√			
BSU07590	*citT*												√			
BSU10420	*comK*		√				√						√	√		
BSU39430	*deoR*												√			
BSU37310	*fnr*						√			√						
BSU13880	*glcT*												√			
BSU17450	*glnR*		√													
BSU40050	*gntR*	√	√						√	√	√	√			√	√
BSU06140	*gutR*		√	√	√	√	√	√	√	√	√	√	√		√	√
BSU39340	*hutP*					√	√			√			√			
BSU03470	*hxlR*														√	
BSU39770	*iolR*												√			
BSU39080	*licT*												√			
BSU02680	*lmrA*		√	√	√		√	√	√		√	√	√		√	√
BSU04250	*lrpC*												√		√	
BSU36600	*mta*									√			√			
BSU22770	*mtrB*			√	√			√	√		√	√				
BSU32420	*pucR*			√	√			√	√		√	√			√	
BSU38070	*sacT*												√			
BSU38420	*sacY*												√			
BSU24610	*sinR*						√						√		√	√
BSU33790	*sdpR*			√	√			√	√	√						
BSU13450	*sigI*		√				√		√							
BSU09520	*sigM*								√							
BSU01730	*sigW*								√							
BSU23100	*sigX*								√				√			
BSU38700	*sigY*								√							
BSU13310	*tnrA*			√	√			√	√			√	√			
BSU07820	*treR*												√			
BSU02440	*ycbA*								√				√			√
BSU13150	*ykmA*		√				√									√
BSU14730	*ylaC*								√							
BSU19540	*yodB*														√	
BSU33660	*yvaN*			√	√		√	√	√		√	√	√	√	√	

Some of these organisms also include plasmids (Table 7.4). Although TFs from *B. subtilis* 168 were compared against TFs from the strains of non-model organisms using homology information, this comparison should also be made against TFs from the plasmids of these strains. TFs from *B. subtilis* 168 with homologues from a plasmid should not be used as OTFs for the strain that includes the plasmid. Therefore, the plasmids were also included for the identification of OTFs. The results confirmed that the previously identified OTFs do not have any homologous TFs in plasmids and can therefore be used reliably.

Table 7.4: List of strains with plasmids.

Strain	Plasmids
B. anthracis (A0248)	pXO1 (NC_012656), pXO2 (NC_012655)
B. cytotoxicus (NVH 391-98)	pBC9801 (NC_009673)
B. thuringiensis (Al Hakam)	pALH1 (NC_008598)
B. weihenstephanensis (KBAB4)	pBWB401 (NC_010180), pBWB402 (NC_010181), pBWB403 (NC_010182), pBWB404 (NC_010183)
G. kaustophilus (HTA426)	pHTA426 (NC_006509)
G. thermodenitrificans (NG80-2)	pLW1071 (NC_009329

BacillusRegnet was able to predict OTFs using a comparative genomics approach. In this approach, 39 TFs that exist in *B. subtilis* 168 but not in other *Bacillus* organisms were identified as OTFs. Although there are more OTFs, not all of them have binding sequences available. With improvements in the quality and coverage of data about TFs for *B. subtilis* 168 and other non-model organisms, this number can be increased.

7.5 Discussion

Currently, the number of hosts that can be used for synthetic biology is limited to well-known model organisms such as *B. subtilis* and *E. coli*. However, gene regulatory predictions from the model organism *B. subtilis* can be used to inform experimental designs in non-model *Bacillus* species [342], and to extract orthogonal parts for the design of synthetic genetic circuits. The BacillusRegNet system described in this chapter extends the list of organisms available to synthetic biologists interested in working with other organisms related to *B. subtilis* 168. Furthermore, the system developed promises to increase our understanding of closely related *Bacillus* species by providing putative gene regulatory networks for a range of these organisms.

Genome sequences not only provide information that is fundamental to biology, but can also be viewed as providing coding sequences that can act as a resource for parts

provision for synthetic biology [20]. The approaches described in this chapter show how whole genome sequences provide a valuable resource for the extraction of orthogonal genetic part definitions from closely related species. BacillusRegNet allows the prediction of not only the TFs and their regulated genes, but also the target binding sequences which are required for the engineering of transcriptional control. In addition homology information about TFs is also available in the database. Therefore, TFs that do not have any homologues in other organisms and their target binding sequences can provide a catalogue of orthogonal parts. The extraction of orthogonal parts was demonstrated for non-model *Bacillus* species using the data from *B. subtilis* 168. Although currently there are only 16 species in the BacillusRegNet database, the list can be increased by using the scripts built into the RegNet system. This approach can also be extended to minimal genomes being developed for *B. subtilis* [134]. The system currently includes 836 TFs, 696 of which were predicted in 15 non-model organisms. The number of TFs predicted for an organism varies due to several factors.

Firstly, the size of the total regulatory network varies between organisms. For example, the growth and survival conditions of organisms may affect the complexity of their regulatory networks and hence the number of TFs used for their gene regulation [154]. The number of TFs follows a power law increase with the number of genes increasing in genomes [342]. *B. licheniformis* (ATCC 14580), *B. megaterium* (DSM 319), *B. anthracis* (Sterne), and *B. cereus* (B4264) have more gene regulation relationships relative to some other organisms. This increase may be due to the increase in the number of genes and proteins in these organisms. Additional TFs may be required to regulate these additional genes. On the other hand, *B. tusciae* (DSM 2912) has a genome of the smallest size and thus has the lowest number of predicted TFs (at only 14).

The RegNet system works by predicting TFs in the genome to be analysed that are orthologous to those from *B. subtilis* 168. Organisms that are closer in taxonomic terms will therefore contain more orthologous TFs and therefore more will be predicted. For example, although the genome sizes of *B. amyloliquefaciens* (FZB42) and *B. licheniformis* (ATCC 14580) are not the biggest, these organisms have the most predicted TFs (75 and 72 respectively). This result was expected, since these organisms are classified in the *Bacillus subtilis* group in a taxonomic analysis performed based on the NCBI taxonomy (Figure E-1). The number of gene regulation relationships predicted for these organisms is also higher than for *B. subtilis* 168. This increase could be due to not having a full coverage of the gene regulatory networks for *B. subtilis* 168,

or if there are new regulation relationships that do not exist in *B. subtilis* 168. TFBSs and promoters for *B. subtilis* 168 used in BacillusRegNet do not represent the entire set of the gene regulatory networks of the organism. These sequences were derived from DBTBS. However, a recent study identified 2935 promoters [343], around four times the number of promoters used in BacillusRegNet. Therefore, the number of promoters predicted for non-model organisms can be more than those for *B. subtilis* 168 in BacillusRegNet. For example, compared to 317 SigA promoters available for *B. subtilis* 168, 574 SigA promoters were predicted for *B. amyloliquefaciens* (FZB42). In addition, *B. tusciae* (DSM 2912) is the most distant organism to *B. subtilis* 168 in the taxonomy and also has the lowest number of predicted TFs.

Although the two *Geobacillus* species are not classified directly under the *Bacillus* taxonomy, 37 and 38 TFs respectively could be predicted. Interestingly some *Geobacillus* species used to be classified under the genus *Bacillus*. For example, *G. stearothermophilus* was known as *Bacillus stearothermophilus* until 2001 [344]. G*eobacillus* species have many proteins homologous to those in *B. subtilis* despite relatively small genome sizes. In addition 12 and 8 OTFs could be identified for *G. kaustophilus* (HTA426), and *G. thermodenitrificans* (NG80-2) strains respectively.

Conversely, for organisms that are taxonomically distant, a smaller number of OTFs was predicted. As expected, for *B. tusciae* (DSM 2912) the most putative orthogonal factors (24 TFs) were identified, due to not being a very close relative of *B. subtilis* 168. On the other hand, organisms closer to *B. subtilis* 168 in the taxonomy (such as *B. amyloliquefaciens*) have the fewest number of OTFs. Although *B. cytotoxicus* (NVH 391-98) is a *B. cereus* group organism, the number of OTFs predicted was 15, which is higher compared to other *Bacillus cereus* species. This result is due to *B. cytotoxicus* (NVH 391-98) being the most distant organism taxonomically within the *B. cereus* group of organisms [345].

Some of the OTFs identified are sigma factors. These sigma factors such as SigI and SigX can be used to develop orthogonal transcriptional machineries using responsive promoters in the relevant species. Although the OTFs identified can be used in cascades of transcriptional control networks, some of them can be used to convert environmental signals directly into cellular responses. For example, promoters repressed by AraR can be activated by arabinose [165]. ArsR can repress transcription in the absence of metals such as arsenic, cadmium and copper [346]. In addition, CitT is a response regulator that can be used to engineer signal transduction systems [122].

GutR, YvaN, LmrA and GntR are TFs that were identified as orthogonal parts for

most of the species used in BacillusRegNet. GutR and SenS TFs had been previously reported as TFs exclusive to *B. subtilis* [132]. The work presented here confirms these findings. GutR has a binding sequence available in BacillusRegNet and was therefore included in the list of OTFs (Table 7.3). Although BacillusRegNet shows SenS as not having any homologues to TFs from the 15 non-model organisms, this protein was not included in the list of OTFs due to the unavailability of its target binding sequences in BacillusRegNet.

Although LmrA exists in the *Bacillus* group, and in frequently found in the *B. subtilis* group, the *B. cereus* group of bacteria do not include orthologues of this TF in BacillusRegNet. LmrA can also be used as an orthogonal part in *Geobacillus* bacteria. The BLAST results list only 24% maximum identity for *Geobacillus* species using this TF.

YvaN also do not have homologous proteins in the *B. cereus* group of bacteria. However, YvaN has an orthologous protein in *G. thermodenitrificans* (NG80-2). GntR has homologous proteins in species both close and distant to *B. subtilis* 168, such as *B. licheniformis* (ATCC 14580) and *B. tusciae* (DSM 2912) respectively. Although GntR could be used as an OTF in eight species, there is no recognition pattern identified for the use of this TF. However, BLAST results for *Geobacillus* species list identical proteins at less than 29%. Therefore, GntR can be used in *Geobacillus* species as an orthogonal part.

High-throughput experiments can also be used to reconstruct gene regulatory networks [347]. However, the generation of wet lab based high-throughput data is expensive and is difficult to carry out for every new species [154]. Using the RegNet system, regulatory networks of *Bacillus* species were successfully constructed in a time- and cost-effective manner.

Using the BacillusRegnet system described in this chapter, the gene regulatory networks of 15 *Bacillus* species were constructed on a genome-wide scale using *B. subtilis* 168 as the model organism. 696 TFs and 7,856 target binding sequences (including promoters) were predicted for these non-model organisms. Out of 140 *B. subtilis* 168 TFs in the system, 39 TFs that do not exist in other organisms in BacillusRegNet were then identified as putative orthogonal parts for these non-model but closely related organisms. The target binding sequences of these TFs are also available from the system. These identified TFs and their binding sequences are orthogonal parts that allow the construction of reliable gene regulatory networks with no undesired interactions in the host cells specified. Currently, however, these results

are not computationally available. Therefore, a future work is to build an interface in order to retrieve the list of OTFs within BacillusRegNet. Moreover, in the future, the list of orthogonal parts will be integrated to the BacillOndex dataset presented in Chapter 3.

Chapter 8. Discussion and Future Work

8.1 Introduction

This thesis presents data integration approaches that facilitate the automation of the design of large-scale biological systems. The tools and techniques described here facilitate the computational exploration of large design spaces for such systems using currently available biological data. The engineering of biological systems is a complex process, and requires the automation of the design of large-scale biological systems [14]. For computational approaches, designs must be explicitly specified using existing biological knowledge, and should be analysed via simulations. Models used in these simulations can then be converted into DNA sequences in order to test these systems *in vivo* or *in vitro*. In this thesis, these issues were addressed by: using ontologies to represent biological knowledge, including information about biological parts and their relationships; specifying a mapping between biological parts and mathematical models in order to construct models of large systems computationally; developing a repository of modular models that are publicly available and computationally accessible; and automating the conversion of models of large biological systems that are produced computationally into DNA sequences.

Currently, most synthetic biology applications are still small and designed manually [37] following a life-cycle of specification, design, modelling, fabrication, and experimental tests [22, 48]. Potential novel applications require larger, more complex designs. However, the manual design of biological systems is challenging due to the increasing size and complexity of these systems, and the vast amount of data about them available [348]. Therefore, the design of biological systems benefits from being automated as much as possible [14, 20].

Predicting the behaviour of even a simple biological device is not a trivial matter [11]. Therefore, computational design and simulation should be used to construct predictable and robust parts [20, 21] and to combine these parts into systems with predictable behaviours. Tools for the computer-aided [11, 33, 40, 44, 47] and automated [41-43] design of biological systems have already been developed by different research groups. These tools need access to models of biological parts in order to design

predictable systems. Moreover, in order to reuse and exchange models between tools, standard modelling frameworks are essential [11, 30, 41]. The use of standards can also facilitate the construction of computational workflows for the execution of different tools and the retrieval of data in order to automate the design of complex biological systems.

Due to the size and complexity of biological systems, the possible number of solutions to any design problem, particularly for large-scale designs, grow exponentially [21, 30, 73, 74]. One way to constrain this design space is by using existing biological information [75]. To facilitate the computational design of complex biological systems, as much information about parts and their interactions as possible should be available to computational tools. Although extensive amounts of information exist regarding regulatory networks, genes, proteins, pathways, metabolites and molecular interactions [349], data are spread across a myriad of heterogeneous databases [34]. For computationally automated design, this knowledge must be integrated and presented to computational tools in suitable formats.

8.2 Data integration for the computational design of biological circuits

In this project data integration was employed to create specifications for biological parts and their constraints, in order to aid the computational design of biological systems for *Bacillus subtilis* (Chapter 3). To facilitate machine access, this information was represented using existing standards such as ontologies (Chapter 4) and computational modelling languages (Chapter 5).

In order to investigate the behaviour of biological circuits constructed from individual parts, a mapping between physical parts and their *in silico* representations was developed (Chapter 5) [80]. Using this mapping, a computationally-accessible integrated dataset was used to create dynamic models of a range of parts (Chapter 5).

Computational parts models can be joined together, to form models of complete biological circuits. These models can also be marked up with DNA sequence information, which enables the automated conversion of models built with modular models of parts into DNA sequences (Figure 8.1) (Chapter 6).

In this project, these techniques were applied to *B. subtilis*, but they are readily applicable to other species. To demonstrate the value of cross-species analysis, the gene regulatory network of *B. subtilis* was transferred to 15 other closely related, but non-model, organisms (Chapter 7). The ability to infer gene regulatory networks in other

species means that these species can also potentially be used as hosts for synthetic biology projects. In addition, genome-wide comparisons of these species and *B. subtilis*, permitted the identification of a set of orthogonal TFs and their binding sequences, providing potential candidates for engineering novel transcriptional regulatory networks specific to these organisms (Chapter 7). A wealth of information brought together by data integration can be used to facilitate the design of biological systems, as discussed in more detail in the following sections.

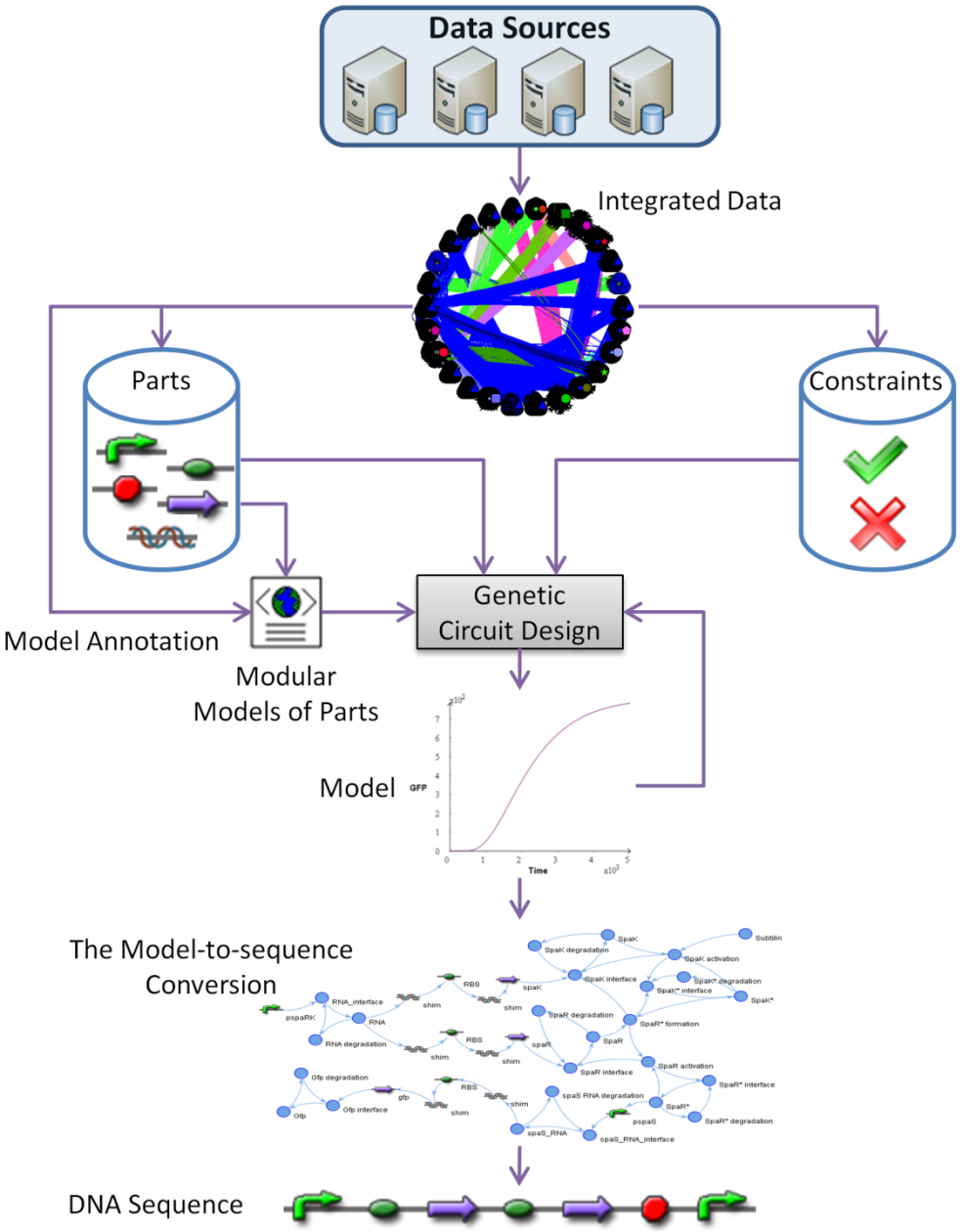


Figure 8.1: Data are integrated and used to create models of biological parts and their constraints. In order to simulate the behaviour of genetic constructs built with these parts, modular models are used, annotated with information from the integrated data. Models are then automatically converted into DNA sequences ready for synthesis.

8.2.1 Genome mining and data integration for the construction of biological parts in synthetic biology

The design of complex biological systems is limited, in part, by the number of available parts [20, 38, 71, 118]. Although much effort has gone into the experimental creation and characterisation of parts [12, 13, 70, 71, 106], wet lab approaches are heavily constrained by time and cost. Genomic information from publicly-available databases can be mined in order to identify and catalogue biological parts whose sequences are known [50, 57, 58]. However, these parts must be joined to form biologically workable designs [254]. In addition to sequence-based descriptions of parts, extensive information on factors such as interactions between biological molecules should be analysed to facilitate the rational design of genetic circuits [20].

In this project, knowledge integration was applied to the model organism *B. subtilis*. A set of databases holding information on *B. subtilis* was integrated to produce a semantically-enriched biological network, called BacillOndex. In this network, 33,043 biological concepts and 94,774 relationships connecting these concepts were captured. BacillOndex contains information about 26 different types of biological concepts including sequence-based features such as promoters, operators and coding sequences (CDSs), and other cellular molecules such as proteins, transcription factors (TFs), enzymes and compounds. Information about biological pathways and reactions are also linked to these concepts. In addition, concepts that can be used for the classification of cellular molecules were included, such as: Gene Ontology [204] terms to describe the cellular locations, molecular functions and biological processes of proteins; orthology classification numbers from the Kyoto Encyclopedia of Genes and Genomes [169] and the Clusters of Orthologous Groups [264] databases; and Enzyme Commission numbers [271]. In addition, gene expression values across a range of microarray experiments were normalised and added to BacillOndex. Furthermore, transcriptional network motifs such as autoregulatory genes and feed-forward loops (FFLs) were identified. Interactions between these concepts were represented using semantically defined labels, linking information about biological parts stored in BacillOndex.

Several promoter libraries [70, 71, 106] have previously been developed. However, the number of TFs which interact with these promoters is only around half a dozen [325]. As an alternative approach, promoters mined from the *B. subtilis* genome can provide the basis for a rich repository of promoters that can work with different TFs and sigma factors. In addition, operator parts can be used to assist with the rewiring of transcriptional networks. For example, promoters and their operators, with the spacer

sequences between them, were combined to create composite promoter sequences in this work. Such composite promoters are responsive to multiple inputs and can act as logic gates [78, 189, 293, 297].

CDSs are also important biological parts. In BacillOndex, information about CDSs was linked to information about the proteins and RNAs with which they interact, in order to facilitate the choice of appropriate parts based on the properties, types and interactions of the molecules they encode. Other basic biological parts such as ribosome binding sites (RBSs), shims and terminators have also been catalogued.

Deriving biological parts via genome mining and integrating data from several resources increases the number of parts available for circuit design for *B. subtilis*. This approach is useful since experimental approaches to the identification and characterisation of parts are difficult, and time and resource intensive. These features were linked to information about their signal transduction systems, biochemical pathways, transcriptional regulatory networks and gene expression data.

This information is valuable for the computational design of biological systems, but is usually not held in existing part repositories. One of the early, and still the best known, example of such a repository is the MIT Registry of Standard Biological Parts [12]. This repository is centralised and includes several thousand parts submitted from different laboratories. However, the repository is more suited for manually searching for nucleotide sequence information about biological parts. The computational design of genetic circuits using these parts hence requires the manually mining of information about interactions between parts and then making it available to designs tools. Moreover, tools such as the Joint BioEnergy Institute Inventory of Composable Elements registry platform [231] and BrickIt[48] have been developed which can be installed locally to serve as repositories of parts. Although these repositories allow individual laboratories to store information about parts, populating these repositories with useful information, which often requires integration of data from several sources, remains a challenge.

In addition to data integration, BacillOndex addresses the issue of the representation of data for manual and computational analyses by using biological networks, which have already been extensively used in systems biology. The modelling of biological information has usually been carried out at the gene level in these networks [149, 205, 350]. However, in synthetic biology, elements of genes such as promoters, operators, RBSs and terminators are used as biological parts. Therefore, BacillOndex demonstrates

[48] http://brickit.sourceforge.net/

the use of biological networks for data integration and representation in synthetic biology.

Although data integration for synthetic biology has been acknowledged as helpful in the construction of functional large-scale biological systems [254], data integration has not yet been utilised to any great extent. Applications are still small, some of them are merely proofs of concepts, and they are designed manually by domain experts. However, as the field of synthetic biology matures and tools to automate the design of large-scale applications become available, the lack of data for these tools to use will hinder the development of complex designs. Recently, topological information about interactions between promoters and their corresponding TFs has been applied to construct solutions for the high-level specification of given designs [320]. Although useful, even such databases are far from representing a complete set of the molecular interactions in cells. Biological systems are complex, and therefore as much data as possible should be integrated in order to predict their behaviour, for example relating to protein-protein interactions. Combining heterogeneous data in the form of biological networks as presented in this thesis is one of the first practical examples of large-scale data integration in synthetic biology.

Furthermore, a workflow-based approach, as demonstrated in this project, facilitates the reuse of data integration steps for multiple species. In addition, the data model developed is sufficiently generic to accommodate a wide range of biological concepts for prokaryotic organisms. The data integration workflow presented here can be extended to other prokaryotes, accepting genome sequences as input, and mining sequence features to identify biological parts and molecular interactions, represented in computationally-accessible formats.

8.2.2 Facilitating the computational design and automation of synthetic biological systems

8.2.2.1 Data standards

Synthetic biology is a relatively young field, and it is important that standards for the representation and exchange of data are adopted by the research community. The importance of data standards in systems biology has already been well established [301]. Large volumes of data generated by systems biology experiments are stored in biological databases. Data standards facilitate the use of different tools to analyse particular types of data from these databases. For example, there are more than 230

different SBML tools, all of which can process SBML models[49] and each has different advantages. The availability of tools that can accept and generate data in standard formats allows the construction of automated analysis pipelines. In this process, the output of one tool can be used as input for the next tool in order to achieve complex analyses that none of the tools can carry out alone. This approach has been successful in the development of large-scale bioinformatics analysis pipelines, such as those constructed for the analysis of newly sequenced genomes, which requires the use of many different types of bioinformatics tools [351]. In addition, ontologies have been extensively used to unify the meaning of terms for describing biological concepts. For example, the Gene Ontology (GO) [204] and Sequence Ontology (SO) [221] are two ontologies used to standardise the annotation of proteins and sequence features respectively.

Synthetic biology also uses data extensively. The Synthetic Biology Open Language (SBOL) is in the process of being developed as a standard for the electronic exchange of data about DNA sequences and the composition of genetic circuits [63]. In developing data standards, the involvement of the research community in the development of tools is necessary [301]. SBOL has so far been successful in bringing together researchers in order to create a community interested in the computational design of biological systems. This language is very important in order to establish data exchange, for example, between part repositories and design tools. SBOL compliant tools are already being developed. One such tool is a library called libSBOL[50], which can be used by tools to generate data in SBOL format. The first data exchange using libSBOL, in which I participated, was demonstrated at the 2012 SBOL Workshop at Seattle, USA, where data were exchanged between the SVP repository presented in Chapter 5 of this thesis and the iBioSim [47] and TinkerCell [40] tools. The language is currently not able to capture information about the constraints on biological parts. Such information is important to construct biologically feasible circuits and would be especially valuable in the computational design of genetic circuits. To facilitate the development of a fully automated design process, data about biological molecules and their interactions must be represented in machine-readable formats.

As part of this project, domain knowledge about *B. subtilis* was represented as an ontology, SynthBiOnt. This ontology formally captures details about the interactome of biological molecules, making the data available for machine access. Information about

[49] http://sbml.org/SBML_Software_Guide (accessed 05/04/2012)
[50] https://github.com/synbiodex

the genome, gene regulatory networks, protein-protein interactions, biochemical pathways, and gene expression data were modelled using the ontology. For example, specific terms were used for biological parts, gene products and their functional annotations, compounds, enzymes, biological reactions and pathways. Relationships between biological concepts, such as protein-protein interactions, were captured using the Web Ontology Language (OWL), which facilitates the use of many off-the-shelf tools such as Pellet [228] and HermiT [227] reasoners for automated reasoning and Protégé[51] for the manual browsing of the data.

SynthBiOnt is a major step in the process of computationally representing the relationships between biological parts and molecular interactions. Information about these relationships is invaluable in order to create dynamic models of parts, and hence to construct biologically feasible designs using automated approaches. For example, in order to identify an inducible promoter in SynthBiOnt, terms representing a promoter, the operator included within the sequence of the promoter and the activator that binds to the operator can be accessed. Moreover, restrictions representing the part-whole and binding relationships of the parts are defined explicitly for machine access. Therefore, a promoter that has a binding site for an activator can be defined as an inducible promoter. These relationships form the basis of the construction of dynamic models, and the semantic definitions of terms and their relationships facilitate the identification of suitable parts. The identification of parts using the ontology is further discussed in Section 8.2.3.1.

SynthBiOnt is thus a unique resource that provides an extensive amount of information available to computational tools for the large-scale design of biological systems. A wide range of queries can be constructed using this ontology. For example, the integration of gene expression and gene regulatory networks allows us to search for promoters based on their gene expression values. Promoters can be retrieved based on the existence of binding sequences or queries can be restricted to promoters without specifying any TF binding site. Parts that are regulated by a particular TF, or a kinase protein activating a response regulator type TF, or biological pathways involved for the production of a compound, can all be queried.

Furthermore, SynthBiOnt complements SBOL and uses existing standards such as the widely-used GO and SO as a step towards standardising the modelling and querying of data for synthetic biology. The data model of SBOL is already supported by an ontology with terms representing sequence features and their annotations and nucleotide

[51] http://protege.stanford.edu

sequences. SynthBiOnt includes terms from the SBOL ontology and is SBOL compliant. Therefore, SynthBiOnt can also be queried using terms from SBOL, providing access to information about the relationships between parts for the computational design of biological systems. Each sequence feature term in SynthBiOnt is also linked to the SO via subclassing, allowing the use of standard terms for the querying of biological parts. Moreover, annotations of proteins relating to biological processes, molecular functions and cellular compartments have been modelled with GO terms.

SynthBiOnt can also be accessed via a Resource Description Framework (RDF) triple store and be queried using SPARQL [211], since it uses the OWL/RDF format. One of the aims in synthetic biology is to create a web of part repositories that can be distributed and machine-accessible [231]. The use of Semantic Web technologies and unique URIs that represent biological parts can facilitate the achievement of this aim by using the existing Internet infrastructure, as demonstrated here. Moreover, the use of these standards and terms from SBOL, SO and GO can help in unifying the meaning of data across a number of repositories, and hence may enable the execution of federated queries in order to extract parts and their interactions.

As the complexity of designs and number of parts used in these designs increases, SynthBiOnt could be useful in standardising the representation of data for machine-access, mediating the access of tools to the domain knowledge available for synthetic biology. Although information about the connectivity of parts is useful in constructing biological circuits, network topology information alone is not sufficient to understand the dynamics of these systems [191]. Quantitative models should be used to link DNA sequences to the dynamic behaviour of parts and genetic circuits [49].

8.2.2.2 Mapping physical parts to their dynamic models

Computational design and simulation have been recognised as important aspects of synthetic biology [20-22]. Modelling enables the analysis of genetic circuits *in silico* prior to an experimental construction [53]. Therefore, computer-aided design tools use models of biological parts in predicting the behaviour of these constructs [43]. However, many of these tools lack access to models of biological parts [80]. In order to exchange and join models together using computational approaches, models should be composable and in standard formats.

The modelling approach presented in Chapter 5 defines a mapping between physical parts such as promoters and RBSs and their dynamic models, called Standard Virtual

Parts (SVPs). SVPs are modular, composable models with standard inputs and outputs, which allow models of larger biological systems to be constructed. The behaviour of entire genetic constructs can be assessed by combining models of their constituent physical parts. Inputs and outputs are annotated with machine-accessible data. The availability of such informative metadata for the composition of SVPs enables the computational construction of larger models.

Modular modelling approaches have also been previously demonstrated in a number of studies in the design of regulatory networks [11, 30, 41]. In these approaches, fragments of models representing biological parts were combined in order to create simulatable models that can predict the behaviour of a designed system. For example, Marchisio and Stelling used modular models of parts and used the drag-and-drop feature of the ProMoT modelling tool in order to facilitate the computer-aided design of biological systems [11]. Models were encoded using the language that is specific to this tool, although they could be exported in the SBML format. Moreover, Rodrigo and co-workers developed models of parts using SBML as well as an application to automate the composition of the models [41]. Both approaches used ODEs and models were constructed for biological parts such as promoters, RBSs, and CDSs. The composition of models was facilitated by the use of commonly defined biological signals, such as polymerases per second (PoPS) and ribosomes per second (RiPS) [116]. The applications were specifically developed to understand the contents of these models.

SVPs were also developed to contribute to these efforts in the modular modelling of synthetic genetic circuits. In addition to using standard modelling languages, the SVPs were implemented using models with part-level abstraction. Unlike previous approaches, these modular models were defined as black boxes with inputs and outputs using modelling entities that represent biological signals such as PoPS and RiPS. Moreover, these inputs and outputs were annotated using machine-level information. Therefore, the composition of these models is not application specific. Although previous approaches have demonstrated the concept of using modular models in synthetic biology, the libraries of modular models available were collections of a small number of SBML files. Models of large numbers of parts would facilitate the construction of complex systems using automated approaches. Publicly-accessible repositories of such models would be valuable for use with many different design tools.

There is currently a lack of modular models that can be used by computational tools, a situation which limits the design of complex genetic constructs. The manual creation of such models is difficult. Therefore, we used a machine-readable

specification of biological parts and their interactions from the ontology to construct a repository of SVPs. These SVPs are available in SBML format, and they can be accessed via a REST-based Web service with an API that provides programmatic access.

These models can be reused and exchanged between tools. The availability of such models, and computational access to the model repository, can facilitate the construction of large-scale biological systems. In addition, the use of data integration as presented here is a novel approach to creating models of parts by drawing data from bioinformatics databases and producing dynamic models. It is difficult for one research group to develop design tools and repositories of parts, molecular interactions, and dynamic models for every species to be engineered. What is needed is an integration of tools via standards. This modelling approach, along with the SVP repository and the Web service interface that provides computational access to models, allows automated workflows to be developed for the design of complex biological circuits.

As part of the development of the SVP repository, SVPs were annotated with information about the types of parts they represent, and their DNA sequences. SVPs and larger models built with them therefore represent synthetic genetic circuits both mathematically and biologically. The necessary metadata was captured inside the XML structure of the parts models, allowing the distribution of information regarding the syntax and semantics in the same XML files. These models can therefore be exchanged and reused between computer tools without losing information about their biological context [53]. The availability of DNA sequence metadata in dynamic models further enables the automation of the derivation of DNA sequences for computationally produced models.

8.2.2.3 From dynamic models to DNA sequences

Several CAD tools have been developed for synthetic biology [22, 32, 50]. These tools allow the manual construction of genetic circuits using the available information about parts. However, most of these tools assume that the user will create a design manually, and such an endeavour is not feasible for large biological systems [66]. For large-scale synthetic biology, model-driven approaches should be developed that automate the construction of dynamic models and the choice of appropriate solutions via simulations. Domain specific languages [123], such as GEC [43], Eugene [21] and Proto [117], are useful to search the solution space for genetic circuits. These languages allow the specification of high-level definitions of genetic circuits using types of biological parts

such as promoters, RBSs and CDSs. Tools that implement these languages can generate a number of solutions based on the topology of a specified circuit. On the other hand, evolutionary algorithms are more flexible in modelling genetic circuits without committing to any architectural specification [18]. A similar approach has been applied for the *de novo* automated design of small RNA circuits using simulated annealing [352]. RNAs were designed via computational modelling by exploring the space of around 10^{40} possible nucleotide sequences.

The automation of the design of biological circuits includes several steps. For example, using a bottom-up approach, existing biological knowledge is first used to construct dynamic models. Appropriate models that achieve the desired behaviour are then selected. Computationally produced models can represent genetic constructs to be tested *in vivo* or *in vitro*, and hence should be converted into DNA sequences for experimental testing. However, computationally-produced models can be large and complex, and difficult to convert. This process should also be automated. Therefore, these models need to be interpreted so that modelling entities that represent biological parts must be identified, and the DNA sequences of parts must be retrieved and placed in the correct order [36].

In Chapter 6, the conversion of models into DNA sequences was described explicitly using an algorithm in order to automate this conversion. This algorithm relies on the availability, within parts models, of metadata about the types and DNA sequences of parts. In order to support this automation a model-to-sequence annotation approach has been developed. Accordingly, SVPs were annotated with the types and DNA sequences of biological parts.

The ordering of parts to produce the DNA sequence specifications is derived from the graph analysis of transcriptional and translational fluxes from the models. SVPs have defined inputs and outputs representing biological signals, such as PoPS and RiPS, which facilitate the construction of models. The conversion of PoPS to mRNAs, mRNAs to RiPS, and RiPS to proteins allows the tracking of fluxes between promoter, RBS and CDS parts using SVPs. Therefore, dynamic models constructed with SVPs can be used to derive the ordering of parts.

The algorithm was implemented as a tool called MoSeC, which can accept CellML or SBML models. The tool initially converts models into graphs in which nodes represent modelling entities, such as *species* and *reactions* in SBML and *components* in CellML. Due to slight differences in representing ODEs, graph representations of SBML and CellML models are not identical. However, the conversion algorithm is

applicable to both types of models. MoSeC produces DNA sequences in the form of standard GenBank or EMBL files. The use of these standard DNA formats facilitates the reuse and exchange of the DNA sequences of genetic constructs.

It is expected that automated approaches will play a critical role in the design of large-scale novel biological systems [348]. However, the verification of designs is as important as creating them. Analogies for the large-scale design of biological systems have been drawn from electronic design automation [15, 207, 314]. Although electronic design automation is a well-established industry, it was reported that the physical verification and the creation of designs accounted 45% and 55% of the physical electronic design automation market respectively in 2000 [307]. It is also very important that the identified solutions for genetic circuits are carefully transformed into DNA sequences. Tools such as MoSeC can therefore be very valuable to derive DNA sequences, especially from models for large-scale biological systems, efficiently and reliably.

The model-to-sequence conversion presented here is a step towards the fully-automated design of these systems using a model-driven design approach. In the future, MoSeC could be directly incorporated into a design process in which the construction of models is automated and resulting files are used as input to MoSeC in order to send the corresponding DNA sequences representing the genetic circuits in standard formats for synthesis. Therefore, MoSeC and the annotation approaches presented here are useful for large-scale synthetic biology. Using computational approaches, a vast number of solutions and hence DNA sequences can be generated for a given requirement. However, not all of them are biologically plausible, and hence it is valuable to constrain the design space.

8.2.3 Constraining the design space

The behaviour of a biological system is determined by many factors, including the interactions of biological molecules such as DNA and proteins [83]. However, even for simple circuits, the design space can be very large [30]. Furthermore, not all designs are biologically plausible. For example, in order to construct a genetic circuit that includes a promoter, an RBS and a CDS using a library of 10 parts for each type of parts, 27,000 different designs can be constructed. However, only 1000 of them would have parts in the right order, and hence may be biologically plausible. For computational approaches that have access to large numbers of parts, solution spaces may not be easily searchable. Therefore, the cellular context in which these parts work and interact should be

investigated [28, 254]. Some of the techniques presented in this project enable the constraining of the design space of genetic circuits by using the properties and interactions of parts, and via their dynamic models.

8.2.3.1 Automated identification of parts

Currently, most parts are extracted manually from the literature, or they are engineered by using variants of existing biological parts such as promoters [70, 71] and TFs [325]. These parts are essentially sequence features from genomes. As the scale of synthetic biology applications increases, large numbers of parts will be needed. The mining of genomes as presented in this project is one way of increasing the number and diversity of biological parts available. Such parts should then be annotated and classified for use in synthetic biology, a task that is not efficient to achieve manually, and therefore should be automated. Moreover, the automated identification of suitable parts by genetic circuit design tools would be useful in order to constrain the design spaces for genetic circuits [49].

The SynthBiOnt ontology presented here was used to classify operators, promoters, and CDSs. The binding of a TF to an operator affects the probability of transcription from a promoter. Therefore, operators were classified based on the type of regulation they are involved in. Operators are usually included as part of promoter parts. The relationships between operators and promoters were modelled with whole-part relationships using cardinality restrictions. Therefore, queries were able to classify promoters that have one operator for an activator or repressor, or which have no operator. For example, 51 inducible and 85 repressible promoters could be classified for use in genetic circuit designs. Promoters were also classified using information about sigma factors, which affect the initiation of transcription and therefore the timing of gene expression in cells. For example, promoters that are controlled by the alternative sigma factor SigD are active in motile cells [120]. This machine-level information can help in identifying promoters that could work for a particular type of genetic circuit design. Moreover, CDSs were classified, for example, using standard GO terms to identify parts that encode for kinases and response regulators. For example, the query ‘`CDS and (encodes some (Protein and (has_function some go:GO_0000155))`’ was able to retrieve CDS parts encoding proteins that are assigned the `GO_0000155` (‘two-component sensor activity’) term from the GO.

SynthBiOnt, hence, demonstrates an approach to automating the identification of parts from a wealth of existing knowledge for large-scale synthetic biology. This

ontology can be used directly as input to existing off-the-shelf reasoners in order to classify parts. In addition, OWL-like or SPARQL queries can extract useful information about parts and interactions from the ontology. This approach means that biological parts for the rational design of biological circuits can be computationally identified.

Furthermore, machine-accessible information about relationships between biological parts is very important in order to constrain the design spaces of genetic circuits and to find biologically-plausible solutions. This information is machine-accessible in SynthBiOnt and can be used to choose appropriate parts. For example, in order for a TF to function its binding site must also be present in the biological system to be engineered. Therefore, parts that have desired interactions within the engineered circuits and cells could be computationally identified, reducing the number of infeasible designs. The availability of this information can also facilitate the use of computer tools in order to map high-level designs to individual parts as described below [207, 320].

8.2.3.2 From high-level designs to individual parts

A genetic circuit designed at a high level of abstraction should eventually be mapped to individual DNA sequences [43, 207]. High-level specifications of genetic circuits do not include the DNA sequences of biological parts; however, they can include the order and types of parts or information about biological constraints. For example, the GEC language allows the regulation type of a promoter to be defined [43]. A conceptual TF specified in such a design can be defined to be activating or repressing the promoter. The target system can then be decomposed into well-defined biological parts. However, for large-scale circuits this mapping can be difficult. In order to automate this process, extensive information about the relationships of the biological parts which a target system can be decomposed into should be machine-accessible [320].

SynthBiOnt could therefore be useful for design tools to access information about the relationships between biological parts in order to implement high-level definitions of target systems using biological parts. However, more complex systems would possibly need to be defined with more abstract definitions. Rather than specifying the type of a part, such as a promoter, biological systems can be defined with subsystems each of which achieves a particular goal. For example, such subsystems can involve sensing a biological signal, amplifying the signal or checking the threshold of the signal in order to activate a reporter subsystem [31]. Therefore, large biological systems can initially be decomposed into subsystems.

One way to reduce this complexity is to identify network motifs with known

functions [195]. The topology of a biological system may affect its robustness to parameter variation, and therefore is important in achieving a desired behaviour especially when biochemical parameters are not fully known [353]. SynthBiOnt includes information about transcriptional network motifs such as autoregulatory genes that can be used to achieve bistability [196] or fast response times [195]. FFLs [197] that can act as pulse generators and signal filters were also identified. Such network motifs can be used as a basis to achieve desired behaviours [30, 85, 191]. Therefore, they are useful in order to constrain the initial design space of possible solutions and to construct reliable biological circuits. Details about the properties and connectivity of biological parts can be used to identify biological parts required for genetic circuit designs; however, the behaviour of the resulting genetic circuits should be investigated by computational simulations in order to rank possible solutions.

8.2.3.3 Model-driven design to identify solutions

How can one automate the design of a genetic circuit using individual biological parts, each with its own characteristics, and then choose a biologically-plausible solution? The use of abstractions that hide the physical details of molecular interactions has already facilitated the construction of biological devices from simple biological parts. The automation of this process is a *systems design* problem. Henzinger and Sifakis describe systems design as "the process of deriving, from requirements, a model from which a system can be generated more or less automatically" [306]. This model-driven approach has been successfully used to automatically design and verify integrated electronic circuits *in silico* before implementing them physically.

Design automation for synthetic biological systems is also gaining popularity in order to construct large-scale genetic circuits that are not possible to achieve manually [314]. Design, simulation, synthesis and testing will also be key areas in the design of these systems. However, as mentioned above, these systems will have large design spaces. SVPs are suitable for automating the searching of large design spaces for both bottom-up and top-down approaches.

These SVPs can be joined computationally to construct large, simulatable models in order to choose solutions in a bottom-up approach. An SVP's output can be connected to another SVP's input of the same type. These inputs and outputs make the SVPs composable and determine how they can be combined. Moreover, the SVP repository presented in this work provides computationally-accessible methods which can be used to choose appropriate SVPs for initial designs. In the repository, SVPs are categorised

according to their types, including information about the classification of parts from the SynthBiOnt ontology. GO terms and COG numbers are linked to the records of SVPs, allowing them to be selected using these criteria, prior to the assembly of larger models. In addition, the repository stores information about the interactions between SVPs. This information can be used to identify suitable SVPs, restricting the number of solutions to biologically plausible designs.

Top-down approaches in the automation of genetic circuits have been demonstrated via the DSLs and the tools implementing them [123]. A design specified with these languages is then mapped to individual parts. However, the solutions should be verified via simulations. SVPs already provide a mapping between biological functions and DNA sequences. Each SVP may contain a set of equations that takes inputs and convert them into outputs. Therefore, SVPs can be combined to integrate all these transfer functions for the verification of selected solutions. These SVPs can therefore be plugged into tools that implement DSLs. Model annotation is very important in facilitating these model-driven approaches in synthetic biology.

Although model annotation has been used extensively in systems biology, the needs of model annotation for synthetic biology are different. In systems biology, models are usually used to understand biological systems. Therefore, entities representing biological reactions and cellular molecules such as proteins that participate in those reactions are annotated to aid machine-level understanding of these entities. However, in synthetic biology the emphasis is on the design of new biological systems using biological parts. Using the model-driven design of these systems, models of biological parts should be machine-accessible in order to design new biological systems and search the space of possible solutions. In this work, the automation of the composition of models from modular models in constructing new models, and the conversion of constructed models into DNA sequences, were both facilitated via the annotation of models. Annotations used here could also potentially be generically applicable to the needs of model annotation in synthetic biology.

8.2.4 Designing biological circuits across multiple organisms

The number of organisms that can currently be used as chassis for synthetic biology are currently largely limited to well-studied model organisms such as *E.coli* and *B. subtilis* [2]. The use of various organisms that have different phenotypes and can live in a range of environments would be valuable for diverse applications. There are many industrial strains that are already used in biotechnology to produce different types of products,

such as vitamins [328], antibiotics and enzymes [329]. Moreover, organisms such as *Geobacillus* species can be tolerant to high temperatures and have potential in the production of biofuels [330]. The gene regulatory networks of these organisms can be engineered in order to control the expression of genes in creating novel applications for non-model organisms. However, although the genomes of many non-model organisms have been sequenced, detailed knowledge about their gene regulatory networks is not available. Moreover, novel circuits introduced into host cells may cause undesired interactions [51]. In order to minimise crosstalk, a set of orthogonal TFs that can be used across different organisms would be valuable [49].

In Chapter 7, the BacillusRegNet database, which is based on the RegNet [154] system, was described. This system was used to transfer knowledge about experimentally-identified gene regulatory networks of *B. subtilis* to closely related, non-model organisms. The list of non-model organisms currently includes 13 *Bacillus* and 2 *Geobacillus* species. BacillusRegNet was used to predict promoter and TF binding sequences, and whether the genes are inhibited or activated by these TFs in these organisms. In addition, genome-wide comparisons of the regulatory networks between *B. subtilis* and each of these organisms were generated, in order to identify a set of 'orthogonal' TFs (OTFs) present in *B. subtilis* but not in the other organisms. For example, the GutR TF, which has been previously reported to be specific to *B. subtilis* [132], was found to be orthogonal in all organisms in BacillusRegNet, except in the *B. subtilis* group of organisms. These OTFs are potentially valuable for engineering crosstalk-free genetic circuits in the non-model organisms.

The use of OTFs in synthetic biology is not a new idea. OTFs such as TetR, CI and LacI, where the latter can be used in *B. subtilis* or *lacI* deficient *E.coli* cells, do not exist in target organisms, and they have been extensively used in many designs since the development of early applications in synthetic biology [70, 71, 107, 108]. However, the number of these parts is small, and therefore new TFs that would not interfere with the existing transcriptional machinery of cells would be valuable for larger designs. Experimental approaches have recently been used to create synthetic TFs that bind to specific DNA sequences [325]. For example, proteins called transcription activator-like effectors have been designed in order to control the activation or repression of promoters and were tested in human cells [354].

The comparative genomics approach presented in this project is complementary to experimental approaches aiming to increase the number of OTFs. In this approach existing knowledge about model organisms can be used computationally to find OTFs

for non-model organisms. This method is cost and time effective compared with the use of experimental techniques. Moreover, as new organisms are sequenced they can be added to BacillusRegNet.

BacillusRegNet increases the number of chassis organisms available to synthetic biologists. The construction of genome-wide gene regulatory networks for non-model organisms potentially facilitates the rational design of novel transcriptional regulatory networks in these organisms. The binding sequences of identified OTFs are also available. Therefore, these orthogonal parts can be used to rewire the transcriptional regulatory networks in order to construct novel applications specific to non-model organisms.

The novel approaches presented in this thesis are useful in order to facilitate the computational design of synthetic biological systems. Although most genetic circuits are still designed manually, the manual specification of genetic circuits is challenging when solutions for large-scale biological systems must be found. This project demonstrates how large-scale data integration can help in automating the design of functional large-scale biological systems.

This work is probably the first large-scale integration of data from several heterogeneous data sources in synthetic biology. Unlike most data integration approaches in systems biology so far, which generally focus upon the integration of data at the gene level, in this study data integration is applied in order to extract information about biological parts. Mining information about biological parts from the genome was combined with integrating data about different aspects of the cell biology of *B. subtilis*. Such information provides a more compherensive basis for the use of biological parts in the design of biological systems.

Moreover, such informative data were presented as machine-accessible, facilitating the qualitative analysis of relationships between biological parts. This information was modelled in the form of an ontology using existing standards such as SBOL, SO, and GO, and could potentially be used in order to standardise the modelling of existing knowledge for machine-access in synthetic biology. This information was further used as a basis for the construction of modular models of biological parts, and to annotate these models for machine access.

Although modular modelling approaches for synthetic biology have previously been demonstrated, those models do not include machine-level metadata, and hence can only be used by applications specifically written to use fragments of models of parts. Model annotation has already been extensively used in order to provide biological contexts for

systems biology models, but such approaches have not been utilised very often in synthetic biology. In this project, SVPs were annotated with metadata, facilitating the computational composition of these models. Therefore, SVPs can be combined to construct simulatable models in order to predict the behaviour of large-scale biological systems. Moreover, a repository of modular models for large numbers of parts has not previously been available. The SVP repository presented here can be accessed computationally, and therefore is a useful resource for design tools. Furthermore, model annotation has been extended to include the types and DNA sequences of parts represented by SVPs. The availability of this information is necessary to interpret computationally constructed models in order to derive the DNA sequences necessary to encode biological systems. This process was automated using an algorithm and a tool that implements the algorithm. This approach is novel in automatically specifying genetic circuits at the DNA level using dynamic models, which has not previously been demonstrated.

Modular modelling and the annotation of these models demonstrated in this project can facilitate the use of model-driven design for synthetic biology in order to automate the design of biological systems. In this process, models can be used to design a biological system, explore the space of possible solutions via computational modelling, and turn the model that represents a solution for a given set of requirements into a DNA sequence in order to use cells as platforms to experimentally test the designs. Therefore, these approaches are valuable in the construction of predicable, large-scale biological systems. Although annotations were applied to SVPs, any dynamic model can be annotated using the approach described here. Therefore, the annotation approaches presented in this work represent a demonstration of the strategy for model annotation in synthetic biology.

Finally, BacillusRegNet increases the number of chassis organisms available to synthetic biology and can guide the design of novel regulatory networks using existing knowledge. Using BacillusRegNet, 696 putative TFs and 7,856 binding sites including promoters have been identified from 15 organisms. Moreover, 39 OTFs with known binding sequences, and a list of organisms in which these OTFs can function as orthogonal parts, have been identified. These OTFs could potentially be used to engineer useful applications in those non-model organisms.

8.3 Future work

Data integration is a valuable approach informing the design of synthetic genetic

circuits. However, this approach has some drawbacks. The source databases used in this project may not be error-free. For example, the STRING database provides valuable data about physical and functional protein-protein interactions, but some of the data are computationally produced using text mining techniques, and such an approach may introduce false positives. Therefore, more work is needed to manually validate the data. Moreover, source databases may not include up-to-date records. For example, after the re-sequencing of the *B. subtilis* genome in 2009, 407 genes were renamed and additional locus tags were introduced for new open reading frames. The results were stored in the BacilluScope database [81]. However, the KEGG database included records referring to old locus tags. As a result, not all information about CDSs and proteins from these databases could be integrated. Therefore, biological entities from old databases should be manually annotated with new data, such as changed locus tag names, prior to integration.

The SVP repository currently includes models in the SBML Level 2 format. However, SBML Level 3 has recently been proposed[52] in order to explicitly represent modular models, which facilitates the construction of complex models from several smaller ones. Therefore, it would be valuable to update the repository to conform to this new version of SBML. The repository should also be updated to provide models in the CellML1.1 format, which supports explicit modularity.

The content in the SVP repository can further be improved. The RBS sequences in the repository are putative RBS regions and do not include annotations for the exact binding locations of ribosomes. Tools such as MAST from the MEME suite [279] could be used to search the entire *B. subtilis* genome, using consensus RBS motifs, and to annotate and validate these RBS sequences. In addition, tools such as RBSCalculator [110] and RBSDesigner [114] could be used to estimate the translational efficiency of these RBS sequences, in order to construct predictable biological systems using these RBSs. The SVP repository currently includes information about parts and interactions for several transcriptional regulatory networks in *B. subtilis*. This repository should be extended to other biological pathways and new types of parts, incorporating interactions such as methylation and protein binding. This information is already available in the BacillOndex dataset, but has not yet been incorporated into the SVP repository.

MoSeC is currently used manually. However, in order to incorporate the tool into automated workflow-based processes, the tool needs to be run from the command line.

[52] http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Hierarchical_Model_Composition (accessed 21/05/2012)

A Web service interface that takes dynamic models as input and returns the DNA sequences in standard formats using MoSeC would also be valuable. Moreover, in addition to exporting the DNA sequences in EMBL and GenBank format, SBOL export should be added. MoSeC could also be enhanced to automatically annotate [202] existing synthetic biology models. Such a feature would be valuable for the computational reuse and exchange of synthetic biology models.

Furthermore, the process of converting dynamic models into DNA sequences in MoSeC should be optimised. For example, for models that contain operon structures, it is not clear which RBS-CDS pair to place first. Such an optimisation needs machine learning for synthetic biology using the available biological knowledge. Machine learning can also potentially be applied to automate the selection of plasmids and integration sites for the efficient transformation of genetic circuits into chromosomes. Moreover, DNA sequences can be optimised for codon usage based on a selected chassis, in order to increase the efficiency of the translation of proteins [16]. Currently, terminator sequences are specified via the annotation of models and only one terminator sequence is allowed. However, using a DNA sequence more than once for the same genetic circuit may prevent the successful chromosomal integration of the circuit. Therefore, MoSeC should be updated to give access to the SVP repository in order to use different terminator sequences for each transcriptional unit in genetic circuit designs.

BacillusRegNet is useful for the extraction of orthogonal TFs and their binding sequences. However, this information is currently stored in a relational database, and for the purposes of this project was extracted manually by querying the database. To facilitate an automated approach, the system could be extended by implementing a Web service that provides this information directly from the database. The system currently uses model organisms in order to predict TFs in non-model organisms. The use of information about the binding domains of TFs can be used to predict additional TFs that could not be identified for non-model organisms [132, 342, 351]. Such an approach could be useful for the identification of orthogonal TFs for *B. subtilis* which exist in other species but not in this species. Identified orthogonal TFs from BacillusRegNet also have models in the SVP repository. These models should be annotated with lists of organisms in which these TFs can be used as orthogonal parts.

8.4 Conclusion

The approaches and tools developed in this project facilitate the integration of existing

biological data to constrain the design space of synthetic genetic circuits. Extensive amounts of biological data about biological parts and their interactions were integrated and represented in computationally-accessible formats, using existing standard wherever possible, and proposing new standards for areas in which standards do not currently exist. The availability of such machine-accessible data and the approaches presented in this thesis facilitate the automation of the design of genetic circuits. These approaches are especially important for the time- and cost-effective design of complex and large-scale biological systems.

The BacillOndex dataset is a semantically-rich biological network that facilitates the graph-based analyses of biological entities and their relationships. Many aspects of the cell biology of *B. subtilis* are captured in this dataset. Although data sources specific to *B. subtilis* were used, many freely-available databases provide data about a wide range of prokaryotes. The automated, workflow-based approach demonstrated here, and the data model developed, can be extended to other prokaryotic genomes to integrate data in order to extract and characterise biological parts and their relationships.

Knowledge about the model organism *B. subtilis* was presented in the form of an ontology, SynthBiOnt, formally capturing a wide range of information about biological parts and their relationships. This ontology uses terms from existing standards such as SBOL, SO and GO. The ontology makes existing biological knowledge about this organism available to a wide range of off-the-shelf analysis and reasoning tools. Automated reasoning and querying of the ontology can be used to design tools to access and classify a large number of parts and interactions. Furthermore, the use of Semantic Web technologies for the representation of parts is a step towards the linking of part repositories using existing Internet infrastructure.

In addition to the qualitative analysis of parts and their interactions, the work described here provides a mapping between physical parts and their modular mathematical models, in the form of SVPs [80]. This modular modelling approach facilitates the model-driven design of genetic circuits. In addition, information from the ontology was used to create a repository of modular models of biological parts and their relationships. The repository of SBML models is available to design tools, and this approach facilitates the exchange and reuse of dynamic models.

These modular models are annotated with appropriate metadata. These machine-readable annotations make it possible to compose models independent of applications, and enable the automation of this process. Such automation can reduce the number of iterations needed to produce robust and predictable designs through computational

simulation. Therefore, the availability of these models and informative metadata for their composition facilitates large-scale synthetic biology.

These models are linked to information about DNA sequences via machine-readable annotations. Metadata in dynamic models was used to automatically derive DNA sequences from models composed of modular parts. This approach is the first such automated approach, to the best of the present author's knowledge. Automating this conversion is especially important for computationally produced large models, and is essential to the fully automated design of biological systems.

Designing biological systems requires an extensive amount of information about the organisms engineered. As a result, the lack of knowledge about non-model organisms limits the number of hosts available to synthetic biology. Using RegNet, in collaboration with the developers of this system, data integration has been applied to the genome-level prediction of transcriptional regulatory networks of 13 *Bacillus* and 2 *Geobacillus* species. Promoters, TF binding sequences, and the transcriptional activation and inhibition of genes in these organisms have been inferred. Using a comparative genomics approach, this platform enabled the identification of orthogonal TFs and their binding sequences from *B. subtilis* that can be used in closely related non-model organisms. These orthogonal parts can therefore be used for the design of novel and predictable transcriptional networks that do not have undesired interactions with the host systems.

In this project, data integration has been utilised in order to facilitate the construction of biologically plausible circuits using existing biological knowledge. The novel approaches and tools that have been presented here are useful for computational tools, and therefore for the automation of the design of large-scale biological systems. Although these approaches and tools are demonstrated for *B. subtilis*, they could be usefully applied to the design of biological circuits for other organisms.

Appendix A. Parsers Used by the BacillOndex Plugin

A.1 BacilluScope

This parser parses data that are available as text files from the BacilluScope database.

Parser Parameters:

- **DirectoryPath**: Directory path containing BacilluScope files.
- **TabDelimetedDataFile**: Tab-delimited BacilluScope file containing information about the genome.
- **COGClassificationDataFile**: Tab-delimited BacilluScope file containing the COG classifications.
- **SynonymExcludeList**: Comma-separated list of gene name and synonym pairs. Format: <Gene name>.<synonym>. The synonyms in the list are excluded from the given genes.

A.2 Dbtbs

This parser is used to parse DBTBS XML file to create concepts and their relationships for the gene regulatory networks of *B. subtilis*.

Parser Parameters:

- **DirectoryPath**: Directory path containing DBTBS files.
- **DbtbsXmlFile**: DBTBS XML file.
- **SynonymExcludeList**: Comma-separated list of gene name and synonym pairs. Format: <Gene name>.<synonym>. The synonyms in the list are excluded from the given genes.
- **SynonymIncludeList**: Comma-separated list of gene name and synonym pairs. Format: <Gene name>.<synonym>. The synonyms in the list are added to the given genes.

A.3 String

This parser parses data that are available as text files from the STRING database.

Parser Parameters:

- **DirectoryPath**: Directory path containing STRING files.

- **ProteinActionsDataFile**: Tab-delimited protein actions file from STRING. The file contains protein-protein interactions.
- **ProteinLinksDataFile**: Tab-delimited protein-protein links from STRING. The file includes scored links between proteins.
- **TaxonId**: The taxon ID of the organism.
- **FusionScoreThreshold**: Protein fusion score threshold value to include a protein-protein link.
- **CombinedScoreThreshold**: Combined score threshold value to include a protein-protein link.
- **CoexpressionScoreThreshold**: Co-expression score threshold value to include a protein-protein link.
- **CooccurranceScoreThreshold**: Co-occurrence score threshold value to include a protein-protein link.

A.4 KEGGExpression

This parser parses data from KEGG EXPRESSION to find minimum and maximum gene expression values. The values are normalised according to an algorithm developed by Dawes and Glassey [277].

Parser Parameters:

- **KeggExpressionFolder**: Directory path containing KEGG EXPRESSION files.
- **AccessionFilePath**: Tab-delimited file that contains mappings between the locus tags and the open reading frame (ORF) IDs. The first column contains the locus tags and the second column contains the ORF IDs. The values are preceded by a prefix separated by a colon, for example: 'bsu:BSU00010 subtilist-bsu:BG10065'
- **Species**: KEGG prefix for the organism.
- **A**: A data normalisation parameter from Dawes and Glassey [277], which is the maximum difference in a gene's rankings across experiments.
- **X**: A data normalisation parameter from Dawes and Glassey [277], denoting the percentage of genes that affect the normalisation the least and the most. These genes are excluded from the normalisation step.
- **Debug**: True to save information about the normalisation steps.
- **DebugOutputFolder**: Directory path to save information.

A.5 Name To Accession Converter

This transformer creates accessions using concept names given with a pattern.

Parameters:

- **RegExpPatternToSearchNames**: The regular expression pattern to search for concept names. The names that match the regular expression are recorded as accessions for the concept.
- **ConceptClassRestriction**: Comma-separated list of concept classes to search for.
- **CV**: The name of the CV to create the accessions for.

A.6 Concept Remover

This transformer removes the concepts given with their concept class names.

Parameters:

- **ConceptListToRemove**: List of comma-separated concept class names.

A.7 Name Remover

This transformer removes concept names that do not match with a given pattern.

Parameters:

- **PatternToKeepNames**: Regular expression pattern to keep the names.
- **ConceptClassRestriction**: Comma-separated list of concept class names to search for.
- **KeepPreferredNamesIfNoMatchFound**: False as default. True to keep the preferred concept names if the given pattern is not found.
- **AlwaysKeepPreferredNames**: False as default. True to keep the preferred names whether or not there is a match.

A.8 Relation Collapser With Name Preference

This transformer is a modified version of Ondex's `Relation Collapser` transformer. It provides additional 'PreferredNameCV' and 'NameUnmatchPattern' parameters. Parameter definitions are based on those of `Relation Collapser`'s.

Parameters:

- **RelationType**: The relationship type to collapse.
- **ConceptClassRestriction**: A concept class restriction as an ordered pair.
- **CVRestriction**:A CV restriction as an ordered pair.

- **CloneGDS**: True to add inherited GDS properties to the new collapsed concepts.
- **PreferredNameCV**: Optional. Name of the CV to use in the resulting concepts.
- **NameUnmatchPattern**: Optional. Regular expression to set preferred names. Preferred names that do not match the expression from datasets are set as the preferred name in the integrated dataset.

A.9 Network Motif Generator

This transformer traverses the graph and finds the motifs that match with the known transcriptional network motifs such as positive and negative autoregulatory genes and FFLs.

A.10 Sequence Location Updater

This transformer annotates promoter, operator and terminator concepts with start and end locations on the genome.

Parameters:

- **LogFile**: The file path of the log file.
- **FASTAFilePath:** The path of the FASTA file for the genome sequence.

A.11 Sequence Relation Updater

This transformer rearranges the relationships of operators, promoters, and CDSs based on their chromosomal locations. New relationships, such as `part_of` and `upstream_of`, are established between the concepts.

A.12 Sequence Extractor

This transformer extracts the RBS and spacer sequences (shims) between the annotated promoters and their downstream CDSs.

Parameters:

- **Log Directory**: The path of the log directory.
- **FASTAFilePath:** The path of the FASTA file for the genome sequence.

Appendix B. Microarray Experiments

Table B-1: The list of microarray experiments from which the normalised minimum and maximum gene expression values were derived.

PMID	Description	Experiment ID	Experiment description
11160890	Glucose repression	ex0000258	Glucose repression
		ex0000259	CcpA-independent glucose repression
11557812	DegU, ComA and PhoP regulons	ex0000260	DegU regulon, *degU* overexpression
		ex0000261	ComA regulon, *comA* overexpression
		ex0000262	PhoP regulon, *phoP* overexpression
11948146	ComK regulon	ex0000659	ComK regulon, *comK* disruption, experiment1
		ex0000660	ComK regulon, *comK* disruption, experiment3
		ex0000661	ComK regulon, *comK* disruption, experiment3
11717295	Two-component systems	ex0000263	CitT regulon, *citT* overexpression
		ex0000264	DesR regulon, *desR* overexpression
		ex0000265	LytT regulon, *lytT* overexpression
		ex0000266	YbdJ regulon, *ybdJ* overexpression
		ex0000267	YcbB regulon, *ycbB* overexpression
		ex0000268	YcbL regulon, *ycbL* overexpression
		ex0000269	YccH regulon, *yccH* overexpression
		ex0000270	YclJ regulon, *yclJ* overexpression
		ex0000271	YdbG regulon, *ydbG* overexpression
		ex0000272	YdfI regulon, *ydfI* overexpression
		ex0000273	YesN regulon, *yesN* overexpression
		ex0000274	YfiK regulon, *yfiK* overexpression
		ex0000275	YhcZ regulon, *yhcZ* overexpression
		ex0000276	YkoG regulon, *ykoG* overexpression
		ex0000277	YrkP regulon, *yrkP* overexpression
		ex0000278	YtsA regulon, *ytsA* overexpression
		ex0000279	YufM regulon, *yufM* overexpression
		ex0000280	YvcP regulon, *yvcP* overexpression
		ex0000281	YvfU regulon, *yvfU* overexpression
		ex0000282	YvqA regulon, *yvqA* overexpression
		ex0000283	YvqC regulon, *yvqC* overexpression
		ex0000284	YvrH regulon, *yvrH* overexpression
		ex0000285	YvrH regulon, *yvrH* overexpression
		ex0000286	YxjL regulon, *yxjL* overexpression
12618455	CodY regulon, *codY* mutant/wild type	ex0000381	CodY regulon, *codY* mutant/wild type
12644242	Overexpression of the seven sigma factors of extracytoplasmic function family	ex0000744	SigM regulon, *sigM* overexpression
		ex0000745	SigY regulon, *sigY* overexpression
		ex0000746	YlaC regulon, *ylaC* overexpression
		ex0000747	SigV regulon, *sigV* overexpression
		ex0000748	SigX regulon, *sigX* overexpression
		ex0000749	SigW regulon, *sigW* overexpression
		ex0000758	SigZ regulon, *sigZ* overexpression
12823818	Genome-wide screening for	ex0000395	Genome-wide screening for TnrA-regulated genes associated with a TnrA box

	TnrA-regulated genes associated with a TnrA box		
12897001	Genes required for mannitol and glucitol assimilation in *B. subtilis*	ex0000369	Growth on mannitol/glucose
		ex0000370	Growth on sorbitol (glucitol)/glucose
12949160	A transcriptome comparison of a wild-type *B. subtilis* strain growing under glucolytic or gluconeogenesis conditions	ex0000358	Growth on malate/glucose
14769884	*sigY* operon	ex0000340	SigY regulon, *sigY* overexpression
		ex0000377	SigY regulon, *sigY* disruption (N starvation)
		ex0000782	time course during N starvation, 0 h
		ex0000785	time course during N starvation, 6 h
		ex0000818	time course during N starvation, 12 h
15033535	SigD-regulated genes in *Bacillus subtilis*	ex0000360	SigD regulon, *sigD* disruption
		ex0000940	SigD regulon, *sigD* disruption
		ex0000941	SigD regulon, sigD disruption
		ex0000942	SigD2 regulon, *sigD2* disruption
		ex0000943	SigD2 regulon, *sigD2* disruption
		ex0000944	SigD3 regulon, *sigD3* disruption
		ex0000945	SigD3 regulon, *sigD3* disruption
14730579	ABC transporter solute-binding proteins of *Bacillus subtilis* membrane	ex0001360	ABC transporter solute-binding proteins of *Bacillus subtilis* membrane
15317768	LmrA regulon, *lmrA* inactivation	ex0000798	LmrA regulon, *lmrA* inactivation
16166527	PerR regulon, *perR* inactivation	ex0001437	PerR regulon, *perR* inactivation
		ex0001438	PerR regulon, *perR* inactivation
		ex0001439	PerR regulon, *perR* inactivation
		ex0001440	PerR regulon, perR inactivation
16553878	YvaN regulon, *yvaN* inactivation	ex0001597	YvaN regulon, yvaN inactivation
		ex0001598	YvaN regulon, *yvaN* inactivation
		ex0001599	YvaN regulon, *yvaN* inactivation
		ex0001600	YvaN regulon, *yvaN* inactivation
17227471	*rapD*, a direct target of transcription repression by RghR, negatively regulates *srfA* expression	ex0000824	YvaN regulon, *yvaN* disruption
16306698	functional analysis of the YvrGHb two-component system	ex0001750	YvrH regulon, *yvrH* inactivation
		ex0001751	YvrH regulon, *yvrH* inactivation
		ex0001752	YvrH regulon, yvrH inactivation
		ex0001753	YvrH regulon, *yvrH* inactivation

17850253	DegU regulon, *degU* inactivation	ex0001755	DegU regulon, *degU* inactivation
		ex0001756	DegU regulon, *degU* inactivation
		ex0001757	DegU regulon, *degU* inactivation
		ex0001758	DegU regulon, *degU* inactivation
		ex0001759	DegU regulon, *degU* inactivation
		ex0001760	DegU regulon, *degU* inactivation
		ex0001761	DegU regulon, *degU* inactivation
		ex0001762	DegU regulon, *degU* inactivation

Appendix C. The SBOL Mapping

```
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  bo:   <http://www.bacillondex.org#>
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  so:   <http://purl.org/obo/owl/SO#>
PREFIX  sbol: <http://sbols.org/sbol.owl#>
PREFIX  fn: <http://www.w3.org/2005/xpath-functions#>

CONSTRUCT
{
   ?SequenceFeatureResource a sbol:DnaComponent ;
              a ?SOClass ;
              a ?SequenceFeature ;
              sbol:dnaSequence ?DnaSequence ;
              rdfs:label ?name ;
              rdfs:comment ?description .
   ?DnaSequence  a sbol:DnaSequence ;
               sbol:nucleotides ?NA .
}
WHERE
 {
  OPTIONAL {?SequenceFeature rdfs:label ?name . }
  OPTIONAL {?SequenceFeature rdfs:comment ?description . }
  ?SequenceFeature rdfs:subClassOf  ?NARestriction ;
               rdfs:subClassOf  ?SuperClass .
  ?NARestriction rdf:type          owl:Restriction ;
             owl:onProperty     bo:NA ;
             owl:hasValue ?NA .
  ?SuperClass rdfs:subClassOf ?SOClass .

 LET ( ?DnaSequence := IRI(fn:concat(str(?SequenceFeature),"_sbolsequence")))
 LET ( ?SequenceFeatureResource := IRI(fn:concat(str(?SequenceFeature),"_sbol")))

 FILTER (
    regex(str(?SuperClass) , "http://www.bacillondex.org#Promoter") ||
    regex(str(?SuperClass) , "http://www.bacillondex.org#CDS") ||
    regex(str(?SuperClass) , "http://www.bacillondex.org#RBS") ||
    regex(str(?SuperClass) , "http://www.bacillondex.org#Terminator") ||
    regex(str(?SuperClass) , "http://www.bacillondex.org#Shim") ||
    regex(str(?SuperClass) , "http://www.bacillondex.org#Operator")
   )
 }
```

Figure C-1: The mapping of promoters, CDSs, RBSs, terminators, shims and operators to `DnaComponent` resources from SBOL. Names and descriptions are mapped as `rdfs:label` and `rdfs:comment` respectively. Nucleotide sequences are mapped to the `sbol:nucleotides` of `DnaSequence` resources.

```
PREFIX  rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX  bo:   <http://www.bacillondex.org#>
PREFIX  owl:  <http://www.w3.org/2002/07/owl#>
PREFIX  rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  so:   <http://purl.org/obo/owl/SO#>
PREFIX  sbol: <http://sbols.org/sbol.owl#>
PREFIX  xsd:  <http://www.w3.org/2001/XMLSchema#>
PREFIX  fn:   <http://www.w3.org/2005/xpath-functions#>
PREFIX  afn:  <http://jena.hpl.hp.com/ARQ/function#>
PREFIX  apf:  <http://jena.hpl.hp.com/ARQ/property#>

CONSTRUCT
{
   ?SequenceFeatureResource sbol:annotation ?DnaAnnotation .
   ?DnaAnnotation a sbol:SequenceAnnotation ;
              sbol:strand "+" ;
              sbol:bioStart ?begin ;
              sbol:bioEnd ?end ;
              sbol:subComponent ?SequencePartResource .
}
WHERE
{
  ?SequenceFeature bo:BEGIN   ?sequenceFeatureBegin ;
               rdfs:subClassOf   ?HasPartRestriction ;
               rdfs:subClassOf  ?SuperClass .
  ?HasPartRestriction rdf:type    owl:Restriction ;
             owl:onProperty     bo:has_part ;
             owl:someValuesFrom  ?SequencePart .
  ?SequencePart bo:BEGIN   ?partBegin ;
            bo:END   ?partEnd .

  LET ( ?partName := fn:substring (str(?SequencePart),28))
  LET ( ?DnaAnnotation := IRI(fn:concat(str(?SequenceFeature),"_sbol_annotation_", ?partName)))
  LET (?begin := fn:abs(xsd:integer(?sequenceFeatureBegin) - xsd:integer(?partBegin)) + 1)
  LET (?end := fn:abs(xsd:integer(?sequenceFeatureBegin) - xsd:integer(?partEnd)) + 1)
  LET ( ?SequenceFeatureResource := IRI(fn:concat(str(?SequenceFeature),"_sbol")))
  LET ( ?SequencePartResource := IRI(fn:concat(str(?SequencePart),"_sbol")))

  FILTER (
     regex(str(?SuperClass) , "http://www.bacillondex.org#Promoter") ||
     regex(str(?SuperClass) , "http://www.bacillondex.org#CDS") ||
     regex(str(?SuperClass) , "http://www.bacillondex.org#RBS") ||
     regex(str(?SuperClass) , "http://www.bacillondex.org#Terminator") ||
     regex(str(?SuperClass) , "http://www.bacillondex.org#Shim") ||
     regex(str(?SuperClass) , "http://www.bacillondex.org#Operator")
   )
}
```

Figure C-2: The mapping of `SequenceAnnotation` classes. The sequence features that are part of promoters, operators, CDS, RBSs, shims and terminators are mapped as sequence annotations from SBOL. Annotations are linked to the sequence features by the `sbol:subComponent` property.

Appendix D. Classification Results of Biological Parts

Table D-1: The list of inducible promoters that can be activated by one TF only. Columns show the labels, start and end positions and sigma factors associated with the promoters. A value in the TF column includes the comma-separated list of synonyms for each corresponding TF.

Name	Start	End	Sigma factor	TF
rocA	3880585	3880546	SigL	AhrC
cotV_full	1252094	1252036	SigK	GerE
narG	3830030	3829964	SigA	Fnr
cssR-S2_full	3385630	3385706	SigA	YvqA,CssR
spoIVCA_full	2655030	2654979	SigE	SpoIIID
cgeC_full	2148624	2148494	SigK	GerE
yjcZ-P3_full	1253007	1253082	SigK	GerE
yczA_full	276758	277048	SigA	TrnD-Trp
fnr_full	3832318	3832249	SigA	YpxD,ResD
yurH_full	3343778	3343692	SigA	YunI,PucR
dltA_full	3952033	3952091	SigX	Spo0A
alkA_full	203565	203490	SigA	AdaA
cwlH_full	2649828	2649669	SigK	GerE
ykoL_full	1398081	1398157	SigA	PhoP
gutB_full	667324	667419	SigA	GutR
spoIVCB	2652915	2652962	SigE	SpoIIID
spoIIAA	2445069	2445021	SigH	Spo0A
ybgJ_full	265325	265250	SigA	YcbA
yccC_full	290828	290893	SigA	ScgR,TnrA
spo0A-P2	2518914	2518868	SigH	Spo0A
spoIIGA	1603685	1603752	SigA	Spo0A
rapA_full	1315763	1315848	SigA	ComA,ComAA
glpD_full	1004814	1004895	SigA	GlpP
phoD_full	283918	283989	SigA	PhoP
ymaA_full	1868500	1868577	SigA	YpxD,ResD
ywoE_full	3753910	3753809	SigA	YunI,PucR
thrS_full	2961536	2961251	SigA	TrnI-Thr
yhcY_full	1008562	1008651	SigA	LiaR,YvqC
glpQ_full	233993	233919	SigA	PhoP
mta	3764962	3764921	SigA	YwnD,Mta
ywkA_full	3802337	3802235	SigA	YufM
spoIIE	70458	70517	SigA	Spo0A
htpG_full	4091387	4091330	SigA	
yvgR_full	3434275	3434196	SigA	CysL,YwfK,Ipa-89d
valS_full	2869661	2869375	SigA	TrnSL-Val2
cotJA_full	755829	755882	SigE	SpoIIID
pucH_full	3328746	3328650	SigA	YunI,PucR
sinI-P1	2552271	2552330	SigA	Spo0A
kinC	1518262	1518312	SigA	Spo0A

maeN_full	3244667	3244754	SigA	YufM
pstS_full	2581724	2581652	SigA	PhoP
narK	3833597	3833538	SigA	FNR,Fnr
ftsA-P1_full	1596377	1596453	SigA	WalR,YycF
ykzB_full	1397851	1397916		ScgR,TnrA
rocD	4145588	4145549	SigL	AhrC
guaD_full	1383243	1383164	SigA	YunI,PucR
gabP-P2_full	686667	686600	SigA	ScgR,TnrA
glpT_full	235528	235435	SigA	GlpP
acoA_full	878851	878979	SigL	YfjG,YzcB,AcoR
nrgA_full	3756688	3756762	SigA	ScgR,TnrA
ydfJ_full	589613	589694	SigA	YdfI

Table D-2: The list of repressible promoters that can be repressed by one TF only. Columns show the labels, start and end positions and sigma factors associated with the promoters. A value in the TF column includes the comma-separated list of synonyms for each corresponding TF.

Name	Start	End	Sigma factor	TF
spoVE-P2_full	1590218	1590276	SigE	SpoIIID
exuP1_full	1300358	1300435	SigA	ExuR,YjmH
araR_full	3485603	3485656	SigA	AraC,YvbS,AraR
cydA	3979429	3979382	SigA	GraR,CcpA,AlsA
dhbA_full	3292444	3292363	SigA	YqkL,Fur
ysiA	2918628	2918562	SigA	FadR,YsiA
yknW_full	1503496	1503560	SigW	AbrB,CpsX
hemA	2879249	2879182	SigA	YgaG,PerR
yhcL_full	986899	986971	SigA	CymR,YrzC
fabHA	1208114	1208167	SigA	YlpC,FapR
purA_full	4156857	4156752	SigA	YabI,PurR
ykvW_full	1451189	1451269	SigA	YgaG,PerR
yxzE_full	3982908	3982971	SigW	AbrB,CpsX
spoVG-P1_full	55755	55833	SigH	AbrB,CpsX
feuA_full	183406	183323	SigA	YqkL,Fur
yvqI	3399006	3398947	SigA	AbrB,CpsX
purE_full	698279	698389	SigA	YabI,PurR
dppA	1360319	1360383	SigA	AbrB,CpsX
gapB_full	2968148	2968059	SigA	YqzB,CcpN
abrB-P2	45257	45200	SigA	AbrB,CpsX
mrgA	3383477	3383540	SigA	YgaG,PerR
pckA	3129432	3129505	SigA	YqzB,CcpN
yrrT_full	2788605	2788497	SigA	CymR,YrzC
nadB_full	2849571	2849464	SigA	YrxA,NadR,NiaR
yciA	364181	364238	SigA	YqfV,Zur
spoIID_full	3777844	3777781	SigE	SpoIIID
ydgB	602777	602726	SigK	GerE
cggR_full	3483912	3483799	SigA	CggR,YvbQ
clpP-S1	3546151	3546213	SigA	YacG,CtsR
aprX_full	1862896	1862727	SigA	DinR,LexA
ybaK_full	155965	156093	SigG	DinR,LexA
acuA	3040002	3040068	SigA	GraR,CcpA,AlsA

yhaG_full	1075288	1075143	SigA	MtrB
groES_full	649794	649867	SigA	YqeS,YqxE,HrcA
cysK_full	81690	81770	SigA	CymR,YrzC
yuxH	3259348	3259283	SigA	Spo0A
spo0E	1430587	1430651	SigA	AbrB,CpsX
des_full	2089307	2089372	SigA	DesR,YocG
rapG_full	4140160	4140219	SigA	YvaN,RghR
ywfK_full	3865293	3865222	SigA	CysL,YwfK,Ipa-89d
yxkC_full	3989157	3989217	SigD	ScgR,TnrA
yhfL	1100888	1100959	SigA	FadR,YsiA
hutP	4041405	4041467	SigA	GraR,CcpA,AlsA
treP	850283	850324	SigA	TreR,YfxA
bofA_full	29705	29756	SigE	SpoIIID
lrpC-P1_full	475899	476038	SigA	YdaI,LrpC
nifS_full	2849464	2849571	SigA	YrxA,NadR,NiaR
fur_full	2450378	2450309	SigA	YgaG,PerR
mmgA_full	2514131	2514065	SigE	GraR,CcpA,AlsA
citZ	2982511	2982446	SigA	CcpC,YkuM
abnA	2950187	2950133	SigA	AraC,YvbS,AraR
xpt_full	2320307	2320206	SigA	YabI,PurR
abrB-P1	45269	45214	SigA	AbrB,CpsX
spoVE-P1_full	1590077	1590137	SigE	SpoIIID
ykuF	1477998	1478051	SigA	FadR,YsiA
yxjC_full	4004245	4004156	SigE	GraR,CcpA,AlsA
sigH	116530	116579	SigA	AbrB,CpsX
yvlA_full	3608908	3608825	SigW	AbrB,CpsX
rocR_full	4145663	4145716	SigA	RocR
ctsR_full	101366	101437	SigA	YacG,CtsR
hrcA-S2_full	2629731	2629652	SigA	YqeS,YqxE,HrcA
xsa	2915285	2915239	SigA	AraC,YvbS,AraR
gltC_full	2014693	2014759	SigA	GltC
kinA_full	1469927	1470001	SigH	Spo0A
cotX-Px	1251238	1251191	SigK	SpoIIID
yklA	1380901	1380956	SigA	YkmA,OhrR
resD_full	2417972	2417903	SigA	PhoP
gltR	2725741	2725799	SigA	YrdL,GltR
yqxL_full	2561508	2561450	SigB	DinR,LexA
fapR	1661892	1661946	SigA	YlpC,FapR
glpF	1002307	1002371	SigA	GraR,CcpA,AlsA
infC_full	2953587	2953478	SigA	RplT
P_D-6_full	3636033	3635865	SigD	CodY
dra	4052362	4052293	SigA	DeoR,YxxC
spoVD	1584149	1584192	SigE	SpoIIID
ydjK_full	676113	676240	SigA	IolR
cotH_full	3717240	3717114	SigK	GerE
bglP	4035863	4035814	SigA	GraR,CcpA,AlsA
ahpC_full	4118845	4118911	SigA	YgaG,PerR
spoIIIAA_full	2537697	2537640	SigE	SpoIIID
ywjF	3816599	3816557	SigA	FadR,YsiA
lmrA_full	290788	290720	SigA	Lin-2,YccB,LmrA
kinB	3230002	3230045	SigA	AbrB,CpsX
ycdH	308229	308305	SigA	YqfV,Zur
sigW_full	194782	194861	SigW	AbrB,CpsX

Table D-3: The list of inducible promoters that can be activated by two TFs. Columns show the labels, start and end positions, and sigma factors associated with the promoters. TFs are shown in TF1 and TF2 columns. A value in the TF column includes the comma-separated list of synonyms for each corresponding TF.

Name	Start	End	Sigma factor	TF1	TF2
addB_full	1136202	1136299	SigA	ComK	ComK
comC_full	2865294	2865193	SigA	ComK	ComK
ctaA-Ps_full	1559088	1558940	SigE	YpxD,ResD	YpxD,ResD
bmr	2494587	2494631	SigA	BmrR	YwnD,Mta
sacP_full	3905217	3905117	SigA	Ipa-13r,SacY	Ipa-47d,SacT
pheS_full	2930828	2930551	SigA	TrnD-Phe	TrnB-Phe
sspG_full	3353944	3354045	SigK	GerE	GerE
nasD_full	358294	358206		ScgR,TnrA	YpxD,ResD
ackA_full	3016513	3016385	SigA	GraR,CcpA,AlsA	GraR,CcpA,AlsA
spo0F-P2_full	3810049	3809913	SigH	Spo0A	Spo0A
cotB_full	3716002	3715918	SigK	GerE	GerE
comF-P1_full	3643667	3643575	SigA	ComK	ComK
blt	2716863	2716907	SigA	YwnD,Mta	Bmr2R,BmtR,BltR
tyrS_full	3038211	3037941	SigA	TrnSL-Tyr1	TrnD-Tyr
pucJ_full	3330362	3330480	SigA	ScgR,TnrA	YunI,PucR

Table D-4: The list of repressible promoters that can be repressed by two TFs. Columns show the labels, start and end positions, and sigma factors associated with the promoters. TFs are shown in TF1 and TF2 columns. A value in the TF column includes the comma-separated list of synonyms for each corresponding TF.

Name	Start	End	Sigma factor	TF1	TF2
treP_full	850283	850347	SigA	TreR,YfxA	TreR,YfxA
yjbD-Pm_full	1227524	1227658	SigM	YodB	YgaG,PerR
pta_full	3866432	3866335	SigA	GraR,CcpA,AlsA	GraR,CcpA,AlsA
epr_full	3939434	3939848	SigD	Hpr,ScoC,CatA	Sin,FlaD,SinR
hutP_full	4041405	4041473	SigA	CodY	GraR,CcpA,AlsA
hemA_full	2879249	2879148	SigA	YgaG,PerR	YgaG,PerR
ykuN_full	1486950	1487028	SigA	YqkL,Fur	YqkL,Fur
glnR	1877886	1877938	SigA	GlnR	ScgR,TnrA
abrB-P2_full	45257	45174	SigA	Spo0A	AbrB,CpsX
ftsA-P2_full	1596294	1596392	SigH	AbrB,CpsX	AbrB,CpsX
spoVD_full	1584149	1584193	SigE	SpoIIID	SpoIIID
gltR_full	2725741	2725827	SigA	YrdL,GltR	YrdL,GltR
pyrR_full	1618044	1618215	SigA	PyrR	YabI,PurR
yveK_full	3530075	3529906	SigA	Sin,FlaD,SinR	Sin,FlaD,SinR
perR_full	944399	944458	SigA	YgaG,PerR	YgaG,PerR
spoIVCB_full	2652915	2652984	SigK	GerE	GerE
iolR_full	4084676	4084774	SigA	IolR	IolR
ccpC-P1_full	1485932	1486027	SigA	GraR,CcpA,AlsA	CcpC,YkuM
sda_full	2647226	2647376	SigA	DinR,LexA	DnaA
dppA_full	1360319	1360395	SigA	CodY	AbrB,CpsX
tagA_full	3681225	3681351	SigA	PhoP	PhoP
spo0A_full	2519072	2518953	SigA	Spo0A	Spo0A
xsa_full	2915324	2915239	SigA	AraC,YvbS,AraR	AraC,YvbS,AraR

pbpE_full	3535570	3535486	SigW	AbrB,CpsX	AbrB,CpsX
xynP_full	1886984	1887272	SigA	GraR,CcpA,AlsA	XylR

Table D-5: The list of constitutive SigA promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
ezrA-P1	3031571	3031494	opuA-P2	320909	320956
lepA	2632843	2632774	valS	2869661	2869611
phoA	1018540	1018471	nrgA	3756698	3756762
resA	2421425	2421371	trpE	2377666	2377612
yceC	312022	312068	yusL	3372502	3372410
dnaJ	2625985	2625933	ldh	329676	329723
fbp	4127978	4128028	pstS	2581701	2581652
gtaB	3665509	3665555	ykoL	1398114	1398157
rpmH-TR2	4215536	4215486	arsR		
pdhC	1530383	1530460	recA	1764569	1764624
qcrA	2365199	2365155	galE	3991035	3990987
yhcY	1008592	1008651	nifS	2849485	2849549
amyE	327447	327513	rrnO-P2	9592	9654
glyA	3790521	3790476	rrnD-P1	946379	946441
lonS2	2882863	2882800	opuA-P1	320943	320994
codV	1687120	1687168	spoIIIJ	4214698	4214647
phrE-P1			Site I	52704	52748
licR	3964043	3964002	ctaA	1559035	1558988
clpE-s1	1437966	1437923	mtnW-P1	1426807	1426882
ylaC	1544046	1544096	yurH	3343738	3343692
rrnA-P2	30123	30181	pyrG	3812524	3812472
hrcA-S2	2629731	2629665	pgk	3481685	3481628
hbs-P1	2386042	2385985	yccC	290848	290893
menB	3149751	3149688	yfhR-P5	936868	936912
trnD-Leu2	953138	953197	gntR-P1	4113336	4113398
liaG	3397815	3397765	pbpF	1083734	1083798
malA	889954	890000	maeN	3244703	3244754
comK	1117024	1117092	purT	243832	243873
pssA-P2	247574	247618	alsS	3711440	3711378
ywcJ			gltC	2014693	2014755
ytlI	3008806	3008879	ytmI	3008896	3008822
tagA	3681253	3681299	srfAA	376632	376694
gabT	441504	441560	lmrA	290788	290733
rrnO-P1	9494	9556	secA	3630950	3630898
ydfK	594164	594120	spo0F-P2	3810042	3809979
sigA-P8	2602931	2602969	citR	1021047	1020983
ylxM	1671746	1671807	pheS	2930828	2930786
htpG	4091387	4091342	dnaA-TR3	110	174
rrnB-P2	3178861	3178799	groES	649794	649854
cssR-S2	3385660	3385706	amhX	325308	325250
menE	3148855	3148790	ppiB	2435869	2435806
cwlF-Pa			yjcI	1258242	1258288
phoD	283940	283989	clpX	2886168	2886119
comQ	3257018	3256950	gabP-P1	686638	686589
ytkD	3135531	3135487	ypuE		

sboA	3835968	3836017	phoP-PA6	2978743	2978694
rapH	750886	750931	mtrA	2385430	2385376
nasB	362846	362781	ftsH	76912	76960
aprE	1105677	1105609	ykvW	1451226	1451269
yqdB	2678145	2678205	rrnA-P1	30039	30105
cggR	3483912	3483857	hemZ	1057609	1057654
pbpD	3233844	3233891	gyrA-TR7	6871	6934
xynA	2055387	2055328	tyrS	3038211	3038151
nadB	2849544	2849482	med-B	1206498	1206548
dctP	500081	500130	cspB	984634	984567
purE	698325	698389	trnD-Asn	951403	951453
cymR	2812265	2812207	comGA	2560145	2560080
ureA-P3			pssA-P1	247514	247559
yvbA	3467421	3467377	exuP1	1300358	1300405
yrrT	2788605	2788537	arfM	3830694	3830636
ytrA	3119524	3119478	gabR	441555	441505
citM	834264	834323	rrnD-P2	946476	946538
sda	2647330	2647376	pabB	82788	82837
argC	1194958	1195018	rrnB-P1	3178956	3178892
rbsR	3701330	3701396	pucJ	3330435	3330480
divIB			cssR-S1	3385562	3385633
nfrA	3912286	3912242	ybgJ	265324	265250
degQ	3257355	3257288	addB	1136244	1136299
yhjL	1129703	1129644	ykrV	1425560	1425620
PA,1			ywbI	3933190	3933142
ykrT	1424770	1424702	yaaJ	25764	25807
bglS			pucA	3341098	3341052
yhcL	986906	986971	acsA	3040013	3039944
drm-P2	2448495	2448449	ywfK	3865293	3865251
spo0B	2854661	2854595	spo0A	2519072	2519019
ftsA-P3	1596140	1596205	citG-P1a	3390762	3390706
asd	1745895	1745953	kdgR	2325725	2325678
ykuN	1486953	1487001	sacX	3941899	3941966
rpsD	3035501	3035564	yaaB-TR6		
ybcO	213824	213891	rapG	4140181	4140219
perR	944399	944444	nasA	362854	362914
gutB	667356	667419	des	2089325	2089372
ywkA	3802294	3802235	cdd	2611426	2611373
med-A	1206498	1206548	xylA	1891760	1891821
yfhR-P3	936596	936643	cysK	81690	81738
PA,2			purA	4156815	4156752
comF-P1	3643621	3643575	ssuB	961224	961285
alkA	203554	203490	gltA	2014735	2014671
mtnW-P2	1426823	1426886	ald	3278250	3278304
rapB	3772211	3772158	sdhC	2908871	2908810
yneJ	1924405	1924451	yxeK	4062281	4062215
icd	2981114	2981052	spoIIIE	1752152	1752195
ogt	1421278	1421336	bglC		
ptsH			phoP-P1	2978631	2978588
thrZ			yciC		
sigA-P1	2604095	2604050	citG-P1b	3390742	3390692
pmi			scr	26337	26389

ydjK	676113	676158	flgB	1691170	1691216
sacP	3905217	3905167	dhbA	3292444	3292405
araE	3485608	3485565	guaD	1383212	1383164
rapD	3744278	3744324	rpoB	121665	121709
gabP-P2	686648	686600	yflG	840570	840505
yocH	2093846	2093798	yjbD-P3	1227570	1227628
infC	2953587	2953549	thrS	2961536	2961475
ccpC-P1	1485969	1486022	comC	2865253	2865193
cysH	1630064	1630118	lysC	2911125	2911061
trxA	2913405	2913359	sigX		
ezrA-P2	3031543	3031469	phrI-P1	548269	548338
yveK	3529955	3529906	yhaG	1075288	1075228
licB	3961961	3961916	gapB	2968148	2968091
resD	2417970	2417923	phoP-P2	2978610	2978566
speD	2966972	2966927	yczA	276758	276802
araR	3485603	3485648	drm-P1	2448525	2448467
xynP	1886984	1887041	ftsA-P1	1596400	1596453
rapA	1315788	1315848	hbs-P3	2386076	2386024
opuE-P1	728471	728429	gudB	2403439	2403390
xpt	2320250	2320206	rpoE	3813201	3813138
ywoE	3753857	3753809	glpD	1004814	1004877
lrpC-P1	475990	476038	nusA		
sigM-PA	1030205	1030159	pta	3866401	3866359
glpT	235528	235466	hmp	1372703	1372763
adaA	203559	203625	comEA		
ykrU	1425631	1425562	sinI-P2	2552352	2552397
yvgR	3434240	3434196	gcaD	56276	56331
clpE-s2	1437846	1437804	iolR	4084731	4084774
rpmH-TR1	4215475	4215404	rsbR	519317	519382
ackA	3016434	3016385	yneI	1923955	1924001
spo0A	2519067	2519010	bsrB	2096154	2096106
citB	1926564	1926629	opuBA	3463361	3463307
yaaA-TR5	3139	3190	tuaA		
degS	3646709	3646643	wapA	4030623	4030580
ydfJ	589631	589694	ctsR	101366	101417
yydF	4127722	4127663	mmsA	4084613	4084567
mecA	1228967	1229031	pucH	3328693	3328650
ytlI	3008806	3008879	odhA	2111770	2111707
tagD	3681131	3681086	glpQ	233974	233919
ahpC	4118845	4118891	sipS-T5-1		
yfkJ	861909	861960	ptsG	1459303	1459348
fnr	3832304	3832249	yqxM	2555349	2555304
ymaA	1868523	1868577	ggaA		
htrA	1359389	1359339	sigA-P2	2604068	2604022
phoB-Pv	621567	621519	ispA		
feuA	183406	183354	citA	1020985	1021049
menF			sacB	3535769	3535823
guaA	692563	692626	araA	2949012	2948967
rocR	4145668	4145716	rapC	428763	428802
rncS			nadE	338161	338206
sinR-P3	2552589	2552639	gntZ-P2	4117012	4117079
csbA-PA	3615414	3615371	pdhA	1528191	1528266

ansA	2456920	2456857	ilvB		
speE	3849753	3849689	ccdA	1923086	1923134
tetL	4189221	4189174	pyrR	1618114	1618157
comA	3253517	3253469	gltX	110951	111013
yfmP	812075	812120	lonS1		
aprX	1862775	1862727	sigD	1716433	1716477
fur	2450356	2450309			

Table D-6: The list of SigA promoters that are not included in the previous table. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
sinI-P1	2552271	2552330	ykoL_full	1398081	1398157
bmr	2494587	2494631	mmsA_full	4084681	4084465
glnR	1877886	1877938	pheS_full	2930828	2930551
ysiA	2918628	2918562	gapB_full	2968148	2968059
gltR	2725741	2725799	ahpC_full	4118845	4118911
yklA	1380901	1380956	pucH_full	3328746	3328650
fabHA	1208114	1208167	glpT_full	235528	235435
glpF	1002307	1002371	ykuN_full	1486950	1487028
abrB-P1	45269	45214	pucJ_full	3330362	3330480
sigH	116530	116579	citM_full	834116	834370
bglP	4035863	4035814	gutB_full	667324	667419
hutP	4041405	4041467	bglP_full	4035863	4035760
xsa	2915285	2915239	comF-P1_full	3643667	3643575
treP	850283	850324	ftsA-P1_full	1596377	1596453
yciA	364181	364238	xpt_full	2320307	2320206
mrgA	3383477	3383540	treP_full	850283	850347
ycdH	308229	308305	abrB-P2_full	45257	45174
hemA	2879249	2879182	yccC_full	290828	290893
mta	3764962	3764921	rbsR_full	3701237	3701407
yuxH	3259348	3259283	yhaG_full	1075288	1075143
acuA	3040002	3040068	lrpC-P1_full	475899	476038
dppA	1360319	1360383	nifS_full	2849464	2849571
clpP-S1	3546151	3546213	nrgA_full	3756688	3756762
abnA	2950187	2950133	ackA_full	3016513	3016385
spo0E	1430587	1430651	ybgJ_full	265325	265250
narG	3830030	3829964	glpF_full	1002307	1002388
dra	4052362	4052293	exuP1_full	1300358	1300435
yvqI	3399006	3398947	ccpC-P1_full	1485932	1486027
ywjF	3816599	3816557	araE_full	3485608	3485490
kinB	3230002	3230045	dppA_full	1360319	1360395
kinC	1518262	1518312	tyrS_full	3038211	3037941
abrB-P2	45257	45200	tagD_full	3681268	3680986
spoIIE	70458	70517	phoD_full	283918	283989
yhfL	1100888	1100959	yczA_full	276758	277048
citZ	2982511	2982446	recA_full	1764463	1764624
pckA	3129432	3129505	rapG_full	4140160	4140219
spoIIGA	1603685	1603752	maeN_full	3244667	3244754
cydA	3979429	3979382	ydjK_full	676113	676240
fapR	1661892	1661946	feuA_full	183406	183323
narK	3833597	3833538	nasA_full	362838	362914

ykuF	1477998	1478051	fur_full	2450378	2450309
blt	2716863	2716907	srfAA_full	376515	376726
rapA_full	1315763	1315848	glpQ_full	233993	233919
purA_full	4156857	4156752	yurH_full	3343778	3343692
hutP_full	4041405	4041473	glnR_full	1877877	1877938
citZ_full	2982511	2982367	ydfJ_full	589613	589694
sda_full	2647226	2647376	ywfK_full	3865293	3865222
iolR_full	4084676	4084774	araR_full	3485603	3485656
cysK_full	81690	81770	guaD_full	1383243	1383164
nadB_full	2849571	2849464	infC_full	2953587	2953478
xsa_full	2915324	2915239	sinI-P1_full	2552238	2552369
gabP-P2_full	686667	686600	yhcY_full	1008562	1008651
phoB-Pv_full	621587	621519	tagA_full	3681225	3681351
alkA_full	203565	203490	des_full	2089307	2089372
ywoE_full	3753910	3753809	thrS_full	2961536	2961251
ykvW_full	1451189	1451269	xynP_full	1886984	1887272
yrrT_full	2788605	2788497	rapH_full	750779	750931
hemA_full	2879249	2879148	pyrR_full	1618044	1618215
yvqI_full	3399035	3398947	comC_full	2865294	2865193
rocR_full	4145663	4145716	lmrA_full	290788	290720
spo0A_full	2519072	2518953	dhbA_full	3292444	3292363
phoP-PA6_full	2978755	2978561	nasB_full	362856	362781
cssR-S2_full	3385630	3385706	spoIIE_full	70375	70517
gltA_full	2014778	2014663	glpD_full	1004814	1004895
ymaA_full	1868500	1868577	pstS_full	2581724	2581652
gltR_full	2725741	2725827	yveK_full	3530075	3529906
purE_full	698279	698389	aprX_full	1862896	1862727
clpE-s2_full	1437959	1437743	gltC_full	2014693	2014759
hrcA-S2_full	2629731	2629652	htpG_full	4091387	4091330
cggR_full	3483912	3483799	sacX_full	3941835	3942007
sacP_full	3905217	3905117	yhcL_full	986899	986971
ctsR_full	101366	101437	spoIIGA_full	1603634	1603752
fnr_full	3832318	3832249	pta_full	3866432	3866335
valS_full	2869661	2869375	cydA_full	3979497	3979219
perR_full	944399	944458	ywkA_full	3802337	3802235
addB_full	1136202	1136299	yvgR_full	3434275	3434196
resD_full	2417972	2417903	groES_full	649794	649867

Table D-7: The list of SigB promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
ydaP	488772	488814	relA	2822846	2822803
ydhK	624374	624415	yvgO		
ywmE	3774621	3774573	opuE-P2	728482	728438
ydbP	508311	508267	ydaT	493501	493461
ytxG-PB	3048082	3048037	yqxL_full	2561508	2561450
yfhK	928733	928775	csbA-PB	3615469	3615427
yycD	4158944	4158985	nadE	338213	338263
csbC	4087895	4087949	ywtG	3694003	3693951
yvrE	3406607	3406567	rsbV	522016	522058
phoP-PB1	2978583	2978538	yqhA		

ydaS	492974	492933	yotK	2153457	2153415
csbD	3770367	3770314	yflA	844711	844752
yvyD-P1	3631650	3631602	yjgB	1285505	1285465
gsiB	494441	494482	yoxC	2019368	2019408
yjbC-PB	1226864	1226911	ctc	58708	58749
yfkJ	861948	861992	katE-S1		
yxkO	3973339	3973298	yqhQ		
gspA	3945482	3945444	yhdN	1030207	1030246
ykzA	1381943	1381998	aldY	3986362	3986402
trxA	2913621	2913580	yacL	108559	108600
yoxA	2000931	2000891	sigA-P7	2604107	2604059
yrvD	2826232	2826191	yocK	2097678	2097629
yhdF	1022165	1022206	ypuB	2434409	2434368
gtaB	3665544	3665586	ctsR	101314	101365
ykgA	1371943	1371901	ydaD	471639	471683
csbX			yqgZ	2564482	2564427
cdd	2611426	2611373	dps-PB	3136735	3136692
yheK			yqxL	2561508	2561468
bmrU	2493595	2493638	clpP-S2	3546087	3546150
ycnH	442756	442796	ydbD	496627	496582
ysnF			katX		
ywjC	3818832	3818883	ydaG	473739	473778
ydfO	596815	596864	csbB	930710	930755
ytkL	3010664	3010623			

Table D-8: The list of SigD promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
yjbJ	1235803	1235751	degR	2308444	2308396
yscB	2953735	2953791	yvyF	3641133	3641089
cheV	1473532	1473580	P_D-6_full	3636033	3635865
P_D-6	3636033	3635988	yoaH	2031238	2031192
hemAT	1114017	1113969	mcpC	1463571	1463614
ybdO	224994	225048	PD	3663151	3663103
yxkC	3989157	3989206	mcpA	3208221	3208170
mcpB	3212497	3212454	nfrA	3912286	3912241
dltA	3951975	3952022	yjfB	1283391	1283342
flgB	1691011	1691062	P_D-8	3634819	3634775
yxkC_full	3989157	3989217	lytF	1013344	1013291
tlpC	374560	374513	epr_full	3939434	3939848
epr	3939798	3939848	tlpA	3210337	3210289
P_D-2	1435321	1435275	yjcP	1266556	1266603
tlpB	3206122	3206076	ylqB	1671718	1671669
yfmT	807027	807081	lytD	3687547	3687498
sigA-P6	2601574	2601522			

Table D-9: The list of SigE promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
yhaX	1056632	1056679	ylbO		
yfhS	937061	937013	ydcA	515689	515643
yxjC_full	4004245	4004156	phoB-Ps	621639	621591
ykvI	1438012	1438054	spoIVFA	2857702	2857651
yxjC	4004245	4004204	yaaH	25224	25182
spoIIM	2451119	2451078	spoIVA-P2		
ytxC	2962493	2962447	bofA_full	29705	29756
yndA			yheC		
cwlJ	282409	282448	yqfZ	2588636	2588685
spoVE-P2	1590218	1590260	yknT	1495430	1495383
yteV	3078632	3078595	ydhD		
spoIID_full	3777844	3777781	yfnE	801172	801214
cotE-P1	1774933	1774975	bvx	3171697	3171657
spoIVCB	2652915	2652962	yjcA-P1	1252104	1252154
ycgF	334563	334608	ywdL	3893279	3893331
yitC	1172580	1172632	spoVD_full	1584149	1584193
usd	3748900	3748858	spoIIIAG	2534062	2534009
bdbD			yncD		
yhcO	989634	989683	spoVD	1584149	1584192
spoVK-P1	1874033	1874075	safA	2845936	2845891
ylbJ	1571940	1571893	ysnD		
spoVE-P1_full	1590077	1590137	nucB	2652859	2652815
yqfC			spoIVCA	2655028	2654979
exuP2	1303322	1303370	ctaA-Ps_full	1559088	1558940
yybI	4177753	4177708	yhxC		
yvjB			asnO	1157142	1157186
yuzC	3257629	3257678	yhbH	975104	975149
G4	2047717	2047675	ypqA	2337521	2337569
gerM	2903166	2903120	mmgA	2514131	2514089
yjbX	1248609	1248649	spoVE-P2_full	1590218	1590276
spoVID	2872829	2872786	yyaD	4204515	4204471
ytvI	2983093	2983139	spoIIP	2634504	2634461
spoIVCA_full	2655030	2654979	yngJ	1957421	1957380
cwlD	156549	156597	spoVE-P1	1590077	1590125
phoB-Ps_full	621639	621529	spoIIIAA	2537681	2537640
ydcC	516075	516123	ytrH	2994702	2994740
prkA	973070	973114	yloB		
ybaN	160616	160573	ctaA-Ps	1559029	1558977
spoVR	1015585	1015627	spoVB	2829486	2829528
ykvU	1449186	1449232	phoP-PE2	2978603	2978554
ypjB	2362399	2362353	yrrR		
yjfA			mmgA_full	2514131	2514065
yjdH			spoIVA-P1	2387863	2387822
yunB	3323283	3323244	yabP	68139	68183
dacB	2424594	2424552	cotJA_full	755829	755882
yjbE			bofA	29705	29745
spoIIIAA_full	2537697	2537640	yhjR		
cotJA	755839	755882	spoVM	1655319	1655362
spoIID	3777823	3777781			

Table D-10: The list of SigF promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
ywnJ	3759712	3759673	spoIVB	2520477	2520436
yrrR	2791492	2791441	gerAA	3390714	3390754
yhfM	1103010	1102971	bofC		
yyaC	4204797	4204840	lonB	2884717	2884669
yhfW	1115648	1115607	gpr		
dacF	2446325	2446286	sspN	1930210	1930251
yhcM			ywhE	3849749	3849794
spoIIR	3795422	3795382	spoIIQ	3760607	3760541
yphA			rsfA		
yqhQ			yuiC	3299291	3299252
ytkD	3135585	3135534	yqzG	2555466	2555505
katX	3964935	3964976	yfhD		
ylbB	1565716	1565762	csfC		
yhcN	988958	989001	ytfI	3019512	3019473
sigG	1605560	1605612	csfB		

Table D-11: The list of SigG promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
sspK	928358	928312	yvaB	3446150	3446098
yrrD	2806234	2806181	csgA	227992	228045
gerAA	3390714	3390754	ydfS	600919	600969
ykvV	1450580	1450621	yfjS		
sspI-P2	2931620	2931666	sspN	1930210	1930250
splB-P3	1461705	1461749	yxeD	4066138	4066189
yuzA	3224794	3224843	spoVT	63979	64018
sspC	2156160	2156205	yveA	3539098	3539142
ybaK_full	155965	156093	yhcV	997120	997159
gpr	2635674	2635633	yfkD		
sigG	1605560	1605618	ybaK	156047	156093
yoaR	2042896	2042856	exoA		
sspM	2339615	2339656	yitG	1177313	1177273
sspJ	3421674	3421632	bofC	2837488	2837442
yvdQ	3542761	3542712	ydfR	600912	600844
yraG	2752741	2752781	sspB	1050390	1050344
sspE-P7	937845	937894	glcU	444390	444436
gerD	159139	159081	ywhE	3849749	3849794
yndD	1907313	1907358	yhcQ	991331	991286
sspO	1926505	1926464	yqfS	2594269	2594225
spoVAA	2443368	2443311	sspD	1414072	1414026
yvdP			rsfA		
sspH	885573	885614	gerBA	3688733	3688786
splA-P1	1461397	1461440	ylaJ		
ypzA	2308444	2308482	dacF	2446325	2446286
sspL	2310807	2310846	sspF	53083	53133
yvjB	3623879	3623827	sleB	2400144	2400098
yteA	3153940	3153986	sspA	3025710	3025660

yhcN	988958	989004	yozQ	2028777	2028824
yckD	369711	369752	gerKA	420047	420100
sspI-P1	2931599	2931645	yisY		
spoIVB	2520477	2520436			

Table D-12: The list of SigH promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
yoxA	2000918	2000874	phrC-P2	429840	429882
kinA	1469927	1469979	kinA_full	1469927	1470001
hbs-P2	2385892	2385836	spo0A-P2	2518914	2518868
spo0M	954240	954199	phrE-P2		
phrE-P3			citG-P2	3390486	3390439
spoIIAA_full	2445122	2445021	ureA-P2		
spoVG-P1	55787	55833	minC	2859320	2859279
phrG	4141080	4141121	ftsA-P2_full	1596294	1596392
phrI-P2	548355	548396	ftsA-P2	1596294	1596343
yvyD-P2	3631639	3631596	phrK	2062988	2063029
spoIIAA	2445069	2445021	sigA-P3		
phrF	3847007	3847047	ytxG-PH	3048135	3048086
ywkC			spo0F-P2_full	3810049	3809913
spoVG-P1_full	55755	55833	spo0F-P2	3810009	3809958
spoVS	1769772	1769813	cwlF-Ph	1018951	1018994
sigA-P4					

Table D-13: The list of SigK promoters. Columns show the labels, start and end positions of these promoters.

Name	Start	End	Name	Start	End
cotB	3715964	3715918	yjcZ-P3	1253038	1253082
yitC	1172580	1172632	spoIVCB	2652915	2652962
yoaN	2038846	2038795	cotX-Px_full	1251265	1251191
ytlA	3132286	3132329	ytaA		
yhcO	989634	989683	cotM-P1		
gerPA			ykuD	1477070	1477030
ydgB	602777	602726	spoVK-P2	1874144	1874190
ymaG			yabG	51529	51574
cgeA	2148603	2148650	cotD_full	2333121	2333038
yfhP-P2	935593	935555	yngK	1959635	1959593
cwlH	2649709	2649669	cotY	1250587	1250540
sspG	3354003	3354045	spoIVCB_full	2652915	2652984
cotT	1280949	1280894	cotV	1252083	1252036
yodH	2133401	2133438	yqfQ		
cotA	685080	685033	ytcC		
yitB	1172550	1172506	yozR	2123939	2124020
cotB_full	3716002	3715918	ylbD	1567583	1567633
cotV_full	1252094	1252036	yfhP-P1	935604	935567
cotC_full	1905362	1905216	ysnD	2897725	2897770
cotE-P2	1774999	1775046	cwlC	1873633	1873591

yxeE			cotH	3717240	3717181
cotX-Px	1251238	1251191	yjcC	1253314	1253369
cotF	4167039	4167083	cotSA	3161972	3161928
spoVFA	1744274	1744324	yobW	2084153	2084196
cwlH_full	2649828	2649669	yhjR	1136208	1136154
ypqA	2337521	2337560	cgeC_full	2148624	2148494
yfnH	798321	798365	yndL	1915081	1915132
gerE	2905055	2905008	yjcZ-P3_full	1253007	1253082
yrkC	2715561	2715511	cotD	2333085	2333038
yjcB-P2			ywrJ		
cotC	1905262	1905216	cgeC	2148541	2148494
cotG	3717149	3717193	yfnE	801172	801214
spsA	3893198	3893155	ftsY-PK		
cotH_full	3717240	3717114	ykvP		
sspG_full	3353944	3354045	ycsK		
tgl	3212535	3212575			

Table D-14: The list of SigW promoters. Columns show the labels and start and end positions of the promoters.

Name	Start	End	Name	Start	End
pbpE	3535526	3535486	yvlA	3608867	3608825
pspA	671169	671213	pbpE_full	3535570	3535486
ywnJ	3759671	3759619	divIC	69018	69059
ydbS	512749	512791	yrhH	2778545	2778501
yoaF	2027445	2027483	yqeZ	2619857	2619819
yeaA	683427	683385	yjoB	1314385	1314427
yobJ	2071153	2071111	yndN	1916595	1916639
yjbC-P2	1226822	1226865	xpaC	35765	35807
ywbN	3928076	3928032	yceC	312040	312082
abh	1517766	1517807	yknW_full	1503496	1503560
ywrE	3718738	3718778	sigW_full	194782	194861
yxjI	3997154	3997196	sppA	3021112	3021072
yuaF	3182624	3182582	ybfO	249911	249949
yfhL	929352	929391	yxzE	3982908	3982950
pbpX	1765780	1765818	yqjL	2476904	2476941
ywaC	3950662	3950621	sigW	194782	194824
yknW	1503518	1503560	yxzE_full	3982908	3982971
yoaG	2028707	2028669	ysdB	2951419	2951461
ythP	3072165	3072128	yvlA_full	3608908	3608825
yozO	2099997	2099959			

Table D-15: The list of SigX promoters. Columns show the labels and start and end positions of these promoters.

Name	Start	End	Name	Start	End
abh	1517766	1517807	ywoA	3758484	3758527
divIC	69018	69059	yjbC-P2	1226822	1226865
PX	3663161	3663202	yrhH	2778545	2778501
yqjL	2476904	2476941	ywnJ	3759671	3759619
rapD	3744267	3744308	sigX	2415265	2415224

pbpX	1765780	1765818	pssA-P3	247672	247718
dltA_full	3952033	3952091	csbB	930702	930743
ywbN	3928076	3928032	dltA	3952033	3952082

Table D-16: The list of SigI, SigL, SigM, SihY and YlaC promoters. Columns show the labels, start and end positions, and associated sigma factors with the promoters.

Name	Start	End	Sigma factor	Name	Start	End	Sigma factor
sigI-S2	1411729	1411782	SigI	yraA			SigM
levD	2762897	2762858	SigL	ysxA			SigM
levD_full	2763010	2762858	SigL	yrhJ			SigM
ptb	2504726	2504685	SigL	yjbD-Pm	1227524	1227583	SigM
rocA_full	3880730	3880546	SigL	ywoA	3758484	3758527	SigM
rocD_full	4145683	4145549	SigL	yqjL			SigM
rocD	4145588	4145549	SigL	sigM-PM	1030126	1030083	SigM
rocG	3882135	3882094	SigL	yjbD-Pm_full	1227524	1227658	SigM
rocA	3880585	3880546	SigL	ybgB	258455	258495	SigY
acoA	878940	878979	SigL	sigY	3970918	3970878	SigY
acoA_full	878851	878979	SigL	ylaA	1541817	1541871	YlaC

Table D-17: The list of repressor binding sites. Columns show the labels and start and end positions of the operators. Operators are identified by the names of repressors and promoters which the operators are part of on the genome. Operator names have the format ‘Repressor name’_‘Promoter name’.

Name	Start	End	Name	Start	End
CcpA_araA	2948922	2948889	CtsR_clpE-P1	1437959	1437936
DnaA_dnaA	1843	1871	Hpr_aprE	1105950	1105921
AbrB_pbpE	3535570	3535546	AraR_araA		
AhrC_argC	1194970	1194986	XylR_xylA	1891802	1891841
Fur_yfhC	923530	923579	PhoP_tagA	3681375	3681415
AbrB_sigH	116540	116579	LexA_yorB	2188195	2188172
Fur_yfmC	826791	826742	AraR_araE	3485534	3485510
DnaA_dnaA	246	263	Fur_yxeB	4067117	4067166
SinR_yveK	3529985	3529960	GerE_spoIVCB	2652945	2652975
PerR_perR	944427	944452	LexA_vpr	3907649	3907672
AbrB_comK	1116992	1117111	MtrB_yhaG	1075217	1075143
GlnR_ureA-P3			YdiH_ldh	329676	329723
IolR_mmsA	4084681	4084619	PurR_xpt	2320307	2320255
PerR_yjbD	1227579	1227658	SpoIIID_spoVE-P1	1590079	1590137
Spo0A_kinA	1469981	1470001	TnrA_ywlF	3791802	3791786
PurR_purA	4156857	4156804	CcpA_acuA	3040018	3040031
IolR_iolR	4084730	4084751	CcpA_hutP	4041452	4041465
PerR_mrgA	3383477	3383530	Fur_yfkM	859408	859457
GlnR_nasA	362838	362856	LexA_uvrB	3615027	3615004
BirA_bioW	3095544	3095505	DnaA_dnaA	186	219
PerR_perR	944436	944458	CcpA_phoP-PA6	2978693	2978664
DegU_site I	4030625	4030602	MtrB_trpE	2377584	2377529

DeoR_dra	4052362	4052324	PurR_purR	54380	54432
AbrB_sigW	194835	194861	ExuR_exuP1	1300392	1300435
AhrC_argC	1194941	1194958	SinR_epr	3939434	3939466
Fur_fhuD	3418409	3418458	CcpA_xynP	1887249	1887272
LexA_yolD	2272034	2272011	SpoIIID_bofA	29797	29817
Fur_yuiI	3293433	3293384	DnaA_dnaA	58	77
DegU_site II	4030579	4030563	LexA_yqzH	2465838	2465861
Fur_dhbA	3292412	3292363	GlnR_glnR	1877900	1877916
Spo0A_spo0A-Pv	2518987	2518953	AbrB_aprE	1105685	1105613
CcpA_yxkJ	3979777	3979799	Fur_ydbN	506676	506627
LexA_yolC	2272041	2272064	CcpA_pta	3866432	3866398
YrzC_yhcL	986899	986959	TnrA_glnR	1877900	1877916
CcpA_citM	834348	834370	CcpA_exuP1	1301623	1301645
CcpC_citB	1926572	1926590	SpoIIID_spoVE-P2	1590218	1590276
CodY_comK	1116996	1117060	LexA_yqjW	2465833	2465810
CodY_hag	3635924	3635865	LexA_yqzH	2465810	2465833
LexA_dinC	3683335	3683358	SpoIIID_spoVD	1584177	1584193
SpoIIID_spoIID	3777844	3777784	CcpA_gntR	4113520	4113542
Fur_yclN	432263	432312	YlpC_fapR	1661892	1661954
Xre_xre-O2	1321771	1321794	DnaA_dnaA	93	121
CodY_dppA	1360358	1360395	Fur_ykuN	1486987	1487028
AbrB_yvqI	3399006	3398956	CcpA_yxjC	4004178	4004156
PerR_ahpC	4118850	4118911	YrzC_ssuB	961211	961270
SpoIIID_cotD	2332989	2332964	CcpA_xylA	1891935	1891964
LexA_ruvA	2836853	2836830	PurR_nusB	2529815	2529763
AraR_araA	2948935	2948915	LexA_yhjD	1121437	1121414
LexA_yolD	2272064	2272041	DnaA_dnaA	24	50
AbrB_yknW	1503496	1503541	PerR_hemA	2879189	2879148
YvbA_yvbA	3467421	3467346	YlpC_yhdO	1031282	1031343
YlpC_fabI	1247680	1247740	GlnR_glnR	1877877	1877893
LrpC_lrpC	475899	475913	AbrB_abrB-P1,P2	45250	45221
CcpA_glpF	1002317	1002335	SinR_yveK	3530075	3530036
AraR_abnA	2950187	2950159	CcpA_ydhO	627939	627961
LexA_yolC	2272011	2272034	AbrB_kinB	3230005	3230033
CcpA_kdgA	2323248	2323226	YrzC_yxeK		
DnaA_dnaA	11	37	TnrA_pel	827950	827966
LexA_sda	2647325	2647348	Fur_yoaJ	2033688	2033639
CcpA_acsA	3039924	3039911	YdiH_ywcJ		
Fur_ycgT	352789	352829	CcpA_treP	850682	850704
CcpA_msmX	3985268	3985246	PurR_glyA	3790600	3790541
PurR_purE	698279	698342	HrcA_hrcA	2629678	2629652
YdiH_cydA	3979260	3979219	PurR_pbuG	694338	694390
CcpA_sigL	3513250	3513217	IolR_iolR	4084676	4084727
CcpA_acoA	879432	879454	SpoIIID_cotD	2333082	2333063
SpoIIID_spoIIIAA	2537697	2537643	PerR_katA	961069	961012
LexA_recA	1764554	1764577	IolR_ydjK	676152	676240
TnrA_ywdI	3896214	3896198	CcpA_iolB	4082181	4082159
LexA_yqhB	2561450	2561473	HtrA_htrA	1359452	1359340
LexA_yqxL	2561473	2561450	GerE_ydgB	602777	602754
CcpA_mmsA	4084492	4084465	CtsR_clpP	3546165	3546181
YkmA_yklA	1380924	1380953	CcpA_yobO	2077380	2077402
LexA_yozM	2064927	2064950	LexA_yneA	1918376	1918399
SpoIIID_bofA	29727	29756	YrzC_cysK	81721	81770

LexA_yqjW	2465861	2465838	CcpC_ccpC-P1	1486003	1486027
CtsR_clpE-P2	1437767	1437743	GntR_gntR	4113373	4113394
YsiA_yusL	3372502	3372410	LmrA_yxaG	4107334	4107265
CodY_srfAA	376640	376726	CcpC_citZ	2982474	2982459
CcpA_malA	889989	890015	Zur_yciC	366024	366043
LexA_yhaO	1064778	1064801	RplT_infC	2953532	2953478
Zur_yciA	364211	364238	LexA_yneA	1918310	1918333
LexA_yhjE	1121414	1121437	YrzC_yrrT	2788546	2788497
YrzC_ytmI	3008846	3008805	CcpA_bglP	4035863	4035814
SpoIIID_spoVD	1584153	1584171	GltR_gltR	2725793	2725827
Fur_ykuN	1486950	1486991	CcpA_ccpC-P2	1485932	1485953
CcpA_rbsR	3701368	3701401	TreR_treP	850316	850347
YdiH_cydA	3979292	3979251	CcpC_citB	1926537	1926560
LexA_ydgG	608801	608824	CcpA_dra	4052243	4052221
TnrA_yttA	3108551	3108567	SpoIIID_cotD	2333121	2333101
YrxA_nifS	2849464	2849571	LexA_pcrA	719330	719353
PurR_guaC	3302981	3303033	YrzC_ydbM	505088	505151
LexA_yhaZ	1056273	1056250	YlpC_fabHA	1208114	1208167
TnrA_yycC	4159511	4159527	LexA_lexA	1918399	1918376
PerR_hemA	2879222	2879194	LexA_yneA	1918339	1918362
YsiA_ywjF	3816599	3816557	YodB_yjbD	1227579	1227628
Hpr_sinI-P1	2552312	2552337	CcpA_hutP	4041653	4041666
YdiH_ldh	329707	329760	GerE_cotH	3717131	3717114
HrcA_groES	649841	649867	Hpr_nprE	1541661	1541639
Fur_yfiZ	920339	920388	CcpA_galT	3920424	3920402
GerE_cotM			PhoP_tagA	3681322	3681351
CtsR_clpE-P2	1437813	1437790	LexA_dinB	608824	608801
AbrB_ftsA-P2	1596369	1596392	LexA_ykvR	1447123	1447145
YsiA_ysiA	2918628	2918562	YlpC_fabHB		
SinR_yqxM	2555311	2555281	YdiH_cydA	3979327	3979286
ComK_rok	1493661	1493677	CcpA_dctP	500081	500130
SinR_aprE	1105891	1105846	LexA_parE	1933253	1933276
YwfK_ywfK	3865251	3865222	TnrA_yodF	2130116	2130132
Zur_ycdH	308278	308297	DnaA_dnaA		
YvaN_rapH	750852	750882	Xre_xre-O4	1321736	1321712
CtsR_clpE-P1	1437888	1437865	LexA_yozL	2064920	2064897
YrzC_ytlI	3008800	3008850	LexA_lexA	1918362	1918339
CcpA_araB	2946696	2946674	LexA_ypuD	2432256	2432233
AbrB_pbpE	3535538	3535513	CcpA_acoR	883687	883712
CcpA_bglS	4012603	4012590	Fur_ywbL	3930601	3930552
YrxA_nadB	2849571	2849464	PyrR_pyrP		
LmrA_lmrA	290761	290720	PhoP_tagD	3681013	3680986
CcpA_cydA	3979427	3979398	YsiA_ykuF	1477998	1478060
AbrB_dppA	1360349	1360382	Fur_ybbB	185067	185018
GltC_gltC	2014745	2014759	Hpr_nprE	1541742	1541717
AbrB_spoVG-P1	55755	55824	AraR_OR-x2	2915285	2915254
CcpA_araE	3485503	3485490	CcpA_levD	2762914	2762900
GltR_gltR	2725781	2725795	Spo0A_yuxH	3259348	3259283
IolR_mmsA	4084579	4084558	CcpA_mmgA	2514078	2514065
TreR_treP	850283	850316	LexA_yozM	2064897	2064920
Fur_yfiZ	920404	920451	AbrB_yxzE	3982930	3982971
LexA_lexA	1918333	1918310	CtsR_clpE-P1	1437932	1437908
SinR_yqxM	2555387	2555359	PhoP_tagD	3681174	3681114

CggR_cggR	3483844	3483799	YqzB_pckA	3129440	3129502
Spo0A_abrB-P2	45201	45174	CodY_ilvB		
PerR_ykvW	1451189	1451239	Hpr_epr	3939509	3939537
YocG_des	2089307	2089323	AraR_araR	3485637	3485656
YvaN_rapH	750779	750808	LexA_xkdA		
Spo0A_spo0A-Pv	2519029	2519005	YdiH_alsS	3711415	3711380
Hpr_aprE	1105661	1105640	GerE_spoIVCB	2652953	2652984
Fur_yusV	3380121	3380074	TnrA_alsT	1938864	1938880
CcpA_citZ	2982384	2982367	ArsR_arsR	2657723	2657706
PhoP_tagA	3681225	3681310	CcpA_licB	3961961	3961916
LexA_aprX	1862896	1862873	LexA_ydiO	655154	655177
Fur_feuA	183372	183323	Fur_ywjA	3821543	3821494
AbrB_spo0E	1430595	1430629	PyrR_pyrB	1620326	1620385
Hpr_aprE	1105918	1105893	MtrB_pabA	84236	84287
XylR_xynP	1887028	1887065	LexA_uvrC	2912951	2912928
AhrC_argC	1195117	1195134	DnaA_sda	2647226	2647331
LexA_hbs	2386106	2386083	MtrB_ycbK	277316	277374
LexA_yozL	2064950	2064927	LexA_ybaK	155965	155988
Fur_yusV	3380056	3380007	PyrR_pyrR	1618156	1618215
PhoP_resD	2417972	2417903	YsiA_yhfL	1100888	1100969
Xre_xre-O1	1321804	1321831	TnrA_yxkC	3989201	3989217
CodY_hutP	4041448	4041473	CcpA_amyE	327490	327513
PurR_ytiP	3068849	3068902	Fur_yhfQ	1107640	1107689
LexA_dinC	3683365	3683388	DnaA_dnaA	1767	1837
Xre_xre-O3	1321765	1321738	CcpA_pta	3866374	3866335
RocR_rocR	4145663	4145683	GlnR_ureA-P3		
PurR_pyrR	1618044	1618109	Hpr_sinI-P1	2552238	2552267
CcpC_citZ	2982443	2982420	Fur_ydhU		
Hpr_aprE	1105705	1105685	TnrA_gltA	2014688	2014663
AraR_araE	3485575	3485556	YvaN_rapG	4140160	4140188
PerR_fur	2450378	2450346	YqzB_gapB	2968140	2968059
SpoIIID_cotX	1251230	1251201	AbrB_ftsA-P2	1596335	1596353
PhoP_tagD	3680970	3680924	ComK_rok	1493641	1493661
LexA_dnaE	2994696	2994673	CtsR_ctsR	101409	101437
Fur_nasE	355782	355733	GlnR_nasB	362856	362838
AraR_OR-x1	2915324	2915293	TnrA_glnR	1877877	1877893
CcpA_lcfA	2919920	2919898	TnrA_ilvB		
SpoIIID_cotC	1905307	1905237	AbrB_sinI-P1	2552347	2552369
AbrB_yvlA	3608908	3608884			

Table D-18: The list of activator binding sites. Columns show the labels and start and end positions of the operators. Operators are identified by the names of activators and promoters which the operators are part of on the genome. Operator names have the format ‘activator name’_‘promoter name’.

Name	Start	End	Name	Start	End
GlpP_glpT	235457	235435	ComK_rapH	750853	750886
ComA_srfAA	376560	376597	ComK_comF-P1	3643667	3643638
YvqC_yhcY	1008562	1008591	TrnD-Tyr_tyrZ	3947078	3947094
TnrA_nasD	358266	358250	Spo0A_kinC	1518277	1518306
CitT_citM	834227	834277	ComA_rapC-P1	428727	428805
GerE_cotC	1905362	1905347	GerE_cotX	1251265	1251245

ResD_hmp	1372674	1372713	TrnSL-Tyr1_tyrS	3037957	3037941
GlpP_glpD	1004874	1004895	Hpr_yclF	417769	417741
GltR_gltA	2014759	2014745	ComK_comEA		
GerE_cotX	1251245	1251227	PhoP_yjdB	1270167	1270144
TrnD-Tyr_tyrS	3037957	3037941	ResD_A1	1559161	1559191
AcoR_acoA	878851	878914	ComK_addB	1136225	1136251
TnrA_yccC	290828	290851	YufM_ywkA	3802337	3802275
ComK_comC	2865265	2865236	SpoIIID_spoIVCB	2653062	2653086
GltC_gltA	2014778	2014669	PucR_ureA-P4		
BkdR_ptb	2504785	2504752	YwfK_yvgR	3434275	3434250
PhoP_glpQ	233993	233944	PucR_yurH	3343778	3343743
PucR_pucJ	3330379	3330407	ResD_ymaA	1868500	1868525
YycF_ftsA-P1	1596377	1596409	DegU_comK	1116971	1117005
ComK_yneB	1918618	1918640	YufM_maeN	3244667	3244720
YtlI_ytmI	3008917	3008872	TrnSL-Val2_valS	2869417	2869375
PhoP_phoP	2978672	2978619	ComK_yneB	1918587	1918609
ResD_nasD	358294	358251	ComK_comGA	2560165	2560128
YycF_tagD	3681268	3681238	SpoIIID_spoIVCB	2652918	2652938
Spo0A_spo0F-P2	3810049	3810012	Spo0A_spoIIAA	2445076	2445058
PhoP_phoD	283918	283967	PucR_guaD	1383243	1383208
GerE_ftsY-PK			LevR_levD	2762954	2762931
RocR_rocG	3880730	3880707	DegU_aprE	1105704	1105645
CcpA_ackA	3016513	3016499	HxlR_hxlA		
TnrA_glnQ	2801879	2801895	ComK_yhjB	1120958	1120937
ComA_pel	827869	827884	FNR_arfM	3830694	3830671
TrnS-Leu2_ilvB	2897138	2897120	Spo0A_spo0F-P2	3809955	3809913
Spo0A_spoIIGA	1603634	1603648	AhrC_rocD	4145571	4145553
Spo0A_spo0A-Ps	2518889	2518876	YycF_yocH	2093882	2093853
ComK_yvrP	3415393	3415373	TnrA_gabP-P2	686667	686651
Spo0A_spoIIE	70427	70443	GerE_cotY	1250592	1250576
ComK_recA	1764463	1764483	SacT_sacX	3941965	3942007
TrnD-Gly_glyQ	2608771	2608746	LicT_bglP	4035792	4035760
PhoP_pstS	2581724	2581674	RocR_rocA	3880686	3880658
GerE_sspG	3353980	3354012	TrnS-Leu2_leuS	3105197	3105180
AdaA_alkA	203565	203520	GerE_cgeA	2148574	2148605
TrnI-Thr_thrZ			ComK_yvrP	3415429	3415410
GerE_cotB	3715982	3715962	AbrB_hutP	4041649	4041671
TnrA_nasA	362838	362856	TnrA_oppA	1219618	1219634
ComK_nucA	372710	372686	Spo0A_spoIIE	70406	70422
TnrA_ykzB	1397851	1397867	GerE_yjcZ-P3	1253007	1253052
PhoP_ykoL	1398081	1398140	TrnI-Thr_thrS	2961285	2961251
ComA_rapA	1315763	1315802	Spo0A_spoIIAA	2445122	2445091
ComK_thdF	4212969	4212948	GerE_cgeA	2148498	2148529
GerE_cotV	1252094	1252073	TrnI-Thr_thrZ	3856284	3856221
Spo0A_ybcO	213824	213891	ComK_smf	1682503	1682520
ComK_yyaF	4201054	4201037	FNR_narG	3830030	3830001
Mta_bmr	2494593	2494614	TrnD-Ser_serS	20782	20805
Hpr_yclF	417797	417769	Spo0A_dltA	3952071	3952091
Spo0A_spoIIGA	1603693	1603710	SacY_sacX	3941965	3942007
ResD_cydA	3979497	3979448	AbrB_rbsR	3701375	3701407
ResD_ctaA-Ps;A3	1558984	1558940	DegU_sipT	1511245	1511266
ComK_addB	1136202	1136225	GerE_cgeA	2148520	2148551
YufM_yflS	829281	829336	ResD_fnr	3832318	3832301

SacT_sacP	3905159	3905117	PhoP_yfkN	859567	859541
ComK_recA	1764550	1764570	TnrA_pucJ	3330362	3330378
ComK_comK	1116955	1116990	ResD_ctaA-Pv;ctaA-Ps;A2	1559088	1559036
SpoIIID_spoIVCB	2653031	2653053	RocR_rocD	4145642	4145622
ComA_srfAA	376604	376641	PhoP_tuaA		
TnrA_nrgA	3756688	3756704	GerE_cotG	3717107	3717138
HutP_hutH	4041959	4042010	Spo0A_spoIIE	70461	70481
CitT_citM	834116	834145	PucR_ywoE	3753910	3753860
SpoIIID_spoIVCA	2654848	2654831	PerR_srfAA	376515	376565
htpG_-10-+17	4091356	4091330	BkdR_ptb	2504820	2504787
GerE_cotC	1905298	1905277	ComK_ybdK	221758	221776
ComK_comGA	2560200	2560165	PucR_pucH	3328746	3328706
GlpP_glpF	1002366	1002388	YdfI_ydfJ	589613	589651
TrnD-Trp_yczA	277027	277048	ComK_comF-P1	3643638	3643608
GltR_gltA	2014737	2014723	TrnSL-Tyr1_tyrZ	3947078	3947094
YycF_ykvT	1448278	1448309	Spo0A_sinI-P1	2552271	2552298
LevR_levD	2763010	2762968	CcpA_ackA	3016463	3016430
ComK_comC	2865294	2865265	CcpA_ilvB		
TrnI-Thr_thrZ	3856537	3856474	ComK_thdF	4212948	4212926
BltR_blt	2716867	2716897	ComK_comEA		
PhoP_phoA			TnrA_ywrD	3720794	3720778
ComK_radC	2862759	2862738	BmrR_bmr	2494591	2494621
TrnB-Gly1_glyQ	2608771	2608746	TrnJ-Gly_glyQ	2608771	2608746
Spo0A_spoIIE	70375	70401	AbrB_citB	1926577	1926625
BkdR_ptb	2504806	2504773	ComK_comK	1116990	1117018
ComK_radC	2862781	2862759	GutR_gutB	667324	667352
PhoP_phoP	2978600	2978561	PhoP_resA	2421448	2421381
FNR_narK	3833597	3833576	PhoP_phoP	2978483	2978455
TrnD-Trp_trpS	1219204	1219181	GerE_sspG	3353944	3353975
ComK_nucA	372739	372710	GerE_cotB	3716002	3715983
Spo0A_spoIIAA	2445096	2445077	Mta_mta	3764956	3764934
PhoP_phoP	2978755	2978727	CssR_cssR-S2	3385630	3385667
ComK_smf	1682487	1682503	ComK_yhjB	1120922	1120902
TnrA_tnrA	1397824	1397808	Spo0A_spoIIAA	2445053	2445038
RocR_rocG	3880686	3880658	TrnD-Cys_cysE	112689	112721
TrnD-Phe_pheS	2930589	2930551	TrnB-Phe_pheS	2930589	2930551
GerE_ftsY-PK			SacY_sacB	3535823	3535865
SacY_sacP	3905159	3905117	GerE_cgeC	2148624	2148593
AbrB_rbsR	3701237	3701313	ComK_yyaF	4201070	4201054
ComK_ywpH	3740687	3740667	GerE_cwlH	2649828	2649808
Spo0A_spoIIGA	1603649	1603664	Mta_blt	2716869	2716890
DegU_sipS-T6			LicT_bglS	4012805	4012766
GerE_cgeA	2148546	2148577	RocR_rocD	4145683	4145663
ComK_ywpH	3740710	3740687	Spo0A_spoIIGA	1603679	1603696
YcbA_ybgJ	265325	265282	DegU_sacX	3941835	3941891
ComK_recA	1764493	1764518	GerE_cgeA	2148563	2148594
ComA_degQ	3257381	3257339	SpoIIID_cotJA	755829	755858
TrnI-Gly_glyQ	2608771	2608746	AhrC_rocA	3880568	3880550
YvqC_yvqI	3399035	3399006	SacT_sacB	3535823	3535865
TnrA_nasB	362856	362838	GerE_cotY	1250600	1250577
SpoIIID_spoIVCA	2655030	2654997	ComK_recA	1764518	1764538
ComK_ybdK	221720	221737	RocR_rocA	3880730	3880707

Table D-19: The list of CDSs which encode for kinases. Columns show the comma-separated list of labels and start and end positions of these CDSs.

Name	Start	End	Name	Start	End
yfiJ	903811	905013	*ykrQ,kinE*	1419213	1421429
ybdK	221950	222912	*yvqE,liaS*	3396117	3395035
comP	3255838	3253529	*kinC*	1518333	1519619
glnJ,yzgA,ycbA	265476	266708	*yycG,walK*	4153688	4151853
ycbM	279059	279994	*kinB*	3230067	3231353
spo0D,spo0B	2854559	2853981	*yvqB,cssS*	3386398	3387753
desK,yocF	2090574	2091686	*resE,ypxE*	2417184	2415415
glnK,nrgB	3758016	3758366	*cheN,cheA*	1712815	1714833
degS	3646536	3645379	*ydfH*	587744	588967
ykoH	1392643	1394007	*ytvA*	3106210	3106995
yflR,citS	830945	832573	*lytS*	2958267	2956486
kinA,gsiC,spoIIF,spoIIJ	1470026	1471846	*ykvD,kinD*	1433006	1431486
yrkQ	2704448	2703150	*ytsB,bceS*	3113194	3112190
yxdK	4072287	4071310	*yvrG*	3408356	3406614
yesM	758719	760452	*yvcQ*	3567428	3566358
yvfT	3497610	3496495	*yhcY*	1008668	1009807
ywpD	3742384	3743220	*ydbF,dctS*	497768	499375
yxjM	3993162	3994382	*phoR*	2977807	2976068
yufL	3237157	3238758	*yhcK*	986813	985734
hprK,yvoB,ptsK	3595174	3594242	*yclK*	427247	428668

Table D-20: The list of CDSs which encode for response regulators. Columns show the comma-separated list of labels and start and end positions of these CDSs.

Name	Start	End	Name	Start	End
ycbB,yzgB,glnL	266719	267663	*glvR,yfiA,malR*	891436	892200
bceR,ytsA	3113882	3113187	*spo0A*	2518826	2518023
ydfI	588960	589601	*ybdJ*	221258	221929
phoP	2978522	2977800	*yfiK*	905010	905672
yclJ	426577	427260	*spo0F*	3809924	3809550
gerE	2904951	2904727	*walR,yycF*	4154403	4153696
cheY	1703834	1704196	*yoxH,ccdB,yneI*	1924030	1924392
citT,yflQ	832545	833225	*cheV*	1473605	1474516
yxdJ	4072973	4072284	*ywpD*	3742384	3743220
cheL,cheB	1711736	1712809	*yhcZ*	1009804	1010448
desR,yocG	2091705	2092304	*ykoG*	1391953	1392639
yufM	3238751	3239458	*yvrH,yvrHb*	3409066	3408353
lytT	2956508	2955783	*yvcP*	3568135	3567422
cssR,yvqA	3385724	3386401	*dctR,ydbG*	499365	500045
natR,yccH	296285	295584	*yvfU*	3496478	3495876
ycbL	278377	279057	*ypxD,resD*	2417903	2417181
yesN	760452	761558	*yxjL*	3994369	3995025
liaR,yvqC	3395057	3394422	*degU,iep*	3645296	3644607
yrkP	2705130	2704435	*comAA,comA*	3253448	3252804

Table D-21: The list of CDSs which encode for activators with known binding sites in BacillOndex. Columns show the comma-separated list of labels and start and end positions of these CDSs.

Name	Start	End	Name	Start	End
citT,yflQ	832545	833225	*licT*	4013699	4012866
phoP	2978522	2977800	*adaA*	203729	204364
cssR,yvqA	3385724	3386401	*ipa-13r,sacY*	3943667	3944509
ydfI	588960	589601	*spoIIID*	3748702	3748421
bltR,bmr2R,bmtR	2716856	2716035	*fnr*	3832228	3831512
ahrC	2522773	2522324	*gutR*	667264	664775
ytlI	3008938	3009864	*ypxD,resD*	2417903	2417181
hutP1,hutP	4041492	4041938	*gltR,yrdL*	2725837	2726727
ipa-47d,sacT	3906972	3906142	*abrB,cpsX*	45138	44848
yunI,pucR	3328762	3330357	*bmrR*	2495898	2496734
liaR,yvqC	3395057	3394422	*comK*	1117109	1117687
degU,iep	3645296	3644607	*glpP*	1001744	1002322
yzcB,yfjG,acoR	883758	885575	*yufM*	3238751	3239458
mta,ywnD	3764906	3764133	*glnJ,yzgA,ycbA*	265476	266708
walR,yycF	4154403	4153696	*glnK,nrgB*	3758016	3758366
scgR,tnrA	1397743	1397411	*ipa-89d,cysL,ywfK*	3865208	3864309
gltC	2014779	2015681	*ygaG,perR*	944487	944924
catA,scoC,hpr	1073717	1073106	*glvR,yfiA,malR*	891436	892200
ccpA,graR,alsA	3045169	3044165	*gerE*	2904951	2904727
rocR	4145747	4147132	*bkdR,yqiR*	2506867	2504789
comAA,comA	3253448	3252804	*hxlR,yckH*	376032	376394
levR	2765832	2763025	*spo0A*	2518826	2518023

Table D-22: The list of CDSs which encode for repressors with known binding sites in BacillOndex. Columns show the comma-separated list of labels and start and end positions of these CDSs.

Name	Start	End	Name	Start	End
lrpC,ydaI	476059	476493	*ccpA,graR,alsA*	3045169	3044165
ygaG,perR	944487	944924	*ahrC*	2522773	2522324
gntR	4113417	4114148	*rex,ydiH*	647091	647738
ipa-89d,cysL,ywfK	3865208	3864309	*yvbS,araC,araR*	3485670	3486758
comK	1117109	1117687	*sinR,flaD,sin*	2552653	2552988
phoP	2978522	2977800	*ykuM,ccpC*	1486045	1486926
fapR,ylpC	1661967	1662533	*rplT*	2952583	2952224
glnR	1877959	1878366	*yccB,lin-2,lmrA*	290698	290132
yqxE,yqeS,hrcA	2629637	2628606	*yabI,purR*	54441	55298
yjmH,exuR	1308802	1309803	*catA,scoC,hpr*	1073717	1073106
ccpN,yqzB	2605597	2604959	*spoIIID*	3748702	3748421
codY	1690119	1690898	*yodB*	2127683	2127345
gerE	2904951	2904727	*gltC*	2014779	2015681
iolR	4084799	4085554	*yacG,ctsR*	101449	101913
yvbA,sdpR	3467326	3467054	*degU,iep*	3645296	3644607
birA	2355895	2354918	*niaR,yrxA,nadR*	2850716	2851258
yqkL,fur	2450290	2449841	*ykdA,htrA*	1359285	1357936
yqfV,zur	2591865	2591428	*xylR*	1891666	1890512
pyrR	1618304	1618849	*yrzC,cymR*	2812133	2811717
abrB,cpsX	45138	44848	*mtrB,TRAP*	2384761	2384534

yvbQ,cggR	3483774	3482752	*dnaA2,dnaA,dnaA1*	410	1750
yfxA,treR	853556	854272	*scgR,tnrA*	1397743	1397411
rghR,yvaN,rghRA	3456667	3457074	*rocR*	4145747	4147132
desR,yocG	2091705	2092304	*xre*	1321670	1321329
ohrR,ykmA	1381877	1381434	*ysiA,fadR*	2918541	2917957
lexA,dinR	1918256	1917639	*gltR,yrdL*	2725837	2726727
arsR,yqcJ	2657634	2657317	*deoR,yxxC*	4053320	4052379
spo0A	2518826	2518023			

Appendix E. Taxonomy of the Organisms from BacillusRegNet

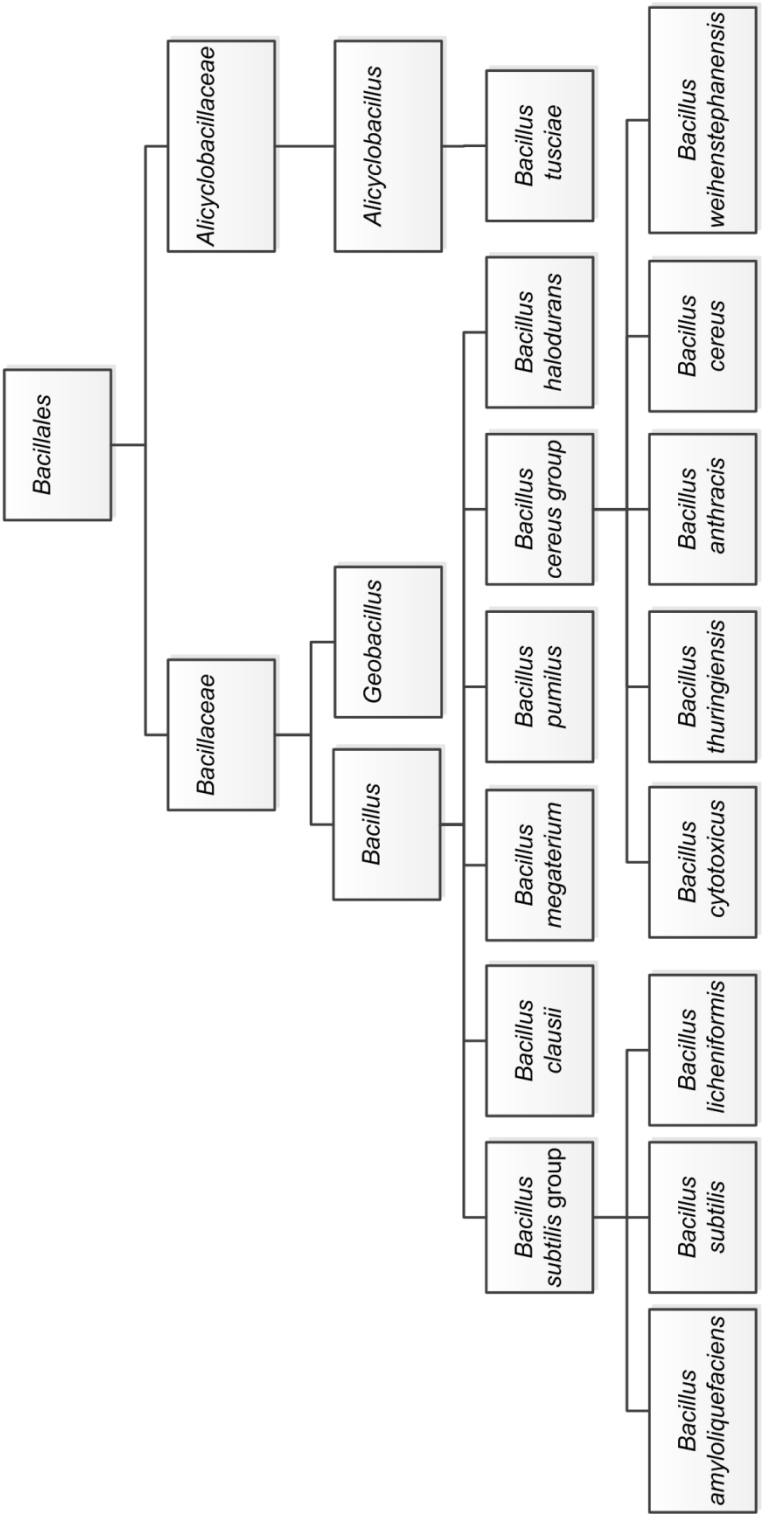


Figure E-1: The taxonomy tree of the organisms from BacillusRegNet.

Bibliography

1. What's in a name? Nature Biotechnology, 2009. **27**(12): p. 1071-1073.

2. Kitney, R. and P. Freemont, Synthetic biology - the state of play. FEBS Letters, 2012. **586**(15): p. 2029-2036.

3. Sinha, J., S.J. Reyes, and J.P. Gallivan, Reprogramming bacteria to seek and destroy an herbicide. Nature Chemical Biology, 2010. **6**(6): p. 464-470.

4. Ro, D.-K., E.M. Paradise, M. Ouellet, K.J. Fisher, K.L. Newman, J.M. Ndungu, K.A. Ho, R.A. Eachus, T.S. Ham, J. Kirby, M.C.Y. Chang, S.T. Withers, Y. Shiba, R. Sarpong, and J.D. Keasling, Production of the antimalarial drug precursor artemisinic acid in engineered yeast. Nature, 2006. **440**(7086): p. 940-943.

5. Choudhary, S. and C. Schmidt-Dannert, Applications of quorum sensing in biotechnology. Applied Microbiology and Biotechnology, 2010. **86**(5): p. 1267-1279.

6. Kirby, J.R., Synthetic biology: Designer bacteria degrades toxin. Nature Chemical Biology, 2010. **6**(6): p. 398-399.

7. Lee, S.K., H. Chou, T.S. Ham, T.S. Lee, and J.D. Keasling, Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels. Current Opinion in Biotechnology, 2008. **19**(6): p. 556-563.

8. Atsumi, S. and J.C. Liao, Metabolic engineering for advanced biofuels production from *Escherichia coli*. Current Opinion in Biotechnology, 2008. **19**(5): p. 414-419.

9. Du, J., Z. Shao, and H. Zhao, Engineering microbial factories for synthesis of value-added products. Journal of Industrial Microbiology & Biotechnology, 2011: p. 1-18.

10. Endy, D., Foundations for engineering biology. Nature, 2005. **438**(24): p. 449-453.

11. Marchisio, M.A. and J. Stelling, Computational design of synthetic gene circuits with composable parts. Bioinformatics, 2008. **24**(17): p. 1903-1910.

12. Peccoud, J., M.F. Blauvelt, Y. Cai, K.L. Cooper, O. Crasta, E.C. DeLalla, C. Evans, O. Folkerts, B.M. Lyons, S.P. Mane, R. Shelton, M.A. Sweede, and S.A. Waldon, Targeted Development of Registries of Biological Parts. PLoS ONE, 2008. **3**(7): p. e2671.

13. Galdzicki, M., C. Rodriguez, D. Chandran, H.M. Sauro, and J.H. Gennari, Standard Biological Parts Knowledgebase. PLoS ONE, 2011. **6**(2): p. e17005.

14. Hallinan, J., S. Park, and A. Wipat, Bridging the Gap between Design and Reality - A Dual Evolutionary Strategy for the Design of Synthetic Genetic Circuits, in *BIOINFORMATICS 2012 - International Conference on Bioinformatics Models, Methods and Algorithms*, J. Schier, C.M.B.A. Correia, A.L.N. Fred, and H. Gamboa, Editors. 2012, SciTePress. p. 263-268.

15. Lux, M.W., B.W. Bramlett, D.A. Ball, and J. Peccoud, Genetic design automation: engineering fantasy or scientific renewal? Trends in Biotechnology, 2011. **30**(2): p. 120–126.

16. Young, E. and H. Alper, Synthetic Biology: Tools to Design, Build, and Optimize Cellular Processes. Journal of Biomedicine and Biotechnology, 2010. **2010**.

17. Fritz, B.R., L.E. Timmerman, N.M. Daringer, J.N. Leonard, and M.C. Jewett, Biology by Design: From Top to Bottom and Back. Journal of Biomedicine and Biotechnology, 2010. **2010**.

18. Hallinan, J.S., G. Misirli, and A. Wipat, Evolutionary computation for the design of a stochastic switch for synthetic genetic circuits, in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*. 2010, IEEE: Buenos Aires, Argentina. p. 768-774.

19. Keasling, J.D. and K.N. Timmis, Microbial Production of Isoprenoids, in *Handbook of Hydrocarbon and Lipid Microbiology*. 2010, Springer Berlin Heidelberg. p. 2951-2966.

20. Smolke, Christina D. and Pamela A. Silver, Informing Biological Design by Integration of Systems and Synthetic Biology. Cell, 2011. **144**(6): p. 855-859.

21. Bilitchenko, L., A. Liu, S. Cheung, E. Weeding, B. Xia, M. Leguia, J.C. Anderson, and D. Densmore, Eugene - A Domain Specific Language for Specifying and Constraining Synthetic Biological Parts, Devices, and Systems. PLoS ONE, 2011. **6**(4): p. e18882.

22. MacDonald, J.T., C. Barnes, R.I. Kitney, P.S. Freemont, and G.-B.V. Stan, Computational design approaches and tools for synthetic biology. Integrative Biology, 2011. **3**(2): p. 97-108.

23. Fordyce, P.M., D. Gerber, D. Tran, J. Zheng, H. Li, J.L. DeRisi, and S.R. Quake, *De novo* identification and biophysical characterization of transcription-factor binding sites with microfluidic affinity analysis. Nature Biotechnology, 2010. **28**(9): p. 970-975.

24. Spiller, D.G., C.D. Wood, D.A. Rand, and M.R.H. White, Measurement of single-cell dynamics. Nature, 2010. **465**(7299): p. 736-745.

25. van Oijen, A.M., Single-molecule approaches to characterizing kinetics of biomolecular interactions. Current Opinion in Biotechnology, 2011. **22**(1): p. 75-80.

26. Bennett, M.R. and J. Hasty, Microfluidic devices for measuring gene network dynamics in single cells. Nature Reviews Genetics, 2009. **10**(9): p. 628-638.

27. Liang, J., Y. Luo, and H. Zhao, Synthetic biology: putting synthesis into biology. Wiley Interdisciplinary Reviews: Systems Biology and Medicine, 2010. **3**(1): p. 7-20.

28. Koide, T., W. Lee Pang, and N.S. Baliga, The role of predictive modelling in rationally re-engineering biological systems. Nature Reviews Microbiology, 2009. **7**(4): p. 297-305.

29. Guido, N.J., X. Wang, D. Adalsteinsson, D. McMillen, J. Hasty, C.R. Cantor, T.C. Elston, and J.J. Collins, A bottom-up approach to gene regulation. Nature, 2006. **439**(7078): p. 856-860.

30. Rodrigo, G., J. Carrera, and A. Jaramillo, Computational design of synthetic regulatory networks from a genetic library to characterize the designability of dynamical behaviors. Nucleic Acids Research, 2011. **39**(20): p. e138.

31. Slusarczyk, A.L., A. Lin, and R. Weiss, Foundations for the design and implementation of synthetic genetic circuits. Nature Reviews Genetics, 2012. **13**(6): p. 406-420.

32. Marchisio, M.A. and J. Stelling, Computational design tools for synthetic biology. Current Opinion in Biotechnology, 2009. **20**(4): p. 479-485.

33. Hill, A.D., J.R. Tomshine, E.M.B. Weeding, V. Sotiropoulos, and Y.N. Kaznessis, SynBioSS: the synthetic biology modeling suite. Bioinformatics, 2008. **24**(21): p. 2551-2553.

34. Galperin, M.Y. and X.M. Fernández-Suárez, The 2012 Nucleic Acids Research Database Issue and the online Molecular Biology Database Collection. Nucleic Acids Research, 2012. **40**(D1): p. D1-D8.

35. Gruber, T.R., Towards Principles for the Design of Ontologies Used for Knowledge Sharing, in *Formal Ontology in Conceptual Analysis and Knowledge Representation*, N. Guarino and R. Poli, Editors. 1993, Kluwer Academic Publishers.

36. Misirli, G., J.S. Hallinan, T. Yu, J.R. Lawson, S.M. Wimalaratne, M.T. Cooling, and A. Wipat, Model annotation for synthetic biology: automating model to nucleotide sequence conversion. Bioinformatics, 2011. **27**(7): p. 973-979.

37. Agapakis, C.M. and P.A. Silver, Synthetic biology: exploring and exploiting genetic modularity through the design of novel biological networks. Molecular BioSystems, 2009. **5**(7): p. 704-713.

38. Lu, T.K., A.S. Khalil, and J.J. Collins, Next-generation synthetic gene networks. Nature Biotechnology, 2009. **27**(12): p. 1139-1150.

39. Kaznessis, Y., Models for synthetic biology. BMC Systems Biology, 2007. **1**(1): p. 47.

40. Chandran, D., F. Bergmann, and H. Sauro, TinkerCell: modular CAD tool for synthetic biology. Journal of Biological Engineering, 2009. **3**(1): p. 19.

41. Rodrigo, G., J. Carrera, and A. Jaramillo, Asmparts: assembly of biological model parts. Systems and Synthetic Biology, 2007. **1**(4): p. 167-170.

42. Rodrigo, G., J. Carrera, and A. Jaramillo, Genetdes: automatic design of transcriptional networks. Bioinformatics, 2007. **23**(14): p. 1857-1858.

43. Pedersen, M. and A. Phillips, Towards programming languages for genetic engineering of living cells. Journal of The Royal Society Interface, 2009: p. rsif.2008.0516.focus.

44. Cai, Y., B. Hartnett, C. Gustafsson, and J. Peccoud, A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. Bioinformatics, 2007. **23**(20): p. 2760-2767.

45. Funahashi, A., M. Morohashi, H. Kitano, and N. Tanimura, CellDesigner: a process diagram editor for gene-regulatory and biochemical networks. BIOSILICO, 2003. **1**(5): p. 159-162.

46. Li, C., M. Donizelli, N. Rodriguez, H. Dharuri, L. Endler, V. Chelliah, L. Li, E. He, A. Henry, M. Stefan, J. Snoep, M. Hucka, N. Le Novere, and C. Laibe, BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. BMC Systems Biology, 2010. **4**(1): p. 92.

47. Myers, C.J., N. Barker, K. Jones, H. Kuwahara, C. Madsen, and N.-P.D. Nguyen, iBioSim: a tool for the analysis and design of genetic circuits. Bioinformatics, 2009. **25**(21): p. 2848-2849.

48. Chandran, D., W.B. Copeland, S.C. Sleight, and H.M. Sauro, Mathematical modeling and synthetic biology. Drug Discovery Today: Disease Models, 2008. **5**(4): p. 299-309.

49. Clancy, K. and C.A. Voigt, Programming cells: towards an automated 'Genetic Compiler'. Current Opinion in Biotechnology, 2010. **21**(4): p. 572-581.

50. Medema, M.H., R. van Raaphorst, E. Takano, and R. Breitling, Computational tools for the synthetic design of biochemical pathways. Nature Reviews Microbiology, 2012. **10**(3): p. 191-202.

51. Alterovitz, G., T. Muso, and M.F. Ramoni, The challenges of informatics in synthetic biology: from biomolecular networks to artificial organisms. Briefings in Bioinformatics, 2009. **11**(1): p. 80-95.

52. Hallinan, J., M. Pocock, and W. Anil, Evolving genetic circuit designs for synthetic biology, in *Synthetic Biology 4.0*. 2008: Hong Kong.

53. Endler, L., N. Rodriguez, N. Juty, V. Chelliah, C. Laibe, C. Li, and N. Le Novère, Designing and encoding models for synthetic biology. Journal of The Royal Society Interface, 2009: p. rsif.2009.0035.focus.

54. Searls, D.B., Data integration: challenges for drug discovery. Nature Reviews Drug Discovery, 2005. **4**(1): p. 45-58.

55. Manconi, A., P. Rodriguez-Tomé, M. Biba, and F. Xhafa, A Survey on Integrating Data in Bioinformatics, in *Learning Structure and Schemas from Documents*. 2011, Springer Berlin / Heidelberg. p. 413-432.

56. Antezana, E., M. Kuiper, and V. Mironov, Biological knowledge management: the emerging role of the Semantic Web technologies. Briefings in Bioinformatics, 2009. **10**(4): p. 392-407.

57. Lanza, A.M., N.C. Crook, and H.S. Alper, Innovation at the intersection of synthetic and systems biology. Current Opinion in Biotechnology, 2012.

58. De Las Heras, A., C.A. Carreño, E. Martínez-García, and V. De Lorenzo, Engineering input/output nodes in prokaryotic regulatory circuits. FEMS Microbiology Reviews, 2010. **34**(5): p. 842-865.

59. Shadbolt, N., W. Hall, and T. Berners-Lee, The Semantic Web Revisited. Intelligent Systems, IEEE, 2006. **21**(3): p. 96-101.

60. Belleau, F., M.-A. Nolin, N. Tourigny, P. Rigault, and J. Morissette, Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. Journal of Biomedical Informatics, 2008. **41**(5): p. 706-716.

61. Cheung, K.-H., M. Samwald, R.K. Auerbach, and M.B. Gerstein, Structured digital tables on the Semantic Web: toward a structured digital literature. Molecular Systems Biology, 2010. **6**.

62. Bard, J.B.L. and S.Y. Rhee, Ontologies in biology: design, applications and future challenges. Nature Reviews Genetics, 2004. **5**(3): p. 213-222.

63. Galdzicki, M., M.L. Wilson, C.A. Rodriguez, L. Adam, A. Adler, J.C. Anderson, J. Beal, D. Chandran, D. Densmore, O.A. Drory, D. Endy, J.H. Gennari, R. Grünberg, T.S. Ham, A. Kuchinsky, M.W. Lux, C. Madsen, G. Misirli, C.J. Myers, J. Peccoud, H. Plahar, M.R. Pocock, N. Roehner, T.F. Smith, G.-B. Stan, A. Villalobos, A. Wipat, and H.M. Sauro, Synthetic Biology Open Language (SBOL) Version 1.0.0, in *BBF RFC #84*. 2011.

64. Cuellar, A.A., C.M. Lloyd, P.F. Nielsen, D.P. Bullivant, D.P. Nickerson, and P.J. Hunter, An Overview of CellML 1.1, a Biological Model Description Language. SIMULATION, 2003. **79**(12): p. 740-747.

65. Hucka, M., A. Finney, H.M. Sauro, H. Bolouri, J.C. Doyle, H. Kitano, S.F. and the rest of the, A.P. Arkin, B.J. Bornstein, D. Bray, A. Cornish-Bowden, A.A. Cuellar, S. Dronov, E.D. Gilles, M. Ginkel, V. Gor, I.I. Goryanin, W.J. Hedley, T.C. Hodgman, J.H. Hofmeyr, P.J. Hunter, N.S. Juty, J.L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L.M. Loew, D. Lucio, P. Mendes, E. Minch, E.D. Mjolsness, Y. Nakayama, M.R. Nelson, P.F. Nielsen, T. Sakurada, J.C. Schaff, B.E. Shapiro, T.S. Shimizu, H.D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics, 2003. **19**(4): p. 524-531.

66. Marchisio, M.A. and J. Stelling, Automatic Design of Digital Synthetic Gene Circuits. PLoS Computational Biology, 2011. **7**(2): p. e1001083.

67. Barrett, C.L., T.Y. Kim, H.U. Kim, B.Ø. Palsson, and S.Y. Lee, Systems biology as a foundation for genome-scale synthetic biology. Current Opinion in Biotechnology, 2006. **17**(5): p. 488-492.

68. Lim, W.A., Designing customized cell signalling circuits. Nature Reviews Molecular Cell Biology, 2010. **11**(6): p. 393-403.

69. Tamsir, A., J.J. Tabor, and C.A. Voigt, Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. Nature, 2011. **469**(7329): p. 212-215.

70. Cox, R.S., M.G. Surette, and M.B. Elowitz, Programming gene expression with combinatorial promoters. Molecular Systems Biology, 2007. **3**.

71. Ellis, T., X. Wang, and J.J. Collins, Diversity-based, model-guided construction of synthetic gene networks with predicted functions. Nature Biotechnology, 2009. **27**(5): p. 465-471.

72. Khalil, A.S. and J.J. Collins, Synthetic biology: applications come of age. Nature Reviews Genetics, 2010. **11**(5): p. 367-379.

73. Densmore, D., J.T. Kittleson, L. Bilitchenko, A. Liu, and J.C. Anderson, Rule based constraints for the construction of genetic devices, in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*. 2010. p. 557-560.

74. Hatzimanikatis, V., C. Li, J.A. Ionita, C.S. Henry, M.D. Jankowski, and L.J. Broadbelt, Exploring the diversity of complex metabolic networks. Bioinformatics, 2005. **21**(8): p. 1603-1609.

75. Hallinan, J., M. Pocock, M. Taschuk, and W. Anil, Data Integration to Constrain Computational Modelling in Synthetic Biology, in *BioSysBio*. 2008: Cambridge.

76. Marchisio, M.A. and F. Rudolf, Synthetic biosensing systems. The International Journal of Biochemistry & Cell Biology, 2010. **43**(3): p. 310-319.

77. Stelling, J., Mathematical models in microbial systems biology. Current Opinion in Microbiology, 2004. **7**(5): p. 513-518.

78. Buchler, N.E., U. Gerland, and T. Hwa, On schemes of combinatorial transcription logic. Proceedings of the National Academy of Sciences, 2003. **100**(9): p. 5136-5141.

79. Del Vecchio, D., A.J. Ninfa, and E.D. Sontag, Modular cell biology: retroactivity and insulation. Molecular Systems Biology, 2008. **4:161**.

80. Cooling, M.T., V. Rouilly, G. Misirli, J. Lawson, T. Yu, J. Hallinan, and A. Wipat, Standard virtual biological parts: a repository of modular modeling components for synthetic biology. Bioinformatics, 2010. **26**(7): p. 925-931.

81. Lister, A.L., M. Pocock, M. Taschuk, and A. Wipat, Saint: a lightweight integration environment for model annotation. Bioinformatics, 2009. **25**(22): p. 3026-3027.

82. Misirli, G., J. Hallinan, J. Weile, S. Cockell, and A. Wipat, BacillOndex: Data integration and visualisation for *Bacillus subtilis*, in *School of Computing Science Technical Report Series 1237*. 2011, School of Computing Science, University of Newcastle upon Tyne.

83. Hartwell, L.H., J.J. Hopfield, S. Leibler, and A.W. Murray, From molecular to modular cell biology. Nature, 1999. **402**: p. C47-52.

84. de Lorenzo, V. and A. Danchin, Synthetic biology: discovering new worlds and new words. The new and not so new aspects of this emerging research field. EMBO Reports, 2008. **9**(9): p. 822-827.

85. Hallinan, J., Synthetic Biology: Life, Jim, but Not As We Know It, in *Design by Evolution*, P.F. Hingston, L.C. Barone, and Z. Michalewicz, Editors. 2008, Springer Berlin Heidelberg. p. 53-68.

86. Purnick, P.E.M. and R. Weiss, The second wave of synthetic biology: from modules to systems. Nature Reviews Molecular Cell Biology, 2009. **10**(6): p. 410-422.

87. Andrianantoandro, E., S. Basu, D.K. Karig, and R. Weiss, Synthetic biology: new engineering rules for an emerging discipline. Molecular Systems Biology, 2006. **2**.

88. Li, B. and L. You, Synthetic biology: Division of logic labour. Nature, 2011. **469**(7329): p. 171-172.

89. Canton, B., A. Labno, and D. Endy, Refinement and standardization of synthetic biological parts and devices. Nature Biotechnology, 2008. **26**(7): p. 787-793.

90. Mukherji, S. and A. van Oudenaarden, Synthetic biology: understanding biological design from synthetic circuits. Nature Reviews Genetics, 2009. **10**(12): p. 859-871.

91. Ninfa, A.J., Use of two-component signal transduction systems in the construction of synthetic genetic networks. Current Opinion in Microbiology, 2010. **13**(2): p. 240-245.

92. Benson, D.A., I. Karsch-Mizrachi, K. Clark, D.J. Lipman, J. Ostell, and E.W. Sayers, GenBank. Nucleic Acids Research, 2012. **40**(D1): p. D48-D53.

93. Baker, D., G. Church, J. Collins, D. Endy, J. Jacobson, J. Keasling, P. Modrich, C. Smolke, and R. Weiss, Engineering life: building a fab for biology. Scientific American, 2006. **294**(6): p. 44-51.

94. Oldham, P., S. Hall, and G. Burton, Synthetic Biology: Mapping the Scientific Landscape. PLoS ONE, 2012. **7**(4): p. e34368.

95. Chen, Z., M. Wilmanns, and A.-P. Zeng, Structural synthetic biotechnology: from molecular structure to predictable design for industrial strain development. Trends in Biotechnology, 2010. **28**(10): p. 534-542.

96. Friedland, A.E., T.K. Lu, X. Wang, D. Shi, G. Church, and J.J. Collins, Synthetic Gene Networks That Count. Science, 2009. **324**(5931): p. 1199-1202.

97. Baumgardner, J., K. Acker, O. Adefuye, S. Crowley, W. DeLoache, J. Dickson, L. Heard, A. Martens, N. Morton, M. Ritter, A. Shoecraft, J. Treece, M. Unzicker, A. Valencia, M. Waters, A.M. Campbell, L. Heyer, J. Poet, and T. Eckdahl, Solving a Hamiltonian Path Problem with a bacterial computer. Journal of Biological Engineering, 2009. **3**(1): p. 11.

98. Haynes, K., M. Broderick, A. Brown, T. Butner, J. Dickson, W.L. Harden, L. Heard, E. Jessen, K. Malloy, B. Ogden, S. Rosemond, S. Simpson, E. Zwack, A.M. Campbell, T. Eckdahl, L. Heyer, and J. Poet, Engineering bacteria to solve the Burnt Pancake Problem. Journal of Biological Engineering, 2008. **2**(1): p. 8.

99. Stojanovic, M.N. and D. Stefanovic, A deoxyribozyme-based molecular automaton. Nature Biotechnology, 2003. **21**(9): p. 1069-1074.

100. Ham, T.S., S.K. Lee, J.D. Keasling, and A.P. Arkin, Design and Construction of a Double Inversion Recombination Switch for Heritable Sequential Genetic Memory. PLoS ONE, 2008. **3**(7): p. e2815.

101. Quiroz, J.N.A., R.B. Flores, T.G.B. Cisneros, I.Y.F. Rosales, A.G. Naranjo, J.C.G. Sanchez, M.E.G. Jimenez, R.E.G. Padilla, A.J.L. Baena, P.A.L. Hernandez, P.G. Padilla, R.P. Miller, I.N.R. Gaspar, J.C.R. Chico, A.R. Martinez, J.P. Romero, A.S. Arzate, J.S.A. Barradas, D.A. Diaz, A.B. Bracho, C. Benitez, C.I.F. Arteag, F.H. Quiroz, G.J. Martinez, J.L. Rabadan, M.C.O. Salvador, P.P. Longoria, R.P. Orozco, F.R. Corona, E.S. Majarrez, E.S. Hassan, C.S. Sanchez, U.V. Saldana, and P.B.Z. Segura, Biological implementation of algorithms and unconventional computing. Synthetic Biology, IET, 2007. **1**(1.2): p. 59-60.

102. Y.-H. Percival Zhang, B.R.E., Jonathan R. Mielenz, Robert C. Hopkins, Michael W. W. Adams, High-Yield Hydrogen Production from Starch and Water by a Synthetic Enzymatic Pathway. 2007.

103. Van Tittelboom, K., N. De Belie, W. De Muynck, and W. Verstraete, Use of bacteria to repair cracks in concrete. Cement and Concrete Research, 2010. **40**(1): p. 157-166.

104. Levskaya, A., A.A. Chevalier, J.J. Tabor, Z.B. Simpson, L.A. Lavery, M. Levy, E.A. Davidson, A. Scouras, A.D. Ellington, E.M. Marcotte, and C.A. Voigt, Synthetic biology: Engineering *Escherichia coli* to see light. Nature, 2005. **438**(7067): p. 441-442.

105. Kortmann, J., S. Sczodrok, J. Rinnenthal, H. Schwalbe, and F. Narberhaus, Translation on demand by a simple RNA-based thermosensor. Nucleic Acids Research, 2011. **39**(7): p. 2855-2868.

106. Davis, J.H., A.J. Rubin, and R.T. Sauer, Design, construction and characterization of a set of insulated bacterial promoters. Nucleic Acids Research, 2010. **39**(3): p. 1131-1141.

107. Elowitz, M.B. and S. Leibler, A synthetic oscillatory network of transcriptional regulators. Nature, 2000. **403**(6767): p. 335-338.

108. Gardner, T.S., C.R. Cantor, and J.J. Collins, Construction of a genetic toggle switch in *Escherichia coli.* Nature, 2000. **403**(6767): p. 339-342.

109. Rodrigo, G. and A. Jaramillo, Computational design of digital and memory biological devices. Systems and Synthetic Biology, 2007. **1**(4): p. 183-195.

110. Salis, H.M., E.A. Mirsky, and C.A. Voigt, Automated design of synthetic ribosome binding sites to control protein expression. Nature Biotechnology, 2009. **27**(10): p. 946-950.

111. Vogel, C., R. de Sousa Abreu, D. Ko, S.-Y. Le, B.A. Shapiro, S.C. Burns, D. Sandhu, D.R. Boutz, E.M. Marcotte, and L.O. Penalva, Sequence signatures and mRNA concentration can explain two-thirds of protein abundance variation in a human cell line. Molecular Systems Biology, 2010. **6**.

112. Rackham, O. and J.W. Chin, A network of orthogonal ribosome·mRNA pairs. Nature Chemical Biology, 2005. **1**(3): p. 159-166.

113. Na, D., S. Lee, and D. Lee, Mathematical modeling of translation initiation for the estimation of its efficiency to computationally design mRNA sequences with desired expression levels in prokaryotes. BMC Systems Biology, 2010. **4**(1): p. 71.

114. Na, D. and D. Lee, RBSDesigner: software for designing synthetic ribosome binding sites that yields a desired level of protein expression. Bioinformatics, 2010. **26**(20): p. 2633-2634.

115. Arkin, A., Setting the standard in synthetic biology. Nature Biotechnology, 2008. **26**(7): p. 771-774.

116. Braff, J.C., C.M. Conboy, and D. Endy, Definitions and Measures of Performance for Standard Biological Parts, in *International Conference Systems Biology (ICSB 2005)*. 2005.

117. Beal, J., T. Lu, and R. Weiss, Automatic Compilation from High-Level Biologically-Oriented Programming Language to Genetic Regulatory Networks. PLoS ONE, 2011. **6**(8): p. e22490.

118. Tyo, K.E.J., K. Kocharin, and J. Nielsen, Toward design-based engineering of industrial microbes. Current Opinion in Microbiology, 2010. **13**(3): p. 255-262.

119. Haldenwang, W.G., The sigma factors of *Bacillus subtilis.* Microbiological Reviews, 1995. **59**(1): p. 1-30.

120. Cozy, L.M. and D.B. Kearns, Gene position in a long operon governs motility development in *Bacillus subtilis.* Molecular Microbiology, 2010. **76**(2): p. 273-285.

121. Montero Llopis, P., A.F. Jackson, O. Sliusarenko, I. Surovtsev, J. Heinritz, T. Emonet, and C. Jacobs-Wagner, Spatial organization of the flow of genetic information in bacteria. Nature, 2010. **466**(7302): p. 77-81.

122. Szurmant, H. and J.A. Hoch, Interaction fidelity in two-component signaling. Current Opinion in Microbiology, 2010. **13**(2): p. 190-197.

123. Beal, J., A. Phillips, D. Densmore, and Y. Cai, High-Level Programming Languages for Biomolecular Systems, in *Design and Analysis of Biomolecular Circuits*, H. Koeppl, G. Setti, M. di Bernardo, and D. Densmore, Editors. 2011, Springer New York. p. 225-252.

124. Barbe, V., S. Cruveiller, F. Kunst, P. Lenoble, G. Meurice, A. Sekowska, D. Vallenet, T. Wang, I. Moszer, C. Medigue, and A. Danchin, From a consortium sequence to a unified sequence: the *Bacillus subtilis* 168 reference genome a decade later. Microbiology, 2009. **155**(6): p. 1758-1775.

125. Harwood, C.R., *Bacillus subtilis* and its relatives: molecular biological and industrial workhorses. Trends in Biotechnology, 1992. **10**: p. 247-256.

126. Florez, L.A., S.F. Roppel, A.G. Schmeisky, C.R. Lammers, and J. Stulke, A community-curated consensual annotation that is continuously updated: the *Bacillus subtilis* centred wiki SubtiWiki. Database, 2009. **2009**(0): p. bap012-.

127. Westers, L., H. Westers, and W.J. Quax, *Bacillus subtilis* as cell factory for pharmaceutical proteins: a biotechnological approach to optimize the host organism. Biochimica et Biophysica Acta (BBA) - Molecular Cell Research, 2004. **1694**(1-3): p. 299-310.

128. Tosato, V. and C.V. Bruschi, Knowledge of the *Bacillus subtilis* genome: impacts on fundamental science and biotechnology. Applied Microbiology and Biotechnology, 2004. **64**(1): p. 1-6.

129. Kobayashi, K., S.D. Ehrlich, A. Albertini, G. Amati, K.K. Andersen, M. Arnaud, K. Asai, S. Ashikaga, S. Aymerich, P. Bessieres, F. Boland, S.C. Brignell, S. Bron, K. Bunai, J. Chapuis, L.C. Christiansen, A. Danchin, M. Débarbouillé, E. Dervyn, E. Deuerling, K. Devine, S.K. Devine, O. Dreesen, J. Errington, S. Fillinger, S.J. Foster, Y. Fujita, A. Galizzi, R. Gardan, C. Eschevins, T. Fukushima, K. Haga, C.R. Harwood, M. Hecker, D. Hosoya, M.F. Hullo, H. Kakeshita, D. Karamata, Y. Kasahara, F. Kawamura, K. Koga, P. Koski, R. Kuwana, D. Imamura, M. Ishimaru, S. Ishikawa, I. Ishio, D. Le Coq, A. Masson, C. Mauël, R. Meima, R.P. Mellado, A. Moir, S. Moriya, E. Nagakawa, H. Nanamiya, S. Nakai, P. Nygaard, M. Ogura, T. Ohanan, M. O'Reilly, M. O'Rourke, Z. Pragai, H.M. Pooley, G. Rapoport, J.P. Rawlins, L.A. Rivas, C. Rivolta, A. Sadaie, Y. Sadaie, M. Sarvas, T. Sato, H.H. Saxild, E. Scanlan, W. Schumann, J.F.M.L. Seegers, J. Sekiguchi, A. Sekowska, S.J. Séror, M. Simon, P. Stragier, R. Studer, H. Takamatsu, T. Tanaka, M. Takeuchi, H.B. Thomaides, V. Vagner, J.M. van Dijl, K. Watabe, A. Wipat, H. Yamamoto, M. Yamamoto, Y. Yamamoto, K. Yamane, K. Yata, K. Yoshida, H. Yoshikawa, U. Zuber, and N. Ogasawara, Essential *Bacillus subtilis* genes. Proceedings of the National Academy of Sciences of the United States of America, 2003. **100**(8): p. 4678-4683.

130. Goelzer, A., F. Bekkal Brikci, I. Martin-Verstraete, P. Noirot, P. Bessieres, S. Aymerich, and V. Fromion, Reconstruction and analysis of the genetic and metabolic regulatory networks of the central metabolism of *Bacillus subtilis.* BMC Systems Biology, 2008. **2**(1): p. 20.

131. Daniel, L., V. Hera, and K. Roberto, Generation of multiple cell types in *Bacillus subtilis.* FEMS Microbiology Reviews, 2009. **33**(1): p. 152-163.

132. Moreno-Campuzano, S., S.C. Janga, and E. Perez-Rueda, Identification and analysis of DNA-binding transcription factors in *Bacillus subtilis* and other Firmicutes- a genomic approach. BMC Genomics, 2006. **7**(1): p. 147.

133. Hallinan, J., G. Misirli, M. Pocock, Errington J., Harwood C., and A. Wipat, *Bacillus subtilis* as a chassis for synthetic biology, in *BioSysBio*. 2009.

134. Tanaka, K., C.S. Henry, J.F. Zinner, E. Jolivet, M.P. COhoon, F. Xia, V. Bldnenko, S.D. Ehrlich, R.L. Stevens, and P. Noirot. Systematic deletions in *Bacillus subtilis* genome coupled with large scale metabolic model refinement. 2012 [cited 09/05/2012]; Available from: www.mcs.anl.gov/uploads/cels/papers/P2030-0212.pdf.

135. Kobayashi, H., M. Kærn, M. Araki, K. Chung, T.S. Gardner, C.R. Cantor, and J.J. Collins, Programmable cells: Interfacing natural and engineered gene networks. Proceedings of the National Academy of Sciences of the United States of America, 2004. **101**(22): p. 8414-8419.

136. French, C.E., K.d. Mora, N. Joshi, A. Elfick, J. Haseloff, and J. Ajioka, SYNTHETIC BIOLOGY AND THE ART OF BIOSENSOR DESIGN, in *The Science and Applications of Synthetic and Systems Biology: Workshop Summary*. 2011, National Academies Press: Washington, D.C. p. 178-201.

137. Saeidi, N., C.K. Wong, T.-M. Lo, H.X. Nguyen, H. Ling, S.S.J. Leong, C.L. Poh, and M.W. Chang, Engineering microbes to sense and eradicate *Pseudomonas aeruginosa*, a human pathogen. Molecular Systems Biology, 2011. **7**.

138. Bischofs, I.B., J.A. Hug, A.W. Liu, D.M. Wolf, and A.P. Arkin, Complexity in bacterial cell-cell communication: Quorum signal integration and subpopulation signaling in the *Bacillus subtilis* phosphorelay. Proceedings of the National Academy of Sciences, 2009. **106**(16): p. 6459-6464.

139. Shong, J., M.R. Jimenez Diaz, and C.H. Collins, Towards synthetic microbial consortia for bioprocessing. Current Opinion in Biotechnology, 2012.

140. Schadt, E.E., M.D. Linderman, J. Sorenson, L. Lee, and G.P. Nolan, Computational solutions to large-scale data management and analysis. Nature Reviews Genetics, 2010. **11**(9): p. 647-657.

141. Kunst, F., N. Ogasawara, I. Moszer, A.M. Albertini, G. Alloni, V. Azevedo, M.G. Bertero, P. Bessieres, A. Bolotin, S. Borchert, R. Borriss, L. Boursier, A. Brans, M. Braun, S.C. Brignell, S. Bron, S. Brouillet, C.V. Bruschi, B. Caldwell, V. Capuano, N.M. Carter, S.K. Choi, J.J. Codani, I.F. Connerton, N.J. Cummings, R.A. Daniel, F. Denizot, K.M. Devine, A. Dusterhoft, S.D. Ehrlich, P.T. Emmerson, K.D. Entian, J. Errington, C. Fabret, E. Ferrari, D. Foulger, C. Fritz, M. Fujita, Y. Fujita, S. Fuma, A. Galizzi, N. Galleron, S.Y. Ghim, P. Glaser, A. Goffeau, E.J. Golightly, G. Grandi, G. Guiseppi, B.J. Guy, K. Haga, J. Haiech, C.R. Harwood, A. Henaut, H. Hilbert, S. Holsappel, S. Hosono, M.F. Hullo, M. Itaya, L. Jones, B. Joris, D. Karamata, Y. Kasahara, M. Klaerr-Blanchard, C. Klein, Y. Kobayashi, P. Koetter, G. Koningstein, S. Krogh, M. Kumano, K. Kurita, A. Lapidus, S. Lardinois, J. Lauber, V. Lazarevic, S.M. Lee, A. Levine, H. Liu, S. Masuda, C. Mauel, C. Medigue, N. Medina, R.P. Mellado,

M. Mizuno, D. Moestl, S. Nakai, M. Noback, D. Noone, M. O'Reilly, K. Ogawa, A. Ogiwara, B. Oudega, S.H. Park, V. Parro, T.M. Pohl, D. Portetelle, S. Porwollik, A.M. Prescott, E. Presecan, P. Pujic, B. Purnelle, G. Rapoport, M. Rey, S. Reynolds, M. Rieger, C. Rivolta, E. Rocha, B. Roche, M. Rose, Y. Sadaie, T. Sato, E. Scanlan, S. Schleich, R. Schroeter, F. Scoffone, J. Sekiguchi, A. Sekowska, S.J. Seror, P. Serror, B.S. Shin, B. Soldo, A. Sorokin, E. Tacconi, T. Takagi, H. Takahashi, K. Takemaru, M. Takeuchi, A. Tamakoshi, T. Tanaka, P. Terpstra, A. Tognoni, V. Tosato, S. Uchiyama, M. Vandenbol, F. Vannier, A. Vassarotti, A. Viari, R. Wambutt, E. Wedler, H. Wedler, T. Weitzenegger, P. Winters, A. Wipat, H. Yamamoto, K. Yamane, K. Yasumoto, K. Yata, K. Yoshida, H.F. Yoshikawa, E. Zumstein, H. Yoshikawa and A. Danchin, The complete genome sequence of the Gram-positive bacterium *Bacillus subtilis.* Nature, 1997. **390**(6657): p. 249-256.

142. Richardson, E.J. and M. Watson, The automatic annotation of bacterial genomes. Briefings in Bioinformatics, 2012.

143. Baker, P.G., C.A. Goble, S. Bechhofer, N.W. Paton, R. Stevens, and A. Brass, An ontology for bioinformatics applications. Bioinformatics, 1999. **15**(6): p. 510-520.

144. Wolstencroft, K., P. Lord, L. Tabernero, A. Brass, and R. Stevens, Protein classification using ontology classification. Bioinformatics, 2006. **22**(14): p. e530-e538.

145. Smith, B., W. Ceusters, B. Klagges, J. Kohler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. Rector, and C. Rosse, Relations in biomedical ontologies. Genome Biology, 2005b. **6**(5): p. R46.

146. Antezana, E., M. Egana, W. Blonde, A. Illarramendi, I. Bilbao, B. De Baets, R. Stevens, V. Mironov, and M. Kuiper, The Cell Cycle Ontology: an application ontology for the representation and integrated analysis of the cell cycle process. Genome Biology, 2009. **10**(5): p. R58.

147. Stein, L.D., Integrating biological databases. Nature Reviews Genetics, 2003. **4**(5): p. 337-345.

148. Balakrishnan, R., J. Park, K. Karra, B.C. Hitz, G. Binkley, E.L. Hong, J. Sullivan, G. Micklem, and J. Michael Cherry, YeastMine—an integrated data warehouse for *Saccharomyces cerevisiae* data as a multipurpose tool-kit. Database, 2012. **2012**.

149. Simon J Cockell, J.W., Phillip Lord, Claire Wipat, Dmytro Andriychenko, Matthew Pocock, Darren Wilkinson, Malcolm Young and Anil Wipat, An integrated dataset for *in silico* drug discovery. Journal of Integrative Bioinformatics, 2010. **7**(3): p. 116.

150. Ge, H., A.J.M. Walhout, and M. Vidal, Integrating 'omic' information: a bridge between genomics and systems biology. Trends in Genetics, 2003. **19**(10): p. 551-560.

151. Kasif, S. and M. Steffen, Biochemical networks: The evolution of gene annotation. Nature Chemical Biology, 2010. **6**(1): p. 4-5.

152. Hawkins, R.D., G.C. Hon, and B. Ren, Next-generation genomics: an integrative approach. Nature Reviews Genetics, 2010. **11**(7): p. 476-486.

153. An, W. and J.W. Chin, Synthesis of orthogonal transcription-translation networks. Proceedings of the National Academy of Sciences, 2009. **106**(21): p. 8477-8482.

154. Baumbach, J., T. Wittkop, C.K. Kleindt, and A. Tauch, Integrated analysis and reconstruction of microbial transcriptional gene regulatory networks using CoryneRegNet. Nature Protocols, 2009. **4**(6): p. 992-1005.

155. de Hoon, M.J.L., Y. Makita, K. Nakai, and S. Miyano, Prediction of Transcriptional Terminators in *Bacillus subtilis* and Related Species. PLoS Computational Biology, 2005. **1**(3): p. e25.

156. Szallasi, Z., J. Stelling, and V. Periwal, System modeling in cell biology : from concepts to nuts and bolts. 2006: MIT Press.

157. Schlitt, T. and A. Brazma, Modelling in molecular biology: describing transcription regulatory networks at different scales. Philosophical Transactions of the Royal Society B: Biological Sciences, 2006. **361**(1467): p. 483-494.

158. Celniker, S.E., L.A.L. Dillon, M.B. Gerstein, K.C. Gunsalus, S. Henikoff, G.H. Karpen, M. Kellis, E.C. Lai, J.D. Lieb, D.M. MacAlpine, G. Micklem, F. Piano, M. Snyder, L. Stein, K.P. White, and R.H. Waterston, Unlocking the secrets of the genome. Nature, 2009. **459**(7249): p. 927-930.

159. Swinburne, I.A., D.G. Miguez, D. Landgraf, and P.A. Silver, Intron length increases oscillatory periods of gene expression in animal cells. Genes & Development, 2008. **22**(17): p. 2342-2346.

160. Prather, K.L.J. and C.H. Martin, *De novo* biosynthetic pathways: rational design of microbial chemical factories. Current Opinion in Biotechnology, 2008. **19**(5): p. 468-474.

161. Smith, B., The Logic of Biological Classification and the Foundations of Biomedical Ontology, in *Invited Papers from the 10th International Conference in Logic Methodology and Philosophy of Science*. 2005, King's College Publications: Orviedo, Spain. p. 505-520.

162. Goble, C. and R. Stevens, State of the nation in data integration for bioinformatics. Journal of Biomedical Informatics, 2008. **41**(5): p. 687-693.

163. Pasquier, C., Biological data integration using Semantic Web technologies. Biochimie, 2008. **90**(4): p. 584-594.

164. Cohen-Boulakia, S. and U. Leser, Next generation data integration for Life Sciences, in *Data Engineering (ICDE), 2011 IEEE 27th International Conference*. 2011. p. 1366-1369.

165. Sá-Nogueira, I. and L.J. Mota, Negative regulation of L-arabinose metabolism in *Bacillus subtilis*: characterization of the *araR* (*araC*) gene. Journal of Bacteriology, 1997. **179**(5): p. 1598-608.

166. Sierro, N., Y. Makita, M. de Hoon, and K. Nakai, DBTBS: a database of transcriptional regulation in *Bacillus subtilis* containing upstream intergenic conservation information. Nucleic Acids Research, 2008. **36**(suppl 1): p. D93-D96.

167. Epp, C.D., Definition of a gene. Nature, 1997. **389**(6651): p. 537-537.

168. Brachman, R.J., What IS-A Is and Isn't: An Analysis of Taxonomic Links in Semantic Networks. Computer, 1983. **16**(10): p. 30-36.

169. Kanehisa, M., M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi, KEGG for linking genomes to life and the environment. Nucleic Acids Research, 2008. **36**(suppl 1): p. D480-D484.

170. Vallenet, D., L. Labarre, Z. Rouy, V. Barbe, S. Bocs, S. Cruveiller, A. Lajus, G. Pascal, C. Scarpelli, and C. Medigue, MaGe: a microbial genome annotation system supported by synteny results. Nucleic Acids Research, 2006. **34**(1): p. 53-65.

171. von Mering, C., L.J. Jensen, M. Kuhn, S. Chaffron, T. Doerks, B. Kruger, B. Snel, and P. Bork, STRING 7-recent developments in the integration and prediction of protein interactions. Nucleic Acids Research, 2007. **35**(suppl 1): p. D358-362.

172. Cheung, K.-H., A.K. Smith, K.Y.L. Yip, C.J.O. Baker, and M.B. Gerstein, Semantic Web Approach to Database Integration in the Life Sciences, in *Semantic Web*, C.J.O. Baker and K.-H. Cheung, Editors. 2007, Springer US. p. 11-30.

173. Shannon, P., A. Markiel, O. Ozier, N.S. Baliga, J.T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks. Genome Research, 2003. **13**(11): p. 2498-2504.

174. Kohler, J., J. Baumbach, J. Taubert, M. Specht, A. Skusa, A. Ruegg, C. Rawlings, P. Verrier, and S. Philippi, Graph-based analysis and visualization of experimental results with ONDEX. Bioinformatics, 2006. **22**(11): p. 1383-1390.

175. Contrino, S., R.N. Smith, D. Butano, A. Carr, F. Hu, R. Lyne, K. Rutherford, A. Kalderimis, J. Sullivan, S. Carbon, E.T. Kephart, P. Lloyd, E.O. Stinson, N.L. Washington, M.D. Perry, P. Ruzanov, Z. Zha, S.E. Lewis, L.D. Stein, and G. Micklem, modMine: flexible access to modENCODE data. Nucleic Acids Research, 2011. **40**(D1): p. D1082-D1088.

176. Widom, J., Research problems in data warehousing, in *International Conference on Information and Knowledge Management, Proceedings*. 1995. p. 25-30.

177. Smith, R.N., J. Aleksic, D. Butano, A. Carr, S. Contrino, F. Hu, M. Lyne, R. Lyne, A. Kalderimis, K. Rutherford, R. Stepan, J. Sullivan, M. Wakeling, X. Watkins, and G. Micklem, InterMine: a flexible data warehouse system for the integration and analysis of heterogeneous biological data. Bioinformatics, 2012. **28**(23): p. 3163-3165.

178. Lenzerini, M., Data integration: A theoretical perspective, in *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. 2002. p. 233-246.

179. Luciano, J. and R. Stevens, e-Science and biological pathway semantics. BMC Bioinformatics, 2007. **8**(Suppl 3): p. S3.

180. Craddock, T., C.R. Harwood, J. Hallinan, and A. Wipat, e-Science: relieving bottlenecks in large-scale genome analyses. Nature Reviews Microbiology, 2008. **6**(12): p. 948-954.

181. Pautasso, C., O. Zimmermann, and F. Leymann, Restful Web Services vs. "Big" Web Services: Making the Right Architectural Decision, in *Proceedings of the 17th international conference on World Wide Web*. 2008, ACM: Beijing, China. p. 805-814.

182. Kanehisa, M., S. Goto, M. Hattori, K.F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa, From genomics to chemical genomics: new developments in KEGG. Nucleic Acids Research, 2005. **34**(suppl 1): p. D354-D357.

183. Baumbach, J., CoryneRegNet 4.0 - A reference database for corynebacterial gene regulatory networks. BMC Bioinformatics, 2007. **8**(1): p. 429.

184. Curbera, F., M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. Internet Computing, IEEE, 2002. **6**(2): p. 86-93.

185. Grunberg, R., Draft of an RDF-based framework for the exchange and integration of Synthetic Biology data, in *BBF RFC #31*. 2009.

186. Missier, P., N. Paton, and P. Li, Workflows for Information Integration in the Life Sciences, in *Search Computing*, S. Ceri and M. Brambilla, Editors. 2011, Springer Berlin / Heidelberg. p. 215-225.

187. Hull, D., K. Wolstencroft, R. Stevens, C. Goble, M.R. Pocock, P. Li, and T. Oinn, Taverna: a tool for building and running workflows of services. Nucleic Acids Research, 2006. **34**(suppl 2): p. W729-W732.

188. Voigt, C.A., D.M. Wolf, and A.P. Arkin, The *Bacillus subtilis sin* Operon: An Evolvable Network Motif. Genetics, 2005. **169**(3): p. 1187-1202.

189. Bolouri, H. and E. Davidson, Modeling transcriptional regulatory networks. BioEssays, 2002. **24**(12): p. 1118-1129.

190. Barkai, N. and S. Leibler, Robustness in simple biochemical networks. Nature, 1997. **387**(6636): p. 913-917.

191. Silva-Rocha, R. and V. de Lorenzo, Noise and Robustness in Prokaryotic Regulatory Networks. Annual Review of Microbiology, 2010. **64**(1): p. 257-275.

192. Ribeiro, P., F. Silva, and M. Kaiser, Strategies for Network Motifs Discovery, in *2009 Fifth IEEE International Conference on e-Science*. 2009. p. 80-87.

193. Milo, R., S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon, Network Motifs: Simple Building Blocks of Complex Networks. Science, 2002. **298**(5594): p. 824-827.

194. Alon, U., Biological Networks: The Tinkerer as an Engineer. Science, 2003. **301**.

195. Alon, U., Network motifs: theory and experimental approaches. Nature Reviews Genetics, 2007. **8**(6): p. 450-461.

196. Süel, G.M., J. Garcia-Ojalvo, L.M. Liberman, and M.B. Elowitz, An excitable gene regulatory circuit induces transient cellular differentiation. Nature, 2006. **440**(7083): p. 545-550.

197. Mangan, S. and U. Alon, Structure and function of the feed-forward loop network motif. Proceedings of the National Academy of Sciences, 2003. **100**(21): p. 11980-11985.

198. Han, J.-D.J., Understanding biological functions through molecular networks. Cell Research, 2008. **18**(2): p. 224-237.

199. Breitkreutz, B.-J., C. Stark, and M. Tyers, Osprey: a network visualization system. Genome Biology, 2003. **4**(3): p. R22.

200. Pavlopoulos, G., A.-L. Wegener, and R. Schneider, A survey of visualization tools for biological network analysis. Biodata mining, 2008. **1**(1): p. 12.

201. Cruz, I.F. and H. Xiao, The Role of Ontologies in Data Integration. Journal of Engineering Intelligent Systems, 2005. **13**(4): p. 245–252.

202. Lister, A., P. Lord, M. Pocock, and A. Wipat, Annotation of SBML models through rule-based semantic integration. Journal of Biomedical Semantics, 2010. **1**(Suppl 1): p. S3.

203. Taubert, J., K. Sieren, M. Hindle, B. Hoekman, C. Rawlings, and J. Kohler, The OXL format for the exchange of integrated datasets. Journal of Integrative Bioinformatics, 2007. **4**(3): p. 62.

204. Consortium, T.G.O., Creating the Gene Ontology Resource: Design and Implementation. Genome Research, 2001. **11**(8): p. 1425-1433.

205. Weile, J., M. Pocock, S.J. Cockell, P. Lord, J.M. Dewar, E.M. Holstein, D. Wilkinson, D. Lydall, J. Hallinan, and A. Wipat, Customizable views on semantically integrated networks for systems biology. Bioinformatics, 2011. **27**(9): p. 1299-1306.

206. Lysenko, A., M. Platel, K. Pak, J. Taubert, C. Hodgman, C. Rawlings, and M. Saqi, Assessing the functional coherence of modules found in multiple-evidence networks from *Arabidopsis*. BMC Bioinformatics, 2010. **12**(1): p. 203.

207. Goler, J.A., B.W. Bramlett, and J. Peccoud, Genetic design: rising above the sequence. Trends in Biotechnology, 2008. **26**(10): p. 538-544.

208. Blondé, W., V. Mironov, A. Venkatesan, E. Antezana, B. De Baets, and M. Kuiper, Reasoning with bio-ontologies: using relational closure rules to enable practical querying. Bioinformatics, 2011.

209. Magrane, M. and U. Consortium, UniProt Knowledgebase: a hub of integrated protein data. Database, 2011. **2011**.

210. Antoniou, G. and F. van Harmelen, A Semantic Web Primer. 2008: The MIT Press.

211. Allemang, D. and J. Hendler, Semantic Web for the Working Ontologist: Effective Modeling in RDFS and OWL. 2008: Morgan Kaufmann.

212. Horridge, M. OWL Syntaxes. 2010 [cited 03/05/2012]; Available from: http://ontogenesis.knowledgeblog.org/88.

213. Ian, H., Ontologies and the semantic web. Communications of the ACM, 2008. **51**(12): p. 58-67.

214. Pan, J., C.J.O. Baker, and K.-H. Cheung, OWL for the Novice: A Logical Perspective. 2007, Springer US. p. 159-182.

215. Smith, B., Beyond Concepts: Ontology as Reality Representation, in *International Conference on Formal Ontology and Information Systems*, A. Varzi and L. Vieu, Editors. 2004.

216. Kerrien, S., S. Orchard, L. Montecchi-Palazzi, B. Aranda, A. Quinn, N. Vinod, G. Bader, I. Xenarios, J. Wojcik, D. Sherman, M. Tyers, J. Salama, S. Moore, A. Ceol, A. Chatr-aryamontri, M. Oesterheld, V. Stumpflen, L. Salwinski, J. Nerothin, E. Cerami, M. Cusick, M. Vidal, M. Gilson, J. Armstrong, P. Woollard, C. Hogue, D. Eisenberg, G. Cesareni, R. Apweiler, and H. Hermjakob, Broadening the horizon - level 2.5 of the HUPO-PSI format for molecular interactions. BMC Biology, 2007. **5**(1): p. 44.

217. Smith, B., M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L.J. Goldberg, K. Eilbeck, A. Ireland, C.J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R.H. Scheuermann, N. Shah, P.L. Whetzel, and S. Lewis, The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. Nature Biotechnology, 2007. **25**(11): p. 1251-1255.

218. Burge, S., E. Kelly, D. Lonsdale, P. Mutowo-Muellenet, C. McAnulla, A. Mitchell, A. Sangrador-Vegas, S.-Y. Yong, N. Mulder, and S. Hunter, Manual GO annotation of predictive protein signatures: the InterPro approach to GO curation. Database, 2012. **2012**.

219. Natale, D.A., C.N. Arighi, W.C. Barker, J.A. Blake, C.J. Bult, M. Caudy, H.J. Drabkin, P. D'Eustachio, A.V. Evsikov, H. Huang, J. Nchoutmboube, N.V. Roberts, B. Smith, J. Zhang, and C.H. Wu, The Protein Ontology: a structured representation of protein forms and complexes. Nucleic Acids Research, 2010.

220. Meehan, T., A. Masci, A. Abdulla, L. Cowell, J. Blake, C. Mungall, and A. Diehl, Logical Development of the Cell Ontology. BMC Bioinformatics, 2011. **12**(1): p. 6.

221. Eilbeck, K., S. Lewis, C. Mungall, M. Yandell, L. Stein, R. Durbin, and M. Ashburner, The Sequence Ontology: a tool for the unification of genome annotations. Genome Biology, 2005. **6**(5): p. R44.

222. Stevens, R., M. Egaña Aranguren, K. Wolstencroft, U. Sattler, N. Drummond, M. Horridge, and A. Rector, Using OWL to model biological knowledge. International Journal of Human-Computer Studies, 2007. **65**(7): p. 583-594.

223. Horridge, M., Patel-Schneider, P., Manchester syntax for OWL 1.1, in *International Workshop OWL: Experiences and Directions, OWLED 2008*. 2008.

224. Baader, F., I. Horrocks, U. Sattler, D. Hutter, and W. Stephan, Description Logics as Ontology Languages for the Semantic Web, in *Mechanizing Mathematical Reasoning*. 2005, Springer Berlin / Heidelberg. p. 228-248.

225. Rector, A., N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang, C. Wroe, E. Motta, N. Shadbolt, A. Stutt, and N. Gibbins, OWL Pizzas: Practical Experience of Teaching OWL-DL: Common Errors & Common Patterns, in *Engineering Knowledge in the Age of the Semantic Web*. 2004, Springer Berlin / Heidelberg. p. 63-81.

226. Stevens, R. Closing Down the Open World: Covering Axioms and Closure Axioms. 2011 [cited 04/05/2012]; Available from: http://ontogenesis.knowledgeblog.org/1001.

227. Motik, B., R. Shearer, and I. Horrocks, Hypertableau reasoning for description logics. Journal of Artificial Intelligence Research, 2009. **36**(1): p. 165-228.

228. Sirin, E., B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz, Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web, 2007. **5**(2): p. 51-53.

229. Egana, M., Antezana, E., Stevens, R., Transforming the Axiomisation of Ontologies:The Ontology Pre-Processor Language, in *Proceedigns of OWLED 2008 DC OWL:Experiences and Directions*. 2008.

230. Sirin, E., Bulka, B., Smith, M, Terp: Syntax for OWL-friendly SPARQL Queries, in *7th OWL Experiences and Directions Workshop*. 2010: San Francisco

231. Ham, T.S., Z. Dmytriv, H. Plahar, J. Chen, N.J. Hillson, and J.D. Keasling, Design, implementation and practice of JBEI-ICE: an open source biological part registry platform and tools. Nucleic Acids Research, 2012.

232. McDaniel, R. and R. Weiss, Advances in synthetic biology: on the path from prototypes to applications. Current Opinion in Biotechnology, 2005. **16**(4): p. 476-483.

233. Klipp, E., R. Herwig, A. Kowald, C. Wierling, and H. Lehrach, Modeling of Gene Expression, in *Systems Biology in Practice*. 2005, Wiley-VCH Verlag GmbH & Co. KGaA. p. 257-288.

234. Wilkinson, D.J., Stochastic modelling for quantitative description of heterogeneous biological systems. Nature Reviews Genetics, 2009. **10**(2): p. 122-133.

235. Sprinzak, D. and M.B. Elowitz, Reconstruction of genetic circuits. Nature, 2005. **438**(7067): p. 443-448.

236. Barabasi, A.-L. and Z.N. Oltvai, Network biology: understanding the cell's functional organization. Nature Reviews Genetics, 2004. **5**(2): p. 101-113.

237. Zheng, Y. and G. Sriram, Mathematical Modeling: Bridging the Gap between Concept and Realization in Synthetic Biology. Journal of Biomedicine and Biotechnology, 2010. **2010**.

238. Bolouri, H., Computational Modelling Of Gene Regulatory Networks - A Primer. 2008: Imperial College Press.

239. Plotkin, J.B., Transcriptional regulation is only half the story. Molecular Systems Biology, 2010. **6**.

240. Bagh, S., M. Mandal, and D.R. McMillen, Minimal genetic device with multiple tunable functions. Physical Review E, 2010. **82**(2): p. 021911.

241. Alon, U., An Introduction to Systems Biology: Design Principles of Biological Circuits. 2006: Chapman & Hall/CRC.

242. Bintu, L., N.E. Buchler, H.G. Garcia, U. Gerland, T. Hwa, J. Kondev, and R. Phillips, Transcriptional regulation by the numbers: models. Current Opinion in Genetics & Development, 2005. **15**(2): p. 116-124.

243. Hedley, W.J., M.R. Nelson, D.P. Bellivant, and P.F. Nielsen, A short introduction to CellML. Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences, 2001. **359**(1783): p. 1073-1089.

244. Hoops, S., S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, COPASI—a COmplex PAthway SImulator. Bioinformatics, 2006. **22**(24): p. 3067-3074.

245. Garny, A., D. Noble, P.J. Hunter, and P. Kohl, Cellular Open Resource (COR): current status and future directions. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2009. **367**(1895): p. 1885-1905.

246. Krause, F., J. Uhlendorf, T. Lubitz, M. Schulz, E. Klipp, and W. Liebermeister, Annotation and merging of SBML models with semanticSBML. Bioinformatics, 2010. **26**(3): p. 421-422.

247. Vijayalakshmi, C., L. Endler, C. Laibe, C. Li, and N. Le Novère, Curation and annotation for BioModels Database, a resource of published quantitative kinetic models. Nature Precedings, 2009.

248. Novere, N.L., A. Finney, M. Hucka, U.S. Bhalla, F. Campagne, J. Collado-Vides, E.J. Crampin, M. Halstead, E. Klipp, P. Mendes, P. Nielsen, H. Sauro, B.

Shapiro, J.L. Snoep, H.D. Spence, and B.L. Wanner, Minimum information requested in the annotation of biochemical models (MIRIAM). Nature Biotechnology, 2005. **23**(12): p. 1509-1515.

249. Beard, D.A., R. Britten, M.T. Cooling, A. Garny, M.D.B. Halstead, P.J. Hunter, J. Lawson, C.M. Lloyd, J. Marsh, A. Miller, D.P. Nickerson, P.M.F. Nielsen, T. Nomura, S. Subramanium, S.M. Wimalaratne, and T. Yu, CellML metadata standards, associated tools and repositories. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 2009. **367**(1895): p. 1845-1867.

250. Hucka, M., F. Bergmann, S. Hoops, S. Keating, S. Sahle, and D. Wilkinson, The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core (Release 1 Candidate). Nature Precedings, 2010(713).

251. Mirschel, S., K. Steinmetz, M. Rempel, M. Ginkel, and E.D. Gilles, ProMoT: modular modeling for systems biology. Bioinformatics, 2009. **25**(5): p. 687-689.

252. Chandran, D., F.T. Bergmann, and H.M. Sauro, Computer-aided design of biological circuits using TinkerCell. Bioengineered Bugs, 2010. **1**(4): p. 276-283.

253. Clarke, E.J. and C.A. Voigt, Characterization of combinatorial patterns generated by multiple two-component sensors in *E. coli* that respond to many stimuli. Biotechnology and Bioengineering, 2010. **108**(3): p. 666-675.

254. A. O'Malley, M., A. Powell, J.F. Davies, and J. Calvert, Knowledge-making distinctions in synthetic biology. BioEssays, 2008. **30**(1): p. 57-65.

255. Antoine, D., Scaling up synthetic biology: Do not forget the chassis. FEBS Letters, 2012(0).

256. Pleiss, J., The promise of synthetic biology. Applied Microbiology and Biotechnology, 2006. **73**(4): p. 735-739.

257. Westerhoff, H.V. and B.O. Palsson, The evolution of molecular biology into systems biology. Nature Biotechnology, 2004. **22**(10): p. 1249-1252.

258. Henry, C., J. Zinner, M. Cohoon, and R. Stevens, *i*Bsu1103: a new genome-scale metabolic model of *Bacillus subtilis* based on SEED annotations. Genome Biology, 2009. **10**(6): p. R69.

259. Oh, Y.-K., B.O. Palsson, S.M. Park, C.H. Schilling, and R. Mahadevan, Genome-scale Reconstruction of Metabolic Network in *Bacillus subtilis* Based on High-throughput Phenotyping and Gene Essentiality Data. Journal of Biological Chemistry, 2007. **282**(39): p. 28791-28799.

260. Camon, E., M. Magrane, D. Barrell, D. Binns, W. Fleischmann, P. Kersey, N. Mulder, T. Oinn, J. Maslen, A. Cox, and R. Apweiler, The Gene Ontology Annotation (GOA) Project: Implementation of GO in SWISS-PROT, TrEMBL, and InterPro. Genome Research, 2003. **13**(4): p. 662-672.

261. Winnenburg, R., T. Wächter, C. Plake, A. Doms, and M. Schroeder, Facts from text: can text mining help to scale-up high-quality manual curation of gene products with ontologies? Briefings in Bioinformatics, 2008. **9**(6): p. 466-478.

262. Goto, S., S. Kawashima, Y. Okuji, T. Kamiya, S. Miyazaki, Y. Numata, and M. Kanehisa, KEGG/EXPRESSION: A Database for Browsing and Analysing Microarray Expression Data. Genome Informatics, 2000. **11**: p. 222-223.

263. Harwood, C.R. and A. Wipat, Sequencing and functional analysis of the genome of *Bacillus subtilis* strain 168. FEBS Letters, 1996. **389**(1): p. 84-87.

264. Tatusov, R.L., D.A. Natale, I.V. Garkavtsev, T.A. Tatusova, U.T. Shankavaram, B.S. Rao, B. Kiryutin, M.Y. Galperin, N.D. Fedorova, and E.V. Koonin, The COG database: new developments in phylogenetic classification of proteins from complete genomes. Nucleic Acids Research, 2001. **29**(1): p. 22-28.

265. von Mering, C., L.J. Jensen, B. Snel, S.D. Hooper, M. Krupp, M. Foglierini, N. Jouffre, M.A. Huynen, and P. Bork, STRING: known and predicted protein-protein associations, integrated and transferred across organisms. Nucleic Acids Research, 2005. **33**(suppl 1): p. D433-D437.

266. Jensen, L.J., M. Kuhn, M. Stark, S. Chaffron, C. Creevey, J. Muller, T. Doerks, P. Julien, A. Roth, M. Simonovic, P. Bork, and C. von Mering, STRING 8-a global view on proteins and their functional interactions in 630 organisms. Nucleic Acids Research, 2009. **37**(suppl 1): p. D412-D416.

267. Mellor, J.C., I. Yanai, K.H. Clodfelter, J. Mintseris, and C. DeLisi, Predictome: a database of putative functional links between proteins. Nucleic Acids Research, 2002. **30**(1): p. 306-309.

268. Licata, L., L. Briganti, D. Peluso, L. Perfetto, M. Iannuccelli, E. Galeota, F. Sacco, A. Palma, A.P. Nardozza, E. Santonico, L. Castagnoli, and G. Cesareni, MINT, the molecular interaction database: 2012 update. Nucleic Acids Research, 2011. **40**(D1): p. D857-D861.

269. Joshi-Tope, G., M. Gillespie, I. Vastrik, P. D'Eustachio, E. Schmidt, B. de Bono, B. Jassal, G.R. Gopinath, G.R. Wu, L. Matthews, S. Lewis, E. Birney, and L. Stein, Reactome: a knowledgebase of biological pathways. Nucleic Acids Research, 2005. **33**(suppl 1): p. D428-D432.

270. Stark, C., B.-J. Breitkreutz, A. Chatr-aryamontri, L. Boucher, R. Oughtred, M.S. Livstone, J. Nixon, K. Van Auken, X. Wang, X. Shi, T. Reguly, J.M. Rust, A. Winter, K. Dolinski, and M. Tyers, The BioGRID Interaction Database: 2011 update. Nucleic Acids Research, 2011. **39**(suppl 1): p. D698-D704.

271. Bairoch, A., The ENZYME database in 2000. Nucleic Acids Research, 2000. **28**(1): p. 304-305.

272. Dimmer, E.C., R.P. Huntley, Y. Alam-Faruque, T. Sawford, C. O'Donovan, M.J. Martin, B. Bely, P. Browne, W. Mun Chan, R. Eberhardt, M. Gardner, K. Laiho, D. Legge, M. Magrane, K. Pichler, D. Poggioli, H. Sehra, A. Auchincloss, K. Axelsen, M.-C. Blatter, E. Boutet, S. Braconi-Quintaje, L. Breuza, A. Bridge, E. Coudert, A. Estreicher, L. Famiglietti, S. Ferro-Rojas, M. Feuermann, A. Gos,

N. Gruaz-Gumowski, U. Hinz, C. Hulo, J. James, S. Jimenez, F. Jungo, G. Keller, P. Lemercier, D. Lieberherr, P. Masson, M. Moinat, I. Pedruzzi, S. Poux, C. Rivoire, B. Roechert, M. Schneider, A. Stutz, S. Sundaram, M. Tognolli, L. Bougueleret, G. Argoud-Puy, I. Cusin, P. Duek- Roggli, I. Xenarios, and R. Apweiler, The UniProt-GO Annotation database in 2011. Nucleic Acids Research, 2012. **40**(D1): p. D565-D570.

273. Pelchat, M. and J. Lapointe, Aminoacyl-tRNA synthetase genes of *Bacillus subtilis*: organization and regulation. Biochemistry and Cell Biology, 1999. **77**(4): p. 343-347.

274. Tipton, K. and S.a. Boyce, History of the enzyme nomenclature system. Bioinformatics, 2000. **16**(1): p. 34-40.

275. Klipp, E. and W. Liebermeister, Mathematical modeling of intracellular signaling pathways. BMC Neuroscience, 2006. **7**(Suppl 1): p. S10.

276. Ogata, H., S. Goto, W. Fujibuchi, and M. Kanehisa, Computation with the KEGG pathway database. Biosystems, 1998. **47**(1-2): p. 119-128.

277. Dawes, N.L. and J. Glassey, Normalisation of Multicondition cDNA Macroarray Data. Comparative and Functional Genomics, 2007. **2007**.

278. Hermsen, R., S. Tans, and P.R. ten Wolde, Transcriptional Regulation by Competing Transcription Factor Modules. PLoS Computational Biology, 2006. **2**(12): p. e164.

279. Bailey, T.L., M. Boden, F.A. Buske, M. Frith, C.E. Grant, L. Clementi, J. Ren, W.W. Li, and W.S. Noble, MEME Suite: tools for motif discovery and searching. Nucleic Acids Research, 2009. **37**(suppl 2): p. W202-W208.

280. López, D. and R. Kolter, Extracellular signals that define distinct and coexisting cell fates in *Bacillus subtilis.* FEMS Microbiology Reviews, 2010. **34**(2): p. 134-149.

281. Veening, J.-W., W.K. Smits, and O. Kuipers, Bistability, epigenetics, and bet-hedging in bacteria. Annual Review of Microbiology, 2008. **62**(1): p. 193-210.

282. Dixit, M., C.S. Murudkar, and K.K. Rao, *epr* Is Transcribed from a σ^D Promoter and Is Involved in Swarming of *Bacillus subtilis.* Journal of Bacteriology, 2002. **184**(2): p. 596-599.

283. Zheng, G., L.Z. Yan, J.C. Vederas, and P. Zuber, Genes of the *sbo-alb* Locus of *Bacillus subtilis* Are Required for Production of the Antilisterial Bacteriocin Subtilosin. Journal of Bacteriology, 1999. **181**(23): p. 7346-7355.

284. Chai, Y., R. Kolter, and R. Losick, Paralogous antirepressors acting on the master regulator for biofilm formation in *Bacillus subtilis.* Molecular Microbiology, 2009. **74**(4): p. 876-887.

285. Colledge, V.L., M.J. Fogg, V.M. Levdikov, A. Leech, E.J. Dodson, and A.J. Wilkinson, Structure and Organisation of SinR, the Master Regulator of Biofilm Formation in *Bacillus subtilis.* Journal of Molecular Biology, 2011. **411**(3): p. 597-613.

286. Skerker, J.M., B.S. Perchuk, A. Siryaporn, E.A. Lubin, O. Ashenberg, M. Goulian, and M.T. Laub, Rewiring the Specificity of Two-Component Signal Transduction Systems. Cell, 2008. **133**(6): p. 1043-1054.

287. Kim, K.H., D. Chandran, and H.M. Sauro, Toward Modularity in Synthetic Biology: Design Patterns and Fan-out, in *Design and Analysis of Bio-Molecular Circuits*. 2011, Springer.

288. Yu Lin, Zuoshuang Xiang, and Y. He, Towards a Semantic Web application: Ontology-driven ortholog clustering analysis, in *International Conference on Biomedical Ontology*. 2011: Buffalo, NY.

289. Natale, D., C. Arighi, W. Barker, J. Blake, T.-C. Chang, Z. Hu, H. Liu, B. Smith, and C. Wu, Framework for a Protein Ontology. BMC Bioinformatics, 2007. **8**(Suppl 9): p. S1.

290. Noy, N. and Deborah, Ontology Development 101: A Guide to Creating Your First Ontology, in *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*. 2001.

291. Stevens, R. and D. Hull. Defining Definitions. 2010 [cited 04/05/2012]; Available from: http://ontogenesis.knowledgeblog.org/824.

292. Beisswanger, E., V. Lee, J.-J. Kim, D. Rebholz-Schuhmann, A. Splendiani, O. Dameron, S. Schulz, and U. Hahn, Gene Regulation Ontology (GRO): design principles and use cases. Studies in health technology and informatics, 2008. **136**: p. 9-14.

293. Silva-Rocha, R. and V. de Lorenzo, Mining logic gates in prokaryotic transcriptional regulation networks. FEBS Letters, 2008. **582**(8): p. 1237-1244.

294. Boyle, P.M. and P.A. Silver, Harnessing nature's toolbox: regulatory elements for synthetic biology. Journal of The Royal Society Interface, 2009. **6**(Suppl 4): p. S535-S546.

295. Weiss, R., S. Basu, S. Hooshangi, A. Kalmbach, D. Karig, R. Mehreja, and I. Netravali, Genetic circuit building blocks for cellular computation, communications, and signal processing. Natural Computing, 2003. **2**(1): p. 47-84.

296. Horrocks, I., P.F. Patel-Schneider, and F. van Harmelen, From SHIQ and RDF to OWL: the making of a Web Ontology Language. Web Semantics: Science, Services and Agents on the World Wide Web, 2003. **1**(1): p. 7-26.

297. van Hijum, S.A.F.T., M.H. Medema, and O.P. Kuipers, Mechanisms and Evolution of Control Logic in Prokaryotic Transcriptional Regulation. Microbiology and Molecular Biology Reviews, 2009. **73**(3): p. 481-509.

298. Barnard, A., A. Wolfe, and S. Busby, Regulation at complex bacterial promoters: how bacteria use different promoter organizations to produce different regulatory outcomes. Current Opinion in Microbiology, 2004. **7**(2): p. 102-108.

299. Harvie, D.R., C. Andreini, G. Cavallaro, W. Meng, B.A. Connolly, K.-i. Yoshida, Y. Fujita, C.R. Harwood, D.S. Radford, S. Tottey, J.S. Cavet, and N.J. Robinson, Predicting metals sensed by ArsR-SmtB repressors: allosteric interference by a non-effector metal. Molecular Microbiology, 2006. **59**(4): p. 1341-1356.

300. Galdzicki, M., D. Chandran, A. Nielsen, J. Morrison, Cowell M, R. Grunberg, S. Sleight, and H. Sauro, Provisional BioBrick Language (PoBoL), in *BBF RFC #31*. 2009.

301. Brazma, A., M. Krestyaninova, and U. Sarkans, Standards for systems biology. Nature Reviews Genetics, 2006. **7**(8): p. 593-605.

302. Jensen, L.J. and P. Bork, Ontologies in Quantitative Biology: A Basis for Comparison, Integration, and Discovery. PLoS Biology, 2010. **8**(5): p. e1000374.

303. Gonzalez-Beltran, A., B. Tagger, and A. Finkelstein, Federated ontology-based queries over cancer data. BMC Bioinformatics, 2012. **13**(Suppl 1): p. S9.

304. Feiler, P., B. Lewis, S. Vestal, and E. Colbert, An Overview of the SAE Architecture Analysis & Design Language (AADL) Standard: A Basis for Model-Based Architecture-Driven Embedded Systems Engineering, in *Architecture Description Languages*, P. Dissaux, M. Filali-Amine, P. Michel, and F. Vernadat, Editors. 2005, Springer Boston. p. 3-15.

305. Balasubramanian, K., A. Gokhale, G. Karsai, J. Sztipanovits, and S. Neema, Developing applications using model-driven design environments. Computer, 2006. **39**(2): p. 33-40.

306. Henzinger, T. and J. Sifakis, The Embedded Systems Design Challenge, in *FM 2006: Formal Methods*, J. Misra, T. Nipkow, and E. Sekerinski, Editors. 2006, Springer Berlin / Heidelberg. p. 1-15.

307. MacMillen, D., R. Camposano, D. Hill, and T.W. Williams, An industrial view of electronic design automation. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, 2000. **19**(12): p. 1428-1448.

308. Machado, D., R. Costa, M. Rocha, E. Ferreira, B. Tidor, and I. Rocha, Modeling formalisms in Systems Biology. AMB Express, 2011. **1**(1): p. 45.

309. Klein, C., C. Kaletta, and K.D. Entian, Biosynthesis of the lantibiotic subtilin is regulated by a histidine kinase/response regulator system. Applied and Environmental Microbiology, 1993. **59**(1): p. 296-303.

310. Bongers, R.S., J.-W. Veening, M. Van Wieringen, O.P. Kuipers, and M. Kleerebezem, Development and Characterization of a Subtilin-Regulated Expression System in *Bacillus subtilis*: Strict Control of Gene Expression by Addition of Subtilin. Applied and Environmental Microbiology, 2005. **71**(12): p. 8818-8824.

311. Ozbudak, E.M., M. Thattai, I. Kurtser, A.D. Grossman, and A. van Oudenaarden, Regulation of noise in the expression of a single gene. Nature Genetics, 2002. **31**(1): p. 69-73.

312. Bray, D., R.B. Bourret, and M.I. Simon, Computer simulation of the phosphorylation cascade controlling bacterial chemotaxis. Molecular Biology of the Cell, 1993. **4**(5): p. 469-82.

313. Dräger, A., N. Rodriguez, M. Dumousseau, A. Dörr, C. Wrzodek, N. Le Novère, A. Zell, and M. Hucka, JSBML: a flexible Java library for working with SBML. Bioinformatics, 2011. **27**(15): p. 2167-2168.

314. Densmore, D. and S. Hassoun, Design Automation for Synthetic Biological Systems. Design & Test of Computers, IEEE, 2012. **PP**(99): p. 1-1.

315. Swainston, N. and P. Mendes, libAnnotationSBML: a library for exploiting SBML annotations. Bioinformatics, 2009. **25**(17): p. 2292-2293.

316. Cooling, M.T., P. Hunter, and E.J. Crampin, Modelling biological modularity with CellML. IET Systems Biology, 2008. **2**(2): p. 73-79.

317. Bard, J., S. Rhee, and M. Ashburner, An ontology for cell types. Genome Biology, 2005. **6**(2): p. R21.

318. Fabret, C.l., V.A. Feher, and J.A. Hoch, Two-Component Signal Transduction in *Bacillus subtilis*: How One Organism Sees Its World. Journal of Bacteriology, 1999. **181**(7): p. 1975-1983.

319. Courtot, M., N. Juty, C. Knupfer, D. Waltemath, A. Zhukova, A. Drager, M. Dumontier, A. Finney, M. Golebiewski, J. Hastings, S. Hoops, S. Keating, D.B. Kell, S. Kerrien, J. Lawson, A. Lister, J. Lu, R. Machne, P. Mendes, M. Pocock, N. Rodriguez, A. Villeger, D.J. Wilkinson, S. Wimalaratne, C. Laibe, M. Hucka, and N. Le Novère, Controlled vocabularies and semantics in systems biology. Molecular Systems Biology, 2011. **7**.

320. Yaman, F., S. Bhatia, A. Adler, D. Densmore, J. Beal, R. Weiss, and N. Davidsohn, Toward Automated Selection of Parts for Genetic Regulatory Networks, in *3rd International Workshop on Bio-Design Automation*. 2011.

321. Peccoud, J., J.C. Anderson, D. Chandran, D. Densmore, M. Galdzicki, M.W. Lux, C.A. Rodriguez, G.-B. Stan, and H.M. Sauro, Essential information for synthetic DNA sequences. Nature Biotechnology, 2011. **29**(1): p. 22-22.

322. Wimalaratne, S.M., M.D.B. Halstead, C.M. Lloyd, M.T. Cooling, E.J. Crampin, and P.F. Nielsen, A method for visualizing CellML models. Bioinformatics, 2009. **25**(22): p. 3012-3019.

323. Hasty, J., M. Dolnik, V. Rottschäfer, and J.J. Collins, Synthetic gene network for entraining and amplifying cellular oscillations. Physical Review Letters, 2002. **88**(14): p. 148101.

324. Temme, K., R. Hill, T.H. Segall-Shapiro, F. Moser, and C.A. Voigt, Modular control of multiple pathways using engineered orthogonal T7 polymerases. Nucleic Acids Research, 2012.

325. Rao, C.V., Expanding the synthetic biology toolbox: engineering orthogonal regulators of gene expression. Current Opinion in Biotechnology, 2012. **23**(5): p. 689-694.

326. Takami, H., Y. Takaki, G.-J. Chee, S. Nishi, S. Shimamura, H. Suzuki, S. Matsui, and I. Uchiyama, Thermoadaptation trait revealed by the genome sequence of thermophilic *Geobacillus kaustophilus.* Nucleic Acids Research, 2004. **32**(21): p. 6292-6303.

327. Chen, X.H., A. Koumoutsi, R. Scholz, A. Eisenreich, K. Schneider, I. Heinemeyer, B. Morgenstern, B. Voss, W.R. Hess, O. Reva, H. Junge, B. Voigt, P.R. Jungblut, J. Vater, R. Sussmuth, H. Liesegang, A. Strittmatter, G. Gottschalk, and R. Borriss, Comparative analysis of the complete genome sequence of the plant growth-promoting bacterium *Bacillus amyloliquefaciens* FZB42. Nature Biotechnology, 2007. **25**(9): p. 1007-1014.

328. Eppinger, M., B. Bunk, M.A. Johns, J.N. Edirisinghe, K.K. Kutumbaka, S.S.K. Koenig, H. Huot Creasy, M.J. Rosovitz, D.R. Riley, S. Daugherty, M. Martin, L.D.H. Elbourne, I. Paulsen, R. Biedendieck, C. Braun, S. Grayburn, S. Dhingra, V. Lukyanchuk, B. Ball, R. Ul-Qamar, J.r. Seibel, E. Bremer, D. Jahn, J. Ravel, and P.S. Vary, Genome Sequences of the Biotechnologically Important *Bacillus megaterium* Strains QM B1551 and DSM319. Journal of Bacteriology, 2011. **193**(16): p. 4199-4213.

329. Rey, M., P. Ramaiya, B. Nelson, S. Brody-Karpin, E. Zaretsky, M. Tang, A. de Leon, H. Xiang, V. Gusti, I.G. Clausen, P. Olsen, M. Rasmussen, J. Andersen, P. Jorgensen, T. Larsen, A. Sorokin, A. Bolotin, A. Lapidus, N. Galleron, S.D. Ehrlich, and R. Berka, Complete genome sequence of the industrial bacterium *Bacillus licheniformis* and comparisons with closely related *Bacillus* species. Genome Biology, 2004. **5**(10): p. r77.

330. Tang, Y.J., R. Sapra, D. Joyner, T.C. Hazen, S. Myers, D. Reichmuth, H. Blanch, and J.D. Keasling, Analysis of metabolic pathways and fluxes in a newly discovered thermophilic and ethanol-tolerant *Geobacillus* strain. Biotechnology and Bioengineering, 2009. **102**(5): p. 1377-1386.

331. Baumbach, J., On the power and limits of evolutionary conservation—unraveling bacterial gene regulatory networks. Nucleic Acids Research, 2010. **38**(22): p. 7877-7884.

332. Venancio, T. and L. Aravind, Reconstructing prokaryotic transcriptional regulatory networks: lessons from actinobacteria. Journal of Biology, 2009. **8**(3): p. 29.

333. Beckstette, M., R. Homann, R. Giegerich, and S. Kurtz, Fast index based algorithms and software for matching position specific scoring matrices. BMC Bioinformatics, 2006. **7**(1): p. 389.

334. Oberto, J., FITBAR: a web tool for the robust prediction of prokaryotic regulons. BMC Bioinformatics, 2010. **11**(1): p. 554.

335. Novichkov, P.S., D.A. Rodionov, E.D. Stavrovskaya, E.S. Novichkova, A.E. Kazakov, M.S. Gelfand, A.P. Arkin, A.A. Mironov, and I. Dubchak, RegPredict:

an integrated system for regulon inference in prokaryotes by comparative genomics approach. Nucleic Acids Research, 2010. **38**(suppl 2): p. W299-W307.

336. Baumbach, J., T. Wittkop, K. Rademacher, S. Rahmann, K. Brinkrolf, and A. Tauch, CoryneRegNet 3.0-An interactive systems biology platform for the analysis of gene regulatory networks in corynebacteria and *Escherichia coli*. Journal of Biotechnology, 2007. **129**(2): p. 279 - 289.

337. Baumbach, J., S. Rahmann, and A. Tauch, Reliable transfer of transcriptional gene regulatory networks between taxonomically related organisms. BMC Systems Biology, 2009. **3**(1): p. 8.

338. Baumbach, J., K. Brinkrolf, L. Czaja, S. Rahmann, and A. Tauch, CoryneRegNet: An ontology-based data warehouse of corynebacterial transcription factors and regulatory networks. BMC Genomics, 2006. **7**: p. 24.

339. Krawczyk, J., T.A. Kohl, A. Goesmann, J. Kalinowski, and J. Baumbach, From *Corynebacterium glutamicum* to *Mycobacterium tuberculosis*—towards transfers of gene regulatory networks and integrated data analyses with MycoRegNet. Nucleic Acids Research, 2009. **37**(14): p. e97.

340. Dehal, P.S., M.P. Joachimiak, M.N. Price, J.T. Bates, J.K. Baumohl, D. Chivian, G.D. Friedland, K.H. Huang, K. Keller, P.S. Novichkov, I.L. Dubchak, E.J. Alm, and A.P. Arkin, MicrobesOnline: an integrated portal for comparative and functional genomics. Nucleic Acids Research, 2009. **38**(suppl 1): p. D396-D400.

341. Helmann, J.D. and J. C. P. Moran, RNA Polymerase and Sigma Factors, in *Bacillus subtilis and its closest relatives: from genes to cells. ASM Press, Washington, DC*. 2002, Amer Society for Microbiology. p. 289-312.

342. Charoensawan, V., D. Wilson, and S.A. Teichmann, Genomic repertoires of DNA-binding transcription factors across the tree of life. Nucleic Acids Research, 2010. **38**(21): p. 7364-7377.

343. Buescher, J.M., W. Liebermeister, M. Jules, M. Uhr, J. Muntel, E. Botella, B. Hessling, R.J. Kleijn, L. Le Chat, F. Lecointe, U. Mäder, P. Nicolas, S. Piersma, F. Rügheimer, D. Becher, P. Bessieres, E. Bidnenko, E.L. Denham, E. Dervyn, K.M. Devine, G. Doherty, S. Drulhe, L. Felicori, M.J. Fogg, A. Goelzer, A. Hansen, C.R. Harwood, M. Hecker, S. Hubner, C. Hultschig, H. Jarmer, E. Klipp, A. Leduc, P. Lewis, F. Molina, P. Noirot, S. Peres, N. Pigeonneau, S. Pohl, S. Rasmussen, B. Rinn, M. Schaffer, J. Schnidder, B. Schwikowski, J.M. Van Dijl, P. Veiga, S. Walsh, A.J. Wilkinson, J. Stelling, S. Aymerich, and U. Sauer, Global Network Reorganization During Dynamic Adaptations of *Bacillus subtilis* Metabolism. Science, 2012. **335**(6072): p. 1099-1103.

344. Nazina, T.N., T.P. Tourova, A.B. Poltaraus, E.V. Novikova, A.A. Grigoryan, A.E. Ivanova, A.M. Lysenko, V.V. Petrunyaka, G.A. Osipov, S.S. Belyaev, and M.V. Ivanov, Taxonomic study of aerobic thermophilic bacilli: descriptions of *Geobacillus subterraneus* gen. nov., sp. nov. and *Geobacillus uzenensis* sp. nov. from petroleum reservoirs and transfer of *Bacillus stearothermophilus*, *Bacillus thermocatenulatus*, *Bacillus thermoleovorans*, *Bacillus kaustophilus*, *Bacillus thermodenitrificans* to *Geobacillus* as the new combinations *G.*

stearothermophilus, *G. thermocatenulatus*, *G. thermoleovorans*, *G. kaustophilus*, *G. thermoglucosidasius* and *G. thermodenitrificans.* International Journal of Systematic and Evolutionary Microbiology, 2001. **51**(2): p. 433-46.

345. Guinebretiere, M.-H., S. Auger, N. Galleron, M. Contzen, B. De Sarrau, M.-L. De Buyser, G. Lamberet, A. Fagerlund, P.E. Granum, D. Lereclus, P. De Vos, C. Nguyen-The, and A. Sorokin, *Bacillus cytotoxicus* sp. nov. is a new thermotolerant species of the *Bacillus cereus* group occasionally associated with food poisoning. International Journal of Systematic and Evolutionary Microbiology, 2012.

346. Waldron, K.J. and N.J. Robinson, How do bacterial cells ensure that metalloproteins get the correct metal? Nature Reviews Microbiology, 2009. **7**(1): p. 25-35.

347. Nicolas, P., U. Mäder, E. Dervyn, T. Rochat, A. Leduc, N. Pigeonneau, E. Bidnenko, E. Marchadier, M. Hoebeke, S. Aymerich, D. Becher, P. Bisicchia, E. Botella, O. Delumeau, G. Doherty, E.L. Denham, M.J. Fogg, V. Fromion, A. Goelzer, A. Hansen, E. Härtig, C.R. Harwood, G. Homuth, H. Jarmer, M. Jules, E. Klipp, L. Le Chat, F. Lecointe, P. Lewis, W. Liebermeister, A. March, R.A.T. Mars, P. Nannapaneni, D. Noone, S. Pohl, B. Rinn, F. Rügheimer, P.K. Sappa, F. Samson, M. Schaffer, B. Schwikowski, L. Steil, J. Stülke, T. Wiegert, K.M. Devine, A.J. Wilkinson, J. Maarten van Dijl, M. Hecker, U. Völker, P. Bessières, and P. Noirot, Condition-Dependent Transcriptome Reveals High-Level Regulatory Architecture in *Bacillus subtilis.* Science, 2012. **335**(6072): p. 1103-1106.

348. Suarez, M., G. Rodrigo, J. Carrera, and A. Jaramillo, Computational Design in Synthetic Biology, in *Synthetic Biology*, M. Schmidt, A. Kelle, A. Ganguli-Mitra, and H. Vriend, Editors. 2010, Springer Netherlands. p. 49-63.

349. Ng, A., B. Bursteinas, Q. Gao, E. Mollison, and M. Zvelebil, Resources for integrative systems biology: from data through databases to networks and dynamic system models. Briefings in Bioinformatics, 2006. **7**(4): p. 318-330.

350. Kim, T.Y., H.U. Kim, and S.Y. Lee, Data integration and analysis of biological networks. Current Opinion in Biotechnology, 2010. **21**(1): p. 78-84.

351. Sirota-Madi, A., T. Olender, Y. Helman, C. Ingham, I. Brainis, D. Roth, E. Hagi, L. Brodsky, D. Leshkowitz, V. Galatenko, V. Nikolaev, R. Mugasimangalam, S. Bransburg-Zabary, D. Gutnick, D. Lancet, and E. Ben-Jacob, Genome sequence of the pattern forming *Paenibacillus vortex* bacterium reveals potential for thriving in complex environments. BMC Genomics, 2010. **11**(1): p. 710.

352. Rodrigo, G., T.E. Landrain, and A. Jaramillo, *De novo* automated design of small RNA circuits for engineering synthetic riboregulation in living cells. Proceedings of the National Academy of Sciences, 2012. **109**(38): p. 15271-15276.

353. von Dassow, G., E. Meir, E.M. Munro, and G.M. Odell, The segment polarity network is a robust developmental module. Nature, 2000. **406**(6792): p. 188-192.

354. Garg, A., J.J. Lohmueller, P.A. Silver, and T.Z. Armel, Engineering synthetic TAL effectors with orthogonal target sites. Nucleic Acids Research, 2012. **40**(15): p. 7584-7595.

Table 1: Binding sequence predictions for *Bacillus amyloliquefaciens* FZB42. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
RBAM_000010	*dnaA*	RBAM_000010	*dnaA*	CAAACTGTGGATAAGTT
RBAM_000560	*purR*	RBAM_034060	*glyA*	AATACTTACGCTTACTACTTTTACGAAAAGGACATATTACAAG CACAAAAG
RBAM_000560	*purR*	RBAM_006840	*purE*	TCTAAACACGAACATTAAAAGAAAGAATTTTTATATCGTTCGA TAATGTCG
RBAM_000560	*purR*	RBAM_006790	*pbuG*	TCAATTAACGAATAAACCAAATTTTTTTAACGGGATTGTTCGT TTTTTTTA
RBAM_000560	*purR*	RBAM_037450	*purA*	ACAAATTCCGCTTGTTTGTATCTAACTATTCTAAATTTACAAG CCTAAATG
RBAM_000560	*purR*	RBAM_020260	*xpt*	GATTTTTGTGCTTATTAGCTTTGGCTAAGAAACATTTTCAAGC CTTTATAA
RBAM_000560	*purR*	RBAM_000560	*purR*	CTAATTTAGGCATACAATTCAATATAATAAAAATTTAATAAGC CTAAAACC
RBAM_000560	*purR*	RBAM_029180	*guaC*	TATTTTTCTGCTTGTAAAATTAAAATAAATTAAGATAATAAGC GAAAATCC
RBAM_000560	*purR*	RBAM_022650	*nusB*	ACACTTTAGGCTTACTTTAATTATGCCAAACACTTTTATAAGC CGTAATCC
RBAM_000560	*purR*	RBAM_015300	*pyrR*	GCTCTTTTTGGTAACTAAAGTTACCAGTCCGCGAAGTACGGCC TTATAATC
RBAM_000560	*purR*	RBAM_027090	*ytiP*	AGCTATTTTGCTTATTATATCTACTACAAAACTTTTTTACAAGC AGTAACC
RBAM_001080	*ctsR*	RBAM_031850	*clpP*	AAACTGGAAATAACTG
RBAM_001080	*ctsR*	RBAM_001080	*ctsR*	GTCAAATATAGTCAAA
RBAM_001230	*sigH*	RBAM_030160	*fumC*	AAAAGGGTTTCATCTGTTACTGGCGAATAATACTCAG
RBAM_001230	*sigH*	RBAM_022550	*spo0A*	CAAAGATATACCACTATTATTGGTAAATATGGATTTT
RBAM_001230	*sigH*	RBAM_026900	*ytxG*	AAAGGGATTTACCGAGAAACGGAGAAATAGTATACAT
RBAM_001230	*sigH*	RBAM_033830	*ureA*	AACCAATTATAAGTGATTTAAAAATTTATAAGATATT
RBAM_001230	*sigH*	RBAM_015110	*ftsA*	AAGAGGATATACATAGAATATAACGAATAGTTTCATT
RBAM_001230	*sigH*	RBAM_034290	*spo0F*	TAAAGGAAATGTGAAAATCAAACAGAATACATAAACA
RBAM_001230	*sigH*	RBAM_019170	*yojL*	ATAGGAGGTTTTTCGTTATATGAAGAAAAAAATCGCA
RBAM_001230	*sigH*	RBAM_013770	*kinA*	GAAGGAGAATGCTTATTTTTTAGCGAATCAATCTAGT
RBAM_001230	*sigH*	RBAM_032460	*yvyD*	GCAGGGATCAGAAGGGGTAAAAGAGAAATAGTTACAT
RBAM_001230	*sigH*	RBAM_000580	*spoVG*	GCAGGAATTTAAAAAGAATCGTGGAATTGTTAGACTA
RBAM_001230	*sigH*	RBAM_009680	*lytE*	GAAACAGATAAAAACTTTAAAAAAGAGACACATAAAA
RBAM_001230	*sigH*	RBAM_009630	*lytF*	ATAAGAAAAAAGGACTATTTCTATTAATATGATAAAC
RBAM_002260	*sigW*	RBAM_027110	*ythP*	GAAACTGAAACTTTTTTTATACATGTCCGTATGAA
RBAM_002260	*sigW*	RBAM_003150	*yceC*	TTTATGAAACTTTGATATAATAAAAGACGTATATC
RBAM_002260	*sigW*	RBAM_022130	*yqjL*	TTTGAGAAACGATTGCCGAAGCTTCTCCGTCTTCC
RBAM_002260	*sigW*	RBAM_026460	*sppA*	AAAAATGAATCTTTTTTTACGGGGGAACGTACGAA
RBAM_002260	*sigW*	RBAM_025890	*ysdB*	AAAAATGAAACCTTTTGTTTTGTTTTTCGTATTAC
RBAM_002260	*sigW*	RBAM_002260	*sigW*	AAAAATGAAACCTTTTAAAACAATCAACGTATAGA
RBAM_002260	*sigW*	RBAM_019030	*yozO*	TTCTTGAAACCTTTTCTTCTGTTGAACCGTATTAC
RBAM_002260	*sigW*	RBAM_006580	*pspA*	GAAAATGAAACTTTTTTCGAAAAAAACAGTATAGA
RBAM_002260	*sigW*	RBAM_006140	*pbpE*	AATATTGAAACTTCTAAAGCGGCCGTTCGTATAAA
RBAM_002260	*sigW*	RBAM_004930	*ydbS*	GAAAGTGAATCTTTTTTGCAAATCAGCCGTATTTG
RBAM_002260	*sigW*	RBAM_016790	*pbpX*	ATGTTACAACTTTTTTCACCCTTCACTCGTCTAAC
RBAM_002260	*sigW*	RBAM_000340	*xpaC*	AGAAGAGGAACTTGTTTAGACATGGAATGTAATAG
RBAM_002260	*sigW*	RBAM_036110	*yxjI*	TGTTTGAATACTTTTCGTTTGTTTTTTCGTACAAT
RBAM_002260	*sigW*	RBAM_035680	*ywaC*	TGTAAAGAACCTGACCGCGGACTGACCCGTCTTAT
RBAM_002260	*sigW*	RBAM_023690	*yqeZ*	AAAAATGAAACTAAAGTTCATTTGTTACGTATATA
RBAM_002260	*sigW*	RBAM_011490	*yjbC*	AAATTGAAACCATGTTAAAGGATCACACGTCTTAA
RBAM_002260	*sigW*	RBAM_011160	*fosB*	TACGATTCCTATTACTCTTCATTAAAAAAGAAGTA
RBAM_002810	*ycbA*	RBAM_002800	*ybgJ*	AAGAATGATTTTGTCGTATTTTGTATGATTCTGTAGATGTCATT
RBAM_002990	*lmrA*	RBAM_002990	*lmrA*	AAATCAAGATAATAGACCAGTCACTATATTTTAAATTT
RBAM_002990	*lmrA*	RBAM_032230	*yxaF*	ATATGATTTTAGCTAACAAAAAGCATTATACGTCCTCC

RBAM_004470	*lrpC*	RBAM_004470	*lrpC*	TTTTCCTCTATACTA
RBAM_005070	*sigB*	RBAM_018740	*yoaB*	GATCTCTTCCTATATGAAGAATGGAAATATACGCAAAA
RBAM_005070	*sigB*	RBAM_026340	*ytkL*	AGTTTTTTCGTTTGTCAAAAAGGGAACTTGATAAATAT
RBAM_005070	*sigB*	RBAM_032960	*ywtG*	GGTTTGAAACGGACGGAAACAGGCAAAAGAAGCAGTAG
RBAM_005070	*sigB*	RBAM_019060	*dhaS*	GAAGTATGAGAAATGGGCCGCAGTCGTGTAAAAACTGT
RBAM_005070	*sigB*	RBAM_007060	*opuE*	GGTTTCGAGAGTTGATTGATCGGGGATACTTTGTTAGG
RBAM_005070	*sigB*	RBAM_032820	*gtaB*	GGTGTGAATTTTAATCAATTGTGGTAAAGTGGAAAAAG
RBAM_005070	*sigB*	RBAM_026150	*phoP*	GGATAAAATGAAATCATAAAGTGATATGCAGCACATAA
RBAM_005070	*sigB*	RBAM_033170	*alsS*	AGATAAAGAAGCTAGCGCAAAGACGACACAAAATTTTT
RBAM_005070	*sigB*	RBAM_025560	*trxA*	AAGATTATTTCCGGCGGGTTCGGGAGAAAATAATATAA
RBAM_005070	*sigB*	RBAM_017940	*alsT*	AAAAATAATATTATGGGATTAAATTTACATTTCATTTG
RBAM_005070	*sigB*	RBAM_037400	*yycF*	GGTTTCAAAAAAATATTATTTTTTAAAAGAAAATAAAA
RBAM_005070	*sigB*	RBAM_012970	*ohrB*	ATGTTTAAAAATATGAGCAGGGGCAATATATACAGTAA
RBAM_005070	*sigB*	RBAM_006270	*ydjK*	AGATTAACTATTGATCATATAGTGAAAATAAGTTGATT
RBAM_005070	*sigB*	RBAM_025370	*ilvB*	AATTATAAATGTCTAGGGAAACATGGTCATTACTTTCG
RBAM_005070	*sigB*	RBAM_032460	*yvyD*	CGTTTCAGCAGGGATCAGAAGGGGTAAAAGAGAAATAG
RBAM_005070	*sigB*	RBAM_036810	*csbC*	ATGTTTATATTTTTCTCTTCGGGGAATGAAATAGTTAT
RBAM_005070	*sigB*	RBAM_012870	*ykgA*	GAGTAAGTAGACATGGGATAATTTGTTATGAAACTAGC
RBAM_005070	*sigB*	RBAM_006150	*ydhK*	TGTTTTACTGAAGAAAGGAAAGGGAAAGTGAAAAATCA
RBAM_005070	*sigB*	RBAM_036780	*iolA*	TTTTCCCGATCAAAAAAAGTATGGTAAAAATAATGAAA
RBAM_005070	*sigB*	RBAM_024710	*relA*	TGTTGAATATGCAAAAGAAAAAGAAATTGATTTAGTTG
RBAM_005070	*sigB*	RBAM_031850	*clpP*	TGTTTGAACTTTTTGTTTTATGGGAAAAATGGTGTTAG
RBAM_005070	*sigB*	RBAM_001140	*yacL*	CGCTTAAACCTGTCGGAATCCGGGTATATTATTGTTAG
RBAM_005070	*sigB*	RBAM_000610	*ctc*	GGTTTAAATCCCAGTTATTGTGGGTATTGTTCTAATAG
RBAM_005070	*sigB*	RBAM_001080	*ctsR*	GGTTTTATGCTTGTGCAAAGTGGAAATAATAGAGATAG
RBAM_005070	*sigB*	RBAM_004890	*ydbP*	CCTTTTTATTTTATATTAAAATGGCATGGAAAAGGAGA
RBAM_005070	*sigB*	RBAM_035970	*yxkO*	GGTTTAAAATCTCTTCTTGTTTGGAATATAAATAGGAG
RBAM_005070	*sigB*	RBAM_035940	*katX*	GTTTTAAAAAACGGCTCTGCCGGGCATATTGTTACCGT
RBAM_005070	*sigB*	RBAM_004730	*gsiB*	TTGTTTAGCAGTTCTGAAATCGGGAAACCAACAACCAA
RBAM_005070	*sigB*	RBAM_012260	*yjgB*	CGTTTATCTCATGAAACATACGGGTATGAGCGGTAAAA
RBAM_005070	*sigB*	RBAM_031320	*araE*	TTTATTCTATTATTCGCGGGAGGGTAAAATTCATGAAG
RBAM_005070	*sigB*	RBAM_036170	*katE*	AGTTTATATGAAGAACGCCGCGGGTAAATGTGCTGTAG
RBAM_005070	*sigB*	RBAM_035670	*gspA*	AGGTTTACTTTTTTAAAAAAGGGAAAGACATTTGTACA
RBAM_005070	*sigB*	RBAM_023600	*cdd*	TTAATAAAATGTAACGGTTTTTACCCGCACTTTTTGGT
RBAM_005070	*sigB*	RBAM_004410	*ydaD*	TGGTTCCATACCGGCGATTCCGGATATGTGGATGAAGA
RBAM_005070	*sigB*	RBAM_023530	*yqxD*	GGGTTTTTGGGTATGCAAATAGGGAATAAACAAAAAGG
RBAM_005070	*sigB*	RBAM_011490	*yjbC*	TTGTTTAAACAATCGGAAATGGGGTATATCTAAAAGTG
RBAM_005070	*sigB*	RBAM_034980	*rocA*	TGAGCACACATTACAAAATACGGGGAGGAATTCTAATG
RBAM_005070	*sigB*	RBAM_008690	*csbB*	GGTTTAAACGTCATTAAAAAAAGGAAAAGCCTTACTGA
RBAM_005070	*sigB*	RBAM_009090	*katA*	AAAAGTTTAGGTTTGGGAAAAAAGCTATTATTTTTTATT
RBAM_005070	*sigB*	RBAM_028160	*gbsA*	AAACAATTCAGTAAAGAAATACAAGAATTAGAATTGCT
RBAM_005070	*sigB*	RBAM_022810	*yqhQ*	TGAGTAAAATCTAAAGAAACGGGGTATACTGCATGTAG
RBAM_005070	*sigB*	RBAM_004160	*gabD*	TGGCTTTACTATAGCAGAGAAAGGAATGTTCAATATGC
RBAM_005070	*sigB*	RBAM_027660	*dps*	TGTTTTAACACACGATATATAGGGTATATCAATAGTAA
RBAM_005070	*sigB*	RBAM_030330	*yvrE*	GCTGTTTAACCGTTATTACCGGGGAACGAATACGAAAG
RBAM_005070	*sigB*	RBAM_023100	*yqgZ*	TGTTTAATGAAAAAAAGTTCAGGGTACTTATGTTAACA
RBAM_005070	*sigB*	RBAM_023080	*yqhA*	GATCTTATCATTAACGACTGAAGGAATTGATGAAATAT
RBAM_005070	*sigB*	RBAM_023060	*yqxL*	GGTTTAGCCTGCTTCATTTACGGTAATAAAACAAGAAA
RBAM_005070	*sigB*	RBAM_003460	*ycgO*	AAATAGCCAATAAAGCGGATGTCTGCCTGCAGTTTAGA
RBAM_005070	*sigB*	RBAM_010580	*yhxC*	TATAAGAATTAAAAAGGTTATTCTGAGACAAGTATGTA
RBAM_005070	*sigB*	RBAM_026900	*ytxG*	ATGTTTATCATTTTAAGAATTGGGTATATACCACTATA
RBAM_005070	*sigB*	RBAM_003360	*nadE*	TGATTCATTTTCAGCATGAACGGGAAATGGATCATAAA
RBAM_005070	*sigB*	RBAM_008110	*yfkJ*	GGTTTTTGATCCTCAAAATGCGGGGAAAGAATAGGGAG
RBAM_005070	*sigB*	RBAM_007610	*yhxD*	AACAAAAGGGAAATGGTTAGCCTTTGACGTGAATGTGC
RBAM_005070	*sigB*	RBAM_022300	*bmrU*	AGTTTATGACCCTTTAAAAACGGGAAACTTGTTAAGGA
RBAM_005070	*sigB*	RBAM_007590	*yfmT*	CGTATATACGCTGCCTGAAATCGGAATGGATGAAGAAA
RBAM_005070	*sigB*	RBAM_019330	*yodE*	GAGGGCCGATATTTCGGCTATGTCTGTTGCCAATTTTG
RBAM_006420	*rex*	RBAM_033170	*alsS*	AAGAGTGTATAGTGAATAATATCACAAATTACT

RBAM_006420	*rex*	RBAM_025370	*ilvB*	AGATTTAACAGATTATTAATATTTTTACGACAA
RBAM_006420	*rex*	RBAM_036010	*cydA*	TGCTTAAACACTTTGTAACTCGTTTATATAGTA
RBAM_006420	*rex*	RBAM_003290	*ldh*	TTTTCAAACACTTCATAACGTGTTATATTTACA
RBAM_006530	*gutR*	RBAM_006540	*gutB*	TAAAAAGTACAGTGCGGCTGTCCTTTTAT
RBAM_007790	*citT*	RBAM_007810	*citM*	AACAAAATGAACGAAAAGGTTATAAAAAATTAAAAATCTTTA AAATCCAAA
RBAM_007790	*citT*	RBAM_036180	*citH*	ATCACTCATGTCAGTTCCCCCTTTTAAAACGAAACATTCTACA GCACATTT
RBAM_007990	*treR*	RBAM_035290	*sacP*	AAAATCTTTCCCCCTATGTAGTACTTAATAGT
RBAM_007990	*treR*	RBAM_007970	*treP*	AGTCATAAACATGTATATACAAGTTTGTAAGA
RBAM_008340	*acoR*	RBAM_008300	*acoA*	CGAGACAAAGGGAACAGATCGAGACAAAATGAGACACCTGT CTCAAACTGTCTCCATTGTAAAA
RBAM_008820	*perR*	RBAM_018460	*fenA*	ATTTAATTGTAATTATTTCAC
RBAM_008820	*perR*	RBAM_035710	*dltA*	AATATAAATGCTATTCAATTT
RBAM_008820	*perR*	RBAM_003680	*srfAC*	GACTTTAGTCAAAATTGATGA
RBAM_008820	*perR*	RBAM_003660	*srfAB*	ACTACTTGAACTATTTAATTA
RBAM_008820	*perR*	RBAM_003650	*srfAA*	TATTATAAAAATTTTTATTTA
RBAM_008820	*perR*	RBAM_021640	*fur*	ATCAGTTTATAATAATTATAG
RBAM_008820	*perR*	RBAM_013620	*ykvW*	ACTATTAATAATAGTTTTTCT
RBAM_008820	*perR*	RBAM_036960	*ahpC*	AATCTTAATAATAACTTTCTA
RBAM_008820	*perR*	RBAM_025230	*hemA*	ACTATGTTATAATCATTATAA
RBAM_008820	*perR*	RBAM_035940	*katX*	ACTTTTAGCCAAAATTTTTTG
RBAM_008820	*perR*	RBAM_036170	*katE*	CTTATTTCAAAATGTGTATGA
RBAM_008820	*perR*	RBAM_008820	*perR*	AATATTTGTAATATTTCATTA
RBAM_008820	*perR*	RBAM_011500	*spxA*	CCTTTTTTAAAGTCATTTCAA
RBAM_008820	*perR*	RBAM_009090	*katA*	ATTATTTTATAATCATTATAA
RBAM_009530	*glpP*	RBAM_009540	*glpF*	GATGGAGACCCGGAGACCCA
RBAM_009530	*glpP*	RBAM_002610	*glpT*	GATGGAGATGAGGAGAACCA
RBAM_009760	*sigM*	RBAM_033710	*ywoA*	AAGTAAGACAAATGAAAAAATACTAATAAAAATTATGGTT
RBAM_009760	*sigM*	RBAM_009760	*sigM*	GCACGTTGAAATTTGGAAAGAATACGCACATATTGTATCT
RBAM_009760	*sigM*	RBAM_011500	*spxA*	TTCACACTTATAAAGATGAAAAATGGTACGGAGAATCATA
RBAM_010230	*hpr*	RBAM_003840	*yclF*	GTATAAATTATATTGACACA
RBAM_010230	*hpr*	RBAM_010500	*aprE*	GATATACCTAAATAGAAATA
RBAM_010230	*hpr*	RBAM_014550	*nprE*	CTTACGACCAAATAATATTG
RBAM_010600	*comK*	RBAM_025120	*comC*	GCTGAAAATATATTTTC
RBAM_010600	*comK*	RBAM_012450	*rapA*	TTTTGGGGGTAAATTAT
RBAM_010600	*comK*	RBAM_016780	*recA*	GATCAAAAATAAAGTTT
RBAM_010600	*comK*	RBAM_023890	*comEA*	GATTAAAATGTGATCTT
RBAM_010600	*comK*	RBAM_004600	*rapH*	TAAGAAAAAATCTTTTG
RBAM_010600	*comK*	RBAM_003610	*nucA*	TCTTTATATAAAATTGT
RBAM_010600	*comK*	RBAM_010790	*addB*	TTTTGCTATAAAAAATT
RBAM_010600	*comK*	RBAM_010600	*comK*	CTTTAAAAGGTGATTTT
RBAM_010600	*comK*	RBAM_023050	*comGA*	GCATCAAAAAAAATATT
RBAM_010600	*comK*	RBAM_034530	*rapF*	CTTTCGAACAATAAAGA
RBAM_010600	*comK*	RBAM_019730	*rapA1*	TTATAATGTATTAGTCT
RBAM_010600	*comK*	RBAM_003080	*rapJ*	TTTTAGAAACAAACGAG
RBAM_010600	*comK*	RBAM_019340	*yodF*	TCTTAAGAAAAAAAAGA
RBAM_010600	*comK*	RBAM_038120	*trmE*	TCTTAGCGTAAAAAACG
RBAM_010600	*comK*	RBAM_038010	*engD*	CTTTACGAAAAAAGAGT
RBAM_010600	*comK*	RBAM_014000	*rok*	AGAAAATAAGCAATTTT
RBAM_012750	*htrA*	RBAM_030110	*yvtA*	ACATCCTTTCAACGTCATTATACTAGTTTTAACACACTCAAGC CGCATTTTCATAATTTCACATATTCTTTTCATTTTTATCCCACA ATGTTTGTTTATGATGTGTATAAGGG
RBAM_012750	*htrA*	RBAM_037350	*yyxA*	GTTTCAGTTATTTGATACCGATCCGAAAATCCCGACGCCGCTG TGCGCCGTGTAATTTTTCCAAATGAATCATGATTATTAATCCG CGATCTGACCCACACTATAAGCAAATC
RBAM_012750	*htrA*	RBAM_012750	*htrA*	TCCGATGGAAATTGTCAATATAAAGCGCCGCGCGTCCGATTTC GTCTTTTTCACAATTTCCCACAATCTTTTCATTATTATCCCACA CTTTTTGTTTAAGATGGAGTCAGTAA
RBAM_013130	*tnrA*	RBAM_013130	*tnrA*	GCAGTCTTTTGGACTGT
RBAM_013130	*tnrA*	RBAM_024530	*glnQ*	AGAGAGATTTTATAACA
RBAM_013130	*tnrA*	RBAM_011430	*oppA*	ACTGTCTTTTTGACTGG

RBAM_013130	*tnrA*	RBAM_003760	*yckI*	ACAGTTTATACTTTGGT
RBAM_013130	*tnrA*	RBAM_003540	*nasA*	TGTCGCATTATCTTACA
RBAM_013130	*tnrA*	RBAM_003530	*nasB*	ACATTCTATTACGCTGT
RBAM_013130	*tnrA*	RBAM_022260	*yqiZ*	AGAGACTTTAATAGTCT
RBAM_013130	*tnrA*	RBAM_003000	*yccC*	TCTTTTATTTTCATACG
RBAM_013130	*tnrA*	RBAM_033680	*nrgA*	ACAGTCTATTGGAATGT
RBAM_013130	*tnrA*	RBAM_033270	*ywrD*	TGTGAGATTTTCTTTCA
RBAM_013130	*tnrA*	RBAM_033170	*alsS*	ACTTTCTTCTTTAAAGT
RBAM_013130	*tnrA*	RBAM_017940	*alsT*	ACAATTTTCTAGATTGT
RBAM_013130	*tnrA*	RBAM_025370	*ilvB*	GCCATTTTGTTTAGTTT
RBAM_013130	*tnrA*	RBAM_017250	*glnR*	TGTTAAGATTCCTTACA
RBAM_013130	*tnrA*	RBAM_036070	*yxkC*	ACCGCTTATTTAATTGC
RBAM_013130	*tnrA*	RBAM_007720	*pel*	ACATTATTTAACCTTGT
RBAM_013130	*tnrA*	RBAM_019340	*yodF*	TGTGATCTTATCTGACA
RBAM_013130	*tnrA*	RBAM_034080	*ywlF*	ACAGTCTTTTAGACGTT
RBAM_013130	*tnrA*	RBAM_018620	*gltA*	TGTCAGATTTTATGACC
RBAM_013220	*sigI*	RBAM_013220	*sigI*	CAGAGAAAACCCCCTTAATTCTGCATGGGGGCACGAAATCAT GTATAGTACGTC
RBAM_013910	*ccpC*	RBAM_026180	*citZ*	AAGAATTTGTTATGT
RBAM_013910	*ccpC*	RBAM_017800	*citB*	GATATTTACTTATGT
RBAM_013910	*ccpC*	RBAM_009700	*citA*	AATAATTTTATATGA
RBAM_014590	*ylaC*	RBAM_014560	*ylaA1*	TTCCGATCTGTAAAACTTTTTTTATAGCCGTTTGTCTAAACGGG TGAGGCCTTAA
RBAM_015150	*sigE*	RBAM_022450	*mmgA*	CACTCATAACTCGGTCAGCCGGTTCATATATTGTAACAGG
RBAM_015150	*sigE*	RBAM_033910	*spoIID*	AAGTCATATTAGCTTGTCCCTGCCCATAAGATGAAGTAGA
RBAM_015150	*sigE*	RBAM_014730	*ctaA*	AAAACAGGTCAAACAATTCTTCCACTTACATTTTACTTAT
RBAM_015150	*sigE*	RBAM_007550	*yfnE*	TGAGGATGATAACAAATAAATGATCATATACAGGAAGTGA
RBAM_015150	*sigE*	RBAM_033700		TAGAAATAAAATATACTTGTCACATTACGAAAAAAGATAC
RBAM_015150	*sigE*	RBAM_021650	*spoIIM*	TCTGTCATGTTTTCTCTTTTCTTTCATACAATCTACTAAA
RBAM_015150	*sigE*	RBAM_010060	*yhaX*	AAAGAAAATAATATACTCTTACATACAAGAGAAAGTTGAA
RBAM_015150	*sigE*	RBAM_001830	*ybaN*	TCGTTATATTCATCTCCGCCGCGGCATACGATGTAATCAA
RBAM_015150	*sigE*	RBAM_020960	*spoIVA*	ACAGGAATGAACCTTTTTCCCGCGCATACAAATAGGGAGA
RBAM_015150	*sigE*	RBAM_026210	*ytvI*	GTATTCATATCCTTTTTTTTGACGAATAAACTAATGTAAT
RBAM_015150	*sigE*	RBAM_001790	*cwlD*	TAATCATAATTTCTTCCCTTGTCCCATACTTATTAATAAC
RBAM_015150	*sigE*	RBAM_021330	*dacB*	GTATTCATAAGTCATGGACATGCGCATAAACTTGTACAAA
RBAM_015150	*sigE*	RBAM_026150	*phoP*	TCAAAAATGTGATGAAAGTATTGTCATAAGAAAAAAGGGA
RBAM_015150	*sigE*	RBAM_013600	*ykvU*	CGAAAATATTTTCAGAACTTGTCTCATATGATGATGGAAG
RBAM_015150	*sigE*	RBAM_038040	*yyaD*	TATGAATGATTGCCGCTGTCACTCAATATAGTAGCAATAT
RBAM_015150	*sigE*	RBAM_026000	*ytxC*	TACGTCTATTTTAAAAACATCCCCCATATACTTGTAACAG
RBAM_015150	*sigE*	RBAM_014010	*yknT*	GAGGAATAGCCGTTGATCCTTTCCCATATTGTAGTGTTAA
RBAM_015150	*sigE*	RBAM_018390	*yngJ*	ATGGAATGAATAAAGACCCCCATTAATAAAATGTAGAAAG
RBAM_015150	*sigE*	RBAM_025450	*gerM*	CCGGTCTATTTCCCGGACAGGGCTCGTATACATCATAGTA
RBAM_015150	*sigE*	RBAM_013470	*ykvI*	TTCGTCTTTTCGGGCGACTTGTCTCATAAGCATATATAAA
RBAM_015150	*sigE*	RBAM_037400	*yycF*	CAAAGTTTTTTTATAATAAAAAATTTTCTTTTATTTTTTG
RBAM_015150	*sigE*	RBAM_018230	*yxjC*	CAGGAAATAAATGAAAGAAAAAAGAATACATATTTATAAA
RBAM_015150	*sigE*	RBAM_024890	*safA*	TTATCATCATGCCCAGTCCATCGGCATATAGTGTGAAGAA
RBAM_015150	*sigE*	RBAM_024780	*spoVB*	TAGAAATAACATATACTAAGGTCCATCAAACTCTTCCGTC
RBAM_015150	*sigE*	RBAM_025170	*spoVID*	TTATCATATTTTTCCGGATTGCGGCATACACCTTTAGTGA
RBAM_015150	*sigE*	RBAM_013100	*ydhD*	TTGGAATCGGACCGGATGATCATTCATAAAGATAAGAAGA
RBAM_015150	*sigE*	RBAM_000690	*yabP*	TTGTTCTAAACAAGGCTCCCGCCTCATACAATGCAGTAAT
RBAM_015150	*sigE*	RBAM_029450	*yunB*	AAATACTATGTCCCCTCTTACAAGCATACATTGTGATATG
RBAM_015150	*sigE*	RBAM_025030	*spoIVFA*	AGCGCATATTTGCAATGGACAAGACATAGGATGTACAAAC
RBAM_015150	*sigE*	RBAM_004970	*ydcC*	TCTGCATATTAGCCGGAACCCCTTCATATATTTGATAGTG
RBAM_015150	*sigE*	RBAM_004950	*ydcA*	GTGATAGTTTATATACTTCCCCAAGGCCGATTATACGTCT
RBAM_015150	*sigE*	RBAM_016870	*cotE*	AAGTAAAGTTTCTGGGCAGGGCTGCATACAATGAAACAGA
RBAM_015150	*sigE*	RBAM_009670	*phoA*	AAGATAGATAAAATACACAGAGAAAAAATTTCAAAAATAG
RBAM_015150	*sigE*	RBAM_009660	*spoVR*	ATTTCATCTTTTGAGAGGAGGGGTCATACATTATAAATAA
RBAM_015150	*sigE*	RBAM_017220	*spoVK*	CCATTCTCCATCACCTCATTACAGCATATGATGATATTCA

RBAM_015150	*sigE*	RBAM_023830	*spoIIP*	TGGTTCTACCTCCTCTAGCTTGTTCATAGAGTAATTACTA
RBAM_015150	*sigE*	RBAM_017040	*nucB*	AAGTATAGTTGAAGAATTCCTCCTACACTTACTTACGCAA
RBAM_015150	*sigE*	RBAM_009260	*yhbH*	TAAATCTTTTCGCCGGACTTGTTTCATAAACAGGACCAAG
RBAM_015150	*sigE*	RBAM_009250	*prkA*	AAAGCATGTATTATCACGTCACCGCATAGATTGTAACAAA
RBAM_015150	*sigE*	RBAM_010950	*asnO*	AAATCAACTATCTGCGTTTCTTTGCATAAACTGGGTAAAG
RBAM_015150	*sigE*	RBAM_022760	*spoIIIAA*	TGAGACTACTTCCGTCAGCTAGAGCATATAGTGTAACAAA
RBAM_015150	*sigE*	RBAM_027560	*asnB*	TTAGAAAATTACACTTATGGAGGCTATACTATGTGTGGAT
RBAM_015150	*sigE*	RBAM_002920	*cwlJ*	TCTGTCATCACTGGGCCGCGGCTGAATATGATGAAGGGAG
RBAM_015150	*sigE*	RBAM_003320	*ycgF*	ATATCATAGCACGGCCCATTCATGAATAACTTGTACAAGT
RBAM_015150	*sigE*	RBAM_014890	*ylbJ*	TGGTCTAAATTGAACCCCTATGCTCGTATATTAGTACAGG
RBAM_015160	*sigG*	RBAM_030180	*gerAA*	ATAGTATATCATTTTTTTAACAGGAAAAGATAACCT
RBAM_015160	*sigG*	RBAM_022560	*spoIVB*	GGTTATAAATCAAGCGCAGGAAGGCAAAAGTATAGA
RBAM_015160	*sigG*	RBAM_010340	*pbpF*	TGACATAATTCGAAACAATATTTCCCATACTACAAG
RBAM_015160	*sigG*	RBAM_002580	*ybeC*	CCGCTTTAATGCGATAAAGATTGACAAAACTATTTT
RBAM_015160	*sigG*	RBAM_019380	*ctpA*	TATTATTACATTCTTATTTTTCCACCACAAAAGAAC
RBAM_015160	*sigG*	RBAM_021590	*dacF*	GGCGTATAAAACCACCGGATTTGGAAAAAATAAAAA
RBAM_015160	*sigG*	RBAM_021550	*spoVAA*	CGGGAAAAACTCATTGTCTATCTGAGATATTATAAA
RBAM_015160	*sigG*	RBAM_001810	*gerD*	TATGTATAAATCCATTCCGATGAATCATATTAAAAA
RBAM_015160	*sigG*	RBAM_032920	*gerBA*	CCAGAATAAAATTCCTTGTTTGTCGCAAATTAAAAT
RBAM_015160	*sigG*	RBAM_013700	*spIB*	AAAAAGTATAAGCTTTTTTTCCCCCCTTGTTATTGA
RBAM_015160	*sigG*	RBAM_021080	*sleB*	CGTGCATAAAAATAACCTCGTGACAGACAATACGGG
RBAM_015160	*sigG*	RBAM_006230	*yveA*	CAAAAAACAAAAAGACTTATACTCCAAACTTACACA
RBAM_015160	*sigG*	RBAM_024860	*bofC*	ACTCTTTATATTGCGGTGTTAACATAAAAGTAATAA
RBAM_015160	*sigG*	RBAM_020470	*ponA*	AACAATAATATAGGCGAAAGCCTTTCCAAATAAAGG
RBAM_015160	*sigG*	RBAM_032380	*yvjB*	TTACATGAATTCTTTGCAATTTACACATACTATCTC
RBAM_015160	*sigG*	RBAM_020230	*ydfR*	AATGAATAAAATTAAATGTAAGGGCAATAATTTTTC
RBAM_015160	*sigG*	RBAM_023840	*gpr*	TCGCATGATTCGTTTTGTAAATGGCCACACTATGTA
RBAM_015160	*sigG*	RBAM_009490	*yhcV*	GGAAATATTACTCTGTTTAGGGGCTCGTATTATTTA
RBAM_015160	*sigG*	RBAM_009440	*yhcQ*	TTTGTAAAGATTGTCTTCACTGGCGCAAAATAGTAA
RBAM_015160	*sigG*	RBAM_030720	*yvaB*	TCAGTATAAAAAAAACATTCATATTTCATAATAAATG
RBAM_015160	*sigG*	RBAM_027820	*yteA*	ATTTCTTATTATAAACCTTTTTGCGCAAATGAATAA
RBAM_015160	*sigG*	RBAM_003870	*gerKA*	CGTGCATAATCTTTTGCCGTACAGGAAGAATATGAA
RBAM_015160	*sigG*	RBAM_004170	*glcU*	CGGGAATAATCGGTTCGATTCCAGACAAAATAACAG
RBAM_015160	*sigG*	RBAM_010780	*ydfS*	TGTTTTAAAAAATATCGTTTTTGACCAAAATAAAGA
RBAM_015160	*sigG*	RBAM_034630	*ywhE*	ATGTTTAAAAAACCCCCTTTTTTCACATAATAATAG
RBAM_015300	*pyrR*	RBAM_015320	*pyrB*	ATAAAACCTTTTAAAGAAAGTCCAGAGAGGCTTGGAAGGGTTATAACGGGAAGA
RBAM_015300	*pyrR*	RBAM_015300	*pyrR*	GAATAGATTCTTTAACACAGTCCAGAGAGGCTGAGAAGGATAACGGATAGGACA
RBAM_015710	*fapR*	RBAM_009770	*yhdO*	ATAGAGATCGGCGGAGGAAAAATATAATTATTAGTACCAGGTAATAAAA
RBAM_015710	*fapR*	RBAM_011730	*fabI*	TTAATTCTAGTCCATGATTTTGATACAGAAATTCTGCTAATGTCCTCCG
RBAM_015710	*fapR*	RBAM_015710	*fapR*	CATTCCAAACTATGAGATTATATACTACTATTAGTACCTAGTCTTAATT
RBAM_015710	*fapR*	RBAM_011330	*fabHA*	TATTGCCAACAGAAAAAAAATAGGGTAGAATTAGTACCTGATACTAATA
RBAM_015710	*fapR*	RBAM_010400	*fabHB*	GCGGATGCTGCTTGGGTTTTTTTATTAAAATTAGTACCAAGTAATAATA
RBAM_016000	*codY*	RBAM_010600	*comK*	ATTAATTTTTATAATCTTTAGACAA
RBAM_016000	*codY*	RBAM_018460	*fenA*	ATATATGATATAAGATTTTTTACAA
RBAM_016000	*codY*	RBAM_033170	*alsS*	AAAGACGACACAAAATTTTTTGAAA
RBAM_016000	*codY*	RBAM_032510	*hag*	AAAAATTTTCTTCAGAATGACGCTG
RBAM_016000	*codY*	RBAM_025370	*ilvB*	AAAAATGATAAAAACTTTTAATATT
RBAM_016000	*codY*	RBAM_012770	*dppA*	TAGAATATTCATAATTCAATATCGA
RBAM_016000	*codY*	RBAM_036400	*hutP*	TGTCAATGATCAATAGTCTTATAAA
RBAM_016000	*codY*	RBAM_035710	*dltA*	AGGAATTTTGAATATGATGCAAAAA
RBAM_016000	*codY*	RBAM_003680	*srfAC*	TAACGTGACAAAAACAGATAACAGA
RBAM_016000	*codY*	RBAM_003660	*srfAB*	CGCGGACTCGTATTTCTCTTAGAAA
RBAM_016000	*codY*	RBAM_003650	*srfAA*	AATAATATTTTTAAAAATAAATTGC
RBAM_016310	*sigD*	RBAM_028160	*gbsA*	CAATTTAATTATAACAATTTTTGTAATTTAAAAATAAATT

RBAM_016310	*sigD*	RBAM_003660	*srfAB*	GCTAGAAAGTATAGTCGACTTTGTTTACTTTCTTCTGTAG
RBAM_016310	*sigD*	RBAM_003650	*srfAA*	ATTTTCATATAAAGTGAACGTAAGTAGATATATAGAAATA
RBAM_016310	*sigD*	RBAM_016010	*flgB*	TGTTTAAGGAATAGCTTAACCTTTTGGAATTTAGAGTAAT
RBAM_016310	*sigD*	RBAM_010560	*hemAT*	TCCTTCATTCAAGTTCCTTCCATACCGATAAAAAGAAAAA
RBAM_016310	*sigD*	RBAM_010500	*aprE*	ACTATAAGTAATAGTAATAAAAATATTTTACCAAAGTGTC
RBAM_016310	*sigD*	RBAM_007590	*yfmT*	TAGTTCAATTTTCTCACCTTAAGACCGATACAAAAGATGA
RBAM_016310	*sigD*	RBAM_019170	*yojL*	CCGTAAAGAGAATTGTATGAAAAACAGACAAGAAAGGTGG
RBAM_016310	*sigD*	RBAM_013800	*cheV*	TACATTCACTTTTTCCGTTTACAGCCGATATATACAGTAC
RBAM_016310	*sigD*	RBAM_019060	*dhaS*	GCTAACTAAAATGGCCTCCACAATAATATTTTACTGACAT
RBAM_016310	*sigD*	RBAM_013710	*mcpC*	TTATTCATTTTCATGACCGAAAGACCGATATAACTCTCAG
RBAM_016310	*sigD*	RBAM_018460	*fenA*	GTTTTTCATTATAGCCATTATATGAAGGAAAAAAAATGTA
RBAM_016310	*sigD*	RBAM_013450	*motA*	ACCCTAAAGTTCCATGCACTGCCACCGATATTAACCATAG
RBAM_016310	*sigD*	RBAM_032590	*yvyF*	GAGCTAAATCATTCTTTTTTTCTGCCGATATAATTGATAG
RBAM_016310	*sigD*	RBAM_032510	*hag*	AATATAAACAATAGAAACGGCCATCCGATATATATAGTGT
RBAM_016310	*sigD*	RBAM_036780	*iolA*	AGTTTTTTTCATACCATTTTTATTACTTTTTTAAAACTAA
RBAM_016310	*sigD*	RBAM_009680	*lytE*	CAGATAAAAACTTTAAAAAAGAGACACATAAAATAGATAG
RBAM_016310	*sigD*	RBAM_009630	*lytF*	ATGATAAACGATTGCAGACAAGTGCCGATATAAAGATTAT
RBAM_016310	*sigD*	RBAM_035710	*dltA*	GATGGAATAGATAGTATTATTAAAGAAGGAGTAAAGTGTT
RBAM_016310	*sigD*	RBAM_035660	*epr*	CAGGAAAAAAAGAATTCCTCACTGCCGATATATAAAGAAA
RBAM_016310	*sigD*	RBAM_036070	*yxkC*	CGTATAAAAAATATGAACGATCTGCCGATACTAAAAAAGG
RBAM_016310	*sigD*	RBAM_023510	*rpoD*	GGCTCAAGAAATTGTTGCATTGAACCGATCATTAAAGTAA
RBAM_016310	*sigD*	RBAM_028320	*mcpB*	TGTTAACAAAATGTGAAGGCTGAGCCGATATATAAGGTAT
RBAM_016310	*sigD*	RBAM_034980	*rocA*	GAAGCAGGCGAAAGCCGTCTTCTAAAAGAATAAAAATAGT
RBAM_017250	*glnR*	RBAM_003540	*nasA*	TGTCGCATTATCTTACA
RBAM_017250	*glnR*	RBAM_003530	*nasB*	ACATTCTATTACGCTGT
RBAM_017250	*glnR*	RBAM_017250	*glnR*	TGTTAAGATTCCTTACA
RBAM_017340	*xylR*	RBAM_017350	*xylA*	AAGTTAGTTTGTTTGGGCAACAAACTAATGTGCAA
RBAM_017340	*xylR*	RBAM_017320	*xynP*	AAGTTAGTTTGTTCGGTCAACAAACTAATAAAACC
RBAM_017650	*lexA*	RBAM_023070	*yqhB*	TGGGCTTACATGGAAGTAGAGA
RBAM_017650	*lexA*	RBAM_023060	*yqxL*	AGAGATGAAGGTACATTCGGGT
RBAM_017650	*lexA*	RBAM_021830	*yqjW*	AAAACAGAACAAATGTTCCCGG
RBAM_017650	*lexA*	RBAM_010150	*yhaO*	TTTGCAGAACGTGCATTCGTAT
RBAM_017650	*lexA*	RBAM_010040	*yhaZ*	GAAAGAGAACATACATTCTCAT
RBAM_017650	*lexA*	RBAM_026280	*dnaE*	AAAAAAGAACATTTGTTTCTTA
RBAM_017650	*lexA*	RBAM_007010	*pcrA*	ATGATAGAACATATGTTTGATA
RBAM_017650	*lexA*	RBAM_037680	*dinB*	ACTCCTTGCATGCAAGACATAG
RBAM_017650	*lexA*	RBAM_025550	*uvrC*	GGAGAAAAACAAATGTTCGTGA
RBAM_017650	*lexA*	RBAM_017900	*parE*	TTCGCTTGAGTACAAGAATAAG
RBAM_017650	*lexA*	RBAM_024850	*ruvA*	CAAAGCGAACATACGTTAACTT
RBAM_017650	*lexA*	RBAM_017650	*lexA*	TATTCTTACAAACAAGCGAAAA
RBAM_017650	*lexA*	RBAM_032310	*uvrB*	GTAGCTTGAAATCAAGCATAAA
RBAM_017650	*lexA*	RBAM_012500	*xkdA*	AAAAAAGAACATATGTTCGAAA
RBAM_017650	*lexA*	RBAM_009780	*yhdP*	CAAAAGGAGTGAAGGTTCTAAT
RBAM_017650	*lexA*	RBAM_016780	*recA*	AAATCCGAATATGCGTTCGCTT
RBAM_017650	*lexA*	RBAM_011840	*yjcD*	TTTTCCTCATGTACTTTCTTTT
RBAM_017650	*lexA*	RBAM_017070	*aprX*	ATAAACGAACAAAAGTTCCATC
RBAM_017650	*lexA*	RBAM_000060	*gyrB*	CTCGCTAACATATAAGTAAATC
RBAM_017650	*lexA*	RBAM_035320	*vpr*	AGATACTAGCAAACGTTTCATT
RBAM_017650	*lexA*	RBAM_010700	*yhjE*	TTCTCTTGGAAACAAGAATAAA
RBAM_018630	*gltC*	RBAM_018620	*gltA*	AGAGTAAAACTCTA
RBAM_018630	*gltC*	RBAM_018630	*gltC*	ATCTCAAAATGAGA
RBAM_019300	*yodB*	RBAM_011500	*spxA*	GAGACTGTTCACACTTACTTTTTTATAGTATAATACCTATAAA GATTCCT
RBAM_020590	*birA*	RBAM_018290	*bioW*	ATAAGTGTTAACCTAAAATTTATTTTGGTTAACATTAATT
RBAM_021240	*sigX*	RBAM_002700	*pssA*	ATGAAACTTTTTATGTCCGCAATCGTTATATGTAA
RBAM_021240	*sigX*	RBAM_033710	*ywoA*	GTATGAACTGCGTGTCATTATGTCAGTGCAAAGTA
RBAM_021240	*sigX*	RBAM_022130	*yqjL*	GAGAAACGATTGCCGAAGCTTCTCCGTCTTCCCTA
RBAM_021240	*sigX*	RBAM_033580	*rapD*	GTCTTCTTTGCCTATCGCTACCCCTTATTAAAGAA
RBAM_021240	*sigX*	RBAM_032800	*lytR*	TTGTAACCTTTTTTTAAGAATTTCGTCTATTACAT

RBAM_021240	*sigX*	RBAM_021240	*sigX*	ATGTAACTTTTCAACGCATTTTTACGACAAAAAAG
RBAM_021240	*sigX*	RBAM_018460	*fenA*	TTGTAATTATTTCACTTTCTTATCCTCTTAAATTG
RBAM_021240	*sigX*	RBAM_016790	*pbpX*	TTACAACTTTTTTCACCCTTCACTCGTCTAACATC
RBAM_021240	*sigX*	RBAM_035710	*dltA*	TTGAAACTTTTTGTTAATCTAATCGTCATATACTG
RBAM_021240	*sigX*	RBAM_011490	*yjbC*	TTGAAACCATGTTAAAGGATCACACGTCTTAATAG
RBAM_021240	*sigX*	RBAM_008690	*csbB*	CTGTAACAAAAAAAGGGTTTAAACGTCATTAAAAA
RBAM_021240	*sigX*	RBAM_003680	*srfAC*	TAGAACTCTACCTACTTTACCCTCTGTAGAAACTG
RBAM_021240	*sigX*	RBAM_003660	*srfAB*	ATCTGACGCTTGATGAACTTGATAAATTAATGGAA
RBAM_021240	*sigX*	RBAM_003650	*srfAA*	ATGAAACTTTTCACCCATTTTTCGGTGATAAAAAC
RBAM_021260	*resD*	RBAM_028210	*hmp1*	TGACGACTTTTTTAAAG
RBAM_021260	*resD*	RBAM_003530	*nasB*	AAAAGTTTTCTTCTATT
RBAM_021260	*resD*	RBAM_034460	*fnr*	GTAAGTGTTCTAACAAT
RBAM_021260	*resD*	RBAM_026710	*hmp*	GTCAGTGTTATATGAAT
RBAM_021260	*resD*	RBAM_017170	*nrdI*	TATGAATTTTTTATAAA
RBAM_021260	*resD*	RBAM_036010	*cydA*	TGACGAATTTGTGAAAC
RBAM_021560	*sigF*	RBAM_009090	*katA*	AAAAAGCTATTATTTTTATTAATAAAATATTAGTAA
RBAM_021560	*sigF*	RBAM_022810	*yqhQ*	ATGGTTTTAAAACGCTTCAGGCAGACGAAATAAATA
RBAM_021560	*sigF*	RBAM_027640	*ytkD*	GCTCGTCGACCTATTGGATACTTTGAAAAATAAATT
RBAM_021560	*sigF*	RBAM_034630	*ywhE*	TATGTTTAAAAAACCCCCTTTTTTCACATAATAATA
RBAM_021560	*sigF*	RBAM_030180	*gerAA*	CATAGTATATCATTTTTTTAACAGGAAAAGATAACC
RBAM_021560	*sigF*	RBAM_022560	*spoIVB*	TAAAAAAACAAGGTAAAGAGATTGTCACATAGGTTA
RBAM_021560	*sigF*	RBAM_010570	*yhfW*	CATGTATGAACAGAGTCTTATTGGAAAAATTAAGAA
RBAM_021560	*sigF*	RBAM_014810	*ylbB*	TCCTTTTAAAAATGCCCTTTCCTGACAATCATAGTA
RBAM_021560	*sigF*	RBAM_010340	*pbpF*	ACGATCATATAGTTTTCTCACCATATTCAAAGATAA
RBAM_021560	*sigF*	RBAM_033730	*spoIIQ*	AGTGTTCAGAATGTTGCTGAGGTGATGAACAAATGA
RBAM_021560	*sigF*	RBAM_034130	*spoIIR*	CCGGTTTAAACTTTTTCCGTCATGTCCATAATGGGA
RBAM_021560	*sigF*	RBAM_026440	*ytfI*	TCTCTATATCGTCATTGCGTTTACGCCCAAAAAGCA
RBAM_021560	*sigF*	RBAM_021590	*dacF*	GGCGTATAAAACCACCGGATTTGGAAAAAATAAAAA
RBAM_021560	*sigF*	RBAM_032920	*gerBA*	TCCAGAATAAAATTCCTTGTTTGTCGCAAATTAAAA
RBAM_021560	*sigF*	RBAM_038060	*yyaC*	TAAGCATAAAAAAAGGAAGCGCTGGATATACTGTAA
RBAM_021560	*sigF*	RBAM_024860	*bofC*	AACGGTTAAGCACCGGCATCTTTGGTCAAACTAGCA
RBAM_021560	*sigF*	RBAM_020470	*ponA*	CCCTTGGATACTCACCGAGAATTGATTATATTAGTA
RBAM_021560	*sigF*	RBAM_024410	*yrrR*	CAAATTGTTACTTTGTGGAGCAGGTTTTACACCTAC
RBAM_021560	*sigF*	RBAM_035940	*katX*	GACGCTTACTTCACTGATCCGTATGCGATATCAGGC
RBAM_021560	*sigF*	RBAM_023840	*gpr*	AATGTAAAGTATTTTTCGTTGATGGGTAAATTCGCA
RBAM_021560	*sigF*	RBAM_029120	*yuiC*	GCGACTAAAAAGGTCTGCAACTGAAAAAATGGCGGG
RBAM_021560	*sigF*	RBAM_036170	*katE*	TATCTTAAAACCGTTACTCCCCTAGTTTGTCTGGAC
RBAM_021560	*sigF*	RBAM_003870	*gerKA*	CGTGCATAATCTTTTGCCGTACAGGAAGAATATGAA
RBAM_021640	*fur*	RBAM_030060	*yusV*	TTGATTAACTATTACTATAAGTCTCAGTAATTAAAGA
RBAM_021640	*fur*	RBAM_034370	*ywjA*	TGTGTATAGTATAATTGAGAAATATTATCATTAAGAA
RBAM_021640	*fur*	RBAM_026720	*yraA*	AACACTGACTGTTCATATGACTGTTTCTTCACCAAAT
RBAM_021640	*fur*	RBAM_008040	*yfkM*	TATCTAAAGTACTACTAAGAACATTTTTCTACTTTTT
RBAM_021640	*fur*	RBAM_013920	*ykuN*	ATTTTTTATTGACAATGAAAATCATTATCATTTAAAG
RBAM_021640	*fur*	RBAM_002130	*ybbB*	TTGGTACAATTTAAATGATAACAGTTATCATTTAAAT
RBAM_021640	*fur*	RBAM_036560	*yxeB*	CTGCTATATTAATAATGATAATCATTATCACTGAAAT
RBAM_021640	*fur*	RBAM_032060	*yvcC*	GGAGGCAGATGAATTTGATAATATTGACTAATAGAGT
RBAM_021640	*fur*	RBAM_029160	*yumC*	ATTGTATGTTATAATTAATAACAATTTTCAATTAGGA
RBAM_021640	*fur*	RBAM_029060	*yuiI*	GGTGTCTTTTGATATTGAAAATCATTATCAATAAGCA
RBAM_021640	*fur*	RBAM_029050	*dhbA*	AGTATCAACTATTACTATTAGTAATAGTTATATCTTT
RBAM_021640	*fur*	RBAM_030440	*fhuD*	AAACAAGAATGGTAATGATAATGATTATCAATGACTT
RBAM_021640	*fur*	RBAM_008550	*yfhC*	CCGTTATATTATAAATGATAATCATTTTCATTTGGAT
RBAM_021640	*fur*	RBAM_030410	*fhuC*	CATATAAACGACTACTTTTGCTTTATCTCCTTCCTCT
RBAM_021640	*fur*	RBAM_004050	*yclN*	AACAGCAACTATTACTATTAGTAATGCTCACCCTATA
RBAM_021640	*fur*	RBAM_008400	*yfiB*	GTTTGTTTATTTTATTTATATGTATGTATAACTTGTT
RBAM_021640	*fur*	RBAM_003480	*ycgT*	GAAAGTAACTATAACTAAGAGTTAATTTTTACAAACT
RBAM_021640	*fur*	RBAM_010510	*yhfQ*	ATTTTATAAAAATAATGATAATGAATATCATTGATAC
RBAM_022550	*spo0A*	RBAM_000750	*spoIIE*	TTTTGACAAAATCC
RBAM_022550	*spo0A*	RBAM_035710	*dltA*	ATTAGACATATTTT

RBAM_022550	*spo0A*	RBAM_003680	*srfAC*	ACGTGACAAAAACA
RBAM_022550	*spo0A*	RBAM_003660	*srfAB*	ACTATAACAGCCTA
RBAM_022550	*spo0A*	RBAM_003650	*srfAA*	ATTCGACAGCTTTT
RBAM_022550	*spo0A*	RBAM_015140	*spoIIGA*	CCTCGACAAATAAA
RBAM_022550	*spo0A*	RBAM_014230	*kinC*	TTATATACATCTTT
RBAM_022550	*spo0A*	RBAM_018460	*fenA*	TATACGACAGCTTT
RBAM_022550	*spo0A*	RBAM_028820	*yuxH*	TTTAGACAAATTTC
RBAM_022550	*spo0A*	RBAM_013770	*kinA*	TTTTGACCAAGTTC
RBAM_022580	*ahrC*	RBAM_007590	*yfmT*	AATAGTAATATATGTG
RBAM_022580	*ahrC*	RBAM_019060	*dhaS*	ATTTATAAAACTCCTT
RBAM_022580	*ahrC*	RBAM_037250	*rocD*	AATGTAAAAATTAGCA
RBAM_022580	*ahrC*	RBAM_036780	*iolA*	ATGAAAAAATTTTGAT
RBAM_022580	*ahrC*	RBAM_028160	*gbsA*	AATTTAAAAATAAATT
RBAM_022580	*ahrC*	RBAM_004150	*gabT*	ACCATGAAAAGTAGTA
RBAM_022580	*ahrC*	RBAM_011220	*argD*	AGGAACAAAAATCGTC
RBAM_022580	*ahrC*	RBAM_011190	*argC*	TAATTAAAAATAAGTA
RBAM_022840	*mntR*	RBAM_004660	*mntH*	AATTTGCATCTAGGAAACT
RBAM_022930	*sinR*	RBAM_035660	*epr*	CAGATGTCTTAAAGCATTTTCTCC
RBAM_022930	*sinR*	RBAM_022960	*yqxM*	GTATGTCAAGAAATTTCTCTTGCC
RBAM_022930	*sinR*	RBAM_010500	*aprE*	GTTTGGCAAGCTGGGTCGTTTGTT
RBAM_022930	*sinR*	RBAM_031670	*epsA*	TTGTTCATTATAAGAAATTTTTCG
RBAM_023410	*zur*	RBAM_003550	*yciC*	ATTTAGCATTAGTAAGAT
RBAM_023410	*zur*	RBAM_003110	*ycdH*	CTTTAGCAGTAGTAAGAC
RBAM_023510	*rpoD*	RBAM_006710	*gabP*	TTTCAACTGCCTGTGGCATTTATGATAGAATAATA
RBAM_023510	*rpoD*	RBAM_025870	*abnA*	CATCGGGATGATTATGAGAATACGGTAAAATTGAT
RBAM_023510	*rpoD*	RBAM_037830		GTATAAGTAAAATGAATAAAAAGAATATAATGTAT
RBAM_023510	*rpoD*	RBAM_025860	*araA*	GAATTGATATTATTTTAACAAGCATGTTTATTTTC
RBAM_023510	*rpoD*	RBAM_019170	*yojL*	TTTTGTGAAATGTTCTCAATTTATGTAAAATGTGT
RBAM_023510	*rpoD*	RBAM_021460	*ppiB*	TATATAAAAGGGTCTATAAGTACAAGAAGAAATAA
RBAM_023510	*rpoD*	RBAM_013850	*ykuF*	ATTTTAATACTTACTAATAAGTAAGTTATTTTTCC
RBAM_023510	*rpoD*	RBAM_021440	*sipS*	TTGCCCTCTGTGCAGGGATTCATGGTAAAATAAAA
RBAM_023510	*rpoD*	RBAM_019130	*sucA*	AAAATAAAATTTTACTAAATTAAAAAAGCCATAAC
RBAM_023510	*rpoD*	RBAM_032960	*ywtG*	TCGTCGCTTTGTAAAAAATAATTAGTAAAATAATG
RBAM_023510	*rpoD*	RBAM_020940	*folE*	TTTTAGATGCACTTAAAAACGGTGATAAAATCCAA
RBAM_023510	*rpoD*	RBAM_018630	*gltC*	AGTTTTCAAAACAATATAAACAATATATAATTTAG
RBAM_023510	*rpoD*	RBAM_018620	*gltA*	GATTTAATATATAACAAATATAACAAAACTTTTGA
RBAM_023510	*rpoD*	RBAM_014230	*kinC*	TAGTCAATAATGCAGTCAACAATGTTACAATATAT
RBAM_023510	*rpoD*	RBAM_019060	*dhaS*	CGAATAACATCGGCAAAAAAGAACACTTAACGTTA
RBAM_023510	*rpoD*	RBAM_025710	*pheS*	GGTTGCGAAAAGCCTTGGATTTCACTATAATAGAT
RBAM_023510	*rpoD*	RBAM_007060	*opuE*	GAGATAAAATAGGATTTAGCCCCTCTTTCTCTTTG
RBAM_023510	*rpoD*	RBAM_032880	*tagA*	GGAGTAATATAGTGTATCGGGACTTTTCCAATTCG
RBAM_023510	*rpoD*	RBAM_032870	*tagD*	GCTTAACCTTTTCAGGGCTATGTGATATAATGAGG
RBAM_023510	*rpoD*	RBAM_018580	*yofA*	GCCTGAAATCAAAGCCATTTTCGTGTATGATAAGG
RBAM_023510	*rpoD*	RBAM_026180	*citZ*	TCTACAAATTAATCAGATTATTGTTTATAATAAGA
RBAM_023510	*rpoD*	RBAM_038140	*spoIIIJ*	GGAGTAATATATTCTTTAGAAGCAGTTTTATCGAG
RBAM_023510	*rpoD*	RBAM_032820	*gtaB*	AGGTGTGAATTTTAATCAATTGTGGTAAAGTGGAA
RBAM_023510	*rpoD*	RBAM_026150	*phoP*	AATTGTCTGAACGGGCAGTATTGGATAAAATGAAA
RBAM_023510	*rpoD*	RBAM_020840	*trpE*	TAAATAGTATCTGTTTTTTATGACTGTACATTATG
RBAM_023510	*rpoD*	RBAM_032800	*lytR*	AGAATATGATTGAAAAAAAATTCCATTAAAATTCC
RBAM_023510	*rpoD*	RBAM_006550	*gutA*	GATTGATATATACGGAATCTTCCGTTATGCGAATA
RBAM_023510	*rpoD*	RBAM_014150	*sipT*	TTATGTATTTTATGATGAGTTATGATAAAGTAAAG
RBAM_023510	*rpoD*	RBAM_006540	*gutB*	TTTTATCAATTCAGTGTTTTTTGTATATGAAAAAA
RBAM_023510	*rpoD*	RBAM_037650	*sacB*	AGTTGTCAATAGAATCAGAGTCTAATAGAATGATG
RBAM_023510	*rpoD*	RBAM_021290	*resA*	AGATTAAAAGTGTATTGGAAGTTTTTAATGTTTTG
RBAM_023510	*rpoD*	RBAM_013660	*ptsG*	ACCAGAAAAGTTAATGTTATTATTGAAAAATGAAT
RBAM_023510	*rpoD*	RBAM_025620	*lcfA*	TTTCCAATTTCAATCACCCTTCTGCTATAATAATA
RBAM_023510	*rpoD*	RBAM_021240	*sigX*	AAAATAAAAACTTGCAAAAATATTTCGTCTGTTTC

RBAM_023510	*rpoD*	RBAM_013620	*ykvW*	ATTATCAAAAAGAAATTAAAACAATTATAATTGAA
RBAM_023510	*rpoD*	RBAM_018460	*fenA*	GATATAAGATTTTTTACAATCTCTTTTAAATTAAA
RBAM_023510	*rpoD*	RBAM_026060	*gapB*	TACTGGCGGATTTCTTTTAATGTGTTATACTATAT
RBAM_023510	*rpoD*	RBAM_026050	*speD*	CCTTGCAAAATAAACTTAAACACAGTATACTATCT
RBAM_023510	*rpoD*	RBAM_025570	*xsa*	GATTGACATGTACGAACCAATATAATAAGCTTTTG
RBAM_023510	*rpoD*	RBAM_029920	*yusL*	TATTGAAGTCATCCTGAAAATCTTTTAAGATGAAA
RBAM_023510	*rpoD*	RBAM_033170	*alsS*	CACTTATTATAGTGTTTAATGAATTTTAAAATATA
RBAM_023510	*rpoD*	RBAM_025560	*trxA*	ATTAGCATGAACGAATGGGAGATGCTATACTAAAA
RBAM_023510	*rpoD*	RBAM_025540	*lysC*	TATTGTGCTTTTCTGAAAAATCTGCTAGAATGTGA
RBAM_023510	*rpoD*	RBAM_025520	*sdhC*	AAATGAAATTGTCAATAAATTTTAATAAAGTGCTT
RBAM_023510	*rpoD*	RBAM_021110	*gudB*	CGTCTTTTTTTAGGCGAAATAATGGAAAAATCATT
RBAM_023510	*rpoD*	RBAM_032650	*degS*	TTTAGAATTTTTGCGCATATTTTGGTATCATGAAA
RBAM_023510	*rpoD*	RBAM_032620	*comFA*	AAAAAAATAAAAAATTAAAAGCGTTAAGCTGTTCA
RBAM_023510	*rpoD*	RBAM_037450	*purA*	GATTGACTTTCTGTTTCGGCACTGATACACTTAAT
RBAM_023510	*rpoD*	RBAM_033080	*rbsR*	TGTTGCCAACCGAAATAATTCGTGTTACTCTTTTT
RBAM_023510	*rpoD*	RBAM_005860	*ydeL*	TATTTCTAAACCCATTTTCAAAGTTTATAATGACT
RBAM_023510	*rpoD*	RBAM_037400	*yycF*	AATAAAAAATTTTCTTTTATTTTTTGTACTGTTTT
RBAM_023510	*rpoD*	RBAM_013460	*clpE*	ACTTGAAAGTCAAAGAAGGTCAGAGTATACTATTA
RBAM_023510	*rpoD*	RBAM_017800	*citB*	TATTTACTTATGTATGATTTTATACTAAGATGAAA
RBAM_023510	*rpoD*	RBAM_036990	*ydgF*	TATTGACAATGCATTCAAGTTCCATTAAAATAAAA
RBAM_023510	*rpoD*	RBAM_036960	*ahpC*	GCTTGACAAAAAATAAATGTTATTTAATAATCTAT
RBAM_023510	*rpoD*	RBAM_024980	*spo0B*	GTTTTTTTAACAAAGCCTTCTCTGTTATAATTCAT
RBAM_023510	*rpoD*	RBAM_024930	*nifS*	TGGATATAAATGTGTTGCTATTTTATTTGAGTTCG
RBAM_023510	*rpoD*	RBAM_024920	*nadB*	GCTTGAGTTTATTTTATCGTTGTGTAAATATAGGT
RBAM_023510	*rpoD*	RBAM_012950	*ohrA*	GATTGACGGTTCGTATTATATTGTGTAAAATTAAA
RBAM_023510	*rpoD*	RBAM_006270	*ydjK*	TATTGACTGAAAGCGCTTTTAAAATTATGATAATA
RBAM_023510	*rpoD*	RBAM_037350	*yyxA*	ACATTAAAAAGGTTTACTTAGTACTAATAATTAGG
RBAM_023510	*rpoD*	RBAM_025370	*ilvB*	GGTTCTTTTTTAATAAAAGAGCTGCCATAATAAAA
RBAM_023510	*rpoD*	RBAM_017750	*ccdC*	ATTGGTTTTTTTTATGTTTTTGGAGTATAGTATAT
RBAM_023510	*rpoD*	RBAM_005770		CATGGATTGATAAAGCGCTTTCCGCTATAATAAGA
RBAM_023510	*rpoD*	RBAM_017730	*ccdA*	ATAATAATATAGGTGTAAATATGACAAAAAGTTTA
RBAM_023510	*rpoD*	RBAM_013370	*mtnW*	CAATGCGATAATTCGAAATACTTTATAAAATCATT
RBAM_023510	*rpoD*	RBAM_013360	*mtnE*	TCTTTACTTTTGTTATTTAACCATATATGATGATG
RBAM_023510	*rpoD*	RBAM_013350	*ykrU*	GTAGTAGTATATACCAATTTATTGTTTTCATTTCT
RBAM_023510	*rpoD*	RBAM_013340	*mtnK*	GTTTTCCAAGACTTGCGGCGTGATTTATATTAATT
RBAM_023510	*rpoD*	RBAM_036870	*des*	ACCTTAGTATTATTTTTTTTCCCCTATACTGTAAG
RBAM_023510	*rpoD*	RBAM_001320	*rpoB*	GTTTGACTTGAATTTTCAGCTATGTTAATATTGTA
RBAM_023510	*rpoD*	RBAM_036820	*htpG*	GTTTCTAAATACCTATTAAATATGTTAATATTATG
RBAM_023510	*rpoD*	RBAM_036810	*csbC*	GAATGAAATAGTTATCTAAATAAACTATTATGCGT
RBAM_023510	*rpoD*	RBAM_032450	*secA*	TCTTTGGAAATAGCAAAAGGTATGTTATGATAATG
RBAM_023510	*rpoD*	RBAM_020470	*ponA*	CCTTGGATACTCACCGAGAATTGATTATATTAGTA
RBAM_023510	*rpoD*	RBAM_000850	*pabB*	ACTTTTTTCACCAGTGTGATTGCTTTAAAATAAAA
RBAM_023510	*rpoD*	RBAM_000840	*cysK*	TTAGTAAAATATTATTTTTATTTCTTAACATGGTA
RBAM_023510	*rpoD*	RBAM_037260	*rocR*	TAAATAAAATTTTATTACTTACAATTCTGCTAACT
RBAM_023510	*rpoD*	RBAM_037250	*rocD*	TCAATCGTCTTAACATTCATTATTTTAAAATAAAT
RBAM_023510	*rpoD*	RBAM_037240	*rocE*	TATTTGCGCGGCGACAGTATTTTAGTATAATGTAT
RBAM_023510	*rpoD*	RBAM_025230	*hemA*	TCATTATAAATAATAAATTCTATGTTAGAATGATT
RBAM_023510	*rpoD*	RBAM_036790	*iolR*	ACATGAATTAAACGCTTACAAATGCTAAAATGATG
RBAM_023510	*rpoD*	RBAM_036780	*iolA*	GTAGTAAAATCGTAAACATTCGCAAATTAAGTACA
RBAM_023510	*rpoD*	RBAM_001230	*sigH*	GTTGACGCTTTTTCGCCCATTAATGTATAATATTT
RBAM_023510	*rpoD*	RBAM_012770	*dppA*	TATTGCATGTTTCATCCTTTTTTAATATAATTTGT
RBAM_023510	*rpoD*	RBAM_012750	*htrA*	TAAATAATAAAGTTTAGGCTACCTTTAACAGTTAT
RBAM_023510	*rpoD*	RBAM_000750	*spoIIE*	TGTTGACGGAATGAAGAAACCTTTGTATTATATAA
RBAM_023510	*rpoD*	RBAM_029590	*pucF*	GAGATAATAGGTTTTTTATTTAGATTATAAGATAG
RBAM_023510	*rpoD*	RBAM_029570		CGGGCAATATGACTTACCATATTTTTTACTGTTTA
RBAM_023510	*rpoD*	RBAM_031850	*clpP*	GTTTGACCTTTATTGACCAAAAATGTATCATGTAA
RBAM_023510	*rpoD*	RBAM_031840		GGTTGACATACAATTTTCTGATTATTTTAATCAAA
RBAM_023510	*rpoD*	RBAM_029500	*pucH*	TTTTTAATATCGCCAAAACGACTACATAATGCTAG

RBAM_023510	*rpoD*	RBAM_025140	*valS*	ATTGACGAATCAAAAACACTTTTACTATAATAAAG
RBAM_023510	*rpoD*	RBAM_025120	*comC*	TTTTAGTCATGGTGTTTTTCTTTGCTATAATCTGA
RBAM_023510	*rpoD*	RBAM_001170	*gltX*	TTTTGATGCCCGGTCTATTTGGTGGTAGAATAGAT
RBAM_023510	*rpoD*	RBAM_024630	*yrzC*	CAATGAAAATTAGGATAAGTTATGTTATAATAACA
RBAM_023510	*rpoD*	RBAM_032230	*yxaF*	TTTTGACTTTTTTTTGTCCCTTTTCTATACTAAAA
RBAM_023510	*rpoD*	RBAM_020260	*xpt*	ACTATGCAAAAGAAGTGTATCTTGTTATAATTTGA
RBAM_023510	*rpoD*	RBAM_037050		TTTTTTGTATTTTAGTAGTAAAGTTCGATAGTTTA
RBAM_023510	*rpoD*	RBAM_037000	*fbp*	AAAATAAAATTACCTTGAACTTACGTAACAGTTAT
RBAM_023510	*rpoD*	RBAM_001080	*ctsR*	TTTTTGTGTTTTTCTGCATACGTTATATAATATGA
RBAM_023510	*rpoD*	RBAM_000590	*glmU*	TCTTGAAATCAACAGCTATTTAGGATATATTTTTC
RBAM_023510	*rpoD*	RBAM_031670	*epsA*	CAAGTAATATTCTTTAAAAAGCAAGAAATATTTTA
RBAM_023510	*rpoD*	RBAM_009760	*sigM*	GAGATAATATTGTTATAAAAAATAATGGACCATGA
RBAM_023510	*rpoD*	RBAM_017350	*xylA*	GAAATAATATAGATTACACAAGCATTTTTTGATTT
RBAM_023510	*rpoD*	RBAM_005380		AACATCAGGGAAGGGGAAATTTTTATAAAATGAAT
RBAM_023510	*rpoD*	RBAM_017320	*xynP*	AGTTGAAAAAGCAGAAAAGATTTTATAATATGAAA
RBAM_023510	*rpoD*	RBAM_009700	*citA*	ATTATTTTTTAAATATTATATTTACTATAATAACA
RBAM_023510	*rpoD*	RBAM_036480	*deoC*	AATTTACATATGTTCAAAATTCGGTTATTCTAAAC
RBAM_023510	*rpoD*	RBAM_028820	*yuxH*	ATTATAAAAAGTGTAAATCTGTTTAAAGCAGTTCA
RBAM_023510	*rpoD*	RBAM_036400	*hutP*	TTTTGACTTCTGATTCGCAAACACTTAATATAAAA
RBAM_023510	*rpoD*	RBAM_012450	*rapA*	GTTCTTCTTTTTCTAGCAAATATGATAAAATATGA
RBAM_023510	*rpoD*	RBAM_004800	*dctP*	CTTTTGATATTTGTTATATTTGCCTTTTAACTTCT
RBAM_023510	*rpoD*	RBAM_009690	*citR*	ACAATAATATCATTTATATTATAAATTTTTTATTA
RBAM_023510	*rpoD*	RBAM_009680	*lytE*	TTTGAAATCAAATTGAAATATTTAATACCCTTAAA
RBAM_023510	*rpoD*	RBAM_009670	*phoA*	AAATTCCCATAATTTATAAAGTTAAACTAAAGTTT
RBAM_023510	*rpoD*	RBAM_017250	*glnR*	AATATAACATCACCTATAATGAGACTAAGATAAGA
RBAM_023510	*rpoD*	RBAM_009630	*lytF*	CATAAGAAAAAAGGACTATTTCTATTAATATGATA
RBAM_023510	*rpoD*	RBAM_011930	*yjcI*	TATTGAAATTCATTTTATAGTGCGTTAATATAGAA
RBAM_023510	*rpoD*	RBAM_036360	*bglP*	ATGATAAAATTGGCAACACCTGTTTGATCTCTTCA
RBAM_023510	*rpoD*	RBAM_016780	*recA*	TCTTGGCAAATGCCAGTAAACAAGGTATAGTATAT
RBAM_023510	*rpoD*	RBAM_024380	*yrrT*	AAGATGAAATAGAAACTAATTTAGGTTAACTTTTA
RBAM_023510	*rpoD*	RBAM_012380	*uxaC*	GATTGATATGTGAGAAAGAGGAAGCTAAAATAAAA
RBAM_023510	*rpoD*	RBAM_023890	*comEA*	TCTGAACGGGACTGGTAATGGAGATTAAAATGTGA
RBAM_023510	*rpoD*	RBAM_009580	*yhcY*	AAAGGAATATACAGAAAAGAATTGATATAATATGG
RBAM_023510	*rpoD*	RBAM_017170	*nrdI*	ACTACTAAGCAGTTTTTTTTCATGTTATGATAGAC
RBAM_023510	*rpoD*	RBAM_035800	*licR*	CATTGACTTGAAATCCTCCCCATTATAGAATTTAC
RBAM_023510	*rpoD*	RBAM_009540	*glpF*	GATTGACACCGCTTACACCCATTGTTACAATAGAC
RBAM_023510	*rpoD*	RBAM_023810	*lepA*	AATTGAATGTTTACAATCCTATTGATATAATCTAA
RBAM_023510	*rpoD*	RBAM_016650	*ftsK*	ATACTAAAATGTAAAAAATTGCGTCATTCCACTAC
RBAM_023510	*rpoD*	RBAM_024210	*manR*	AACTTTTTACTCTTTTGTTTCACTATAAAATGAAA
RBAM_023510	*rpoD*	RBAM_016600	*dapG*	GTAATAGTTTAGCGTAAATTGCGGTAAGGCGTTTA
RBAM_023510	*rpoD*	RBAM_004600	*rapH*	CAAATAACATCCTTTTCAATATTATAATGTTTTTA
RBAM_023510	*rpoD*	RBAM_000230	*yaaJ*	AGGATAAAATTGGAAATACTCTGTACTTTTTTGAA
RBAM_023510	*rpoD*	RBAM_035760	*licH*	GTAGTAATAGTCTGTCTTGACTAGGTTTTACTTCG
RBAM_023510	*rpoD*	RBAM_023790	*hrcA*	AATTGACATTTTTCTTTTGGTTTGTTACTTTTGAT
RBAM_023510	*rpoD*	RBAM_029050	*dhbA*	AATTGACTGCGTGTTTTGATTTGGATATGATTGTC
RBAM_023510	*rpoD*	RBAM_035710	*dltA*	TAGATAGTATTATTAAAGAAGGAGTAAAGTGTTCT
RBAM_023510	*rpoD*	RBAM_017070	*aprX*	GAAGTAATATTTGCTTGTTTTCAAGGTAGGGTTGA
RBAM_023510	*rpoD*	RBAM_031330	*araR*	TATTTCTTATTCGTCAAAAAGTATTAGACAGTTAA
RBAM_023510	*rpoD*	RBAM_031320	*araE*	AATTGACAGATTATGAAAAACTGCTTATTCTTTAT
RBAM_023510	*rpoD*	RBAM_031310	*cggR*	AGTTGAATTAACAATGTCATCCTGTTAAAATAATT
RBAM_023510	*rpoD*	RBAM_009400	*yhcL*	ACTTGACCGATCGGAATTGATGTAATATAGTGTTT
RBAM_023510	*rpoD*	RBAM_036190	*bglS*	TTTTGAGTATCGTAAAATACCATTTTAAGATTGAC
RBAM_023510	*rpoD*	RBAM_036180	*citH*	TGTGTAAAACTTTATTCGGTAAAAGTGTCTTTATT
RBAM_023510	*rpoD*	RBAM_030860	*ytlI*	ATTTGTTTACTAAGCTTTGATTGTTTATACTATAA
RBAM_023510	*rpoD*	RBAM_030850	*ytmI*	AATATCATATTTGTTAGTTTCGAATCATTTGTTTA
RBAM_023510	*rpoD*	RBAM_012180	*lacG*	TATTGACAACAAAAATGTATGCGTTTACTATTAAA
RBAM_023510	*rpoD*	RBAM_035660	*epr*	GAGGTAGTATAAAACCTTTGCGAATGTAAACGTAT
RBAM_023510	*rpoD*	RBAM_031290	*pgk*	ACTTGGTTCTTTCGGACAGAAACGCTATAATGAAA

RBAM_023510	*rpoD*	RBAM_035610		CCTTCATAAAAAGACGAAAACACGATATGATAAGA
RBAM_023510	*rpoD*	RBAM_023600	*cdd*	GCGTGAAAAACCAATCATAATTATGTAAAATAAAA
RBAM_023510	*rpoD*	RBAM_036080	*galE*	GGATGATTGAGGAAGAAAAAACCGATATAATATGG
RBAM_023510	*rpoD*	RBAM_036020	*cimH*	TATATGAACAATTTAAATTTTTTAGTATACTTTAA
RBAM_023510	*rpoD*	RBAM_004470	*lrpC*	TCTTGTCATTTTTCGGATTTCTCTTGATAATATAG
RBAM_023510	*rpoD*	RBAM_036010	*cydA*	AATTTCATATGATTTTTTAAATTTAACAAGTATAT
RBAM_023510	*rpoD*	RBAM_008820	*perR*	TTTGTAATATTTCATTATTAGAAAAAATCTTCTCT
RBAM_023510	*rpoD*	RBAM_004400	*ydaB*	GTTTTACAAGTTATATTTTTTGGTTTAAAATGGGT
RBAM_023510	*rpoD*	RBAM_035570	*ywbI*	AATATAACATGTCTTCCAAAACGATATCCATTTTA
RBAM_023510	*rpoD*	RBAM_015970	*xerC*	ATTGCAACTTCGATGCATTATATGATACCATTTAA
RBAM_023510	*rpoD*	RBAM_000010	*dnaA*	CATTGCAAGTCCTCGCTTAATTTGATATTATATTT
RBAM_023510	*rpoD*	RBAM_023530	*yqxD*	TTTGTATTATAATAAAAAATTGTGATAAAATGTTT
RBAM_023510	*rpoD*	RBAM_031100	*opuCA*	ACTTTTTATTTTACAAATTTCATCTTATAATAAAG
RBAM_023510	*rpoD*	RBAM_011520	*mecA*	TTTTTTATTTCTTTTATAACTGTTTTATCATAAAA
RBAM_023510	*rpoD*	RBAM_003900	*yxeK*	AATTGTACATTTTGATTTATTCTTATAAAATTACT
RBAM_023510	*rpoD*	RBAM_011500	*spxA*	ACTGTTCACACTTACTTTTTTATAGTATAATACCT
RBAM_023510	*rpoD*	RBAM_008740	*fabL*	TCTGGAATATTTCGGTCTTTTTCGGTAAAGTAAAA
RBAM_023510	*rpoD*	RBAM_030610	*yvgR*	ATTTTACTTACATACAAACTTTCGGTAAAGTTTCA
RBAM_023510	*rpoD*	RBAM_027810	*menF*	CTTTATTCATTTGCGCAAAAAGGTTTATAATAAGA
RBAM_023510	*rpoD*	RBAM_031050	*opuBA*	AAAATTAAATGTTTCAAATTAAATATTACCTTTTG
RBAM_023510	*rpoD*	RBAM_034990	*rocG*	CTAGTCTTATTGACTATTACTACAAAAACATCTTT
RBAM_023510	*rpoD*	RBAM_009110	*ssuB*	CGCATAAAATTAGGAATATTTTTTGTCAACTGTTC
RBAM_023510	*rpoD*	RBAM_034980	*rocA*	TTTTCTTATTTTTATCACAAATTGCTATGCTGGTA
RBAM_023510	*rpoD*	RBAM_022960	*yqxM*	GAACTAATATGAAACAAGTATGTCAAGAAATTTCT
RBAM_023510	*rpoD*	RBAM_028210	*hmp1*	ATTTGTTTTGCTCAGAAAGATATCTAATAATGAAA
RBAM_023510	*rpoD*	RBAM_015710	*fapR*	CATTGCATTCCAAACTATGAGATTATATACTACTA
RBAM_023510	*rpoD*	RBAM_011330	*fabHA*	TATTGCCAACAGAAAAAAAATAGGGTAGAATTAGT
RBAM_023510	*rpoD*	RBAM_011300	*med*	CCTTCCATGTTATGTGCATTACCGCTACAATAAAC
RBAM_023510	*rpoD*	RBAM_034850	*eutD*	AAATAAGATTTCTTTTTGAAAGCGCTATAATAGAA
RBAM_023510	*rpoD*	RBAM_034830	*ywfK*	ACTTTACTGCGGTCTTAATCATTAGTAAAATATAT
RBAM_023510	*rpoD*	RBAM_028160	*gbsA*	TTTTGACAGGCAAAAAAACATGTGTTAAATTAATA
RBAM_023510	*rpoD*	RBAM_004160	*gabD*	GAAATGATATCGTCTCTTTCCTTACAAGTTATACG
RBAM_023510	*rpoD*	RBAM_004150	*gabT*	AAAGTAGTATGGTTTTTCTTCAGTCTTACTATTCT
RBAM_023510	*rpoD*	RBAM_004140	*gabR*	TCTTATCATTCTGACTTCTTTTTGGTATGATGAAA
RBAM_023510	*rpoD*	RBAM_035290	*sacP*	TATTGACGAAAGCGCTATCATGAAATAGAATGAAA
RBAM_023510	*rpoD*	RBAM_023290	*pstS*	AAAAAAATATGTTTTTTTGTAGGCCTAAATGTTTT
RBAM_023510	*rpoD*	RBAM_027640	*ytkD*	GGCTTCCGAACAAACGCAACTGTGGTATGATGTAT
RBAM_023510	*rpoD*	RBAM_003680	*srfAC*	TTAGTCAAAATTGATGAACCCCGTCAAACTGTTAC
RBAM_023510	*rpoD*	RBAM_003660	*srfAB*	ACTTGATAAATTAATGGAAATTCTCTAACAGAAAA
RBAM_023510	*rpoD*	RBAM_003650	*srfAA*	AAATTAAAATGCATTTATTATAAAAATTTTTATTT
RBAM_023510	*rpoD*	RBAM_010790	*addB*	ATTGGTCATTTTCGTCAACTTTCGATAAAATGTAG
RBAM_023510	*rpoD*	RBAM_016010	*flgB*	CATAGACTTTAAGCCTGTTATTTCTTACAATAAGC
RBAM_023510	*rpoD*	RBAM_027580	*pckA*	TTTATCATATCTGATAAAAGTCTATATACAATATG
RBAM_023510	*rpoD*	RBAM_027550	*ytnA*	TATTACGCAATATTCAGAAAAAGGTTATAATAGGA
RBAM_023510	*rpoD*	RBAM_007970	*treP*	GATTGACGACATGTATATACATCATTACAATAGAG
RBAM_023510	*rpoD*	RBAM_003590	*yckE*	TTTTTCGTCTTGTATAATTTGTCGTATACCTGTCC
RBAM_023510	*rpoD*	RBAM_011190	*argC*	CAATTGAATTAATTTTTATTCATGTTATAATGTTA
RBAM_023510	*rpoD*	RBAM_003550	*yciC*	ACTTGATTTTAAACGGAGAATACATCAGAATAAAT
RBAM_023510	*rpoD*	RBAM_007900	*nagP*	AAGTTCGAATGGTGTTGACCAAATATTTATCTTTT
RBAM_023510	*rpoD*	RBAM_003540	*nasA*	ATATATTCTTGATTCAGCAATTTGAAATGATAGAA
RBAM_023510	*rpoD*	RBAM_003530	*nasB*	GGTTTAGTATTTTGAGTGTAGTTTTATATTCTTCC
RBAM_023510	*rpoD*	RBAM_034630	*ywhE*	TTTTTCATATCTCCTACCTCCCCTTTTTCATTTCT
RBAM_023510	*rpoD*	RBAM_034620	*speE*	TCTTTACTTTTTCCCCTCCATCCTCTATACTTTTT
RBAM_023510	*rpoD*	RBAM_008360	*glvA*	TTTTGAGAAAAATAACTAAAAATGATTAAATGATC
RBAM_023510	*rpoD*	RBAM_030240	*yvqG*	GAAATATTATTTTACTTACTTTTTCCTACCTTGCA
RBAM_023510	*rpoD*	RBAM_010600	*comK*	TTTTTCTTTTTGTTATAAATTTATCATTAAATTCT
RBAM_023510	*rpoD*	RBAM_007890	*yflG*	TTTTCTATTTATAAACCAGTTGTGGTAAGCTTGAA
RBAM_023510	*rpoD*	RBAM_023050	*comGA*	TTATTTTTTATTTTTTTACGTTTTACACACTGTAA

RBAM_023510	*rpoD*	RBAM_027400	*ytrA*	CAAATAACATAATGCACATGATGAACTACATTATG
RBAM_023510	*rpoD*	RBAM_003460	*ycgO*	AAAGTGGTATTTGTATATTTCGCCTCCTCCTTTAC
RBAM_023510	*rpoD*	RBAM_007810	*citM*	TATTTACTTCAATATCCAATTCTCATAAGATAGCA
RBAM_023510	*rpoD*	RBAM_015400	*cysH*	ATCTTAACACGATTTTAAATCATGTTTAGATTTTT
RBAM_023510	*rpoD*	RBAM_002990	*lmrA*	TTAGTTCTATTATCTGGTCAGTGATATAAAATTTA
RBAM_023510	*rpoD*	RBAM_034530	*rapF*	TGTGGGGATAATAGTGGATTTTTGAAATGATATCC
RBAM_023510	*rpoD*	RBAM_026920	*ytpT*	TCTTTTCTGAGCCGCCCGGCAATGGTATACTAAAT
RBAM_023510	*rpoD*	RBAM_030160	*fumC*	TTATCTTTTCCTGTTAAAAAAATGATATACTATGA
RBAM_023510	*rpoD*	RBAM_022550	*spo0A*	TCTTCACTTCTTTCCTGGATTCATGTAAAATAAGA
RBAM_023510	*rpoD*	RBAM_030120	*cssR*	AAGATAACAGGGTTATTATTCCTCTTTTCGTTTCG
RBAM_023510	*rpoD*	RBAM_002930	*phoD*	TCATAGCGGTTTAACATTTCTTTTTTATAATAAGG
RBAM_023510	*rpoD*	RBAM_030110	*yvtA*	GCTTTGCTTTTCTCCTTATTATTGGGACAATAGAA
RBAM_023510	*rpoD*	RBAM_010500	*aprE*	TAGTTACTATAAGTAATAGTAATAAAAATATTTTA
RBAM_023510	*rpoD*	RBAM_019730	*rapA1*	CCCTTCCCAACATATTACATTATGATAAAATATTA
RBAM_023510	*rpoD*	RBAM_003360	*nadE*	AAAGTAGGACAGGCTAACTTTTTCCTTTTACGTTG
RBAM_023510	*rpoD*	RBAM_015300	*pyrR*	CATTGACAGCGACTTTCTTTTCTGAAATAATAAAA
RBAM_023510	*rpoD*	RBAM_034470	*narK*	AACATACTATTCTGACTATAACTTAGTCACTTTCC
RBAM_023510	*rpoD*	RBAM_034440	*arfM*	AAAGTAATATTCATGTATGTGAGAGGACTATTCAC
RBAM_023510	*rpoD*	RBAM_034430	*narG*	AATATAACATAATTTTCCGTTATAGGCAAAATGCC
RBAM_023510	*rpoD*	RBAM_026810	*acuA*	TGATTAATATCATCTACAAGTGTCAAAAGAGTTAA
RBAM_023510	*rpoD*	RBAM_026800	*acsA*	AATTGAGAAAACTGTGAACATCTACTATAATTAGT
RBAM_023510	*rpoD*	RBAM_030010	*yusT*	ATTGTAATGGAGATGTGCATTTTCGTATAATGATT
RBAM_023510	*rpoD*	RBAM_010420	*gltT*	CGTTGATTTTCCAAAACTGTTATAATACAATGTAG
RBAM_023510	*rpoD*	RBAM_008110	*yfkJ*	GAAATAAGATTGTCTTAAGTAAATCTTATTTTCGA
RBAM_023510	*rpoD*	RBAM_002800	*ybgJ*	AATATGTAATTGTACCTTGTGCGATAAAAACTTTT
RBAM_023510	*rpoD*	RBAM_010400	*fabHB*	TTTTTTATTAAAATTAGTACCAAGTAATAATATCA
RBAM_023510	*rpoD*	RBAM_015260	*ileS*	AAGCTTATATTATGCAAGTATTTAAGTCTACTTTT
RBAM_023510	*rpoD*	RBAM_003290	*ldh*	TCTTGCAAAAGTTTGTGAAGTATTGCACAATATAA
RBAM_023510	*rpoD*	RBAM_007640	*yfmP*	CGTTTACGTTAAGGTTCAAATGGTGTATAATAGAA
RBAM_023510	*rpoD*	RBAM_003280	*amyE*	CGTTGCTTTGATAGAGAGTGTGTGATAAGTTGAAA
RBAM_023510	*rpoD*	RBAM_003250	*amhX*	CTATTCAGAAAATACAGATATGTGCTATATTATAT
RBAM_023510	*rpoD*	RBAM_026790	*tyrS*	CGTTGACATCGTATCTTATTTATGTTAAAAATGAG
RBAM_023510	*rpoD*	RBAM_003220	*opuAA*	GATATAAAAAGGAATTATAAGGAATAAAAATCCTC
RBAM_023510	*rpoD*	RBAM_022380	*bkdR*	GAAATAACAGGTTAGCTTTTTTATAAAACAGTTCT
RBAM_023510	*rpoD*	RBAM_034330	*ywjF*	GAAGTATTATTTTTTGTAGTATACTTATCTTCTCT
RBAM_023510	*rpoD*	RBAM_026710	*hmp*	TTCTTAAGTTTTTGCTAAGAAAAGAGTCCAGTTAG
RBAM_023510	*rpoD*	RBAM_034310	*pyrG*	TCTTGACTTTCATTGTCGAACTATGTAGTATGTAT
RBAM_023510	*rpoD*	RBAM_014730	*ctaA*	GAGATAGTATGTTCTATTACTGGACATTTTTTTGT
RBAM_023510	*rpoD*	RBAM_002750	*gltP*	AGTATAATTTACTAATGCGAATGTATTACATTTCG
RBAM_023510	*rpoD*	RBAM_010340	*pbpF*	GCTTGCTAGTATATCAAAAGAGTGGTATAAGTTTC
RBAM_023510	*rpoD*	RBAM_002700	*pssA*	ATTGTTCAAATAACACCAAATGCTGTATAATAAAA
RBAM_023510	*rpoD*	RBAM_033860	*rapB*	GTTATACAATTTTATATTGGTTATCTTTAATCTTT
RBAM_023510	*rpoD*	RBAM_007590	*yfmT*	AACATAATATAACAGCTTTAAAGAGCGTCAGTTAA
RBAM_023510	*rpoD*	RBAM_033830	*ureA*	GGTTAATATTCACTAAATTTTTAAATATTCTATAA
RBAM_023510	*rpoD*	RBAM_015140	*spoIIGA*	TTTTTAACATGAAATGGGAGTTGTAATTAACTGTC
RBAM_023510	*rpoD*	RBAM_003150	*yceC*	ATTCTTATGTTTTATGAAACTTTGATATAATAAAA
RBAM_023510	*rpoD*	RBAM_015110	*ftsA*	TCTAAAAAAAATAAAAAAAATACGATATAAAAGAG
RBAM_023510	*rpoD*	RBAM_034290	*spo0F*	CTGATAATATTAATTTCCTTTACACTTTTAGTTTG
RBAM_023510	*rpoD*	RBAM_003110	*ycdH*	TTTTTAGGTAAATTGGATATCAATTTAGAAAAAAA
RBAM_023510	*rpoD*	RBAM_014690	*ylaM*	AATTTCATATTTTCAAAAGCGTACTTAACAGTGAA
RBAM_023510	*rpoD*	RBAM_002690	*purT*	GTCTTTATTTTTTGAATATTTCCGTTATAATGAGT
RBAM_023510	*rpoD*	RBAM_034210	*ywkA*	TCTTGAATTTTCATTTTTACAGGCATAAAATGACT
RBAM_023510	*rpoD*	RBAM_033790	*xynA*	AAATTATTATAACTACGGTTATGTATAATTTTTCC
RBAM_023510	*rpoD*	RBAM_002610	*glpT*	CGTTTACATTCACGTCAGACTGTGATAATTTAAGA
RBAM_023510	*rpoD*	RBAM_002600	*glpQ*	TGTGTAAGAAAACAACATAAAAGATTAATATTTTT
RBAM_023510	*rpoD*	RBAM_027070	*ytkP*	ACAATAATATTTTGCCTATTCGCCTCTCCTCTTGG
RBAM_023510	*rpoD*	RBAM_003080	*rapJ*	TGTTTGCTCACTGCCATAAATTTGCTATGATAAAA

RBAM_023510	*rpoD*	RBAM_021710	*ansA*	ACTTGATTAATTTCATTCGTCTGTCTATAATTTGA
RBAM_023510	*rpoD*	RBAM_026590	*rpsD*	CCTAAACATGTACTACTCATTTTTGTACATTATAT
RBAM_023510	*rpoD*	RBAM_018960	*yocH*	TTTTGTTATTGAACTGACATTTTCATATGTTACGA
RBAM_023510	*rpoD*	RBAM_003000	*yccC*	ATTTAAAATATAGTGACTGGTCTATTATCTTGATT
RBAM_023510	*rpoD*	RBAM_014590	*ylaC*	TCTGGATGACAGTGTGGTTTTATTTTTAAATGAAA
RBAM_023510	*rpoD*	RBAM_026540	*ezrA*	AATGTACCATAGTATTATTGTTACTTTTGCTGTCT
RBAM_023510	*rpoD*	RBAM_022170	*yqjI*	AAAATAAAATTAAATTTAAAATTTTACCAATTTGC
RBAM_023510	*rpoD*	RBAM_026520	*nifZ*	TTTTGTTATGGTAGCATAAATATCATACGATGTAC
RBAM_023510	*rpoD*	RBAM_033680	*nrgA*	AAATGTTTTATCGTTCTTTTTTCTCTATAATAATG
RBAM_023510	*rpoD*	RBAM_033650	*ywoE*	CAATGTTATTTACGGAATTTTCAGTTATATTTTTT
RBAM_023510	*rpoD*	RBAM_019360	*bglA*	TATTGATAAATTATACCGGTACAATTATAATGTAA
RBAM_023510	*rpoD*	RBAM_007390	*yetO*	CTTACTGTATTTTGCTATTTTTGAACTAAAGTTTT
RBAM_023510	*rpoD*	RBAM_021640	*fur*	AGTTGGAACTCTGCGCGTATTTTGTTATAATGAGT
RBAM_023510	*rpoD*	RBAM_021610	*drm*	ATTGTCAACAAGAATGTAAACGGTTTATACTTGAG
RBAM_023510	*rpoD*	RBAM_026490	*ytcI*	ATTTTGACATATGTTAATGGTTTCAAAAAAACTTA
RBAM_023510	*rpoD*	RBAM_034060	*glyA*	TGTTACATTCACCTGAAAAACCATGTACAATAAGG
RBAM_023510	*rpoD*	RBAM_006840	*purE*	CGTTGACATTATCCAAGTCCGTTGTTAAGATAAAC
RBAM_023510	*rpoD*	RBAM_014440	*pdhC*	CTTTTCTCTTATATGAAGGAATCTGTTTACTAGGT
RBAM_023510	*rpoD*	RBAM_026400	*ackA*	ACTTGAAATGCTGAGTTGACAATGTTTAAATGGAA
RBAM_023510	*rpoD*	RBAM_010070	*hemZ*	GATTGATTTTAGCCTGTTCTTTCTGTACACTAAAG
RBAM_023510	*rpoD*	RBAM_002460	*ybxG*	AATTGTATTTTCCACGGAAATTTATTATGATGGAA
RBAM_023510	*rpoD*	RBAM_014420	*pdhA*	TCTTTGTAATTTTTTTTAATTATAGTACAGTAGAG
RBAM_023510	*rpoD*	RBAM_037970	*adaA*	CGCGTACTTATGCAATGATTTTGTTTATACTGAAG
RBAM_023510	*rpoD*	RBAM_025990	*thrS*	AGTTGCATTTTTCTCACAGACATTTTATAATACAT
RBAM_023510	*rpoD*	RBAM_033580	*rapD*	ACCTTCATATAATACTTAGTTAAAGAAAAAGTTGG
RBAM_023510	*rpoD*	RBAM_025930	*infC*	TCTTGACTAATGATCCGGTATTGTGTAGAATAGTT
RBAM_023510	*rpoD*	RBAM_013920	*ykuN*	TTAGTAATAGTAAATTTCACAATATATACAATTAC
RBAM_023510	*rpoD*	RBAM_013910	*ccpC*	GAAATAAAATAATCTTACCATCTATTCTTTATGCT
RBAM_023510	*rpoD*	RBAM_006760	*guaA*	TCTTGACCGCATATCGTCCTGTTGTTAGAATAAAT
RBAM_023550	*yqzB*	RBAM_027580	*pckA*	AAACCGCAAATAGTATAGACTATTTTCAGATATATGTTATACT AATTCCACATTCAG
RBAM_023550	*yqzB*	RBAM_026060	*gapB*	ACATGACCGCCTAAAGAAAATTACACAATATGATATATATTTA TTTCCATTTAATCT
RBAM_023790	*hrcA*	RBAM_023790	*hrcA*	TTAGCACTCGCTGACCTTGAGTGCTAA
RBAM_024630	*yrzC*	RBAM_004850	*ydbM*	TCTCTATAATACTTATTAACTTTATCGGAATTATAATG
RBAM_024630	*yrzC*	RBAM_024380	*yrrT*	CACGATTAATCCATACTATACTCATAGGAATTATAAAA
RBAM_024630	*yrzC*	RBAM_009400	*yhcL*	TATTGTCAATCCTTACTTGACCGATCGGAATTGATGTA
RBAM_024630	*yrzC*	RBAM_030860	*ytlI*	TATATATTAAGGTTAAACAAATGATTCGAAACTAACAA
RBAM_024630	*yrzC*	RBAM_030850	*ytmI*	AACAATCAAAGCTTAGTAAACAAATTGGAATTATATAT
RBAM_024630	*yrzC*	RBAM_003900	*yxeK*	AACTAAATAAGAATATTTTAATGATCCTTTATATCCAA
RBAM_024630	*yrzC*	RBAM_009110	*ssuB*	TCCTTATAAAAAACAGTTGACAAGTCGGAATAATTTGA
RBAM_024630	*yrzC*	RBAM_027070	*ytkP*	TTAGCCTTACGGCTATTTTATTACGCCGTTACTTCGAA
RBAM_024630	*yrzC*	RBAM_000840	*cysK*	TAAATATTATGGTTATTTAAATCAGCCTTTAACTCCAC
RBAM_024940	*yrxA*	RBAM_026520	*nifZ*	AAAAATCCTCCTAAATACTATGTTGAAATATTCTGTCAATTAA TTGTCAGATAATATAACAAAACGAACATTACTTATTATATGGC AATGTACAGAGAAGTCAATACT
RBAM_024940	*yrxA*	RBAM_024930	*nifS*	TACATCACCCTCCTGTTATATTTACACCTGTCTTGACACCTATA TTTACACAACGATAAAATAAACTCAAGCTTTTTTTAACATAAA CGGATAGGAGACGGGCTGTAT
RBAM_024940	*yrxA*	RBAM_024920	*nadB*	TATGTCGGGCAGAGGATAGGCAAATACAATTTTTTTCGAACTC AAATAAAATAGCAACACATTTATATCCACAGTTCTGTCCACAT TTATATTGTCCTCCCACTACAT
RBAM_025610	*ysiA*	RBAM_013850	*ykuF*	ACATCTGTGATAAAATTATGAATGATTATTCATTCAATA
RBAM_025610	*ysiA*	RBAM_025620	*lcfA*	GTCCTACACGTATAAGTAGGGCTTCACGGCACTGTCGTC
RBAM_025610	*ysiA*	RBAM_029920	*yusL*	TTTTTTTTGCCGAAAAAATGAATGACTATTCATTCAAAG
RBAM_025610	*ysiA*	RBAM_004400	*ydaB*	TTTACTTCTTTAACAGTGCGGAAAAAAGTTTTAAATACT
RBAM_025610	*ysiA*	RBAM_034330	*ywjF*	ATAACTGACTTGTGAGTAAGTAATAAATACTATTTCATT
RBAM_026150	*phoP*	RBAM_008030	*yfkN*	AACTTTCACAAAATATTCTTACA
RBAM_026150	*phoP*	RBAM_002600	*glpQ*	ATTCTTTTGTTGTATTTTCTAAT
RBAM_026150	*phoP*	RBAM_026150	*phoP*	CTACTTTCATAACAGTATTCTTT
RBAM_026150	*phoP*	RBAM_021290	*resA*	CTAATTTTCACATAACCTTCAAA

RBAM_026150	*phoP*	RBAM_037400	*yycF*	AAAATTTTCTTTTATTTTTTGTA
RBAM_026150	*phoP*	RBAM_009670	*phoA*	TTTATTGGGGAACAATTTTTTTA
RBAM_026150	*phoP*	RBAM_023290	*pstS*	AAAACCTTAACACTTGATTTATA
RBAM_026150	*phoP*	RBAM_002930	*phoD*	AAAATTAAAGAACACTTCTTTTT
RBAM_026150	*phoP*	RBAM_032880	*tagA*	ACATTTTGATAACAATTCAAAGA
RBAM_026150	*phoP*	RBAM_032870	*tagD*	AGAAACTTAACAATAGTTTTACA
RBAM_026860	*ccpA*	RBAM_026400	*ackA*	ACATTCGCCATGGT
RBAM_026860	*ccpA*	RBAM_033170	*alsS*	ACTTTCGAGATGTT
RBAM_026860	*ccpA*	RBAM_025370	*ilvB*	ACTTTCGCATATGT
RBAM_026860	*ccpA*	RBAM_007590	*yfmT*	ACTTATGCATATAT
RBAM_026860	*ccpA*	RBAM_026490	*ytcI*	ACTTAAGCCACATT
RBAM_026860	*ccpA*	RBAM_025860	*araA*	ACGTTCGCGAAATA
RBAM_026860	*ccpA*	RBAM_019060	*dhaS*	TGTATGCGCTTACT
RBAM_026860	*ccpA*	RBAM_026180	*citZ*	ACATTCGTAAAAGA
RBAM_026860	*ccpA*	RBAM_006550	*gutA*	ACTTTTGCCTAAGA
RBAM_026860	*ccpA*	RBAM_025620	*lcfA*	ATGAAACGCTTTCC
RBAM_026860	*ccpA*	RBAM_033080	*rbsR*	ACATTTGCCAATGT
RBAM_026860	*ccpA*	RBAM_006270	*ydjK*	TGATAACGCTTTCA
RBAM_026860	*ccpA*	RBAM_018230	*yxjC*	ACATTCGCGAATGA
RBAM_026860	*ccpA*	RBAM_036810	*csbC*	ACTTTATCAATAGA
RBAM_026860	*ccpA*	RBAM_037260	*rocR*	TTTTTGCGTTTCCC
RBAM_026860	*ccpA*	RBAM_029610	*yurJ*	AAAATGCGTTTTAA
RBAM_026860	*ccpA*	RBAM_036780	*iolA*	ACATTCGCAAATTA
RBAM_026860	*ccpA*	RBAM_017590	*yobO*	AATAAGCGCTTAAA
RBAM_026860	*ccpA*	RBAM_031840		ATAAAGCGCTTTCA
RBAM_026860	*ccpA*	RBAM_017350	*xylA*	TGTATAATCTTTAA
RBAM_026860	*ccpA*	RBAM_017320	*xynP*	ACTTTCGCGAAAAT
RBAM_026860	*ccpA*	RBAM_009700	*citA*	TGAAACAGCTTGCA
RBAM_026860	*ccpA*	RBAM_036480	*deoC*	AAAAAACTCATTCA
RBAM_026860	*ccpA*	RBAM_036400	*hutP*	TTAAAACGCTTTCA
RBAM_026860	*ccpA*	RBAM_004800	*dctP*	ACTTTTGCGATAGT
RBAM_026860	*ccpA*	RBAM_031510	*sigL*	ACCTTTGCGATAAG
RBAM_026860	*ccpA*	RBAM_036360	*bglP*	ACATTCGCAACTGT
RBAM_026860	*ccpA*	RBAM_009540	*glpF*	TGACACCGCTTACA
RBAM_026860	*ccpA*	RBAM_031320	*araE*	ACATTCGCAAAAAT
RBAM_026860	*ccpA*	RBAM_036190	*bglS*	ACTTTCGCAACCGT
RBAM_026860	*ccpA*	RBAM_036180	*citH*	TGAAAGCGCTTACC
RBAM_026860	*ccpA*	RBAM_035650	*ywbA*	ACATTCGCAAAGGT
RBAM_026860	*ccpA*	RBAM_036040	*msmX*	TCTTTCGCAAATGT
RBAM_026860	*ccpA*	RBAM_036010	*cydA*	TGAATCCGTTTGCA
RBAM_026860	*ccpA*	RBAM_004400	*ydaB*	TACAAACGTTTAAA
RBAM_026860	*ccpA*	RBAM_034980	*rocA*	ACTTAAGCAAACTT
RBAM_026860	*ccpA*	RBAM_034850	*eutD*	ACTTTCGCGATATT
RBAM_026860	*ccpA*	RBAM_028160	*gbsA*	GCAAAACGCTTTAT
RBAM_026860	*ccpA*	RBAM_035290	*sacP*	ACTTTCGCGTAATT
RBAM_026860	*ccpA*	RBAM_007970	*treP*	ACTTTCGCGATATT
RBAM_026860	*ccpA*	RBAM_008390		TGACGCCGATTTCA
RBAM_026860	*ccpA*	RBAM_008360	*glvA*	ACATTTGCAATAGT
RBAM_026860	*ccpA*	RBAM_008300	*acoA*	ACCTTAACAATTTT
RBAM_026860	*ccpA*	RBAM_007810	*citM*	ACATTCGCATAAGT
RBAM_026860	*ccpA*	RBAM_010530	*yhfS*	ACTTTTACAATGTT
RBAM_026860	*ccpA*	RBAM_026810	*acuA*	TGAAAACGCTTTAT
RBAM_026860	*ccpA*	RBAM_022450	*mmgA*	ACATTCGCAATAGA
RBAM_026860	*ccpA*	RBAM_026800	*acsA*	TATTTCGCAAAAGT
RBAM_026860	*ccpA*	RBAM_003280	*amyE*	ACATTCGCAATTGT
RBAM_026860	*ccpA*	RBAM_022380	*bkdR*	GGAAAACGCATGCC
RBAM_026860	*ccpA*	RBAM_002750	*gltP*	TGATTACGCTTACA
RBAM_028630	*yufM*	RBAM_036020	*cimH*	AAATTCATAATATATGAACAATTTAAATTTTTTAGTATACTTTAAATGGGG

RBAM_028630	*yufM*	RBAM_026990	*malS*	TACAAGATCAGGTCGAAACCTTTGAAAACGAGTATCAGCTGACAGAAGGCA
RBAM_028630	*yufM*	RBAM_007730	*yflS*	TTCCTTTGTGTTTTTTAATTAACAAAAACGTTTATTAACTTAGTTAAGTAG
RBAM_028630	*yufM*	RBAM_034210	*ywkA*	ATCAATTTATTGAATTTTTTCTCATACTTTTTTAATTCATTAAAAGTAGAG
RBAM_028760	*comA*	RBAM_034530	*rapF*	AACGCGATAGGAAGG
RBAM_028760	*comA*	RBAM_007720	*pel*	AACGCCAAAGGCGTT
RBAM_028760	*comA*	RBAM_033860	*rapB*	TGCGTCAAGCGGCTA
RBAM_028760	*comA*	RBAM_003080	*rapJ*	AAAGCCAAAGGCAGC
RBAM_028760	*comA*	RBAM_012450	*rapA*	TGCGGATATCCGAAA
RBAM_028760	*comA*	RBAM_035710	*dltA*	AAAGCGGGAAGAGGT
RBAM_028760	*comA*	RBAM_003660	*srfAB*	TCCGGGATGCCGTCA
RBAM_028760	*comA*	RBAM_003650	*srfAA*	TGCGGGATGCCGCAA
RBAM_028760	*comA*	RBAM_004010	*rapC*	TGCGCTCTGCCGAAA
RBAM_029510	*pucR*	RBAM_033650	*ywoE*	ACACATTACACGAGTTTTTAATTTTGC
RBAM_029510	*pucR*	RBAM_029590	*pucF*	CATCTTGTCAGTTTTTCAGTTTAAAAG
RBAM_029510	*pucR*	RBAM_029500	*pucH*	ACCAATTATACTTATTTTTAGTAGTTT
RBAM_030010	*yusT*	RBAM_018620	*gltA*	TAGAGTAAAACTCTA
RBAM_030010	*yusT*	RBAM_030010	*yusT*	ATCACAAAAAATGAT
RBAM_030210	*yvqC*	RBAM_009580	*yhcY*	ATGCGGAGTAGGCTTTCATACTCCCTT
RBAM_030860	*ytlI*	RBAM_030850	*ytmI*	TTCATTAATTTTCTTGAACCTTTCCTTTACTTTTTTTGATTAGTCA
RBAM_030940	*yvaN*	RBAM_012450	*rapA*	CACTTTTGGGGGTAAATTATGTCTTAT
RBAM_030940	*yvaN*	RBAM_004600	*rapH*	AGCATAACTAGGACAGTAACTATAAAT
RBAM_030940	*yvaN*	RBAM_004010	*rapC*	AAAAACAAAATCAGCTGCGGATTCAAG
RBAM_030940	*yvaN*	RBAM_019730	*rapA1*	CATACCGAATTGAGAAATTAAGAAAAA
RBAM_030940	*yvaN*	RBAM_033860	*rapB*	GACAAACGCAGTTCGCCGATACTACAA
RBAM_030940	*yvaN*	RBAM_003080	*rapJ*	CACTTAAGAGAGAAAGAAATTAACTAA
RBAM_031310	*cggR*	RBAM_031310	*cggR*	CTATGATGGGACGTTTTTTGTCATAGCGGGACATTTTCTGTCCAGT
RBAM_031330	*araR*	RBAM_025870	*abnA*	TTTTTGTCTGTACAAATA
RBAM_031330	*araR*	RBAM_025860	*araA*	TTTAACAAGCATGTTTAT
RBAM_031330	*araR*	RBAM_032960	*ywtG*	TTTATTAATCATTTTATT
RBAM_031330	*araR*	RBAM_006270	*ydjK*	GTTTTGTACGGTCAGATT
RBAM_031330	*araR*	RBAM_036810	*csbC*	TTTTTTTATGTTTATATT
RBAM_031330	*araR*	RBAM_031330	*araR*	AATTTGTCCGTATACATT
RBAM_031330	*araR*	RBAM_031320	*araE*	TTACATATGCCTGTTTAA
RBAM_031510	*sigL*	RBAM_008300	*acoA*	CAAAGGCTGGCATGCTTCTTGCATTTATAAGTGTGA
RBAM_031510	*sigL*	RBAM_019060	*dhaS*	GACGGTTAATAATACTTAGTAAATATTTTGAGGAAA
RBAM_031510	*sigL*	RBAM_021110	*gudB*	GGCAGAAAAAAATCCGCTTTATTACCTTTTTAGTAA
RBAM_031510	*sigL*	RBAM_037250	*rocD*	TTCGGATTGGCACGGAACTTGCTTTATAAAAAGGGA
RBAM_031510	*sigL*	RBAM_036780	*iolA*	AAAAGTATGGTAAAAATAATGAAAAAATTTTGATTT
RBAM_031510	*sigL*	RBAM_034990	*rocG*	AAAAACCTGGCATGAATCTTGCATATAAAAAGGGGC
RBAM_031510	*sigL*	RBAM_034980	*rocA*	GCTATGCTGGTATGCTTCTTGCATCTTATAAAGCGT
RBAM_031510	*sigL*	RBAM_028160	*gbsA*	TTAATATTGTTAAAAACATTAAATTTTTATTTAACA
RBAM_031510	*sigL*	RBAM_004160	*gabD*	TGTAGTTGATCGTAAGTTACCGAAATGATATCGTCT
RBAM_031510	*sigL*	RBAM_011220	*argD*	TGCGTAATTACTTTCTTCCGTAGTAAGGGCAATAGC
RBAM_032640	*degU*	RBAM_021440	*sipS*	GAGGAAATCACTTGAAATCA
RBAM_032640	*degU*	RBAM_014150	*sipT*	ATAAAATACTACTCAATACT
RBAM_032640	*degU*	RBAM_035660	*epr*	GCTTACATTTGCATACAGAA
RBAM_032640	*degU*	RBAM_035290	*sacP*	CTTTATTTCATCCAAAAAAA
RBAM_032640	*degU*	RBAM_007970	*treP*	TACAAGTTTGTAAGACAAAA
RBAM_032640	*degU*	RBAM_010600	*comK*	AAAAAATATCATATACCTAT
RBAM_032640	*degU*	RBAM_010500	*aprE*	AATAAAACAAACTGAAAAAA
RBAM_034460	*fnr*	RBAM_034470	*narK*	GCACACTACATTAAGTGTTAG
RBAM_034460	*fnr*	RBAM_034440	*arfM*	AGCTGTGAAATACATCACTGC
RBAM_034460	*fnr*	RBAM_034430	*narG*	GTGTGTGACATAGTTCACAAG
RBAM_034830	*ywfK*	RBAM_030610	*yvgR*	ATCATTAGTAATCTTAATCGTTGTCA
RBAM_034830	*ywfK*	RBAM_034830	*ywfK*	AGTAATCATTTTATATAAATATACTA
RBAM_035300	*sacT*	RBAM_035290	*sacP*	GCGGGATTGTGACTGGTAATGCAGGCAAGACCTAAAATT

RBAM_035300	*sacT*	RBAM_007970	*treP*	CAAAGTCCCAGAAAGCGTCATATATACACTACTTTCGCG
RBAM_035300	*sacT*	RBAM_037650	*sacB*	GCAGGATTGTTACTGATAAAGCAGGCAAGACCTAAAATG
RBAM_035610		RBAM_035610		GACTCTTACGTTGCGTAATA
RBAM_036200	*licT*	RBAM_036360	*bglP*	GGATTGTTACTGCAAATCGCAGGCAAAACCTAA
RBAM_036200	*licT*	RBAM_036190	*bglS*	GGATTGTTACTGATAAGCAGGCAAAACCTAAAT
RBAM_036400	*hutP*	RBAM_036410	*hutH*	TGATAGGGGGCTATGCGTGAAATATCATTTTCATGGCATAGCTCCTTTTTGT
RBAM_036490	*deoR*	RBAM_036480	*deoC*	CATGAACATAATTTCAATTACCAATTTACATATGTTCAA
RBAM_036790	*iolR*	RBAM_006270	*ydjK*	TAGTTAAAAAATAGATTTTC
RBAM_036790	*iolR*	RBAM_036810	*csbC*	TGCGCGCCTAGTTAATTTTC
RBAM_036790	*iolR*	RBAM_036790	*iolR*	CTATTGATTAACTTCTGGTG
RBAM_036790	*iolR*	RBAM_036780	*iolA*	GTGGTCTTCAATTAGTTATC
RBAM_036790	*iolR*	RBAM_031320	*araE*	CTTATTCTTTATTTGTGCGT
RBAM_036790	*iolR*	RBAM_034980	*rocA*	ATTTTGAATGGGTGCAAAAT
RBAM_036790	*iolR*	RBAM_028160	*gbsA*	TTGTTAAGTCATTTCTTTAT
RBAM_036790	*iolR*	RBAM_004160	*gabD*	CTGTGAGTTCAATTATTGGA
RBAM_036790	*iolR*	RBAM_007590	*yfmT*	TACAAATGTACAAAGTTATA
RBAM_036790	*iolR*	RBAM_032960	*ywtG*	TTATTAATCATTTTATTACC
RBAM_036790	*iolR*	RBAM_019060	*dhaS*	CTTCTGTTTCATTTTTGACT
RBAM_036850	*yocG*	RBAM_036870	*des*	TCATATTCGGGGCATGA
RBAM_036850	*yocG*	RBAM_005380		TCATCTATGGTGTATGA
RBAM_037260	*rocR*	RBAM_007590	*yfmT*	TTAATTTCTTAAAAGGTGA
RBAM_037260	*rocR*	RBAM_021110	*gudB*	AACGAAAAATCATGACGCG
RBAM_037260	*rocR*	RBAM_037250	*rocD*	GCGTTTTTTTAAAACGCGA
RBAM_037260	*rocR*	RBAM_036780	*iolA*	TTACTTTTTTAAAACTAAA
RBAM_037260	*rocR*	RBAM_034980	*rocA*	GCGTTTTAAAGAAACGCAA
RBAM_037260	*rocR*	RBAM_028160	*gbsA*	GGTGAAAGGTTTTTTTGAC
RBAM_037260	*rocR*	RBAM_004160	*gabD*	GTTTCTTCCGGAAAGGTAA
RBAM_037260	*rocR*	RBAM_004150	*gabT*	CACGAAAAAAACCTTTTCTT
RBAM_037260	*rocR*	RBAM_022380	*bkdR*	ATCGAAAAAATATTTTGTC
RBAM_037260	*rocR*	RBAM_037260	*rocR*	AGCGCAAAATTTTTTTGCG
RBAM_037400	*yycF*	RBAM_018960	*yocH*	ACAAAGTACATTTCCTTGACATTAGAT
RBAM_037400	*yycF*	RBAM_032870	*tagD*	AACTTAACAATAGTTTTACAAAAATAT
RBAM_037400	*yycF*	RBAM_013590	*ykvT*	ATTGTGCACATTTGTTTTACATTTTAG

Table 1: Binding sequence predictions for *Bacillus anthracis* A0248. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BAA_0001	*dnaA*	BAA_0001	*dnaA*	TAGAATAGGTGTTAAAC
BAA_0055	*purR*	BAA_0777		GAACATCTCACTTGCTACAAACTTACGCGAATGATTTATAAATTGATAAAG
BAA_0055	*purR*	BAA_5586	*glyA*	TGAAAACATGAAGGTTCGAAAGAATTACAACCTGAATTTTCGTTTTTTCTG
BAA_0055	*purR*	BAA_0343	*purE*	TGATTTTGTGCTTGCAATAACTAATCATTATTTTTTTGTAAGCTAATAAAC
BAA_0055	*purR*	BAA_0313		TCAAATTATGCTTGCTTTAATATTTTCAATTATTTTTACAAACATAAATTT
BAA_0055	*purR*	BAA_4326		GCTAAAAATGCATAAATATTTATTAAGGTAAATATTCATTTGCAATAGTAA
BAA_0055	*purR*	BAA_4055	*pyrR*	CATAAGAAAACTGAGATACTGACAACAGTTAAATAATGATCCTTTAACACA
BAA_0055	*purR*	BAA_1660	*xpt*	ACAAAACTCGAACGAAAATGATGTTTTTATAAAATTTGTACGTGTTTTATG
BAA_0055	*purR*	BAA_5751	*purA*	ATTAAGAATGCTTGCTTTTTCAAAAAAATATTTTTTTACAAGCGAAAAACC
BAA_0055	*purR*	BAA_5736	*guaC*	CCTTTGGACGAACATTTTTAAACAAACTGAATAAAATATTCGTATTTTAGG
BAA_0055	*purR*	BAA_0855		CAATTTTTTGCTTATAATATATATTGACAACACTTTTAATAAGCATTTGTC
BAA_0109	*sigH*	BAA_1840	*fumC*	GAGGAATTTTTTTAAATAAATATAGAATATCTATATA
BAA_0109	*sigH*	BAA_0692	*aspA*	AAAGGAAATTATGGTGCCTCGTACGGAAGTTATAGGT
BAA_0109	*sigH*	BAA_5451	*yfiA*	AGAAGGAGTTTTTAAGCCAGTGTCGAAATTATAGTAC
BAA_0109	*sigH*	BAA_4537	*dnaG*	TAAAGGAATTTTTTAATTTATATCGAATACAAAATAC
BAA_0109	*sigH*	BAA_4413	*spo0A*	AAAGGGAATTTTCACACAATTGTCGAACAATTCATGT
BAA_0109	*sigH*	BAA_4072	*ftsA*	AAAAGGGAAACGTATAAAGACGTTGAATATTTTTCTT
BAA_0191	*gntR*	BAA_0191	*gntR*	ATACTTGTATACAAGTATACTT
BAA_0238		BAA_0592		ATATCGATATTACAT
BAA_0238		BAA_0238		GATTCAAAATGATAT
BAA_0238		BAA_3587		ATATAAAATAAAGAT
BAA_0306		BAA_1993	*ldh*	ACGGATACATTTTCAAATACTTGAATATCTTTT
BAA_0306		BAA_2012	*cydA1*	ACAAATTTATCTTTCTATATTTCTAAGAATTGA
BAA_0306		BAA_1484	*ilvB*	AGCAAATAATCTTCTTAAGGCTTTCAATCAATA
BAA_0306		BAA_0972	*alsS*	ACTATAATGATAAATAATACTTCACCACATTGA
BAA_0306		BAA_5272	*ldh*	ACAAAAACTTTGCTCAATATTTCACAAAGTATC
BAA_0306		BAA_5136	*ldh*	GTATTAAACACTTCATTAAGTGTTATTATGCAC
BAA_0306		BAA_5061	*cydA2*	AATAAGATATTTTCAGATAATCGACATAATTCT
BAA_0382		BAA_1581	*gudB*	AATACACAATTACTTTGTG
BAA_0382		BAA_0957	*gapN*	GAGTAGTGTAAAAAGGTGG
BAA_0382		BAA_1236	*rocD*	AATGATATAATATTTAGAG
BAA_0382		BAA_0383	*gabD*	ATTGCAAATGTTTGTTGGT
BAA_0382		BAA_0381	*gabT*	AGTGAAAAGTTATTTTCCC
BAA_0382		BAA_0363		TCGTTTTTGCAAAACCTTA
BAA_0382		BAA_3638	*dhaS*	GTGTTTACTTTAAACTTAA
BAA_0382		BAA_0551	*rocR1*	TCGTTTTTTTAAAACGCTA
BAA_0382		BAA_4408		TATGCGTAAATATTTTGAT
BAA_0382		BAA_2898		ACATTTTTATAAAATTTAA
BAA_0382		BAA_2824		AAAACAAAATTATATTTTA
BAA_0551	*rocR1*	BAA_2839	*acoA*	TCAAATTAGGTGAACAAAAGTGGACAAAACGAGACAGGTGTCTCATTTTGT CCACTTTTTTATT
BAA_0599		BAA_4334		ACTTTATGAGAATAATTATCA
BAA_0599		BAA_3883		TTATTTTGCTAATAATTATAC
BAA_0599		BAA_3080		ACAATTTTAGGATGATTATAT
BAA_0599		BAA_0951		CCCTTATCATTATGATTATCG
BAA_0599		BAA_1276	*spxA*	TCTATTGTATAATGATTATAG
BAA_0599		BAA_0599		AATATTCTTAATATTTTATTA
BAA_0599		BAA_0474		TATCTTATTAATGTTGTATTA
BAA_0599		BAA_0402	*ahpC*	AATCTTAATAATATCAATTAA
BAA_0599		BAA_4716	*hemA*	AATATTAGTAATATTTCATCC
BAA_0660		BAA_0661		GCGTAGTAGATTATTTAATTAAACCATTTACATTTGAACGATTTAAAGAGG

BAA_0660		BAA_1871	*malS*	CTCTTTTTTGTTTACAATATAGAAATGCTATATATGAATAGAGGTAGGGAT
BAA_0712	*treR*	BAA_0713	*treB*	CAACTGATTGAATATATATGCTCAATTATACA
BAA_1024		BAA_0644		TTTGGAAAAAGACTTAAAAATAAAATAAAACTTTCGCATTAGTAAAAATTC
BAA_1086	*sigB*	BAA_2084	*dps1*	AAAGGTTTATGTAGGGGTTTATTTATAAACAATTTAAG
BAA_1086	*sigB*	BAA_2074		GGATTAATAGATTTATTGAAAGAATAACAATTACTAAA
BAA_1086	*sigB*	BAA_2067	*nadE*	ATTTTAAGGATTAAGTGCGAAAGAGCAAAAATTTTTGT
BAA_1086	*sigB*	BAA_1484	*ilvB*	TTGATAAAAGTTTTTAAAAGGGGGCTTATGAGAACAAT
BAA_1086	*sigB*	BAA_0972	*alsS*	TTTTTTGTGTATTTTTAGAAGAGGTATAAAAATATTAT
BAA_1086	*sigB*	BAA_0957	*gapN*	AAAAGTGTTTATAAAGTAAACTTTTTACAATAGTTTGG
BAA_1086	*sigB*	BAA_0951		ATGTTTACGTCCTAATAAAAAGGGTATCTATTAAATTA
BAA_1086	*sigB*	BAA_5746	*yycF*	TTTGTGTAATTTTACAAAATTGAGTAGTTATTTTAAAA
BAA_1086	*sigB*	BAA_5713		TCATTACAATTATTAGCTTTTGGATATTATTAAAATTC
BAA_1086	*sigB*	BAA_1322		TAAAATAAAAAGAGAGGAAAGTTATAATATAACTGTTT
BAA_1086	*sigB*	BAA_0856		GTTTTTATTATGCACATTATAAGAAAAAGGAGTTGAAA
BAA_1086	*sigB*	BAA_0720		TATTAAAAATATATAAGACTTATAAGAAATATTTTTCC
BAA_1086	*sigB*	BAA_5451	*yfiA*	AATATAAGCAATAAGTGCTACAATTATCCTATTTTATT
BAA_1086	*sigB*	BAA_5410	*clpP*	GATGTTGAAGATACATAAGCAGGGAATTTTAAAAACAT
BAA_1086	*sigB*	BAA_5332		ATCATACATGTAAACGATTTTTCTCCTCAGTACTTTGG
BAA_1086	*sigB*	BAA_5331		ATTTTCTTATTGTTAGGAAGAGGGGGCAATGCAATTAT
BAA_1086	*sigB*	BAA_5319		AAAAATAAACTTAAAAGACAAAAAGAAAGAAAATAAGT
BAA_1086	*sigB*	BAA_4871		CAAAAAAAGAAAATCGGTAACGTAATTTAAAAGATTAG
BAA_1086	*sigB*	BAA_4844	*phoP*	GGTTTCAAGTCTATAACTTGGAGGAAAAGAATGAACAA
BAA_1086	*sigB*	BAA_0383	*gabD*	CGTTTCGCCCTATAGGCGGAAGAGAATTAAAAAAGGTG
BAA_1086	*sigB*	BAA_0363		AAAAATAATATAAAATTTTTAATTAATTAAAAATTTTA
BAA_1086	*sigB*	BAA_4654	*relA*	AAATAATAACCAACGGGTCAAAGAATGCGTAATCTTCA
BAA_1086	*sigB*	BAA_5086		TCTTTCCATTACAAAAAAACAGGGAGAGATGAAAAATG
BAA_1086	*sigB*	BAA_0100		CGTTTTGATTAGCGAAACAATGGTTAATAATGAGTAGG
BAA_1086	*sigB*	BAA_4441		TGTTTAAATACTGCGAAATTTGTCTATAAAAAGAATAA
BAA_1086	*sigB*	BAA_3732		AAAAGACTTATAAAAATGAAATTAAGGAAAACTTTTGT
BAA_1086	*sigB*	BAA_3731		TTTTTCATAATGAATTAAAAAAGGAGTGGAGAAAATGG
BAA_1086	*sigB*	BAA_3638	*dhaS*	TAAGAGAATGTAATGTATAAAAAATGTTAACATTGTGT
BAA_1086	*sigB*	BAA_3080		GCTATAAGTAAAAAGGTATTAAAGAAGATAAAGTTGTC
BAA_1086	*sigB*	BAA_2566	*csbX*	AAAAGAAGTAATATGTGTGAACTAAGATTTAACTTTTG
BAA_1086	*sigB*	BAA_2439		GTTTTTATTTTTTCTAAAAAAGGAGAATAAAGATGGAG
BAA_1118	*glpP*	BAA_0744	*glpT*	CACGGAGAATCGGAGTACAC
BAA_1118	*glpP*	BAA_1119	*glpF*	GACGGAGAAAAGTAGAGATC
BAA_1136	*hpr*	BAA_0772		ATTCATTTTAAAGAAAAAAG
BAA_1136	*hpr*	BAA_0696		ATAAAGGTATATTAAGATAA
BAA_1136	*hpr*	BAA_0683		GATATAAATATTTTGAAGAA
BAA_1136	*hpr*	BAA_5313	*nprB*	ATAATGTTTAAACCTAAAAT
BAA_1528		BAA_2326		TCGTTTTCAACATTGGAATAATAATAA
BAA_1565	*resD*	BAA_2211	*nirB*	AAGTGTTTTTTATATGT
BAA_1565	*resD*	BAA_2198		GTAAGTTTTTGAAAAGT
BAA_1565	*resD*	BAA_2012	*cydA1*	CTATATTTTTTAATACT
BAA_1565	*resD*	BAA_1536	*hmp*	TTAGATATTATTGTAAA
BAA_1565	*resD*	BAA_5061	*cydA2*	GAAGGTGTTTAAAGAGT
BAA_1872		BAA_0559	*glsA1*	TTACAATTTATGCAGAAGAACAGGATTTGCTTCGTTATTAGGAC
BAA_1872		BAA_3205	*glsA2*	TAAGTGTGTTGGCTATGATTGTATTTGGTTCTGTTGCAAAGATT
BAA_1959		BAA_1960	*deoC*	AATGAACAAAATTTCAAAAGTGTATTTACATTTGTTCAA
BAA_2198		BAA_2191	*narG*	AGTGTAGACATAAAATGTTGG
BAA_2565		BAA_1484	*ilvB*	TCTTTCGCAACTCT
BAA_2565		BAA_0972	*alsS*	ACATTCGCATAAGA
BAA_2565		BAA_4899	*ackA*	TACTTTGCAAAAGT
BAA_2565		BAA_2406	*mmgD*	TGTAAAGGCTTACA
BAA_2565		BAA_1960	*deoC*	AGAATACGTTTTAT
BAA_2565		BAA_1869		ACTTTCGCAAGAGT
BAA_2565		BAA_2283		ACTTTAGCAAAAAA
BAA_2565		BAA_2012	*cydA1*	TGGAAACGCTCGAA

BAA_2565		BAA_1520		TCCTTAGCAAGAGT
BAA_2565		BAA_1475		TCTTTTGTAACAGT
BAA_2565		BAA_0981		TCTTTCATAAATGT
BAA_2565		BAA_0957	*gapN*	TCTTTAGCAAAAGT
BAA_2565		BAA_1365	*potA*	ACCTTCCCCAAAAA
BAA_2565		BAA_5663	*eutD*	AGATAATGTTTTCC
BAA_2565		BAA_0748	*rbsR*	ACATTTGCCAATGT
BAA_2565		BAA_0713	*treB*	ACATTTGCCATAGT
BAA_2565		BAA_1179		ATTTTACGCATACA
BAA_2565		BAA_1119	*glpF*	TGACACCGCTTTCA
BAA_2565		BAA_0661		TGAAAGCGTTTTAA
BAA_2565		BAA_0649		TAAATACGAATACA
BAA_2565		BAA_0644		ACTTTCGCATTAGT
BAA_2565		BAA_5401	*sigL*	ACCTTACCCACAGA
BAA_2565		BAA_4927	*acuA*	TCCTTCGCAAAGAT
BAA_2565		BAA_4926	*acsA*	TAGAAACGCTTCCT
BAA_2565		BAA_0551	*rocR1*	TATAACCGTCTTCA
BAA_2565		BAA_4907		ACTTTTTAAACAGT
BAA_2565		BAA_4850	*citZ*	ACCCTCTTAAAAGT
BAA_2565		BAA_0383	*gabD*	TTACAACACTTTAA
BAA_2565		BAA_0363		ACATTTGCAAAAAG
BAA_2565		BAA_5061	*cydA2*	GGATATCGCTTTAT
BAA_2565		BAA_0191	*gntR*	TAAAAGCGCTTGCA
BAA_2565		BAA_4408		ATGTTCGCCAGAGT
BAA_2565		BAA_4255		ACTTTTGCAAAATA
BAA_2565		BAA_3638	*dhaS*	AATAAACGCTGTCT
BAA_2565		BAA_3506		TGAGAACGATGCAA
BAA_2565		BAA_2898		AATTTAGGCAATGA
BAA_2565		BAA_2839	*acoA*	GTTTTATGCTTTCA
BAA_2565		BAA_2824		TGAAAACGCTTTTA
BAA_3517	*sigI*	BAA_3517	*sigI*	ATTTTTCGACCCCCATAAAACTTTGTATTCCTCCGAATATGTATAGTGAAAAAA
BAA_3688	*htrA*	BAA_3688	*htrA*	CACCATCCTCATCGTCTTCTTTTACGATACAGAAATCGACCTTATCTTCACTTAGCAGATAGATCTTGACAAAATCATCTACCTCTTCTTTAATATAACTGTGTATTTCTTAA
BAA_3688	*htrA*	BAA_5741		ACAAAGTGTACTACGCCGTCTCGGCTACAAGATACGAAAAGTATTGTTATTATCTTTCGATCGAGAATAATGTCTATGCCCTATGCACTCACTGGCATACTTTCCACAATAAT
BAA_3779	*lexA*	BAA_2352		TAAGATTGTAAACCAGATTTAG
BAA_3779	*lexA*	BAA_2127		TTTAAACAACAATCATACGCTT
BAA_3779	*lexA*	BAA_1102		AAAATAGAGTGAAAGTTTTATG
BAA_3779	*lexA*	BAA_0691		AACGTTAATAAACAAGCTTAAT
BAA_3779	*lexA*	BAA_4860	*dnaE*	TCTTTTTTCATACAAGAGAATG
BAA_3779	*lexA*	BAA_4772	*uvrC*	TTTTCTTGCATACAACCCACAA
BAA_3779	*lexA*	BAA_0359	*pcrA*	ATGGCTATAAAATAAATCTTAT
BAA_3779	*lexA*	BAA_4668	*ruvA*	ACCTCTTGTAAACAACCCCATC
BAA_3779	*lexA*	BAA_4603	*vpR*	ATCGGTTGTAAGCAAAAATATA
BAA_3779	*lexA*	BAA_0005	*gyrB*	TTAATAGTAAAAATGTTCAAGT
BAA_3779	*lexA*	BAA_3941	*recA*	AAAATCGAATAAATGTTCGATT
BAA_3779	*lexA*	BAA_3779	*lexA*	CAAGCTTGAATACAAACATCTT
BAA_3779	*lexA*	BAA_3686	*parE*	TATCTTTGTATACAAGACTAAA
BAA_3779	*lexA*	BAA_3329		ATAAAAGAACATAAGTTCATTT
BAA_3989	*codY*	BAA_1484	*ilvB*	ACAACTTTTTATAACATAAGAAAGC
BAA_3989	*codY*	BAA_0972	*alsS*	CAAGGTACTGTAAACATTTTGGTAC
BAA_4016		BAA_1261	*fabH*	TCTAGACAAAAATACATTTATGTTTTAAAATTAGTACCAAGTCATAATA
BAA_4016		BAA_0241		TACACTTTTGTAAGAATTGAAATGTAACAATTAATACTTAAATTTATTA
BAA_4016		BAA_1896		AATCTATATTGACATTTTTATAACCTGATATTAGTATCAGGTTATAAAA
BAA_4055	*pyrR*	BAA_4055	*pyrR*	ATAATGATCCTTTAACACAGCCCCGTGAGGTTGAGAAGGTAACGGTTTGAAATA
BAA_4069	*sigE*	BAA_3696		TAATCATAATACAGCGGAATCAAAAATAAAATAAAAACAG
BAA_4069	*sigE*	BAA_2715		GTTAAACTCACAATACGTTTTTTCTCTTACTTATAAGAAT

BAA_4069	*sigE*	BAA_3152		TAATCATATTACTTAAAAATTATGCGTAAGAATATAGTAA
BAA_4069	*sigE*	BAA_2606		TAGGAATTTTTTTGTGCATTAGTACAGAATCTGTTTTAAA
BAA_4069	*sigE*	BAA_3054		ATAAATTATAGAATAATTAAATGGTACTTTAAATAACAAA
BAA_4069	*sigE*	BAA_2410		TTATCATTTCATTTACGTGGAAATGAACGCTTATACTGAC
BAA_4069	*sigE*	BAA_2313	*asnO3*	AGTGCCTAGTTTTCATGTTTTCATAAAAAATTGTTAAAAA
BAA_4069	*sigE*	BAA_1826	*asnO2*	AAAAATAATAAAAAACATACATTACCTTATGAAAGGGAAT
BAA_4069	*sigE*	BAA_2283		TGGAAGTAATTTAGATGGTACGAGAATAAAAAAAGAAGAA
BAA_4069	*sigE*	BAA_2134	*spoIIP*	ACGGTCTATTCTCTAGAGCTTGTACATAACTTGAAATAGA
BAA_4069	*sigE*	BAA_2083		TACGTCTAATTGTCCAAGCTCATACATATGCATAATAGTA
BAA_4069	*sigE*	BAA_1599	*spoIVA*	AAAATTATCAGTATACTTTAACCTGTTTTAATATACTATA
BAA_4069	*sigE*	BAA_0976		TCGTTCTATTTGCTACATTTATTTCATAAATTATGGATAG
BAA_4069	*sigE*	BAA_5746	*yycF*	ATTATAGGAAAAATAAACACATTAAAATGTTTTAACTCAT
BAA_4069	*sigE*	BAA_0876	*spoVR*	TTCTCATGAAAGCTTGTTCTCAATCATACACTTTCTTAAG
BAA_4069	*sigE*	BAA_1239	*asnO1*	GTAATGGCTGACATACTCGCGCCCCGACCAATATATTTTA
BAA_4069	*sigE*	BAA_5556	*spoIID*	TTTAAGATATGAATACGTGCATGGTCATTTTTATAAATCT
BAA_4069	*sigE*	BAA_0633		AATAAAAATGTTATACGTATACTATCTCATTGGTTGGTAA
BAA_4069	*sigE*	BAA_0632		AAAGAATGTTTGAGGACATCCTAGCATACCTTTTATAATA
BAA_4069	*sigE*	BAA_4852	*ytvI*	TTTCACCATCCAAATGTAAGCATGTATACATTTTAAAAAA
BAA_4069	*sigE*	BAA_4844	*phoP*	GAGGTATAATAAATATAGATATATAAATTTTTATATTATA
BAA_4069	*sigE*	BAA_5278		TAGCTAGGTTCAATAAGCACTTGAACCAAATCTACTTCTT
BAA_4069	*sigE*	BAA_4733	*gerM*	GTGGCAATTTTGAAAGTGAAATGGCATATGAAGTTGTAAG
BAA_4069	*sigE*	BAA_5135	*glgB*	TTAAACATTTTTAAGAAAAAAACGTATATTTTTTATAAAT
BAA_4069	*sigE*	BAA_4660	*spoVB*	CTTGTCATTCTTGTCTCGTACGCGCATATAAATTATTGTA
BAA_4069	*sigE*	BAA_0278		TCTGCATATTTTATATCTTTGTCTCATATATTTGTGGTAG
BAA_4069	*sigE*	BAA_4593		TATACAAATTATAACTGATATTTACATAAGTTAACAAAAA
BAA_4069	*sigE*	BAA_0167		TTGTTCTCTTTTTTATATTTACAGCATATATGTAACTGAA
BAA_4069	*sigE*	BAA_0163	*cwlD*	CAAGCATAAACTTTCCTCTTGTCCCATATAAGTAATTAGA
BAA_4069	*sigE*	BAA_4434	*spoIIIAA*	TACTTCTGAAATAGACAACCTCTGCATATGGTTGTACAAA
BAA_4069	*sigE*	BAA_3862	*spoVK*	TAGTCTTTTGTACAAAGTGTAGCCCATATATTACTGGAAG
BAA_4069	*sigE*	BAA_4262		GATAAAAATAATATAATTTATAAAACTAAAAAATCATACA
BAA_4069	*sigE*	BAA_4181	*ctaA*	GCGCAAAATAAGATATGTATTACCTTACTTTTATAAAGCG
BAA_4069	*sigE*	BAA_4162		TGGTCTAAATTCACTACGTATCCACGTATACTGAAATGAA
BAA_4217		BAA_2406	*mmgD*	TTTATTTATTTAAAA
BAA_4217		BAA_4850	*citZ*	TGTATTTTGTTAAAA
BAA_4217		BAA_3705	*acnA*	ATTATTTACTTATTT
BAA_4316	*sigF*	BAA_2403		AAATATTAGACACTTAATCCTTTTATAACTTACGAG
BAA_4316	*sigF*	BAA_2303		AAACAAAATAAATCATTAACTTACGAAAAAAAAGTA
BAA_4316	*sigF*	BAA_1639		ACATATTATTACATTTTGGAATCGTCAAAATACTGG
BAA_4316	*sigF*	BAA_1542		AGTAATTGTAAAGTCTTTTAAATAGAAAAAAACTAA
BAA_4316	*sigF*	BAA_0951		AATACTAAAAATGTATGTTTTTGCTATTATGTACAA
BAA_4316	*sigF*	BAA_0872	*gerYA*	TAAGTATAAATTCCCGTCTTTCCCAAAAACTAACAC
BAA_4316	*sigF*	BAA_0791	*gerLA*	CCAAAAATGTATCTATAAAGAAAGACAATATAATAG
BAA_4316	*sigF*	BAA_0717	*gerKA*	AAGCATAAATTTCTCACAAAAAGCAAAAATTAACAG
BAA_4316	*sigF*	BAA_5597		ACACCTTACTCTATTTTTTTGTGGAAAAATTCATAT
BAA_4316	*sigF*	BAA_5551		AATGTCTAAAACAGCTTTTCATTAATCAATATGCAA
BAA_4316	*sigF*	BAA_1158	*pbpF*	TATACATAAAGATACTTCTTTTTCAAATAATAAATG
BAA_4316	*sigF*	BAA_4997	*gerHA*	AAAGGTAATAGTATTTATGATTTTTCGAAACCAGAA
BAA_4316	*sigF*	BAA_0425		CACACATGAATTTCATATAATATGGAAATATTAAAA
BAA_4316	*sigF*	BAA_0233		GAGGATAAAAACGATACATAAGTCGAATAATTATTT
BAA_4316	*sigF*	BAA_5054	*mutTA*	AACATTAACTACTTTTGTTATCTTTAATTTTCCTAC
BAA_4316	*sigF*	BAA_4565	*gpr*	AAGAATGACCTTATATAACTTTGGGAAATCTAACGA
BAA_4316	*sigF*	BAA_4528	*nfo*	TTTAATACGATAGTAATATAGATAACATATAACTTC
BAA_4316	*sigF*	BAA_4445	*splB*	ATTAATAAAACGTAAAAAGAATAAAAGAAATACTTT
BAA_4316	*sigF*	BAA_4441		CATGTTTAAATACTGCGAAATTTGTCTATAAAAAGA
BAA_4316	*sigF*	BAA_4319	*dacF*	TGGTATTAAAAAATAATAAATTGGAAAAAATAGGTT
BAA_4316	*sigF*	BAA_4313	*spoVAA*	TATGTTAATTGATGGTACAATCCAAAATATTAAAAA
BAA_4316	*sigF*	BAA_3659	*gerSA*	ACACTATAATTATAGCGTTAAAGTTAAAAATATACA

BAA_4316	*sigF*	BAA_4068	*sigG*	CACAGTCATACATTTCGCGTTTGGAAAAAAGAATCA
BAA_4316	*sigF*	BAA_2812	*sleB*	GAAAATAGGAAAAACATATAACAAATGTATAAAAAA
BAA_4316	*sigF*	BAA_3199	*gerAA*	AGTCAAGAGATAAACAAGTGAAAAAAATATTATTCA
BAA_4316	*sigF*	BAA_3178		AATTATTATTTCATAATTTGTAGCATATTAAAAACA
BAA_4316	*sigF*	BAA_3080		CATGTATAAATTCATTTTTAAGATTGAAATTTTAAT
BAA_4334		BAA_3765		ATTACTTTCTTGTACCAACCATATTTCTTTACATACT
BAA_4334		BAA_2501		ACATATAGATGAAAAACCGTATTCTTTTCATTAAGAA
BAA_4334		BAA_2426	*bacA*	TTTTTACATCGATAATGATAATCATTATCATTATTTG
BAA_4334		BAA_0958		TGATTGTATGGGGTTTGATAACATTTTTCAAATGAAA
BAA_4334		BAA_0698		TATGCCAACTTTTATTAAGAGTAAAGGCAACTGTTAC
BAA_4334		BAA_5441		TTAATAAACTATTACCAATAGTAAATCCCTCAGAAAA
BAA_4334		BAA_0409		TTGTTTTACTATTACTAATAGTGATAACTAAGAAATT
BAA_4334		BAA_5196		TATGCTATCATTTATTTGAAAGTATTATCATTTGCAG
BAA_4334		BAA_4615		ATAATCAACCCTTACTGATATTTATAAATGACGGTAA
BAA_4408		BAA_4407	*yqiS*	TTATACTCACACGTTTTTTTTCGTACGTTATTTA
BAA_4413	*spo0A*	BAA_0074	*spoIIE*	TTTTGACAAAAATC
BAA_4413	*spo0A*	BAA_4070	*spoIIGA*	TCTAATACAGATCT
BAA_4416	*argR*	BAA_3638	*dhaS*	TCCTAAAAAATATTAA
BAA_4416	*argR*	BAA_0957	*gapN*	TTGAATAAAAATATCT
BAA_4416	*argR*	BAA_1236	*rocD*	AAGAATAAAATTAAAC
BAA_4416	*argR*	BAA_0383	*gabD*	TTGGATAAAAAAGCGA
BAA_4416	*argR*	BAA_0381	*gabT*	AAAGTAAATCTACGTA
BAA_4416	*argR*	BAA_0363		AACGTAAAAATAATTA
BAA_4416	*argR*	BAA_4376	*argC*	AATTAAAAAGTAAGTA
BAA_4444		BAA_1947		AATTTGCACTAAGGAAACA
BAA_4536	*sigA*	BAA_2898		ATTTTAAATTTATCGTAAATAAGTGTATTATTTTT
BAA_4536	*sigA*	BAA_2849	*proV2*	TTAGTATCATTATTTATTACGTTCCGTATATTTGG
BAA_4536	*sigA*	BAA_2824		TATTGAAAACGCTTTTATTCAGTTTTATAATAAAG
BAA_4536	*sigA*	BAA_3269	*cypD*	AATATTATATAATAAAAAAGACGTTCCATATTTAA
BAA_4536	*sigA*	BAA_3205	*glsA2*	CTAATATCATTTCTTTTTTAGTAAGATATATTTCA
BAA_4536	*sigA*	BAA_3184	*ansA2*	AGATTAAAATAATTTAATTAATATTGTATCTTTTT
BAA_4536	*sigA*	BAA_3127	*lysP*	AATTGCCATATATTAAAAATCTTAGCAATATATAA
BAA_4536	*sigA*	BAA_3054		ATAATTAAATGGTACTTTAAATAACAAATTGTTAT
BAA_4536	*sigA*	BAA_3051		ATAGTTAAATTGTTTTTAAGCTTTACTTTAGTTAG
BAA_4536	*sigA*	BAA_2494	*asd1*	ATTTTCCAAATGATTAAATTCCTTATAAAATGAAA
BAA_4536	*sigA*	BAA_2446	*thrS1*	AAAATAATACTTTACCTCTCTACTTTTTACTTTCT
BAA_4536	*sigA*	BAA_2426	*bacA*	TATATAATATAGGCTTTATGTATAATTTAGAATTT
BAA_4536	*sigA*	BAA_2406	*mmgD*	ATAGTAATATTTTATTTAACTTTAGAAGATTTTTG
BAA_4536	*sigA*	BAA_2403		TATTAAAATTTCACTTTTTACATTCTATGCTTAAA
BAA_4536	*sigA*	BAA_1993	*ldh*	TTTTCAAATACTTGAATATCTTTTATAAAATGTAA
BAA_4536	*sigA*	BAA_1960	*deoC*	TATTTACATTTGTTCAACTAAGGGTTAGAATGAAG
BAA_4536	*sigA*	BAA_2344	*proV1*	ATTTGAAATGAATTCCATATTCCATCAAAATAAAG
BAA_4536	*sigA*	BAA_2326		GTAGAATTTAGAAATTTCTTAATGTTATAATGATA
BAA_4536	*sigA*	BAA_1896		CATTATCAATTTTAGAAACTTCTCGTATATTATTT
BAA_4536	*sigA*	BAA_1880	*dapG1*	TTTTGCTATATTTTTCACAAACTTTTAAGATAAAA
BAA_4536	*sigA*	BAA_1871	*malS*	TTTTTTGTTTACAATATAGAAATGCTATATATGAA
BAA_4536	*sigA*	BAA_1869		AAACTAATATGCTTTGTTAGACTAATGAATAGTTC
BAA_4536	*sigA*	BAA_1840	*fumC*	TCTTATCATTTAAGAGAAAGTCTGTTATTATAAGA
BAA_4536	*sigA*	BAA_1834		TATTGATTTTTCCCTTTCAATCATTTAACATAAAA
BAA_4536	*sigA*	BAA_2243	*ileS*	TGTTTATTTTTATATAAAACTTAATCAACTGTTAA
BAA_4536	*sigA*	BAA_2211	*nirB*	GGCTTAAAATTGTTAAACACTTAGTAAAGTTTTCA
BAA_4536	*sigA*	BAA_1719		AGTTGTGAATATTGGAAAAATATGTTAATTTCTCA
BAA_4536	*sigA*	BAA_2198		AAAGTAAAAAAATAAACGGAAATAATATTTTTTCT
BAA_4536	*sigA*	BAA_2191	*narG*	AGATTAATATATTCTTTAAAAAACGTTTGAGTAAT
BAA_4536	*sigA*	BAA_2151		TATTTATAAAATTTTCTAAATGTTATATCATCATT
BAA_4536	*sigA*	BAA_1660	*xpt*	GCTTTTCAAAGTATAAAAAGTGCGTTATAATCCTT
BAA_4536	*sigA*	BAA_1639		AATTTACACTTTTTGATGAACTTTATTTATTAGAT

BAA_4536	*sigA*	BAA_2067	*nadE*	TATTGTATATTCGTTAATTTTGCGAAATAATAAAG
BAA_4536	*sigA*	BAA_2012	*cydA1*	TATGTAAAATAATTACTCTTTTTAGATTAAGTTCT
BAA_4536	*sigA*	BAA_1581	*gudB*	ACTTTTCTAAAAGCAATAAAAGGACTATAATGATA
BAA_4536	*sigA*	BAA_1565	*resD*	TTTTGGAAAAAAATTGAACAAACTTTATTATTCTT
BAA_4536	*sigA*	BAA_1542		TTTGTAAAATAGCTTTTCAAATGTTTTTGAATTGT
BAA_4536	*sigA*	BAA_1536	*hmp*	ATAATAACATTTATTTATATCTCCACTTTGCTTGT
BAA_4536	*sigA*	BAA_1520		TGTTGTCAGATTTTAAATTTTGATTTATGATTTTT
BAA_4536	*sigA*	BAA_1510	*cysH*	AACATTTTATTTTATTAATTTTATATAGCCTTTTG
BAA_4536	*sigA*	BAA_1484	*ilvB*	AATAAAGTTGGATAACAACTTTTTATAACATAAGA
BAA_4536	*sigA*	BAA_1475		ATTGTCAAAATAGGTATTTTTTCGGTACAATTAAT
BAA_4536	*sigA*	BAA_0987	*secA1*	TCGATAGTATGGTTTAAAGAATATGTAACTTATGA
BAA_4536	*sigA*	BAA_0981		GCTGATATATTTTGAAAATCGCCGATATAATTCAA
BAA_4536	*sigA*	BAA_0980		GTTTGTTTTAAATAGGATATTAAAATAAGGTAAAA
BAA_4536	*sigA*	BAA_0972	*alsS*	TGTTGGAAAAATAATGCATTTGTACTATAATGATA
BAA_4536	*sigA*	BAA_0957	*gapN*	ATATTAGAATGTTGAAATAACTCATAAGACAGTTA
BAA_4536	*sigA*	BAA_0943	*blT*	CAAATAAAAACTCCTTTTGACTTACTATCAGTAAG
BAA_4536	*sigA*	BAA_5751	*purA*	AGAATATTATTCTACACAAAGAAAAAAAAATATAT
BAA_4536	*sigA*	BAA_5746	*yycF*	ATTTTACAAAATTGAGTAGTTATTTTAAAATAGAA
BAA_4536	*sigA*	BAA_5741		ATTTAGCCCATTTGAGTTTAGATAATAACATGAAA
BAA_4536	*sigA*	BAA_5731	*galE2*	TATGTGCATTGAGATTATAAAATTTTATAATTAAT
BAA_4536	*sigA*	BAA_1379		ATTGATACAAAGAAAAAAGAAGTGATACAATCTTT
BAA_4536	*sigA*	BAA_1344	*sucA*	TTTGGTATTATGTAAATTATTGTAATAACATAAGA
BAA_4536	*sigA*	BAA_1324	*trpE*	TGTGTATAAAAAATTAAATAGATGAAATCGTTTAA
BAA_4536	*sigA*	BAA_1322		TATGGATTTTTCAGAATAATCATGATAGATTATAA
BAA_4536	*sigA*	BAA_0898		TTCGGATTCACATTAAAGAGATAGATAAAATAAAA
BAA_4536	*sigA*	BAA_5683		AATTGACATTAGGAAATTATTGCTATAAAATCAAC
BAA_4536	*sigA*	BAA_5663	*eutD*	AGTTTAAAATATAGAAGAAAAGCAGTATTCTTAAA
BAA_4536	*sigA*	BAA_5644	*speE*	TTTGTACTTTTAACTATACATTTTGTATGTTTTGT
BAA_4536	*sigA*	BAA_1278	*mecA*	ATTTCCATTCTGGAGAATCTTATCATAAAATGAGA
BAA_4536	*sigA*	BAA_1276	*spxA*	AAAATAACATACTCTATTTTCTATTTTTTCTTTCT
BAA_4536	*sigA*	BAA_5626	*maP3*	CTTTTTTATGTTTAAATGTTTTAGAAACACTATTT
BAA_4536	*sigA*	BAA_1261	*fabH*	TCTAGACAAAAATACATTTATGTTTTAAAATTAGT
BAA_4536	*sigA*	BAA_5615		CGACTAAAATAGCAAAAAATGTTACTTCTTGTTTT
BAA_4536	*sigA*	BAA_1256	*clpB*	TATTGTATAAAAAAGTCAGTTGGCGTATAATCAAA
BAA_4536	*sigA*	BAA_5608	*pyrG*	AATTTTCTTTTTTGTTTTTTGTTTAGAAAATAAAA
BAA_4536	*sigA*	BAA_1236	*rocD*	GTTTCGCTTTTTTTACATTTAAATGATATAATATTT
BAA_4536	*sigA*	BAA_0798		TGTTCCAATTTATTTAGTTGTACTAAAAAATATTA
BAA_4536	*sigA*	BAA_0776		TGTAGCATAACAGATAGAAATGTTTTATAATTTGA
BAA_4536	*sigA*	BAA_0748	*rbsR*	GATTGACGTATTTCTAGAAATACGTTATATTATTT
BAA_4536	*sigA*	BAA_0744	*glpT*	CGTTTACATTTTTGTCACAAAGTTGTATAATTTTC
BAA_4536	*sigA*	BAA_0713	*treB*	TGTTGACTAACTTATATATACGAGTTAATATGTAA
BAA_4536	*sigA*	BAA_5586	*glyA*	TCTTTTATGATATAGAAAGTCATGTTAGAATGTAG
BAA_4536	*sigA*	BAA_1182		AATATAATATTATATTCCAATTATTAGACACTTAT
BAA_4536	*sigA*	BAA_5533	*lytR*	AATTTATCTTTTATTTTTAATCAAGAATAATGTAA
BAA_4536	*sigA*	BAA_5532	*galE1*	AAAATATTATTCTCTCACTTATTCTTTAGGATTAT
BAA_4536	*sigA*	BAA_1179		ACTTGCTACTTTAATTTTCTTTTAATATAATTAAT
BAA_4536	*sigA*	BAA_1158	*pbpF*	AAGATAATATAGTTGAGAGAATTTTTTGAAGCTTC
BAA_4536	*sigA*	BAA_1119	*glpF*	TGTTGACACCGCTTTCATTAACGGTTAACATTATA
BAA_4536	*sigA*	BAA_0692	*aspA*	TGTTAGAAAATATTCCTATACATGATTTACTATTA
BAA_4536	*sigA*	BAA_0676	*ald1*	CGTTTTCATAACTTTTAAGTTATGCTAGAATAAGG
BAA_4536	*sigA*	BAA_0665		TTTGTTTCTTTATCAGTAAGAAAGATAAAATAAAT
BAA_4536	*sigA*	BAA_0661		AAATTAATTTGGTAAATGTAAACTTGCTAAATTTC
BAA_4536	*sigA*	BAA_0644		TATTTACCATTCAAATAGATGCTTTTATGATAAAC
BAA_4536	*sigA*	BAA_5450	*secA2*	AAAATAAAATTATTTTTCTCCTCGCATAAAGGATA
BAA_4536	*sigA*	BAA_5436		AATGTAATATTCTTTCAAAGATATTATTCAAATTA
BAA_4536	*sigA*	BAA_5410	*clpP*	GTTTGACCTTTATTGACCATAATTGTATTATAAGA
BAA_4536	*sigA*	BAA_1057	*pssA*	CTCATACGATAAAGTATAAGTAAATTTACTTTTGT
BAA_4536	*sigA*	BAA_0599		AGTACACTCTTTATAAGAATTATAAAATAATAATC

BAA_4536	*sigA*	BAA_4950		ATTGGAAATATAGATGAGATTATAAAACAATTATA
BAA_4536	*sigA*	BAA_0592		TATTTCATAAATCCCGTATTAATAGTATAATTATT
BAA_4536	*sigA*	BAA_4927	*acuA*	TAGATAATATTATTGTAATAAATTTTAAAAGTCAT
BAA_4536	*sigA*	BAA_4926	*acsA*	TACTGAAAATTTTAAATAATGTTATTATAATAGAT
BAA_4536	*sigA*	BAA_4922	*tyrS1*	TTTGACACGCCTATATAAATGATGGTATAAAGAAA
BAA_4536	*sigA*	BAA_0561		TATTTATTATTGTACTTTTTCACCTTACTATTTTG
BAA_4536	*sigA*	BAA_4919	*rpsD*	GGTAAAAGATATAATAGCGTTTGTGTAAAATAAAA
BAA_4536	*sigA*	BAA_4918	*megL*	AAAATAAAATGTGTTTGCGATAATATAGAAAATGG
BAA_4536	*sigA*	BAA_4911	*ezrA*	AGGTCAAAAACAAGCCATTACATGGTATGATTAGT
BAA_4536	*sigA*	BAA_0559	*glsA1*	GTTTTATCATTCCACTTTTTCATGTTATTATTTAT
BAA_4536	*sigA*	BAA_0553		TTTTGCACTCTTTAATAATTTATTTTACGATGAAG
BAA_4536	*sigA*	BAA_0551	*rocR1*	TTTTGCGATTTTATAAGAATTTTAATAAAATTATA
BAA_4536	*sigA*	BAA_4907		GGTTCAAAATTGACAGAAAATTCTTTATTATATAA
BAA_4536	*sigA*	BAA_5343	*tyrS2*	TGTTGACAATTTGTTTTCAATTCAATATAATGGCT
BAA_4536	*sigA*	BAA_4899	*ackA*	AATATACTATACAGTATAAAAATTTATATATGTTC
BAA_4536	*sigA*	BAA_4884	*ald2*	TTTGTCAAATATTGAGATATTATGTTAAAATATAA
BAA_4536	*sigA*	BAA_4850	*citZ*	TGTTCACAAAAGTCAGATAATTGTTTATAATAGGA
BAA_4536	*sigA*	BAA_4844	*phoP*	TTTATATTATATGTTTAATAATCACATTCAGTTAC
BAA_4536	*sigA*	BAA_4836	*speD1*	TGTTGCAAAATGAAAATAAACAAAGTATACTAACA
BAA_4536	*sigA*	BAA_4831	*thrS*	CATATAATTACCTACATAAATGAAATATATTGAAA
BAA_4536	*sigA*	BAA_0474		GATATGATATCTTATTAATGTTGTATTATTGATAA
BAA_4536	*sigA*	BAA_4829	*infC*	TATTGCAAGTATGTTAGTTGTTTGTTACAATATTT
BAA_4536	*sigA*	BAA_0463		CTTTTTATCAATAGGTAAATAATGCTACAATATAA
BAA_4536	*sigA*	BAA_4814	*pheS*	AGTTGCGAACAAATAAAAAACTTCTTATAATAAGT
BAA_4536	*sigA*	BAA_0403		AAGATAAAATGTTTCTTAACACACTATTTATTTTA
BAA_4536	*sigA*	BAA_0402	*ahpC*	ATTTTATTTATCACACAATTCTTTGTAAAATAGAA
BAA_4536	*sigA*	BAA_5281		TATTGCGAAAATTTAAACAGTATCATATATTTATT
BAA_4536	*sigA*	BAA_5272	*ldh*	TATGTATTATTTATAAAAAGGTCAACATTGTTTAT
BAA_4536	*sigA*	BAA_5236		TAGATAATACGTACTTTATAAAATATTTCTTTTAC
BAA_4536	*sigA*	BAA_4770	*sdhC*	CTAATAATATATTGTAAAGAGTAAATATTTTTGTC
BAA_4536	*sigA*	BAA_0399	*amhX*	AATTGACTTTTTTCTTAAAATTCAATATGTTCATA
BAA_4536	*sigA*	BAA_0383	*gabD*	ATAATAAAACTTTCGTATATTTTGTATACCTAGCT
BAA_4536	*sigA*	BAA_0381	*gabT*	TTAATAAAAATTTTGTCATAGTTAATCTCAGAAAA
BAA_4536	*sigA*	BAA_0363		AAAATAATATAAAATTTTTAATTAATTAAAAATTT
BAA_4536	*sigA*	BAA_4716	*hemA*	ATTAAAACATTTTAGCTATATTGCTTTTTATTTAT
BAA_4536	*sigA*	BAA_4708	*valS*	GTGATAACATATATTATTAATTGTTAGTATATTTA
BAA_4536	*sigA*	BAA_0343	*purE*	ATTTCTACTACAAGTATATATTTTAAATAATGAGA
BAA_4536	*sigA*	BAA_0311	*guaA*	TTTTAGAAAAGGAAATGAATTTCTGTAGAATTTTG
BAA_4536	*sigA*	BAA_5136	*ldh*	TATATAATATTTTTAGCAACTATATAAATTTATTA
BAA_4536	*sigA*	BAA_5124	*menF*	TGATTCACATCTTTTTCGTAGTAAATCACAGTTGC
BAA_4536	*sigA*	BAA_5121	*menB*	ATGTGGAACAACCGGAAAAGTTTGATACAATAGTA
BAA_4536	*sigA*	BAA_4680		TGGATATAAATGTATCAAAATTTTATTACTGTTCT
BAA_4536	*sigA*	BAA_4679	*nadB*	TCTTGTCATTATTTTAAAACTATGTAAATATAGGT
BAA_4536	*sigA*	BAA_4646		AATTTCTTATAATACTATATTATGCTATAATTTCA
BAA_4536	*sigA*	BAA_4623		GATATCACATTGTTTCTTTTCTTCCTTTATGGTAC
BAA_4536	*sigA*	BAA_0238		ATTCAAAATGATATTAACTATCATTTATAATGATA
BAA_4536	*sigA*	BAA_5061	*cydA2*	AAAGTGATATTTGATTAACATCTATTACATTTTTT
BAA_4536	*sigA*	BAA_5054	*mutTA*	GAAATAAAATGTATTAAAACATTAACTACTTTTGT
BAA_4536	*sigA*	BAA_4593		TGTTTAATATTGACTATAAATGTATTCAATTGTTT
BAA_4536	*sigA*	BAA_4563	*lepA*	CATTGAATCTTTGCTGCTCTATTGATATAATCAGT
BAA_4536	*sigA*	BAA_0195	*yqjI*	TATATGATATTATGTCTCATTGTATGTACCTTTCC
BAA_4536	*sigA*	BAA_0191	*gntR*	GCTTTCCTCTAATAGAACCAAAGGAAACGATAAAA
BAA_4536	*sigA*	BAA_0118	*rpoB*	CCTATAATATAGTATTTTAGTTTTTTTGCAACTGC
BAA_4536	*sigA*	BAA_0109	*sigH*	AAAATGACATATTATAAAGATTTATTTATCGCCAG
BAA_4536	*sigA*	BAA_4499	*metI*	GAAATAATAAAATAAATACATCGATTTTCCTTTAC
BAA_4536	*sigA*	BAA_4487		ATTTACAATAGTAAGAAAAATAAACTATAATGAAG
BAA_4536	*sigA*	BAA_4484	*comGA*	AAAGTCGTATACTTGTAGAAGATTAAATCTATTTA

BAA_4536	*sigA*	BAA_0081	*pabB*	GAGTGAATGACTAGAAAACTTCATGTAAAATAAGA
BAA_4536	*sigA*	BAA_0074	*spoIIE*	TGTTGACTTTAAATTATATCTGAGGTAAGATACTA
BAA_4536	*sigA*	BAA_4419	*ispA*	AGAATAAAATTTCTCCCATACCTTAATAGATTCGA
BAA_4536	*sigA*	BAA_4413	*spo0A*	GCTCCGCAAGAGGTGTTTTTTGTTGTAAAATAAAA
BAA_4536	*sigA*	BAA_0059	*glmU*	GTAAAACATCTTATAAGAATTATGGTAAAATTTAA
BAA_4536	*sigA*	BAA_4408		TATTTTGTTTATTTTAATGATGTTATATGAAAAGG
BAA_4536	*sigA*	BAA_0024		AAGATAAAAATTTATCGGATTATTTCTTAAATTTC
BAA_4536	*sigA*	BAA_0001	*dnaA*	CATTGCTATAGCTACTTTTTTTTGATATTATAGTT
BAA_4536	*sigA*	BAA_3957		GAAGGATTTTGTTTAGAAGCTATGGTATAATATAA
BAA_4536	*sigA*	BAA_3941	*recA*	GTTGGCAAATTGAATTGAAAATAGGTATAATAAGA
BAA_4536	*sigA*	BAA_4376	*argC*	AATTGACTCTATAAATATACAATATTATAATCATA
BAA_4536	*sigA*	BAA_4334		ATTGGATTGCCACTTATTTTTATTTTATTATTTGC
BAA_4536	*sigA*	BAA_4330	*deoB*	TATTAAAAAGAAAGCGTAAACATGTTATGATATCA
BAA_4536	*sigA*	BAA_3894	*adaA*	AAAGTACGATAGTGTTTTATCTTTTTGTTAGTATA
BAA_4536	*sigA*	BAA_3883		GTTTTTCAAAAGCGTAAAAACAAGCTAGTATGTAA
BAA_4536	*sigA*	BAA_3877	*gabP*	AATTGTTAAATTCAGAATATTTTGATATATTTTAA
BAA_4536	*sigA*	BAA_4292	*ptsG*	CATTGAATCGCTTACAATAGTTATGTATAATAACT
BAA_4536	*sigA*	BAA_4278	*mtnW*	AATTGACAACATGAAAAATATCTGACAAAATTCAG
BAA_4536	*sigA*	BAA_4276		TCTTTACAAAATACTGGAAAGATTATATTATTTGT
BAA_4536	*sigA*	BAA_4275		TGTTTATTATATTAGAAAGGTCATAAAACATTTCT
BAA_4536	*sigA*	BAA_4273	*mtnK*	TTTTTAAGATTTTTTAATATGTTGAGAATAGTTCT
BAA_4536	*sigA*	BAA_4245	*kinB3*	GTTTACATTCATTTCGAAAATCTGTAACAATTATT
BAA_4536	*sigA*	BAA_4227		TAAAGATTTCTATTAAATAAAATGATAAAATGAAT
BAA_4536	*sigA*	BAA_4217		TCGATAAAATTTTAATAACTTATTTTTTCATTCCG
BAA_4536	*sigA*	BAA_4207	*pdhA*	AATGTATAATAAGATTCTCTTTTTTGTTCAATTTT
BAA_4536	*sigA*	BAA_3752	*ccdA2*	ATTTTACACATAAAAGTAATCATGTTACAGTGTAA
BAA_4536	*sigA*	BAA_3732		CATGAATTATATTGTTAGTATGAGATATAGTTTAT
BAA_4536	*sigA*	BAA_3705	*acnA*	TATTTACTTATTTAGAAAGTTGTAATAAGATATTA
BAA_4536	*sigA*	BAA_4181	*ctaA*	CTTTGCTATTTTATTATTTTTTATATATCATAGGA
BAA_4536	*sigA*	BAA_3688	*htrA*	TTTTGTCTACAATGTTAAATAAGCGTATTTTTTTA
BAA_4536	*sigA*	BAA_3638	*dhaS*	ATTTGAATTTTCAGTATTTGTTAATTATGATTTAA
BAA_4536	*sigA*	BAA_4072	*ftsA*	TTGTTCCATATATTGAGTTGTATGGTATAAATAAA
BAA_4536	*sigA*	BAA_4070	*spoIIGA*	TTTATATGTAAGTAAAAAAATATAGCATTCTATGA
BAA_4536	*sigA*	BAA_4060	*ileS*	GCTTGACGTTTTTCTCATTATTACATATAATTTTA
BAA_4536	*sigA*	BAA_4055	*pyrR*	TTTATAAAATGCTTTTATCATAACTGTTTAGTATT
BAA_4536	*sigA*	BAA_3590		TATTTGCAAAATTTAATAGGTTTGGTATGATTAAT
BAA_4536	*sigA*	BAA_3587		AAGGTAACATACTTTCAAGACTATAGTATATTTTA
BAA_4536	*sigA*	BAA_3549		GATTTAAGTGTAGTTATATTTTTCTTAAAATAGGA
BAA_4536	*sigA*	BAA_3506		TTTAGTAAATTAATAGTTAATCTGGTAAAATAAAA
BAA_4536	*sigA*	BAA_2976		AGTTGACTTATAGGAAATATTTGTATAATATGTAG
BAA_4536	*sigA*	BAA_2955		CATTTTAATTTTCATAAAAAAGTGATAAAATGTAA
BAA_4646		BAA_4623		AAATAATAATAGCTAAACAAGTGATTAAAAAGCAAGAT
BAA_4646		BAA_4487		TTCCGGAATTCCCTAGTTGACTTATAGGTTTTATTGAT
BAA_4646		BAA_2976		TTTTAATAAAACCTAGTTGACTTATAGGAAATATTTGT
BAA_4646		BAA_3155		TTAGTCAAAAACATTGTATTTTTATCGGAATAGTGAAA
BAA_4681		BAA_4680		TGTTGCCTCCTACTTTACACTTTTCAACTGTCTTGACACCTATATTTACATAGTTTTAAAATAATGACAAGAGTTTTTTTATAATTCTTATTTTCACTATCATTTAAA
BAA_4681		BAA_4679	*nadB*	AAATTTACTATCACTTTTATTCTTAATATTTTTTTTGAGAACAGTAATAAAATTTTGATACATTTATATCCACAGTTCTGTCAACTTTTCACATTTCATCCTCCGTTGT
BAA_4777		BAA_0981		TTATATGTCTTCTGTTTGACTTTTAAAATCTTCTCGATA
BAA_4777		BAA_5615		ACAACTGACTTACGAGTGAGTAATATATTACTTCTATCC
BAA_4777		BAA_1179		AATTTTCTTTTAATATAATTAATTTAGAATTTTTAAATA
BAA_4777		BAA_5281		CGACTTATGTCGAAAAATTGAATGAGCATTCATTCAAGA
BAA_4777		BAA_4227		TTAAATAAAATGATAAAATGAATATGTATTCATTTTTTG
BAA_4777		BAA_3506		TTCATTTACCGACAAAAAATAAACAAAATATAAGTGATT
BAA_4844	*phoP*	BAA_4844	*phoP*	AAATTTTTATATTATATGTTTAA
BAA_4844	*phoP*	BAA_4367	*cpdB*	ACAAATACTTTTCAAATGTTATT

BAA_4844	*phoP*	BAA_3590		ATACTTACCTTTAATTTCTTCAA
BAA_4844	*phoP*	BAA_5746	*yycF*	ATAAACACATTAAAATGTTTTAA
BAA_4844	*phoP*	BAA_0798		ACATTGTTAAGAGTAAATTTACA
BAA_4844	*phoP*	BAA_4593		AAATGTTAAATAAGAGATTTACG
BAA_4844	*phoP*	BAA_3054		TATTGTTTAACAATAAATTTTTA
BAA_4844	*phoP*	BAA_1565	*resD*	ACGTTAGTAATACAAATTGAAAT
BAA_4844	*phoP*	BAA_1379		AAACTTTTATTTTGTTTGTTAAT
BAA_5236		BAA_5236		GACTCTAACGTTGCGTCATA
BAA_5401	*sigL*	BAA_0957	*gapN*	AAATCGTTTTCACAAATATTTCATTTGAAAAATGTT
BAA_5401	*sigL*	BAA_1236	*rocD*	AGAGTATTTACAAACTTCTTGTAATAACAAAAGGGG
BAA_5401	*sigL*	BAA_0383	*gabD*	AAGGAAATAATAGAGAAATTGAAAATGTAATTGAGC
BAA_5401	*sigL*	BAA_0381	*gabT*	TATGTATGAGCACCGTTCCTATATGAAAATGACGGC
BAA_5401	*sigL*	BAA_0363		CAGGATCCTTTTTAATTTTTGCATTTTTATTAATTG
BAA_5401	*sigL*	BAA_4407	*yqiS*	AAAAAGTTGGCACGGTATTTGCTTAATAAAAAGACG
BAA_5401	*sigL*	BAA_3638	*dhaS*	ATTTTAATTATTAACGTTCAGGTTCTGGTTAATTGC
BAA_5401	*sigL*	BAA_2839	*acoA*	TAGAATTTGGCACAGTACTTGCAATATAAAAGATGA
BAA_5401	*sigL*	BAA_1581	*gudB*	TTTATAAATTTATAAATTCATAAAATGAAAAGATTT
BAA_5436		BAA_3269	*cypD*	ACTATTTGTAATATTATATAATAAAA
BAA_5436		BAA_5436		AGAGATAGTAGTCTTTATGAATACTA

Table 1: Binding sequence predictions for *Bacillus anthracis* Sterne. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BAS0001	*dnaA*	BAS0001	*dnaA*	TAGAATAGGTGTTAAAC
BAS0045		BAS0711		CAATTTTTTGCTTATAATATATATTGACAACACTTTTAATAAGCATTTGTC
BAS0045		BAS0660		GAACATCTCACTTGCTACAAACTTACGCGAATGATTTATAAATTGATAAAG
BAS0045		BAS5320		ATTAAGAATGCTTGCTTTTTCAAAAAAATATTTTTTTTACAAGCGAAAAACC
BAS0045		BAS5309		CCTTTGGACGAACATTTTTAAACAAACTGAATAAAATATTCGTATTTTAGG
BAS0045		BAS5165	*glyA*	TGAAAACATGAAGGTTCGAAAGAATTACAACCTGAATTTTCGTTTTTTCTG
BAS0045		BAS0275		TGATTTTGTGCTTGCAATAACTAATCATTATTTTTTTGTAAGCTAATAAAC
BAS0045		BAS0255		TCAAATTATGCTTGCTTTAATATTTTCAATTATTTTTACAAACATAAATTT
BAS0045		BAS3993		GCTAAAAATGCATAAATATTTATTAAGGTAAATATTCATTTGCAATAGTAA
BAS0045		BAS3742		CATAAGAAAACTGAGATACTGACAACAGTTAAATAATGATCCTTTAACACA
BAS0045		BAS1475		ACAAAACTCGAACGAAAATGATGTTTTTATAAAATTTGTACGTGTTTTATG
BAS0093		BAS1667		TAATGAAAATTTTAAAAGAAGAACTAATTCTTTCATT
BAS0093		BAS1637	*fumC*	GAGGAATTTTTTTAAATAAATATAGAATATCTATATA
BAS0093		BAS0576	*aspA*	AAAGGAAATTATGGTGCCTCGTACGGAAGTTATAGGT
BAS0093		BAS5039		AGAAGGAGTTTTTAAGCCAGTGTCGAAATTATAGTAC
BAS0093		BAS3758		AAAAGGGAAACGTATAAAGACGTTGAATATTTTTCTT
BAS0093		BAS4195	*dnaG*	TAAAGGAATTTTTTAATTTATATCGAATACAAAATAC
BAS0093		BAS4076		AAAGGGAATTTTCACACAATTGTCGAACAATTCATGT
BAS0162		BAS0162		ATACTTGTATACAAGTATACTT
BAS0205		BAS0498		ATATCGATATTACAT
BAS0205		BAS3297		ATATAAAATAAAGAT
BAS0205		BAS0205		GATTCAAAATGATAT
BAS0249		BAS4762	*ldh*	GTATTAAACACTTCATTAAGTGTTATTATGCAC
BAS0249		BAS4690		AATAAGATATTTTCAGATAATCGACATAATTCT
BAS0249		BAS1803		ACAAATTTATCTTTCTATATTTCTAAGAATTGA
BAS0249		BAS1784	*ldh*	ACGGATACATTTTCAAATACTTGAATATCTTTT
BAS0249		BAS1308		AGCAAATAATCTTCTTAAGGCTTTCAATCAATA
BAS0249		BAS0823		ACTATAATGATAAATAATACTTCACCACATTGA
BAS0249		BAS4869	*ldh*	ACAAAAACTTTGCTCAATATTTCACAAAGTATC
BAS0311		BAS0312		ATTGCAAATGTTTGTTGGT
BAS0311		BAS0310		ACCTTTTCTTACAACCGTA
BAS0311		BAS0295		TCGTTTTTGCAAAACCTTA
BAS0311		BAS3348		GTGTTTACTTTAAACTTAA
BAS0311		BAS1401		AATACACAATTACTTTGTG
BAS0311		BAS0808		GAGTAGTGTAAAAAGGTGG
BAS0311		BAS1071	*rocD*	AATGATATAATATTTAGAG
BAS0311		BAS4072		TATGCGTAAATATTTTGAT
BAS0311		BAS2645		ACATTTTTATAAAATTTAA
BAS0311		BAS2574		AAAACAAAATTATATTTTA
BAS0311		BAS0464		TCGTTTTTTTAAAACGCTA
BAS0505		BAS2208		AGTTTGTTAACAGAGTTACAA
BAS0505		BAS0803		CCCTTATCATTATGATTATCG
BAS0505		BAS1109	*spxA*	TCTATTGTATAATGATTATAG
BAS0505		BAS0505		AATATTCTTAATATTTTATTA
BAS0505		BAS0397		TATCTTATTAATGTTGTATTA
BAS0505		BAS0330		AATCTTAATAATATCAATTAA
BAS0505		BAS4363	*hemA*	AATATTAGTAATATTTCATCC
BAS0505		BAS4001		ACTTTATGAGAATAATTATCA
BAS0505		BAS2940		AGTATAAAAAAGATTATTTAC

BAS0505		BAS2816		ACAATTTTAGGATGATTATAT
BAS0546		BAS0547		GCGTAGTAGATTATTTAATTAAACCATTTACATTTGAACGATTTAAAGAGG
BAS0597		BAS0598		CAACTGATTGAATATATATGCTCAATTATACA
BAS0870		BAS0530		TTTGGAAAAAGACTTAAAAATAAAATAAAACTTTCGCATTAGTAAAAATTC
BAS0928		BAS4484		GGTTTCAAGTCTATAACTTGGAGGAAAAGAATGAACAA
BAS0928		BAS0084		CGTTTTGATTAGCGAAACAATGGTTAATAATGAGTAGG
BAS0928		BAS4302		AAATAATAACCAACGGGTCAAAGAATGCGTAATCTTCA
BAS0928		BAS4105		TGTTTAAATACTGCGAAATTTGTCTATAAAAAGAATAA
BAS0928		BAS3435		AAAAGACTTATAAAAATGAAATTAAGGAAAACTTTTGT
BAS0928		BAS3434		TTTTTCATAATGAATTAAAAAAGGAGTGGAGAAAATGG
BAS0928		BAS2940		ACTGTACCATTTGAATATAGTGGGAAATTAACACATGA
BAS0928		BAS3348		TAAGAGAATGTAATGTATAAAAAATGTTAACATTGTGT
BAS0928		BAS2816		GCTATAAGTAAAAAGGTATTAAAGAAGATAAAGTTGTC
BAS0928		BAS2331		AAAAGAAGTAATATGTGTGAACTAAGATTTAACTTTTG
BAS0928		BAS1871		AAAGGTTTATGTAGGGGTTTATTTATAAACAATTTAAG
BAS0928		BAS1861		GGATTAATAGATTTATTGAAAGAATAACAATTACTAAA
BAS0928		BAS1855	*nadE*	ATTTTAAGGATTAAGTGCGAAAGAGCAAAAATTTTTGT
BAS0928		BAS2217		GTTTTTATTTTTTCTAAAAAAGGAGAATAAAGATGGAG
BAS0928		BAS1308		TTATTTCAACCTATTGTTGAAAAATATTGTATTCTTTC
BAS0928		BAS0823		TTTTTTGTGTATTTTTAGAAGAGGTATAAAAATATTAT
BAS0928		BAS0808		AAAAGTGTTTATAAAGTAAACTTTTTACAATAGTTTGG
BAS0928		BAS0803		ATGTTTACGTCCTAATAAAAAGGGTATCTATTAAATTA
BAS0928		BAS0712		GTTTTTATTATGCACATTATAAGAAAAAGGAGTTGAAA
BAS0928		BAS1154		TAAAATAAAAAGAGAGGAAAGTTATAATATAACTGTTT
BAS0928		BAS0605		ATATTCTTTATAAAAAGGAGGGGGAATATTGGAAGCTT
BAS0928		BAS4925		ATCATACATGTAAACGATTTTTCTCCTCAGTACTTTGG
BAS0928		BAS4913		AAAAATAAACTTAAAAGACAAAAAGAAAGAAAATAAGT
BAS0928		BAS5319		TTTGTGTAATTTTACAAAATTGAGTAGTTATTTTAAAA
BAS0928		BAS5287		TCATTACAATTATTAGCTTTTGGATATTATTAAAATTC
BAS0928		BAS4713		TCTTTCCATTACAAAAAAACAGGGAGAGATGAAAAATG
BAS0928		BAS0312		CGTTTCGCCCTATAGGCGGAAGAGAATTAAAAAAGGTG
BAS0928		BAS0295		AAAAATAATATAAAATTTTTAATTAATTAAAAATTTTA
BAS0928		BAS5039		AATATAAGCAATAAGTGCTACAATTATCCTATTTTATT
BAS0928		BAS5000	*clpP*	GATGTTGAAGATACATAAGCAGGGAATTTTAAAAACAT
BAS0928		BAS4507		CAAAAAAAGAAAATCGGTAACGTAATTTAAAAGATTAG
BAS0958		BAS0959		GACGGAGAAAAGTAGAGATC
BAS0958		BAS0628		CACGGAGAATCGGAGTACAC
BAS0976		BAS0580		ATAAAGGTATATTAAGATAA
BAS0976		BAS0655		ATTCATTTTAAAGAAAAAAG
BAS0976		BAS0567		GATATAAATATTTTGAAGAA
BAS0976		BAS4907		ATAATGTTTAAACCTAAAAT
BAS1348		BAS5265		TTATATCAATGTTATGAATAAGTACAA
BAS1348		BAS2108		TCGTTTTCAACATTGGAATAATAATAA
BAS1386		BAS1997		AAGTGTTTTTTATATGT
BAS1386		BAS1984		GTAAGTTTTTGAAAAGT
BAS1386		BAS1803		CTATATTTTTTAATACT
BAS1386		BAS1357		TTAGATATTATTGTAAA
BAS1386		BAS4690		GAAGGTGTTTAAAGAGT
BAS1669		BAS0471		TTACAATTTATGCAGAAGAACAGGATTTGCTTCGTTATTAGGAC
BAS1669		BAS2932		TAAGTGTGTTGGCTATGATTGTATTTGGTTCTGTTGCAAAGATT
BAS1753		BAS1754		AATGAACAAAATTTCAAAAGTGTATTTACATTTGTTCAA
BAS1984		BAS1977		AGTGTAGACATAAAATGTTGG
BAS3231		BAS3231		GGAGAAAAGCGGCAGGAAATAGTGAGAACAGCAGATTTTCTTATACATAATGAG
BAS3479		BAS0291		ATGGCTATAAAATAAATCTTAT
BAS3479		BAS4498	*dnaE*	TCTTTTTTCATACAAGAGAATG
BAS3479		BAS4416	*uvrC*	TTTTCTTGCATACAACCCACAA
BAS3479		BAS0005	*gyrB*	TTAATAGTAAAAATGTTCAAGT
BAS3479		BAS4316	*ruvA*	ACCTCTTGTAAACAACCCCATC

BAS3479		BAS4252		ATCGGTTGTAAGCAAAAATATA
BAS3479		BAS3479		GAAGTTTGCACTCAAGTGAAAA
BAS3479		BAS3392		TATCTTTGTATACAAGACTAAA
BAS3479		BAS3172		TTTGCTTGCTAACTAATAATAT
BAS3479		BAS3053		ATAAAAGAACATAAGTTCATTT
BAS3479		BAS1913		TTTAAACAACAATCATACGCTT
BAS3479		BAS2134		TAAGATTGTAAACCAGATTTAG
BAS3479		BAS0943		AAAATAGAGTGAAAGTTTTATG
BAS3479		BAS0575		AACGTTAATAAACAAGCTTAAT
BAS3679		BAS2208		AAATTTTAAATGAACAAGGAAGAGA
BAS3679		BAS1308		ACAACTTTTTATAACATAAGAAAGC
BAS3679		BAS0823		CAAGGTACTGTAAACATTTTGGTAC
BAS3705		BAS1691		AATCTATATTGACATTTTTATAACCTGATATTAGTATCAGGTTATAAAA
BAS3705		BAS1095		TCTAGACAAAAATACATTTATGTTTTAAAATTAGTACCAAGTCATAATA
BAS3705		BAS0207		TACACTTTTGTAAGAATTGAAATGTAACAATTAATACTTAAATTTATTA
BAS3742		BAS3742		ATAATGATCCTTTAACACAGCCCCGTGAGGTTGAGAAGGTAACGGTTTGAAATA
BAS3755		BAS1419		AAAATTATCAGTATACTTTAACCTGTTTTAATATACTATA
BAS3755		BAS0827		TCGTTCTATTTGCTACATTTATTTCATAAATTATGGATAG
BAS3755		BAS0731		TTCTCATGAAAGCTTGTTCTCAATCATACACTTTCTTAAG
BAS3755		BAS1074		GTAATGGCTGACATACTCGCGCCCCGACCAATATATTTTA
BAS3755		BAS0519		AATAAAAATGTTATACGTATACTATCTCATTGGTTGGTAA
BAS3755		BAS0518		AAAGAATGTTTGAGGACATCCTAGCATACCTTTTATAATA
BAS3755		BAS5319		ATTATAGGAAAAATAAACACATTAAAATGTTTTAACTCAT
BAS3755		BAS4875		TAGCTAGGTTCAATAAGCACTTGAACCAAATCTACTTCTT
BAS3755		BAS4761		TTAAACATTTTTAAGAAAAAAACGTATATTTTTTATAAAT
BAS3755		BAS5136		TTTAAGATATGAATACGTGCATGGTCATTTTTATAAATCT
BAS3755		BAS0237		TCTGCATATTTTATATCTTTGTCTCATATATTTGTGGTAG
BAS3755		BAS0150		TTGTTCTCTTTTTTATATTTACAGCATATATGTAACTGAA
BAS3755		BAS0146		CAAGCATAAACTTTCCTCTTGTCCCATATAAGTAATTAGA
BAS3755		BAS4490		TTTCACCATCCAAATGTAAGCATGTATACATTTTAAAAAA
BAS3755		BAS4484		GAGGTATAATAAATATAGATATATAAATTTTTATATTATA
BAS3755		BAS4378		GTGGCAATTTTGAAAGTGAAATGGCATATGAAGTTGTAAG
BAS3755		BAS4308		CTTGTCATTCTTGTCTCGTACGCGCATATAAATTATTGTA
BAS3755		BAS3858		GCGCAAAATAAGATATGTATTACCTTACTTTTATAAAGCG
BAS3755		BAS3840		TGGTCTAAATTCACTACGTATCCACGTATACTGAAATGAA
BAS3755		BAS4244		TATACAAATTATAACTGATATTTACATAAGTTAACAAAAA
BAS3755		BAS4097		TACTTCTGAAATAGACAACCTCTGCATATGGTTGTACAAA
BAS3755		BAS3555		TAGTCTTTTGTACAAAGTGTAGCCCATATATTACTGGAAG
BAS3755		BAS3402		TAATCATAATACAGCGGAATCAAAAATAAAATAAAAACAG
BAS3755		BAS2905		TAGTTGTACAAACTTTAACCACTGAATAGATTTAATATAG
BAS3755		BAS2886		TAATCATATTACTTAAAAATTATGCGTAAGAATATAGTAA
BAS3755		BAS2789		ATAAATTATAGAATAATTAAATGGTACTTTAAATAACAAA
BAS3755		BAS2467		GTTAAACTCACAATACGTTTTTTCTCTTACTTATAAGAAT
BAS3755		BAS1921		ACGGTCTATTCTCTAGAGCTTGTACATAACTTGAAATAGA
BAS3755		BAS2369		TAGGAATTTTTTTGTGCATTAGTACAGAATCTGTTTTAAA
BAS3755		BAS1870		TACGTCTAATTGTCCAAGCTCATACATATGCATAATAGTA
BAS3755		BAS2191		TTATCATTTCATTTACGTGGAAATGAACGCTTATACTGAC
BAS3755		BAS1625		AAAAATAATAAAAAACATACATTACCTTATGAAAGGGAAT
BAS3755		BAS2095		AGTGCCTAGTTTTCATGTTTTCATAAAAAATTGTTAAAAA
BAS3755		BAS2065		TGGAAGTAATTTAGATGGTACGAGAATAAAAAAAGAAGAA
BAS3892		BAS3408		ATTATTTACTTATTT
BAS3892		BAS2188		TTTATTTATTTAAAA
BAS3892		BAS4488		TGTATTTTGTTAAAA
BAS3983		BAS0350		CACACATGAATTTCATATAATATGGAAATATTAAAA
BAS3983		BAS5176		ACACCTTACTCTATTTTTTTGTGGAAAAATTCATAT
BAS3983		BAS5131		AATGTCTAAAACAGCTTTTCATTAATCAATATGCAA
BAS3983		BAS4683		AACATTAACTACTTTTGTTATCTTTAATTTTCCTAC
BAS3983		BAS4630		AATGAATAATATACAAAATTAGCCACAAAATAGCAC

BAS3983		BAS0200		GAGGATAAAAACGATACATAAGTCGAATAATTATTT
BAS3983		BAS3986		TGGTATTAAAAAATAATAAATTGGAAAAAATAGGTT
BAS3983		BAS3983		CTTACCCTAAATGATAGTACCTTTTAAAATACCTAC
BAS3983		BAS3981		TATGTTAATTGATGGTACAATCCAAAATATTAAAAA
BAS3983		BAS4220		AAGAATGACCTTATATAACTTTGGGAAATCTAACGA
BAS3983		BAS3754		CACAGTCATACATTTCGCGTTTGGAAAAAAGAATCA
BAS3983		BAS4186		TTTAATACGATAGTAATATAGATAACATATAACTTC
BAS3983		BAS4109		ATTAATAAAACGTAAAAAGAATAAAAGAAATACTTT
BAS3983		BAS4105		CATGTTTAAATACTGCGAAATTTGTCTATAAAAAGA
BAS3983		BAS2940		AGTATAAAAAAGATTATTTACCTTTATGGATTTCCA
BAS3983		BAS2928		AGTCAAGAGATAAACAAGTGAAAAAAATATTATTCA
BAS3983		BAS2910		AATTATTATTTCATAATTTGTAGCATATTAAAAACA
BAS3983		BAS3369		ACACTATAATTATAGCGTTAAAGTTAAAAATATACA
BAS3983		BAS2816		CATGTATAAATTCATTTTTAAGATTGAAATTTTAAT
BAS3983		BAS2562		GAAAATAGGAAAAACATATAACAAATGTATAAAAAA
BAS3983		BAS2185		AAATATTAGACACTTAATCCTTTTATAACTTACGAG
BAS3983		BAS2085	*acpD*	AAACAAAATAAATCATTAACTTACGAAAAAAAAGTA
BAS3983		BAS1458		ACATATTATTACATTTTGGAATCGTCAAAATACTGG
BAS3983		BAS0998		TATACATAAAGATACTTCTTTTTCAAATAATAAATG
BAS3983		BAS1363		AGTAATTGTAAAGTCTTTTAAATAGAAAAAAACTAA
BAS3983		BAS0803		AATACTAAAAATGTATGTTTTTGCTATTATGTACAA
BAS3983		BAS0727		TAAGTATAAATTCCCGTCTTTCCCAAAAACTAACAC
BAS3983		BAS0675		AAAGGTAACTTATTTGAAGTTATGAAAAAAATTAGA
BAS3983		BAS0602		AAGCATAAATTTCTCACAAAAAGCAAAAATTAACAG
BAS4001		BAS2275		ACATATAGATGAAAAACCGTATTCTTTTCATTAAGAA
BAS4001		BAS2204		TTTTTACATCGATAATGATAATCATTATCATTATTTG
BAS4001		BAS0809		TGATTGTATGGGGTTTGATAACATTTTTCAAATGAAA
BAS4001		BAS0582		TATGCCAACTTTTATTAAGAGTAAAGGCAACTGTTAC
BAS4001		BAS4797		TATGCTATCATTTATTTGAAAGTATTATCATTTGCAG
BAS4001		BAS0337		TTGTTTTACTATTACTAATAGTGATAACTAAGAAATT
BAS4001		BAS5030		TTAATAAACTATTACCAATAGTAAATCCCTCAGAAAA
BAS4001		BAS3583		ATATATACTTAGTAACGATAATCATTATCAATGAAAA
BAS4001		BAS3579		GCCAACTACATATCTTGATATGTCTCATCAATTAGAT
BAS4001		BAS3467		TTAACGAACAAATGTTAAAAATGATGATGATGCAAAT
BAS4072		BAS4071		TTATACTCACACGTTTTTTTTCGTACGTTATTTA
BAS4072		BAS2588		TCAAATTAGGTGAACAAAAGTGGACAAAACGAGACAGGTGTCTCATTTTGTCC ACTTTTTTATT
BAS4076		BAS0061		TTTTGACAAAAATC
BAS4076		BAS3756		TCTAATACAGATCT
BAS4076		BAS2208		TGTTAACAGAGTTA
BAS4079		BAS0808		TTGAATAAAAATATCT
BAS4079		BAS1071	*rocD*	AAGAATAAAATTAAAC
BAS4079		BAS0312		TTGGATAAAAAAGCGA
BAS4079		BAS0310		ATGAATACAAAAAAAT
BAS4079		BAS0295		AACGTAAAAATAATTA
BAS4079		BAS3348		TCCTAAAAAATATTAA
BAS4079		BAS4040	*argC*	AATTAAAAAGTAAGTA
BAS4108		BAS1743		AATTTGCACTAAGGAAACA
BAS4194		BAS3892		TCGATAAAATTTTAATAACTTATTTTTTCATTCCG
BAS4194		BAS3883		AATGTATAATAAGATTCTCTTTTTTGTTCAATTTT
BAS4194		BAS3858		CTTTGCTATTTTATTATTTTTTTATATATCATAGGA
BAS4194		BAS4293		AATTTCTTATAATACTATATTATGCTATAATTTCA
BAS4194		BAS4271		GATATCACATTGTTTCTTTTCTTCCTTTATGGTAC
BAS4194		BAS4244		TGTTTAATATTGACTATAAATGTATTCAATTGTTT
BAS4194		BAS4218		CATTGAATCTTTGCTGCTCTATTGATATAATCAGT
BAS4194		BAS3758		TTGTTCCATATATTGAGTTGTATGGTATAAATAAA
BAS4194		BAS3756		TTTATATGTAAGTAAAAAAATATAGCATTCTATGA
BAS4194		BAS3746	*ileS*	GCTTGACGTTTTTCTCATTATTACATATAATTTTA
BAS4194		BAS3742		TTTATAAAATGCTTTTATCATAACTGTTTAGTATT

BAS4194		BAS4159		TAGGTAAGAATGTTAGAAATAGGAATTAATGTAAG
BAS4194		BAS4158		GAAATAATAAAATAAATACATCGATTTTCCTTTAC
BAS4194		BAS4148		ATTTACAATAGTAAGAAAAATAAACTATAATGAAG
BAS4194		BAS4145		AAAGTCGTATACTTGTAGAAGATTAAATCTATTTA
BAS4194		BAS3652		CCATCATTCTTACTGTAGAATATGATAAAATTAAC
BAS4194		BAS3645		GAAGGATTTTGTTTAGAAGCTATGGTATAATATAA
BAS4194		BAS3629	*recA*	TATTGCCTAAGGGTGTAACTATTCATATAGTTGCT
BAS4194		BAS4082		AGAATAAAATTTCTCCCATACCTTAATAGATTCGA
BAS4194		BAS4076		GCTCCGCAAGAGGTGTTTTTTGTTGTAAAATAAAA
BAS4194		BAS4072		TATTTTGTTTATTTTAATGATGTTATATGAAAAGG
BAS4194		BAS4040	*argC*	AATTGACTCTATAAATATACAATATTATAATCATA
BAS4194		BAS4001		ATTGGATTGCCACTTATTTTTATTTTATTATTTGC
BAS4194		BAS3585		AAAGTACGATAGTGTTTTATCTTTTTGTTAGTATA
BAS4194		BAS3583		TCTTGAAATAATTGCATATTAAATATATACTTAGT
BAS4194		BAS3570		AATTGTTAAATTCAGAATATTTTGATATATTTTAA
BAS4194		BAS3455		ATTTTACACATAAAAGTAATCATGTTACAGTGTAA
BAS4194		BAS3435		CATGAATTATATTGTTAGTATGAGATATAGTTTAT
BAS4194		BAS3408		TATTTACTTATTTAGAAAGTTGTAATAAGATATTA
BAS4194		BAS2993		AATATTATATAATAAAAAAGACGTTCCATATTTAA
BAS4194		BAS2932		CTAATATCATTTCTTTTTTAGTAAGATATATTTCA
BAS4194		BAS2916		AGATTAAAATAATTTAATTAATATTGTATCTTTTT
BAS4194		BAS3395		TTTTGTCTACAATGTTAAATAAGCGTATTTTTTTA
BAS4194		BAS3380		TATTTTCAAAAATAATATAATAGTGTAACATAAAG
BAS4194		BAS3348		ATTTGAATTTTCAGTATTTGTTAATTATGATTTAA
BAS4194		BAS3300		TATTTGCAAAATTTAATAGGTTTGGTATGATTAAT
BAS4194		BAS2862		AATTGCCATATATTAAAAATCTTAGCAATATATAA
BAS4194		BAS3297		AAGGTAACATACTTTCAAGACTATAGTATATTTTA
BAS4194		BAS3262		GATTTAAGTGTAGTTATATTTTTCTTAAAATAGGA
BAS4194		BAS3220		TTTAGTAAATTAATAGTTAATCTGGTAAAATAAAA
BAS4194		BAS2789		ATAATTAAATGGTACTTTAAATAACAAATTGTTAT
BAS4194		BAS2788		AATATCATATATTATTATAAATAATGTAGATGAAA
BAS4194		BAS2786		ATAGTTAAATTGTTTTTAAGCTTTACTTTAGTTAG
BAS4194		BAS2714		AGTTGACTTATAGGAAATATTTGTATAATATGTAG
BAS4194		BAS2700		CATTTTAATTTTCATAAAAAAGTGATAAAATGTAA
BAS4194		BAS2645		ATTTTAAATTTATCGTAAATAAGTGTATTATTTTT
BAS4194		BAS3048		TATTTTTTTCTCATAAAAAGTCTGATAAATTAGCA
BAS4194		BAS2597		TTAGTATCATTATTTATTACGTTCCGTATATTTGG
BAS4194		BAS2574		TATTGAAAACGCTTTTATTCAGTTTTATAATAAAG
BAS4194		BAS1997		GGCTTAAAATTGTTAAACACTTAGTAAAGTTTTCA
BAS4194		BAS1984		AAAGTAAAAAAATAAACGGAAATAATATTTTTTCT
BAS4194		BAS1977		AGATTAATATATTCTTTAAAAAACGTTTGAGTAAT
BAS4194		BAS1938		TATTTATAAAATTTTCTAAATGTTATATCATCATT
BAS4194		BAS1855	*nadE*	TATTGTATATTCGTTAATTTTGCGAAATAATAAAG
BAS4194		BAS1803		TATGTAAAATAATTACTCTTTTTAGATTAAGTTCT
BAS4194		BAS2268		ATTTTCCAAATGATTAAATTCCTTATAAAATGAAA
BAS4194		BAS2222		AAAATAATACTTTACCTCTCTACTTTTTACTTTCT
BAS4194		BAS2208		CATTTAAAAAGACAAATCTATTAGAAATTTTAAAT
BAS4194		BAS2204		TATATAATATAGGCTTTATGTATAATTTAGAATTT
BAS4194		BAS1784	*ldh*	TTTTCAAATACTTGAATATCTTTTATAAAATGTAA
BAS4194		BAS1754		TATTTACATTTGTTCAACTAAGGGTTAGAATGAAG
BAS4194		BAS2188		ATAGTAATATTTTATTTAACTTTAGAAGATTTTTG
BAS4194		BAS2185		TATTAAAATTTCACTTTTTACATTCTATGCTTAAA
BAS4194		BAS2125		ATTTGAAATGAATTCCATATTCCATCAAAATAAAG
BAS4194		BAS2108		GTAGAATTTAGAAATTTCTTAATGTTATAATGATA
BAS4194		BAS1691		CATTATCAATTTTAGAAACTTCTCGTATATTATTT
BAS4194		BAS1676		TTTTGCTATATTTTTCACAAACTTTTAAGATAAAA
BAS4194		BAS1667		CATTTATAAAAATGTAATTACTTTTAAAATTTTCT
BAS4194		BAS1666		AAACTAATATGCTTTGTTAGACTAATGAATAGTTC

BAS4194		BAS1637	*fumC*	TCTTATCATTTAAGAGAAAGTCTGTTATTATAAGA
BAS4194		BAS1633		TTTGAATGAGCCTTCTTACAATTGGTATAATGACA
BAS4194		BAS2027	*ileS*	TGTTTATTTTTATATAAAACTTAATCAACTGTTAA
BAS4194		BAS1528		AGTTGTGAATATTGGAAAAATATGTTAATTTCTCA
BAS4194		BAS1475		GCTTTTCAAAGTATAAAAAGTGCGTTATAATCCTT
BAS4194		BAS1458		AATTTACACTTTTTGATGAACTTTATTTATTAGAT
BAS4194		BAS1432		ATAGTGTCCTAGTTTTTATTTTTTATATAATTTAT
BAS4194		BAS1401		ACTTTTCTAAAAGCAATAAAAGGACTATAATGATA
BAS4194		BAS0998		AAGATAATATAGTTGAGAGAATTTTTTGAAGCTTC
BAS4194		BAS0959		TGTTGACACCGCTTTCATTAACGGTTAACATTATA
BAS4194		BAS0901		CTCATACGATAAAGTATAAGTAAATTTACTTTTGT
BAS4194		BAS1386		TTTTGGAAAAAAATTGAACAAACTTTATTATTCTT
BAS4194		BAS1363		TTTGTAAAATAGCTTTTCAAATGTTTTTGAATTGT
BAS4194		BAS1357		ATAATAACATTTATTTATATCTCCACTTTGCTTGT
BAS4194		BAS1330		AACATTTTATTTTATTAATTTTATATAGCCTTTTG
BAS4194		BAS1308		AATAAAGTTGGATAACAACTTTTTATAACATAAGA
BAS4194		BAS1300		ATTGTCAAAATAGGTATTTTTTCGGTACAATTAAT
BAS4194		BAS0838	*div*	TCGATAGTATGGTTTAAAGAATATGTAACTTATGA
BAS4194		BAS0832		GCTGATATATTTTGAAAATCGCCGATATAATTCAA
BAS4194		BAS0831		GTTTGTTTTAAATAGGATATTAAAATAAGGTAAAA
BAS4194		BAS0823		TGTTGGAAAAATAATGCATTTGTACTATAATGATA
BAS4194		BAS0808		ATATTAGAATGTTGAAATAACTCATAAGACAGTTA
BAS4194		BAS1213		ATTGATACAAAGAAAAAAGAAGTGATACAATCTTT
BAS4194		BAS0795		CAAATAAAAACTCCTTTTGACTTACTATCAGTAAG
BAS4194		BAS0751		TTCGGATTCACATTAAAGAGATAGATAAAATAAAA
BAS4194		BAS1177	*sucA*	TTTGGTATTATGTAAATTATTGTAATAACATAAGA
BAS4194		BAS1156		TGTGTATAAAAAATTAAATAGATGAAATCGTTTAA
BAS4194		BAS1154		TATGGATTTTTCAGAATAATCATGATAGATTATAA
BAS4194		BAS1111		ATTTCCATTCTGGAGAATCTTATCATAAAATGAGA
BAS4194		BAS1109	*spxA*	AAAATAACATACTCTATTTTCTATTTTTTCTTTCT
BAS4194		BAS0681		TGTTCCAATTTATTTAGTTGTACTAAAAAATATTA
BAS4194		BAS0659		TGTAGCATAACAGATAGAAATGTTTTATAATTTGA
BAS4194		BAS0631		GATTGACGTATTTCTAGAAATACGTTATATTATTT
BAS4194		BAS0628		CGTTTACATTTTTGTCACAAAGTTGTATAATTTTC
BAS4194		BAS1095		TCTAGACAAAAATACATTTATGTTTTAAAATTAGT
BAS4194		BAS1090		TATTGTATAAAAAAGTCAGTTGGCGTATAATCAAA
BAS4194		BAS1078		AGTTGATTTTTCTTTTCACTTCGAATATGATATGA
BAS4194		BAS1071	*rocD*	GTTTCGCTTTTTTACATTTAAATGATATAATATTT
BAS4194		BAS1022		AATATAATATTATATTCCAATTATTAGACACTTAT
BAS4194		BAS1019		ACTTGCTACTTTAATTTTCTTTTAATATAATTAAT
BAS4194		BAS4990		AGTTGAATAAGTTTCTATCTCCTGTTACAATAATA
BAS4194		BAS0598		TGTTGACTAACTTATATATACGAGTTAATATGTAA
BAS4194		BAS4936		TGTTGACAATTTGTTTTCAATTCAATATAATGGCT
BAS4194		BAS0576	*aspA*	TGTTAGAAAATATTCCTATACATGATTTACTATTA
BAS4194		BAS0561		CGTTTTCATAACTTTTAAGTTATGCTAGAATAAGG
BAS4194		BAS0551		TTTGTTTCTTTATCAGTAAGAAAGATAAAATAAAT
BAS4194		BAS0547		AAATTAATTTGGTAAATGTAAACTTGCTAAATTTC
BAS4194		BAS0530		TATTTACCATTCAAATAGATGCTTTTATGATAAAC
BAS4194		BAS0505		AGTACACTCTTTATAAGAATTATAAAATAATAATC
BAS4194		BAS5320		AGAATATTATTCTACACAAAGAAAAAAAAATATAT
BAS4194		BAS5319		ATTTTACAAAATTGAGTAGTTATTTTAAAATAGAA
BAS4194		BAS5314		ATTTAGCCCATTTGAGTTTAGATAATAACATGAAA
BAS4194		BAS5304		TATGTGCATTGAGATTATAAAATTTTATAATTAAT
BAS4194		BAS4877		TATTGCGAAAATTTAAACAGTATCATATATTTATT
BAS4194		BAS4869	*ldh*	TATGTATTATTTATAAAAAGGTCAACATTGTTTAT
BAS4194		BAS0498		TATTTCATAAATCCCGTATTAATAGTATAATTATT
BAS4194		BAS4836		TAGATAATACGTACTTTATAAAATATTTCTTTTAC
BAS4194		BAS0472		TATTTATTATTGTACTTTTTCACCTTACTATTTTG

BAS4194		BAS0471		GTTTTATCATTCCACTTTTTCATGTTATTATTTAT
BAS4194		BAS0466		TTTTGCACTCTTTAATAATTTATTTTACGATGAAG
BAS4194		BAS0464		TTTTGCGATTTTATAAGAATTTTAATAAAATTATA
BAS4194		BAS5265		GTGGTAATATAGTTACAATACTTATTCATGTTTCA
BAS4194		BAS5258		AATTGATATTAGGAAATTATTGCTATAAAATCAAC
BAS4194		BAS5238	*eutD*	AGTTTAAAATATAGAAGAAAAGCAGTATTCTTAAA
BAS4194		BAS5219		TTTGTACTTTTAACTATACATTTTGTATGTTTTGT
BAS4194		BAS5204		CTTTTTTATGTTTAAATGTTTTAGAAACACTATTT
BAS4194		BAS4762	*ldh*	TATATAATATTTTTAGCAACTATATAAATTTATTA
BAS4194		BAS4751		TGATTCACATCTTTTTCGTAGTAAATCACAGTTGC
BAS4194		BAS0397		GATATGATATCTTATTAATGTTGTATTATTGATAA
BAS4194		BAS4748		ATGTGGAACAACCGGAAAAGTTTGATACAATAGTA
BAS4194		BAS0385		CTTTTTATCAATAGGTAAATAATGCTACAATATAA
BAS4194		BAS0331		AAGATAAAATGTTTCTTAACACACTATTTATTTTA
BAS4194		BAS0330		ATTTTATTTATCACACAATTCTTTGTAAAATAGAA
BAS4194		BAS0327		AATTGACTTTTTTCTTAAAATTCAATATGTTCATA
BAS4194		BAS0312		AATATTTCATCGCCTCAAAAATTTTTATTCAGTTA
BAS4194		BAS0310		TATGTAAAACGTTATTTTTCTCTTTTGTTACATTT
BAS4194		BAS5194		CGACTAAAATAGCAAAAAATGTTACTTCTTGTTTT
BAS4194		BAS5187	*pyrG*	AATTTTCTTTTTTGTTTTTTGTTTAGAAAATAAAA
BAS4194		BAS5165	*glyA*	TCTTTTATGATATAGAAAGTCATGTTAGAATGTAG
BAS4194		BAS5115		AATTTATCTTTTATTTTTAATCAAGAATAATGTAA
BAS4194		BAS5114		AAAATATTATTCTCTCACTTATTCTTTAGGATTAT
BAS4194		BAS4690		AAAGTGATATTTGATTAACATCTATTACATTTTTT
BAS4194		BAS4683		GAAATAAAATGTATTAAAACATTAACTACTTTTGT
BAS4194		BAS4661		GTATACATTATTTATATAATTGTGTTATACTATTG
BAS4194		BAS0295		AAAATAATATAAAATTTTTAATTAATTAAAAATTT
BAS4194		BAS0275		ATTTCTACTACAAGTATATATTTTAAATAATGAGA
BAS4194		BAS0254	*guaA*	TTTTAGAAAAGGAAATGAATTTCTGTAGAATTTTG
BAS4194		BAS0205		ATTCAAAATGATATTAACTATCATTTATAATGATA
BAS4194		BAS5038	*secA*	AAAATAAAATTATTTTTCTCCTCGCATAAAGGATA
BAS4194		BAS5025		AATGTAATATTCTTTCAAAGATATTATTCAAATTA
BAS4194		BAS5000	*clpP*	GTTTGACCTTTATTGACCATAATTGTATTATAAGA
BAS4194		BAS4585		ATTGGAAATATAGATGAGATTATAAAACAATTATA
BAS4194		BAS4561		TAGATAATATTATTGTAATAAATTTTAAAAGTCAT
BAS4194		BAS4560		TACTGAAAATTTTAAATAATGTTATTATAATAGAT
BAS4194		BAS4556		TTTGACACGCCTATATAAATGATGGTATAAAGAAA
BAS4194		BAS4554	*rpsD*	GGTAAAAGATATAATAGCGTTTGTGTAAAATAAAA
BAS4194		BAS4553		AAAATAAAATGTGTTTGCGATAATATAGAAAATGG
BAS4194		BAS4547		AGGTCAAAAACAAGCCATTACATGGTATGATTAGT
BAS4194		BAS4546		TGTGTAATTTTGCAGGCGATGTTGCTATAATGAAT
BAS4194		BAS4543		GGTTCAAAATTGACAGAAAATTCTTTATTATATAA
BAS4194		BAS4535		AATATACTATACAGTATAAAAATTTATATATGTTC
BAS4194		BAS4521		TTTGTCAAATATTGAGATATTATGTTAAAATATAA
BAS4194		BAS0166		TATATGATATTATGTCTCATTGTATGTACCTTTCC
BAS4194		BAS0162		GCTTTCCTCTAATAGAACCAAAGGAAACGATAAAA
BAS4194		BAS0102	*rpoB*	CCTATAATATAGTATTTTAGTTTTTTTGCAACTGC
BAS4194		BAS4488		TGTTCACAAAAGTCAGATAATTGTTTATAATAGGA
BAS4194		BAS4484		TTTATATTATATGTTTAATAATCACATTCAGTTAC
BAS4194		BAS4477		TGTTGCAAAATGAAAATAAACAAAGTATACTAACA
BAS4194		BAS4472	*thrS*	CATATAATTACCTACATAAATGAAATATATTGAAA
BAS4194		BAS4471	*infC*	TATTGCAAGTATGTTAGTTGTTTGTTACAATATTT
BAS4194		BAS4456	*pheS*	AGTTGCGAACAAATAAAAAACTTCTTATAATAAGT
BAS4194		BAS0093		AAAATGACATATTATAAAGATTTATTTATCGCCAG
BAS4194		BAS4422		GATTGTAACGTTATGGTGACTATAATATAATGAAA
BAS4194		BAS0067		CTTCTATAATGTTTAAATTAACTCTTAAACATTAT
BAS4194		BAS0061		TGTTGACTTTAAATTATATCTGAGGTAAGATACTA
BAS4194		BAS4414		CTAATAATATATTGTAAAGAGTAAATATTTTTGTC

BAS4194		BAS0048	*glmU*	GTAAAACATCTTATAAGAATTATGGTAAAATTTAA
BAS4194		BAS0020		AAGATAAAAATTTATCGGATTATTTCTTAAATTTC
BAS4194		BAS3997		TATTAAAAAGAAAGCGTAAACATGTTATGATATCA
BAS4194		BAS0001	*dnaA*	CATTGCTATAGCTACTTTTTTTTGATATTATAGTT
BAS4194		BAS3960		CATTGAATCGCTTACAATAGTTATGTATAATAACT
BAS4194		BAS3946	*mtnW*	AATTGACAACATGAAAAATATCTGACAAAATTCAG
BAS4194		BAS3945		TCTTTACAAAATACTGGAAAGATTATATTATTTGT
BAS4194		BAS3944		TGTTTATTATATTAGAAAGGTCATAAAACATTTCT
BAS4194		BAS3943	*mtnK*	TTTTTAAGATTTTTTAATATGTTGAGAATAGTTCT
BAS4194		BAS3916		GTTTACATTCATTTCGAAAATCTGTAACAATTATT
BAS4194		BAS3900		TAAAGATTTCTATTAAATAAAATGATAAAATGAAT
BAS4194		BAS4363	*hemA*	ATTAAAACATTTTAGCTATATTGCTTTTTATTTAT
BAS4194		BAS4355	*valS*	GTGATAACATATATTATTAATTGTTAGTATATTTA
BAS4194		BAS4328		TGGATATAAATGTATCAAAATTTTATTACTGTTCT
BAS4194		BAS4327		TCTTGTCATTATTTTAAAACTATGTAAATATAGGT
BAS4197		BAS4661		AAGAAATAAATAGTATACATTATTTATATAATTGTGTTATACTATTGTTGGGGATTA
BAS4293		BAS0067		TATAGACAAAACCAATAAAAATACTCGGTGTTAGGAGT
BAS4293		BAS4271		AAATAATAATAGCTAAACAAGTGATTAAAAAGCAAGAT
BAS4293		BAS4148		TTCCGGAATTCCCTAGTTGACTTATAGGTTTTATTGAT
BAS4293		BAS2889		TTAGTCAAAAACATTGTATTTTTATCGGAATAGTGAAA
BAS4293		BAS2714		TTTTAATAAAACCTAGTTGACTTATAGGAAATATTTGT
BAS4329		BAS4546		ATTAATTCAATACGGTAATCGTTACCGCAGTCATGATGAGAACCTTGCAGAGAGCTTAAATTATGCGGAGAAATTATTCCGTGAATATCAATACGATGCGGCTTTAGA
BAS4329		BAS4328		TGTTGCCTCCTACTTTACACTTTTCAACTGTCTTGACACCTATATTTACATAGTTTTAAAATAATGACAAGAGTTTTTTTATAATTCTTATTTTCACTATCATTTAAA
BAS4329		BAS4327		AAATTTACTATCACTTTTATTCTTAATATTTTTTTGAGAACAGTAATAAAATTTTGATACATTTATATCCACAGTTCTGTCAACTTTTCACATTTCATCCTCCGTTGT
BAS4421		BAS0832		TTATATGTCTTCTGTTTGACTTTTAAAATCTTCTCGATA
BAS4421		BAS1019		AATTTTCTTTTAATATAATTAATTTAGAATTTTTAAATA
BAS4421		BAS4877		CGACTTATGTCGAAAAATTGAATGAGCATTCATTCAAGA
BAS4421		BAS5194		ACAACTGACTTACGAGTGAGTAATATATTACTTCTATCC
BAS4421		BAS4422		AGTACTTACTTGTAAGTAAATAAAAAATACATAAACTTC
BAS4421		BAS3900		TTAAATAAAATGATAAAATGAATATGTATTCATTTTTTG
BAS4421		BAS3220		TTCATTTACCGACAAAAAATAAACAAAATATAAGTGATT
BAS4484		BAS4484		AAATTTTTATATTATATGTTTAA
BAS4484		BAS4031	*cpdB*	ACAAATACTTTTCAAATGTTATT
BAS4484		BAS3476		AATCTTTACACATTCTATACACT
BAS4484		BAS3300		ATACTTACCTTTAATTTCTTCAA
BAS4484		BAS0681		ACATTGTTAAGAGTAAATTTACA
BAS4484		BAS5319		ATAAACACATTAAAATGTTTTAA
BAS4484		BAS4244		AAATGTTAAATAAGAGATTTACG
BAS4484		BAS2789		TATTGTTTAACAATAAATTTTTA
BAS4484		BAS1386		ACGTTAGTAATACAAATTGAAAT
BAS4484		BAS1213		AAACTTTTATTTTGTTTGTTAAT
BAS4575		BAS1308		TCTTTCGCAACTCT
BAS4575		BAS0823		ACATTCGCATAAGA
BAS4575		BAS4535		TACTTTGCAAAAGT
BAS4575		BAS0959		TGACACCGCTTTCA
BAS4575		BAS1300		TCTTTTGTAACAGT
BAS4575		BAS0832		TCTTTCATAAATGT
BAS4575		BAS0808		TCTTTAGCAAAAGT
BAS4575		BAS1199		ACCTTCCCCAAAAA
BAS4575		BAS0631		ACATTTGCCAATGT
BAS4575		BAS1019		ATTTTACGCATACA
BAS4575		BAS4992		ACCTTACCCACAGA
BAS4575		BAS0598		ACATTTGCCATAGT
BAS4575		BAS0547		TGAAAGCGTTTTAA
BAS4575		BAS0535		TAAATACGAATACA

BAS4575		BAS0530		ACTTTCGCATTAGT
BAS4575		BAS0464		TATAACCGTCTTCA
BAS4575		BAS5238	*eutD*	AGATAATGTTTTCC
BAS4575		BAS0312		TTACAACACTTTAA
BAS4575		BAS4690		GGATATCGCTTTAT
BAS4575		BAS0295		ACATTTGCAAAAAG
BAS4575		BAS4561		TCCTTCGCAAAGAT
BAS4575		BAS4560		TAGAAACGCTTCCT
BAS4575		BAS4543		ACTTTTTAAACAGT
BAS4575		BAS0162		TAAAAGCGCTTGCA
BAS4575		BAS4488		ACCCTCTTAAAAGT
BAS4575		BAS4422		ACCTTCACGATAAT
BAS4575		BAS3925		ACTTTTGCAAAATA
BAS4575		BAS4072		ATGTTCGCCAGAGT
BAS4575		BAS3380		ACCTTCGCTAAAAT
BAS4575		BAS3348		AATAAACGCTGTCT
BAS4575		BAS3220		TGAGAACGATGCAA
BAS4575		BAS2645		AATTTAGGCAATGA
BAS4575		BAS2588		GTTTTATGCTTTCA
BAS4575		BAS2574		TGAAAACGCTTTTA
BAS4575		BAS1803		TGGAAACGCTCGAA
BAS4575		BAS1754		AGAATACGTTTTAT
BAS4575		BAS2188		TGTAAAGGCTTACA
BAS4575		BAS1666		ACTTTCGCAAGAGT
BAS4575		BAS1610		ACTTTCGTCCAATT
BAS4575		BAS2065		ACTTTAGCAAAAAA
BAS4836		BAS4836		GACTCTAACGTTGCGTCATA
BAS4990		BAS4990		CCTATGTGGGACGCAAAATGTCACTACGGGACGTAAAGTGACCACA
BAS4992		BAS1401		TTTATAAATTTATAAATTCATAAAATGAAAAGATTT
BAS4992		BAS0808		AAATCGTTTTCACAAATATTTCATTTGAAAAATGTT
BAS4992		BAS1071	*rocD*	AGAGTATTTACAAACTTCTTGTAATAACAAAAGGGG
BAS4992		BAS0312		AAGGAAATAATAGAGAAATTGAAAATGTAATTGAGC
BAS4992		BAS0310		AGAATGTTGGCATACATTTTGCAATAAAAAGAGAAA
BAS4992		BAS0295		CAGGATCCTTTTTAATTTTTGCATTTTTATTAATTG
BAS4992		BAS4071		AAAAAGTTGGCACGGTATTTGCTTAATAAAAAGACG
BAS4992		BAS3348		ATTTTAATTATTAACGTTCAGGTTCTGGTTAATTGC
BAS4992		BAS2588		TAGAATTTGGCACAGTACTTGCAATATAAAAGATGA
BAS5025		BAS2993		ACTATTTGTAATATTATATAATAAAA
BAS5025		BAS5025		AGAGATAGTAGTCTTTATGAATACTA
BAS5200		BAS2788		TTAATTAGGAGGAGTGG
BAS5314		BAS3395		CACCATCCTCATCGTCTTCTTTTACGATACAGAAATCGACCTTATCTTCACTTA GCAGATAGATCTTGACAAAATCATCTACCTCTTCTTTAATATAACTGTGTATTT CTTAA
BAS5314		BAS5314		ACAAAGTGTACTACGCCGTCTCGGCTACAAGATACGAAAAGTATTGTTATTAT CTTTCGATCGAGAATAATGTCTATGCCCTATGCACTCACTGGCATACTTTCCAC AATAAT

Table 1: Binding sequence predictions for *Bacillus clausii* KSM-K16. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
ABC0001	*dnaA*	ABC0001	*dnaA*	ACAATTGTGGATAAAAT
ABC0075	*purR*	ABC3216	*guaC*	CTATGGGCGAATGATTTTTGTTCTTTTATTCTATAACATTCGTTTTTAGGG
ABC0075	*purR*	ABC2338	*pyrR*	ACCTCCTTTGCAAGTTTGTTATATCAATATATAGAACACTTGACCTTTGCC
ABC0075	*purR*	ABC1443	*xpt*	AAAGTTTCGGCTTTCTAGTGAGATAAAAATATCACCAGAAAGCCGAAAAAT
ABC0075	*purR*	ABC0959	*pbuG*	TCAACTTCTGCTTATTACCCTCCCCATATGAAGGGTAACAAGCAAAAAAAT
ABC0075	*purR*	ABC1024	*purE*	TCTATTTTTGCTTGTTACATTTTATGAAAGCTATTTTACAAGCTGATAAAC
ABC0075	*purR*	ABC3863	*glyA*	GTTGAAACTGAATAAAAGAAAAAAATCGCTTATTATCTTCCGGAAAAACCT
ABC0075	*purR*	ABC4100	*purA*	CAAAAATATGCTTACAACCCCTAAAAATCATACCTTTACAAGCACAGATTA
ABC0133	*sigH*	ABC3069		GAAGGAATATTGAAGGGAATTGTAGAAGTAAGCAAGG
ABC0133	*sigH*	ABC2457	*spo0A*	TAAAGGATTGCAGCCATCCTTGTCGAATCGATTACAT
ABC0133	*sigH*	ABC2354	*ftsA*	TAAAGGATTCCATAAATGCATGTTGAATACTTTACGT
ABC0133	*sigH*	ABC1746	*fumC*	AAATGAATATTATGAAACAGAAGAGGAAGGCATCTAT
ABC0239	*sigW*	ABC1671		GAACTGAAACCTTTAGGCGAATGGAATCGTACATA
ABC0239	*sigW*	ABC1396	*pbpE*	AATCATGAAAGTTGCGTAGAATGGGATCGTTCACT
ABC0239	*sigW*	ABC0519		CGGTTTGCCAACTTTAACGTTGTCTATTGTTGGAG
ABC0239	*sigW*	ABC0239	*sigW*	TACACGAAACCATCTGTCAGTGAGTTCCGTATGTT
ABC0239	*sigW*	ABC2747	*sppA*	CTATATGATCTTTCTCCGTCTTTCCCCGAAAACAC
ABC0239	*sigW*	ABC2538		GAACCTGGCTAACGAGGAGCATTACCAGGTGTATA
ABC0239	*sigW*	ABC2529		AAACATACTTTTACTGTAATTGCTTAAAAATAACA
ABC0421	*iolR*	ABC1818		TGCATATTTGGTGTATTTTC
ABC0421	*iolR*	ABC1452	*gabD*	CTTTAAAGCGTTTTGCAGTT
ABC0421	*iolR*	ABC0980	*gbsA*	TATATAATTTGTTTTGTTTC
ABC0421	*iolR*	ABC0965		CTTTTTCTTCTATATTAGCT
ABC0421	*iolR*	ABC1114		CTTTTCAACGATTAAAGAGG
ABC0421	*iolR*	ABC0422	*iolA*	TGGTTGTTTACTAGTTTTTC
ABC0421	*iolR*	ABC0421	*iolR*	CTTTTTGATCATTTGTTGGT
ABC0421	*iolR*	ABC3810		CTCTTGCGGCATTAATCGAA
ABC0575	*xylR*	ABC0572	*xylA*	AACTTAGTTTAAATAATAAACAAACCTAATGAAAG
ABC0816	*sigB*	ABC0124		AGGTTTGAGAGTTGTAAACTTGTCTATAATAGCTAAAA
ABC0816	*sigB*	ABC0008		CTAAAATCCTTGATCGGTTCTCAGGAAATTTATTTTGT
ABC0816	*sigB*	ABC3969	*katX*	CGTCTACTGGAAAAGCTAGGCTGGAATAAAAGGAAGGA
ABC0816	*sigB*	ABC3924		TGGTTCCCCTTTGGCCGAAACGGAAGGTGAACAAGAGG
ABC0816	*sigB*	ABC3810		CCTTTCAAAATGAAAGACAGTGGCTACAACTCATGATT
ABC0816	*sigB*	ABC3708		AGAAAAGTAGAAACGGGTATGCCAAATAATATTCCTTT
ABC0816	*sigB*	ABC3698	*gtaB*	TTTTTCCAGCTTGAAAATAGAAGAAATAAAAAATAAAA
ABC0816	*sigB*	ABC3615		GGGTTTGAGCTTATGAAAAATGGTTCAAATTCACTCAA
ABC0816	*sigB*	ABC4099		CATTTACATCTTGATAAAAAGTCGAAAGAAGCGAAAAT
ABC0816	*sigB*	ABC3507	*gsiB*	GGGTTTTAATTGATTTCTTTGAGGTATACAAATACCAC
ABC0816	*sigB*	ABC2731		AGTTTTAGTCGAAAGAAATTCGGCTATAAATCAAAAAG
ABC0816	*sigB*	ABC2645	*ilvB*	TTTTTTGTTGGCGGAAAATAAATAAATAAAAAATCAAA
ABC0816	*sigB*	ABC3098		CGTTCGGTTAAAAGCGGAATTGGAAACAGAAATGCAAG
ABC0816	*sigB*	ABC3069		GGTTTGAGAAGGAATATTGAAGGGAATTGTAGAAGTAA
ABC0816	*sigB*	ABC3026	*clpP*	ATTTACGAGAAAAAGGAGATGTACAGTATCGTATTTGG
ABC0816	*sigB*	ABC2573		CGATTTTGTAAAGGCTATTGTGGAAATAAATGATTTTG
ABC0816	*sigB*	ABC2529		TGCTTTTTTTTTGGGCCAATAGGGCAAACTAAATGATG
ABC0816	*sigB*	ABC1818		AATAAAAAGGGAAAAGGAGAATTTAGCCGGTAATGATG
ABC0816	*sigB*	ABC2151	*nadE*	TGTCTCGAAGTTAATTGAATGGGGCAAAACGTACGGAT

ABC0816	*sigB*	ABC2131		ACGTTTTTTGCTTTGGCAAATGGGTATGGTACAATCAA
ABC0816	*sigB*	ABC1667	*iolS*	GTTTTAAGGGGCGATTATAAGGGAAATAGCATGAAAGA
ABC0816	*sigB*	ABC1570	*relA*	CGTTGAAGAACTTGCCGTTTATGCAAAAGAAAAACAAG
ABC0816	*sigB*	ABC1484		GTTTAAATACACTCACTTTTGAGGTATACCAATACTAA
ABC0816	*sigB*	ABC1452	*gabD*	ACGTTTACATATAGAGGGAAAGGGAACAGAATAAAAAA
ABC0816	*sigB*	ABC0980	*gbsA*	TATGTTTATTTTGTATAAAACGTTTAATATATATTAAA
ABC0816	*sigB*	ABC0965		AGGTTAAACTTGTTTTAACTTAAGTAAATTTCCACTAA
ABC0816	*sigB*	ABC0907	*csbB*	AGATTGAATAGGCGTGAAACAGGGTAAGATAAAATGGA
ABC0816	*sigB*	ABC1283		AGCAACCATTTCACGGGAATATCCCGCTTTAATTTTGG
ABC0816	*sigB*	ABC1259	*alsT*	TCTTTTAAATAGCAAAAAATGTGATATGATGCTTAAGG
ABC0816	*sigB*	ABC1114		GGTATGATGATCGGCACCAACGGAAGTATTAATAGTAC
ABC0816	*sigB*	ABC0422	*iolA*	GGAAATTTCAATAAGTTGTTTATTGGTTGTGAATTGGT
ABC0873		ABC2872	*ldh*	TTTAAAAACACTTTATTACGTGTTATATTTACA
ABC0873		ABC2645	*ilvB*	AAAGGCTATTTGGCCCGTTACTCAAAACTTGTT
ABC1265	*treR*	ABC1453		AGAACAGAAGTGGCGTCCAGAATTTAGAAATA
ABC1265	*treR*	ABC1263	*treP*	TAACTGCATAAACATATATGTCCAATAATAAC
ABC1322	*perR*	ABC1322	*perR*	AATATTAATTATATTTCTTTC
ABC1322	*perR*	ABC1283		ATTTTTAATAATATTTATTGT
ABC1322	*perR*	ABC0269		AAGACGAGTAGTATTGTTTTG
ABC1322	*perR*	ABC0224		TTCATTTTGAAACGATTATAA
ABC1322	*perR*	ABC3969	*katX*	CTTATTCAAGAATCATTGCAA
ABC1322	*perR*	ABC3349		ACTATTAATAATAGTATTCTT
ABC1322	*perR*	ABC3231		GGTTTTTCTTATGTTTATTTG
ABC1322	*perR*	ABC2632	*hemA*	CCTTATTTAGACATGTTATAA
ABC1322	*perR*	ABC2528	*spxA*	TGTATAATAAAATGGTGATGA
ABC1529	*hpr*	ABC0761	*aprE*	AAAAGAACAATAATGAAAAA
ABC1584		ABC3750		TACTTGTTACTGATAAATTTATCGCAACAATATTAAGT
ABC1584		ABC3398		GGCCTTTATAACCAACTAAATTTATGAGAAAAGAAGGA
ABC1584		ABC2897		TGTATATTAAGGGTGAACTGAGTATACTAATATGTCTT
ABC1584		ABC3246	*ssuB*	ACAATGATAGGAATAATTATTACAACCAAAATATCCAA
ABC1584		ABC2792		CTTATTTATGTTAAACTAAATAAATGGAACTTTAAAGA
ABC1584		ABC2099		ATCGCTTTTTTGCGGATGGCCTTATCGAAATTTTTTAA
ABC1584		ABC0109	*cysK*	AAAATATTTTGGTTACTTCTATGATCCCTAACTCCTCC
ABC1657	*hrcA*	ABC1657	*hrcA*	TTAGCACTCGTTGCATTTGAGTGCTAA
ABC1686		ABC2879	*pckA*	TTATTCACTTTAGTATGGACTATTAAATAAAATGTGTTATACTAAAGCCAC TTAAAA
ABC1686		ABC2705	*gapB*	GGATTTAACTTTAACCAAGTGCAAAAACCAAATAAATACAATATGATAAA GTATATT
ABC1690	*sigA*	ABC0980	*gbsA*	AATAAAACATATTTTGCAAATTATATATAATTTGT
ABC1690	*sigA*	ABC0970	*htpG*	TAAATACAATTGCTGTTAATACCGCTAACCGTTAG
ABC1690	*sigA*	ABC0965		AAATTGAATTCATTTAAAGGTGATTTCGAAGGTAT
ABC1690	*sigA*	ABC0964		AGATTGTTTATTGGCCAGGTTCGTCTAAAATAAAA
ABC1690	*sigA*	ABC0936	*guaA*	TAACTAAAAAAGGATATATCTTAAAGAAATCGTTT
ABC1690	*sigA*	ABC0917	*maeN*	AAACTCAAATTTTCTTTCCGGACGAAAAAAATTAA
ABC1690	*sigA*	ABC1322	*perR*	TCTTGCTTAAAGCCAAGTTACAACTTATAATTAAT
ABC1690	*sigA*	ABC1313	*fabL*	ACTTATCGGATTTAAATTAACGTAATACAATTGTT
ABC1690	*sigA*	ABC0810	*rsbR*	TTTGCCATTTATAAGAACGATTCCTTAAAATGTGG
ABC1690	*sigA*	ABC1263	*treP*	CATTGACGTATTTGTATATACAGGTTATTATTGAA
ABC1690	*sigA*	ABC1257	*glpD*	TGTTTACGCAGCAGCTTAAGTGCGCTATAATGAAT
ABC1690	*sigA*	ABC0776	*dppA*	ATGTCCATATATTATACACAAGGAGTAAAATAAAT
ABC1690	*sigA*	ABC0761	*aprE*	AATAGGCAAATCTTTCTAACTTTGATACGTTTAAA
ABC1690	*sigA*	ABC0734	*des*	AATAAAATATTTCGTCAAATTCTGTCTTAATTTGT
ABC1690	*sigA*	ABC0722		TATTATGTTTGTGAATAAATTCACTTAAAATAAGA
ABC1690	*sigA*	ABC0715	*narG*	CCTTGCCTACCTTGAACAAGATACGTATGATCATA
ABC1690	*sigA*	ABC0713		CATTGCCAAATTAGGAAAACAACTATATAATCTTT
ABC1690	*sigA*	ABC1114		GTTTTGATGATCATTTATTACCTGGTATGATGATC
ABC1690	*sigA*	ABC0665	*citH*	TTAATAACATCTAGCAAAAAAGGTTTGTGAACTCC
ABC1690	*sigA*	ABC0643		GAGATAGTATGGTGTTTTAACATTAGAGCATATTT
ABC1690	*sigA*	ABC0611		CATTGACAGTCTATTCGAACACAGTTAGAATTTTT

ABC1690	*sigA*	ABC1071	*opuAA*	TTTGATTAAAAAATTGTAAATAAGGTATAGTTACA
ABC1690	*sigA*	ABC1024	*purE*	TGTTGACACTTACGAGGAGGATGCGTAAACTAAAA
ABC1690	*sigA*	ABC0589		AAATTAGCATTGCTTAGTAACAAAAAAAGCGTTGA
ABC1690	*sigA*	ABC0572	*xylA*	TGAATCAAATTTATTATTTGTTTGGATTACTTTCT
ABC1690	*sigA*	ABC0456		ATCATAAAATATGTTAACTTTCGCGAATTTATTCC
ABC1690	*sigA*	ABC0453		ATTGGCCGTTTATCGCAAAGGCTGATATGATAATG
ABC1690	*sigA*	ABC0450		CTGATCATATCTTTTGTTGACTGGTGTAACATTCA
ABC1690	*sigA*	ABC0432		TATTTATTTTTTTCAGTTATTCTGTTATATTAATT
ABC1690	*sigA*	ABC0422	*iolA*	GTTGACCTTTTATGGCATATGGTTTTAATATATAA
ABC1690	*sigA*	ABC0421	*iolR*	AATATATAATTTTGGTATACGGTATTTTCCAGTTG
ABC1690	*sigA*	ABC0350	*hmp*	ACTTTAAATATATATATAAAATACTTATTTTAAGT
ABC1690	*sigA*	ABC0329		AAAATGATTAGGATTTGACGTGGCGTTAACTTTCT
ABC1690	*sigA*	ABC0326		CTTTCCAAATCTATTTAAGATCTGATAAAGAATGC
ABC1690	*sigA*	ABC0315		TTTTTTACTTAAACGTTAACTTAAATAGAATGGAG
ABC1690	*sigA*	ABC0269		TAAGTAAAATCTGTCTAAGACAAAAAAACAGTTAA
ABC1690	*sigA*	ABC0224		AAAGTAAAACTTTGCTAATATTGTGTCACATTTTC
ABC1690	*sigA*	ABC0194	*ptsG*	GGTTGCAAAACTGTTGAAACAAGAATATCCTGTAT
ABC1690	*sigA*	ABC0142	*rpoB*	AATTGACTGCTGGTTTTGACTATGATATTATCATT
ABC1690	*sigA*	ABC0109	*cysK*	GATTGACTTATCTGAAAATGTTTGCTATCCTATTT
ABC1690	*sigA*	ABC0101	*spoIIE*	TTTTGATATTGGTCCTTAATTCTGACAAATTGCTC
ABC1690	*sigA*	ABC0100		TTTTGTGTCTCATAAACAAAATGGACAAAATAAAA
ABC1690	*sigA*	ABC0078	*gcaD*	CCTTGAAATACCGCATTTTTTAAAGTATAGTTTTA
ABC1690	*sigA*	ABC0017	*rocD*	AAAATAAAAAAAAGTTTTATGGGGAGGAAATTTTC
ABC1690	*sigA*	ABC0016	*rocR*	CTTTTAAAGGAGGGGTATTTTGAAAAAAAATAAAA
ABC1690	*sigA*	ABC0001	*dnaA*	CATTGCGGTATCTAGTTATTTTTGCTATGATTAAG
ABC1690	*sigA*	ABC3978	*thrS*	AAAATAAAATAAATGAAATTTTCCTCACAAAGAAC
ABC1690	*sigA*	ABC3952	*phoB*	AAAACAACATGAAGATAAAAATGTATCAACTTGGT
ABC1690	*sigA*	ABC3950		AAGCTCATATGACCTTCATTTAAACTCAACTTTCT
ABC1690	*sigA*	ABC3938		ACTTTCAAATATATTTTCTCACAGATTTCCTTTTT
ABC1690	*sigA*	ABC3924		TTTTCTTTAAGGCAATGCACCATGCTATGATGAAT
ABC1690	*sigA*	ABC3910	*eutD*	GGTTTCCTTTTTAAATAAAGAGGGGTATCCTAAAT
ABC1690	*sigA*	ABC3908	*gltT*	TCGGTAATATGGTCTAAAGTATAGGTGATTCTTTT
ABC1690	*sigA*	ABC3899	*pbpG*	TCGTTCATATCGCCTACGCAATGCTTAACGTTTCT
ABC1690	*sigA*	ABC3898	*speE*	TCTTTGCAATTCGTAACGCATCCGCTATACTTGCT
ABC1690	*sigA*	ABC3892		GAATGCTCATTCATAAGGATTGTGTTATACTACGC
ABC1690	*sigA*	ABC3886	*pyrG*	ACTTGACTTCTTGATCTGTTATCGGTAAAATCATT
ABC1690	*sigA*	ABC3863	*glyA*	CCTATTCTTTTAGCGATATATACGATAAAATGAAA
ABC1690	*sigA*	ABC3823		GCTTGAATAACAAAACAGAGCAAGCTATTATCGAT
ABC1690	*sigA*	ABC3822	*galE*	TATATTTTGTTTTTTGCATTTCTGGTATACTAGAG
ABC1690	*sigA*	ABC3819	*gde*	CAATTAATACATTCTTTTTGTTCGCGTATCTTTAA
ABC1690	*sigA*	ABC3817	*lytR*	TAAATAAAATAAAAAGTGAATTTTTATGTTTTTCT
ABC1690	*sigA*	ABC3810		GTCTTATAATATTAGTATCTGTTCGTTTGTGTTCA
ABC1690	*sigA*	ABC3792	*bglH*	TGTCTAAGTTTTAAATTAATATTGTTACAATAAAA
ABC1690	*sigA*	ABC3790	*bglP*	TCATGGCTTGGGTTTTTTATATTTATAAACTATTG
ABC1690	*sigA*	ABC3750		AATATATAATACTTTTTTAACATCCATCAATTTCA
ABC1690	*sigA*	ABC3744	*pucE*	CATTTGCAAATGGGCCGTGAATGGTTAAACTGATT
ABC1690	*sigA*	ABC3738	*pucH*	CAATGATTTATCGTAAGCTTACCAGTAAAATCAGT
ABC1690	*sigA*	ABC4100	*purA*	GGTAGAGCTAGCATTTGTAAAAGAATACAATAAGT
ABC1690	*sigA*	ABC3698	*gtaB*	GCTTGAAAATAGAAGAAATAAAAAATAAAATCTTT
ABC1690	*sigA*	ABC4099		AATGTAGAACTATTTTTCAGCTTTCTTCGCTTTTA
ABC1690	*sigA*	ABC4094		CATTTAAGCCTCGACAACAATATGAAAGAATTAGC
ABC1690	*sigA*	ABC4043	*citA*	AATTTCATTTAGATTACCATGGCGCTATGATAAAG
ABC1690	*sigA*	ABC4042	*citR*	GAAATAGTATCGCGGTACCATTAGATTTACTTTAA
ABC1690	*sigA*	ABC4020		AGTTGCGAAATGTCCGTTTGTCTGGTATAATTCGT
ABC1690	*sigA*	ABC3549	*rbsR*	TTGGTAATTTACTTTCTTTAACAGATAACTGTTGG
ABC1690	*sigA*	ABC3483		GGAATAGTATTAGCTGAAGGTGAATAAATCTTTCT
ABC1690	*sigA*	ABC3449	*clpE*	GGTTGAAAGGTCAATGAAGGTCAGTTATAATAAAG
ABC1690	*sigA*	ABC3435		CCTTTCCTTTATAATTTAAAATGATTAAATTTAAA

ABC1690	*sigA*	ABC2990		TATTGCAAACGCTATCATAATTGACTATAGTAAAA
ABC1690	*sigA*	ABC3398		CCTTTATAACCAACTAAATTTATGAGAAAAGAAGG
ABC1690	*sigA*	ABC3391	*gltC*	ATTCGCTATCTGCTATCAAGTATTTTATAATTAAG
ABC1690	*sigA*	ABC3383	*glpF*	CGTTTACAAATTTGTTAAAACAGGATATAATGCGG
ABC1690	*sigA*	ABC3371	*xsa*	TTTTCACACACACTTAAACTAAAGATAAAATTGAA
ABC1690	*sigA*	ABC3349		ATTGTAAAATTACTATTAATAATAGTATTCTTTTA
ABC1690	*sigA*	ABC3337	*nagP*	AAAATACGACGATAAACCTAAAAAAATACCGCTAG
ABC1690	*sigA*	ABC2897		ACTTGACTCATATGATTATACAGAATACTATAAAG
ABC1690	*sigA*	ABC2879	*pckA*	GTATGGACTATTAAATAAAATGTGTTATACTAAAG
ABC1690	*sigA*	ABC2872	*ldh*	ACTTGAAAATTTTTGTGAAATAATGCACAATATAA
ABC1690	*sigA*	ABC3255		ACTACCCAAAATAAGAAATATTCTGTATGATATAG
ABC1690	*sigA*	ABC3253	*manA*	ACTTATCTTCGCTACAAAGATGTTTTATACTAGAG
ABC1690	*sigA*	ABC3246	*ssuB*	CGTTGACGGTGAAAGCGCGCTCCATTAAAATGAAT
ABC1690	*sigA*	ABC3231		ATTTTCACAACGGCTTTGCTCCTTCTTACCGTTTT
ABC1690	*sigA*	ABC2792		TAAATACAATTTGATTTATTTACCTTGAAATTTCT
ABC1690	*sigA*	ABC2788	*acoR*	AGCAGATATTCGATGATGATTCTTTTATATTGAAA
ABC1690	*sigA*	ABC2761	*acuA*	AATATCATATTTGTTCCATTGCAAAAGTTTGTTTT
ABC1690	*sigA*	ABC2760	*acsA*	TTTTGTTTGAAAACGTTACCTTGTTTATACTATAA
ABC1690	*sigA*	ABC2759		CGTATGCGCATCACGCAATTTATGATATACTAAGA
ABC1690	*sigA*	ABC2757	*rpsD*	CATTGACAAGTAGGCGCTTTTCCGCTATGATAGAT
ABC1690	*sigA*	ABC2751	*erzA*	ACCATAATATAATGCATAGTCTTATCCCTATTTCG
ABC1690	*sigA*	ABC2750	*nifZ*	GCTTGCAATAGCGAACGAGGCTTGCTAAGTTAAAA
ABC1690	*sigA*	ABC2740	*ackA*	GCTTGAAATCAAGGCTGGTGAATGCTTAAATGGTA
ABC1690	*sigA*	ABC2715	*citZ*	ATCGCTATTTCCGTTTTGTTACTGATAAAATGGAC
ABC1690	*sigA*	ABC2705	*gapB*	GGTTTATTTATGTTATACTATTTCATATAATATTG
ABC1690	*sigA*	ABC2704	*speD*	AGTTGCAATCGCGGTGTAAACCAAGTATACTGAAT
ABC1690	*sigA*	ABC3153	*adaA*	CCTTATATTTTTTTATTTATAAGGTTATAGTGTTA
ABC1690	*sigA*	ABC3152	*alkA*	ATTGTGATATTGGAATATTTATTTTTTTATATTCC
ABC1690	*sigA*	ABC3141		TTTGGGAATTTCCTATAGATTTTGTTTAACTATAC
ABC1690	*sigA*	ABC3120		GTTTGATATATCAGTAAGAAAACGCTTTAAAAAAA
ABC1690	*sigA*	ABC3101	*tagD*	AAATTAAGATGGCCTAAAGGGGTTCTTTAATGTTT
ABC1690	*sigA*	ABC2697	*thrS*	CTTGCATTTTTAGCTTTGTTTTTTGTATAATGGCT
ABC1690	*sigA*	ABC2689	*infC*	CATTGACAGGAACCATTCACCATGGTAAGATACGT
ABC1690	*sigA*	ABC2680	*pheS*	GCTTGCAGCTTGTGTGGCAATTGATTATAATAAAA
ABC1690	*sigA*	ABC2673	*lcfA*	TTTGAAATAATGTAAGCGATTACAATACAATGAAA
ABC1690	*sigA*	ABC2672		ATTTGACAATCGCCCTTTCCCGTACTATACTAAAT
ABC1690	*sigA*	ABC2664	*sdhC*	TCTTGACTCGGATTTTCCCTAGGAGTAAAATGAAA
ABC1690	*sigA*	ABC2645	*ilvB*	CGTTATGAAAATCTAATAACGATAACACAATGAAG
ABC1690	*sigA*	ABC2636	*clpX*	TTTGCATCGTTTGCTTGAATAGTATTATAATGTGG
ABC1690	*sigA*	ABC2632	*hemA*	TTTTTGAGTCCTTATTTAGACATGTTATAATTAAG
ABC1690	*sigA*	ABC2621	*valS*	TATTGACGATTGCACTGGCAAACGGTAAAATACGG
ABC1690	*sigA*	ABC3098		ATGCTCAAATGGGACTACATACTCCTAATACTTTT
ABC1690	*sigA*	ABC3087	*degS*	AATGGGCAATAAAGCGTGATTTTTGGATAATGATT
ABC1690	*sigA*	ABC3068	*secA*	ACTGCTAATATAGAAATGATAAACGTATCATAAAG
ABC1690	*sigA*	ABC3029	*opuCA*	TATATAAAAAATTGTAAATATTCCGAAATGCTTAG
ABC1690	*sigA*	ABC3026	*clpP*	ATTTGACCTTTATTGACCTTTGCAGTATGATGATT
ABC1690	*sigA*	ABC3022	*cggR*	ATGATAATATTCTTAAAAGGGGTTTCAATATTTTG
ABC1690	*sigA*	ABC2591	*pstS*	TTTTTTTATGCCTTTTTCTTTCCATTAAAATATAG
ABC1690	*sigA*	ABC2578		TCATAAATTTGTCATAATAATATGTTAAATTATGA
ABC1690	*sigA*	ABC2558	*argC*	AGTTGAATTTATTTTTATTCAATCATATAATTAAA
ABC1690	*sigA*	ABC2547	*fabH1*	TCTGGACAGCAACAAAAATGGTTGCTAATATTAGG
ABC1690	*sigA*	ABC2528	*spxA*	TCTATACAACAGTTGAACAAACTTGTATAATAAAA
ABC1690	*sigA*	ABC2508	*comGA*	ATTGTTTTATGATATATTTTTTTGTTTTTTCTACG
ABC1690	*sigA*	ABC2457	*spo0A*	CAGATAAAATTTAATAAATTTCCTAACGTCGGTAG
ABC1690	*sigA*	ABC2454	*bkdR*	GAGATAACATTTCCTCAAACGGTTTTTTCATGTAC
ABC1690	*sigA*	ABC2420	*pdhA*	TATAACATATTGTGTCAAAGTAAAAGAATGATTAA
ABC1690	*sigA*	ABC1993		TCTTCCATTATCAAGCCTCATTTAGGATAATGAAA
ABC1690	*sigA*	ABC1992		TGTGGTTAAAAAAAACAAAATCGGGTATAAACTAA

ABC1690	*sigA*	ABC1965	*blt*	ATTAATCGTTTGCAGAAAAGTATGATATATTGTAC
ABC1690	*sigA*	ABC1926		GCTTTCCTTGATTTAGAGAAAGCATTAAAAGAAAT
ABC1690	*sigA*	ABC2394	*ctaA*	AATTGACTTTACTTAGTACCTTTTTTAAGATGGAA
ABC1690	*sigA*	ABC2354	*ftsA*	GCTTTTCACCTTCTCCATCTTGGTGTAGACTAAAT
ABC1690	*sigA*	ABC2352	*spoIIGA*	GACAGACTTTCTTATGAAGATCTACTATACTAAAT
ABC1690	*sigA*	ABC2342	*ileS*	AGTTGCATTTTCTGGGAACCGTCCTTATAATGAAA
ABC1690	*sigA*	ABC2338	*pyrR*	CATTGACAACCATTGACTAGTTTGCAATAATGATT
ABC1690	*sigA*	ABC2304		GGTTGAATCCTACGCTGAAGATCTATATACTAGTC
ABC1690	*sigA*	ABC1895	*trpE*	AAATAGCAAGATAGTTGAGTTTGAGTATAATAAGA
ABC1690	*sigA*	ABC1883	*folE*	TTTTGCTTTTCCACTGTTCTTTTGCTACTCTTTAG
ABC1690	*sigA*	ABC1867	*metI*	GGTTGCACTTGCGTGAAAGTTGCGGTAAAGTGGAA
ABC1690	*sigA*	ABC1862	*gudB*	GATGGCAATTATGTGAGCATTTCATTAAGATAAAG
ABC1690	*sigA*	ABC1835	*resD*	ACGCTAATAATACAAATTGGAACATAAACAGTTAA
ABC1690	*sigA*	ABC1818		ACAATAAAAAGGGAAAAGGAGAATTTAGCCGGTAA
ABC1690	*sigA*	ABC1805	*mmgD*	TTAATCACATTTGCTGATGTTAAAAGAAAACATAT
ABC1690	*sigA*	ABC1803		AATATATCTGTGATAGAAGTTTTCGTATAATGTTC
ABC1690	*sigA*	ABC2216	*asd*	TGTTTTTTATAAATGAAATAAGGTATACCGGTTCT
ABC1690	*sigA*	ABC2210	*spoIIIE*	GGTGGGCGCCGAAAACATGGTATGATATAATAACA
ABC1690	*sigA*	ABC1783	*drm*	GGTTGAATACAGTCGAGAGAAGGGAAATTATGAAG
ABC1690	*sigA*	ABC1775	*mtnW*	GGTAGCGCTTTATTTGTAATGTTGATATAAAAATC
ABC1690	*sigA*	ABC1746	*fumC*	CATTAAGTTTTCCCGGTAATTCTGGTATGATAGAT
ABC1690	*sigA*	ABC2199	*recA*	TATTGGCAAACGAAGCAAAACACGTTATGATAGAC
ABC1690	*sigA*	ABC2157	*citB*	TAAAAAAGATAAGTTTAAATTATAAAAACAGCTTT
ABC1690	*sigA*	ABC2151	*nadE*	AATGTTGTAACCTCCAGAATACTACTTTCATTTGG
ABC1690	*sigA*	ABC2114	*sucA*	TTTTGAGATACAGGCTGGAAAATGGTAAAATAGGT
ABC1690	*sigA*	ABC1666	*dra*	AGATTAGTATGTATCCTTTTCTGTTTTGCTGTTAG
ABC1690	*sigA*	ABC1660	*dnaJ*	ATTGCAACCGCGCTTTATATGGTGGTATGATGAAC
ABC1690	*sigA*	ABC1657	*hrcA*	CATTGACAAATGGACACCTGTTTGATAGTGTATCA
ABC1690	*sigA*	ABC1655	*lepA*	TATTGAAGCGCCGCCATTGGCTTGCTATAATAAGG
ABC1690	*sigA*	ABC1621	*nasD*	TTTAGCAAAGCTTGGCGAGAAAGGCTATGATATTC
ABC1690	*sigA*	ABC2078		ATGTTGATATGGATTTTTAGTAAAACAAAAGTAGA
ABC1690	*sigA*	ABC2035	*gltA*	AAGATAATACTTGCTTTAAAGGAACTCATCGTTAA
ABC1690	*sigA*	ABC1584		CTATCAATCCAGTATGAAAACATGCTATAATGTCA
ABC1690	*sigA*	ABC1547	*nadB*	ATAATCAAATACATTTATATCCACTGTTCTGTTGA
ABC1690	*sigA*	ABC1546	*nifS*	AGTTGTCTTGTCACCTATATTTACATAAACTAATA
ABC1690	*sigA*	ABC1537	*pbpF*	CTGTGCGCGCTTTTGTATTTTCCGGTATGATAATT
ABC1690	*sigA*	ABC1523	*cssR*	GTTCGCGATTTAAAACGAAATTATGTAAAATAAAG
ABC1690	*sigA*	ABC1498		CTTCTCTGTTATTGTGAAAAAATGCTATGATAAGG
ABC1690	*sigA*	ABC1491	*nfrA1*	ATCATGTGATGGTAATAGTCGTTTTTTTACTTTGT
ABC1690	*sigA*	ABC1473	*pbpD*	TCTGTCCGTCCCAACAAAACAGTGGTATACTAAAC
ABC1690	*sigA*	ABC1453		GGAAAAATATATCATTTTTTCTCCTTTTTCCTTCT
ABC1690	*sigA*	ABC1452	*gabD*	CAAATCATATGTAAATAATTTAGCTATTAAGTTAA
ABC1690	*sigA*	ABC1443	*xpt*	TCTTTTTATTTTTGGTTAGATTCTTTATAATAGGC
ABC1780	*fur*	ABC0094		TCTTTTCACTCTTACTAAAAGTAATAGGATATCTCAC
ABC1780	*fur*	ABC3970		TGTTTATTTTGCAATTCAAAACGGTTCTTGGTTAAAT
ABC1780	*fur*	ABC3488		ATATGCTATATTTATTGATAATGATTATCAATTAATT
ABC1780	*fur*	ABC3476		CACAACCACTATTACTAAAAGTAAACGTTTATTTGTA
ABC1780	*fur*	ABC2925		CATGTATAATGAGAAAGGATTTCATTTTCACTTGGGG
ABC1780	*fur*	ABC2895		TGCATTAACTATTACTCATAGTAATCGACATCCCGTT
ABC1780	*fur*	ABC2609		CCGATTAACACTTGGCATAAGCGTTAGTAAACCATTT
ABC1780	*fur*	ABC1501		TTTAGAGAGAGTCCTTCCTTATCTTTATCAACTATGG
ABC1780	*fur*	ABC1253		AAAATAAGCTATGATAGGAAAATAATATAAAATAAGA
ABC1780	*fur*	ABC1003		AGAATGTAGTACTAAAAATGGTAATGCATTAGACCAC
ABC1803		ABC4043	*citA*	GAAGTTTTCATATTT
ABC1803		ABC2715	*citZ*	AGGCATTGCTTATTT
ABC1803		ABC1805	*mmgD*	ACAATTTTCTTTTGT
ABC1803		ABC2157	*citB*	AGTATTATGCAATAC

ABC1835	*resD*	ABC1621	*nasD*	TTGGATTTTTTTAAAAA
ABC1835	*resD*	ABC0350	*hmp*	ATATATATTTTATGAAT
ABC2167	*lexA*	ABC1294		ATACCGGGAAAACAAACCGGAA
ABC2167	*lexA*	ABC0752		AAAATTGAAGATGAGTTGGGAA
ABC2167	*lexA*	ABC3826		ATGGCTTGCAGATAAGCAAAAA
ABC2167	*lexA*	ABC3418		GTATCTTAACATATATTCGTTA
ABC2167	*lexA*	ABC3059	*uvrB*	AAATCATGCATACAAGCAATTC
ABC2167	*lexA*	ABC2509		TCGGTTTGAAAGTGAACCAAAT
ABC2167	*lexA*	ABC1989	*parE*	TATACAGAACAAACATTCGTAT
ABC2167	*lexA*	ABC2199	*recA*	AAAGACGAATAAATGTTCGTAA
ABC2167	*lexA*	ABC2167	*lexA*	ATCTCTTACAAACAAGACAATA
ABC2244	*sigD*	ABC3699	*hag*	TAAGATGAATATAGCCGATTTTTTACAGCTAATAACTGAT
ABC2244	*sigD*	ABC4049	*mcpB*	CCCTTTAAAAATTGGTAACGGATGACGATAATAACAGTGT
ABC2244	*sigD*	ABC3479	*hemAT*	AAGCTAAACGGAAAAAACTGGCTGCCGATAAAAAAGAAAA
ABC2244	*sigD*	ABC1983	*motA*	ACATTTAAAAGAAGGACGAGAATGCCAAAATTAAAAGATG
ABC2244	*sigD*	ABC1818		AATAAATTTAGTAACTGTATGCAGACTACTACGAAATTTA
ABC2244	*sigD*	ABC1491	*nfrA1*	TATTTCCCAACTTTAGAAGATGTGCCGAAACAAGCGATTA
ABC2244	*sigD*	ABC1452	*gabD*	TGTAAATAATTTAGCTATTAAGTTAAGAAAGACTTTTTTA
ABC2244	*sigD*	ABC0980	*gbsA*	AATTTTTTTTATTGCCGAAGAGAACTTATAATACTCAAAG
ABC2244	*sigD*	ABC0965		AAAAGAAGATATAATCGAATGCTTCCAATTTGAACAAAAT
ABC2244	*sigD*	ABC0761	*aprE*	TAACTTAATGTTAATAATTGTTTCCCAATAGGCAAATCTT
ABC2244	*sigD*	ABC1114		TTCCTGAGTTACTAGAAAGACATGACAATTTAAAAGGAAT
ABC2244	*sigD*	ABC0422	*iolA*	AACCAACAAATGATCAAAAAGTAACCGGTAATGAAAGCGG
ABC2244	*sigD*	ABC3965		CTCTTTACTTTGCTTACGATTTTGCCGATGTAAAAGAAAA
ABC2244	*sigD*	ABC3810		GAACACGGATATAGCTAGTCGAACGCTGAATAAACGTCGC
ABC2273	*codY*	ABC2645	*ilvB*	TCTAATAACGATAACACAATGAAGA
ABC2273	*codY*	ABC0776	*dppA*	ACGGTCTTTTCAACTGTGCTAAGAA
ABC2273	*codY*	ABC3699	*hag*	AGAAAAGGGACAACTCTTATTTAAA
ABC2304		ABC2547	*fabH1*	ATAATCCTAGTCCATGATTAAAATTTTTGTGCAGAAAATTTCATCCTTC
ABC2304		ABC2517	*fabI*	TAGTACCTTGTTAGGCATTATGATAAACTATTAGTACCTGATAATAAAA
ABC2304		ABC2304		AATCCTACGCTGAAGATCTATATACTAGTCTTAGTACCTAGTCTTAATT
ABC2338	*pyrR*	ABC2338	*pyrR*	GAAAACGTCCTTTAAACGAGTCCTGTGAGGCTGGCAAGGAGACGGAATG GCATA
ABC2350	*sigG*	ABC0084	*spoVT*	GGTGAATAGTTCATGGAAACGGAAGAATACTACATT
ABC2350	*sigG*	ABC3969	*katX*	AGATTTGTTATAGGTCAACTGTGTACACATTCGTCT
ABC2350	*sigG*	ABC3946		AAGGTTCAAACTGGCGAATTTCTGAGAAATACAATG
ABC2350	*sigG*	ABC3899	*pbpG*	AAGGTTTAAAAAAACCGCTTTTTTCCCAAACTGTAT
ABC2350	*sigG*	ABC3837	*gerAA*	TTTGTATGAATACGACCCTCCTTCGCATACTAACAG
ABC2350	*sigG*	ABC3703		CATGTTTAAACACGCCCTTTTTATGTTTTTTTGGCA
ABC2350	*sigG*	ABC3491		GCCGTATAAAGAACCACTCTTAGGAAATAGTAAGCC
ABC2350	*sigG*	ABC2885		TTGTATAGCTTTATTGCTTTTGCTCCATAATAACAG
ABC2350	*sigG*	ABC2759		AAAAATGCAAAAGTTAAGTGAAAGATCAAAATGGTA
ABC2350	*sigG*	ABC3062		GCGAAAAAAAACACTTACGTTTTAGCGAAAAACTGTC
ABC2350	*sigG*	ABC2488	*splB*	AGTAATGAAATTCACCCGAGATACACATTCTATAGG
ABC2350	*sigG*	ABC2458	*spoIVB*	CATGGTTATAAATATGGCCGTAGGAGGCAACATTAA
ABC2350	*sigG*	ABC1870	*sleB*	GGCAATAAAATTACATCCTTCGAATATTCTTTTGTC
ABC2350	*sigG*	ABC1792	*dacF*	AATGTATAACCTTTCCAGTAAATGGAGACACTAGCA
ABC2350	*sigG*	ABC1652	*gpr*	GGCGAATGTTTTTTCGCGCGTGAGAGACAATACAAG
ABC2350	*sigG*	ABC2021		CATGCATAAGCACAGCTAATTATTGCACACTAACAG
ABC2350	*sigG*	ABC2003		CATGCATAAAAGAAGTGGCTCAACACAGACTAATGG
ABC2350	*sigG*	ABC1537	*pbpF*	AGGTATATTATACGATCGAGTTCACATTGTACGAAT
ABC2350	*sigG*	ABC1473	*pbpD*	GACGCATGAACATGCTACTTTTTGAAAATTGAATCT
ABC2350	*sigG*	ABC1283		GCTATTACTAAGGTTAATTCCTTATTCAAATCAACA
ABC2350	*sigG*	ABC0657	*gerKA*	AAACATAAATCGACATATTTCTGCACAAACCAAACA
ABC2351	*sigE*	ABC2800		AGGGTATTTGATCTACTAAAAGCTGTAAACAAATATTCTT
ABC2351	*sigE*	ABC3113		TTTTCATGTCCGCCCTTCTTTCCACATAGGTATAGAGTAT
ABC2351	*sigE*	ABC2653	*gerM*	AAAATAAGTTCCATATTTGTTCGACCAGATTAGACTCTCT
ABC2351	*sigE*	ABC2580	*asnB*	GTTTTAACCCAAATACACAAATATGGGATTTGGTACTTTT
ABC2351	*sigE*	ABC2394	*ctaA*	TGCCAAAGTAGTATAAGGAAAACAAGTGCATTTACAGTGT

ABC2351	*sigE*	ABC2356	*spoVE*	TGGACATGTTTGCCGCTTCGGCTTCATATAGTAGTGTTAG
ABC2351	*sigE*	ABC1882	*spoIVA*	CAGGTCTAATCTGCCCCTTCCATCCATATAGTTACATTGG
ABC2351	*sigE*	ABC1828	*dacB*	CCCGTCATATTCATAGCTTGTTCGCATAAAGTGGTACTAA
ABC2351	*sigE*	ABC2181	*spoVK*	AAGTACTTGCTCGAACAATTGCTCCATATAGTAAAAGGGG
ABC2351	*sigE*	ABC2153		TTATCCACGAAGGGCATTCCCATTAATAGAAAGGAAGGAA
ABC2351	*sigE*	ABC1653	*spoIIP*	CTGTTCTATGGCCTCTCCTTGATACATAAGATGGAAGGAC
ABC2351	*sigE*	ABC1563	*spoVB*	CTTGTCATGTCCTGTCCATTGCCGCATAGACATGTATAAA
ABC2351	*sigE*	ABC1506		TTGGCTTTTTCCTGAACGGTCGCTCAGATTATGAAAAGAA
ABC2351	*sigE*	ABC1500	*spoVR*	CGTTCATGTTCCCCCGCTGCTGTTCATAGAATAGGAAGAC
ABC2351	*sigE*	ABC1477		TTGGCAACTCTTGCTCTTGTCTTACATAGGATAAGAAAAG
ABC2351	*sigE*	ABC1474	*prkA*	GTAGCATGTTCATGCGTCATTCGGCATAACGTTAAGTAAT
ABC2351	*sigE*	ABC0806		TCTGCATATTGCCGTCATTCGGCTCATATAGTTTACAGCA
ABC2351	*sigE*	ABC1185		AGGTCGTACAACCCGTTGTTCCTGAATATAGTAGAACTGG
ABC2351	*sigE*	ABC0202		AAAAATAAGTATAGAAGGCGCCTGTTTGTATAGAACAGAA
ABC2351	*sigE*	ABC3952	*phoB*	ACAAACGCACACAAACGCCGGTAGCGCTCTTTTTCCGCGT
ABC2351	*sigE*	ABC3846	*spoIID*	AATGCCTTTGACATAAACTCTTGGCCAAATACGTACATCT
ABC2351	*sigE*	ABC4109		TACTCATAAATGAAGCTCACCTCCCATAGTAGTAACAGTA
ABC2351	*sigE*	ABC4099		TGAAATACACGCTTACGTATATTCCAAGAAAAAAACGTAA
ABC2351	*sigE*	ABC4048		TCATAATTTGTTCTTTAATCGTTTCATAGACAAAATCGGT
ABC2457	*spo0A*	ABC2352	*spoIIGA*	TCGCAAACGGTTTT
ABC2457	*spo0A*	ABC0101	*spoIIE*	CTTCATGCAGCTTT
ABC2460	*ahrC*	ABC1818		AAAGGTAAAAGAATTA
ABC2460	*ahrC*	ABC1452	*gabD*	CAGAATAAAAAAGCCT
ABC2460	*ahrC*	ABC0980	*gbsA*	TTTAATATATATTAAA
ABC2460	*ahrC*	ABC0965		ATGAATTTAATTTTTA
ABC2460	*ahrC*	ABC1114		AACCTTAAGATCATTC
ABC2460	*ahrC*	ABC0422	*iolA*	CACAAAAAACTATGTA
ABC2460	*ahrC*	ABC0017	*rocD*	AAAATAAAAAAAAGTT
ABC2460	*ahrC*	ABC3810		TCTTATAATATTAGTA
ABC2460	*ahrC*	ABC2558	*argC*	TAAATAAAAATAAGTT
ABC2487	*mntR*	ABC3961	*mntA*	AAGTTGCATTAGAGAAACT
ABC2672		ABC2990		TTTTTTTAACGTTAAAAATGAATGAGTGTTCATTCGGTT
ABC2672		ABC2673	*lcfA*	AATTTTATACATTAGTTGTGGATTATTTGCCATTGCTGG
ABC2672		ABC2672		GTACTATACTAAATAACATGAATGAACATTCATTCATGT
ABC2672		ABC1993		ATACTTTTTTCTAACGACTGAATTTAATCTCGTTTAACA
ABC2672		ABC0589		AATTAGCATTGCTTAGTAACAAAAAAAGCGTTGATGTAT
ABC2672		ABC3892		TAGACTGACTTACGAGTAAGTATTCCTAACACAATATGA
ABC2711	*phoP*	ABC2591	*pstS*	AAAACCTTAACATTAGCCATAAA
ABC2711	*phoP*	ABC4099		ACAATAAAGATTTACTTCCTAGA
ABC2711	*phoP*	ABC1835	*resD*	ACGCTAATAATACAAATTGGAAC
ABC2711	*phoP*	ABC3952	*phoB*	ACATGAAGATAAAAATGTATCAA
ABC2711	*phoP*	ABC3101	*tagD*	GCATCTTGAAAAAAGTAATTACG
ABC2764	*ccpA*	ABC2740	*ackA*	ACATACGCGAAAAA
ABC2764	*ccpA*	ABC2645	*ilvB*	TGTTAAAGCTTAAA
ABC2764	*ccpA*	ABC4043	*citA*	AGGAAACGCCTTCA
ABC2764	*ccpA*	ABC3549	*rbsR*	TGTAACCGGTTACA
ABC2764	*ccpA*	ABC3383	*glpF*	TGTAAACGCTTCCT
ABC2764	*ccpA*	ABC3262		CTGTTCGCAATTTT
ABC2764	*ccpA*	ABC3255		TTTAACCGTTTTAT
ABC2764	*ccpA*	ABC2788	*acoR*	AGAAAACGCTTCCT
ABC2764	*ccpA*	ABC2761	*acuA*	TCCTTCGCGAAAGT
ABC2764	*ccpA*	ABC2760	*acsA*	TGAAAGCGCTTCCT
ABC2764	*ccpA*	ABC2715	*citZ*	CGTAACCGCTTTAC
ABC2764	*ccpA*	ABC3120		AGAAAACGCTTTAA
ABC2764	*ccpA*	ABC2673	*lcfA*	TGTAAGCGATTACA
ABC2764	*ccpA*	ABC3025	*sigL*	TCTTTCGCAAATGT
ABC2764	*ccpA*	ABC2454	*bkdR*	ATATACCGTTGACA

ABC2764	*ccpA*	ABC1818		GGGAAACGCTTTCT
ABC2764	*ccpA*	ABC1805	*mmgD*	ACATTCGCGAAATA
ABC2764	*ccpA*	ABC1666	*dra*	ACATTTTTAACAGA
ABC2764	*ccpA*	ABC2078		ATTATACGCTTAAA
ABC2764	*ccpA*	ABC1453		ACATTCTCAAATAT
ABC2764	*ccpA*	ABC1452	*gabD*	ACCATCGCCAATGT
ABC2764	*ccpA*	ABC0980	*gbsA*	ATAAAACGTTTAAT
ABC2764	*ccpA*	ABC0965		TGATTACGCTTTTG
ABC2764	*ccpA*	ABC0917	*maeN*	TCTTTCGCGAATAA
ABC2764	*ccpA*	ABC1343		TTATTACGCTTGCA
ABC2764	*ccpA*	ABC1324		TAGTTCGCCAATCT
ABC2764	*ccpA*	ABC1263	*treP*	AACTTCGCAATAGT
ABC2764	*ccpA*	ABC1114		TTTCAACGATTAAA
ABC2764	*ccpA*	ABC0665	*citH*	TCTTTCGCCAAAGT
ABC2764	*ccpA*	ABC0643		AGGAAGCGCTTCCT
ABC2764	*ccpA*	ABC0589		TTCAACCGATTGCA
ABC2764	*ccpA*	ABC0572	*xylA*	TGACACCGCTTACA
ABC2764	*ccpA*	ABC0456		TGAAAGCGCTTAAA
ABC2764	*ccpA*	ABC0453		TGAAAACGCTTAAA
ABC2764	*ccpA*	ABC0422	*iolA*	ACTTTCGCCAAATG
ABC2764	*ccpA*	ABC0352		ACATTTGTAAAAAT
ABC2764	*ccpA*	ABC0100		ACCTTTGCCAATGT
ABC2764	*ccpA*	ABC0016	*rocR*	TGAAACGGTTTTAA
ABC2764	*ccpA*	ABC3910	*eutD*	ACCTTCGCATTAGA
ABC2764	*ccpA*	ABC3908	*gltT*	TGAAGCGGTTTTCA
ABC2764	*ccpA*	ABC3810		ACGTTTGTATAAAT
ABC2764	*ccpA*	ABC3790	*bglP*	ACTTTCTCATCAAT
ABC2788	*acoR*	ABC0422	*iolA*	AATGGAAATATGTTGACCT
ABC2788	*acoR*	ABC0017	*rocD*	GCGTTTTTTTAAAAAGCAA
ABC2788	*acoR*	ABC3810		CCGATTTGTTAACGTGTAG
ABC2788	*acoR*	ABC1862	*gudB*	ACCGTCAAGTCATTATGCA
ABC2788	*acoR*	ABC1818		ACGCTTTTTTCATATTTAA
ABC2788	*acoR*	ABC1452	*gabD*	GAAATTTCGCAAAACGTCA
ABC2788	*acoR*	ABC0980	*gbsA*	AATGGGCATCTATTTTAGG
ABC2788	*acoR*	ABC0965		TGAATTCATTTAAAGGTGA
ABC2788	*acoR*	ABC0326		AGGCAAAAATGCTTTCGCC
ABC2788	*acoR*	ABC0016	*rocR*	AACGAAAAATTTTTTTGCG
ABC2788	*acoR*	ABC2454	*bkdR*	CCTTATTCGTTATACGTAT
ABC3022	*cggR*	ABC3022	*cggR*	GGCCCTGTATTATACAGACGCTCCCTGCAATATACAGGGCTATTTG
ABC3025	*sigL*	ABC1452	*gabD*	AAATCGATAATTCAATTCTTTCTGAAAAAATATCGG
ABC3025	*sigL*	ABC0980	*gbsA*	GTTTCTTGAATCGACGATGTTGGTATAGTTCAAAAG
ABC3025	*sigL*	ABC0965		GTTAATTTGGCACGCATCTTGCATGATTCATCATGA
ABC3025	*sigL*	ABC1114		GGTACGTTGATATACTATATTTGTTGAGTAAGAAAA
ABC3025	*sigL*	ABC0422	*iolA*	ATTGGTTTAAATAACTTTTTCATTACCTTTATACAA
ABC3025	*sigL*	ABC0017	*rocD*	CATCGGTTGGCATGATGTTTGCATATAGAAACAGTA
ABC3025	*sigL*	ABC3810		CAAGTACTAGACATATCCTTTCAAAATGAAAGACAG
ABC3025	*sigL*	ABC1862	*gudB*	TGATTGAAGGCATGGAACTTGTTTCAGAAGACTAAT
ABC3025	*sigL*	ABC1818		GATGAAGGCGCATGGTCTTTGCATCAATGTACGGTT
ABC3086	*degU*	ABC1453		CAAAGAAATTTTTACATTCT
ABC3086	*degU*	ABC1263	*treP*	TTTTAACTTCTTAGAAGTAA
ABC3086	*degU*	ABC0761	*aprE*	TGTTAACTTAATGTTAATAA
ABC3153	*adaA*	ABC3152	*alkA*	AGAATGTAATAGCAAGATAACAAAATCAGTACTGATTATTATGTGA
ABC3374		ABC3141		CGATATCCTTGTGTTGGCTGCAATAGA
ABC3391	*gltC*	ABC2035	*gltA*	ACGGAAAACCTCTA
ABC3391	*gltC*	ABC3391	*gltC*	AACGCGAAAAGAGA
ABC3435		ABC2035	*gltA*	AAGCAAAAATGAGGT
ABC3435		ABC3435		TAGTAAAAATCACTA
ABC3791	*licT*	ABC3790	*bglP*	AGGATTGTTACTGGCCAGCAGGCAAAACCTAAG

ABC4034		ABC3819	*gde*	ATCCTTTTGACGTTTATATTATTAACT
ABC4034		ABC3738	*pucH*	ACTAATTGGCTTGCGTCATCTTTAAAA
ABC4034		ABC3483		AACGGTTACACTAGCTTAGTGATACCA
ABC4094		ABC4094		CAAAAAAATAGTTTGAAGGCGTTGTCAATAGCTATTAACAATTTGCTTCC AGTTTTTAGCAACTTTACGGGCGTTTACGGATCTGGTAGTCCGCTTGTCCC CATATTATTTCT
ABC4099		ABC3101	*tagD*	TATGATAACGTTTGATTCACATCTTAG

Table 1: Binding sequence predictions for *Bacillus cytotoxicus* NVH 391-98. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
Bcer98_0001	*dnaA*	Bcer98_0001	*dnaA*	GATTTTGTGGATAACTT
Bcer98_0041		Bcer98_0267		TGATTTTGTGCTTGTAAAAAAAATCACAAATTTTTTTGTAAGCTA ATAAAC
Bcer98_0041		Bcer98_0254		AAAAAATATGCTTGTTTTTTTCAAATTGAATATTTTAACAAGCAA AAATTC
Bcer98_0041		Bcer98_3971		ATTAAACACGAACATTCTTAATTAAATTAATAATTTAATTCGATA TTAAAT
Bcer98_0041		Bcer98_3835	*glyA*	TGAAAATATGAAGATTTAAAAGAATTACAACTTGAATTTTCGTTT TTTCTG
Bcer98_0041		Bcer98_4007		TTCAAGAATGCTTGTTTATACTATTTTGAATTTTTTTACAAGCAAA AATTT
Bcer98_0041		Bcer98_2782		ATTCAAACAGGACAATTAGAAATCATTTCTTCTGTTCGTGCTATTT TTACT
Bcer98_0041		Bcer98_2540		ACCTGCACCAGCTATATTCGAGGAAACATTAAATATTTTACGGAA ATAGTA
Bcer98_0041		Bcer98_1302		AAATTTTCAGCTTGTTTTTTCTATAAAGCTATTATAAACATGCACA AAAAG
Bcer98_0041		Bcer98_0631		AAAAAACACGAATTGTATACATCGTTTTATGACTAATATTCGACA TCAAAA
Bcer98_0088		Bcer98_0526	*aspA*	GAAGGGGATTATATTGAAACAAATGGATTAGTATTCA
Bcer98_0088		Bcer98_3727		AAAAGGGTTTTTTAAGCCGGTGTCGAAATAATATACA
Bcer98_0088		Bcer98_2865		AAAGGGAAATTTCACACAATTGTCGAACAATTCATGT
Bcer98_0248		Bcer98_1484	*ldh*	GTTTTAATAACTTTATCGCTCCTTAACCATATA
Bcer98_0248		Bcer98_1120		AATTATAACATTTAAAAGATTTTTTATTGGAAC
Bcer98_0248		Bcer98_1035		AGGGAAAACACTTTGCAAAGTGTTAAAACAACT
Bcer98_0248		Bcer98_3591	*ldh*	ACATAAAGAATGCTCAAAATTTCACAATGTATC
Bcer98_0248		Bcer98_3505	*ldh*	GTATTAAACACTTGAAAAAGTGTTAATGAATAC
Bcer98_0248		Bcer98_3464		AATAATTTTCACATCACCATTTCGCATATTCAT
Bcer98_0280		Bcer98_0452		TACGAAAAATTAGAT
Bcer98_0280		Bcer98_0280		ATCAAAAATAATGAT
Bcer98_0280		Bcer98_2203		ATCAAAAAAATTGAT
Bcer98_0304		Bcer98_2229		GTATTTACTTTAAACTTAA
Bcer98_0304		Bcer98_1476		CCGATGTATTCATACGAAA
Bcer98_0304		Bcer98_0869	*rocD*	AACGCAATAATAAATTGCA
Bcer98_0304		Bcer98_1214		AATACATCATTATTTTATT
Bcer98_0304		Bcer98_0684		ACATTTTATTTGAACTTTC
Bcer98_0304		Bcer98_0303		ATTATAGAATTCATTTGCG
Bcer98_0304		Bcer98_0289		AGTGCAAAAAGGTGTTGGG
Bcer98_0304		Bcer98_2861		CCGTTTCATTGCAACGTAT
Bcer98_0304		Bcer98_1687		GTGACTAATTTCAAAGTGA
Bcer98_0304		Bcer98_0762		CATGGAAATCTATATTTCG
Bcer98_0459		Bcer98_3195		GCTTTGATATAAATATTATAT
Bcer98_0459		Bcer98_1744		ACCATTAATAAAAGGACTGTC
Bcer98_0459		Bcer98_3182	*hemA*	ATTATTTTATAATGATTATAA
Bcer98_0459		Bcer98_0904	*spxA*	TCTATTGTATAATGATTATAG
Bcer98_0459		Bcer98_0518		AACATTTAGCAAATTTTTCTT
Bcer98_0459		Bcer98_0459		AATATTCTTAATATTTTATTA
Bcer98_0459		Bcer98_0397		ATTATTGATAATTTTAATTAA
Bcer98_0459		Bcer98_0315		AATCTTAATAATAGCAATTAA
Bcer98_0805		Bcer98_0806		GACGGAGAAAAGAAGAGATC
Bcer98_0812		Bcer98_1737		AAAAATATTAAACTTTATTA
Bcer98_0812		Bcer98_0531		ATAAAGTTTTTAAAAGAAAG
Bcer98_0812		Bcer98_2213		AATAATTGTATATTAAGTTA
Bcer98_0812		Bcer98_1599		ATAAATTTAAACATTAATAT

Bcer98_0812		Bcer98_0515		ATGAAGTTATATTTATGAAA
Bcer98_0812		Bcer98_3622		AAGAAAATTATTTCTCAATA
Bcer98_1200		Bcer98_0208		CCATATTTTTTAACAAT
Bcer98_1200		Bcer98_3464		TAAGATTTTTTCAGATA
Bcer98_1200		Bcer98_3398		GTAAGTTTTTAAGAAGT
Bcer98_1200		Bcer98_1174		AACTGCTTTTACCGAAT
Bcer98_1468		Bcer98_1469		AATGAACATAATTTCAAAAGCGTATTTACATGTGTTCAA
Bcer98_2344		Bcer98_1548		TTTGGAGAACAAAAGATTCAAA
Bcer98_2344		Bcer98_0794		TTTCCCAGCAAACAATCACTTT
Bcer98_2344		Bcer98_0525		GTCCCTTTTTTAAAAACAAAAA
Bcer98_2344		Bcer98_3710		ATGGCTTATAAACAAGCAAAAG
Bcer98_2344		Bcer98_3288	*dnaE*	TCTTTTTTCAAACAAGAGAATA
Bcer98_2344		Bcer98_3219	*uvrC*	CTTTCTTGCATACAACCCCTCA
Bcer98_2344		Bcer98_3138	*ruvA*	ATCTCTTGTAAACAACCCCATC
Bcer98_2344		Bcer98_2431	*recA*	AAAATCGAATAAATGTTCGATT
Bcer98_2344		Bcer98_2344		CAAGCTTGAATACAAACATCTT
Bcer98_2344		Bcer98_2247		TATCTTTGTATACAAGACTAAA
Bcer98_2344		Bcer98_1678		CTACTTTTCAAACAAGGGAAGA
Bcer98_2480		Bcer98_1120		AACAAAATTATAACATTTAAAAGAT
Bcer98_2480		Bcer98_3195		CAAAACGAAACTATATTTATAATAT
Bcer98_2480		Bcer98_1744		GTCTAATTTTTTAAACCAATAAACA
Bcer98_2506		Bcer98_0202		GATTTGCTTTTCTTAAATTAATACCTAATTTTATTACCAACTATTAAAA
Bcer98_2506		Bcer98_1823		ATTATCATGGACCATAATTAAAAACCTTAATAAGTAAAAATTAGGTTAA
Bcer98_2506		Bcer98_2187		GATTGAATTAACATAAAAATTATAATACAATTTATATTAAGTTTTATAA
Bcer98_2506		Bcer98_0920		TTCAACATGAAATATTAGTATAATCGCTTAAAATGTATTAAACGTTTAC
Bcer98_2506		Bcer98_0890		TCTAGACAAATAAACATTTATGTTTTAGAATTAGTACCACGTCATAATA
Bcer98_2540		Bcer98_2540		ATAATGATCCTTTAACACAGCCCCGTGAGGTTGAGAAGGTAACGGTTTGAAATA
Bcer98_2553		Bcer98_1555		GCGGTCTATTCTATAAAGCTTGTACATAAGTTGAAATAGA
Bcer98_2553		Bcer98_1519		CTGTATAAAACTCTTCAAATTTCTCATGTAATGTAAAGAA
Bcer98_2553		Bcer98_1407		AAAAAATGTTAGATATGTCTAACATTTCAAATCTGTTTTA
Bcer98_2553		Bcer98_0872		AAGATAAGTCGAAAACTTGCCTCATTTCAATATGCCTATA
Bcer98_2553		Bcer98_1232		AAGTCATGAACCTTTCTTTTGGCGAATAAGTTCTAAAAGA
Bcer98_2553		Bcer98_0704		TCGTTCTATTTCGCATATTTCTTTCATAAATTATGGATAG
Bcer98_2553		Bcer98_0646		TTTTCATGAAAGCTTGTTTCCCTTCATACACTTTCTTAAG
Bcer98_2553		Bcer98_0473		CAAAAATATAGCATACTCTCATGCCCGATGTTTATCGCTA
Bcer98_2553		Bcer98_0472		AAAGAATGTTTAGGGACATCCTCGCATACCTTTTATAATA
Bcer98_2553		Bcer98_0144		TCGTTCTCTTTTTCATTCTGATTGCATATATGTAACTGAA
Bcer98_2553		Bcer98_3805		TTGTTCTAAAAATCTCCCATTCCCCATAGAAATAAAATGA
Bcer98_2553		Bcer98_4006		GGTGAGAGTCCTATAATCCACCCAGATGAATTATAGGAAA
Bcer98_2553		Bcer98_3504		AGAGAGTTTACAATACTTTTTTGCCTATAAATAAAGTTTT
Bcer98_2553		Bcer98_3280		AAATAATGAACCTGTTTCCAAACAAAAATACTGTTAAACA
Bcer98_2553		Bcer98_3274		TAATAAATATACCTTCATCATTTGCATATACACGTTCAAA
Bcer98_2553		Bcer98_3203		AAATATAGTATCATTCTACAATTTACTCCCACTTCTTGTT
Bcer98_2553		Bcer98_2722		GTAGATTTTTAAATACTCCTATGTATTCTCTCCCCATTCA
Bcer98_2553		Bcer98_3129		GCGTTAATATATCATTTACTAATTCATATAATCTAACATT
Bcer98_2553		Bcer98_2648		AGAAACGATAAAATAATCAAAAATATATAATAATCCTGAC
Bcer98_2553		Bcer98_2631		TGGTCTAAATTTACTATGTATCCACGTATACTGAAATGAA
Bcer98_2553		Bcer98_3069		TATACAAATTATAACGATAATTTACATAACTTAACAAAAT
Bcer98_2553		Bcer98_2387		ATAAAAAATTACCTAAACCATCCAAATTTTTAACTATGTT
Bcer98_2553		Bcer98_1802		TAGGAATTTTTTTGTGCATAAGTACAGATTCTGTTTTAAA
Bcer98_2553		Bcer98_2256		TAATCATAATACAGGAGAATCAAAAATAAAATAAAAACGA
Bcer98_2553		Bcer98_1727		CGTCGTTTTCGTATACTCGCATATAAATGTGAATAATTTC
Bcer98_2553		Bcer98_1716		AAAAACTGTATAATAAATCAACATGTTTGTATGTATTGGC
Bcer98_2553		Bcer98_1652		AAACGTTATAGTATACTGTTTTCACCTCTTTACACTCACG
Bcer98_2684		Bcer98_3278		TGTATTTTGTTAAAA

Bcer98_2684		Bcer98_2264		ATTATTTACTTATTT
Bcer98_2684		Bcer98_1724		AGTATATTCTTATAT
Bcer98_2771		Bcer98_0823		TATACATAAAGGAGAGCCTTTTTCAAATAATAAATG
Bcer98_2771		Bcer98_1180		TACTTGTAGAATGTCCTTTTCTTGTAAAAAGTATGA
Bcer98_2771		Bcer98_0643		ACCGTATAATATTTCGCTATCCCCAAAAACTAACGA
Bcer98_2771		Bcer98_0597		ACATATATAAATGGGATATGCCTCTTTATATTGACA
Bcer98_2771		Bcer98_0341		CTGATCTACTACGTTTTTTTCGTATATAGTATTCAT
Bcer98_2771		Bcer98_0198		CGGAAAAATAAAGGGCTTTTTATTTAATAAAACATT
Bcer98_2771		Bcer98_3848		AAAAACATATTTTTTAAATGTATCTAAAACTTAAAT
Bcer98_2771		Bcer98_3801		AATGTCTAAAACAGCTTTTTATTAAACAATATCCAA
Bcer98_2771		Bcer98_3659		TTGGATAGATTTTCTAGATATATCGCAAGATAAGGG
Bcer98_2771		Bcer98_3408		CACCATAAACTAACATCATCTTACATAAGTTAAAGA
Bcer98_2771		Bcer98_2895		CATGTTTAAATCATGCGAAATTTGTCTATAAAAAGA
Bcer98_2771		Bcer98_2774		ATGGTATGAAACATATGTAAACTGGAAAAAATAGGA
Bcer98_2771		Bcer98_2769		AATAATAACATCGTAAGTACTAATCGTAATTTTTTA
Bcer98_2771		Bcer98_3090		TACGGGTAACCAATACTTCAATTGAGGTTGCTAAAA
Bcer98_2771		Bcer98_3048		ATTTTTTTGCTGGGTTTACCTAGCAAAAATTTGCAG
Bcer98_2771		Bcer98_1897		TTTACATAAAAGCGCTCCGTTTACAAAACATAACTA
Bcer98_2771		Bcer98_1722		TATAAAATACAAGTTTTTTACTTGTTAACATTGTCT
Bcer98_2771		Bcer98_1642		AATTAAAGATAAATGTATAAAGGGAGAGAATATAAT
Bcer98_2790		Bcer98_0685		GCGATTCAATATTTTAATTTTTTATCTTTTACTTTCT
Bcer98_2790		Bcer98_0359		AGCGTTTACTTTAAGTAATAGTAATTTTTCTCCAATG
Bcer98_2790		Bcer98_0331		ATGTATAACTATTACTAATAGTTATAACTAAGACATT
Bcer98_2790		Bcer98_3539		AGGATAAACATATAAAATATGTAAAGATTAATTAGAT
Bcer98_2790		Bcer98_2409		ATATATAATTAATAACGATAATCATTATCATTATTAA
Bcer98_2790		Bcer98_1781		TTTATCAATTAATAATGTTAAATATTAACCTGTAGAA
Bcer98_2790		Bcer98_1754		TATGTTAACTATTACTAATAGTTATAGCTAAGTTCCT
Bcer98_2861		Bcer98_2860		TTATGCTCAGACGTTTTTTATCGTACGTTATTTA
Bcer98_2865		Bcer98_3195		ATTAAAACACTTAT
Bcer98_2865		Bcer98_2554		TCTAAGACAGATCT
Bcer98_2865		Bcer98_1744		ATTTGACATTATCT
Bcer98_2865		Bcer98_0057		TTTTGACAAAAATC
Bcer98_2868		Bcer98_2229		ATAAATGAAATTTGAA
Bcer98_2868		Bcer98_1476		AAAGTAGATTTAAGTA
Bcer98_2868		Bcer98_0869	*rocD*	ATCGTAAAATTACGGC
Bcer98_2868		Bcer98_0684		TTAATAAAAAAATGTA
Bcer98_2868		Bcer98_0303		ATGAGTATAAAAAAAT
Bcer98_2868		Bcer98_0289		AACGTAAAAATAATTA
Bcer98_2868		Bcer98_2829	*argC*	AATTAAAAAATAAGTA
Bcer98_2897		Bcer98_1463		AATTTGCATTAAGGAAACA
Bcer98_3019		Bcer98_0840		ACTTGATACCCTAATTTTCTTTTAATATAATGTAT
Bcer98_3019		Bcer98_0823		AAAATAATATCGTTGAGAATGATTTTTGAAGCTTC
Bcer98_3019		Bcer98_0806		AAAATAACAAAGATATAAAAATATATCTCCGTACA
Bcer98_3019		Bcer98_1214		ACTTTTCTAAAAGCAATAAAAGGACTATAATAATA
Bcer98_3019		Bcer98_1200		AAAATAATATTCCTTTTATCTTAGTCAGCATATTC
Bcer98_3019		Bcer98_0762		ACTTTGTAAGTCATTCAAATTGAGTTATAATCAAC
Bcer98_3019		Bcer98_0711		TCAATAGTATGGTTTAAAGAGTATATAACTTATGA
Bcer98_3019		Bcer98_0707		TTTCTAAAATCTTCTTGATAAAACAATTTGCTTTT
Bcer98_3019		Bcer98_0706		TATTGACCTAAAAACGCCCTTCAAGTAAACTAAAG
Bcer98_3019		Bcer98_1180		TTTGTAAAATAGCTTTTCAAATGTTTTTGAATTGT
Bcer98_3019		Bcer98_1174		TTTTTATGATATGAGAACTTTATTGTAGGATAAAT
Bcer98_3019		Bcer98_1169		AATTATAAGAGAGTACAATATCTGGTAAAGTAATG
Bcer98_3019		Bcer98_1153		AAAGTAAAATACTAAATAGTAAAATTTCCCTCTGT
Bcer98_3019		Bcer98_1143		AATTCCTTTTTTAAATAAAAAAATATAAAATATAT
Bcer98_3019		Bcer98_1120		ATTTGTTTTAATATTGTAAATTTTCTAAAAAATAA
Bcer98_3019		Bcer98_1114		ATTGTCAAATAATATCATTTTTCGGTACAATTAAA
Bcer98_3019		Bcer98_0684		TGTCAATTATTTTTTTACATAATTGTAAAATAAAC
Bcer98_3019		Bcer98_0603		ATTATGAAATCTTTATAAGTTTTTAAAAATCTTAA

Bcer98_3019		Bcer98_1035		AATGTATAATTAGGAATATGAGAAACAAAATTATA
Bcer98_3019		Bcer98_1012		GTTTTTTTAGGAAATAGGAATGTGTTAAAATGGAG
Bcer98_3019		Bcer98_0592		AATATACATTTTCAGAATATTTTGTTAGAATTAGA
Bcer98_3019		Bcer98_0584		TGTAGCACAACAGAATAAAATGTTTTATAATTTAA
Bcer98_3019		Bcer98_0556		CGTTTACATTCCTGTCACAAAGTTGTATAATTTTT
Bcer98_3019		Bcer98_0526	*aspA*	GTTTGCGCTTTCAAAGAAGGTGCTATATAGTGGAT
Bcer98_3019		Bcer98_0518		TAAACAACATTTTATGTATCAAAAATAAAATTTTT
Bcer98_3019		Bcer98_0509		CGTTTTCATCGATTTGAAATTATGCTAGAATAAGG
Bcer98_3019		Bcer98_0459		ATTTAGCATCTAGGAGGAAACTTTGTAAAATAAGA
Bcer98_3019		Bcer98_0452		TATTTCATAAATCACACACAAATAGTATAATTATT
Bcer98_3019		Bcer98_0427		GCTTTTTATAGAATCATTTGGAAGATATAATATGT
Bcer98_3019		Bcer98_0426		TGTATAATATAGAAGGTTTACTAAGATATTTTTCG
Bcer98_3019		Bcer98_0397		CGTATCATAAAAAGAAAACATGTGATATAATATAT
Bcer98_3019		Bcer98_0315		AAAATGATATTATTTCAATTGTCTTTGTAAGGTAA
Bcer98_3019		Bcer98_0303		TATAGACTTATTTTTTATGTTGAAATATAAAGAAA
Bcer98_3019		Bcer98_0289		AAAATAATATAAAATTTTTAAATAATTAAAAATTT
Bcer98_3019		Bcer98_0280		TCTTGAAATAGCTCTTAATACATATGATAGTGGAA
Bcer98_3019		Bcer98_0267		GAATGACTTAAAATTCTTTTATATTAATAATAATT
Bcer98_3019		Bcer98_0253	*guaA*	AAAATAGCATTGGCTTTATATTAAAGACATCTTAA
Bcer98_3019		Bcer98_0208		AGCCTAAAATATACTAAAAATTGTTAAACACTTGG
Bcer98_3019		Bcer98_0057		TGTTGACTTTGAAATGAATTTGATATAAGATACTA
Bcer98_3019		Bcer98_0044	*glmU*	GTAAACCATTTTATAAGAATTATGGTAAAATTTAA
Bcer98_3019		Bcer98_0017		TTTCATAAAATGCAGAGTAATGTGGTTTAATAAAA
Bcer98_3019		Bcer98_0001	*dnaA*	CATTGCTATGGCTACTTTTTTTTGATATTATAGTT
Bcer98_3019		Bcer98_3965		CAAATAGTATTAATTCTTAAGAGGAATTCACTTAT
Bcer98_3019		Bcer98_3925		AATTGACATTAGAAAATAATTGCTATAAAATCATT
Bcer98_3019		Bcer98_3908	*eutD*	AGTTTAAAATATATGTGAAAAGAAGTATTCTTAAA
Bcer98_3019		Bcer98_3875		CTTTTTTATATTTAAATATTTTTGAAACATTATTT
Bcer98_3019		Bcer98_3866		GCTTTTTGAATATCATGAAATAAACTATGATATAA
Bcer98_3019		Bcer98_3859	*pyrG*	AATTTTCTTTTTTGTTTTTTGTTTAGAAAATAAAA
Bcer98_3019		Bcer98_3835	*glyA*	TCTTTTATGATATAGAAAGTCATGTTAGAATGTAG
Bcer98_3019		Bcer98_3796		AAATGATAAAAAGGGAAATATATCGTATAATTAGG
Bcer98_3019		Bcer98_3726		AGTATCATATTTTTTAGCTCAATAAAATATATTTC
Bcer98_3019		Bcer98_3716		AAGGTAATATCTTTTTGAAATTATTATTCAAATTA
Bcer98_3019		Bcer98_3694	*clpP*	GTTTGACCTTTATTGACCATAATTGTATTATAAGA
Bcer98_3019		Bcer98_3683		AGTTGAATAAGTTTCTATCTCCTGTTACAATAATA
Bcer98_3019		Bcer98_4007		AAATTGACTTCTCTTTTCATTTTGGTACAATTATA
Bcer98_3019		Bcer98_4006		ATTTTACAAATGAGGGTAGTTATTTTAAAATAGAC
Bcer98_3019		Bcer98_3597		ATATTTATTTCAGAAAATTTTATCATTAAATATAT
Bcer98_3019		Bcer98_3591	*ldh*	TGTTTTCTATATTTGAAATCTTTATAATAATAAGT
Bcer98_3019		Bcer98_3564		AATATAATACGGACTTAAAAAGACCTTACACCTCA
Bcer98_3019		Bcer98_3529		ATTAAAGAAGATGTTATGAATCGTTTATAATGAAA
Bcer98_3019		Bcer98_3505	*ldh*	GGAGTAATAAAGTGGAGTAAAAATAACATAGTTTA
Bcer98_3019		Bcer98_3494		GAATTCACATCTTTTTTCGTAATATATCACAGTTG
Bcer98_3019		Bcer98_3464		TAAATCAAATGGTTTTAGATTTTTTGTGTTTTTCT
Bcer98_3019		Bcer98_3438		GTATACATTATTTATATAATTGTGTTATACTTTTG
Bcer98_3019		Bcer98_2985		TTTTGTTGTAAAGTTAAACACATAATAAAATTTCA
Bcer98_3019		Bcer98_2975		ATTTACAATAGTAAGAAAAATAAATTATAATAAGA
Bcer98_3019		Bcer98_2928		TAGGTCATATTGTTTATTCGCTTATATAATATTCT
Bcer98_3019		Bcer98_3399		ATTACAATTCATAATTTACAATTTTTAAAATGTGA
Bcer98_3019		Bcer98_3398		TCATTCATATCTTTCTTCATTCTATTATAATATAT
Bcer98_3019		Bcer98_3351		TATTTATATAAGACTGCTATTTTTTTAAAGTTTAC
Bcer98_3019		Bcer98_3336		TAGATAATATTATTGTAATAAATTTTAAAAGTCAT
Bcer98_3019		Bcer98_3335		TACTGAAAATTTTAAATAATGTTATTATAATAGAT
Bcer98_3019		Bcer98_3333		TTATTAATATAGCTAAACAGCTAACATAAAGTAAA
Bcer98_3019		Bcer98_3327	*rpsD*	AAAATAAAATGCATTTATTTTTGTAATTTCCTCCT
Bcer98_3019		Bcer98_3326		TCCTCCTTTAATGTTTTTATTTACGTAAAATAAAA
Bcer98_3019		Bcer98_3321		CGGGGAATATTGTTTTCTTTGTACTTTAAAATTTT

Bcer98_3019		Bcer98_3320		TGTGTAAATTAACAGATGATGTTGTTATAATGAAT
Bcer98_3019		Bcer98_3317		GGTTCAAAATTTACAGAAAATTCTTTATTATAAAA
Bcer98_3019		Bcer98_3310		TATGTAATTTAAATATACACAAGATTAAACTGAAT
Bcer98_3019		Bcer98_2865		GCTCCGCAAGAGCTGTTTTTTGTTGTAAAATAAAA
Bcer98_3019		Bcer98_2861		AATCTCCTTTTCAGATAATTTTGTATATACTAAAT
Bcer98_3019		Bcer98_2829	*argC*	AATTGACTCTATAAATATACAATATTATAATCATA
Bcer98_3019		Bcer98_3299		TTTGTAAAATATTGAAGGATTATGTTAGAATATAG
Bcer98_3019		Bcer98_3278		CATTCACAAAAGTCAGATAATTGTTTATAATAAGG
Bcer98_3019		Bcer98_3274		TAAGTAAAATTCTATCCATTCAATGTATCAGTTCA
Bcer98_3019		Bcer98_3267		TGTTGCAAAATGAAAATAAACAAAGTATACTAACA
Bcer98_3019		Bcer98_3262	*thrS*	ACATGAATTTGAGCACAATATGAAAAAAAATAAAA
Bcer98_3019		Bcer98_3261	*infC*	TATTGCATGTATTCGTGTTGTTTGTTACAATGTTT
Bcer98_3019		Bcer98_3249	*pheS*	GGTTGCGAATCGTTAAAAAACTTCTTATAATAAGT
Bcer98_3019		Bcer98_3226		GATTGTAACGGAGTTATGGCTATAATATAATGAAA
Bcer98_3019		Bcer98_3217		CTAATAATATATTGTAAAGAGCGAATATTTTTTGT
Bcer98_3019		Bcer98_2786		TATTAAAAAGAAAGCGCAAACATGTTATGATGTAA
Bcer98_3019		Bcer98_2749		GATTGAATCGCTTACAATTGTTTTGTATAATAGGT
Bcer98_3019		Bcer98_2735	*mtnW*	CATTGACAACTTAAGTTATCTCTGTCAAAATTCAG
Bcer98_3019		Bcer98_2734		TCTTTACAAAATCTTGGAAAGGATATATTATTTGT
Bcer98_3019		Bcer98_2733		TGTTTATTATATAGGAAAGGTTCTAAAACATTTCT
Bcer98_3019		Bcer98_2732	*mtnK*	ATTGACAAATGATTTTTATTCTTGTTATAGTCATC
Bcer98_3019		Bcer98_2708		ATGTTTATATTTATCTTTTTAGATTTAAAACTTTG
Bcer98_3019		Bcer98_3195		TTTTGTGAATATATCGATAATTTTATATAACGTTA
Bcer98_3019		Bcer98_3182	*hemA*	TAACGTACAAATAGAGTTATTATTTTATAATGATT
Bcer98_3019		Bcer98_3174	*valS*	GTGATAACATATATTATTTATTGTTAATATATTTA
Bcer98_3019		Bcer98_3146		TGGATATAAATGTATTAAAATTTTATTACTGTTCT
Bcer98_3019		Bcer98_3145		TCTTGTCATTATTTTAAAATTATGTAAATATAGGT
Bcer98_3019		Bcer98_3110		AATTTCTTATAATACTATATTATGCTATAATTTCA
Bcer98_3019		Bcer98_2692		GTATTTCTATAGTCTTTTTTTATTTTTATATTTCT
Bcer98_3019		Bcer98_2684		TACATAATATAAATTTTTCTTTATTTCTACTTTCT
Bcer98_3019		Bcer98_2674		ATTGACATCGTGATTAGAATAGTTGTAAACTAAAA
Bcer98_3019		Bcer98_2648		TTTTCATAATAAAAGTATTTTTCACTAAAATATGT
Bcer98_3019		Bcer98_3087		TAATTAATAAGTAGCGTAACCTTTATTACCTTTTC
Bcer98_3019		Bcer98_3069		TGTTTAATATTGCTATTAAATGTATTGAATTGTTT
Bcer98_3019		Bcer98_3046		ATTTGTCTCTTAATGTGTACTCGCATAAGATAAAA
Bcer98_3019		Bcer98_2554		AGCTCCTTTTTATTTTGAAAACTATTACAATAAGC
Bcer98_3019		Bcer98_2544	*ileS*	ACTTGACGTTTTTCTCATTATTACATATAATTTTA
Bcer98_3019		Bcer98_2540		TATTGACAAAATGAATAAAAACTGCGATACTAACA
Bcer98_3019		Bcer98_2453		CCATCATTCTTATTGCAGAATATGATAAAATTAAC
Bcer98_3019		Bcer98_2445		TTCTTAGTATTCAATCGATTTCGGTAATCACTTCA
Bcer98_3019		Bcer98_2431	*recA*	GTTGGCAAATTAAAATAAAAATAGGTATAATAAGA
Bcer98_3019		Bcer98_2416		AATTGTATTTTTCAGATAAATAACTTATTATAAAA
Bcer98_3019		Bcer98_2409		TCTTGAAATATTTTTGTATGGAATATATAATTAAT
Bcer98_3019		Bcer98_1987		CATTTTCATCTTCCTCAAAAAATGATAAAATAGAT
Bcer98_3019		Bcer98_1908		ATTTTGCTATAGTAAAAATATGGCTAATAAATTGC
Bcer98_3019		Bcer98_2363		AAAGTAAAATTTATAAAATCCTTACATGTATTTTT
Bcer98_3019		Bcer98_2325		GATTGACTTTTCGCTATAATTATCTGAAAATTAAA
Bcer98_3019		Bcer98_2317		ATTTATGCATAATCTTTAAATTTATAATAATAAAA
Bcer98_3019		Bcer98_2308		ACTTGAAGGAATGAAAAATCTTTGTTAAGGTAGGA
Bcer98_3019		Bcer98_1823		TGCCTAATATATTTGTTAAGTATGAAAACAGTTAT
Bcer98_3019		Bcer98_2264		TATTTACTTATTTAGAAAGTTGTAATAAGATATTA
Bcer98_3019		Bcer98_2249		TATGTAACTTAGCTTGAAAATATAACAAAATGACA
Bcer98_3019		Bcer98_2244		TATTTTCAAAAATGATAGAATGGTGTAACATAATA
Bcer98_3019		Bcer98_2229		ATTTGAATTTTCAGTTTTTATTATTTATGATTAAA
Bcer98_3019		Bcer98_2205		TATTCAAAAATTTTCATAGATTTGGTATGATTAAG
Bcer98_3019		Bcer98_2203		GAAATAACATACTTACAAGTCTATAGTACATTTTA
Bcer98_3019		Bcer98_1767		AAAAAAATAAAAATTTTAAAATATAGTGCTGTTCG
Bcer98_3019		Bcer98_1754		TTTATAATAGATGATATTAATATTTTACTATATAA

Bcer98_3019		Bcer98_1744		TTTGACATTATCTAGTCAATTTATCTATTATGTCA
Bcer98_3019		Bcer98_1724		AATTGAAATCTTCTAAAAACTCTGATACATTGGAA
Bcer98_3019		Bcer98_1722		GTTATACTTTTAGATTGCTTTTTGATAAAATTATT
Bcer98_3019		Bcer98_2187		GATTGAATTAACATAAAAATTATAATACAATTTAT
Bcer98_3019		Bcer98_1687		TTGCTAATAACTGTTATATTTTTGATAAATTAGAA
Bcer98_3019		Bcer98_2075		ACTGTACTTTTCGTAATAATTGTTTTATATGATGT
Bcer98_3019		Bcer98_2051		TTTTGTCTTTTTCTGAATTTTCTTTTAATTTTTAA
Bcer98_3019		Bcer98_2000		AGTTGACTTATCGGAAAGGTTTGTATAATATATAG
Bcer98_3019		Bcer98_1567		TATTTATAAAATTTTCTGAATGTTATATCATTACT
Bcer98_3019		Bcer98_1510	*nadE*	AATTGATTAAATCAAAGAGTTTATTTATAATAGCT
Bcer98_3019		Bcer98_1484	*ldh*	GTAATAAAATAATTTTCTTCATACAAAACATTTGC
Bcer98_3019		Bcer98_1476		CAAATATAATTGCATAATGTCTTTTTACAGTTTTT
Bcer98_3019		Bcer98_1469		CTTTTCCATATACAGTTCATTTTAATATAAGAAAA
Bcer98_3019		Bcer98_1430		ACTTGATTCCTCGGAAAATTTAAATTATTATAAAA
Bcer98_3019		Bcer98_0961	*sucA*	TATACTATATTCTGTTATATTGTTGTATAATATAT
Bcer98_3019		Bcer98_0949		TGTGGATTTTCTACCATAATCATGCTAGATTATAA
Bcer98_3019		Bcer98_0906		ATTTCCATTCTGGAGAATCTTATCATAAAATGAAA
Bcer98_3019		Bcer98_0904	*spxA*	GCTAGACAGTTCATGATAGATTTGATAAAATAAAA
Bcer98_3019		Bcer98_1336		CTTGGAAGGAAAAAATAGTTTTTGCTACAATTTAA
Bcer98_3019		Bcer98_1302		GCTTTTCAAAGTAAAAAAAGTGCGTTATAATCCAT
Bcer98_3019		Bcer98_0890		TCTAGACAAATAAACATTTATGTTTTAGAATTAGT
Bcer98_3019		Bcer98_0885		GATTGTATAAAAAAGGAGATAAGAGTATAATAAAT
Bcer98_3019		Bcer98_0869	*rocD*	GTTTCGCTTTTTCACATTTAAATGATATAATATTT
Bcer98_3023		Bcer98_3438		AATAAATAAATAGTATACATTATTTATATAATTGTGTTATACTTTTGATGGGGATTA
Bcer98_3110		Bcer98_2975		TTCCACAATCACCTAGTTGACTTATAGGATTTCTTGAA
Bcer98_3110		Bcer98_3087		TAGAACAATTAATTATTCATCGCATTGGAAATAATGGA
Bcer98_3110		Bcer98_2066		TCGCTTTAAAACATTGTATTTTTATCGGAATAATGTAA
Bcer98_3110		Bcer98_2000		TCCTCTCAAAACAAAGTTGACTTATCGGAAAGGTTTGT
Bcer98_3147		Bcer98_3320		ACTAACCAGTTCGTATAAATCATCTTTTTAATTAAGTTATACCATTAGCGATAGCATCGGTTCTATACCTAAATCGGATATCAAATCTGTTGCGCCTCTCTAATAAGG
Bcer98_3147		Bcer98_3146		TATTGCCTCCTAATTTACACATTACAATTGTCTTGACACCTATATTTACATAATTTTAAAATAATGACAAGAGTTTTTTCGAAGTTATTCATAACAACGTGGATGGAT
Bcer98_3147		Bcer98_3145		TAGGTAGGTGCAACAATACTTATTGAAGCTTTTTTGAGAACAGTAATAAAATTTTAATACATTTATATCCACAGTTCTGTTAACATTACACATTTAATCCTCCGTTAT
Bcer98_3225		Bcer98_3597		CGACGTATGTCGAAAAATTGAATGAGCATTCATTCAAGA
Bcer98_3225		Bcer98_3226		AGCACTTACCCGTAAGTAAAATATCAATGCTTTTACTTT
Bcer98_3225		Bcer98_2692		CCATTTTACTTAAACATAAGTAAAAACCTCCTCCCCTAT
Bcer98_3225		Bcer98_0840		TAAACTTACAATGAACTATGGGATTAAAAGAAAATTATA
Bcer98_3225		Bcer98_0707		AAAAAGAAAGTAAAAATGTGAAGGAAATGTCTATCTATA
Bcer98_3225		Bcer98_3866		ACAACTGACTTACGAGTGAGTAATATATTACTCCTATCC
Bcer98_3274		Bcer98_0603		ATATTTTTAGTTATAGTTTAAAA
Bcer98_3274		Bcer98_4006		ACAAATTCCTAATAACTGGTATA
Bcer98_3274		Bcer98_2821		ATAATATAAAAAGACTTTTCAAA
Bcer98_3274		Bcer98_3274		ATATTTTATAGAATTCATTTTAA
Bcer98_3274		Bcer98_2341		ACATTCTAAACATTATATACAAT
Bcer98_3274		Bcer98_2205		ACAATTTTTTATAAGTTTTTAAA
Bcer98_3274		Bcer98_1200		CACTTTCCTACTATATTTTTATC
Bcer98_3274		Bcer98_3069		CATAACTTAACAAAATGTTTACG
Bcer98_3342		Bcer98_3310		TACTTTGCAAAAGT
Bcer98_3342		Bcer98_1120		TATTTCGCCAAATT
Bcer98_3342		Bcer98_3336		TCCTTCGCAAAGGT
Bcer98_3342		Bcer98_3335		TGGAAACGCTTCCT
Bcer98_3342		Bcer98_3317		TGAAAAATTTTTCA
Bcer98_3342		Bcer98_2861		TGAAAAAGCTTGCA
Bcer98_3342		Bcer98_3278		AATTTTTCCAATGT
Bcer98_3342		Bcer98_3226		ATGTACCGCTTCCA
Bcer98_3342		Bcer98_2722		TTGTAATGTTTACA

Bcer98_3342		Bcer98_2717		ACTTTTGCAAAATA
Bcer98_3342		Bcer98_2244		TGTAAACAATTTAA
Bcer98_3342		Bcer98_2229		TTACTGCGTTTACA
Bcer98_3342		Bcer98_1724		TGTAAAGGCTTACA
Bcer98_3342		Bcer98_1687		ACTTACGCCGAAAA
Bcer98_3342		Bcer98_1476		ATTTTTGCAATTCT
Bcer98_3342		Bcer98_1469		TGTTCACGCTTTTT
Bcer98_3342		Bcer98_0840		TCCTTCGGAAACGA
Bcer98_3342		Bcer98_0806		TGACACCGCTTTCA
Bcer98_3342		Bcer98_0762		CTAGAACGATTTCA
Bcer98_3342		Bcer98_1153		ACATTAGGAAAAGA
Bcer98_3342		Bcer98_1114		TCAAACAGCTTTCT
Bcer98_3342		Bcer98_0684		TCTTTAGCAAAAGT
Bcer98_3342		Bcer98_1007		AAAAAGTGCTTTAA
Bcer98_3342		Bcer98_0289		ACATTTGCAAAAAG
Bcer98_3342		Bcer98_3908	*eutD*	ATTTAACTCTTCCA
Bcer98_3342		Bcer98_3685		ACCTTACCCACAGA
Bcer98_3342		Bcer98_3464		AGATAATGTTTAAA
Bcer98_3398		Bcer98_3399		TGACACTATCTATAGTGTAGG
Bcer98_3564		Bcer98_3564		GACTCTCACGTTGCGTGATA
Bcer98_3683		Bcer98_3683		CTCATGTGGGACACAAAATGTCACTGCGGGACATAAAGCGACCACG
Bcer98_3685		Bcer98_2229		GTTTTAAATATGTTCGTTCTGGATCAGGCTAATTGC
Bcer98_3685		Bcer98_1476		AATCAATTACGAATGTTTTTGCCTTTTCAATAGCCT
Bcer98_3685		Bcer98_0869	*rocD*	AGAGTATTTACAAACTTCTTGTAATAACAAAAGGGG
Bcer98_3685		Bcer98_1214		AATGAAAAGATTTTCGTTATTTTCCTGATATTATTA
Bcer98_3685		Bcer98_0684		AATAAACTTGAAAGAATACTGTATTGATATAACAAT
Bcer98_3685		Bcer98_0303		CATAAGTTGGCATACATTTTGCAATGGATATAAGAA
Bcer98_3685		Bcer98_0289		CCGGATCCTTTTTAATTTTTGCATTTTTTATTAATTG
Bcer98_3685		Bcer98_2860		AGAAAGTTGGCATGGTATTTGCTTTATACATAGATG
Bcer98_3716		Bcer98_3716		AGGGTTAGTATTTTTTTTGAATACTA
Bcer98_4001		Bcer98_2249		CTCATTAGAAAAATTGGATACATTGAATCGAACTTTTATATTGTTTTACTGTGAAAGAAAACGTTTACTTACATGTTTAAGAGTAAAAAAACAGATCTTATTACTTATTCCTG

Table 1: Binding sequence predictions for *Bacillus halodurans* C-125. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BH0001	*dnaA*	BH0001	*dnaA*	TGGAATAGGTGTGTCCA
BH0062	*purR*	BH0623	*purE*	TTCATTTTTGCTTGTAACTATATATCTTTTGTTTATAACAAGCTCAAAACA
BH0062	*purR*	BH0608		CCTCATTCTGCTTATAATAAGCAACAAAAATTGAATTACAAGCAGAAAACA
BH0062	*purR*	BH0062	*purR*	CTATCTTTGGCATACTTTTACCATATAAATGAAATTTATAAGCCAAAAACG
BH0062	*purR*	BH3765	*glyA*	GAAGAACATGAAAGTTTTCGCTGTTAATTTCAAAATTATGCGGAAATGAAC
BH0062	*purR*	BH4028	*purA*	TATGAACCGAACAAATTCAGGGCAATATCTTACAAATGTTCGGATTTGATT
BH0062	*purR*	BH2541	*pyrR*	ATTAACGTGGTTCGCTCTTAACTTATTCGAGGAAATTACAATCAGGTCACT
BH0062	*purR*	BH2030		ACTGAATCGTCTTGCTGGAAAGAAAAATGCCATCTTTTTCTCGAAAAATTT
BH0062	*purR*	BH1514		GCAAAACACGAATAAATTTTCTGAATTTAATTCATTCATTCGTTTTTTCCT
BH0115	*sigH*	BH1445	*fumC*	GGAACCTTCTTTAATGTAAATAGAGAAAATCACACAA
BH0115	*sigH*	BH1375	*dnaG*	AGAAGATTTTATCGAATCGAGCAGGAATTTAGCAAAT
BH0115	*sigH*	BH3608		AAAGGATTTACAAATGGGTATGTAGAATATATAGAGT
BH0115	*sigH*	BH2773	*spo0A*	TAAAGAAAATTCGACAAAAGGAATAAATTCTTTGTAG
BH0115	*sigH*	BH2559	*ftsA*	AAAGGAAACGGCACTTATTCTGTTGAATAAGTTACGT
BH0263	*sigW*	BH0263	*sigW*	AATCATGAAACATCTTAAACGTTCATTCGTATAAG
BH0263	*sigW*	BH3198		CAAAATGCACAACGAACGTTTTTCCGTAATAAAGG
BH0263	*sigW*	BH1885		AAACTTGAAATAGGGCTAGCTGTATACCGTTATTT
BH0263	*sigW*	BH1356		AAAGTTGAAACTTATTTCTACACATTTCGTAAAAA
BH0408	*mta*	BH1775	*blt*	GGCTATCAACTTGAAGGAGA
BH0408	*mta*	BH0408	*mta*	GACTCTCACGTTACGTGATG
BH0529	*sigB*	BH1980	*katB*	TTTTTTTAACCTGTTTAACTAGGCTAAAGAACCTCTTT
BH0529	*sigB*	BH2312		GTATTGAAAAGGGTTGGCAATTGAAAAAATTAAAAAAA
BH0529	*sigB*	BH2285	*nadE*	GAATTAATCGACGACAATTACTGGAATGGAAGAAAAAA
BH0529	*sigB*	BH2237	*gapN*	TATAAACAAGGAAAAGGAAAAGGCTTTAGAATTTAGGA
BH0529	*sigB*	BH1779		AGTGAAAAAGTAAGAGGAAACGCACTTACCCGACTTGC
BH0529	*sigB*	BH1772		ATTTTTAGAATTTTTAGAATCTAGAATTCATAAGAATA
BH0529	*sigB*	BH1738		TTATTAATCGACTTATTGCATGTGAACAGATTAGGGGA
BH0529	*sigB*	BH2190		GCATTCGAGCTTTTCTGTATGGGGAAATGAATTAGCCG
BH0529	*sigB*	BH2010		AGGTTTTGATCCCTAAAAAAAAGAATTAGAATAAAAGT
BH0529	*sigB*	BH1306	*katX*	AAATTCCTATATACGGGGTTAAAAAGTATCTCGTTTGA
BH0529	*sigB*	BH0800		GATTAAGAAGTTACAGGTGAAAGGGAAAACTAAGTTGC
BH0529	*sigB*	BH1284		AGATTCTCATTGTTAGAAATTGGATAGAAATTGTTAGA
BH0529	*sigB*	BH1242	*relA*	TTTTTAAAACATTTGCAATGATAGAAGTGAATCTTTAC
BH0529	*sigB*	BH0681	*aldA*	TACAAGATAAAAAAAGGAAGAGGTGTATCCTACTTGCA
BH0529	*sigB*	BH0660		AGTTTCATCACGAAAATCGGCAGGAAATTAACAGGATC
BH0529	*sigB*	BH1018	*dps*	ATGTTTAAAACGAATACGTTCGGGTAAATGTAAAATGA
BH0529	*sigB*	BH1005		GATTTATCGGCTATCGAAAACGGGACCTTGCTGGGTAA
BH0529	*sigB*	BH0539	*dhaS*	GCTTAAAACGATTGAATAAGGGGGAAAAAAGAATGCAA
BH0529	*sigB*	BH0106		AGGTTTTAATTCTGAACTTTTGTCTATAATAATAGAAG
BH0529	*sigB*	BH3940		TGATCAATTATCAGTAATGGAGGCAAGAGAATGTTAAC
BH0529	*sigB*	BH3926		GCTTTCATTTTTGTTGTTAACTACAAATCAACAATATC
BH0529	*sigB*	BH3608		AAATTGCTTTAAATGGAGACTAATTTCTGCAAATTTTT
BH0529	*sigB*	BH4033		GCTTCAAAAATTAAAGGTAAGGGCAAAAAAGGAGGCGA
BH0529	*sigB*	BH4027		GTCGTGCGCAAAAACGTGCAAATCCCTTTTATTTTTGC
BH0529	*sigB*	BH3564	*clpP*	TGTTTAATTCATTTCAAAAAAGGTAACGATGGTAAGCA
BH0529	*sigB*	BH3316	*gabD*	TTTTACGAATGTTTGCTAAGCGGGAGATTGATATAATA
BH0529	*sigB*	BH2803		TGTTTGATGTAAAATAAAAGAGGGAATATATTTTTTTG
BH0529	*sigB*	BH2714		TATTCTATATGTATCCTTTTTGGGTACATATATATTGA
BH0529	*sigB*	BH3178		TGTTTAAACAATGGGAGACCGGGAAAGAAGGACACTAG

BH0529	*sigB*	BH3157	*phoP*	GTGTTAGATAGAGGGGATTTGGGGAAACCGTTCAAATA
BH0529	*sigB*	BH3118		AGACTGCATGACATCGGTTTTTGTAGTGCATCATTTTT
BH0529	*sigB*	BH3061		TAAAAAAATAATAAGGTAACAACAACCTCTCTCTGTTA
BH0551		BH3937	*lctE*	ACAATTTAAATGTGAAATATTTCACAAATTAAT
BH0551		BH3061		AAGGTTATCTAGCTCGTTATTCTAAACTTGTGA
BH0700		BH2757	*xylA*	AACTTAGTTTGTCTATTGAATAAACTAAGTATAAG
BH0757		BH2309		TCCAAACCTAGTAGTTTTATTTTACTA
BH0781		BH3232	*bglS*	AATTGTTTATTGAGAAAAGAATCGTTACAAAAA
BH0781		BH3231		AGATGAGAACGACGAAACGAAAGAAAACGAAAA
BH0781		BH0595	*bglP*	AGCTAGCAAATGATAAAGGGCTATAAATCAAAA
BH0781		BH0296		GGATTGTAACTGCTTCGGGCAGGCAAAACCTAA
BH0873	*treR*	BH2216		TAACTGATTGAACATATATGTTCTATTTTATG
BH0951		BH2861	*spxA*	ATGTTTTTGTTAAAATTGTAA
BH0951		BH3048	*hemA*	ATCTTATAATAATAATTCTAA
BH0951		BH1980	*katB*	AGGTTGTCATAAAAATTTCCA
BH0951		BH0951		AATATTTCTAATATTACATTC
BH0951		BH1306	*katX*	CCTCTATTAAAATATGGATAA
BH0951		BH0744		AATTTTAATAATAACAAATTA
BH0992		BH2237	*gapN*	AATCTTAAATCCTTTTTCA
BH0992		BH2010		ACAATGGCTTTAAACCTAA
BH0992		BH0991		TTTGCATATTCATTTGGTA
BH0992		BH0681	*aldA*	TAGGCAATATCCTTTTCCT
BH0992		BH1005		AATTAAAAAGTCTCTTCAT
BH0992		BH0539	*dhaS*	AACGAAAAAAGCATTTGAA
BH0992		BH3943	*rocD*	ACGTTTAATAAAAAAGCAA
BH0992		BH3940		CATGACAAGTTATCGTGCG
BH0992		BH2312		CATTTTATAGTATTTTGCT
BH0992		BH3944	*rocR*	AACGAAAAAATAATTTGCA
BH0992		BH2766		CCAATTTATTAAATCCAGA
BH0992		BH2739		ATATTTTATTTACATGTAA
BH0992		BH1879		GAGTATTCTTAACCGCTAA
BH1091	*glpP*	BH1095	*glpD*	GTCGGAGATTAGGAGAGGCC
BH1091	*glpP*	BH1092	*glpF*	GACGGAGATTTGGAGATCAA
BH1185	*hpr*	BH0855		CTATATATTTATACAAATAG
BH1185	*hpr*	BH0684	*alp*	TATATAAATATTTTTATATA
BH1200		BH2214		TTTCATAATCATCTCGGACGACAAAAA
BH1216		BH1218	*nadB*	ACCAGTTAGAGGTGATAAACAATTCGCTATTTTTTGAGAACAGCGATCAAATTTAAACACATTTATATCTACAGTTCTGTATACATTTACTCCCCACAGATTTAACTC
BH1216		BH1217	*nifS*	CTCAATTTAGACACCCCTCATTTACATATGTCTTGACATCTATATTTACACAAATTTAAACTAGCGACAAGAGTTTTTTATCGCTTAACAAATAGTGGAGATTGACCA
BH1259		BH3271		AAGTTGATAAGGATAGGTTAGTCACCCTTATTGGCTAA
BH1259		BH2178		GCTTCCTTAAGTTTCAACAGTTCCTAAAAATTTTTCTT
BH1259		BH1211		GACAAGAAAACCCGACTATTTTCATCTGATTTCATTAG
BH1344	*hrcA*	BH1344	*hrcA*	AATCGTGAGCTTGATACTCTCACGATT
BH1372		BH3302	*pckA*	TTTTTAATAATAGTATAGACTATTCATTTAAATGTGTTATACTAAATAAAAATCATG
BH1372		BH3149	*gapB*	TATTAGCAGGAAAAGCTTATTTTCACTATATGATAAATAATATTTACTTAATAATCA
BH1376	*sigA*	BH0296		GAGATAATTTGATATCTTGTAAAAAATTTCTTTAT
BH1376	*sigA*	BH0232		ATGAAAGAATAGCTAATAAAGGTGATATTAGTCCA
BH1376	*sigA*	BH0215		AAGAGACTGTTAGCTTTTAAGACGAAAAAATAGGA
BH1376	*sigA*	BH0213		TTACTAAAATAAGTTATAACTTTATTAAAACTGAT
BH1376	*sigA*	BH0126	*rpoB*	AATTGACTGACGATTGCTTTTGTGTTATTGTTATT
BH1376	*sigA*	BH0115	*sigH*	CCTGCGAAAGTCAGCAAGAGAGCTTTATAATGAAA
BH1376	*sigA*	BH0090	*pabB*	ACTTTCCTCATCTATGCAGTTCATGTAAAATAGAG
BH1376	*sigA*	BH0078	*spoIIE*	TATTGACTTTTGTTTGGTCATCATATATATTATTA
BH1376	*sigA*	BH0065	*gcaD*	TCTTGAAAACACAACCGTTTTAAGATATATTCGTA
BH1376	*sigA*	BH0033		TCTTGCCAAGCTGCCTGTGATTTGTTATAGTAATT
BH1376	*sigA*	BH0001	*dnaA*	CCTTGCAACTACTTATTTTTTTTGCTAAGATATTA

BH1376	*sigA*	BH3944	*rocR*	GAATTAGTATTGGATATTTTGCTTTTTTTATTAAAC
BH1376	*sigA*	BH3943	*rocD*	CAAATTATTTTTTCGTTTTATAGGTTATGATTAAG
BH1376	*sigA*	BH3940		TATTTCTCATAGAGACAAATGGTGGTGCGATGAAA
BH1376	*sigA*	BH3937	*lctE*	GCTTTAATTATTGTTTATTTTTTGTTCTAATATGA
BH1376	*sigA*	BH3924	*mmgD*	ATTGGAAAAATTTATGGTTATTCATTAAAATTATA
BH1376	*sigA*	BH3916		GGTATCACATATTAGTAAACAGTCTTGTCAATTTG
BH1376	*sigA*	BH3823	*eutD*	GTCTTAAAATATTTTTTATTTTGAAATTTCTTGAA
BH1376	*sigA*	BH3820		CTCTTTATTTATTAATAAATAAAGATAAAATGTAA
BH1376	*sigA*	BH3812		CATTGAAAGGATAACAATTGAAGTATATAAGAACA
BH1376	*sigA*	BH3811		ACAAGAATATATGAAGTTAACAATAGGAAAGTTAC
BH1376	*sigA*	BH3802		AATACTCATTCATAAAAAGGTATGTTATAATAACA
BH1376	*sigA*	BH3792	*pyrG*	TTTTGTTTTTTTGTTTTAGTTCGGAAATAATGATT
BH1376	*sigA*	BH3765	*glyA*	ACATGAAAGTTTTCGCTGTTAATTTCAAAATTATG
BH1376	*sigA*	BH3727	*rbsR*	TATTGACAAAGATTCGGTTTCATTTTATGATATCT
BH1376	*sigA*	BH3670	*lytR*	AGGTTAACATTTAATAGTTATTGAAATAGATTTTA
BH1376	*sigA*	BH3629	*degS*	GTTTTACAAATGGCAGAAATAATGGTGTAATATTG
BH1376	*sigA*	BH3626	*comFA*	AAAATAAAAGGGTGGTAAGCAATGTTACTATTTTT
BH1376	*sigA*	BH3606	*secA*	TCTTGTTATTAGGACAAGGAGTTGGTAAAATGAAG
BH1376	*sigA*	BH4028	*purA*	GATTGACGTTTGTTGCCTAAATTGATATACTGTTT
BH1376	*sigA*	BH4027		ACAATGTTATGATTTTATCTCTCGAACACCTTTGC
BH1376	*sigA*	BH4022		CCGTTAAGATAATCTTCATATTTTTTAACACGTTT
BH1376	*sigA*	BH3564	*clpP*	ATTTGACCTTCTTTGACCTTTTAGGTAATATGTAC
BH1376	*sigA*	BH3561		GCTTGTGAAGTGACCGTATCCTTGATAGAATAAAA
BH1376	*sigA*	BH3488		CCTTGCGTGAATATGAAAAATTAGGTATAGTGAAT
BH1376	*sigA*	BH2900	*argC*	AGTTGAATTTATTTTTATTCAGTTTTATAATCAGA
BH1376	*sigA*	BH3316	*gabD*	CATTGTGAATTCGATAGAGGGAGCTTACAATAGAA
BH1376	*sigA*	BH3302	*pckA*	GTATAGACTATTCATTTAAATGTGTTATACTAAAT
BH1376	*sigA*	BH2889	*med*	CCATTTTCGGTATTATCATTTTTGCTATAATAACA
BH1376	*sigA*	BH2883		AGTAGACAGGGGGAGAAAAATAAGATAAAATTTAG
BH1376	*sigA*	BH2861	*spxA*	AATAGACGAATTCAAATGTTTTTGTTAAAATTGTA
BH1376	*sigA*	BH2832	*comGA*	ATGATAGCATCGTTTAAAAAATCACAAACACTGTA
BH1376	*sigA*	BH3271		TGTTGTCTACTTAAGATTCATGACTTATTATACCC
BH1376	*sigA*	BH3235	*acuA*	ATGATAATATCATGTCAATCTTTTAAAAAAGTTAA
BH1376	*sigA*	BH3234	*acsA*	AATTGAAAAAATTTTCTAACTGTACTATAATAGTA
BH1376	*sigA*	BH3232	*bglS*	AGATTGAATTTATTTAAGATTCATTTATAATAGGG
BH1376	*sigA*	BH3231		AGTATTAAGAAATTACTGATTATCGTAAAATACAT
BH1376	*sigA*	BH3229	*pdpF*	AAGCAAAAATAGTATGACGGAGTATTTTCAGTTCT
BH1376	*sigA*	BH3209	*rpsD*	AACGTAGTAAGTTTTATTGTTTCCTCCTCAGCTAA
BH1376	*sigA*	BH3205		CGATACAAAATACACATTTTCATGGTATCATATGA
BH1376	*sigA*	BH3201		TATTCAAAATAAAAAATGAATTCAGTATAATAGTA
BH1376	*sigA*	BH2781		AAAATAACTTCTTTCTTAAGGTTGTTTACTACCTC
BH1376	*sigA*	BH2773	*spo0A*	AAATTATGGGATTTTTTAATTTTTTTAAAATTTGA
BH1376	*sigA*	BH2766		GGAATAAAATTGCTTTTTTCAGCCCGAACAGCTTC
BH1376	*sigA*	BH2757	*xylA*	TATTGAATAAACTAAGTATAAGTGATATACTCTCA
BH1376	*sigA*	BH2739		TGTAGAAAATTTAGACAAAATCGTATAAAATAAAT
BH1376	*sigA*	BH2717		GAGGTAGTATCTGATCTGAATGGTTATACATTTGC
BH1376	*sigA*	BH3192	*ackA*	ACTTGAAAACTGAAGATTCTGGTGCTTAAATGGGT
BH1376	*sigA*	BH3180		CCTTTTTATGTGAGGATGACTCGGATATGGTAACA
BH1376	*sigA*	BH3160	*citZ*	CATTCACAAAAGTTGAAATAATGTTTATAATAAGA
BH1376	*sigA*	BH3157	*phoP*	AAGATCATATTTTCTATAAAAGATATTCCACTCTC
BH1376	*sigA*	BH3149	*gapB*	TAAATAATATTTACTTAATAATCATACAGTATTAT
BH1376	*sigA*	BH3148		TATTGCAAAGAAGGATTAAACAAAGTATACTATTT
BH1376	*sigA*	BH3141	*thrS*	AGAGTAGTATGGACTTGCGCTTTTTAACTTTTTCC
BH1376	*sigA*	BH3140	*infC*	AGTTGACACCTCTATAGGATTATGATAAAGTTAAC
BH1376	*sigA*	BH3118		GTAGTAATATGTTCATTTGCTTGTTGTTGACTTTT
BH1376	*sigA*	BH3111	*pheS*	AGTTGCAACCTCAAGTGAATTTCACTATAATAGAC
BH1376	*sigA*	BH3104		GATTCAAAATTGTAAGCGATTACAATACAATAGAA

BH1376	*sigA*	BH2695		TATGTACACATTTATATTTACAGTTTATAATTATT
BH1376	*sigA*	BH2655	*pdhA*	TATTTAACATAATAAAAGAGCTTTTCTACTGATTA
BH1376	*sigA*	BH2627		AAAATCATATTCGCGAAAGAAAGAAAAACAGTACA
BH1376	*sigA*	BH2617	*ctaA*	TGTGTAATATAGAGTTTTAGATAGAGCAAAGTAGT
BH1376	*sigA*	BH2603		AGTTTCCTTTTTGACGATTTTAGTCAATAATGTTC
BH1376	*sigA*	BH3096	*lysC*	TCTTGTCAGATTTCCAGCCATCTGATAAGATATTA
BH1376	*sigA*	BH3093	*sdhC*	TCTTAATATTAAGGACAAAAGCCGCGAAAATAAAG
BH1376	*sigA*	BH3061		AAAATAATAAGGTAACAACAACCTCTCTCTGTTAG
BH1376	*sigA*	BH3052	*clpX*	AAGTTAATAACTTACATATTTTTTACGCCAGTATT
BH1376	*sigA*	BH3048	*hemA*	TATTCATTTATAATAATATATATCTTATAATAATA
BH1376	*sigA*	BH3038	*valS*	TGTTGACGATCCAATGAAGGAACGCTAAAATACTT
BH1376	*sigA*	BH2559	*ftsA*	TCTATACGTTTTCTAAAAGTTTCTTTAGAATAATA
BH1376	*sigA*	BH2557	*spoIIGA*	TGATTGAAATAGTTTTCCACAAGAAAAACACATAT
BH1376	*sigA*	BH2545	*ileS*	ACTTGCGTTCTTTTTAGCTCTCGCATATAATGAAT
BH1376	*sigA*	BH2541	*pyrR*	CGTTGACACGATCGTCGCATTTTGTCATAATTGCA
BH1376	*sigA*	BH2494		GAATTAATATATGATGAGAATCATGGATCAGTATA
BH1376	*sigA*	BH1974		CTTTGTTTATTTGCAAAGGATGCTGAAGAATTAAA
BH1376	*sigA*	BH1940		CTTTTTTAACTTGTTTTAACCGTATCATGATAAAA
BH1376	*sigA*	BH1930	*aprX*	AAGATAATGTAGTAGTTAAGTTTGTAGTCTGTAAG
BH1376	*sigA*	BH1923	*bglA*	TAAATAATATATTTAAGCGACAGAGACACCATTAG
BH1376	*sigA*	BH2395	*spoIIIE*	GATGCAGTGATGTTTTTGTCTGTGCTATAATAATT
BH1376	*sigA*	BH2329	*ald*	TATCTGAAATAGTCTTAAAGCCGCCTACGTGTTTC
BH1376	*sigA*	BH2312		TAAGTAAAATATCATAAAACGAGGAAAACGTTTGT
BH1376	*sigA*	BH2309		AAAATATGATAGTCTTTTATATTTTGAAAACGAAT
BH1376	*sigA*	BH1879		TTCCTAAAAAGGGTGATTTTTGTTGTAAAATGAAA
BH1376	*sigA*	BH1875	*araR*	TGGATAGTACATAAATTGTACATATTTCAAGTAGT
BH1376	*sigA*	BH1873	*araA*	ACAATTGCATTGAATAATTAAAAGTTTAGAATAGT
BH1376	*sigA*	BH2299		TTTAGACAATTTTGACAGCTTACGGTATGATAGTG
BH1376	*sigA*	BH2285	*nadE*	TTTCGAATAGAACGGCAAGACGTGCTATCATAATA
BH1376	*sigA*	BH2237	*gapN*	AGTATAGCATTGCTTAGCTGAATGTTTTCATTTCA
BH1376	*sigA*	BH2216		GATTGACTAACTTGTATATACAAGATAAAATACAA
BH1376	*sigA*	BH2214		GTTTACAACTGGGGGAGTGTCCTTTTACAATGAAT
BH1376	*sigA*	BH2206	*sucA*	CTTTGAGCAAATGTGACAAACATGTTAAAATACTT
BH1376	*sigA*	BH1790		TTTTGATTAAAAATGGGGAAGATGATAAATTGTTT
BH1376	*sigA*	BH1775	*blt*	TGTAAATTTCCTACTTTTTTTAGGTTAAAATGATT
BH1376	*sigA*	BH1738		GATGTAAAATTGTCTTTACCTTGCTCCTTTTTGCA
BH1376	*sigA*	BH1736		GAATGGATTTATTTGTAAAGAGTAATAGAATGGAA
BH1376	*sigA*	BH1728	*gltA*	ATTTTCTGTTGTTATAAAAATACATTAAAATAAAT
BH1376	*sigA*	BH2179	*clpE*	CCTTGACAACTTTGAATTCCTTCGTTAAATTAATA
BH1376	*sigA*	BH2144		TGTGTACTATGTTTTTTACAACTGGATTAAGTTGT
BH1376	*sigA*	BH2102	*gltC*	AATAGATATCTGCTTGGAAGTATTTTATAATTAAA
BH1376	*sigA*	BH1659	*trpE*	GTTGACAAAAGATTCCTAATCACGGTAGACTTAAT
BH1376	*sigA*	BH1646	*folE*	TTTGCTTTTCCTCGTCTAATTGTGCTAAAATTTGA
BH1376	*sigA*	BH1627	*metB*	TGTTGCATTTTCGCTAGAATTGTTGTAATTTTTAT
BH1376	*sigA*	BH1622		GGTGGAAAAAAGAAGGTACTTTCTTTATAGTTATA
BH1376	*sigA*	BH2010		GGTTTTGATCCCTAAAAAAAAGAATTAGAATAAAA
BH1376	*sigA*	BH1529		GATGCACACTCCTCTTTTATTATGGTATCCTATGG
BH1376	*sigA*	BH1514		GATATATTACGGTAGTATGTAAATGTAATTTTTTT
BH1376	*sigA*	BH1486	*cysH*	CATTGACATTTTTAGCTAAGATCGATAGAATACTA
BH1376	*sigA*	BH1445	*fumC*	TTTTTCATAATTTTAGTTAATAGGATATCATATAG
BH1376	*sigA*	BH0998		ATTTACTTCTCAAGACTTTATAGGATAAGATATAA
BH1376	*sigA*	BH0991		AGTGGGAATGCCTTAAAAATACTGAAAAATTAGAT
BH1376	*sigA*	BH0951		TTCATACTATACTATTTATAAAGATTATAATGTAA
BH1376	*sigA*	BH0938		CCTGTTCACTCGTTTCGCTATACGATATAATGAAG
BH1376	*sigA*	BH0936		GAAGTAATATAGCATATCGCTTTGCTCACTTGTCC
BH1376	*sigA*	BH0912		AGAATAATATGGTATTTTAACAGTAATGCATATTT
BH1376	*sigA*	BH1352	*dra*	GCTCGCCAATCACTCTGTTCTATGCTACCATACAG

BH1376	*sigA*	BH1348	*dnaJ*	AAGATAATGTAGTAGTTAAGTTTGTAGTCTGTAAG
BH1376	*sigA*	BH1344	*hrcA*	CGTTGACAAATAGGGACCTTTTTGATAAGTTAGCA
BH1376	*sigA*	BH1342	*lepA*	TATTGAATGATCACTTTTACGTTGCTATAATTGAA
BH1376	*sigA*	BH0899		TAAAAATTATATTATTTATTTTTGTTTTATTTTTA
BH1376	*sigA*	BH0874		TGGGTAATAAGTTAGCAAAGCGAATGGACAGATAA
BH1376	*sigA*	BH0855		AAAAGTATGTTATTTCTGAAATTTCTAAAATAAAA
BH1376	*sigA*	BH0844		ATTTTAATACGTTGTCAGCAATTTCTTTCGCTAAG
BH1376	*sigA*	BH1260		CTTATAATATACAAGATATAAATCGTAAACTTCTC
BH1376	*sigA*	BH1218	*nadB*	TCTTGTCGCTAGTTTAAATTTGTGTAAATATAGAT
BH1376	*sigA*	BH1217	*nifS*	TAGATATAAATGTGTTTAAATTTGATCGCTGTTCT
BH1376	*sigA*	BH1211		ATAATAGAATGGTATTTTTCTCAGCTAATAATTTC
BH1376	*sigA*	BH1201	*pbpF*	TTTTCAAACGTTTTTTTAGTTGCGTGAAAATCAAT
BH1376	*sigA*	BH0799		GCTTACATTTTTCTGAAATAACATGTATTCTAGAA
BH1376	*sigA*	BH0764		TGTTAACAAAAAATTAGTACCAGATCATAATAATA
BH1376	*sigA*	BH0745		TAAAAAACATACCATACAAAGATATATAATTTTTT
BH1376	*sigA*	BH0744		TGTTTAATTTTTTGTTTGTTCATGTTATGATATGT
BH1376	*sigA*	BH1172		GTTTATTCTCGGATTGAAAATCAGTTAAAATAATC
BH1376	*sigA*	BH1169	*spoIIIJ*	TTTTTGAAACTGCCATAGAGTGTGGTATGATGAAA
BH1376	*sigA*	BH1108	*galE*	TTTTCAGACTGTCGTTTCTAAAGGGTATGAGAAAA
BH1376	*sigA*	BH0684	*alp*	AATTGATTGACATATATAAATATTTTTATATAGGT
BH1376	*sigA*	BH0681	*aldA*	AACGGCAAAATCACTTGATCCATGTTATAATATTT
BH1376	*sigA*	BH0660		GATTGTTTGAATGTTCTGATTTTTATATTATGTTA
BH1376	*sigA*	BH0623	*purE*	TGTTGACAAGATAAGCATTTGCCTGTAAGATTAAC
BH1376	*sigA*	BH0613	*nasD*	GAAGTAGCAAACGGAAATAATTGAACAAACGATCT
BH1376	*sigA*	BH0609		AATTTGACTTATCGGAATAATGTAGTACGATGGAA
BH1376	*sigA*	BH0607	*guaA*	TCTTGAACCTTTTGTCCTCGACTGATAAAATGTTT
BH1376	*sigA*	BH1095	*glpD*	CCTTGCATTTTTTCCGATGAGCCAGTATACTATAG
BH1376	*sigA*	BH1092	*glpF*	TCTATTACATCTTCTTACGTTTTTAAAACGTTTTT
BH1376	*sigA*	BH1060		ATTTTACTATGTCAGAGTATTCTGTTAAAATACGT
BH1376	*sigA*	BH1058	*hmp*	GTTTAAATATGTATTTTAAATATTGATTAATGAAC
BH1376	*sigA*	BH1047		AAAATAAGATGATAATTTACTTTATACAACTTCCA
BH1376	*sigA*	BH1007	*htpG*	AGAGTAAAATGTATTTTTAATCAACTCTTGTTTTC
BH1376	*sigA*	BH1005		AGGATTAGACAGTAATAAGCTACTTGGCAATTTTT
BH1376	*sigA*	BH0595	*bglP*	CGTTGACATTCGAACGAGCATATAGTATATTTTAG
BH1376	*sigA*	BH0578		AAACATGACCTCCTTAGAAAATTGTTATAATGAGT
BH1376	*sigA*	BH0542		GTTTTCACATTGACCTTTACCAGGAAAACTTTTCT
BH1376	*sigA*	BH0539	*dhaS*	TCTATAATACCTCCTATTTAATCTCAATCGATTTT
BH1376	*sigA*	BH0523	*rsbR*	CTTGGCAAGTATGAAGGTGTCATGAAATAATAACA
BH1376	*sigA*	BH0422		TATTATCAGTTTTTGTTCATTTATGTATAATAATA
BH1376	*sigA*	BH0408	*mta*	TAGATAGCATACGAAGCAATAATACACAAAGTTCA
BH1527		BH3891		GATATTAACTCTTACTTATAGTTATTCCCAAAATCAT
BH1527		BH3408		CAAGTGGTTTGATAATGATAACTGTGCTTATTCACAA
BH1527		BH2336		AAATAAATCTTTTATTGACAAGAATTATGACAAAGAC
BH1527		BH0713		GGAGCTTTCTTAAATCGTTAAACAACAGCATTTAAGG
BH1527		BH1157		AACGTTAACTCCTTTCAAAGCTAATTCTATATTTCCG
BH1527		BH1125		GGTGAATTCTATAACAAATCAGAAAGCTTTAAAAGCA
BH1527		BH1081	*fhuC*	GGAATAAAGTAGATAAATGAGTAATTTTGTGGCATTC
BH1527		BH1039		ACGAGCAACGACGACTAAAACAACCAGCGAACAACGA
BH1527		BH1025		TTTATTTACTATTACGAATAGTAAAAGTGACTATTAC
BH1538	*sigF*	BH1631	*sleB*	GATAAAATTACAAAATATCTCCATTCTGGAAAAGAC
BH1538	*sigF*	BH1597		TGTCATAAAACGGCCTCGCTTATCTCATACTAACGA
BH1538	*sigF*	BH1535	*dacF*	CATGTATAAAAAGTTCATGATAGGAAAAAATGGGAA
BH1538	*sigF*	BH1456		ATACATAAAATTGGTGATATTTATACATAAAATGAT
BH1538	*sigF*	BH1340	*gpr*	CAAGTTAAAAAAGCGTTCCGCTCTAAAAACTGCTCT
BH1538	*sigF*	BH1306	*katX*	AGGAAAAAAAATGTATTACCACCTTTCAAACACGTTC
BH1538	*sigF*	BH1201	*pbpF*	AAAAAAATCAACGCACTTTTAGTTACTAATACCTAC
BH1538	*sigF*	BH1100		CATACATATTTTGCATTCGTTTTCTAATACTAATGG
BH1538	*sigF*	BH3888		GAGGTAAAGTAGGCGTGTGCACGCGCATACTAAATG

BH1538	*sigF*	BH3871		CACGTATATACATTATTTCCTAGGGATTGGTAACAA
BH1538	*sigF*	BH3812		TTAAATAAAAAAGGATAAGCTCCACTGAGTATACCT
BH1538	*sigF*	BH4043	*acpD*	CGGTCTTATTTAGAAGGATGGAGTAAAAAATAACAT
BH1538	*sigF*	BH3599		GGAAGTAAAACTTCAAGTACTTCCCAGAATATCTCC
BH1538	*sigF*	BH3404		TTTGTATAATCTTACTTCTCTTTGGAGATGATGACT
BH1538	*sigF*	BH3309		GCTCATTATTAACATGGAAAACAAGAAAAAAATGTG
BH1538	*sigF*	BH2808	*splB*	AATTGTAATATTAAGTGCGACACCTAAAAATAAACA
BH1538	*sigF*	BH2803		GGAAAAATAAGGGATTGTATCTCCAAAAACACACAA
BH1538	*sigF*	BH3229	*pdpF*	CCTGTATTGAATGTTAGATCGTTTATAAAAAAATAA
BH1538	*sigF*	BH3051	*lonB*	AAGCGAAATTTTATTCGAGAGTTTGGGATACGTGTA
BH1538	*sigF*	BH2554	*sigG*	ACGTATAAAAAGGGAGGATTCCTCTATGAAATGCAC
BH1538	*sigF*	BH1980	*katB*	CACGTTCAGTGAGACCTCTTACTGAATGAAATTTTA
BH1538	*sigF*	BH1934	*gerKA*	ACGTATAAAAAGGGATCTACCGTTTCCAATTCAAAT
BH1538	*sigF*	BH2188	*gerAA*	TAGAATGAATTAAAGCCTCAATGCACATATTACAAC
BH1580	*resD*	BH0613	*nasD*	ATGTTTTCTTTCACAAT
BH1580	*resD*	BH1058	*hmp*	ACATATGTTTTATGAAT
BH1826	*aciR*	BH1822	*acoA*	GGTCACAAAGGTTAGCGATTTTTTATTTAAATGACAACCGTTTGGAAGATTCC ACAAAGGTGGA
BH1826	*aciR*	BH0776		TTAAGGCATTTTTTCCAACTGCCCTCACCTAAAGCAAAAAATAGTATAAATGT TTACATTGATA
BH1875	*araR*	BH1875	*araR*	CATTTGTACGTATAAGTT
BH2102	*gltC*	BH1728	*gltA*	ACAGTAAAAGACAA
BH2102	*gltC*	BH2102	*gltC*	AGAGTAACATTGTC
BH2356	*lexA*	BH0006	*gyrB*	TTGACTTGCGTACATACGCATC
BH2356	*lexA*	BH3763		TTTGTCTGAAAAAAAGACACTT
BH2356	*lexA*	BH3595	*uvrB*	ATAACAAAACAAACGTTCGTAA
BH2356	*lexA*	BH2834		GTTGCTAGACAGCAAAAAAATA
BH2356	*lexA*	BH2741		TTAAACGAACAAATATTCACAT
BH2356	*lexA*	BH3169	*dnaE*	TGGTATTTGTTACAAAACTTTT
BH2356	*lexA*	BH3097	*uvrC*	TATCCCTGTCTACAAAAAAGTA
BH2356	*lexA*	BH1930	*aprX*	ATAATCAAACGAAGGTGATCAA
BH2356	*lexA*	BH2356	*lexA*	TAATCTTGAATACAAACGTAAT
BH2356	*lexA*	BH2140		CAGTCCGAACATGCGTTTGTAT
BH2356	*lexA*	BH0831	*vpr*	TTAACAAACTATACGTTTTTCA
BH2356	*lexA*	BH1224	*ruvA*	TAGGTTAGCAAAAAAGCTAACC
BH2431	*sigD*	BH3940		GTTCACTAAGATAGCCACAGCAAGTCTTTCAGAATCAGGC
BH2431	*sigD*	BH3915	*tlpA*	CAATTTTTAAACATTTCAACACGATCGATATATAGATATA
BH2431	*sigD*	BH3616	*hag*	GGACTAAACTCCTGTGAAATCGTGTCGATATTATTAATGT
BH2431	*sigD*	BH3316	*gabD*	AAGAAAAAGAACTACCAGCATTGGCCTATCAATATGTGAT
BH2431	*sigD*	BH2312		AAATAATATTATATCTGTAAGAATAGTTATCTTATAAGTC
BH2431	*sigD*	BH1827	*mcpC*	ATATTAACTTCTTTATAATAATTGCCGATGCTATGAATAG
BH2431	*sigD*	BH2275		ATATGAGTACATATCTACAGAAAAAGACTTTTAAGTCAAT
BH2431	*sigD*	BH2237	*gapN*	TTAAAAATTAGTACCCGTTTATTTCGCGAAATAAACACTT
BH2431	*sigD*	BH1738		TTCTTTACAAAGTAGTTATCCTCGCAAATGTAATGATGGG
BH2431	*sigD*	BH1712		TTTCTTCAAAATGACGGAAAAAGGACGATAAAAATAGTGA
BH2431	*sigD*	BH2010		ACATTTCTTATTCTCATCATTCTTTCGATCTTTCTACCGA
BH2431	*sigD*	BH0855		TAAAGACTTTATAGACGATCTAGCTTTTCATACAATAAAG
BH2431	*sigD*	BH0684	*alp*	TTATAAAAATATATCCAGGCGACAACTTTTTCTTATTGCT
BH2431	*sigD*	BH0681	*aldA*	GGAGAATAAAAAATCAAAAATTACAAGATAAAAAAAGGAA
BH2431	*sigD*	BH1005		ACCGTTAAAAAATTTCCTCATGAACCGATAATAATAGTGT
BH2431	*sigD*	BH0539	*dhaS*	CTTCCAAATGATGAATTACCAATTCCGATTTTCGATACAG
BH2431	*sigD*	BH0505		TCACTTTAGAAAATTTACATTTAACCGATAGTATAGATAG
BH2462	*codY*	BH3616	*hag*	TGCTCTTTTTTAGTATTTTTTAAAA
BH2462	*codY*	BH3061		AAAGATTTTTTTATTATTCCATTGT
BH2494		BH2883		TAAATCATCATCCATGATTATTATTGAGCATTAGGGATGCTCCTTACTA
BH2494		BH2843		CCTTCAAAAGAATAGAGAACCGTGATATAATTAGGACCAAGTATTAAGA
BH2494		BH2494		AAAAAAACAAGGCTTAATTATATACTACTCTTAGTACCTAGTCATATGT
BH2494		BH0764		TTAATCATGGTCTAGTATTATTATTCACTATTTTATATAGGTTCACTGT
BH2541	*pyrR*	BH2541	*pyrR*	ATAAGCTCCTTTAATGTTAGTCCAGTGAGGCTAAGAAGGAATCACGAAAACG CC

BH2556	*sigE*	BH4053		CATCATTGTCATATGCCTATATAAAAGGAAAGATATGCAA
BH2556	*sigE*	BH4027		AAGAACGGTGCTAAATTCCCTCGTATACAATGTTATGATT
BH2556	*sigE*	BH3276		AACCTTAGTTGTATACCCGTTAGCGATTTATACCTGTCAA
BH2556	*sigE*	BH3157	*phoP*	TTTTCTAGTATAAAAGATATTTTCTATAAGGTGAGAGAAG
BH2556	*sigE*	BH2617	*ctaA*	AAAGGTGGAATAATACTGTGTTTGTGTAATATAGAGTTTT
BH2556	*sigE*	BH3070	*gerM*	TGGTCTAATTTGTGGGCAGGCTCGTATACATTTAGTACAA
BH2556	*sigE*	BH2588		AAGCAAATTTGTCTAACCGTTTTGCTTAGTCTACAAGGCA
BH2556	*sigE*	BH2572	*spoVD*	TTCGCATAAACAACTCCCTATTCGAATAAGTTGAAACAAG
BH2556	*sigE*	BH1903	*spoIIP*	TTCTTCTAAATACTAGACTTCGTTCATAGATTTAGGACAA
BH2556	*sigE*	BH2363		TGAAATCTCCTTCTCTTCTGCCTGCATCCAGTGAAAAAAG
BH2556	*sigE*	BH2292		TGTTCATGAACAGTTCCATTGTCGCATAAATTGTGGGATG
BH2556	*sigE*	BH2289	*spoVK*	AGTAACTCTTTAATCCGTAACTAAGCCTTTGTCTACTAAA
BH2556	*sigE*	BH2170		TATTGTTAAACGTTATTATTCTATCATATTTTCGTATAAG
BH2556	*sigE*	BH1645	*spoIVA*	TAATCATAAAGTGTTCAGGTCGCTCATATAGTTGACAAGA
BH2556	*sigE*	BH2029		ATCGGTATTTTCATAATCGTTAGTTGAGAATAACATGGAA
BH2556	*sigE*	BH1573		CCATTCATACCAATCCTTGCGGTGCATACACTTGTACAAA
BH2556	*sigE*	BH1526	*spoIIM*	ATGGCTTAAATCCGTATTTAGGAAAATATTATGAAACGAC
BH2556	*sigE*	BH1508		TCCGATTGTTCTTTGTATTGCTGGCATACATAATCGGTGA
BH2556	*sigE*	BH0874		TAGAAAAAGCATACCCGCCGATCCACACGATAATTATGAA
BH2556	*sigE*	BH0822		TATAAATATTCCCTAGACTTGTCTCATACATATAGTGAGG
BH2556	*sigE*	BH1233	*spoVB*	AGAAGAAGTAACATGCTAACATCTCGGTTAGTATGTTAGA
BH2556	*sigE*	BH1151		TAGTTCTATTTTTATAGAGTGGATCATAGGGTAAAGGTAT
BH2556	*sigE*	BH1130		CTGTTCTTTTCTTCATTGTGCGATCATACATTTGTAACGG
BH2556	*sigE*	BH1112	*spoVR*	TCATCATCATTAGGGAAGATGATTCATAAAATGAACAAAG
BH2556	*sigE*	BH1031		AGCTAAATTAGTATAAGGTGATAACCTATCCCTAAAGTTA
BH2556	*sigE*	BH1029	*prkA*	ATGAACTAGTCCAGCAAACAACGGCATATGAAGAAGGAAA
BH2556	*sigE*	BH0519		TCTGCATATTGCCCCAGTTTGTCTCATATAGATGTAGTGC
BH2556	*sigE*	BH0243		TATTCATACCTTGTCCGTCTTTTCTATATACATGAATATA
BH2556	*sigE*	BH0239	*cwlD*	ACTACATATTTTCCCACTTGTCCAAATATATATAGGAACG
BH2716		BH2627		CCAAACGGTTGGGTCAATGGTTAGTGGATGGTTCACGTTTCCTT
BH2716		BH2717		CATTACCGAGCTGTAGTCTTCTGTTTCTTTCTGTAGCCTCCATC
BH2773	*spo0A*	BH2557	*spoIIGA*	TTTTGACACAATCA
BH2773	*spo0A*	BH0078	*spoIIE*	TTTTGACAAAATCT
BH2777	*ahrC*	BH3943	*rocD*	ATGATTAAGAAATAAA
BH2777	*ahrC*	BH3940		TAGTTAATAGTCATTA
BH2777	*ahrC*	BH3316	*gabD*	AACGTGTAAATACTTA
BH2777	*ahrC*	BH2312		ATTAAAAAAATAGCTA
BH2777	*ahrC*	BH2237	*gapN*	GTTTTAAAAATTAGTA
BH2777	*ahrC*	BH1738		CTGAACATACATTAAT
BH2777	*ahrC*	BH2010		AAAATAAAAATAACTT
BH2777	*ahrC*	BH0991		ATGAATGAAGCTGAAT
BH2777	*ahrC*	BH0681	*aldA*	ATATTATAAATTTGTA
BH2777	*ahrC*	BH1005		ATTTTTCAGAGAAGTA
BH2777	*ahrC*	BH0539	*dhaS*	ATGCAAAAAATCTCAA
BH2777	*ahrC*	BH2900	*argC*	TAAATAAAAATAAGTC
BH3102		BH3488		TTTTTTTTGAACAAAAAATGAATGAGTATTCATTCGTTT
BH3102		BH3104		GGAATTAAGGAGGAGGAGACAATGACAATTGATCGAGTT
BH3102		BH2144		TTGATGGAGAATAGATTTTGAATGAATATTCATTCAAAA
BH3102		BH3802		TAATTTAAAAAACTTTTATGAGTGAATACTCATTCATAA
BH3157	*phoP*	BH4027		TTATTCTTAAGATAGATTTAACT
BH3157	*phoP*	BH3157	*phoP*	TCAAAAAGATCATATTTTCTATA
BH3157	*phoP*	BH1580	*resD*	AGGTATGTTACACTATTTGTAAA
BH3157	*phoP*	BH0874		TTCTTTTGAAAATAAGAACGAAA
BH3241	*ccpA*	BH3192	*ackA*	TTTTTCGCCAAAGT
BH3241	*ccpA*	BH3316	*gabD*	TCTCTCGCGACAGG
BH3241	*ccpA*	BH3235	*acuA*	ACCTTCGCGAACGT
BH3241	*ccpA*	BH3234	*acsA*	TGCAAGCGCTTCCA

BH3241	*ccpA*	BH3232	*bglS*	ACTTTCGCGAAAGC
BH3241	*ccpA*	BH3231		ATACACCACTTTCA
BH3241	*ccpA*	BH3201		ATCTAACCCTTTCA
BH3241	*ccpA*	BH2766		GCATTATCAATAGA
BH3241	*ccpA*	BH2757	*xylA*	TGATAACGCTTACT
BH3241	*ccpA*	BH2739		TGTAAACCCTTACA
BH3241	*ccpA*	BH3160	*citZ*	TTGTTACGTTTTCC
BH3241	*ccpA*	BH3104		TGTAAGCGATTACA
BH3241	*ccpA*	BH2320	*iolB*	TGAAAACGATTTCA
BH3241	*ccpA*	BH2312		AGAAAACGATTTCA
BH3241	*ccpA*	BH1873	*araA*	TGAAAGCGTTAAAA
BH3241	*ccpA*	BH1822	*acoA*	TGAAAGCGCTTCAT
BH3241	*ccpA*	BH2237	*gapN*	ATAAAGCGCTTTAT
BH3241	*ccpA*	BH2216		TGAAAACGATAACA
BH3241	*ccpA*	BH2029		TATTTCGACTTAGT
BH3241	*ccpA*	BH2010		ACATTCGCGACAGT
BH3241	*ccpA*	BH0912		TCCTTCGCAAAAGA
BH3241	*ccpA*	BH1352	*dra*	CGATAACTATTTAA
BH3241	*ccpA*	BH0776		TTATACTGCATTCA
BH3241	*ccpA*	BH0745		ACTTTCGCGAATTA
BH3241	*ccpA*	BH0681	*aldA*	AGGGAGTGCTTACA
BH3241	*ccpA*	BH1092	*glpF*	TGACAGCGTTTACA
BH3241	*ccpA*	BH1005		AACTTCGCCTACGA
BH3241	*ccpA*	BH0595	*bglP*	ACTTTCGCAACTGT
BH3241	*ccpA*	BH0539	*dhaS*	TTAAAACGATTGAA
BH3241	*ccpA*	BH0512		AATTTTTCCAATTT
BH3241	*ccpA*	BH0296		ACTTTCGCAACTGT
BH3241	*ccpA*	BH3944	*rocR*	TGTTGACGATTGCA
BH3241	*ccpA*	BH3940		TTACAACGTTGACC
BH3241	*ccpA*	BH3924	*mmgD*	ACATTTGCAATAAG
BH3241	*ccpA*	BH3823	*eutD*	TGGAAACGCTTTAA
BH3241	*ccpA*	BH3820		TGACAAGGCTTACA
BH3241	*ccpA*	BH3727	*rbsR*	TGTAACCGGTTACA
BH3241	*ccpA*	BH3563	*sigL*	CGTAAGCGCTTACA
BH3561		BH3561		CTTAATCGGGACGTAAAATGTCTATGAGGGTCAAAAAACGTCCCGG
BH3563	*sigL*	BH3943	*rocD*	ATAAAGTTGGCACAGGATTTGCATTGATTATTAGTA
BH3563	*sigL*	BH3940		AAGAAGTGTCCGAGAAACTTAGAAGTAAAATCGTTG
BH3563	*sigL*	BH3316	*gabD*	AGAGGAGTTTCAAGACTTTTGCACATTTATGAATGT
BH3563	*sigL*	BH2312		ATTGAAAAGGGTTGGCAATTGAAAAAATTAAAAAAA
BH3563	*sigL*	BH1822	*acoA*	TTATTGTTGGCACAACTTTTGCAAGAAACAAAATGT
BH3563	*sigL*	BH2237	*gapN*	AAATAAATAGAAAATTACCAAAATTTTTAATCATGG
BH3563	*sigL*	BH1738		AATGCATACTTACACTGTAAAGACCCGGCTAGTGAA
BH3563	*sigL*	BH1622		GAAGTTTTTAGCTACGTTTCTTAGTAGGCTAAAAAC
BH3563	*sigL*	BH2010		TAAAAATAACTTTCCGGTGAACGCGAGTGTAAAGAA
BH3563	*sigL*	BH0991		TTAGACTTGGCACGCGATTTGCATATTCATTTGGTA
BH3563	*sigL*	BH0776		CGCTCAAAATATTTCTTTAAAAACGCGCTTAAAGTT
BH3563	*sigL*	BH0681	*aldA*	GATAAAAAATAAATGGCCCAAAACAGGGTCGGAGAA
BH3563	*sigL*	BH1005		GTTGAACTTTATCACTTTAATTTTTCAGAGAAGTAA
BH3563	*sigL*	BH0539	*dhaS*	GAAAAATTAACGGCCGTGTTGTATTAACAATGGCTT
BH3628	*degU*	BH2216		AACAAATATTGTTGAATTTA
BH3628	*degU*	BH0855		AATAAAGACTTTAAAGATTT
BH3628	*degU*	BH0684	*alp*	AGGAAACATTACTTCGATTT
BH4022		BH4022		TTTGGATGACTCTCTCTTTCATCATGTCATTTAAAACGTGCGTCAATTTCCCAC GGTTAAAGGTGCTACTTCCATGATTAAGCGTCAATGTTGTTGGCAATTCTATT AGAAGT

Table 1: Binding sequence predictions for *Bacillus licheniformis* ATCC 14580. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BL00075	*perR*	BL03934	*dltA*	AATATTACTGAAAATTACCTC
BL00075	*perR*	BL04024	*dhbF*	AGTATTAGTGGCCTTAGATAC
BL00075	*perR*	BL01727	*lchAB*	TACCTTTAAAAGATTGTTTTT
BL00075	*perR*	BL01726	*lchAA*	AAAAATTATAAAATTATATGA
BL00075	*perR*	BL00751	*mrgA*	AATTTTATTAATATTAAATCA
BL00075	*perR*	BL00623	*hemA*	AATATTAGTAAGATTAAATCT
BL00075	*perR*	BL05249	*fur*	AATATTACTAATAATTTTTTA
BL00075	*perR*	BL00200	*ahpC*	AATATTATTAATATCTCTTTA
BL00075	*perR*	BL00075	*perR*	AATATTTGCAATATTACATTC
BL00075	*perR*	BL03560	*zosA*	AATATTATTAATAATAAACTT
BL00075	*perR*	BL03329	*spxA*	ACTTTTTTAGAGTTATTTCAA
BL00075	*perR*	BL03007	*yvgW*	ACTTCATTAAAAGGAATGTAC
BL00075	*perR*	BL01899	*copA*	AGTTTTAGTACTTGCTTTTTT
BL00075	*perR*	BL00951	*katA*	AATATTATTAATATTTATTTA
BL00075	*perR*	BL00939	*katX*	ATTTATTTATAATTATTATAA
BL00076	*dnaA*	BL00076	*dnaA*	CAAACTGTGGATAAGTT
BL00127	*lexA*	BL01091	*yhjE*	AGTCCTTGTAAACAAGACAAAA
BL00127	*lexA*	BL00588	*pcrA*	ATAATAGAACATATGTTTGAAT
BL00127	*lexA*	BL00408	*dnaE*	CTTTCTTGTAAACAAAGAGTAA
BL00127	*lexA*	BL00325	*uvrC*	ACCGCTTGCATGCCAGCCAGTA
BL00127	*lexA*	BL05164	*recA*	AAAATCGAATATGCGTTCGCTT
BL00127	*lexA*	BL00127	*lexA*	GGGTCTTACAAACAAGCCAAAA
BL00127	*lexA*	BL00081	*gyrB*	TTAGCTAAATAAAAAGAGGAAA
BL00127	*lexA*	BL03950	*vpr*	TATTCTTGCATATAAGGGATGT
BL00127	*lexA*	BL03749	*xkdA*	AAAACAGAACATTCGTTCGAAA
BL00127	*lexA*	BL03394	*uvrB*	TTGGCTTGAAATCAAGCATAAA
BL00127	*lexA*	BL02882	*yhdT*	GCTCTTTTATAACAATCTAAAT
BL00127	*lexA*	BL02863	*yhaO*	ATAGTAGAACGTGCATTCGCTT
BL00127	*lexA*	BL02847	*yhdP*	GTAACAAAACGGACTGTCGATT
BL00127	*lexA*	BL02587	*yugS*	AGCACTTGAAAAGAAGCCCTCA
BL00127	*lexA*	BL02505	*yqxL*	AAAAATGAAAGAATATGTGTAA
BL00127	*lexA*	BL02455	*yqhB*	TTTACTTTACGACAGGCCAAAA
BL00127	*lexA*	BL00903	*aprX*	GGCCTTTGAGAAGACGAAAATT
BL00127	*lexA*	BL00801	*uvrX*	TTAGCTTGCATATAAGCCTCAT
BL00127	*lexA*	BL01186	*parE*	TAAAAAAGACAAAAATCCTTGT
BL00127	*lexA*	BL01144	*ruvA*	ATCGCTTGTATACAATTGAAAA
BL00182	*iolR*	BL01181	*ywtG*	ATATGTAGTGATTAAGTAAT
BL00182	*iolR*	BL00585	*dhaS*	TTTTTAATGGTTTTATTGTT
BL00182	*iolR*	BL00293	*mmsA*	CTCTTAATCGGTTTTTATAC
BL00182	*iolR*	BL00246	*iolA*	ATTTTGATTGACTTATGGGT
BL00182	*iolR*	BL00182	*iolR*	TGGGTATTCAGTTAGTTTTA
BL00182	*iolR*	BL00119	*iolT*	TGGATTGTTAAATAAATTTG
BL00182	*iolR*	BL03906		AGGAAATTAAATTACCTTTC
BL00182	*iolR*	BL02564	*gbsA*	TTGTTAAGTTATTTCTGTAT
BL00182	*iolR*	BL02474	*ywtG*	CAGATTTAAAATTATTTTTT
BL00182	*iolR*	BL02341	*yfmT*	TTTTTGAATGTCTGATTGAA
BL00182	*iolR*	BL01765	*gabD*	ACAAACGTTACAGCCTTTTC
BL00182	*iolR*	BL01710	*ycgN*	TGTTTTTCGAGTAATTTTTC
BL00194	*gntR*	BL00194	*gntR*	ATACTTGTATACAAGTATACTC

BL00230	*deoR*	BL02102		ACGGAAGAAGAATTCATGGAGACGTATAACTTTATCAAA
BL00230	*deoR*	BL00229	*deoC*	ATTGAACAAAATTTCATATGATAATTTACATTTGTTCAA
BL00330	*ysiA*	BL03580	*ykuF*	TTAACTTACTTATGAGTAAGTATTGTTCCCCCTACCTAT
BL00330	*ysiA*	BL01987	*yngI*	AAATCTATTGGGAAGCCCGGAAAAAGGGTTTAAACAATT
BL00330	*ysiA*	BL02180	*yusL*	ATTTTTTTGGAGTAAAAATGAATGAGTATTCATTCAAAG
BL00330	*ysiA*	BL01660	*ydaB*	TATGAAATAGTTTTACAATGGATATATTTTGTTTTAGAA
BL00330	*ysiA*	BL01325	*yngIA*	ATTTTATTCGGGAGGCAAGGAGGCACTATGGTTACATTA
BL00330	*ysiA*	BL01080	*yhfL*	ATTTTGATTATCACAAAATGAATGGTCATTCATTCATTT
BL00330	*ysiA*	BL00331	*lcfA*	ATGAAAAAATATCATTACTTTCTGAATATTGTTTTGATA
BL00330	*ysiA*	BL00330	*ysiA*	TTATTTACGATTAACTCATGAATGAGTATTCATTCATTA
BL00330	*ysiA*	BL03924	*ywjF*	ATAACTGACTTGTGAGTAAGTAATAAATTCTATTTCCAT
BL00396	*phoP*	BL01643	*phoD*	AAATTTCTATTACACTTCTTTTT
BL00396	*phoP*	BL00661	*resA*	TAAATTTTCACACAACCTTCAAA
BL00396	*phoP*	BL05301		AAATTCTTCACAATTATTTAAAT
BL00396	*phoP*	BL00396	*phoP*	AGATTTTAAGCAAAACACTGAAA
BL00396	*phoP*	BL00184	*glpQ*	TTATTTTTAAAATCCTGTTGACA
BL00396	*phoP*	BL03157	*pstS*	AAAACCTTAATAATTCATTTAAA
BL00396	*phoP*	BL03110	*yfkN*	AAACTTTACAAAAAGTTCTTACA
BL00396	*phoP*	BL02371	*yycF*	ATTTTTTTAACTTTTGCTTCATA
BL00396	*phoP*	BL01381	*phoB*	AAAATTTACATAAGATTAATAGA
BL00396	*phoP*	BL00658	*resD*	TTATTTTACTTACATTCTTCTAA
BL00396	*phoP*	BL02464	*tagD*	AATATCTTTACATTAGATTAAGA
BL00396	*phoP*	BL02432	*tagA*	AGAATTAGATTACATTTCTATAA
BL00444	*ccpA*	BL02480	*alsS*	TGAATGCGATTTCA
BL00444	*ccpA*	BL00608	*ilvB*	TCTGAACGATTTCC
BL00444	*ccpA*	BL00419	*ackA*	ACTTTCGCAATTGT
BL00444	*ccpA*	BL00146	*ydaP*	TTGTAACGCTTTAT
BL00444	*ccpA*	BL00060	*gltP*	ACATTCGCTATAGT
BL00444	*ccpA*	BL03925	*mmgA*	ACTTTTGCCTTACA
BL00444	*ccpA*	BL03906		GGTAAACGATTGAA
BL00444	*ccpA*	BL03905	*eutD*	ACATTTGCAATATT
BL00444	*ccpA*	BL03890	*sacP*	ACTTTCGCATAATT
BL00444	*ccpA*	BL03867	*xylA*	TTTTAACGTTTGCA
BL00444	*ccpA*	BL03844	*licH*	ACCTTCGCTATCGG
BL00444	*ccpA*	BL03842	*licC*	ACTTTTGCCAAAGT
BL00444	*ccpA*	BL03454	*sigL*	TGAATCCGTTTTAA
BL00444	*ccpA*	BL02959	*cimH*	TATCAACGCTTTAC
BL00444	*ccpA*	BL02891	*citA*	AAGTTCGTCAAATT
BL00444	*ccpA*	BL02833	*msmX*	TTAAAGCGCTTACA
BL00444	*ccpA*	BL03124	*levD*	TGAAAACGCTTAAT
BL00444	*ccpA*	BL03068	*treP*	ACTTTCGCGATGTT
BL00444	*ccpA*	BL03017	*glvA*	TGTAAACGCTTATA
BL00444	*ccpA*	BL03016	*acoR*	ACTTTCGCGAAAAT
BL00444	*ccpA*	BL03012	*acoA*	TGCTATCGCTTCAA
BL00444	*ccpA*	BL02564	*gbsA*	AAGTTTGCCAGAGT
BL00444	*ccpA*	BL02543	*maeN*	ATAAAACGATCAAA
BL00444	*ccpA*	BL02474	*ywtG*	TTAATCCGCTTGAC
BL00444	*ccpA*	BL02470	*kdgA*	AATCTCCGCTTCCA
BL00444	*ccpA*	BL02438	*rbsR*	ACATTTGCCAATGT
BL00444	*ccpA*	BL02405	*menE*	TTGTACCGCTTGAA
BL00444	*ccpA*	BL01987	*yngI*	TGAAAACGCTTTCT
BL00444	*ccpA*	BL01947		ACTTTCGCAACTGT
BL00444	*ccpA*	BL01919	*fruR*	ACTTTCGTGAAAGT
BL00444	*ccpA*	BL02341	*yfmT*	TTTTACCGATTTAT
BL00444	*ccpA*	BL02334		TGTTGACGCTTTCA
BL00444	*ccpA*	BL01830	*rocR*	TGCCAACTCTTAAT
BL00444	*ccpA*	BL01803		AATAACCGCTTTCA
BL00444	*ccpA*	BL01802	*ywbA*	ACTTGAGCGAAATA

BL00444	*ccpA*	BL01723		GGATAATGCGTACA
BL00444	*ccpA*	BL01710	*ycgN*	TGAGACGGCTTGCT
BL00444	*ccpA*	BL02188		GCCTTAGCGAATAT
BL00444	*ccpA*	BL02179	*yusK*	ACTTCCGCAATTTG
BL00444	*ccpA*	BL02102		TGATCGCTCTTTCA
BL00444	*ccpA*	BL01660	*ydaB*	ACTTTATCAAAATG
BL00444	*ccpA*	BL01646	*ybcD*	AAATTTGCGTTAGT
BL00444	*ccpA*	BL01498	*bkdR*	TTTTTCGCAATTTT
BL00444	*ccpA*	BL00923	*cydA*	ACATTAGTATTAGT
BL00444	*ccpA*	BL00914	*sacX*	ACTTTTGTAACTCT
BL00444	*ccpA*	BL00913	*citM*	ACATTCGCCTAAGT
BL00444	*ccpA*	BL01325	*yngIA*	TTTTGACGATTTCC
BL00444	*ccpA*	BL01182		ATATTCGCGATGGG
BL00444	*ccpA*	BL01181	*ywtG*	ACTTTCGCAAATTT
BL00444	*ccpA*	BL01109	*gltT*	TCTAAATGCTTAAA
BL00444	*ccpA*	BL01080	*yhfL*	TCTAAACGATTTCA
BL00444	*ccpA*	BL00585	*dhaS*	TGTCTCCTTTTTCA
BL00444	*ccpA*	BL00561	*mmgD*	CGTTTTGCAAAAGA
BL00444	*ccpA*	BL00550	*ytcI*	ACCTTCCCAACTCT
BL00444	*ccpA*	BL00440	*acsA*	TAATTCGCAAAAGT
BL00444	*ccpA*	BL00399	*citZ*	ACATTCGTAAAAGA
BL00444	*ccpA*	BL00376	*acuA*	TGAAAACGCTTAAT
BL00444	*ccpA*	BL00352	*araA*	ACATTCGCAAAATT
BL00444	*ccpA*	BL00351	*araB*	ATACAACGATTTTA
BL00444	*ccpA*	BL00331	*lcfA*	ACTTTTTCCACAGG
BL00444	*ccpA*	BL00293	*mmsA*	TATTTCGCAACGGA
BL00444	*ccpA*	BL00246	*iolA*	ACATTCGCAAATTA
BL00444	*ccpA*	BL00229	*deoC*	TCCTTTGTAAACGT
BL00444	*ccpA*	BL00213	*bglP*	ACTTTCGCAACTGT
BL00444	*ccpA*	BL05089	*glpF*	TGACAACGCTTTCA
BL00444	*ccpA*	BL00194	*gntR*	ACTTTCACAAACGT
BL00444	*ccpA*	BL00193	*galT*	AGGTTTGCCAATGT
BL00444	*ccpA*	BL00119	*iolT*	ACTTTCGCGAAAAT
BL00523	*purR*	BL02271	*pyrR*	CTTCCTATGGCCTGTTCGCTTAGTTACTTAAGCAAATTTCTGGGGAGAGAC
BL00523	*purR*	BL01528	*nusB*	AAGCTTTTTGCTTACTATTATTTGCTAACAAGGAATAATAAGCCAATAACT
BL00523	*purR*	BL01476	*purE*	TCTAAACCCGAACATTAGAACAACAAAATATACTATCGTTCGATAATGTCG
BL00523	*purR*	BL00680	*xpt*	ACAATTTATGCTTGCAAGTTTACTTTTTTACACACGTCCAAGCAGAAAAAA
BL00523	*purR*	BL00523	*purR*	CTAATTTAGGCATACAATTCAATATAACAAAAATTTTTATAAGCCTAAAAC
BL00523	*purR*	BL05105	*guaC*	ATATTTTCCGCTTGTAAAATTTATTGAATAATTATTTATAAGCAAAAAAAC
BL00523	*purR*	BL00069	*pbuO*	TGGGCTATTGCTTATTATATCTATAACAAAACGATATTACAAGCAATAACC
BL00523	*purR*	BL03991	*glyA*	GATATATCGAACTATATGATTTGTCTATTCGTTTAATATTCGGTTTTTCCA
BL00523	*purR*	BL02932	*pbuG*	CTTTTTGTTGCTTGCTTCGCTTGTAATTTTAGTACTTACAAGCAAAAAAAC
BL00523	*purR*	BL03156	*purA*	ATAATAACGAACAAAAATGTTCAACGTTAAAATTAATGTTCGGATTTACAA
BL00651	*sigX*	BL03934	*dltA*	ATGAAACTTTTATACAACATTTTCGTCTTATACTG
BL00651	*sigX*	BL04024	*dhbF*	TGAAAACTGGTGGAGACTGTTGTCCTCCTCTAAAG
BL00651	*sigX*	BL03328	*yjbC*	TTGAAACCTTAGCGTAAGACAACCGTCTTAGTATA
BL00651	*sigX*	BL02713	*pbpX*	AAGAAACTTTTTAAGGGGTTATTGCGTCAGTAAAT
BL00651	*sigX*	BL03035	*csbB*	ACACTTTTTACAGCGTTTTCCTTTTTAACAATACA
BL00651	*sigX*	BL02453	*bcrC*	ATGAAACTTTTGCAAAATGAAATACGTCTCCTAAC
BL00651	*sigX*	BL02430	*lytR*	TTGAAACATTTTTCTAATGATTTCGTCTATCCAAA
BL00651	*sigX*	BL01936	*ykcC*	ATGAAAGGAAAGGTGATGAATATGCAGAAACAAGT
BL00651	*sigX*	BL01728	*lchAC*	TAGAACTCTACCTTCTTTACCCGCTGGACAAACTG
BL00651	*sigX*	BL01726	*lchAA*	TTGGAATTTTTTGCGGGTTTAAACGATTTTTTTCG
BL00651	*sigX*	BL00798	*yqjL*	GAGAAACGAACATCCTTGTTTGTGCGTCTTTCCTA
BL00651	*sigX*	BL00652		TTGTTTTAGGGCTATAACACTTCTTTTACCAACTT

BL00651	*sigX*	BL00651	*sigX*	ATGTAACCTTTTAAGATTGACAAACGACAATAAAT
BL00658	*resD*	BL00923	*cydA*	GGAAGCATATTAACAGT
BL00658	*resD*	BL01167	*fnr*	TCACAATATTGTTTATA
BL00658	*resD*	BL05177	*nrdI*	GAATGTTTTTTATTCGC
BL00658	*resD*	BL02894	*hmp*	AACAATGTTTTTATATT
BL00658	*resD*	BL01769	*nasD*	TGACAATTTTGTGAATA
BL00658	*resD*	BL01767	*nasB*	CTAAAATTTTTTGTCAG
BL00730	*liaR*	BL02898	*yhcY*	GGAAAGAGCCAGTTTTCAGAATCTTTT
BL00778	*sigF*	BL02256	*sigG*	ACGTATAAAAAGGGTGGGCTCCTCTATGACTTGAAA
BL00778	*sigF*	BL01751	*gerKA*	AATAAAAAAGAGGAAAGTATTTCGTGATAAGTGGGC
BL00778	*sigF*	BL02089	*gpr*	CAATCTAGAGTTGTTATCTTGTTTAACATAAGATAG
BL00778	*sigF*	BL02023		AATGCTTGTAAAAATTGTGAAAATTTCATAAAATTT
BL00778	*sigF*	BL01519	*spoIVB*	AGCGGATAAGCTGTTCATTTTTTGAACAGCTTATTT
BL00778	*sigF*	BL01512	*yqhQ*	AATGGTTAATGTACTCATAGATTGATACAGATAGGT
BL00778	*sigF*	BL00951	*katA*	CAACGATAATAAATTTTTCTTTATAATAATTATAAA
BL00778	*sigF*	BL00939	*katX*	AAATATTAATAATATTTCTTTTTAAATAATAGCAAC
BL00778	*sigF*	BL00896		CTGAATGATGTCGTTACTTTTTCCCACTAGTTGTAC
BL00778	*sigF*	BL00778	*sigF*	CTTACCCGAAATGCTAGTACCTTTTAAAGTACCTAC
BL00778	*sigF*	BL00775	*dacF*	TATGATAAGGAACTGCGCCTCTTGGCGAAAAAAACA
BL00778	*sigF*	BL00747	*gerAA*	CACAGTATATTGTTTTTTTAACAGGAAAAGATAACC
BL00778	*sigF*	BL01115	*yhfW*	CATGCATGAATATTTCTCAATTGGGAAAATTTAAGA
BL00778	*sigF*	BL00620	*lonB*	TACGTTTATTCCCTCCCTGATCAGGAAATACTATCT
BL00778	*sigF*	BL01071	*pbpF*	ACTATCATATTGTTTTCCTGCTAAATTTTAAGATAG
BL00778	*sigF*	BL00091	*yyaC*	AAAGCATAAAAAAAGAGACAGCTGGATATACTGTAA
BL00778	*sigF*	BL03984	*spoIIR*	GCCGTTTAAACCGTTCCATCTTTGACCATAATATTG
BL00778	*sigF*	BL03962	*ywhE*	CTTGTTTAAATCCTCCTCTTTTTCCTCATAATGATA
BL00778	*sigF*	BL04028	*spoIIQ*	AGTGTTCAGAATGTTGCTGAGGTGATGAGTTATGAA
BL00778	*sigF*	BL03288	*ytkD*	CACGAAAAAGGATGATTGGCTTTGGAAAAAAAGACA
BL00778	*sigF*	BL02732	*ponA*	CACTTGGATACTCGCCAAGAATTGATTATATTAAGA
BL00778	*sigF*	BL03144	*yuiC*	TCGGAAGAAGAAGTTCCCTCTCCCAAAAAAACGCCA
BL00778	*sigF*	BL02536	*pbpD*	ATCGGATAAAAAAACTGCCAATCTATCAAATTATCC
BL00847	*rex*	BL02480	*alsS*	AAAGAAGTATTGGAAAACAATTTCCACAAGATG
BL00847	*rex*	BL01703	*ldh*	TTTTCATACACTTCATAACGTGTTATACTTACA
BL00847	*rex*	BL00946	*ywcJ*	GTTTTAAACACTTTACAAAGTGTTATAGGACGA
BL00847	*rex*	BL00923	*cydA*	ACGTGAATTTCGGGGCATTTTTCTCATGTTGTT
BL00847	*rex*	BL00608	*ilvB*	ACATATATATTGATTATTTTTTGAAAAACAATT
BL00847	*rex*	BL00146	*ydaP*	AAAAAAAGCTGGCACAACATTCTTCACAATCAG
BL00911	*citT*	BL00913	*citM*	GTCAAAAAGAACAAAATGATTTTAAAAAATTAAAAATACATAAAAACCAAA
BL00915	*sacY*	BL03890	*sacP*	CGGGGATTGTGACTGGTAAAGCAGGCAAGACCTAAAATT
BL00915	*sacY*	BL03068	*treP*	GCGGTGCTGCTCTTGAGCAGAAAAGTAAGGTCTTTATAT
BL00915	*sacY*	BL02216	*sacB*	TCGGGATTGTTACTGTCTAAGCAGGCAAGACCTAAAATG
BL00915	*sacY*	BL00914	*sacX*	CCGGGATTGTTACTGGTAAGGCAGGCAAGACCTAAAATG
BL01090	*comK*	BL01908	*rapE*	TTTTAAAGCAAAAAACC
BL01090	*comK*	BL02161		AGAAAAATTATAATCTT
BL01090	*comK*	BL01572	*comGA*	ATTTAAAGATAAAGTCG
BL01090	*comK*	BL01460	*yodF*	TTTTAGTGAAAAAAGAT
BL01090	*comK*	BL01351	*addB*	TTTTAGGCCTAAAGACC
BL01090	*comK*	BL01283	*smf*	AGGAAAAAGGTCTTTAC
BL01090	*comK*	BL00636	*radC*	TGAAATTAAAAGATTTT
BL01090	*comK*	BL01090	*comK*	TTTTTGTATAAAAGTTT
BL01090	*comK*	BL05244		TTTTACTGGGAAATACA
BL01090	*comK*	BL05189	*yneB*	TTTTGCAGCTAAAATCG
BL01090	*comK*	BL05164	*recA*	CTCTAGCTAAAAAGGAG
BL01090	*comK*	BL00110	*trmE*	GATGAAAATGAAATTTA
BL01090	*comK*	BL00101	*engD*	TTTTTACTTAAAAGTAG
BL01090	*comK*	BL02867	*rapI*	TAAAAAAATTTAAGTTT
BL01090	*comK*	BL00559	*rok*	AAAACAAAAAAAATTTC

BL01137	*nadR*	BL02040	*yrvO*	CTAAGGTCTTTCGCGTCCTGCGAAAAACGCCGAGTACCTTCGTCCTGAAG GCATGATTTATTATAGGTTAGAGGTGTTTATTTTGAAAATATCAACTAAA GGCAGATA
BL01137	*nadR*	BL01155	*nadB*	TTAGTGCCGCGAGCAGAGGGTAAGAAATATTTTTTATGAACTCAAATAAA ATAGCTACACATTTATATCCACAGTTCTGTGTACATTTAAATGTCCTCCTC AGCTGTA
BL01137	*nadR*	BL01136	*nifS*	ATGTCGACTCCTCCTGTAAATTTACATGTGTCTTGACACCTATATTTACAC ATCGATAAAATAAACTCAAGTATTTTTTATAAAGAATGGGAGACGAGCGC CGTGATT
BL01137	*nadR*	BL00432	*nifZ*	TAATTTTTCTTGGAGGACCGTAAAACAAATTTAAAAGACAGTTAAATAAA AGCACTATATTATACTGTTTTCAGGTTTCTGATAATATAAAATTTTTAGTC TCTTCAG
BL01167	*fnr*	BL01172	*narG*	GCTAACTTGGTAAGCTGTCGG
BL01167	*fnr*	BL01166	*narK*	GCACACTGCATTAAGTGTTAC
BL01178		BL02653		TTTTAATATGGATATTTT
BL01178		BL02556	*araR*	ATAAGTATGCATGTTCTA
BL01178		BL02474	*ywtG*	TATTGAAACGTACAATTG
BL01178		BL01182		CTTAACATGTGTGAGTAA
BL01178		BL01181	*ywtG*	GAAACCATGCGTGATGTT
BL01178		BL01178		TTTTTGTACGTACATATG
BL01178		BL00353	*abnA*	TGTAACAAGCTTTTTAAC
BL01178		BL00352	*araA*	TTCAACAAGCATGTTTAT
BL01178		BL00119	*iolT*	TTAAATATAACTGACTTT
BL01246	*sigD*	BL00293	*mmsA*	AGCGCAACTATAAAAGTTTCATAGCGGATGAAAACTATAA
BL01246	*sigD*	BL00260	*yoaH*	CTTCTCAAATTTTGACGAAATATGTCGATACTATGGAGGG
BL01246	*sigD*	BL00246	*iolA*	AACTAACTGAATACCCATTCCGGCCTAATTTTATATTGGT
BL01246	*sigD*	BL03934	*dltA*	GGCGAAAAGTATACTCGGACAAAAAAACATTAAATTTTTC
BL01246	*sigD*	BL03906		TTTCCTTTTATTCCATAATATAACACGATATAAAAAATAA
BL01246	*sigD*	BL03887	*nfrA1*	ATAGTAAGACAATTCTATCATTTTACCATATGAAATTGAA
BL01246	*sigD*	BL03682	*sigA*	AGTTGAACCGTTCGCTGAAATAAACCGTTATATATAATAG
BL01246	*sigD*	BL03639	*motA*	GCCTAAAGTTCCTATCACGCAACACCGATATTAACGATAG
BL01246	*sigD*	BL04024	*dhbF*	CACGCAAAAGAAAGCTCTCATGCACTGACAAAACAGATTA
BL01246	*sigD*	BL03579	*cheV*	AGCATCAACTTTTAACAAAAGACGCCGATATAAAAAGTAA
BL01246	*sigD*	BL03572	*mcpC*	TATCTTAATTTTGCGAAATTTTTGCCGATAAAAAGAATAT
BL01246	*sigD*	BL03374	*hag*	GCTATAAACAATTTCAAGGCACCTCCGATATAAAATATGT
BL01246	*sigD*	BL03345	*yjbJ*	GCGTGAAACCAAAAACAAAATCGTCCGATTTTATGAGCAA
BL01246	*sigD*	BL02892	*lytE*	ACCCTTAAAAACTTTTTTTGAAAACGAATAATTAAGGAAT
BL01246	*sigD*	BL02582		TTATTTAAAAAATAGACCCCGTCTCCGATAAAAGTAGAGA
BL01246	*sigD*	BL02581	*mcpA*	AATATGCAAAAGTAAATGAAAGCTCCGATATTAGTTTCAT
BL01246	*sigD*	BL02579	*mcpB*	GCATATAAAAACTTTTAAAAGAATCCGATATTAGGGATAT
BL01246	*sigD*	BL02564	*gbsA*	TAACAAGATTTTAGGCGCAATTCAAACAATTCAATAAAGA
BL01246	*sigD*	BL02465	*lytD*	ATTGTAAAAAAATAATCCGGCTCGCCGATAAAAAGAGAAA
BL01246	*sigD*	BL02341	*yfmT*	AGATTGAATTTTCTTGAAAGATTGCCGATACATAAGATGT
BL01246	*sigD*	BL01765	*gabD*	AGCCTATAAACTAGCAGAGCTAACGTAAGTACTTCGACTC
BL01246	*sigD*	BL01727	*lchAB*	TAAGGCTTCTATAGCCGATAGTTTAGTTTGTCTAGTTTCT
BL01246	*sigD*	BL01726	*lchAA*	AAGTGGATAAAAAGCCACTTTTTTTGTTTTAAAAGTAAAT
BL01246	*sigD*	BL01710	*ycgN*	TACTCGTAGTTTAGCCGTTTCTGGCGAAGCTTAAAGTTTA
BL01246	*sigD*	BL01658	*ybdO*	AAGGTTAATATAAACATAATCAAGCCGATATAGGGAGTAT
BL01246	*sigD*	BL01646	*ybcD*	ATAGGTTAATATAAGGGGTATAAAGGTCTTACAAAATGCT
BL01246	*sigD*	BL01448	*cwlS*	GCTATTAAAAAAATATAAAATAGACCGATACAAATCGTGA
BL01246	*sigD*	BL00904		TACCTATTCCTTTCACTGTATTTACCGATAAGTATATAAA
BL01246	*sigD*	BL01275	*flgB*	GGCTTTTTTTATTTCATGTTTTAAACTATATTATCGTAAT
BL01246	*sigD*	BL01118	*epr*	TCTCTTTGAAAATTCAAAAATCCTCCGATATATATAGCGA
BL01246	*sigD*	BL01114	*hemAT*	AAGTTAAAAAGTAATGAAAAATTGCCGATAAAAAGATAAA
BL01246	*sigD*	BL01111	*apr*	TAATAACCAATAAATTTAAATTGGCCGTTCAAAAAAATGG
BL01246	*sigD*	BL00585	*dhaS*	TGGTCACACTTCCGATTCAGCCAGCCGATTTTTTATGAAA
BL01276	*codY*	BL02480	*alsS*	ATATAAATTATTATGTTATTAAAAT
BL01276	*codY*	BL01728	*lchAC*	AGAGACTTTCCAGACTTCGTGAGAT
BL01276	*codY*	BL01727	*lchAB*	TAAACTGATGGAAATTTTCTAACAA
BL01276	*codY*	BL01726	*lchAA*	AGGTACGACAAAAAAATTATAAAAT
BL01276	*codY*	BL00608	*ilvB*	ATATATATTGATTATTTTTTGAAAA

BL01276	*codY*	BL01090	*comK*	ATAAGTATTTATAATTTTTAGAAAA
BL01276	*codY*	BL00146	*ydaP*	AAAAAGATTGAGAAGAAATGGGAGG
BL01276	*codY*	BL03934	*dltA*	AAAAACATTAAATTTTTCTTAAATA
BL01276	*codY*	BL03772	*dppA*	AATAACGTAAAGATAATTTTAAGAT
BL01276	*codY*	BL04024	*dhbF*	AAATACGTTCTATAACTTTGTAAAA
BL01276	*codY*	BL03374	*hag*	AAAATAAAAACGATATTTGTTAAAG
BL01498	*bkdR*	BL01500	*ptb*	TTATGTTGGAACGTTTTCTATCGTACGTTATTTA
BL01518	*spo0A*	BL02254	*spoIIGA*	GATCGACAAATTCA
BL01518	*spo0A*	BL01728	*lchAC*	ATTTTTGCAGTTTC
BL01518	*spo0A*	BL01727	*lchAB*	TTTTGACAATGTCA
BL01518	*spo0A*	BL01726	*lchAA*	TTTAAACGATTTTT
BL01518	*spo0A*	BL01611	*kinC*	CTAAGACAATATCT
BL01518	*spo0A*	BL00776	*spoIIAA*	TTTTGACGAAAAAA
BL01518	*spo0A*	BL00506	*spoIIE*	TTTTGACAAAATCC
BL01518	*spo0A*	BL03934	*dltA*	TTTCCACAATAACA
BL01518	*spo0A*	BL04024	*dhbF*	CTTTGCCAAATGCA
BL01518	*spo0A*	BL03576	*kinA*	TTTTAAACTGCTTA
BL01518	*spo0A*	BL02610	*yuxH*	TTGTGACGAATTTT
BL01521	*ahrC*	BL03244	*argD*	ATGAAAAACGAAGAAA
BL01521	*ahrC*	BL02564	*gbsA*	AATTTAAAAATAAATT
BL01521	*ahrC*	BL02341	*yfmT*	AACTTCTAAATAAATA
BL01521	*ahrC*	BL01765	*gabD*	TACGTTAAAATAAAGA
BL01521	*ahrC*	BL01762	*gabT*	ACCATGAAAAGTAGTA
BL01521	*ahrC*	BL01737	*rocD*	ACTGCAAAAACAAGCA
BL01521	*ahrC*	BL01710	*ycgN*	ATGACGAAGAAATAAA
BL01521	*ahrC*	BL01646	*ybcD*	TAGGTTAATATAAGGG
BL01521	*ahrC*	BL00585	*dhaS*	ATAAACAAAAATTCTC
BL01521	*ahrC*	BL00293	*mmsA*	TAGCCAAAAATATGTC
BL01521	*ahrC*	BL00246	*iolA*	AGTATCAAGATAAGTA
BL01521	*ahrC*	BL03906		CGATATAAAAAATAAA
BL01521	*ahrC*	BL03241	*argC*	TAATTAAAAATAAGTA
BL01547	*mntR*	BL00140	*mntH*	AATTTGCACTAAGGAAACT
BL01552	*sinR*	BL01118	*epr*	CTCTTTTTTAAAACTAACTTGGAG
BL01552	*sinR*	BL01111	*apr*	TTGGCCGTTCAAAAAAATGGGTCT
BL01640	*glnK*	BL01786	*glsA*	TTAAATTTTTTTGTCGCTTTTTGTAGGTTTATGTAGGTTCCATT
BL01640	*glnK*	BL02973	*ylaM*	CAAAAAATTATAAGCGCTTTCTTTCGAATTGTCACTTCTCATAA
BL01665	*mtaB*	BL01665	*mtaB*	GACCCTCACGTTGCGGGATA
BL01830	*rocR*	BL00585	*dhaS*	GCGGCTGAAACAAACATAA
BL01830	*rocR*	BL00293	*mmsA*	TGGGCGAGAATATTTTTCA
BL01830	*rocR*	BL00246	*iolA*	GTTTTATATTAGAAGGTAA
BL01830	*rocR*	BL03906		GCCTTTTTTACAGACTAAA
BL01830	*rocR*	BL02564	*gbsA*	AATGACAAGATCTTTTGCC
BL01830	*rocR*	BL02341	*yfmT*	GAGTCTTTTTGACTCGCCA
BL01830	*rocR*	BL02226	*gudB*	TGTTTTTACTAAAAAGAAC
BL01830	*rocR*	BL01765	*gabD*	AATGCAATTTTATTTCTGG
BL01830	*rocR*	BL01762	*gabT*	GCTGAAAATTTCTTCTTTA
BL01830	*rocR*	BL01737	*rocD*	AACCGCAAAATTTTTTGCG
BL01830	*rocR*	BL01710	*ycgN*	GCGGTTTCTTTAACGCTTA
BL01830	*rocR*	BL01646	*ybcD*	AGCTCGAAAACATTCTGAA
BL01830	*rocR*	BL03016	*acoR*	AGTGCGGAGTTTTTTTTAT
BL01830	*rocR*	BL01830	*rocR*	GCGTTTTTTAAAACGCCAA
BL01830	*rocR*	BL01498	*bkdR*	TGTGCAAAATTTTTCATAC
BL01954	*cysL*	BL01949	*cysI*	ACTATTAATTATATATAAAATGATTA
BL01954	*cysL*	BL02398	*cypE*	AGGTTTTGTCTTTCCTACTAAAAGTA
BL01954	*cysL*	BL01954	*cysL*	TGTAATCATTTTATATATAATTAATA
BL02004	*gltC*	BL01971	*gltA*	ATCTCAAAATGAGA
BL02004	*gltC*	BL02004	*gltC*	AGAGTAAAACTCTA

BL02041	*cymR*	BL00070	*ytkP*	TTAACCTTAAGGTTATTTCATTACGCCAAACTGACTGT
BL02041	*cymR*	BL03206	*ssuB*	TTTTTATAATATAGGGTTGACAGGTCGGAATAATATGA
BL02041	*cymR*	BL03177	*yhcL*	TAGCTATTAAGGATGAACTGGTTAGTCCAAACAACATG
BL02041	*cymR*	BL02192	*ydbM*	GAACTATAATACTTATTAAATCTATCGGAATTTGGTCA
BL02041	*cymR*	BL02021	*yrrT*	ATCGAATAAATCATACTATACTTATAGGAATTGTAAAA
BL02041	*cymR*	BL02019	*yrhA*	CAGATCTTGTGCTTCGCATGATTAAAAGAATCAATTAA
BL02041	*cymR*	BL00857	*cysK*	TCTATATTAGGGTTATTTTAATGAGCCTTTAACTCCAC
BL02094	*hrcA*	BL02094	*hrcA*	TTAGCACTCAGTTGTGCCGAGTGCTAA
BL02104	*desR*	BL02692		ACTTATATGATGTATGA
BL02104	*desR*	BL02106	*des*	TCATGTTTGAAGCATGA
BL02123	*pucR*	BL02153	*pucF*	CCTTTGTCATTTTAGTCATTTTCAACA
BL02123	*pucR*	BL01095		ACAATTTTCACGTATTTTACGAAAAGA
BL02123	*pucR*	BL01094	*pucH*	AGTAATATTATTCTTGCAGCTTAAAAA
BL02123	*pucR*	BL03735	*guaD*	TCCTATGTCTTTACTTAGTACTTCGTA
BL02208	*sigB*	BL03680	*yqxD*	GGTTTTTGCTTTGATGGTTTTGGGAAAAGTGAAAATGA
BL02208	*sigB*	BL03673	*cdd*	GGTACCATCACTGCTATAGGAAGGAATACACACATTGA
BL02208	*sigB*	BL03605	*clpP*	TGTTTGAGTTTCCTTCAAAATGGGAAAAATAAACCCAG
BL02208	*sigB*	BL03380	*yvyD*	AACAAGAATAAAATAGGCGTACACTTATATAAATTCTA
BL02208	*sigB*	BL03328	*yjbC*	TGTTTAAACTAAAAAGAAAGCGGGTATATCTAAAGTGT
BL02208	*sigB*	BL02889	*yhdF*	CGTTTAAGGATGCCGGAATCGTGGTAACTGAGCACTAA
BL02208	*sigB*	BL02874	*yoxA*	GCTTTAAATATGAATAAGCAGGTGAAAATAATAAACAA
BL02208	*sigB*	BL02819	*ydaG*	TGTTTAATATACAAGACATTGTGGAAAAGTAACAGTGG
BL02208	*sigB*	BL02817	*ydaD*	TGTTTAAACAAAAATTTCTCCGGTTAAAAGTCTAGTAA
BL02208	*sigB*	BL03285	*dps*	TGTTTCCTGCACAAGCTAAACGGGTAAATCAAAAGTAA
BL02208	*sigB*	BL03264	*yacL*	TGATTAAAATTTGCAATTTATGGGTATATTATTCTTAC
BL02208	*sigB*	BL03258	*ctsR*	GGTTTTATGACAGCTAAAAGTGGAAATAATAAAGGACG
BL02208	*sigB*	BL03214		CTTTACATAATAAACGGACGTTCATATAAGTAATTAGC
BL02208	*sigB*	BL03112		GAGGAGATAAGAAAGGGGTGAAAAGTCTTTCTTTTTTG
BL02208	*sigB*	BL02664	*yngB*	AAAGTATCAAACAAGTAGAAAACAAATAGCAAATAAGT
BL02208	*sigB*	BL02636	*ydfO*	GGTTTCTTTCTATCAATCCAGAGGAGAAGCATTTCAAA
BL02208	*sigB*	BL02624		TTAAAACTGTATATGGGTAGAATGAAAATAATACTTAT
BL02208	*sigB*	BL03081	*yfkJ*	GTTTTTTCTTTCTGAAAAGTGGGGAAAGAATAGAGGAG
BL02208	*sigB*	BL03067	*yflA*	AGGTTTATTTCTCCCCTGTAGTGGTATTGACCCGTAAA
BL02208	*sigB*	BL03035	*csbB*	TATGTGAAAAATGTCGCAAAAGGAAAAATTGTTATGTT
BL02208	*sigB*	BL02564	*gbsA*	AAACAATTCAATAAAGACATACAAGAAATAGAATTGCT
BL02208	*sigB*	BL02505	*yqxL*	TGTTTATTTACGCGGAGGGCGGGCAAAGAAATATAAAA
BL02208	*sigB*	BL02503	*ydbD*	GCCAAAATACTAAAGGGAAAAGCCGATTTAGTTTTTGC
BL02208	*sigB*	BL02480	*alsS*	CGTTTTACATATTAATTGTAAGACAAAGAAGTATTGGA
BL02208	*sigB*	BL02474	*ywtG*	GGTTTGAAGAAAGCAAGACAAGGTTAAAGGTTAAAAGA
BL02208	*sigB*	BL02431	*gtaB*	GAAACGCGAAATATGGACAAATCTGAACAAATTCTTAA
BL02208	*sigB*	BL01936	*ykcC*	GTTCACGTTGAAGACGAGTTAGAATGTAGTAGTTTTGC
BL02208	*sigB*	BL02371	*yycF*	TTTTTTAATGATGACATGCAGTGTACTTTGATATAAAA
BL02208	*sigB*	BL02341	*yfmT*	AAAAGGTTCCTTATAGGAAAAGTAAAAAGGATTATTGT
BL02208	*sigB*	BL02213	*ydbP*	CCTTTTTCTGTTATATAAATCGGGTGTATATCAAATTT
BL02208	*sigB*	BL01785	*glnT*	TTTTTAAAATTACAAAAAAAGAGGTGAAAGCATGCAGC
BL02208	*sigB*	BL01765	*gabD*	TGTTTGGATCGTCAATGCAGGGGGAACAGGTATTATAG
BL02208	*sigB*	BL01711	*ycgO*	TGTTTTAAGGTGTAAGAAAAGGGGTTAAACAGGGGGAC
BL02208	*sigB*	BL01710	*ycgN*	CGTTTAAAGGCATGACGAAGAAATAAAACGATAAAAAG
BL02208	*sigB*	BL01706	*nadE*	GAAGAACAATAAAACGGTCCCCCGAGGCAAAAATAAGC
BL02208	*sigB*	BL01646	*ybcD*	ATAGGTTAATATAAGGGGTATAAAGGTCTTACAAAATG
BL02208	*sigB*	BL01573	*yqhA*	TCATTAAGCGATTGATTTAAAGGGAAAACAATGAGGAG
BL02208	*sigB*	BL01512	*yqhQ*	GAGTGTAAAAAGCATGAAAAAGGGTATACTTTAATTAG
BL02208	*sigB*	BL01511	*bmrU*	GGTTTACGGAACGGGCAGCCCGGGGAAAAATAAGTAAA
BL02208	*sigB*	BL00951	*katA*	ACTAGGTAAGAAGAAGGGGGAAATAAACCTTAATTGTT
BL02208	*sigB*	BL00939	*katX*	TTGTTAATTCCAAATAAAGGGGGAAGAAGAATGGATCA
BL02208	*sigB*	BL00927	*yxkO*	TGCATGTTAAAGAGCAGTGAAGGGTATGATTCAGTTAC
BL02208	*sigB*	BL01181	*ywtG*	CGAATAAAACCATGAGATTTAGGGAGAAATCCCACCAA
BL02208	*sigB*	BL01180	*alsT*	TAGAGTAGTAGAATCGTATAAAATAATATATTATTTTC

BL02208	*sigB*	BL01124	*relA*	GAACTATTATTAATGTTATGTCAAAAGTTAATATTTTT
BL02208	*sigB*	BL00608	*ilvB*	TGATTATTTTTTGAAAAACAATTGTGAATGAACAGTGA
BL02208	*sigB*	BL00605	*opuE*	GTTTTTCACGGTTGATTGATTGGGAATAGTTTGATAGG
BL02208	*sigB*	BL01089	*yhxC*	TGTTCTTAAATTTTCCCAATTGAGAAATATTCATGCAT
BL02208	*sigB*	BL00585	*dhaS*	TTTTTAATGGTTTTATTGTTCTGCCATTTTTCATGAAT
BL02208	*sigB*	BL00518	*ctc*	GGTTTAAAGTTTTGTCGTTATGGGAATGGTTTTAGAAA
BL02208	*sigB*	BL05355		GGATACAATCCTCAAGTTAATGGAAATCAATCGAACAG
BL02208	*sigB*	BL00413	*ytkL*	AATTGGCAGTAAGAGAGATCTGCACGCTTTAATTTTTT
BL02208	*sigB*	BL00396	*phoP*	CGTTAAAATAGGAGTGAAAACGTTTATTTAACAAAAGA
BL02208	*sigB*	BL00326	*trxA*	TGTGTAAAACTGTGCAGATAGGGAAACGTATAATCAAG
BL02208	*sigB*	BL00293	*mmsA*	CTGTTTTATACAAAAAAATAAGCGAACAGAAGACAACT
BL02208	*sigB*	BL00261		TTTTTCTTTAAAAAATAGCTTTGTTAAAAAAACAAATG
BL02208	*sigB*	BL00246	*iolA*	GGATTAAAATATAACCAAGAAGTGACCAAAACATGATA
BL02208	*sigB*	BL05038	*gsiB*	GTGTTTACTCTTTTTGAAATGTGGAATGTAAAAACCAT
BL02208	*sigB*	BL00183	*iolS*	TGCTTGAATATGAAATTAAAAAGAAAATATCCGAAAAA
BL02208	*sigB*	BL00146	*ydaP*	CGTTTCAGCAGAAATGGAAGGAGGTATGGAAAAAACAA
BL02208	*sigB*	BL00119	*iolT*	TTTTCATTTAAACAAAGGCGGGGGAAATCAATATGAAT
BL02208	*sigB*	BL03906		GGGAACAAAAACAAAGGAAAATAAGGTATTATATTGTG
BL02208	*sigB*	BL03858		CGGTCAATTATTCATGTATATTGATATTACGCAAATCA
BL02208	*sigB*	BL03854	*gspA*	GGTGTTTATTTTTCGAAAAAAGGGAATAGAACAAAAAA
BL02208	*sigB*	BL03741	*ykgA*	ATTTTCCAAATAAAGGGCGACGCCCATCCCTTATTACT
BL02255	*sigE*	BL02371	*yycF*	TTTTAACTTTTGCTTCATAAACGTTATAATGTATAAGAAA
BL02255	*sigE*	BL02368	*yaaH*	AGAGCATGAAACACTTCCTTTCATCATATAGTGGTAGCAA
BL02255	*sigE*	BL01881	*dacB*	GTGTTCATAACAAAAAGACAAGCGCATAAACTTGTACAAA
BL02255	*sigE*	BL02242	*spoVE*	GGTGACATGTTTGTAAACGCCGTGCATATGCTTAAATAAG
BL02255	*sigE*	BL02237	*spoVD*	CCGTTCTATCAAAACGCTCCGAGGCATAAAATGAAACAAG
BL02255	*sigE*	BL02236	*pbpB*	TCCAGTAGTCAGATACTCGTTAAACCGAATATTTCATCTT
BL02255	*sigE*	BL02214	*ydcA*	GTGATGGGTTATATACTCGCACAGAGTCTGTTATACGTCT
BL02255	*sigE*	BL02200	*ydcC*	TCTGCATATTGTCTGAGACACGCTCATATATTGGGTAGTG
BL02255	*sigE*	BL01705	*ycgF*	TTGTTCTATCATGTCCTCTTTTTGAATACATTGTACAAGT
BL02255	*sigE*	BL02179	*yusK*	TCAGCATCAATCAGGATCACTTGTTATACGATGCCAAAAA
BL02255	*sigE*	BL02178	*yusJ*	AAGCTTAATCAAATTAGTGTTCCCCCTACCTTGTTACCTT
BL02255	*sigE*	BL02146	*yunB*	GAATCGTATATCCCCTCTTACAAGCATACACTTGTGATGT
BL02255	*sigE*	BL01690	*cwlJ*	GAAGAATAGCTTTCTTTCATGGGCAATATCAAAGATAAAT
BL02255	*sigE*	BL02090	*spoIIP*	TTGTTCTACTTCCTCTATCTTGTACATAGTCTATTTACTA
BL02255	*sigE*	BL01540	*spoIIIAA*	CTTGTCATAATGTCTTCCTTTGATCATACATTTTTATAGA
BL02255	*sigE*	BL01534	*spoIIIAG*	TTACAAAAAACGGACAAGTTCAGTCAAACGAAATAAAAAA
BL02255	*sigE*	BL01420	*prkA*	AAAGCATCTAAAGGGACATCTTCGCATAGAGTGAATGTAA
BL02255	*sigE*	BL01419	*glgB*	TTTTTTTATATTCAAATATGACATCAGTCAATGTTTAAAA
BL02255	*sigE*	BL01381	*phoB*	TCATCATAAGCCCATGAAAATTTACATAAGATTAATAGAA
BL02255	*sigE*	BL01342	*asnO*	CAATTCAACTATTATTTTAAGGTGCATAAAATAGGTAAAG
BL02255	*sigE*	BL01133	*spoVB*	CGAACCTGCTACATACGCGTATAGAATAACATATACTAAT
BL02255	*sigE*	BL00642	*spoIVFA*	CCGTCATAATTCCCTGGGACAAGGCATATCATTGAACAAA
BL02255	*sigE*	BL00629	*spoVID*	TGTTCATATTCTGCCTGCTTTTCACATACATCTTTACTGA
BL02255	*sigE*	BL01098	*yjmC*	AGTGTATTATCCCGGTGAAATTCAAATAGATGTGATGAAA
BL02255	*sigE*	BL01097		AAGGACATACACCAGAAATCGTTGCAGGTAATGTGAATAA
BL02255	*sigE*	BL01016	*cwlD*	TAATCATATTTCCTTTCCTTGTCCCATATCTTGTAGTAAG
BL02255	*sigE*	BL01011	*pdaB*	TTGTTATGTTCTATTTTAAACAAGCATAGGATGAAAACAA
BL02255	*sigE*	BL00552	*yknT*	AAGGCATGGCGGGTTTCATTTATGCATATTGTAGTTGTAA
BL02255	*sigE*	BL00455	*ydhD*	GGCGCATGATCCTTCTGCAACGGTCATAGACATAGCATAA
BL02255	*sigE*	BL05250	*spoIIM*	AAGTCATGCGGCTCTCTTTTCTCTCATACAATCTATTAAA
BL02255	*sigE*	BL00396	*phoP*	GAAAGCAATTTTATCCTCACTTTTGCAAATAAATTGTTTT
BL02255	*sigE*	BL00383	*ytxC*	TACGTCTATTTTATTTGTCACCCCCATATACATGTAACAG
BL02255	*sigE*	BL00362	*ytvI*	TTTCTCATAATAACAGGACTCAAGAATAAAATGAAATAAG
BL02255	*sigE*	BL00314	*gerM*	GTCTAATTCAAAAACGGGCTCGTATACATAATAGTACAAA

BL02255	*sigE*	BL00291		AATGCATATTCCCAAGCTTTCACAAGTAAAATAGAGTAAA
BL02255	*sigE*	BL05091	*yhaX*	TTCACTAAGTGAACAAATAATGTTCATAAAGATTTAAAAA
BL02255	*sigE*	BL05056		ACGTATTTGCACAATCATTACTTTGAGATATTTTCAAAGA
BL02255	*sigE*	BL00102	*yyaD*	AACAAATGTCATATAGGTCGACAGAGAAAAAAATACGAAA
BL02255	*sigE*	BL00002	*asnB*	GTTTATAAAATTATACTTGTAAAAAGTTCAAAGAACTTTG
BL02255	*sigE*	BL03927	*mmgC*	GAGGAATTTGTTATACAGTCGCGTCCTGCTGAGCCTTTCT
BL02255	*sigE*	BL03925	*mmgA*	GCCTTCTTTTATCTTCGGTATGTACAAATCATGATAAGGG
BL02255	*sigE*	BL03662	*cotE*	AAGTCAAGTTTATCCTTTTACCTGCATACACTTAAACAGA
BL02255	*sigE*	BL04005	*spoIID*	CAGTCATATTAGCTTGTCCCCGCCCATAAAATGAAAGAGA
BL02255	*sigE*	BL03557	*ykvU*	AGAAAATAATTTTTAGACTTGTCTCATATGATGGGATAAA
BL02255	*sigE*	BL03545	*ykvI*	TCCTTCTTTTTGAAGGACATGGATCATATGGTGTAGAAAG
BL02255	*sigE*	BL02999	*ctaA*	AACACTTGAAAAATACTGAAAGAAATGTCTACTTCCTTTT
BL02255	*sigE*	BL02930		TTGGTCTAACCTTGTCCCTCTCTTCATACGCTTTAATAGA
BL02255	*sigE*	BL02893	*spoVR*	CTTTCATCTTTTCAAAACCCGGCTCATACAATAAAGAGAA
BL02255	*sigE*	BL03237	*spoVK*	TGGTTTGTCCGCACAGCTTACCTGAATACAATAAAAAATA
BL02255	*sigE*	BL02788	*spoIVA*	TCAGTCATGAACCTTTCTCCTCGGCATACAATGAGGAGAA
BL02255	*sigE*	BL03189	*yhbH*	TTTGTCAAAAGTTATTTGCACCTGCATAAGATAACAATAA
BL02255	*sigE*	BL03175	*yhcP*	CAGGGCACTTCCTGAATCATCCCGCATAGAGTGTAGTACA
BL02255	*sigE*	BL03001	*ylbJ*	TGGTCTAAAATAAGGACCAGCCTTCGTATAATGATAAAGA
BL02255	*sigE*	BL01986	*yngJ*	CTGGCATGAAAAATGCACTTTTTTAATATAATGTTACAAA
BL02256	*sigG*	BL03962	*ywhE*	TTGTTTAAATCCTCCTCTTTTTCCTCATAATGATAA
BL02256	*sigG*	BL03699	*nfo*	AAAAAATAAAAAGACAGCAAAAAAGAAAGCAATTTA
BL02256	*sigG*	BL03571	*splB*	CGGTATATCACATCCTCTAAATGCAAAAAGTAATGG
BL02256	*sigG*	BL03525	*ydfS*	ACGAGTAGAACAGATTAAAGTTCATAAACAAAAGAA
BL02256	*sigG*	BL03390	*ctpB*	TTACATGATACGGGCGGTTTTGGCGCATACTACATC
BL02256	*sigG*	BL03282	*yoaR*	AACTAAGAAACACTTTTTACCCCCTAAAATTTTTCT
BL02256	*sigG*	BL02732	*ponA*	TTAAATAAAAAACCTTCGTCTTTTTAATAAAATAAA
BL02256	*sigG*	BL03170	*yhcV*	AAAAATCGTACACGTACTGGACATGTATGATTGTTG
BL02256	*sigG*	BL03159	*yhcQ*	TTGAATAAACATAGCGGCAAAAACGAAATATAATAG
BL02256	*sigG*	BL02536	*pbpD*	TCGGATAAAAAAACTGCCAATCTATCAAATTATCCT
BL02256	*sigG*	BL01976	*yitG*	GGCTAAAAAATGGGACTAACCGAAGAAGAATAAAAA
BL02256	*sigG*	BL01868	*glcU*	TGTGTTAAAAATCGCATCGATAGTGCACCTTAATAG
BL02256	*sigG*	BL02256	*sigG*	ACGTATAAAAAGGGTGGGCTCCTCTATGACTTGAAA
BL02256	*sigG*	BL02222	*sleB*	CGTGTATAAAATCTGCCTCGCTACAAAAGATATGGT
BL02256	*sigG*	BL01751	*gerKA*	TTTCATAAAGCACTATTCACCCGGGAATAGTATAAA
BL02256	*sigG*	BL02089	*gpr*	CGGGAATGTTTGCAGACCTTGTGGGCACACTAACAC
BL02256	*sigG*	BL01519	*spoIVB*	AGTTATAAATCGCCGTCCGGAAGGCAAAAGTAATGA
BL02256	*sigG*	BL01439	*ctpA*	AAAAAACATACGTATACGTTTTCGCCTTATTACCAA
BL02256	*sigG*	BL01414	*yteA*	AATGAAAGTTCTGTTTGAATTCGTTCATACTAAAAA
BL02256	*sigG*	BL00896		ACCAAGAAAAAACAAACTGCACCAAATTTAATGGTG
BL02256	*sigG*	BL00833	*yvaB*	CCAGAATATTTGCCTGAAAACGTCCCATACTAAGCA
BL02256	*sigG*	BL01202	*yitG*	ATACATAAATAAGCCCGCATCATGGAACAATAACAG
BL02256	*sigG*	BL00779	*spoVA A*	GCCAAAAAAACACACCATTAAATACCAGAAAACGCG
BL02256	*sigG*	BL00778	*sigF*	GCAAATAAACTATACAGGCCCGACAAATTTTAGTAA
BL02256	*sigG*	BL00775	*dacF*	TCCGTATAAATCCAACCGGATTGGAAAAAATAGAGA
BL02256	*sigG*	BL00747	*gerAA*	ACAGTATATTGTTTTTTTAACAGGAAAAGATAACCT
BL02256	*sigG*	BL01071	*pbpF*	AACTATCATATTGTTTTCCTGCTAAATTTTAAGATA
BL02256	*sigG*	BL01012	*gerD*	TATGTATAAATTTTTAAAAGACATTCATATTAAAGG
BL02256	*sigG*	BL00514	*spoVT*	AACGTTAAAAAGCAAACGATCGCTTCATCATAAAAA
BL02256	*sigG*	BL00448	*yveA*	CTTTTTGAACATACTACCTTCTATTAAATGTATTAA
BL02271	*pyrR*	BL02273	*pyrB*	TAAAAACCTTTTAAATGAAGTCCAGAGAGGCTTGAAAGGGTTGCGGAGA GAGAA
BL02271	*pyrR*	BL02272	*pyrP*	TAATCCTTTTTTAAATGCATTCCAGTGAGTTTGCAAAGAGGGATGGAACG AAAG
BL02271	*pyrR*	BL02271	*pyrR*	AATCAGATTCTTTAACACAGTCCAGAGAGGCTGAGAAGGATACCGGACA AGCGA
BL02312	*fapR*	BL03313	*fabHA*	TATTGCCAACTAAAAAAAAATAGGGTAGGATTAGTACCAGATACTAATA
BL02312	*fapR*	BL02884		AAAATCATGGTCCATTATTTTTTTTAACAATATTATTATTTCCCTTTGG
BL02312	*fapR*	BL02312	*fapR*	CATTCCAATCTATGAAATTATATACTACTATTAGTACCTAGTCATAATT

BL02312	*fapR*	BL01107	*fabHB*	AATGGTTATAGTATACCTCATATCAACATTTTAGTACCAGATACTAATA
BL02312	*fapR*	BL05113	*fabI*	ATAATACTAGTCCATGATTATTTTACAATTAGTAATCCTCCTAAAGTAT
BL02371	*yycF*	BL03556	*ykvT*	AACAATGACATTTTGTATACAAAAAAA
BL02371	*yycF*	BL02464	*tagD*	AGCTTAACAATATCTTTACATTAGATT
BL02371	*yycF*	BL01953	*yocH*	GAAATAGTCATTTTTTTGACATTGCTT
BL02371	*yycF*	BL01303		AGGAGGAGCATTTTGTTGACATTAGTT
BL02539	*malR*	BL05068	*yflS*	GGTCGGCAGTTTTTTTAATAAAATAAAATTTGAATGATAGCTTGAAAGGAG
BL02539	*malR*	BL00044	*malS*	TGGTTTGCTTTGTTGTTATCCATTCTAAGGGTAACTAACTTTTGAAAGGAA
BL02539	*malR*	BL03976	*maeA*	ATTAATTTTTTGAATTAATTCTTATGCTTTTTTAATTGATTAAAAGTAGCG
BL02539	*malR*	BL02959	*cimH*	AATGGTGATAAACAGGATTAAAAGAAATTTTTTAATATTTTTAATATAGAA
BL02539	*malR*	BL02543	*maeN*	CGCCCTTTTTCGTTTTAATTTAATTATCAATCGATTATTTTAATTTAAAAA
BL02604	*comA*	BL05244		CCCTCCGACGGGCGT
BL02604	*comA*	BL03934	*dltA*	AAGGCTTTTTGACGT
BL02604	*comA*	BL03760	*pel*	AGCGTGATTCCCAAA
BL02604	*comA*	BL04024	*dhbF*	TGAAAGATGGCGAAA
BL02604	*comA*	BL02867	*rapI*	GGCAGGCTGCAGAAA
BL02604	*comA*	BL01908	*rapE*	AAAGCCATAGTGCTT
BL02604	*comA*	BL01728	*lchAC*	GATGCCGTAGGAAGC
BL02604	*comA*	BL01727	*lchAB*	GACGCGGTAAGGCTT
BL02604	*comA*	BL01726	*lchAA*	TAAGCCATAGGGCGT
BL02652	*licT*	BL00213	*bglP*	GGATTGTTACTGCTTTTGCAGGCAAAACCTAAG
BL02652	*licT*	BL01947		GGATTGTTACTGATGATGCAGGCAAAACCTAAA
BL02652	*licT*	BL02334		GACATGTGACTGGTCATGCAGGCAGGATCTAAA
BL02699	*sigW*	BL02713	*pbpX*	TTGAAGAAACTTTTTAAGGGGTTATTGCGTCAGTA
BL02699	*sigW*	BL02699	*sigW*	TTTAATGAAACCTTATAAAAACCAATTCGTATACA
BL02699	*sigW*	BL02657	*pbpE*	TTAGTTGAAACCTCATATTTCCATATGCGTATAAA
BL02699	*sigW*	BL02506	*xpaC*	GAACTGAAACTTTGCTTAGCCGATATCCGTATTAG
BL02699	*sigW*	BL01694	*yceC*	ACCGCGAAACTTTTCTCTGAATCTTTCCGTATATA
BL02699	*sigW*	BL01412	*yqeZ*	AAGATGAAACCTTTGCTACATGTATAACGTATGTA
BL02699	*sigW*	BL01299	*pspA*	AAAACCGAAACTTTTCCTCTGGAAAATAGTATATT
BL02699	*sigW*	BL00798	*yqjL*	CGTGAGAAACGAACATCCTTGTTTGTGCGTCTTTC
BL02699	*sigW*	BL00480	*yeaA*	ATCCCTGAAACTTATTTCCCTTTGTTTCGTATATA
BL02699	*sigW*	BL00425	*sppA*	TCATAATGAACTTTTTCTCTTCATATACGTACTAA
BL02699	*sigW*	BL03845	*ywaC*	TATTCTGCTTTTACAACATATTTTCAAAGTAAACA
BL02699	*sigW*	BL03328	*yjbC*	AAATTTGAAACCTTAGCGTAAGACAACCGTCTTAG
BL02753	*birA*	BL02411	*bioW*	TGTAATGTTAACTCTAATTAATAATAGGTTAACATATGAA
BL02822	*lrpC*	BL02822	*lrpC*	AAAATTTTGTAGTTT
BL02851	*sigM*	BL03329	*spxA*	TCCATACATACAAAGATGAAAAATGGTATGGAGAATCTTA
BL02851	*sigM*	BL02851	*sigM*	TCACGTTGAAGTTTGGAAAGAATACGCAAATATTGTATCT
BL02851	*sigM*	BL02453	*bcrC*	ATGAAACTTTTGCAAAATGAAATACGTCTCCTAACTGTCT
BL02906	*hpr*	BL01845	*yclF*	GATAAAATTTTATTAAAAAA
BL02906	*hpr*	BL01118	*epr*	CAAAGAAATAAATAGAAGCA
BL02906	*hpr*	BL01111	*apr*	ATAAAATTATCTACTAGCTG
BL02968	*ylaC*	BL02966	*ylaA*	TATGATTTTATGAAACAATATGCAGCGTCGTTTGTCTATAAAGATGAGGCCTTAA
BL03016	*acoR*	BL03012	*acoA*	TGACTGCTTCATCTGTTTAACTCTGTAGACAGAGCAAAACAGAGAACAAGACTTTCCTACAGGC
BL03070	*treR*	BL00914	*sacX*	AAATATATACGTCCTATATATCCACCAAACGT
BL03070	*treR*	BL03890	*sacP*	GAATTTTAGGTCTTTTCTGCTTTTTTTTAAAA
BL03070	*treR*	BL03068	*treP*	AACCTATAACATGTATATACAGGTTATAAGAG
BL03125	*levR*	BL03124	*levD*	GCCGTTAAAAAAGTACATTTGGAC
BL03234	*glnR*	BL03234	*glnR*	TGTAAAGAATCCTTACA
BL03234	*glnR*	BL01812	*nasA*	TGTTAGTAAAGCTTACA
BL03234	*glnR*	BL01769	*nasD*	TGTCATAAAAAGTCATA
BL03234	*glnR*	BL01767	*nasB*	TGTCAGGAAATGTAACA
BL03258	*ctsR*	BL03605	*clpP*	AAACTGGAAATAACTG
BL03258	*ctsR*	BL03258	*ctsR*	GTCAAATATAGTCAAA
BL03273	*sigH*	BL02250	*ftsA*	AAGAGGATATACATAGGATATCACGAATATTCACAGT

BL03273	*sigH*	BL01518	*spo0A*	AAAAGGGAATAAAGTGGTGCTGTCGAATTAAACATAT
BL03273	*sigH*	BL01448	*cwlS*	ATGAGAAATTTGTAAAAAAACAAAAAAGCTATTAAAA
BL03273	*sigH*	BL00776	*spoIIAA*	GAAGGAATTTATAAAGTCTGAAGCGAAACACTCATTA
BL03273	*sigH*	BL00732	*fumC*	GAAAGAATACGGTAAAACAAATCCGAAATATAAATAT
BL03273	*sigH*	BL00640	*minC*	AAGAGGATTTTATGAGCCGATGTCGAAAAGAGTATGA
BL03273	*sigH*	BL05376	*ansB*	TTTCGATGCTACGTAAACTTTCGTAGTTAATAGGAAT
BL03273	*sigH*	BL03968	*spo0F*	AAAGGAAAAGAGAAAAACAAAACAGAATAGATAAACT
BL03273	*sigH*	BL03681	*dnaG*	TGAAGGATTTACGTACTAAGCAGCGAATTATGTACGA
BL03273	*sigH*	BL03576	*kinA*	GAAGGAGATTTCTCGCATTTTAGCGAATCCTTTTAAG
BL03273	*sigH*	BL02922	*ansB*	TAAGGAAATTTACTACAAATAGATAAATGAAGTAAAG
BL03273	*sigH*	BL03380	*yvyD*	GCAGGAATTCAGCAGGGAAAAGACGAAATATATTTAC
BL03273	*sigH*	BL02892	*lytE*	TTACGACTTCATTATAAAAATGTAAAATAAAGTATTT
BL03273	*sigH*	BL02874	*yoxA*	ACTGTTTAGTATGTGGAGTAATCTTCTTTAGAGAAAG
BL03361	*degU*	BL03890	*sacP*	AGAAAAGACGAAAAAAAATT
BL03361	*degU*	BL03068	*treP*	TTTGAAAGCGCTACAAAAAT
BL03361	*degU*	BL01607	*sipT*	TTAAAAATAAATTCTCAAAA
BL03361	*degU*	BL00914	*sacX*	AACAATGACCATTCCGTCCG
BL03361	*degU*	BL01118	*epr*	CACAAAAATTGCAAAATTTT
BL03361	*degU*	BL01111	*apr*	AAAAGATATTATTTAATTGT
BL03361	*degU*	BL01090	*comK*	TTCAAATTAATTGGAAAAAA
BL03454	*sigL*	BL03124	*levD*	GTTTTGTTGGCACGATACTTGCATTATAAATAAGCG
BL03454	*sigL*	BL03012	*acoA*	GGAGAGCTGGCACGATTTTTGCATCATAACGTGTGA
BL03454	*sigL*	BL02564	*gbsA*	ACGGGGATTGTCCATAACTTGCCTCCTTTCACAGGA
BL03454	*sigL*	BL02341	*yfmT*	ATAACAGGTTTATATATCCTGCAATATTTATCAATG
BL03454	*sigL*	BL02226	*gudB*	AAAGTGTTGACATTATATATGAATTTCAAACGTTTG
BL03454	*sigL*	BL01765	*gabD*	TCTTAAGTGTTATCTTCCTTGCCTTTTATTTGCCAG
BL03454	*sigL*	BL01737	*rocD*	TAAGAGTTGGCACGATACTTGCATATAAGATAGTGT
BL03454	*sigL*	BL01710	*ycgN*	AAAGGCATGACGAAGAAATAAAACGATAAAAAGGGG
BL03454	*sigL*	BL01500	*ptb*	TTCATACTGGCACGAAACTTGCATGATATAGTGGGC
BL03454	*sigL*	BL00585	*dhaS*	CATCGGTATGTATGCACTTTACTTTTTTAATGGTTT
BL03454	*sigL*	BL00293	*mmsA*	AAAAGATATGTCTGTTTTATACAAAAAAATAAGCGA
BL03454	*sigL*	BL00246	*iolA*	GTAAGTATGTATTACCAACTTTGGACGGGGTAGACA
BL03454	*sigL*	BL03906		CATAATATAACACGATATAAAAAATAAAAAAGCGGA
BL03454	*sigL*	BL03244	*argD*	CGTAAGCGGGAAAGGATCTTTCCTGACGAAAGACGA
BL03463	*cggR*	BL03463	*cggR*	CTGAGGTGGGACGTTTTCAGTCACTGCGGGACATTATTCGTCCAGG
BL03529	*sigI*	BL03529	*sigI*	GCGGGAAAACCCCCTTAATCGTCAAACAGATCACGAATTGTTCTAGGAGATCAA
BL03587	*ccpC*	BL00399	*citZ*	TTTATTCTTTTTTTT
BL03587	*ccpC*	BL02940	*citB*	GATATTTACTTATGT
BL03587	*ccpC*	BL02891	*citA*	GAAATTTTCGTATTT
BL03625	*tnrA*	BL00679	*pbuX*	TCGACGATTTTTTGGCA
BL03625	*tnrA*	BL03767	*dppE*	AGGGATATTTTTTACCG
BL03625	*tnrA*	BL03625	*tnrA*	TGTTAGATTTTCTGACA
BL03625	*tnrA*	BL03323	*oppA*	ACCGTCTTTTGGACTGT
BL03625	*tnrA*	BL02481	*ywrD*	CGTTAGTTTTTCTTCCC
BL03625	*tnrA*	BL02456	*nrgA*	ACATTCCTTTAGACTGT
BL03625	*tnrA*	BL01812	*nasA*	ACAATCATTTCGAATGT
BL03625	*tnrA*	BL01769	*nasD*	ACAGTATTTTTCAGTAT
BL03625	*tnrA*	BL01767	*nasB*	ACAGTCCTTTACATTGT
BL03625	*tnrA*	BL01510	*artM*	ATAGTCTTTAGTAGTCT
BL03625	*tnrA*	BL01180	*alsT*	ACAATCTTTTGAATTGT
BL03625	*tnrA*	BL00608	*ilvB*	TTGATTATTTTTTGAAA
BL03625	*tnrA*	BL00261		ACACTATCTTATAGAGT
BL03625	*tnrA*	BL00146	*ydaP*	TGTAACGCTTTATTCCT
BL03625	*tnrA*	BL03989	*ywlF*	ACGCTTCTTTAGAGTCC
BL03625	*tnrA*	BL03760	*pel*	ACAGTATTTTAAACTTT
BL03625	*tnrA*	BL03234	*glnR*	ACATTTCTTAGGAATGT
BL03625	*tnrA*	BL03112		CCCTTCTTTTCTACTGT

BL03625	*tnrA*	BL02624		TGATTTATTATTTTTCA
BL03625	*tnrA*	BL03067	*yflA*	ACTCTCTTTTTGTCTTT
BL03625	*tnrA*	BL01971	*gltA*	TGTAAGATTTTATGACC
BL03625	*tnrA*	BL01785	*glnT*	TCTCAGTATTTTTTAAA
BL03625	*tnrA*	BL01460	*yodF*	ACACTAATTTAGACTGT
BL03678	*ccpN*	BL00839	*pckA*	AATGAAGAAACGGTATAGACTATAAGAGTAAATGTGTTATACTAATTCTA CATTAAT
BL03678	*ccpN*	BL00390	*gapB*	ACCATACCGATTAATTTTTATTACACAATATGAATAAAACTTATTTAGTAC TACACC
BL03678	*ccpN*	BL03464	*gapA*	AATATAAATTAAATATCTCTTATTTACTTAAAGGAGGAAATCATCATGGC AGTAAAA
BL03682	*sigA*	BL02106	*des*	GTAGTAACATTTTCTTGAAACAAAGTAATTAGTTA
BL03682	*sigA*	BL02102		TTGATATAATTTCAGCAAAAGCTTCCAAGTCTTCT
BL03682	*sigA*	BL01694	*yceC*	TATGTAGTATAGTTTCCGCGACTGAAGCACGTTAG
BL03682	*sigA*	BL01665	*mtaB*	GCTGAACAATTTTTAAAAGACCTGGCAAAGTGATT
BL03682	*sigA*	BL01660	*ydaB*	GTTTTACAATGGATATATTTTGTTTTAGAATGAAT
BL03682	*sigA*	BL01646	*ybcD*	TATGGCGTTGAATATCAGTTTGTCATATACGATAG
BL03682	*sigA*	BL01643	*phoD*	TATGTAAAACAATTTTGTAGCAGAATGTATTTCAG
BL03682	*sigA*	BL01637		ATCTCGCATTTTATCAAGGTGCTTTTACAATAGAT
BL03682	*sigA*	BL01618	*pdhC*	AAATTAATCTAGACTTGTTAAGTTTTCCCTTTTCG
BL03682	*sigA*	BL01616	*pdhA*	TTAGTTTGTCATAAACTATAAGTGATACAGTTTTA
BL03682	*sigA*	BL01611	*kinC*	GTTATAGAATCTTTTTATCTTTGTTATCTCCTTCA
BL03682	*sigA*	BL01607	*sipT*	AATTGCCCTTGCAACGACTTTCTTTAAAAATAAAT
BL03682	*sigA*	BL02098	*dnaJ*	ATTGAAAAAACATAATTTGAAATGATACAATCTAA
BL03682	*sigA*	BL02094	*hrcA*	AATTGACATTTTGATTTCGGTTTGGTAATTTTGAT
BL03682	*sigA*	BL02092	*lepA*	CATTGAATCTTCACAACCCTATTGATATAATCTAA
BL03682	*sigA*	BL02041	*cymR*	AAATGAAAGTTTATATAAAGTGTGCTATAATAACC
BL03682	*sigA*	BL02040	*yrvO*	TAAATAATATCCAATCTCCACAAATAAAACTTTTA
BL03682	*sigA*	BL02021	*yrrT*	TATTTAGTATGATATGAATATCCTTAACATTTTTA
BL03682	*sigA*	BL02019	*yrhA*	TCTTGTGCTTCGCATGATTAAAAGAATCAATTAAA
BL03682	*sigA*	BL02018	*yrhB*	GCTTACTATAAAACGATTGCCCCGGAAATATGGAA
BL03682	*sigA*	BL02016	*aapA*	GATCAATTACTTCAAATTTTAATGTTATAATAGAT
BL03682	*sigA*	BL02004	*gltC*	GGTTTTCAAATCTATACAAACAATATATAATTTAG
BL03682	*sigA*	BL01559	*opuAA*	AATTTATCATTTAAGTAGTTTAAATTTTCATTTTT
BL03682	*sigA*	BL01552	*sinR*	ACCTTAATATTATCTTGATTTCTCTTTTCTCTTTG
BL03682	*sigA*	BL01524	*ispAB*	ACTCGCAGTTTCAAACGGTTCTTGGTAAACTGAAT
BL03682	*sigA*	BL01518	*spo0A*	CCTTTTTTTCTTGGGCGTTGTGCGTTAAAATGAAA
BL03682	*sigA*	BL01498	*bkdR*	GAAATAACAGATAAACTTTTTTTTCAAACAGTTCT
BL03682	*sigA*	BL01476	*purE*	CGTTGACATTATCACAGTCCGTTGTTAAGATAAAC
BL03682	*sigA*	BL01452	*sucA*	TGTAGAAACAAATGAGAAACAGTGGTAAAATGTAC
BL03682	*sigA*	BL01448	*cwlS*	ATTTGTAAAAAAACAAAAAAGCTATTAAAAAAATA
BL03682	*sigA*	BL00958	*ymfP*	CGTTTACGTAAAAGTATAAAGGTAGTATAATAGAA
BL03682	*sigA*	BL00946	*ywcJ*	TGTGAAATGTTTCACAATATCCTGCTATACTTAGG
BL03682	*sigA*	BL00923	*cydA*	ACTTTATTTAAGAAGGAATTCGCGGTAAAGTACAG
BL03682	*sigA*	BL00920	*guaA*	TCTTGACCGCTTTCAGTCGTATTGTTAGAATAAGT
BL03682	*sigA*	BL00914	*sacX*	TTTTGATCGTCTTCTGTTGATTTGATAGGATAGAA
BL03682	*sigA*	BL00913	*citM*	TATTTACTTCAGAAAGTTCTTCTCATATGATGACA
BL03682	*sigA*	BL00903	*aprX*	CTGATCTTATAGTACATAAAACTAAGCAAATTTGG
BL03682	*sigA*	BL01383	*yqjI*	ATTTCAATTTAAATAAAAATTAAACTATAATGCTA
BL03682	*sigA*	BL01381	*phoB*	ACTTTTAAATGTATTCTAATTATCTTTCGTTTTCA
BL03682	*sigA*	BL01363	*rapG*	AATTTATACAGGGAGGAAAATATAGTATGATATGA
BL03682	*sigA*	BL01351	*addB*	AATGGTTATTTTGCGGATGTTTCGATAGAATAAAG
BL03682	*sigA*	BL01325	*yngIA*	AGTACCTGGTACTAGAACAAAGTGCTATAATGGAA
BL03682	*sigA*	BL01303		TTTTAAATATATACATTTTAAAAAAAATCATTTAG
BL03682	*sigA*	BL00858	*pabB*	CTTTTCCTAGAAAAAAGATATGCATTACAATAAGG
BL03682	*sigA*	BL00857	*cysK*	GATTGACAAAATATTTTGATATTGATAAATTAGAT
BL03682	*sigA*	BL00852	*ftsH*	GTTGTATTGGAACGATTTTCTATGATACTATTGAA
BL03682	*sigA*	BL00839	*pckA*	TGATTAAGATGTAATTACAATATGTGTAATTTTTT
BL03682	*sigA*	BL01279	*xerC*	ATTGCCAGATGAACTTGTTATGTGATAACATTTAA
BL03682	*sigA*	BL01275	*flgB*	AATTTGATATAATAGCATTAAAAAAGTATAGTTTT

BL03682	*sigA*	BL01246	*sigD*	TGTTTAAATTTAAGATGACAAATGATTTGATGAAA
BL03682	*sigA*	BL01227	*nusA*	TATATAATTTTGCATGCTTGGATAACTTCCGTTAT
BL03682	*sigA*	BL01210	*asd*	CAATCACCACGGCACACATCTATGTTAAAATAAAA
BL03682	*sigA*	BL01209	*dapG*	TAAATCAAATTCTAAAGATTGCCCTACACTCTTCT
BL03682	*sigA*	BL01204	*spoIIIE*	AAGGTTGTATAGCCTTCTTTTTAAATTTCTTGACT
BL03682	*sigA*	BL00772	*drm*	ATTGTCAACTGTTGTGTAAACGGTTTATACTTGGC
BL03682	*sigA*	BL00771	*xerD*	GTTTTACCGTCTTGTTTTATTATGCTATGATATGG
BL03682	*sigA*	BL00751	*mrgA*	GTATTGATTTTAATTTTGTATTTGTTATAATATAA
BL03682	*sigA*	BL00750	*cssR*	GGATAAAAATGAAAAGAATATGTGAAATTATGAAA
BL03682	*sigA*	BL00734	*htrB*	AAAGTATTAAAGTGTATAAGAAAAGTAAAAATAGG
BL03682	*sigA*	BL00732	*fumC*	TATTGAGCTAAAATTAAAATAAAGATAAAATTTAA
BL03682	*sigA*	BL00708	*uxaC*	AATGGAAAAGAAGGCGCGGTTAGTATAAAATAAGA
BL03682	*sigA*	BL01182		GTAGTTAAATTCTTTTCTCCCAATAATACAATTGT
BL03682	*sigA*	BL01181	*ywtG*	GTAGTCAAAAGTTTTATAAATACTAAAACATTCGC
BL03682	*sigA*	BL01178		TCTTTTAATTCATTGTAAGCCATGATAAAATATGT
BL03682	*sigA*	BL01172	*narG*	TTTCTACAACTTCTAAAAACCTAGTTTAAATTTCT
BL03682	*sigA*	BL01167	*fnr*	ATATTCATATGTTCACAATATTGTTTATATTTAAA
BL03682	*sigA*	BL01166	*narK*	GTTTTTCTTCTTGGGTAAGCTTTATAAACATAAAA
BL03682	*sigA*	BL01158	*spo0B*	GTTTCATTTATACTGCCTTCTCTGTTATAATTCAA
BL03682	*sigA*	BL01155	*nadB*	ACTTGAGTTTATTTTATCGATGTGTAAATATAGGT
BL03682	*sigA*	BL01136	*nifS*	TGGATATAAATGTGTAGCTATTTTATTTGAGTTCA
BL03682	*sigA*	BL01118	*epr*	TCTTTTAATCTATTTTTTATTTAATTAAATTGTAG
BL03682	*sigA*	BL01111	*apr*	TCTACCATATAATTCATTTTTTTTCTATAATAAAT
BL03682	*sigA*	BL01109	*gltT*	AATTTACGATCTCAGTTTAATATTCTAGTATAAAA
BL03682	*sigA*	BL01107	*fabHB*	AATATCATATGGAGTATAGTTGTAAAATCATGGTC
BL03682	*sigA*	BL00680	*xpt*	CCTTTTCATTCGGAACGAAAGTTGATATAATTTGA
BL03682	*sigA*	BL00679	*pbuX*	TGTTGACGGCTTCTGTCTATTCATTTACAAAGAAA
BL03682	*sigA*	BL00661	*resA*	GATTTAAAAGTGTGTTGGAAGTTTTTTCCGTTTAC
BL03682	*sigA*	BL00658	*resD*	GCTTGGCTTACAATCGCTTTTCTAATAAAATGAAT
BL03682	*sigA*	BL00652		ATGATACCATTTTTACTCTTTTCCTTTTGATTTCT
BL03682	*sigA*	BL00651	*sigX*	CCTTTCCATTTTCATATGTTCGTAATATAGTTGTA
BL03682	*sigA*	BL00631	*valS*	ATTGACGAAAATGAAAAAGATTTAGTAATATAAGA
BL03682	*sigA*	BL00623	*hemA*	TATTTAAGATATAATATTAGTAAGATTAAATCTAT
BL03682	*sigA*	BL00621	*lonA*	ATGTTAACATGTCCTTATTTTTTTCATATAATATT
BL03682	*sigA*	BL00619	*clpX*	AATTGATTTTCTTGTAGAAAACCGTTAATATGTTT
BL03682	*sigA*	BL00608	*ilvB*	CTAATAAAAAACTTTTTGTTAACACTTACTTGTCA
BL03682	*sigA*	BL00605	*opuE*	TGACTAGTATAGTTTAAAATCGGAAGATTTAGTTG
BL03682	*sigA*	BL01095		TAAATTAAACAATAAAAAACTCAAAAATACATTAG
BL03682	*sigA*	BL01094	*pucH*	TTTTTAAAACTGGGATCATTACCGCTAAAATTAAT
BL03682	*sigA*	BL01090	*comK*	AATATCATATACCTAATGTGTTATTTTTTAGTTAT
BL03682	*sigA*	BL01080	*yhfL*	CCTTGAGTGTTTTCTGTTCATTTACTATAATGGGA
BL03682	*sigA*	BL01071	*pbpF*	AAGTTAAAATAGTGGGGAAAAAACTATCATATTGT
BL03682	*sigA*	BL00585	*dhaS*	AAGGTAACATCGCAAAAAAGCGGCTTTACGTTACG
BL03682	*sigA*	BL00561	*mmgD*	TGTTTACAATTTTTTGCTTATTCTCCACCTTTTTA
BL03682	*sigA*	BL00550	*ytcI*	AATACAATTGTATGAAAGGATTATGTATAATAAAA
BL03682	*sigA*	BL00520	*glmU*	GCTTGAAATCAACTCATATTTAGGATATATTTTTC
BL03682	*sigA*	BL00506	*spoIIE*	TTTTATTTTCTAATAAAATTTTTATTAAAATTTCT
BL03682	*sigA*	BL05377	*ansA*	TCCTGATGTATCAAGCCTTCTTTGCTAAAATAAAT
BL03682	*sigA*	BL05376	*ansB*	ATAGTAGTATATTTTTCTTTCGTCCCAATTTTTTA
BL03682	*sigA*	BL05309	*yciC*	AGATTTATATTGTATAAAGAACATAAAAAAAGTGT
BL03682	*sigA*	BL05301		CGTTTGAAATAAACATCATAACAGCAAAAATAAAT
BL03682	*sigA*	BL00440	*acsA*	AATTGAGAAAATAAAAAATCTATACTATAATATAT
BL03682	*sigA*	BL00439	*tyrS*	CATTGACAATGCAAAAATATTATGTTAAAAAAGAT
BL03682	*sigA*	BL00433	*ezrA*	AATGTACCATAGTATTCTTGTTACTTTTGCTATCA
BL03682	*sigA*	BL00432	*nifZ*	ATTTTCTGTCAATTTATTTTCGTGATATAATATGA
BL03682	*sigA*	BL00419	*ackA*	CTCTGATCTATTTTTATATTCAGAAAATAATAATT
BL03682	*sigA*	BL05281	*adaA*	GTTTGAAAATTCATGTTTTACCTCCTAAAATATAG

BL03682	*sigA*	BL05249	*fur*	GTAATTAAATATTACTAATAATTTTTTATAAGTAA
BL03682	*sigA*	BL05245	*ppiB*	AAAATAGTAAGTACTAAAAGTGTCCGCTAAGTTGG
BL03682	*sigA*	BL05244		AATTGTGAAAAAATGACCCTTTATGTAAAATATAT
BL03682	*sigA*	BL00399	*citZ*	ATTTAACAAATGTCTGATAATTGTTTATAATATAA
BL03682	*sigA*	BL00398	*icd*	TGTTTTTCTATATTGAAGAATTACATATTATGGGA
BL03682	*sigA*	BL00396	*phoP*	AATTGTCGAACGGATAGGCTTTCGTTAAAATAGGA
BL03682	*sigA*	BL00390	*gapB*	GTATGGCTAATTAAAAATAATGTGTTATACTTATT
BL03682	*sigA*	BL00389	*speD*	CCTTGCAAAAAATATTCAAACAAAGTATACTATTT
BL03682	*sigA*	BL00382	*thrS*	TCTTGATTTTGCGTGAAAAACAATTTATAATACGG
BL03682	*sigA*	BL00376	*acuA*	TATATAATATCATATCTAAAAAATAAAAGAGTTAA
BL03682	*sigA*	BL00374	*rpsD*	AAAATAAAAGTGTTTTTATTTTGGTTTCCTCCTCA
BL03682	*sigA*	BL00357	*infC*	CATTGCTATTTGAAATGTGTCCTGATATAATAGCA
BL03682	*sigA*	BL00353	*abnA*	ATGTATATATTTGTAACAAGCTTTTTAACTGTTTC
BL03682	*sigA*	BL00352	*araA*	AAATTGACAGTTTTTTTCATAATGATATAATGAAG
BL03682	*sigA*	BL00340	*pheS*	AAAGTAATATTTTCCCCGAAAAAAATAACGGATTT
BL03682	*sigA*	BL00331	*lcfA*	CATTACTTTCTGAATATTGTTTTGATATACTCAAA
BL03682	*sigA*	BL00330	*ysiA*	TCTTGACAAAGCGTCCATTTTTATTTACGATTAAC
BL03682	*sigA*	BL00326	*trxA*	GTTAGCGTGAACCAAAGGGAGATGCTATACTAAAA
BL03682	*sigA*	BL00324	*lysC*	TGTTGAACTTTTAAAAAGAATCTGATAGAATTTGT
BL03682	*sigA*	BL00323	*sdhC*	AAAATCTGATATATTTATATTTTGATAAAATAAAA
BL03682	*sigA*	BL05177	*nrdI*	TATTGTGTTTTCTTGAATGATAGTATACTATATAT
BL03682	*sigA*	BL05164	*recA*	TCTTGGCAAATCCGCATAAACAAGGTATAGTAGAT
BL03682	*sigA*	BL05119	*ohrA*	TGTTGTGTTATCACTTTTAATTTTATAAAATTAAA
BL03682	*sigA*	BL00293	*mmsA*	AAAATATGTTTTTTTATTCGCTTGTCTTCTGTTGA
BL03682	*sigA*	BL00286	*manR*	TAGATAATATTTTTTAAGGATATTTTCTTTTTAAA
BL03682	*sigA*	BL00247	*htpG*	TGTGTAATATACTTCGCCTTACGGCGAACAGTTTG
BL03682	*sigA*	BL00246	*iolA*	AAAATAATATGTTACAAAGGGATTTAGTAAAATTG
BL03682	*sigA*	BL00229	*deoC*	AATTTACATTTGTTCAAAAATCGGTTATGCTGAAA
BL03682	*sigA*	BL00213	*bglP*	CGTTGACATGAACAAAAACGGGTGCTAAAGTATAG
BL03682	*sigA*	BL00200	*ahpC*	AATATTTATTATTTGATAATTTGTATATAATTATA
BL03682	*sigA*	BL05089	*glpF*	ACTGGAACTAAAAATTTAAAATGTGAAAAATTATT
BL03682	*sigA*	BL05014	*ybxG*	AATTGTATTTTTGGAAGGAGTTTATTATGATGGAA
BL03682	*sigA*	BL00198	*gntZ*	TAAATGAAATAACAATTCAGATAAACATATTTTTT
BL03682	*sigA*	BL00194	*gntR*	AAAATAACATGGTACCAATTAATTAATTTAATTGT
BL03682	*sigA*	BL00192	*galE*	AGATTTATATTACTTTCCGCGTGGGTTACACTTCT
BL03682	*sigA*	BL00190		ACAAAAAAATTGTTTCGATAAAAAATTTCTTTTTT
BL03682	*sigA*	BL00185	*glpT*	GTAGTAAGTTAGAAAATCAGGACTTTCGAATTTTT
BL03682	*sigA*	BL00184	*glpQ*	TGGATAGAATTTTCTATCCAAAGAAATAAAAATTT
BL03682	*sigA*	BL00182	*iolR*	GTTAAAATGATTTAGGGAAACATTGTATAATAAAA
BL03682	*sigA*	BL00146	*ydaP*	ATGGTAAAATTTTTCATTTTTTGGACGTCTCGGTA
BL03682	*sigA*	BL00136	*thrZ*	TGTTGATATTTACGACCAGTTATTATAAAATATAT
BL03682	*sigA*	BL00119	*iolT*	TATTGACTGAAAGCGCTTTTAAAATTATGATAATA
BL03682	*sigA*	BL00115	*bglD*	ATCCTGAAATTTTCCTCTAACTACTGTACAGTTTT
BL03682	*sigA*	BL00112	*spoIIIJ*	TAGATAATTTTGTTCTTTTACTGATGTAATAGTAG
BL03682	*sigA*	BL00082	*gyrA*	ATGTGTATAAAATCAAGATTATTGATATAATGGAG
BL03682	*sigA*	BL00076	*dnaA*	CATTGCAACCACTCGCTTATTCTGATATTATATTT
BL03682	*sigA*	BL00075	*perR*	TATTTGCAATATTACATTCTTAAAAAAAATCTTCT
BL03682	*sigA*	BL00070	*ytkP*	GATATAAAATTGTTTTTTATACCTGAATATTGGAG
BL03682	*sigA*	BL00060	*gltP*	AAAGTAAGTTTGTATTACATTCGCTATAGTTTTAT
BL03682	*sigA*	BL00049	*ytpT*	TATTTTACTTGTTTTGAAAGCTTATCTACATTTTT
BL03682	*sigA*	BL00044	*malS*	TGTAAAAAATGGTGCTACGGGAAATGTCTCGTAAT
BL03682	*sigA*	BL00008	*ytrA*	GATATAACATAATTCACATGATGTAGCTCATTATG
BL03682	*sigA*	BL03991	*glyA*	TTTTTTCTTTATTAGAATAATTTTTTAAACAAATG
BL03682	*sigA*	BL03980	*ntdA*	ATTTCCCAAAATAATAAAAATGGGGTGAAATTGAT
BL03682	*sigA*	BL03976	*maeA*	TAATTCTTATGCTTTTTTAATTGATTAAAAGTAGC
BL03682	*sigA*	BL03968	*spo0F*	ATAGTAATATATTGTTTTCTCTTATTACCCTTTTT
BL03682	*sigA*	BL03966	*pyrG*	TCTTGACTTTCACAGGCGAACTAGGTAGTATGTAT
BL03682	*sigA*	BL03962	*ywhE*	TTTTTCCTCATAATGATAAGAACATTTTATTTATC

BL03682	*sigA*	BL03936	*ydfJ*	TATTTCATTGTCCACTACAATGAGAAAGAATGTGA
BL03682	*sigA*	BL03934	*dltA*	TATTTCCACAATAACAAAAAATGTTTATAATAGGA
BL03682	*sigA*	BL03924	*ywjF*	GGGATAAAAGGACGTAAATACTTTTATATAACTGA
BL03682	*sigA*	BL03920	*speE*	TCTTTACAATCACCTGCAATTCCTCTATACTTTTT
BL03682	*sigA*	BL03906		AAGGTATTATATTGTGCTATATTTTTTATTTTTC
BL03682	*sigA*	BL03905	*eutD*	CGTTTAAAAAGTGTAAACGTTATAATAAAATACAG
BL03682	*sigA*	BL03890	*sacP*	AGTTGACGAAAGCGTTATCACATAATAAAATGAAA
BL03682	*sigA*	BL03887	*nfrA1*	GCTTTAGAGAAAAGCAGATATTGGATATGATAAAG
BL03682	*sigA*	BL03870	*galE*	AGATTCACAGGCGTTTTATAGATTTTAAAATAGAA
BL03682	*sigA*	BL03867	*xylA*	CATTGCTTTGAAGCTGTGAATTTATTATAGTATAA
BL03682	*sigA*	BL03858		CAGTTAATAAGTACATATAACTATAATGCGTTTAG
BL03682	*sigA*	BL03844	*licH*	GCATCACTTTCAGACAGCTTTGATTCAAAATGAAG
BL03682	*sigA*	BL03840	*licR*	TCAGTATTATCTTAAATGAATTGATACTCTGTTTA
BL03682	*sigA*	BL03772	*dppA*	TATTGCATTTCTATTAAAATTCTAATATAATTTGT
BL03682	*sigA*	BL03742	*htrA*	TATATAAGATAGGAAAAAGAAACTGTTAAAGGGTG
BL03682	*sigA*	BL03735	*guaD*	TGTTTATTTTGCCGGGTCCATCAGCTATAATAAGA
BL03682	*sigA*	BL03680	*yqxD*	TTTGTATTATAATAAAAAATTGTGATAAAATAATT
BL03682	*sigA*	BL03673	*cdd*	GCGTGAAAAAGAAGCCATTTTTATGTAAAATAAAA
BL03682	*sigA*	BL03640	*clpE*	ACTTGAAAGTCAAAGAAGGTCAATGTATACTAATA
BL03682	*sigA*	BL03634	*mtnU*	GTAGTAGTATATAACAATTTACATTTTACATTTCT
BL03682	*sigA*	BL03633	*mtnK*	TGAGGACATTCGCAAAGATTTATATTAAAAAAATG
BL03682	*sigA*	BL03605	*clpP*	GTTTGACCTTTATTGACCAAAAATGTATGATTAAA
BL03682	*sigA*	BL04024	*dhbF*	AGAGTAATTTCGTAGTACTTTACCGGTAGTGTATG
BL03682	*sigA*	BL04020	*dhbA*	ATTTGACCGTTAGCAGTTATTGGTATATTATATTT
BL03682	*sigA*	BL04016	*feuA*	ATATTAAAATTAACTATTATCAATAGTTAATTTAG
BL03682	*sigA*	BL03588	*ykuN*	ATTATCATTTAAAGTGTATCTATACAATAATGAAA
BL03682	*sigA*	BL03587	*ccpC*	TAATTTACATACTATTCTTTTTGAATAACCTTACT
BL03682	*sigA*	BL03580	*ykuF*	TGTTAATTTTTTTTAATTTATATGATAAACTATCA
BL03682	*sigA*	BL03565	*ptsG*	CATTGCATTTGTACCGTTTGACTAATAAAATTTCA
BL03682	*sigA*	BL03560	*zosA*	TTTTTAAAATGTAAATGATAATTAATATCATAAAG
BL03682	*sigA*	BL03540	*mtnW*	CGATGGGATAATTCAAAATAATTTATAAAATCAAT
BL03682	*sigA*	BL03539	*mtnV*	TCTTTACATTTTACATTTAACAATATATGATGATG
BL03682	*sigA*	BL03537	*ogt*	ATTTTAATAACTTCTTGTATTTCCTTTTTAGCTTC
BL03682	*sigA*	BL03474	*opuCA*	ACTTTTTATTTTACAAAGTGTATCTTATAATGAAT
BL03682	*sigA*	BL03465	*pgk*	ACTTGGTTCTTGATGAGAGAAGCGCTATAATGAAA
BL03682	*sigA*	BL03464	*gapA*	GGAGTTATATTTAATTTATAGAGAATAAATGAATT
BL03682	*sigA*	BL03463	*cggR*	AGTTGAATAAACATTTTAACCCTGTTACAATAAGG
BL03682	*sigA*	BL03461	*yvbW*	CATATAATATACTTATATAAAGTCCGTCTTCTTTT
BL03682	*sigA*	BL02999	*ctaA*	GAGATAGTATGTTCTAATGGCCTACCTTTTTTTGT
BL03682	*sigA*	BL02973	*ylaM*	TTTTTAATATTCGCGAAAGAAAGCTTAACAGTGAA
BL03682	*sigA*	BL02968	*ylaC*	GGTGTGGTTTTTTTTCTAAATGGCTCACAATAATA
BL03682	*sigA*	BL02959	*cimH*	AAAATCATATGAATTAAATCAATTTATTGAGTCGT
BL03682	*sigA*	BL02940	*citB*	TATTTACTTATGTTTAAAATTGTTTTAAGATTAAA
BL03682	*sigA*	BL02939	*ccdC*	CATGGTTTTTTTCTTCGTTTTAGAGTATAGTAACA
BL03682	*sigA*	BL02937	*ccdA*	TTATGACATAGAAGGAACATGGTTATAAAATAAAA
BL03682	*sigA*	BL02922	*ansB*	AAAATATATTCCTTTAAATGATGTTTATCTATTTA
BL03682	*sigA*	BL02902	*glpD*	GCTTTTAAATAAGCTGCATGTATGTTACAATCTAA
BL03682	*sigA*	BL03381	*secA*	TGTTTGGAAATGACAAAAGGTATGATATGATATTG
BL03682	*sigA*	BL03363	*comFA*	TATGTCATATTTTCGGTTAGAGGCAAAACGTTTTG
BL03682	*sigA*	BL03360	*degS*	CTTAGAATTTTGGCGCGTATTTTGTTATGATTAAA
BL03682	*sigA*	BL03330	*mecA*	TATTTCTTTTCCGCCTTTTTTATCATAAAATAGAA
BL03682	*sigA*	BL03329	*spxA*	ACTGTTCACACTTACTTTTTTATAGTATAATACTC
BL03682	*sigA*	BL03313	*fabHA*	TATTGCCAACTAAAAAAAAATAGGGTAGGATTAGT
BL03682	*sigA*	BL02898	*yhcY*	CGGGCTTTTTTTTATCAACTATCGGTATAATGAAA
BL03682	*sigA*	BL02894	*hmp*	TATTGCTTTAATTCCATTAAAAGTCTAGTATAAAG
BL03682	*sigA*	BL02892	*lytE*	GAAGTAATATTTTTACATTTTATTTCATAAATTAC
BL03682	*sigA*	BL02891	*citA*	TTTATAATATAGTTGATATTATTATAAATGTTTCC
BL03682	*sigA*	BL02869	*hemZ*	GATTGATTTTGCCCATTTATTTTTGTACACTAAAG

BL03682	*sigA*	BL02867	*rapI*	AATTGTAAAAAAATTTAAGTTTCCATAAAATGATT
BL03682	*sigA*	BL02857	*citR*	CCTTTGTAAATATTATTATAGTTGATATAATATTT
BL03682	*sigA*	BL02854	*ydeL*	GGCTTTTTAAAATGACCAGTTTTGATAAAATAAAA
BL03682	*sigA*	BL02851	*sigM*	ATTAGTTTTTTTGAACGTGTTCAGTTATTATATAG
BL03682	*sigA*	BL02822	*lrpC*	TCGTGAAAATATTAAAATAATGAAATAGAATGGAG
BL03682	*sigA*	BL03288	*ytkD*	CTCTACCGAAAGAACCAATATATGATATGATGAGT
BL03682	*sigA*	BL03273	*sigH*	GTTGACGCTTTTTTGTCCATTACTGTATAATATTT
BL03682	*sigA*	BL03267	*gltX*	TTTTGAAGACAGGTTCATTTGATGATAGAATAGGA
BL03682	*sigA*	BL03261	*clpC*	GGATAATAAAAGGTCTTTCCAGCAATATACTTAAT
BL03682	*sigA*	BL03258	*ctsR*	TGAATATCATTTCAATTTCAGTTTATATCAGTTTC
BL03682	*sigA*	BL03251	*med*	TCTTCCTAATTTACACTTGAAAGGATACAATAAAG
BL03682	*sigA*	BL03241	*argC*	CAATTGAATTAATTTTTATTCATGTTATAATGTTA
BL03682	*sigA*	BL03234	*glnR*	AGTTGACACATTATATAACATCACATATAATAAAT
BL03682	*sigA*	BL03206	*ssuB*	ATCATACAATAAATTATAAAGCCTTTTAAGGTAGT
BL03682	*sigA*	BL02798	*rpoB*	TTTGTATAAAATATACATTATTGGGAACACTAATA
BL03682	*sigA*	BL02786	*folE*	ATGTTAAAATCTACGTGAATTTTTGCCACTTTTTT
BL03682	*sigA*	BL02775	*trpE*	ATTGACAAAAAATACTGAATTGTAATACGATTGTA
BL03682	*sigA*	BL02762	*qcrA*	ATTGAGCAAATATTGAACATTTTGTTACAATATTG
BL03682	*sigA*	BL02732	*ponA*	AGTTGTATTTTTCGCCATCATAAATTATGATTTTA
BL03682	*sigA*	BL03177	*yhcL*	ACTTGACCAATCAGGTTTGTTGTACTATAGTTTTT
BL03682	*sigA*	BL03157	*pstS*	CGAATTAAATGTTTTGGAATTATTAAGTAAATTTG
BL03682	*sigA*	BL03156	*purA*	AATTGACTTTACGGGATGACACTGATAAACTGAAT
BL03682	*sigA*	BL03102	*yflG*	GTTCCTTTTTCGCAAACAAATATGGTAAGCTTGAA
BL03682	*sigA*	BL02692		GAAATAGTATTGTCCTTTGCCTTTATTAACGTTAA
BL03682	*sigA*	BL02665		TGTTTCACATAGTTTAGCTGCTAGAACATCTTTTC
BL03682	*sigA*	BL02664	*yngB*	TTTTTCTTTTCCTCAGAAAACATTTCATAGTTTGT
BL03682	*sigA*	BL02660	*yjcI*	CCTTGAAAAATGTCTTGAATTAGGTTAAAGTAAAG
BL03682	*sigA*	BL02653		TAAGTATAATAGTTAAAGTTGGGTGATTTAGTTGA
BL03682	*sigA*	BL02610	*yuxH*	AGACGGGATTTGCCCAATTATTTGATATACTTAAA
BL03682	*sigA*	BL02608	*comQ*	TCTGTCAATCGGGATTTTGATTTAGTAAAATATTA
BL03682	*sigA*	BL02604	*comA*	CTTGGCATCCGCCAAGTTTTTTCTATAAAATGGTG
BL03682	*sigA*	BL03081	*yfkJ*	CCTTTAATTTTGTGTCTAACTAGAATATTCTGAAA
BL03682	*sigA*	BL03068	*treP*	TATTGACTACTTGTATATACAGGAATACAATAAAC
BL03682	*sigA*	BL03039	*fabL*	GATAAAAAATAGTCTAAAAGTTAGTGATATGATCC
BL03682	*sigA*	BL03017	*glvA*	CATTTAACATTTGCGAATATTCCCCCGAAATTTTT
BL03682	*sigA*	BL03016	*acoR*	TTTTTATCGATATTGAAAGCGCTTTTATAATATAA
BL03682	*sigA*	BL03009	*ald*	AAAATAAAAAGCTAAGTTTGTTTCTTTTAAGATTT
BL03682	*sigA*	BL03007	*yvgW*	TCTATTATATACGCATATACGAGTATATATTTTAT
BL03682	*sigA*	BL02564	*gbsA*	GAAATAGAATTGCTTTCTTTAACTTTACAGTTTTA
BL03682	*sigA*	BL02556	*araR*	AAGATGATATGATTTTTTTCTTTGCCTGCTGTTCG
BL03682	*sigA*	BL02543	*maeN*	AAATTAATAGTTAGCTAATAAAATTAAATTTTTTG
BL03682	*sigA*	BL02536	*pbpD*	CCTTCTCGATATCCAAAAAGAATGGTACGATATGG
BL03682	*sigA*	BL02480	*alsS*	AAATTATTATGTTATTAAAATAATTTTTAAGTAAC
BL03682	*sigA*	BL02474	*ywtG*	ATGTTAACAACTTATTCAGATTTAAAATTATTTTT
BL03682	*sigA*	BL02472	*kdgR*	CTTATAACATTTCCTGTTACGACTTTATAATTTTT
BL03682	*sigA*	BL02466	*pmi*	CAAATATAAAGTTGTATGTACAGATAGTTATTTTC
BL03682	*sigA*	BL02464	*tagD*	ACTTAACGTTTGATAGGCGATGTGCTATAATGAAT
BL03682	*sigA*	BL02456	*nrgA*	CATCATGATATCTCCCTTTTGTCACTATAATGAAA
BL03682	*sigA*	BL02438	*rbsR*	TATTGTGATTTTAAATGGATCGTGATAATCTTTAT
BL03682	*sigA*	BL02432	*tagA*	TAAGTAATATCGTGTAGCGGATAGTTTGCAATTCA
BL03682	*sigA*	BL02431	*gtaB*	ACTTGTTTAAGAATTGTTTTTCTTATATGATGAGT
BL03682	*sigA*	BL02430	*lytR*	AATTTTAAGAAACTTATAATCAAAGTAGACTATTA
BL03682	*sigA*	BL02409	*menF*	TGATTCAAATCTTTTTCAATTTTTTCCACAGTTAT
BL03682	*sigA*	BL02406	*menB*	GGTAGAATAGTAAGTGAGTTTATTTTAAAGTAGGA
BL03682	*sigA*	BL02405	*menE*	TCAAGACATTTTTTGGAGATGATGATATGCTGATG
BL03682	*sigA*	BL01987	*yngI*	CATTCAAATACACGGAGGATACGGGTATATGAAAG
BL03682	*sigA*	BL01979	*yitB*	TTTCGGATATTCACAAAATTGCTGAAATAATAACA
BL03682	*sigA*	BL01971	*gltA*	GATTTAATATATAACAAACATATCTAAACTTTTGG

BL03682	*sigA*	BL01954	*cysL*	AGTTTACTGGAATACTCAACATTAGTAAAATATAT
BL03682	*sigA*	BL01953	*yocH*	TATTGTTAAAAATCAGAATTTTATATAAAATCAAA
BL03682	*sigA*	BL01949	*cysI*	TATATAAAATGATTACAACTCATAAGGTCATTTGA
BL03682	*sigA*	BL01947		CGTTGACAATTGGATAAATGTATGCTACTTTTTAC
BL03682	*sigA*	BL01946		TCCTGCCGTTAACAGAAGTATCAGATAAAGTGTTC
BL03682	*sigA*	BL01937		AATATAGTAACTGCCTTTTATGCCCAAACTTCTTG
BL03682	*sigA*	BL01920		TATTTATCTTCCGTTTGTTATTTTATTGAATCGGA
BL03682	*sigA*	BL01919	*fruR*	TCTTGCACATATCGGCCCGATTCTCTAAAATAAGT
BL03682	*sigA*	BL01908	*rapE*	AGTTTTGCAATTATGAATTGTATGTTAATCTTTAA
BL03682	*sigA*	BL02398	*cypE*	ATTTAAAGAAAAGATAGAGGAGGGCTATCCTGAAG
BL03682	*sigA*	BL02371	*yycF*	TTTTTAACTTTTGCTTCATAAACGTTATAATGTAT
BL03682	*sigA*	BL02356	*yaaJ*	ATTGAAAATCATTAATGATGTCGACTATAATGAGA
BL03682	*sigA*	BL02349	*yciA*	GAACTAAAATAGTGTCGTAACGGAGTTTTATTTAT
BL03682	*sigA*	BL02348	*yciC*	CTTTGCATTTTTAAATAAAGTGTGGTGAAATAAAA
BL03682	*sigA*	BL02341	*yfmT*	AAAAAAATATAGGTTTTCTGGGTATTCTAACTTAA
BL03682	*sigA*	BL02337	*rapK*	CGTTAAATTTTTCGTTGAAAGATTTTACGCTGATA
BL03682	*sigA*	BL02335		TTATTACGAATACGGATCAATATGATATAAAAGTC
BL03682	*sigA*	BL02334		TGTTGACGCTTTCATAAAACCGATATATACTTTAA
BL03682	*sigA*	BL02312	*fapR*	GATTGCATTCCAATCTATGAAATTATATACTACTA
BL03682	*sigA*	BL01899	*copA*	AAAGTTATACCTTGTTTTTTGTGACTTTCAGCTTC
BL03682	*sigA*	BL01857	*gabR*	TCTTATAATTCTGAATATTTTTTGGTATGATGAAA
BL03682	*sigA*	BL01830	*rocR*	TTTTGCGGTTTTTATGCAAAGGTGCAAAAATAAAT
BL03682	*sigA*	BL01812	*nasA*	AGTATAACACGTTATGAAGTGTATGAAAAAGTTAA
BL03682	*sigA*	BL01803		TATGGACAAATTTGTTATAACATGTTAAGTTTCAA
BL03682	*sigA*	BL02282	*cysH*	GATTATTTTTATTTTTTATTAATCCTATAAAAAAA
BL03682	*sigA*	BL02271	*pyrR*	CGTTGACAGACGTTTTCTTTTTTGAAATAATAAAT
BL03682	*sigA*	BL02267	*ileS*	AAGCGTATATTATGCAAGTATTTAACTCACTTTTT
BL03682	*sigA*	BL02254	*spoIIGA*	ACTTCAAAAAATGTACGATTTACGCAACATTAATT
BL03682	*sigA*	BL02250	*ftsA*	CATTGTATTGTTGTTCGGCAAATAATAGAATAGAA
BL03682	*sigA*	BL02245	*divIB*	GGAATACTATATTACATACTTTGTTTGTTCGTTTG
BL03682	*sigA*	BL02226	*gudB*	ACTGTAATATATACTTAAAGTTTGCAAACTCGTAG
BL03682	*sigA*	BL02216	*sacB*	AGTTGAGTCCCAGAGATGAGTATATTAGAATGATG
BL03682	*sigA*	BL02202	*rsbR*	TTTTGTGTTCAAAAATAACAAACGATATAATGGAA
BL03682	*sigA*	BL01786	*glsA*	TTTTTAAATCAATCGGCTTATTTTTTAAATTTTTT
BL03682	*sigA*	BL01769	*nasD*	TTTTTCAAATGTTTTAATTGATATTAAACACTATC
BL03682	*sigA*	BL01767	*nasB*	ATATTCGGATCATCTAACTTCCTTTTCTTTGTTCG
BL03682	*sigA*	BL01765	*gabD*	AGTGGAAAACAACGGGTATATCTGCTATCCTCTTA
BL03682	*sigA*	BL01762	*gabT*	AAAGTAGTATGGTTTTTTATAAGTCTTAATATTCT
BL03682	*sigA*	BL01738	*rocE*	AATTGAAATATCAGGAAAATGTTCAGAAGATTTAA
BL03682	*sigA*	BL01737	*rocD*	TAAATAAAAACGTGGAAACGTATTTTTGGCGTTTT
BL03682	*sigA*	BL01728	*lchAC*	TAAGTAGAACAGTACAAGTAAGTTATGTCTCCTTA
BL03682	*sigA*	BL01727	*lchAB*	ATTAGATAAACTGATGGAAATTTTCTAACAAAAAA
BL03682	*sigA*	BL01726	*lchAA*	CCATGCTGTTTTTTTAATATTTTAATATACTCTTT
BL03682	*sigA*	BL01718	*yckE*	TAGAAAAAATAGCGAAAAGTTCGTAGCAAAATTTC
BL03682	*sigA*	BL01711	*ycgO*	CCTTCAAAAAAAGAGAAATTAGTCTAAAAATTTAG
BL03682	*sigA*	BL01710	*ycgN*	GTTTAAAGGCATGACGAAGAAATAAAACGATAAAA
BL03682	*sigA*	BL01706	*nadE*	ATTCACTTTGAAATGAGACATATGCTATAGTTAAA
BL03682	*sigA*	BL01703	*ldh*	AATTGAAAAAGTATGTGAAGTATTGCACAATATGA
BL03682	*sigA*	BL02188		GGTTTACATATTTCGAATTTTCAGGTATAACTGTT
BL03682	*sigA*	BL02180	*yusL*	TATTGAACTCGCTTCCAAAATCTTTTAAGATGAAA
BL03682	*sigA*	BL02161		GTTGTGAAAAATTATTACTTTTATGTAAAATGATG
BL03682	*sigA*	BL02154	*blt*	ACTAGACTTTAAACCTATCTGTGGATATACTAATT
BL03682	*sigA*	BL02153	*pucF*	AAAGTACAATGGTCATTTTAACATAAATATATCTC
BL03703	*zur*	BL05309	*yciC*	GTTTAGCTTTGTTAATGC
BL03703	*zur*	BL02349	*yciA*	CGAAATCATTTCGATTTA
BL03703	*zur*	BL02348	*yciC*	ATTTAGCATTATTAATGC
BL03742	*htrA*	BL02376	*yyxA*	TCGTTCGGGGTATCCCATGATGCCGCAGAGCCGATGTTTTACGTTTTTCATTACAATGGCCGAAAGCTCGCTTTGATTACGGATACGGGCTATGTCAGCGA

				CAGAATGAAAGG
BL03742	*htrA*	BL00734	*htrB*	ACATCCTTTCTACGTCATTATACTAGTTTTAACATAAGTCTGGTCAATTTT CATAATTTCACATATTCTTTTCATTTTTATCCCACAATGTTTTCGTAACATA TTCATCAGGG
BL03742	*htrA*	BL03742	*htrA*	CTGACATGTCATATTATAAATAGCACCCTATATATTCTATCCTTTTTCTTT GACAATTTCCCACAATGTTTTCATGTTTATCCCACAGTCTTTGTTTATGAT TAAGGCATCAA
BL03937	*ydfI*	BL03936	*ydfJ*	AAAATTTGCCCAAACGTACATGCCCGAATGTACGTTTTT
BL03942	*xylR*	BL03867	*xylA*	TTCCTTCAATCAAATTACCAATTTGTTTGTAACAA
BL05088	*glpP*	BL05089	*glpF*	GACGGAGACCCGGAGACCAC
BL05088	*glpP*	BL00185	*glpT*	AATAAAGAAGGGGAATACCA
BL05088	*glpP*	BL02902	*glpD*	GTTGGAGAAGAGGAGAGACC
BL05249	*fur*	BL01757	*yclN*	GGGAATAACTATTACTATTAGTAATTGCAGCCGCCTC
BL05249	*fur*	BL02185	*yfiZ*	ACAATTAACTTTTACTAAAAGTTCTAGTTACAATAGA
BL05249	*fur*	BL02183	*yusV*	AACAAATACTATTTTCCTGGCTATATGTCTTACAGGG
BL05249	*fur*	BL02042		AGGATTTACTTCAACTATTAGTAAAAGTAAAATACGT
BL05249	*fur*	BL02001	*yfiB*	TTAATTAATTAAAATGGAGAAGCAGGTGCTTTTATGA
BL05249	*fur*	BL02000	*yfiC*	AAAAATAGCCGGCATCGGCACTCATCAACAATTGCTT
BL05249	*fur*	BL01599	*yknV*	TTTTATCATGGAGCTTCCGAATCAATATGATACGATG
BL05249	*fur*	BL01598	*yknU*	AATTGGTTCTTTCCTTCAGTTTCATTTTCAGTTCTTT
BL05249	*fur*	BL01579	*fhuD*	TATATTAACTATTACTAAGAGAAATAGCCCACTAGTT
BL05249	*fur*	BL01463	*yoaJ*	GATTTGTCCTTATAATGATAATGGTTCTCAATTTATA
BL05249	*fur*	BL00705	*fhuC*	CGTATTTTATCTATTTGCTTATGAAAACAAATTAAAG
BL05249	*fur*	BL01086	*yhfQ*	TTTTTAAAATGATAATGATTATCATTGTCATTGATGG
BL05249	*fur*	BL00566	*ywjA*	ATAAAACAGCCTACTTGAGAATAATTTTCAATTATTT
BL05249	*fur*	BL00187	*yxeB*	TATTTACATTGATAATGCGAATCATTATCACTTAGGA
BL05249	*fur*	BL04020	*dhbA*	AAAATTTGCTATTACTAATAGTAATAGTTAACCTTAA
BL05249	*fur*	BL04019	*yuiI*	GACAGTAACTTCTACTTAGGCTTACTGGTGCTTTACC
BL05249	*fur*	BL04016	*feuA*	ACCTATAATTTTAATTGATAATAGTTATCAATTAAAT
BL05249	*fur*	BL04015	*ybbB*	TAACTTTACTATTCATAACTAAAAAATAATATAAAAT
BL05249	*fur*	BL03588	*ykuN*	ACGATTTATTGACATTGATAATCATTATCATTTAAAG
BL05249	*fur*	BL02873	*yheI*	TTGTTTAACTAGTAAGGAAAGTCTTACCTAAATGACA
BL05249	*fur*	BL02872	*yheH*	AAAATTAAAACCTACTAAGCGACAGGCGACAGCTGCG
BL05249	*fur*	BL02726	*bmrA*	AATCGGAATATACTTTAATATTTATTGTCATGTTTTT
BL05249	*fur*	BL03143	*yumC*	GTGATATGATATAATTAAGAACAATTTTCATTTAGGG
BL05249	*fur*	BL03080	*yfkM*	TCCACCTTTCAGATTAGAGTAACAATTTCTATTGTAA
BL05249	*fur*	BL03043	*ygaD*	TGCATCTGAATTTATTGATATTTGGTATGAACGGTAT
BL05249	*fur*	BL03031	*yfhC*	CATTATAATCAACAATGATAATCATTTTCATTAAGGG

Table 1: Binding sequence predictions for *Bacillus megaterium* DSM 319. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BMD_0001	*dnaA*	BMD_0001	*dnaA*	CAAACTGTGGACAACTT
BMD_0064	*purR*	BMD_5250	*purA*	CAAAATACGATATATTTGTAATAACTTTTGTTAAATCGTTCGGATTTAATT
BMD_0064	*purR*	BMD_5146	*glyA*	TCGCAAAATGCTTGAAAGCATTAATTTATATAATTTTATTATGTGTAAAAA
BMD_0064	*purR*	BMD_0271	*purE*	TCTAAATACGAACATTTTAATGATAAAAATAGTAAATATTCGATTTTTAGG
BMD_0064	*purR*	BMD_0266	*pbuG*	ATTCTTTTGGCTTGTTTGCTTGTAAGAAAATATTCTTACAAGCAAAAAACA
BMD_0064	*purR*	BMD_4245	*pyrR*	GCAGAAAAGGCATATTTATAGCCTTTTTGAAGAAATATGTTTAAAACCAAA
BMD_0064	*purR*	BMD_1870		TCAAAAAATAGATATTATTGATAATAAAATTCTTATTCTAGCTCAATCAAA
BMD_0064	*purR*	BMD_1432	*xpt*	GCGATATTGGCTTATAATACAAAACATAGTGTTCATTTACAAGCTTTTATA
BMD_0064	*purR*	BMD_0949		ACTCTTGCAACAGACTAAAAAGATAAGAGAAATAAATTTTCAAAATTCCAA
BMD_0064	*purR*	BMD_1268		GCAAGACGTTACGTTTAGAAAATGATAACAGTTTTAAATAAGGAATGAACG
BMD_0064	*purR*	BMD_1193	*pbuO*	TCATTTTTTGCTTATAATATATAGTACAACGAAAATTACAAGCAATAAAAA
BMD_0117	*sigH*	BMD_5086		GAAGGGATTGTGTCGGGTATTGTAGAATTGTAGAGGT
BMD_0117	*sigH*	BMD_4529	*dnaG*	TGCAGGAATTTAGCACTTAGTGAAGAATATTATATGA
BMD_0117	*sigH*	BMD_4436	*spo0A*	TAAAGGAAATAATGTTGCATTGTCGAATTTGTCCTTT
BMD_0117	*sigH*	BMD_4261	*ftsA*	AAAAGGGAAACAGATGTATTTGTTGAATATTTTCATA
BMD_0117	*sigH*	BMD_4033	*kin*	AATATATATTTAACAAATTTATTATTTTTTTAAAACG
BMD_0117	*sigH*	BMD_2279	*fumC*	TGTCATTTATTAACAATAATACAGTACTTTAAGTAGG
BMD_0117	*sigH*	BMD_5162	*spo0F*	CGTAAAGAAGATGTGGTATTTATATCTATTTAGGAAG
BMD_0117	*sigH*	BMD_5120	*lytF*	TGAAGAAAAAATTGATGAAGTGATCAATCAAAAACAA
BMD_0117	*sigH*	BMD_4652	*minC*	AAAGGAATTGAGTAAGTCTCTGTCGAAATTATATATT
BMD_0187	*sigW*	BMD_2149		TAAAAGGAAAGTTATGTAATAGAAAGAAGTACTTT
BMD_0187	*sigW*	BMD_0711		TAATAAAAACCATTTTTTTACATTATACGTTCTTA
BMD_0187	*sigW*	BMD_4785	*sppA*	ATAAAGTAAACTTTTTAAAAAAAGTTTGTTACAAT
BMD_0187	*sigW*	BMD_4544		AAATTTGAAACTATTGTTAAAGTTATTCGTATATA
BMD_0187	*sigW*	BMD_0187	*sigW*	ATAAGTGAAACTTTATGATAAAGTACACGTATATA
BMD_0187	*sigW*	BMD_3373		AATATTGAAACTTTTATTGAAGTGTTACGTATATA
BMD_0187	*sigW*	BMD_2687		GCATTTGCAATTTTCACCATATTTTAAAGTGATAA
BMD_0229	*sigB*	BMD_3075		TTTTTGAAGCCAATAATAGATAGTAATACAATCAAATA
BMD_0229	*sigB*	BMD_3040		TTTAAATATAATAAAGGTATAATTACTTATTTTTTCGC
BMD_0229	*sigB*	BMD_1994		TGTTGATTTTACTAGATAACATGGTAAAATATTCATTG
BMD_0229	*sigB*	BMD_2336		AGCTTTTTAAATGAAAAGGAGGGTAACAATATGAGTCA
BMD_0229	*sigB*	BMD_1898		TAAAAATATAAAAAAGGAACACCATTCACTAAGTATGC
BMD_0229	*sigB*	BMD_1866		TGTTTTTAGCTGTTGATTGAAGGGAAAAGGTAATGAAA
BMD_0229	*sigB*	BMD_1860	*xylT*	CGAATTTAAAAGGAAATAATAGGTTGCAAAATATGCAA
BMD_0229	*sigB*	BMD_1844	*gapN*	TATAAATAATATATGGGAATAAAATAAAAGAGTTGTGT
BMD_0229	*sigB*	BMD_2252		ACTGTACATAAATGCACGAAGGGGGATTAAAAAAGATG
BMD_0229	*sigB*	BMD_1683		AACTAAACTATTATGGGTTACAAATTTAAATGATTTGG
BMD_0229	*sigB*	BMD_1557		GAAAATTGAGAAAAAGGTAAGGAACTTAGAGAACTTGT
BMD_0229	*sigB*	BMD_1546		GTGTTTGAGAAGCAGACGGGCGGGTAGACAAAAACTAG
BMD_0229	*sigB*	BMD_1482	*galU*	GGGTAAAGTTATTTTATCTATTGGTAAAAATACAAAAC
BMD_0229	*sigB*	BMD_1435		ACAAAAGTATAAATAGTGGATATTGGAAAACAATATGG
BMD_0229	*sigB*	BMD_0979		TGTTTAATAAAAATGAAAAGTTGGTGCTAAACATGATA
BMD_0229	*sigB*	BMD_0954		AATTTAAAGGGGGATATACAATGGAAAAATTAATATCA
BMD_0229	*sigB*	BMD_0953		AATCTTAATATTATGTTTAAAATGACAAACATTTTTGT
BMD_0229	*sigB*	BMD_0863		ATATGTTAATTAATAGTCTTAATATTCAAAAATTTTCA

BMD_0229	*sigB*	BMD_0790	*alsS*	AGGTTAGTTGTCATACAAATTGGGTATTCAATCAGTAG
BMD_0229	*sigB*	BMD_0711		ATGCTTACTTTTCCCTAAAAAGGGTAAAGATAAGATAA
BMD_0229	*sigB*	BMD_1131		TGTTTTGTCTTTCTTTCATTGTGGTATGGAATAGAAAC
BMD_0229	*sigB*	BMD_1126	*galU*	AATTAAAAAGAAAAAGTTTTTGGGGATTTTTTAAGAAG
BMD_0229	*sigB*	BMD_1114	*galU*	TTTATAAAAAATAAGAAAGAAGGATGTATTATGAAAAA
BMD_0229	*sigB*	BMD_0667		GTTTATTATCATATGCGAATGTATGAGTATAATTAATT
BMD_0229	*sigB*	BMD_0662	*rocA*	TTATTTTTTTCTTTATCAAAAAGAAAGAATGATATAAG
BMD_0229	*sigB*	BMD_1041		TTTTTGATTCATGGTATATTTTGCTATAGTAAAAAAGA
BMD_0229	*sigB*	BMD_4857	*dps*	TGTTTTAATCAACTAAAAAAGGGTACAACTATATTATA
BMD_0229	*sigB*	BMD_0493	*mmsA*	GTGATTAAGCACGGTGGAGAAGGGTCTATTGCTTATAC
BMD_0229	*sigB*	BMD_0490		GGGGTAGAATAAAAGGGGAAATTAATTTTCAATTCTTT
BMD_0229	*sigB*	BMD_5245	*walR*	GGTTGAAAATCATATATCTTCATGTATAGTAGAGGATG
BMD_0229	*sigB*	BMD_5226	*katA*	AGTGTACAATATATGGGGTAACTGATGTAGAGTTTTAA
BMD_0229	*sigB*	BMD_4772		TGTTTTGTAACCAAAACAGACGGGTAGGATATAGAAAA
BMD_0229	*sigB*	BMD_4752	*phoP*	GTAGTCATCTCTGTGTGCTGAAACTTTTAAAAATTTGA
BMD_0229	*sigB*	BMD_0372		GGTTCTTATCCACAGAAGAGTGGAAACGATATAAATAA
BMD_0229	*sigB*	BMD_0335		TGTTTTTCACACCTTCTAATAGAAAATAACTCCTATAG
BMD_0229	*sigB*	BMD_5117	*galU*	ATTAACTAATTAAAAGAAAAGTAATATTTACTATTTTT
BMD_0229	*sigB*	BMD_4686	*ilvB*	TTTTCTAAAAATTTTAATTAATTGTAATTAATTTAAAA
BMD_0229	*sigB*	BMD_4617	*relA*	AGTTACAATATGAATTAAAAATGCTAGTTGATAAAAGG
BMD_0229	*sigB*	BMD_5093	*gbsA*	ATTATAAATACCAATGCGAAGGGGAGAAATTACGTTTA
BMD_0229	*sigB*	BMD_5086		AGTTTAGGAAGGGATTGTGTCGGGTATTGTAGAATTGT
BMD_0229	*sigB*	BMD_5044	*clpP*	CGTTTATTTTTTTTAGAAGGGGGAAAAATTTTAATAAT
BMD_0229	*sigB*	BMD_0108		TGTTTGATTCATTAAGATATTGTGAAAATACAAAAAAA
BMD_0229	*sigB*	BMD_4462		GGTTTATTTGATTTTGCTGTAGAATATAAAATACTAAG
BMD_0229	*sigB*	BMD_4404	*mmsA*	CGTTATTACATATGATGAAACGTATAATGCAAAACTTA
BMD_0229	*sigB*	BMD_3967		TAACGGAAAAGTATGGGCGAAGTGAACAGTATAGTGGG
BMD_0229	*sigB*	BMD_3813	*putC*	GTTGCAAATTTTCTAAAAAAGGGGTAATGAAAATGAAC
BMD_0229	*sigB*	BMD_3653		ATTTTCTAAAAGAGAAAATGAGTTGAAAAAGAAAGAAG
BMD_0229	*sigB*	BMD_3619		ACATTGTCATATAAAGGATATATCAATTGCTATTTATG
BMD_0229	*sigB*	BMD_4061		AGTTTCTTCATCATTTTAATACGGTATATGAACGTAAT
BMD_0229	*sigB*	BMD_3497		GTTTTTTATTAGCTTATATAAGGGTATTTAATAAAATA
BMD_0229	*sigB*	BMD_3493		GATGAGTAAGAAATCGGAAACAAAGGTTGAAATTTGAA
BMD_0229	*sigB*	BMD_3490		GGTTTTAAGTGATTGTATATAGGGAAAAATTCTAATAA
BMD_0229	*sigB*	BMD_3463	*mmsA*	CGTTTACTATAATTTAGCGGCAGAAAAATCAGACGTGA
BMD_0229	*sigB*	BMD_3215		TGATTAAAGAAATCAAAGCAGGGATAGAGAAGTAATAG
BMD_0229	*sigB*	BMD_2697		TGGTTCATTCATTTAATGAATGAAAACGAAAATTATAC
BMD_0255		BMD_0790	*alsS*	TTTTAATTCAGTTTTCGATCTAGTTTTTTAACA
BMD_0255		BMD_1131		ATCAAATAAATTTAATAAATCGACTTTTGTTCC
BMD_0255		BMD_0524	*ldh*	ACACTAACATTGTGAAGAACATCACAAAATAAA
BMD_0255		BMD_5217	*cydA*	TATACATACACTTTGTAACTCGTATAGAGAGCT
BMD_0255		BMD_4686	*ilvB*	AATATAAGTTTTCTAAAAATTTTAATTAATTGT
BMD_0255		BMD_2697		AAAGGTTTTATGCGTGGTTCTACATAAAAAAAA
BMD_0255		BMD_1683		TTTTTAAAAACTTATCACCACATTTTTATGTAA
BMD_0255		BMD_0954		ATCGTAGCTGAGTTTATTACCTAAAAAAATATA
BMD_0341	*malR*	BMD_2015		GAAGCTGTGCTTTTTTTAGTTACAAAAAGAATTCTTAAGTTACATAACGTA
BMD_0341	*malR*	BMD_2014		TTTGATTTTAATAAAGTTTTATTTAAAAAATATTAATTTGTAAAAAAGGTA
BMD_0341	*malR*	BMD_5230		TGCAGTTTAGTTATTTTTAATAAAATAACGCTTACAAGCTATTTTGAAGGG
BMD_0341	*malR*	BMD_3745		GTGTATTTCTTTCTTCATTTTCTAAAAACCATTCTACTTCACGGTTAAGCG
BMD_0341	*malR*	BMD_2620		GATAGATTAACAAATCCTTTTCTCCATTTTTATACTCTCGTATATTTCATC
BMD_0417	*perR*	BMD_0417	*perR*	AATATTCTTAATATTTCATTA
BMD_0417	*perR*	BMD_5226	*katA*	AATATTATTAATATTAAATCA
BMD_0417	*perR*	BMD_0375	*cadA*	AATATTTATAATACTTCTTGT
BMD_0417	*perR*	BMD_4672	*hemA*	AATATTATCAATCTCTATTCC
BMD_0417	*perR*	BMD_4385	*fur*	TCTTATTAATAATAACTTTAT
BMD_0417	*perR*	BMD_3164	*cadA*	ACTGTATATAATATTTGTTTG

BMD_0417	*perR*	BMD_3040		AATATAATAAAGGTATAATTA
BMD_0417	*perR*	BMD_1817		GATAAAAATAATCCTTTTCCT
BMD_0417	*perR*	BMD_0714	*spxA*	ATTTTTTAAAAGTAATTATAA
BMD_0532	*glpP*	BMD_0533	*glpF*	GAGGGAGAATAGGAGATCCA
BMD_0585	*hpr*	BMD_4434		AAAAAGGTATATTAAAGTAA
BMD_0585	*hpr*	BMD_2012	*dtpT*	ACAAAGATAAAAAAATATAG
BMD_0585	*hpr*	BMD_2285	*nprM*	TAAAATAATATATAGACGTA
BMD_0854	*citT*	BMD_0514		TTTAGAAGGCAAAAACAAAATGCAAGTGATTGATTATTTTAATAAAG CAAC
BMD_0854	*citT*	BMD_3792		TTCTAAAAAAACCTATATTTCCTTTATTTTTAAAAATGTACTAACATG AAA
BMD_0854	*citT*	BMD_3683		TTAGTAAAGGAGTGATTAGTTAAAGAAACACGAGAATAGAAAATAA TCTAT
BMD_0854	*citT*	BMD_2294	*citM*	TTAACCGAAAATAAAATGATTAACCAAGATAAAATAGACCATTTATT TAGA
BMD_0854	*citT*	BMD_0852	*citM*	ATATTCAAAAATAGAAAAGTTCCAAACAAAGAAGTTGAAAAACATTC CATT
BMD_0901	*rpoN*	BMD_4429	*ptb*	CAGTGGATGGCACAAATCTTGCATGTATTTTAGGTG
BMD_0901	*rpoN*	BMD_4404	*mmsA*	GTGGTAAATAAATAAGTTATTATTTCCATCTAAACA
BMD_0901	*rpoN*	BMD_4340	*gudB*	AAAACATTGTAGCATACCTTGTAAAGTTATTGGAAG
BMD_0901	*rpoN*	BMD_3813	*putC*	TACTATTAAATATAACTTGTTGAAACAGATAACAAA
BMD_0901	*rpoN*	BMD_4063	*gabT*	TAAAAGTTGGCACGTTTTTTGCATATAAACTAATTG
BMD_0901	*rpoN*	BMD_4061		CAAAATCTGGAGCAAATGGTCGAAGAAGGAAAGTTT
BMD_0901	*rpoN*	BMD_3463	*mmsA*	GATATGATGGAAAAGCTTTTAAATTATCAAGAATCA
BMD_0901	*rpoN*	BMD_2413	*rocG*	CAGAAATTGGCACAGGAATTGCATGTTAAATCGTTG
BMD_0901	*rpoN*	BMD_1994		AAAAAGCTCAAACGGTTCATTTACCTTCTAACAGAA
BMD_0901	*rpoN*	BMD_2336		AACGGCGTCTTGTACGACCTCAGTGTGGTCTATAAA
BMD_0901	*rpoN*	BMD_1872	*acoA*	TTCATGTTGGCACGGTACTTGCAAGAGTAATTAAGT
BMD_0901	*rpoN*	BMD_1844	*gapN*	TGAAAATTTTGATAATAATTTCACTTTTATTAAGGG
BMD_0901	*rpoN*	BMD_1546		AACTAGATGACATAGTTCAATAATTAACAAACTAAA
BMD_0901	*rpoN*	BMD_1435		AACCGACAAAGTTATTTTCTGAGTTCGGTAAGACAA
BMD_0901	*rpoN*	BMD_0979		ACAAATTATTTTTACTTTTCAACCACGATTTGTACT
BMD_0901	*rpoN*	BMD_0953		AATATTTTTAGAAAATTCTTTAATTTTCTATTGCTA
BMD_0901	*rpoN*	BMD_0945		TCAATGATGCCAAACATCTTGAATAAATATACTTAA
BMD_0901	*rpoN*	BMD_0863		CCTGCTATTGCAGAAAACTTACTTTCTTTGTCGCTT
BMD_0901	*rpoN*	BMD_0663	*rocD*	AAATTTTTGGTCCGGTAGTAGCATTCTCTAAAGCGC
BMD_0901	*rpoN*	BMD_0662	*rocA*	TAAAAGTTGGCACGCTATTTGCATAACTATAAAGTA
BMD_0901	*rpoN*	BMD_1076		ACGTCTTTGGCACGCTCTTTGCATGTAAAACAAGTA
BMD_0901	*rpoN*	BMD_0493	*mmsA*	AAAATCTTGAGAGATTGCTTGAGAGATTAAAGAAAT
BMD_0901	*rpoN*	BMD_5093	*gbsA*	TGAGTATTAATAAAAGTAGAAGGTAGATTTGAGGAA
BMD_0959	*treR*	BMD_3721		AAATTTCTAGACCTTAAATCTTAGGGACTAAT
BMD_0959	*treR*	BMD_4002		CATAATTTTCAGTTTATATTTTTTAGCATAGT
BMD_0959	*treR*	BMD_0957	*scrA*	CAACTGCTTGAACATATATGTCCAATCTTATT
BMD_1060		BMD_0804	*glsA*	ACAAAAAGTTTTGTAGGTTTTTGTATGCTCTCGTAGGTTGCATT
BMD_1209	*sigI*	BMD_1209	*sigI*	GAAAAAACGCGCCTTCAATGCTAATTGAAGGCGCGTTTTCTATATTA AGACAAA
BMD_1309		BMD_3077	*citA*	CTATTTTACTTATGA
BMD_1309		BMD_2546	*acnA*	TTTATTTACTTATGT
BMD_1309		BMD_4756	*citZ*	ATTATTATTAAAAAA
BMD_1419	*liaR*	BMD_2490	*yhcY*	CGCTTTTCGAATAGGAGTTGATAAAGT
BMD_1472	*desR*	BMD_1474	*des*	AGTATCAAAGATATACT
BMD_1857	*xylR*	BMD_1858	*xylA*	AAGTTAGTTTATTGGATAAACAAACTAACTCAATT
BMD_1857	*xylR*	BMD_3587	*gph*	TTGGCGTAGACAAAAAAGAAATTCCATTTACAAAT
BMD_2054	*gltC*	BMD_2055	*gltA*	ATATCAAAATTAGA
BMD_2054	*gltC*	BMD_2054	*gltC*	AGATTAAAACTATA
BMD_2224		BMD_3248		GTAAATGATTTTTGGTAGGAGCACGA
BMD_2224		BMD_3122		AATAATTACTAGAATTATGATCAATA
BMD_2224		BMD_2224		ATAGTAATGTTTTATAATATTTATAA
BMD_3037	*pucR*	BMD_0755	*ncs1*	TCAACTGTGATGTGTTATATATTTTCA
BMD_3037	*pucR*	BMD_3036		TTCTATTTTATTTATTGTATATTACAA
BMD_3037	*pucR*	BMD_2505	*pucK*	ATCTTCTGTTTTTGGTCAACTTGACTA

BMD_3037	*pucR*	BMD_2011	*allB*	ATCACTTTCACTGGTTTACTTTTTCTT
BMD_3127		BMD_3718		GACTATAGACTAACTCCATA
BMD_3127		BMD_3127		GACCCTTACGCTGCGTTATA
BMD_3309		BMD_2055	*gltA*	ATATCAAAATTAGAT
BMD_3309		BMD_3309		ATCAATAAAAATGAT
BMD_3309		BMD_1680		TGGAGTAAAATTCTT
BMD_3533	*araR*	BMD_3533	*araR*	TATTTGTACGTACATTTG
BMD_3533	*araR*	BMD_1860	*xylT*	AATAAGTGCGAACTCAGT
BMD_3589		BMD_3588	*gutB*	AGAAAAGTACAGTGTTACTGTACTTTTAG
BMD_3899		BMD_5240	*htrA*	GTGAAACTTTAACCGGTTTGCTTTAGCAAACCAGTTAACCTTCACCTC AGAAAACCGCAAAGCGTACTACGGCGTCTCGGATACAAAATACAGA AAGTAGTTCCCAGATTTTT
BMD_3899		BMD_3899		GTCTATGTACCCGCTATTATACTACGTGTTTGCTGTTATAAAGAAATT TTTCACAATTTCACACATTGTTTTGATACTTATCCCAAACTTTTATTTT ATAGTATAGATAACAT
BMD_3938		BMD_3938		ATTTTTTTCTTTTTTATTTACCATTTTAAAAGTGAAGT
BMD_4003	*sacT*	BMD_4002		TCGGGATTGTGACTGGTAAAGCAGGCAAGACCTAGAATG
BMD_4003	*sacT*	BMD_1568	*sacB*	AGACGAGGCAGATTAGTCGGAATAGTTTGCATGACGCCT
BMD_4003	*sacT*	BMD_1536	*sacB*	TGGAAATCGTGGTTTGAAATGTAAAAAAGCCGTACATTT
BMD_4003	*sacT*	BMD_0957	*scrA*	GATGAATTACGAAAAATAATTCAGTTAACAAAAAAAGTT
BMD_4003	*sacT*	BMD_3721		TCTTAACGCTGAATCAATGGATTAGTCATCGATTAAGTC
BMD_4077	*lexA*	BMD_0743		AAAACTTGTATACACTACAGAA
BMD_4077	*lexA*	BMD_1170		TATGGTTGTAACCAAACAATTT
BMD_4077	*lexA*	BMD_4765	*dnaE*	AAAACGTATCAAATGTTCCTCC
BMD_4077	*lexA*	BMD_4714	*uvrC*	TTTAGCGAATATGCGTTCGTTT
BMD_4077	*lexA*	BMD_5143	*vpr*	AAAACAGACCGGAAGCTTTTAA
BMD_4077	*lexA*	BMD_4630	*ruvA*	GAAAAGAGGCAAACGTTCTCTT
BMD_4077	*lexA*	BMD_5069	*uvrB*	AAATACGAATATCAGTTCGTGT
BMD_4077	*lexA*	BMD_3972		TATTTTTACAAATATGTAAATA
BMD_4077	*lexA*	BMD_4106	*recA2*	TTTGCTTGTAAATAAGCGAAAA
BMD_4077	*lexA*	BMD_4077	*lexA*	TAGTCTTATATACAAGAATAAA
BMD_4077	*lexA*	BMD_3500		AATGCTTGTAAACACGATACTA
BMD_4077	*lexA*	BMD_3490		ATACTTTACATACAAATAAAAA
BMD_4077	*lexA*	BMD_2859		AAAAAATAACTCAGGCTCTAAA
BMD_4077	*lexA*	BMD_3020		TGATCTAGTAAATGAGTCAAAT
BMD_4077	*lexA*	BMD_2534	*parE*	AAATAGAAACATACGTTCTGTT
BMD_4077	*lexA*	BMD_1726	*recA1*	TGTCCTTGTAAACAAGCTGTGA
BMD_4077	*lexA*	BMD_2165		CTAAATTAACAGAAATTCGTCA
BMD_4154	*sigD*	BMD_4061		ACTAAACATGATAGCTGAATTGCACCACGGTTAATGCTAA
BMD_4154	*sigD*	BMD_3463	*mmsA*	CAGCTGAACGATGGTTTTCTCATCACGCTAAGATTGTTGT
BMD_4154	*sigD*	BMD_2995		AACATTGGTTATTTCTTTTTACATTCGATAAACTAACTGT
BMD_4154	*sigD*	BMD_2455		TGACTAAAGGAATATGAAGTCTTGCCGATAGAAAGAAAGT
BMD_4154	*sigD*	BMD_1998	*motA*	AGGCTTAAAAATTCTATGAAATAGTCGAAATAAAAGATGA
BMD_4154	*sigD*	BMD_1994		AATTTTAGATAACTAGAATATGTGTAGAAAAATTCAGAAT
BMD_4154	*sigD*	BMD_2336		GGGCAAACCATTCGCTTAATGCTGCCCCTAAATATTGATA
BMD_4154	*sigD*	BMD_1844	*gapN*	ATATTCTACTTTTATCTTCATTTTCCTATATTTATTATAT
BMD_4154	*sigD*	BMD_1546		GAAAATAATTTCAGCTGTGAAAAGCGGATCGGAAAAGCGA
BMD_4154	*sigD*	BMD_1435		TATCACCTATAACCTTTTGTTATACCCATAGTATAGAAGG
BMD_4154	*sigD*	BMD_0979		TTGTTTAATAAAAATGAAAAGTTGGTGCTAAACATGATAG
BMD_4154	*sigD*	BMD_0953		TTTGAATAATAAAACATAAAATCAATAATCTTAATATTAT
BMD_4154	*sigD*	BMD_0863		AAGTTTAAAATGACTGACAATCAGTCGTTATTTAACAAGG
BMD_4154	*sigD*	BMD_1296	*cheV*	ACAAAATAAGAAAATTTTTCTTTACGGATATAAGAGGCGG
BMD_4154	*sigD*	BMD_0662	*rocA*	TACGTTTTATATAACCGCAAACTTCACGTTTTTTTTATTT
BMD_4154	*sigD*	BMD_1098	*hag*	GCTATAAACAACCTTCCCTCTCAGTCGATATATAGTATGA
BMD_4154	*sigD*	BMD_0526		TTCTTCAAAAATGAAAATGAAAAGCCGTTATTAGTAATAT
BMD_4154	*sigD*	BMD_0493	*mmsA*	ATAAAATTAAATCGCCGACTTTTTAGTCTACATTAATAAC
BMD_4154	*sigD*	BMD_5120	*lytF*	AGAGAAATTTAAGGTCGTATAAATATGTATATTATAAAAA
BMD_4154	*sigD*	BMD_5093	*gbsA*	TACCAATTATATAGACTTACAAATTCTATAACAGTTTTTT
BMD_4154	*sigD*	BMD_4404	*mmsA*	ATAAGTTATTATTTCCATCTAAACAGTAAAGAATAAAAAA
BMD_4154	*sigD*	BMD_3882		GCCTCAACTCTTGGATGTAAAACGCCGAAATATAAGAAGA

BMD_4154	*sigD*	BMD_3813	*putC*	ACTATTAAATATAACTTGTTGAAACAGATAACAAAAAAAC
BMD_4184	*codY*	BMD_1456	*dppA*	AAACATAAGAATAATTTATAATGAA
BMD_4184	*codY*	BMD_0954		AACAAAAATGATTGTCCTGCAGCTA
BMD_4184	*codY*	BMD_0790	*alsS*	AAAACATTTTACAAATATGAAGGAA
BMD_4184	*codY*	BMD_1131		AAAGCGTATGATTATATTTCCGATA
BMD_4184	*codY*	BMD_1098	*hag*	TAAATTAATGAAAGCTATAAACAAC
BMD_4184	*codY*	BMD_4686	*ilvB*	AAAAGATTTTTAAAATTAATTAACA
BMD_4184	*codY*	BMD_2697		AAAATAATTCTTCTTATAGAATAGC
BMD_4184	*codY*	BMD_1683		AAAAATGTCTTTAACATAGGAACCT
BMD_4211		BMD_4211		ATAATCATGGATCATAATTTTCAAGCCTACCACCTGTTACTTCGCGTCA
BMD_4211		BMD_3353	*fabI*	AACTTCTTGCCCCGTAAGTCGGACTGTAAGCAAAACTTAATTACCTACT
BMD_4211		BMD_2504		TATTGTGTAAAAAGATGTTACATTTATTTTTTGGTACCAGATATTAATA
BMD_4211		BMD_2082		ATGCCTTTGCGCTTTTTATCTAACAATAAATTAGTACCAGATATTAAAA
BMD_4211		BMD_0696	*fabH*	TATAGACAAAAAATAAAAAATGAAGTAAAATTAGTACCAACTCATAATA
BMD_4245	*pyrR*	BMD_4245	*pyrR*	ATAAGAACCTTTAAGAACAGTCCAGAGAGGCTGAGAAGGAAACGGTGCAGAAAA
BMD_4258	*sigE*	BMD_4885	*glgB*	AAAGGATACAATATACCTTAATAAGTATTTCATTAATTCA
BMD_4258	*sigE*	BMD_0445	*yhbH*	GTATTTGTATATGTACTGTGATCTGTGATATTTTACTTAT
BMD_4258	*sigE*	BMD_0444		AAAGCATGTTCAAACATTTTGCTGCATAGATTGTATTAAT
BMD_4258	*sigE*	BMD_5261		TATCATTGTCATATGCTTCTAAAACTTCTAAGATATGTAT
BMD_4258	*sigE*	BMD_5245	*walR*	TATAATGAAAACATAAATCTCTACCAACTTTTAGTATATA
BMD_4258	*sigE*	BMD_4758	*ytvI*	TGCATTTATAAAATACTCATCGAAAAAGAAAACAACTGAT
BMD_4258	*sigE*	BMD_4752	*phoP*	TAAATATAGAAGAAACTGTTTGGTCATATTTTCTGAAAAT
BMD_4258	*sigE*	BMD_5129	*spoIID*	CTGTCATAAAAGGTTGTCCACCTCCATACATATGGAATAT
BMD_4258	*sigE*	BMD_4691	*gerM*	AAAGAAACATTAAAACCTTTTTATCATACAGATATGGATA
BMD_4258	*sigE*	BMD_4622	*spoVB*	AAAATTATGTACATAAAATGTTTGAGATGAAATTATTAAG
BMD_4258	*sigE*	BMD_0219		TCTGCATATTCACTATTTCTTGTTCATATATTCATAGTGT
BMD_4258	*sigE*	BMD_0174	*pdaB*	CTGTTATTAATTGTAGACAAAACGCATACAAATTAGACAA
BMD_4258	*sigE*	BMD_4527		TTTTTTTACCAACTTTTCTTCTTTCATATATTTTACTGAA
BMD_4258	*sigE*	BMD_4456	*spoIIIAA*	TTGTGCTAACTCTTCCTTATTTTGCATAGGTTGTACTAAA
BMD_4258	*sigE*	BMD_4393		AAAAAATTAAAATTCAATAAAAATTATAAATTTTTACAAA
BMD_4258	*sigE*	BMD_4386	*spoIIM*	ATGTCTTATTCTCTTTTTTTATTTGATAGATTCAAAAGAT
BMD_4258	*sigE*	BMD_4361	*dacB*	ACAAGAAAAAACATACCTGTTTAGAAAAGTAAGTATTGGT
BMD_4258	*sigE*	BMD_4321	*spoIVA*	ATCATAAATTTTTCTATCAAAAGTCATATTTTAATAGGAC
BMD_4258	*sigE*	BMD_3819	*spoIIP*	AAAAAAAGTGTCATACGGGATTATATTCGAAAAAATACAA
BMD_4258	*sigE*	BMD_4286	*ylbJ*	AAGCATATAACTATATATCTTCTTGAACATGTAAACGGAT
BMD_4258	*sigE*	BMD_4097	*cotE*	TTGTCACGTTTTTCTATATCCTTGCATACAATGGAACATA
BMD_4258	*sigE*	BMD_4091	*spoVK*	CCGAAAAGTTTGCTGAAATTTTTACATATATTGCAGGAAA
BMD_4258	*sigE*	BMD_2639		GAAATATTTTTTATTCTATGTAAGTTTTGTTTAACTTACT
BMD_4258	*sigE*	BMD_3001	*spoVB*	AAGGTATATAACGGTTTTTTTTTCACATAGGTTGAACATTA
BMD_4258	*sigE*	BMD_2593	*spoIIP*	TTATCATAATTAGAAATCTCCATTCATATAATCATTACTA
BMD_4258	*sigE*	BMD_2296		TCGGAAAAACTTATAAAAGAAAAGAAAATAAAGGAAGAAA
BMD_4258	*sigE*	BMD_1713		TCAGTTATATATACAAACAATACTCATATACATTTATTGA
BMD_4258	*sigE*	BMD_1658		TTGAAATCTTAACTGAAAATCCAGATTATTTTTTAGAAAA
BMD_4258	*sigE*	BMD_1550		ACGTAGTACAAGCTAAGCAGGCTGCATACATTTATCATAG
BMD_4258	*sigE*	BMD_1351	*ctaA*	AAGTTAGTTCAAATACCTCTTATTTACTTTAAATACAAAT
BMD_4258	*sigE*	BMD_1275		CTCGTCTTTTTTTTATCGTTTCTACATATAGTGAACTATG
BMD_4258	*sigE*	BMD_1205		TTTGAATGATAACTTTAGAAATGTCAAAGACAATTAAATA
BMD_4258	*sigE*	BMD_0656	*asnO*	CAATTCAACTGTAACTATTGCGTGCATACAATGGCATTAT
BMD_4258	*sigE*	BMD_0571		TCCTTCTAATTGTGCAGTTCTGCTCATAAGCTTAGTAACA
BMD_4258	*sigE*	BMD_0540	*spoVR*	AACACATGAATTGACATGTTTGTACATAGATTGAAGTAAG
BMD_4354	*resD*	BMD_0778	*nasB*	TAACAAAATTTTATATA
BMD_4354	*resD*	BMD_1140	*nasD*	ATAAATTTTTTAAGAAA
BMD_4354	*resD*	BMD_0677	*fnr*	GTCAGCGTTTTTAAAAT
BMD_4354	*resD*	BMD_5217	*cydA*	AACAGTGTTCTTTCAGT

BMD_4354	*resD*	BMD_0366	*hmp*	AAAAACTTTATAATATG
BMD_4377	*sigF*	BMD_1452		AGGATTAAAAATATGCGATTTTACAAGTATTAACAG
BMD_4377	*sigF*	BMD_1383	*ponA*	TATAATTACAAGAGATGAAATGTTCTTTATATGAAC
BMD_4377	*sigF*	BMD_0837	*glcU*	GACAAAAAAACGCAATCTTTCATATAGAAAAAGAGA
BMD_4377	*sigF*	BMD_0809		AAGAATAATACATTTACTTATAATATATAATCATAT
BMD_4377	*sigF*	BMD_1204		GTCGAATAAAAAGGAAGTACATGAACAAACTACCAG
BMD_4377	*sigF*	BMD_1145	*glcU*	AAGAAAAAATTTTTTTCTGTTTTTAAAAAATAATCA
BMD_4377	*sigF*	BMD_0600		TTGCATACAATTTACAATTGTAAATCATACTAAAAT
BMD_4377	*sigF*	BMD_1053		TTTTTAAAAAAAGTCACAAAATCGACATATTATTAT
BMD_4377	*sigF*	BMD_4949		AACGAATAATTTGTTTTTATGTTGGAAACAAATAGA
BMD_4377	*sigF*	BMD_4879		CGAGAAAAAATTAATTTCGCTTGTCTAAATTATGCC
BMD_4377	*sigF*	BMD_4854	*ytkD*	AATATTATATATCTCTGAGTTCAGGAGAAATCAATC
BMD_4377	*sigF*	BMD_5262	*yyaC*	AATCTTCAAAATCTTCGTATACTGTTACTATTATGT
BMD_4377	*sigF*	BMD_5226	*katA*	AATAATAAAGATTGTATTATTTTAAGTTTATTCTCC
BMD_4377	*sigF*	BMD_5179	*pbpF*	ACGTATCTAATGGGATTGTATTAAATATGTTTGTAC
BMD_4377	*sigF*	BMD_5153	*spoIIR*	ACTATAATTACCTTTTCTAGCTTACCAAAAACGTGT
BMD_4377	*sigF*	BMD_5128	*spoIIQ*	AATGTATATTTTAGATTCATCTGTTGAAAAATAATC
BMD_4377	*sigF*	BMD_5074	*ctpB*	ATGTATCAAAACCAACATAGATAATATTATTCTTCT
BMD_4377	*sigF*	BMD_4586	*pbpI*	TTGTTAAAAATTGTCTTTTAATTATTATTGTTGAAC
BMD_4377	*sigF*	BMD_4557	*gpr*	ATGCATGAATCTGCCAGTCCTCGGTGACAATAATTT
BMD_4377	*sigF*	BMD_4462		CGAAATTTATTTTTTCTGTAAAGGACAAATACGCAT
BMD_4377	*sigF*	BMD_4437	*spoIVB*	CATTGTCATACACAATTTCCACAAAAAAATATGTTT
BMD_4377	*sigF*	BMD_0074	*spoVT*	AGTGATATTCACCAACTATTTTAATCATCATATGAA
BMD_4377	*sigF*	BMD_3983	*gerAA*	TTCAATAACACGGCTATTTTAAAAATGTATTATGCA
BMD_4377	*sigF*	BMD_4380	*dacF*	AAGGTCGAATAGTTTTTATTCTTCTTTTATACTGAA
BMD_4377	*sigF*	BMD_4376	*spoVA A*	GTGAATTAGAAAATTTAAGAAAATCCATACTACACG
BMD_4377	*sigF*	BMD_4336	*sleB*	CTTGTATAAAAAAACAACCTTTACAAAAAATACTCC
BMD_4377	*sigF*	BMD_3768		AAAGAATAATAAAACATCATATGGAAACGTTAAGCG
BMD_4377	*sigF*	BMD_4005	*azoR*	TCTATTAAAAAAATCATTATAGAACTTAATTACTAT
BMD_4377	*sigF*	BMD_2966		GGGAAAAAAATTATTTATCAAACTTATAAGAAAATA
BMD_4377	*sigF*	BMD_3385		CGAAATTTTACTTTATTTGGAAACAAAAATTTCCCA
BMD_4377	*sigF*	BMD_3384		AGGTATAACAAACTTCCTCCTATGCATTATACCTGC
BMD_4377	*sigF*	BMD_3214		CTAGAATAAAAAAGCTTCTTAGGGAAACAGTAGTTA
BMD_4377	*sigF*	BMD_2782		ATTTATTAAGCCTCATTTTATAATACAAACTACAGA
BMD_4377	*sigF*	BMD_3170		AACGCTTAATAACTCACTTCTCAGAAAATATAGAAG
BMD_4377	*sigF*	BMD_3109		AGTGAATATCAACGAACTTTCTGCTGAAAATGATGT
BMD_4377	*sigF*	BMD_3040		AATAAAAAAAAACTTTACACTTTGAAAAAATTTTAA
BMD_4377	*sigF*	BMD_3025		CATACATTTTTTAGAAAACTATGGTAAAAATAAGAC
BMD_4377	*sigF*	BMD_3019		ATGAATAGAAAAAGTAATCGTTTATAGTAGAATATT
BMD_4377	*sigF*	BMD_3000	*pbpF*	TTCGTATAAAAAGAGCCCTAGTCTGATAAATTAGTT
BMD_4377	*sigF*	BMD_2591		CTAAATAAAAATTCTTAAAAGACAATTAATTAGAGA
BMD_4377	*sigF*	BMD_2203		ATACATTATACGCTTTCATGTGTATGTTAATATTTA
BMD_4377	*sigF*	BMD_1747	*gerKA*	CATAAATATTTATTGTTAATTTTTACATAAAAAACT
BMD_4377	*sigF*	BMD_2184	*splB*	ATGTATAATAAGATATTTGATTATGAACAATATGAC
BMD_4377	*sigF*	BMD_2174	*gerBA*	GATGGATGTTGTATTCCTTTGTATAAGAAACAAAAA
BMD_4377	*sigF*	BMD_2142		GAAAATAGAAAATAAACTATTTTTTCTAGAAAATGT
BMD_4377	*sigF*	BMD_1651		CACGATTAAAATATTGCTGCATTGATAATATATAAA
BMD_4385	*fur*	BMD_2762		AATATTCACTATTACTAAAAGTAAAAGCTAAGTACGT
BMD_4385	*fur*	BMD_3007	*bmrA*	TTTATATTGCGATGATGACAAGTATTCAAGTCTAAAA
BMD_4385	*fur*	BMD_1511		TAAGTTTACTCTTACTATTAGTTATTGTTAATTAACG
BMD_4385	*fur*	BMD_1510	*fhuC*	GCAATTAATTGTTATTGATTATCATTCTCATTTGAAT
BMD_4385	*fur*	BMD_0906	*malQ*	AGCGTGGACAACTATTATTAGTTTTTCTCGAAAAAGG
BMD_4385	*fur*	BMD_1317	*yknU*	TTTTGAAAAAATATCTAAAAATTATGGTAATTTATTT
BMD_4385	*fur*	BMD_0872	*yfmC*	ATAAAACTGATATAATGATAACGATTATCAGTTTCAA
BMD_4385	*fur*	BMD_4999	*yfiZ*	AGAATTCACTATTACTAAAAGTAATTGTTAATTCTAT
BMD_4385	*fur*	BMD_4952	*yumC*	CAAAACGTGTGGTTATGTTAATGATTACAAATACTGA
BMD_4385	*fur*	BMD_0409	*ygaD*	GCGAGTTTATATAACTAATACTAAATCTATAGTTTCA

BMD_4385	*fur*	BMD_0368		TAAATTAAACAAAACTTAAAGAAAAAATTATTTTTAG
BMD_4385	*fur*	BMD_3219	*yclN*	TCAGTTAACTATTACTATTAGTAAATGTATACTATTT
BMD_4430	*bkdR*	BMD_1994		GATGCAAAATCTTGGAGCG
BMD_4430	*bkdR*	BMD_2336		ACTGGAAAAATGTTTGCCG
BMD_4430	*bkdR*	BMD_1844	*gapN*	AATCTATAAATTTTTTTCG
BMD_4430	*bkdR*	BMD_1546		AGGGGGAAAATAATTTCAG
BMD_4430	*bkdR*	BMD_1435		GACACTAATTTTTTTTGAA
BMD_4430	*bkdR*	BMD_0979		AAGTTCTCTTTAATCATAT
BMD_4430	*bkdR*	BMD_0953		GATTAAAATTTCTGTTGTA
BMD_4430	*bkdR*	BMD_0945		GCGTATAATTAAAATATAT
BMD_4430	*bkdR*	BMD_0863		TCGTTGTTTAAAAAATTCA
BMD_4430	*bkdR*	BMD_0663	*rocD*	TCGGTTGATAAAAACGCCT
BMD_4430	*bkdR*	BMD_0662	*rocA*	ACGTTTTATATAACCGCAA
BMD_4430	*bkdR*	BMD_1076		GACTTCTATAAAAAGGCGA
BMD_4430	*bkdR*	BMD_0493	*mmsA*	AAAGGGAAAATCTTGAGAG
BMD_4430	*bkdR*	BMD_5093	*gbsA*	AACTCATAATTATTTTCAT
BMD_4430	*bkdR*	BMD_4429	*ptb*	TTATGTTAGAACGTTTATTGTCGTACGTTATTTT
BMD_4430	*bkdR*	BMD_4404	*mmsA*	ATTAAAAATTTATTTAGCA
BMD_4430	*bkdR*	BMD_4340	*gudB*	ACTTTTTGCAAAAATGCAC
BMD_4430	*bkdR*	BMD_3813	*putC*	TATGATAATTTATATTGAA
BMD_4430	*bkdR*	BMD_4063	*gabT*	AATATAAAATTCCTGTACT
BMD_4430	*bkdR*	BMD_4061		ACGTTTTCTCAAACGATAG
BMD_4430	*bkdR*	BMD_3463	*mmsA*	AGTGCTTAAAACATTTGGA
BMD_4430	*bkdR*	BMD_2413	*rocG*	TGTGAATATTTCTTTTTCT
BMD_4430	*bkdR*	BMD_1876	*acoR*	AAGGAACCATTATGATGCA
BMD_4430	*bkdR*	BMD_1653		TCTACATATTTATTTTTCA
BMD_4430	*bkdR*	BMD_1605		ACTTTTTCTTTAATCAAAA
BMD_4430	*bkdR*	BMD_0666	*rocR*	TTGTTTTCCGTAAGCGCAT
BMD_4430	*bkdR*	BMD_1075		AAGTTTTTGCTTAACATAA
BMD_4430	*bkdR*	BMD_5070		AATTCAAACTGATCTTTCA
BMD_4430	*bkdR*	BMD_4430	*bkdR*	GCATTTTTTTGTAGCGTAA
BMD_4436	*spo0A*	BMD_0084	*spoIIE*	GTTTGACAAAATTT
BMD_4436	*spo0A*	BMD_4259	*spoIIGA*	ACTAAAGGAGTTTT
BMD_4436	*spo0A*	BMD_4033	*kin*	TTTAAAACGGTTTA
BMD_4439	*argR*	BMD_4404	*mmsA*	AATTTTTAAATAAATC
BMD_4439	*argR*	BMD_3813	*putC*	AAATTAAAAAAAACTT
BMD_4439	*argR*	BMD_4063	*gabT*	ATGATATAAATATGTA
BMD_4439	*argR*	BMD_4061		GTGAAGAAATATGAAG
BMD_4439	*argR*	BMD_3463	*mmsA*	AAGAATCAAATGACCA
BMD_4439	*argR*	BMD_1994		GTGTAGAAAAATTCAG
BMD_4439	*argR*	BMD_2336		ATGAGTCAAACTAAAA
BMD_4439	*argR*	BMD_1844	*gapN*	AAAGTGAAAATAATTC
BMD_4439	*argR*	BMD_1546		ATGCGTACAAATTTAA
BMD_4439	*argR*	BMD_1435		ATCAATGAAGATGCTA
BMD_4439	*argR*	BMD_0979		TTTAATAAAAATGAAA
BMD_4439	*argR*	BMD_0953		ATCAATAAATATTTTT
BMD_4439	*argR*	BMD_0945		TTTAATAAAAATGCAT
BMD_4439	*argR*	BMD_0863		ATGAATATAGAGATGT
BMD_4439	*argR*	BMD_0663	*rocD*	ACCTATGAAATAAGTC
BMD_4439	*argR*	BMD_0662	*rocA*	AATAAAAAAAGAAATA
BMD_4439	*argR*	BMD_1076		TTGAAAAGAAATACAA
BMD_4439	*argR*	BMD_0493	*mmsA*	AATTATAAAGAATCTA
BMD_4439	*argR*	BMD_5093	*gbsA*	TTAAATATAAATTTAA
BMD_4439	*argR*	BMD_0678	*argC*	ATAAATAAAAATGTTT
BMD_4465	*mntR*	BMD_2511	*mntH*	AATTTGCATTAAGGAAACA
BMD_4528	*sigA*	BMD_2718	*ileS*	GACGTAATATTCTTTTTTATATATATTTAAAGTAT
BMD_4528	*sigA*	BMD_3164	*cadA*	GTTTGAACATTTTTTAAAATTGACATATATTATAA

BMD_4528	*sigA*	BMD_3148		TATTTCTCTTGCTCATATTGATTATTATCATAAAA
BMD_4528	*sigA*	BMD_3127		TAATTTATATTTACGTATATTTGATTTTCAGTTAA
BMD_4528	*sigA*	BMD_3122		TAATGATTTATATAATTAAATTTGATATAATCTAT
BMD_4528	*sigA*	BMD_3100	*ansA*	TCTTGATTTATTGATTTTTGGTTACTATAATTACA
BMD_4528	*sigA*	BMD_2697		AATATCTTATCGTTTTCAATTACTATTAACGTTCC
BMD_4528	*sigA*	BMD_2687		TTTTGAAACAATCACCAAATTGTGGTATTCTTTCC
BMD_4528	*sigA*	BMD_2620		TTTTGTTTTCATTAAAAATAAATGGTTAAAAATAT
BMD_4528	*sigA*	BMD_2610		AGAGTTATATGTTGTTAATAAACTATACTATTTTG
BMD_4528	*sigA*	BMD_3079		AGTTTACATTTATTTGAATCATAAGTAAAATAGAT
BMD_4528	*sigA*	BMD_3077	*citA*	TAGATAAAATGAATACTAAGTTTATTTACATTTGA
BMD_4528	*sigA*	BMD_3036		AAAATAAAATATTCTTTCAAATCAATTTTATTTAT
BMD_4528	*sigA*	BMD_3000	*pbpF*	TATTTGACAAACACAAATTTATGGGTAAACTAAGT
BMD_4528	*sigA*	BMD_2560	*pucA*	TATTGTAGTTCTGTGTAAAACGTGAAAAGATTTTA
BMD_4528	*sigA*	BMD_2546	*acnA*	TATTTACTTATGTATAATGTTATATTAAGATAGAT
BMD_4528	*sigA*	BMD_2525		GTTTCTATTTTCTGAATCTATTTTTTATAATGACA
BMD_4528	*sigA*	BMD_2505	*pucK*	TAATATTTATTAACATTACATGTGATACAATTAAA
BMD_4528	*sigA*	BMD_2490	*yhcY*	TAATTAAAATTATAAAAAAAGAAGAGTAAAGGAAA
BMD_4528	*sigA*	BMD_2437		AGTAGCAAGTACTAATAGTTAGTAATATAATAATA
BMD_4528	*sigA*	BMD_2434		CTTTTACTTCGGAAAAATTTTTTGTTATTATGTTT
BMD_4528	*sigA*	BMD_2413	*rocG*	AATATTATATTCTCACAACAATTCATTGTCTTTAA
BMD_4528	*sigA*	BMD_1994		TTGTTGATTTTACTAGATAACATGGTAAAATATTC
BMD_4528	*sigA*	BMD_1992		AGTTGACTTATTGGTATTGTTATATTATAAGTAAT
BMD_4528	*sigA*	BMD_2385	*htpG*	CATATAAGATCGTTTGAGAAAGTCTTGCAAGTTGG
BMD_4528	*sigA*	BMD_2359	*thrS*	TACATCTTTTTTTCTTATATGTTCGTTACTTTTCC
BMD_4528	*sigA*	BMD_2341		CAGATAATTTTGCTTTGATAAAACTAAAAAGTTTC
BMD_4528	*sigA*	BMD_2336		GTCGTAAGATAGCCTGTAAATGGCCGATAATTTCT
BMD_4528	*sigA*	BMD_2304		CATTTACAGTAAATAATTTAATGGTTAAAATAAAG
BMD_4528	*sigA*	BMD_1878		TCTGTACTATATACAGTATATTGTATACAATATAT
BMD_4528	*sigA*	BMD_1876	*acoR*	TAATGATAACGTTTTCTTTGTGATTTATAATAATG
BMD_4528	*sigA*	BMD_1860	*xylT*	TACTTAATATATTTCTTCAGATGGAAGTTTGTTTG
BMD_4528	*sigA*	BMD_1858	*xylA*	ATTATAATATTGATTTAAAAGATTTTTTTTGTAAC
BMD_4528	*sigA*	BMD_1844	*gapN*	ATTGTACGAATGAAAGAAATTTATATATATTAAAA
BMD_4528	*sigA*	BMD_1817		ACTATCAAATAAACAAGATAAAAATAATCCTTTTC
BMD_4528	*sigA*	BMD_2294	*citM*	GCATGAAAAAAATGAAAAAAATTACTTTTATAAAA
BMD_4528	*sigA*	BMD_2279	*fumC*	CTTTTTTTATGTTAGAAAATCAAAATAAACTATGA
BMD_4528	*sigA*	BMD_2251		ACAATCATATTTTATCAGATCATTGACATGTTTTT
BMD_4528	*sigA*	BMD_2238		TCATGAGTTTGTTACATATTGCTGATAGAATAAGA
BMD_4528	*sigA*	BMD_2224		TCTAGTTATTACAATAGTAATGTTTTATAATATTT
BMD_4528	*sigA*	BMD_2213		TGTGTTGAATAAGGGAGTTTACTGATAAAATAAAA
BMD_4528	*sigA*	BMD_1777		TTTGTGAAATTCTCTAAATTAATAAAATTGTTTAA
BMD_4528	*sigA*	BMD_1768		AATATAAAAACCTCCAAAATTATCTATAAAATTTT
BMD_4528	*sigA*	BMD_1741		TATTGACATTTAATTTCAAACTAACTATGATGATA
BMD_4528	*sigA*	BMD_1726	*recA1*	CCTTTACAATTAGATGCGTTTATTGTATGATTACA
BMD_4528	*sigA*	BMD_1722		ATATTAATAGCGGACGAATTATAGATAAAATTAAA
BMD_4528	*sigA*	BMD_1701		ACAATCATATTTTATAAAATATTAACTTTATTACT
BMD_4528	*sigA*	BMD_1683		TGTTTAAATTTACTAAACCTTTGGTTAGGATGCTT
BMD_4528	*sigA*	BMD_1680		ATTTTATCTAGAGCTTGATATCTTGTAAAATAATG
BMD_4528	*sigA*	BMD_1653		AATTTCATTTTTATCGAAATTTTTCTATAATATTA
BMD_4528	*sigA*	BMD_1605		CATGTCATAAGGTTCTGATTAGAGACTTCCGTTCG
BMD_4528	*sigA*	BMD_2082		TAAATAAGGTCTTCTAATAAGATAAATTATTTTTC
BMD_4528	*sigA*	BMD_2055	*gltA*	GATTTAATATATAACAAATATAACTAAACTTTTGA
BMD_4528	*sigA*	BMD_2054	*gltC*	AGTTTTCAAATCAATATAAACAATATATAATTTAG
BMD_4528	*sigA*	BMD_2026		TTTTAAAACATCAAATTAAAGCTGATACGCTACAA
BMD_4528	*sigA*	BMD_2015		TTTTTGAAATTTGTGTATGATTCTCTATATTAGAA
BMD_4528	*sigA*	BMD_2014		TAAAATTATTTCAAAATAAATTTTTTATAATTAAA
BMD_4528	*sigA*	BMD_2011	*allB*	GAGTGTAAAAATCAGATAATTCAGTTATATTTACT
BMD_4528	*sigA*	BMD_2006		CATATAACATATGTTATATGTTTTAAACATGTATA
BMD_4528	*sigA*	BMD_1590	*map*	CAAATCGTATACTATTTATAGAATGATTAGATTAA

BMD_4528	*sigA*	BMD_1589		CCTTTAATATAAGGCTGCAAATAATCAAACTTTCT
BMD_4528	*sigA*	BMD_1582		TATTTAAAATTTTCTGTTTTTGTTATTCCTCTTTA
BMD_4528	*sigA*	BMD_1568	*sacB*	TTGATAATATATTATTTAATAAGGTGATAAAGTGT
BMD_4528	*sigA*	BMD_1564	*fruR*	AATATAAATTTTTTTTCTAACATAATCCACATTTT
BMD_4528	*sigA*	BMD_1560		TTTTTAATATTCATAAAATTTAAATTATACTGAAA
BMD_4528	*sigA*	BMD_1546		AAGTTATTAATTGTTTGATTTTCCTCTCTAGTTTA
BMD_4528	*sigA*	BMD_1536	*sacB*	TACGTAATTTTTTCTATTTTTACTCATCCATTTTT
BMD_4528	*sigA*	BMD_1523	*opuAA*	ATTGTTAGATGAAATTTTATTCCCCCTTTATTTAA
BMD_4528	*sigA*	BMD_1512	*megL*	AGGGAACGAAAAAAGAAAAGAGAAATATAATAAAA
BMD_4528	*sigA*	BMD_1482	*galU*	AAACTCCCATTTCAATAAAATAGATAACCATTTTT
BMD_4528	*sigA*	BMD_1474	*des*	TCTTAAATAGTCAGAAAAATCTCTTTATACTGAGG
BMD_4528	*sigA*	BMD_1456	*dppA*	TGTTGTAAAAAGAAGAAAGATAATTTATAATTTTC
BMD_4528	*sigA*	BMD_1435		GGTATCATATCTTCCCTTCAAGAGTATTTAGTTAC
BMD_4528	*sigA*	BMD_1432	*xpt*	TATTTTCAAATAAAAGATTTGCTGATATAATACAA
BMD_4528	*sigA*	BMD_0979		TGTTTTATATAAAAGTTTGTTTAATAAAAATGAAA
BMD_4528	*sigA*	BMD_0957	*scrA*	TGTTGACGAACTTGTATATACAGGTTAGAATAAAA
BMD_4528	*sigA*	BMD_0954		GCTGAGTTTATTACCTAAAAAAATATATAGTGAAT
BMD_4528	*sigA*	BMD_0953		ATCTTAATATTATGTTTAAAATGACAAACATTTTT
BMD_4528	*sigA*	BMD_0945		AATTGATTTTAATAAAAATGCATAGTAAAATAAGT
BMD_4528	*sigA*	BMD_1396	*metB*	TGTTGACAATAAAGAAAATCTTTATTAAAGTTAGT
BMD_4528	*sigA*	BMD_1383	*ponA*	TATAGATAAATTATACTTTTAAGAGTATGATATAT
BMD_4528	*sigA*	BMD_1351	*ctaA*	TGTTTAATATACAAAAAGATTAGGATAAATTGTTT
BMD_4528	*sigA*	BMD_1326	*pdhA*	GTTTTAAAAAATCGTTTATTTGTGGCAAGATAGGA
BMD_4528	*sigA*	BMD_1303		GTTTCACTTTATTCAGAAAAAATAGTAAAATGATA
BMD_4528	*sigA*	BMD_0868		TAAATAAAAAGTAATATTTCTCTCTTCAAAGTAAC
BMD_4528	*sigA*	BMD_0863		ATATTCAAAAATTTTCAAATTTTACTGACTGTTAG
BMD_4528	*sigA*	BMD_0852	*citM*	TATTTTAAGAAAAAACAGAAAATTTTAAAGTTTTA
BMD_4528	*sigA*	BMD_0848		TCTTCTCCTTTTTAATCAAATGATCTACACTTATA
BMD_4528	*sigA*	BMD_0833		AATATAAATAGTTAAAATTATCAGTTAATATAAGG
BMD_4528	*sigA*	BMD_0827		ATTTTAAATCTAGTGAGCATTATGTTAAGATTACA
BMD_4528	*sigA*	BMD_0817		ATTATAAGATTGTATTAATATGTTTAATTAAATGT
BMD_4528	*sigA*	BMD_0804	*glsA*	AATTCGAAATGTTTCATTTAACGGATAAAATTTTA
BMD_4528	*sigA*	BMD_1287		TATATAATATGAAGATTTTATTTGTTACACATTTA
BMD_4528	*sigA*	BMD_1282	*ptsG*	ATATTGTTTTTGCCTAAATATTTTAGTAAAGTTAT
BMD_4528	*sigA*	BMD_1249	*clpE*	ACTTGAAAGTCAAAAAAGGTCAAAGTATAATGAGA
BMD_4528	*sigA*	BMD_1234	*mtnW*	TAAATCATATATTAGATGCTTAAGTTAAATTTATT
BMD_4528	*sigA*	BMD_1233	*mtnE*	ACTTTACAAGTTGAGTATTACAATATATTATTTGT
BMD_4528	*sigA*	BMD_1232	*mtnU*	TGTTTATTATATAACATTATGAGTTGAACATTTCA
BMD_4528	*sigA*	BMD_1231	*mtnK*	TATATAAGAATTTGGATAATTGAATCATTTGTTTA
BMD_4528	*sigA*	BMD_1205		TGTTTACGAGCTGCTTAAAATAAGTAATGATGTAT
BMD_4528	*sigA*	BMD_0790	*alsS*	ATTTTACAAATATGAAGGAATTTCATACTATTAGA
BMD_4528	*sigA*	BMD_0781	*speE*	AGTATAAAATATATCTTGTGTTTAAAAACTTTATC
BMD_4528	*sigA*	BMD_0778	*nasB*	TCATTTCTTTTAGAGAAGTTTATGAAATAATAAAA
BMD_4528	*sigA*	BMD_0769	*licR*	AATTGAAAAAAAAGAGAAAGAATGATAGATTACAA
BMD_4528	*sigA*	BMD_0765	*gabP*	AATGTTGTATGATATTAAGAGGTATAATTGATTTT
BMD_4528	*sigA*	BMD_0755	*ncs1*	CATTAAATATAATAAGGAATAACGTTAGTATTTAT
BMD_4528	*sigA*	BMD_0719	*mecA*	ATTTCCTTTCTTTCGCATTTTGTCATAAAATAGAT
BMD_4528	*sigA*	BMD_0714	*spxA*	AATATCATATTACGTTTATCTATAATAAAAAATTT
BMD_4528	*sigA*	BMD_0712		CTATCGCATTTTTTGTGATTTTTGATAAAATGAAC
BMD_4528	*sigA*	BMD_1161	*nagE*	AAATTCACATTGCGAAAAACCTTATCAACAGATCT
BMD_4528	*sigA*	BMD_1140	*nasD*	TAAGTAATATTTCTTATACTCTCATTATCATTTAT
BMD_4528	*sigA*	BMD_1131		TAAATCAAATAAATTTAATAAATCGACTTTTGTTC
BMD_4528	*sigA*	BMD_1127		TCTTTCTTTTTTGCTCTTATTATAATATAATTTGA
BMD_4528	*sigA*	BMD_1126	*galU*	CATATAGGATATTATATATAATTTTCCTCTTTTCT
BMD_4528	*sigA*	BMD_1114	*galU*	TTTTTTAAAACGTTCATTCATTAAGTAAGATATAT
BMD_4528	*sigA*	BMD_0696	*fabH*	TATAGACAAAAAATAAAAAATGAAGTAAAATTAGT
BMD_4528	*sigA*	BMD_0687	*clpB*	CTTTACTTGTTTTCCTATTGCTGGGTATAATGAAA
BMD_4528	*sigA*	BMD_0678	*argC*	ATTTGATTTTATTTTTATACGGTATTATAATGATA

BMD_4528	*sigA*	BMD_0677	*fnr*	ATAGTAATATTATGGCATATTTTTATTTTAGTTTA
BMD_4528	*sigA*	BMD_0666	*rocR*	ACTATCAGAATTCAGCTATTTTTGTTTAAATAGAG
BMD_4528	*sigA*	BMD_0663	*rocD*	TTTGTAAGATTAATCTTTTATTCCTCCTCATTTAT
BMD_4528	*sigA*	BMD_0662	*rocA*	ATTTTTTTCTTTATCAAAAAGAAAGAATGATATAA
BMD_4528	*sigA*	BMD_0642	*addB*	AGTGGAATTTTGTCTGCAAGTTCAGTAAAATAATA
BMD_4528	*sigA*	BMD_0612	*lcfA*	TCTTTTATATTGTCTTATATTTGATTGCCTTTTTC
BMD_4528	*sigA*	BMD_0600		ATTTTAATATTTTAAAATGTGTGTTTTTAAGTTTT
BMD_4528	*sigA*	BMD_1076		TTTGTCATATTTTGCTGTTGTCTTTATCCCCTTCG
BMD_4528	*sigA*	BMD_1075		TTTATATAATTACGTAAAAATTGAAGTAAAGTTAA
BMD_4528	*sigA*	BMD_1062		AACTTAACATAGACACCTTATGTAATTCACGTTTA
BMD_4528	*sigA*	BMD_1046	*galE*	AAGGTAATTTTTTATTTCATTTTCTATACTTCTTT
BMD_4528	*sigA*	BMD_1038		TATATAGCATATCTTCAAACTATCTTTTCTTTTAT
BMD_4528	*sigA*	BMD_1020		AACGTGGTAGACTATCAAGTCAAGTTTACATTTCT
BMD_4528	*sigA*	BMD_4992	*fadN*	TATTGCGAAAATTTAAAAAATGTTTTATAGTAAAA
BMD_4528	*sigA*	BMD_0573		CTTCATCAAATTGACAGCTTTCGTTTAAAATAAAT
BMD_4528	*sigA*	BMD_4918	*cysH*	TATTCTAAATATGCTAAAATTTTTAAAAAATGCTT
BMD_4528	*sigA*	BMD_4910		GAAATAAAATAATGTTATAAGCGTAAGTCATATGA
BMD_4528	*sigA*	BMD_0533	*glpF*	AAAGTAGTAGATCATTTGAAGATCTATTCAATTAT
BMD_4528	*sigA*	BMD_0524	*ldh*	ATTTTAACAAAGGATAAAAAATTTCCCACACTAAA
BMD_4528	*sigA*	BMD_0514		CTAATAAAATTATTTCGTTGTAAAAATTTTGTTAT
BMD_4528	*sigA*	BMD_4878	*menF*	GCTAGATTTTTTTGAACTGAAAGTTTATAATAAGT
BMD_4528	*sigA*	BMD_4861		AACATAATATTATGAAAAAGGATATTCTACTTTTC
BMD_4528	*sigA*	BMD_4854	*ytkD*	GGCATTACATACTATTTTATTCCCTTTTTAGTTTA
BMD_4528	*sigA*	BMD_0493	*mmsA*	GTTTGACATGATGGAAAAACTTTTAAATTATAAAG
BMD_4528	*sigA*	BMD_4848	*pckA*	GTATAGAATAATTAAATAAATGTGTTATACTAATT
BMD_4528	*sigA*	BMD_0478	*yocH*	AGTTTAAAATTGTACTAAAACTACTTTAAATTAAT
BMD_4528	*sigA*	BMD_4826	*ytkP*	TAGATAAAATTTCCAAAAAAGCCTACTTTATCTCT
BMD_4528	*sigA*	BMD_4812	*ftsK*	ATTGTCAATTCTTAAAAAGGTTTGATAAAATTAAA
BMD_4528	*sigA*	BMD_0453		AAATTAAAATATTTTTGACTTATTATTCGAATTAG
BMD_4528	*sigA*	BMD_0417	*perR*	TATTTTATATACTTTTCAAAGATGAATAATGTATG
BMD_4528	*sigA*	BMD_0405		CAAGTAAAATAGTTCAAAATAGATAAAAGGGATAT
BMD_4528	*sigA*	BMD_5270	*oxaA*	TCTATAGCATATTTTTTCTTACTACTTAAAGTTTG
BMD_4528	*sigA*	BMD_5250	*purA*	AATTGACTTTAGCGTGATAAATTGATACACTTATC
BMD_4528	*sigA*	BMD_5245	*walR*	TATTTGCTTTGTATCTCTTTTGTGGTAATATATTA
BMD_4528	*sigA*	BMD_5240	*htrA*	TGTTTTATGTCTTTCATCAAGGGTCTAAAAAATTA
BMD_4528	*sigA*	BMD_5217	*cydA*	TCTCGAATTATTATGCTCAATGTTTCACAATGCAA
BMD_4528	*sigA*	BMD_4799	*acuA*	TGGATAATATATTGTTTTTATCTGAAAAGTGTTAA
BMD_4528	*sigA*	BMD_4798	*acsA*	AATTGTGAAAAGTCTATTTTTGTTATATAATAGGT
BMD_4528	*sigA*	BMD_4797	*tyrS*	ACTTGAATTTTAAAATAAAATCATTCATAATTGTG
BMD_4528	*sigA*	BMD_4796	*rpsD*	TATTGACAGTCCAATCAAAAATCGCTTTGATGAAA
BMD_4528	*sigA*	BMD_4791	*ezrA*	TAAAACATATTTTTTTAAATTAAAAAACGAGTTTT
BMD_4528	*sigA*	BMD_4790	*iscS*	AATTTAAAAAAGTATTAAGGTATGTTATGATAATA
BMD_4528	*sigA*	BMD_4779	*ackA*	TCTTGAAATCTATTGTCGTGAATGATTAAATGAAT
BMD_4528	*sigA*	BMD_4773	*ald*	CGTTCTTTTTTTAGTTAATCACTGATATAAGAAAA
BMD_4528	*sigA*	BMD_4756	*citZ*	GATTCACAAAAGTCAGATAATTGTTTATAATATAT
BMD_4528	*sigA*	BMD_4752	*phoP*	GTAGTAATATCGTGCTAACTTAAAAATAAACTTCA
BMD_4528	*sigA*	BMD_0391		TGTTGACAAAGCATCCAACTGGTATTAGAATAAAC
BMD_4528	*sigA*	BMD_4746	*gap*	TAAATACAATATGATAAGTTATATTTTTCAATTTA
BMD_4528	*sigA*	BMD_4745	*speH*	TATTGCAAAATAAATGTAAACAAAGTATACTATGT
BMD_4528	*sigA*	BMD_4739	*thrS*	GGTTGATTTCTATTTTGAAAATGAATATAATAAAT
BMD_4528	*sigA*	BMD_4738	*infC*	TGTTGCATAAATGTTTTTCCTTTGATACAATAATT
BMD_4528	*sigA*	BMD_0375	*cadA*	TGTTATATAATACAATTAAATATAATAAACTGTAT
BMD_4528	*sigA*	BMD_0372		ATTTTTTAGTTTAAAAATGATTGTATATAATGTTT
BMD_4528	*sigA*	BMD_4729	*pheS*	GTGATATTATTATGTTATCAATGTATACATTTTTT
BMD_4528	*sigA*	BMD_4720	*lcfA*	GATTGCATCTAAAAAAAGAATGAACTATAATTTTA
BMD_4528	*sigA*	BMD_0366	*hmp*	TTTATTATTATCGCGAACGCCTAAATATAATAAAA
BMD_4528	*sigA*	BMD_4713	*lysC*	TCTTGAAATAATAGAAAATATTTGATAAGCTATTA
BMD_4528	*sigA*	BMD_4711	*sdhC*	TGTGTAAATGTTTTGTGAACTTGTTTACAATAAGA

BMD_4528	*sigA*	BMD_4703	*ssuB*	ATTATACTATTTTAAGTTTTATTAGGTTATTGTTT
BMD_4528	*sigA*	BMD_0315		AGATTAGTAGCGTTACTACATAAAATAACATTTGA
BMD_4528	*sigA*	BMD_5199	*ald*	TCTTTATATTGATATTAAGTTAATGAAATATAAAA
BMD_4528	*sigA*	BMD_5197	*gnd*	ATTATAAAATTGTTTTCATTAACAAATTCTTTTAC
BMD_4528	*sigA*	BMD_5188	*pta*	CCTTTGATTTTTAAAAAAAGTGGAGTATCCTATTA
BMD_4528	*sigA*	BMD_5179	*pbpF*	CTAATGGGATTGTATTAAATATGTTTGTACTTTTT
BMD_4528	*sigA*	BMD_5178	*speE*	TTTTTCATGTTTGTATAAATTATGTTAGGGTAATC
BMD_4528	*sigA*	BMD_5172	*fadF*	TAAGTAATATATGTTATTTTTGTCTATATAATAGT
BMD_4528	*sigA*	BMD_5164	*pyrG*	TCTTTAAACAAGCATTTAAAAAGTTTTTACTTAAA
BMD_4528	*sigA*	BMD_5162	*spo0F*	ATCTGTATTCTATCTTGAAATGAGCTACAATGAAA
BMD_4528	*sigA*	BMD_5146	*glyA*	CTTTTTTATTCTAAATAAATCATGTTACAATGTAG
BMD_4528	*sigA*	BMD_5120	*lytF*	TTTTGTAATTTAATAGACATATTGGAAAAATAAAC
BMD_4528	*sigA*	BMD_5117	*galU*	AAAGTAATATTTACTATTTTTATCTTAATATGTTT
BMD_4528	*sigA*	BMD_5109	*degS*	ATAATCATATTTTACAAATACTTTTCTGTAGTTAA
BMD_4528	*sigA*	BMD_5104	*comFA*	AAACTTAGATTTTTTTAACAAGGTAAACTTTTTTA
BMD_4528	*sigA*	BMD_4686	*ilvB*	AAATTTTAATTAATTGTAATTAATTTAAAATATGA
BMD_4528	*sigA*	BMD_4677	*clpX*	AGTATCTATTCAAGAAAATGTATGGTAAAATGCAA
BMD_4528	*sigA*	BMD_4672	*hemA*	TTTTCGAATGACTTATAGAACATGTTATAATAGTT
BMD_4528	*sigA*	BMD_4663	*valS*	GTTGTAAAAAAGTATATCATATTAATTCTCTTTAA
BMD_4528	*sigA*	BMD_4639	*nifS*	TCTTGACACCTATATTTACATATTATTAAATGAAA
BMD_4528	*sigA*	BMD_4638	*nadB*	AAAGTAAATTATTATACATTTATATCCACAGTTCT
BMD_4528	*sigA*	BMD_0271	*purE*	TCTTTGAAGAAAAAGAAATAAATGGTAAAAAAGAG
BMD_4528	*sigA*	BMD_0265		TTTTACCTTTTTATATCTCTTTTTATAAAAAGAAA
BMD_4528	*sigA*	BMD_4610	*pstS*	AATGTGACATAATTTTTTTCTCTAAATACATTTAA
BMD_4528	*sigA*	BMD_4606	*cymR*	AATTTATCTTATAGGCATAGTATGTTATAATATCA
BMD_4528	*sigA*	BMD_0223	*rsbR*	CATATAACATTATTCTTTGCAAAATTTTTACTATT
BMD_4528	*sigA*	BMD_5093	*gbsA*	AAAATCTTATTTTATATTTATCAACCTAATATTTA
BMD_4528	*sigA*	BMD_5091	*gabP*	TAAATATAAAAGTCTTAAATGAAATAAAAGATTAT
BMD_4528	*sigA*	BMD_5084	*secA*	TGTTACAATTTTACTCAGAATGTTTTATAATGAAA
BMD_4528	*sigA*	BMD_5082	*phoD*	TGAATAAAACAGTTACTTTATTTTACCAATTTTTA
BMD_4528	*sigA*	BMD_5070		AGCATAAAATAGTGAAATATTAGTATTTACTTTGA
BMD_4528	*sigA*	BMD_5044	*clpP*	AAGGTAATATCCTCTCTTTTGAATAGAAAAGTTAT
BMD_4528	*sigA*	BMD_5039	*cggR*	AGTTGAATAAAGGATTCACTCCTGTTAGAATAGAT
BMD_4528	*sigA*	BMD_5037	*pgk*	ATTGGATATCTGAAAATAGAACGACTATAATAAGA
BMD_4528	*sigA*	BMD_5008		GCTGTATATCTTAAAAGTTAATAGGTAAAATGAAT
BMD_4528	*sigA*	BMD_4583		AAGATAAAATAGAAGTTAGTAAAGTTTAAGCTTTA
BMD_4528	*sigA*	BMD_4581	*mccA*	TTTTTACCTTAATTCCGACTTTATTTATAAGTTAA
BMD_4528	*sigA*	BMD_4555	*lepA*	TATTGAATGTTGACGAATGTGTTGTTATAATTTAG
BMD_4528	*sigA*	BMD_0126	*rpoB*	TTTTGAATTATATGGGAAAAACTAATAAAATCAAT
BMD_4528	*sigA*	BMD_0111	*gltX*	AAAATCAAATAGTTTTTATTTTTGTCAACTGAAAC
BMD_4528	*sigA*	BMD_4481	*comGA*	TTTTTAATATTTTTTACTTTTTTCTTCCTCCTTTT
BMD_4528	*sigA*	BMD_0092	*cysK*	GATTGACAATTTAGAAAACATCTGATAAATTGTAT
BMD_4528	*sigA*	BMD_0084	*spoIIE*	ATATTATTATATTTTTTCTTCCCAATAACAGTTAT
BMD_4528	*sigA*	BMD_4436	*spo0A*	AAATTCCATTTAAAAGACTTTTTTCTAAAAAAATA
BMD_4528	*sigA*	BMD_4430	*bkdR*	ATTGTCAAGAGCGTTTCTTTTTTTATATGATAGGT
BMD_4528	*sigA*	BMD_0067	*gcaD*	ACTATATTATAAGAATTACCTATTTTTCCATTTTA
BMD_4528	*sigA*	BMD_4404	*mmsA*	TAATGCAACCACCATTTATTTATTCAATAATAAAG
BMD_4528	*sigA*	BMD_0023	*tadA*	CGTTTGAAGCAATACAATCTTCAGGTACAATTAAG
BMD_4528	*sigA*	BMD_3995		AATTTATAAATGTGTCGGAATTATACTTGTGTTGT
BMD_4528	*sigA*	BMD_0001	*dnaA*	AAGATATTATATATAAATGGGTGTATTATCGTTAA
BMD_4528	*sigA*	BMD_3951	*proY*	TTTTAAAATACTATTACTTAGGTGCAATACTAAAT
BMD_4528	*sigA*	BMD_3948		TAGTTAAAATTTGGGTAATAGTTAAACTACTTTAG
BMD_4528	*sigA*	BMD_3938		TTTTGACAAAAATTTCATTTTCTTATATGATAAGT
BMD_4528	*sigA*	BMD_3936		AAAATAAAATATTTTTTCACCTCCCTTTTCAGTAC
BMD_4528	*sigA*	BMD_3922		CTTTTTATTAGCTTTTTTAAAGATCTACATTAAAA
BMD_4528	*sigA*	BMD_3915	*opu*	GCTTCCTAAATGAATTTAAACGTGATATAATTTTC
BMD_4528	*sigA*	BMD_3900	*cssR*	CGTTTAAAATACAATAGATATGATATTTTATTTTC

BMD_4528	*sigA*	BMD_4385	*fur*	ATTTTGACACCCTCGACACATCTGATATAATGGAT
BMD_4528	*sigA*	BMD_4382	*drm*	TGTGTACAAAAAAATAAAAAGCGTCTATACTAATA
BMD_4528	*sigA*	BMD_4354	*resD*	AAAAAAACATGTGATTCTGATCCTCCAACTTTTAT
BMD_4528	*sigA*	BMD_4340	*gudB*	TAAAAAAAATCTGCTAATTTCGTATGATCTTTTCT
BMD_4528	*sigA*	BMD_4319	*folE*	TTTTGCCTTTGGTATATACATATGCTAGAATTTTT
BMD_4528	*sigA*	BMD_4310	*trpE*	TAAATAATATAGGAGAAAATATATTGGAGTAGATA
BMD_4528	*sigA*	BMD_3899		CTTTTATTTTATAGTATAGATAACATAAAATTTGC
BMD_4528	*sigA*	BMD_3890		GGCTATAAAAACAAAATAAACCTAAAAAAATGAAA
BMD_4528	*sigA*	BMD_3813	*putC*	TGTTGCAAATTTTCTAAAAAAGGGGTAATGAAAAT
BMD_4528	*sigA*	BMD_3812		TAAAAGTAATGGGGAAAAAATCTTTTAAACGTTGT
BMD_4528	*sigA*	BMD_3805	*sigH*	GCTTGAGTAAGTTTATTTCATATGGCATTATTTTT
BMD_4528	*sigA*	BMD_4298	*qcrA*	TATTGACCTTCCAGAAATATTAATATATCATGATT
BMD_4528	*sigA*	BMD_4261	*ftsA*	CAGATGTATTTGTTGAATATTTTCATATAATAAAA
BMD_4528	*sigA*	BMD_4259	*spoIIGA*	GTTTTCCTATATTTTTTGTAAGCTGTACCAGTTAT
BMD_4528	*sigA*	BMD_4249	*ileS*	TTTTGGGATGAGGAGGTCTTTTTGTTATGGTACAA
BMD_4528	*sigA*	BMD_4245	*pyrR*	GTATAAATATCGGAAAAACTTCTTTATACAAATTT
BMD_4528	*sigA*	BMD_4211		TGTTGCAATTAGAAGGCAATGATTATATACTGCTA
BMD_4528	*sigA*	BMD_3792		ATTTCCTTTATTTTTAAAAATGTACTAACATGAAA
BMD_4528	*sigA*	BMD_3745		AAAATAAGATAGAGGTTTTCGATTGAAACGATTTT
BMD_4528	*sigA*	BMD_3733	*ansB*	TGTTTTAATTCTTTTTTTAAGCTTTTATAATGCCA
BMD_4528	*sigA*	BMD_3721		GTTGACATTTTCTTATTTCATAAACTATAATAAGC
BMD_4528	*sigA*	BMD_3718		TTTGAAAAGATTAATTTTATCGTAATAAAATAATG
BMD_4528	*sigA*	BMD_4126	*asd*	TGATTTTTATAAATAAAATAAGGTAAACCAGTTCT
BMD_4528	*sigA*	BMD_4120	*ftsK*	TTCAGTATAATTATGGTGAATATGGTATAATGAAT
BMD_4528	*sigA*	BMD_4106	*recA2*	TTTTTCCAAAACAGCCTTTTTTTTATAAAATAACA
BMD_4528	*sigA*	BMD_3683		GTGTTAATAATGTTGTTAATTTATTTTTCATTATT
BMD_4528	*sigA*	BMD_4067		GGATTAATATGTTCTATTACCATGATAAACTTTCT
BMD_4528	*sigA*	BMD_4063	*gabT*	GGATTGTAATGATATAAATATGTAAATAATATTTT
BMD_4528	*sigA*	BMD_4061		TCTATCAGTTTCTTCATCATTTTAATACGGTATAT
BMD_4528	*sigA*	BMD_4002		CATTGACGAATATTGTTGCTTTAAGTACAATGGAA
BMD_4528	*sigA*	BMD_4000	*oxaA*	TGTGTCGTAAATATGAAACTTTTTAAAAAGTAATA
BMD_4528	*sigA*	BMD_3588	*gutB*	AATTAGCTCATGTAATAAAAAACTCTAAAATGTAT
BMD_4528	*sigA*	BMD_3587	*gph*	ATTTTAAAAACGAATGTTTAAAAATTATAGTGTAT
BMD_4528	*sigA*	BMD_3533	*araR*	TCTTTTCATTCATTCACGTTCATGATAAAATATGT
BMD_4528	*sigA*	BMD_3463	*mmsA*	GAATTCGATATGATGGAAAAGCTTTTAAATTATCA
BMD_4528	*sigA*	BMD_3425	*uxaC*	TTTTGTATACATGCTAAATTATTTTTATAATTAAT
BMD_4528	*sigA*	BMD_3407		TTTGTATTATGGTATATATTTTATATTAAATATCA
BMD_4528	*sigA*	BMD_3400	*fbp*	CATTTTACCTATACGTAAAATAGGTTAAAATAAAG
BMD_4528	*sigA*	BMD_2995		ATTGGTTATTTCTTTTTACATTCGATAAACTAACT
BMD_4528	*sigA*	BMD_2994		TTGATAAAATCATTTTTGTTGTTATTTTTCGTCAT
BMD_4528	*sigA*	BMD_2944	*amt*	AGTTGCTAAATACATATAAATCAAATATGATAGTA
BMD_4528	*sigA*	BMD_2926	*odhA*	TTTTGATTGACGAGTAAAAGAGTGGTAAAATATAA
BMD_4528	*sigA*	BMD_3359		TATTGATAAATTGTATCGATACAATTATACTGAGA
BMD_4528	*sigA*	BMD_3330		TAGATAAAATTGATTTTTGTTTGAACTGAATATCC
BMD_4528	*sigA*	BMD_3311	*gltP*	AGAGTCAAACAATTTTTTCGTATTCTTAATTTTTT
BMD_4528	*sigA*	BMD_3309		AAAGTCACATCTCATCGCCACTATAGTACATTTTA
BMD_4528	*sigA*	BMD_3248		CTTGTTAAATAGCGATGAAATTTTTATTTCTTTCG
BMD_4532	*ccpN*	BMD_4848	*pckA*	TTTAGTTAAATAGTATAGAATAATTAAATAAATGTGTTATACTAATTATAAATTTCA
BMD_4532	*ccpN*	BMD_4746	*gap*	AAAAAATTAATATTATTCACTTTTTAAATATTTATGTTATACTATTCAATATAAAAA
BMD_4606	*cymR*	BMD_1992		AAAACATAAAACCTAGTTGACTTATTGGTATTGTTATA
BMD_4606	*cymR*	BMD_2026		TATAAATTTAGACGGACTAGAGGTCAAAAAATTAACCA
BMD_4606	*cymR*	BMD_1589		CTGCCTTTATTAAAAAAACATACAAAGGAAATTATATT
BMD_4606	*cymR*	BMD_1582		ATAATATAGAGTTTAACATAATCGTTTACAAAAACAAC
BMD_4606	*cymR*	BMD_0868		CAGATTTAATTCTTAGTTAATTTATGTATTTAATTAGT
BMD_4606	*cymR*	BMD_4826	*ytkP*	CATCTAAAAACCCTAAAAATTGTAGCGTTATATGATAA
BMD_4606	*cymR*	BMD_4703	*ssuB*	AATATATAAAACACAGTTGACAAATCGGATTAATATGA

BMD_4606	*cymR*	BMD_4583		ATAGCTAAAAACATACTATATTCATAAGAATTATTAGT
BMD_4606	*cymR*	BMD_4581	*mccA*	ATGGAATTAAGGCTGAAATAAATATTCAATTCAATATA
BMD_4606	*cymR*	BMD_0092	*cysK*	AAAAATATAACAAAATTAGGTAAATGGAAAGACAAAAT
BMD_4606	*cymR*	BMD_2954		TATATAATTAGGCTATCCCTGTGAGACAGTATATTTTT
BMD_4606	*cymR*	BMD_3330		TTAACTAAAAACAAACTTGACTTATAGGAATAGTGAGT
BMD_4640	*nadR*	BMD_4790	*iscS*	AATACGGTAACCGCTATAGAAGTACTCATCACTCAATTGCACAGTCTT TAGAAGAAGCAGAAGAAGCATTTCGTCATTATGAATACAAAGAAGC ATTAAAACAGGCAT
BMD_4640	*nadR*	BMD_4639	*nifS*	TAACATTTTACACCTCTTTATTTACATGTGTCTTGACACCTATATTTAC ATATTATTAAATGAAAGACAAGAAGTTTTTTTATCGGAGGGAGAAAA TGATTTATCTGG
BMD_4640	*nadR*	BMD_4638	*nadB*	GGTCTATTTAGTAAAAGAGGGAGGCTATTTTTTTGAAGAACAGAAAG TAAATTATTATACATTTATATCCACAGTTCTGTGTACATTTATTTCTCC ACATTTTACAAT
BMD_4719	*fadR*	BMD_2434		AATAATACAAAATCAGTAATTTTATAAGTATTTGCTTTA
BMD_4719	*fadR*	BMD_1303		AATATTTTTAGAAGTAATTGTATGACGAACAATTTGCAG
BMD_4719	*fadR*	BMD_0612	*lcfA*	ATTATTATGACTTAATAATGAATGGGTGTTCATTCATTT
BMD_4719	*fadR*	BMD_4992	*fadN*	CTTTTTTTACGCATAAAATGAATGACCATTCATTCAATA
BMD_4719	*fadR*	BMD_4910		ATGAAAATCTTTTGCGAAAGTACATATGAAATTGTAATA
BMD_4719	*fadR*	BMD_4720	*lcfA*	TCCTTTCTCTTGTAAGTTATTGCTAACGTAGATTTTTTT
BMD_4719	*fadR*	BMD_5172	*fadF*	ATAACTGACTTACGAGTAAGTAATATATGTTATTTTTGT
BMD_4752	*phoP*	BMD_5245	*walR*	TCATTTTTAATATTTTTAATACA
BMD_4752	*phoP*	BMD_4752	*phoP*	AAAATTTTTAAACTGCCTTAAAT
BMD_4752	*phoP*	BMD_4610	*pstS*	ATTTATTTAACGCTTTATTTACA
BMD_4752	*phoP*	BMD_5082	*phoD*	CAATCCTTTATATTCCTTTTACA
BMD_4752	*phoP*	BMD_3995		AACTTTTCAACACCAGCTTAATT
BMD_4752	*phoP*	BMD_2688	*yfkN*	ACGATATGATAAAAACTTGTCAA
BMD_4752	*phoP*	BMD_4354	*resD*	AGCTTCTTTATTTTATTTATTCA
BMD_4752	*phoP*	BMD_1205		CACTCTTTTACAGTGGGTAAACA
BMD_4804	*ccpA*	BMD_1683		TGAATAGGATTTCT
BMD_4804	*ccpA*	BMD_1131		AAAAAACGCTGTAT
BMD_4804	*ccpA*	BMD_4779	*ackA*	CGGTAGCGTTTTCA
BMD_4804	*ccpA*	BMD_4686	*ilvB*	TGAAAACGCATTCA
BMD_4804	*ccpA*	BMD_2697		CGAAACCTTTTTCT
BMD_4804	*ccpA*	BMD_2296		ATTTTCGACATATT
BMD_4804	*ccpA*	BMD_2294	*citM*	ACTTTGGCAATAGT
BMD_4804	*ccpA*	BMD_1777		ACTTTCGCTATAGT
BMD_4804	*ccpA*	BMD_1658		ACTTTCGCAAAGGA
BMD_4804	*ccpA*	BMD_1653		TGAATATGATTAGA
BMD_4804	*ccpA*	BMD_1605		AATTTCGCCTCCGT
BMD_4804	*ccpA*	BMD_2015		AGTAAATGCTTAAT
BMD_4804	*ccpA*	BMD_1564	*fruR*	ACTTTCGTGAAAGT
BMD_4804	*ccpA*	BMD_1546		CATGTAGCAAATGT
BMD_4804	*ccpA*	BMD_1428	*msmX*	TTTCTCCGCTTACT
BMD_4804	*ccpA*	BMD_0979		ACCCTCGCAAATTT
BMD_4804	*ccpA*	BMD_0957	*scrA*	ACTTTTGCCTTTGT
BMD_4804	*ccpA*	BMD_0953		TGATAACGTTTACA
BMD_4804	*ccpA*	BMD_0901	*rpoN*	AAGCAACGCTTGAA
BMD_4804	*ccpA*	BMD_0863		AATTTCGCAAGAGA
BMD_4804	*ccpA*	BMD_0852	*citM*	ACTTTCGCCTAAGT
BMD_4804	*ccpA*	BMD_0833		ACATTCGCAATTGT
BMD_4804	*ccpA*	BMD_0666	*rocR*	TCCTTCGCGATAAT
BMD_4804	*ccpA*	BMD_0612	*lcfA*	TGAAAAAGCTTCTA
BMD_4804	*ccpA*	BMD_1075		AATTACGTAAAAAT
BMD_4804	*ccpA*	BMD_1062		AATGTCGCAAAAGT
BMD_4804	*ccpA*	BMD_1020		TGTATACGCTTTGT
BMD_4804	*ccpA*	BMD_0563	*msmX*	TCTTTTGCCAAAGT
BMD_4804	*ccpA*	BMD_4910		AGAAAACGCTTTCA
BMD_4804	*ccpA*	BMD_0533	*glpF*	TGACAACGCTTTCA
BMD_4804	*ccpA*	BMD_0514		GGATAATGCTTTCT

BMD_4804	*ccpA*	BMD_0493	*mmsA*	TGAAAACGTTTGGC
BMD_4804	*ccpA*	BMD_0462		ACTTTCGGTATAGG
BMD_4804	*ccpA*	BMD_0453		ACTTTCGCAAAATA
BMD_4804	*ccpA*	BMD_5217	*cydA*	TAAAAACGTTCAAA
BMD_4804	*ccpA*	BMD_4799	*acuA*	TGAAAACGCTTTAA
BMD_4804	*ccpA*	BMD_4798	*acsA*	AATTTCGCAAAAGT
BMD_4804	*ccpA*	BMD_4756	*citZ*	ACATTCGTAAAAGA
BMD_4804	*ccpA*	BMD_4720	*lcfA*	ACTTTCTCACTAAT
BMD_4804	*ccpA*	BMD_5188	*pta*	GATAACTGCTTCCA
BMD_4804	*ccpA*	BMD_5093	*gbsA*	AAACTCACAAAAGT
BMD_4804	*ccpA*	BMD_5070		TCGTTCGCAAAACA
BMD_4804	*ccpA*	BMD_5041	*sigL*	ACTTTCGCAAAAAT
BMD_4804	*ccpA*	BMD_4430	*bkdR*	ACTTACGCAAATGT
BMD_4804	*ccpA*	BMD_4404	*mmsA*	ACCTTTGTAATTTT
BMD_4804	*ccpA*	BMD_3922		TAATAATGTTTTAA
BMD_4804	*ccpA*	BMD_4393		TGTAAACGCTTACT
BMD_4804	*ccpA*	BMD_3813	*putC*	TGTAAGCCCTTACA
BMD_4804	*ccpA*	BMD_3792		AGAAAACGCTTTCA
BMD_4804	*ccpA*	BMD_3721		TCTTTCGCAACTGT
BMD_4804	*ccpA*	BMD_3683		TGTAAACCCTTACA
BMD_4804	*ccpA*	BMD_4061		TGATTACGCTTAAA
BMD_4804	*ccpA*	BMD_4002		ACCTTCGCATAATT
BMD_4804	*ccpA*	BMD_3587	*gph*	TGAAAACGCATTCT
BMD_4804	*ccpA*	BMD_3465	*iolB*	ACCTTCTCGAACTT
BMD_4804	*ccpA*	BMD_3463	*mmsA*	ACGTTCCCAAATGT
BMD_4804	*ccpA*	BMD_3407		ACTTGCGTGAAAGC
BMD_4804	*ccpA*	BMD_3311	*gltP*	ACTTTTTCAAAAAT
BMD_4804	*ccpA*	BMD_2639		ACTTTCCCCTTTGC
BMD_4804	*ccpA*	BMD_2610		ACTTTATCGTTTTT
BMD_4804	*ccpA*	BMD_3077	*citA*	ACATTCGCAATTGT
BMD_4804	*ccpA*	BMD_2437		TCCTCAGCAAAAAT
BMD_4804	*ccpA*	BMD_2434		TTTAAACTTTTACT
BMD_4804	*ccpA*	BMD_1994		TGATAGCGTTTTCA
BMD_4804	*ccpA*	BMD_2367	*yurJ*	GGTAAACGATTCCA
BMD_4804	*ccpA*	BMD_2336		GCCTTCGTCAAAGT
BMD_4804	*ccpA*	BMD_1876	*acoR*	TGATAACGTTTTCT
BMD_4804	*ccpA*	BMD_1872	*acoA*	TAAAGCCGTTTTCA
BMD_4804	*ccpA*	BMD_1860	*xylT*	TGAAAACGCATTCA
BMD_4804	*ccpA*	BMD_1858	*xylA*	ATTTTAGAAAAAGT
BMD_4804	*ccpA*	BMD_1844	*gapN*	TGATAATAATTTCA
BMD_5039	*cggR*	BMD_5039	*cggR*	GCAAGGTGGGACATAATATGTCTATACGGGACAAAATGTGTCCAGC
BMD_5108	*degU*	BMD_4002		AAAAAATACTGTATGTTTTT
BMD_5108	*degU*	BMD_0957	*scrA*	AAAAACAACTGCTTGAACAT
BMD_5108	*degU*	BMD_3721		CAAAAAAATCAAAAAAACCT
BMD_5245	*walR*	BMD_2238		CCGATAACTTCTCCTTTACTTTCTTTA
BMD_5245	*walR*	BMD_0478	*yocH*	TATAAGGACATTACCTTCACTTTACTT
BMD_5245	*walR*	BMD_2994		AAAAACAATATTTTTTTGACTTATATG

Table 1: Binding sequence predictions for *Bacillus pumilus* SAFR-032. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BPUM_0001	*dnaA*	BPUM_0001	*dnaA*	TACAATAGGTGTTTAAC
BPUM_0031	*purR*	BPUM_3693	*purA*	ATAAATTTTGCTTGTTTTAAACTAAAGCTCTATAATTACAAGCCGAAAGTG
BPUM_0031	*purR*	BPUM_3335	*glyA*	AGAACGAGCGGTTTATACAAGATCATAGCTTAGATATATTCTAATAAAATA
BPUM_0031	*purR*	BPUM_2646	*ytiP*	GGGTATTTTGCTTATAATATAATTAACAAAACATTATTACAAGCATTTACC
BPUM_0031	*purR*	BPUM_1944	*xpt*	AAGATTTTTGCTTGCAAAACTAGAATAAATAATAAATTCAAGCCATAAAAA
BPUM_0031	*purR*	BPUM_2164	*nusB*	GCCATTTCCGAATCATAATGACCTCATACTTGGAATGATTCGTTTTTCGTG
BPUM_0031	*purR*	BPUM_1446	*pyrR*	CCGAAATAGCTAAGTAATTCTTTATTGCTAACTGTCCGGCAAGAAAAAAAC
BPUM_0031	*purR*	BPUM_0596	*purE*	TCTAAATCCGAACATTAAAACGACGACTTCATAGATTGTTCGATAAATCCG
BPUM_0031	*purR*	BPUM_0592	*pbuG*	CATTATTCTGCTTACTTCGCCTGCAAAGAAACTTTTTACAAGCGAAAAGCA
BPUM_0068	*ctsR*	BPUM_0068	*ctsR*	GTCAAATATAGTCAAA
BPUM_0068	*ctsR*	BPUM_3102	*clpP*	AAACTGGAAATAACTG
BPUM_0083	*sigH*	BPUM_1421	*ftsA*	AAAAGGATATACATATGATGTAACGAATATTTTCAGT
BPUM_0083	*sigH*	BPUM_0897	*lytE*	AAATGAAATATAAAGAACCCTAAAAAAAAATAAATTT
BPUM_0083	*sigH*	BPUM_1290	*kinA*	GAAGGAAAATACGGGTTTTCTAGCGAATCTTTTTTAT
BPUM_0083	*sigH*	BPUM_2957	*fumC*	TAAAGGATTTTGGTTTTTTGTATAGAATAGAGTTGAG
BPUM_0083	*sigH*	BPUM_3358	*spo0F*	TAAAGGAAATGTGAAAATCAAACAGAATTATTTAACA
BPUM_0083	*sigH*	BPUM_2864	*spo0M*	TTAGGACTAAAGTATGAATGAAATGAATCTTTCTTTC
BPUM_0083	*sigH*	BPUM_3179	*raiA*	GCAGGAGAAAAGGAGGGAAAAGTAGAAATAGGTTTAC
BPUM_0083	*sigH*	BPUM_2254	*dnaG*	TGAAGGATTTTTGAATTAAAAGGCGAATAATGTACGA
BPUM_0083	*sigH*	BPUM_2154	*spo0A*	TAAAGGGGATAGCATGGTTTTGTCGAATTGAACACTC
BPUM_0083	*sigH*	BPUM_2078	*spoIIAA*	GAAGGATTTACTTAATAAAAAAGAGAAACAATCTTTA
BPUM_0160	*sigW*	BPUM_3495	*ywaC*	TTTTTAAAACCTTTCGAGTCTCGTCTCCGTCTAGT
BPUM_0160	*sigW*	BPUM_2585	*sppA*	TCCTGTGAAACATTTCATGCAAATATTCGTATCAA
BPUM_0160	*sigW*	BPUM_2272	*yqeZ*	AAAATGAAACCTATGATACATTTGTAACGTATGTA
BPUM_0160	*sigW*	BPUM_2113	*yqjL*	TAAGAGAAACATCTCTGCCTTGTGTGCCGTCTTTT
BPUM_0160	*sigW*	BPUM_1076	*yjbC*	CTAGAGGAGACTTTTCTGCGTCTTACTAGTACAGG
BPUM_0160	*sigW*	BPUM_0260	*yceC*	TGTTTGAAACCTATCTTGAACGTAGATCGTATATA
BPUM_0160	*sigW*	BPUM_0160	*sigW*	GAGAATGAAACCTTTTGGAAAGAGGTTCGTATACA
BPUM_0208		BPUM_1742		AGATATTTTATTGTCAGGCAGGAAACCAATCAA
BPUM_0208		BPUM_0209		AGGATTGTGACTGTTCGGCAGGCAAAACCTAAA
BPUM_0208		BPUM_3644		GACATGTGACTGGTCATGCAGGCAGGATCTAAA
BPUM_0208		BPUM_3566		GGATTGTTACTGGTCAAGCAGGCAAAACCTAAA
BPUM_0245	*ycbA*	BPUM_1374	*gls*	GAGTCTTTCCTTCTCGTACTTTTTCTCTGTTTTTAACGTAGGTG
BPUM_0245	*ycbA*	BPUM_0244	*glsA1*	GTAAGTTTTTCTGTAGATTTTCGTCAGATTGTGTAGTTTCCAAG
BPUM_0446	*sigB*	BPUM_3271	*alsS*	TATAACCTTTAAAAGGTAAAACTTTACATATCATTTAG
BPUM_0446	*sigB*	BPUM_3252	*ywtG*	TGTTTGAAGTGACAGAAAGTGAGGAAAATGATTTTTCA
BPUM_0446	*sigB*	BPUM_3219	*gtaB*	AATATAAATGAAAAAAGATAATTTTAATGAAATTTGTA
BPUM_0446	*sigB*	BPUM_2735	*gbsA*	TAACTTCGTATAGAGGGAAACTGTCTTTTTAAACTTGT
BPUM_0446	*sigB*	BPUM_2703	*dpsA*	GGTTTCAGTATAATAGGAAAAGGGTATTCTCATACTAA
BPUM_0446	*sigB*	BPUM_3179	*raiA*	CGTTTGAGCAGGAGAAAAGGAGGGAAAAGTAGAAATAG
BPUM_0446	*sigB*	BPUM_3158		TGATTAATTCATCTATTAAAGGGAGGAACGATGATTGA
BPUM_0446	*sigB*	BPUM_3102	*clpP*	GGTTTAAGAAAATAAATTTTGGGAAAAATAACATGTGG
BPUM_0446	*sigB*	BPUM_2573	*ytkL*	AGTTTTTCAGTTTTTTATACAGGGAATAGATGAAATGA
BPUM_0446	*sigB*	BPUM_2553	*phoP*	TTAAAATTTAACAAGAGAACAATTCGAAAGCAATTTTA
BPUM_0446	*sigB*	BPUM_2488		AGTAAGTTGGGTAAAGAAATTAACATAATAAATTTAGG

BPUM_0446	*sigB*	BPUM_2472	*ilvB*	TTTTTATATGGTCTAAAATCTCGATCTTGAATAGAAAA
BPUM_0446	*sigB*	BPUM_2401	*relA*	AGAGAAATGTAGATAGCTTAAAAAGAACTATTATTTGA
BPUM_0446	*sigB*	BPUM_2351		GAAGACAATTCTAGTGGGAAAAAGGTTCGGAAAGTTTA
BPUM_0446	*sigB*	BPUM_2346	*yjqC*	GTATTAAACTTAATGAGAAAAGGATATTTCTCTGCAGA
BPUM_0446	*sigB*	BPUM_1857	*dhaS*	TATGTAAATAACATCGGTTAAAACGTACTTTACGTTAC
BPUM_0446	*sigB*	BPUM_2263	*cdd*	GGTACCATTTCTTCTGAAAGTGGGAACTAGGAATGAAC
BPUM_0446	*sigB*	BPUM_2255	*yqxD*	GTTTTTCCTATGGTGGTAAACGGGTAAACAAAAACAAT
BPUM_0446	*sigB*	BPUM_2207	*yqgZ*	TCTTTCGGATAAATTCAAGTAGGATAAAAATACAAAAA
BPUM_0446	*sigB*	BPUM_2206	*yqhA*	AGTTTTAAGTGAAAAAGGAGCGGAAGAAGTAAATGAAG
BPUM_0446	*sigB*	BPUM_1776		TCGTTTTAGATTTTTCACAGTGGGTACAGTTTTATATC
BPUM_0446	*sigB*	BPUM_1749	*mmsA*	TTTTTTTATAAAAAGGAGAGAGGGAACATGACCAAAAC
BPUM_0446	*sigB*	BPUM_2181	*yqhQ*	GAGTGTAAAATTTTTATAAGAGGGTATACTGGCTAAAA
BPUM_0446	*sigB*	BPUM_2140	*bmrU*	CGTTTATACGCCGCGAAATTGGGGAAAAGTGATGATAA
BPUM_0446	*sigB*	BPUM_0976	*yhxC*	TGTATAATTTTTCCACAATGAAGAAAATCATACATTCG
BPUM_0446	*sigB*	BPUM_0892	*katX2*	ATGTTTGATTCAATATAATTAGGGTATATAGATTGCTA
BPUM_0446	*sigB*	BPUM_0800	*csbB*	GGTTTCGTTTCCTGACAAAAAAGGAATAATGAACATAT
BPUM_0446	*sigB*	BPUM_0735	*yflA*	AGGAGGATAGAAAAGGGATTCTCAAGAAAAGCTTTTGG
BPUM_0446	*sigB*	BPUM_1193		GTTTCAACCATGACTAAAGCGAGGAATAATAGAACTAT
BPUM_0446	*sigB*	BPUM_0643	*opuE*	AGTATAGTTTAAAATGACAAGATTTCGTTGAAATGTGT
BPUM_0446	*sigB*	BPUM_1076	*yjbC*	GCGTTTAAATAGCAGGAAATCGGGAATATCTAAGGAAG
BPUM_0446	*sigB*	BPUM_0391	*ydaG*	TGTTTAATGAATGGCTTGTCGTGGAAACCTAAGAGTAC
BPUM_0446	*sigB*	BPUM_0390	*ydaD*	AATAACTAGTACATTGGCTAAAATAGGTAGAACTTTGT
BPUM_0446	*sigB*	BPUM_0289	*nadE*	TGTTTGATCATGAATGAAATAGGGGAACGTTTAACAAA
BPUM_0446	*sigB*	BPUM_0249	*yccK*	AAGCAGTAAAGTAAGGGCGATAGACTGATACTCTTTTC
BPUM_0446	*sigB*	BPUM_0074	*yacL*	CGGTTAAAAATGGATGAATATGGGTATATTAATGATGC
BPUM_0446	*sigB*	BPUM_0068	*ctsR*	GGTTTGACACCATATAAATGTGGAAATAATATTCTTAG
BPUM_0446	*sigB*	BPUM_0036	*ctc*	GGTTTAAAATGATTCGGTTATGGGTAATTATGAGAGTA
BPUM_0446	*sigB*	BPUM_3715		AAATTAAATTATAGAGCTTAAGCTCTATTATATTTCTG
BPUM_0446	*sigB*	BPUM_3712	*katX1*	GGTTCATCGATTGGAACAATTGGTTGTTTAAAGAGATG
BPUM_0446	*sigB*	BPUM_3689	*yycF*	CGTTTTTCTTATGAAAAGAAGGCATAAAAAGGGAGAAG
BPUM_0455	*acoR*	BPUM_0451	*acoA*	TGAGAAGAAGTAGACATATTGAGACAAATAGAGATTCAAGTCTCA TTTTGTCTCTTTTTTGATC
BPUM_0528	*rex*	BPUM_0277	*ldh*	TTTTCAAACACTTCATAACGTGTTATTCTTACA
BPUM_0528	*rex*	BPUM_3519	*cydA*	TTTTAAAACACTTCACCACTCGTATTTACAAAG
BPUM_0528	*rex*	BPUM_3271	*alsS*	TGAAATGTATAGTAAATCATATCATGAAAATAT
BPUM_0528	*rex*	BPUM_2472	*ilvB*	CGAGTGACATTGCTCACTCGTTACCAAAATGGA
BPUM_0708	*citT*	BPUM_3618		ACATAAAATATAAATTTAATCTTACAAATATAAAATAGATTTAAAA AACTG
BPUM_0708	*citT*	BPUM_2496		TAAGAAGCCAGCAAGGCTGATTATAAAAATAATTTATAGAATATAC GAAAA
BPUM_0708	*citT*	BPUM_0710	*citM*	AACAAAATGAACAAAAAGCATTTATTGCCCGAAAAAGAATTAAAA TTCATA
BPUM_0730	*treR*	BPUM_0728	*treP*	TGTTGACTACATGTATATACAAATATAAAATA
BPUM_0730	*treR*	BPUM_3452	*sacP*	AACAACTTTTTACTTATACCTGAAATAGAGGG
BPUM_0827	*perR*	BPUM_0636		GATAGTTATACTTGTTTACCA
BPUM_0827	*perR*	BPUM_0629		AGAGTTAATAAATGTTTATAT
BPUM_0827	*perR*	BPUM_0319		TTTTTACGGTAAATATTTTCA
BPUM_0827	*perR*	BPUM_2458	*hemA*	AACATTAATAATATTTAATCT
BPUM_0827	*perR*	BPUM_2084	*fur*	AATATTAGCAATAATTTATTA
BPUM_0827	*perR*	BPUM_0892	*katX2*	ATTAATTTAAAATTATTATAA
BPUM_0827	*perR*	BPUM_0827	*perR*	AATATTTCTAATATTATATTC
BPUM_0827	*perR*	BPUM_1275	*ykvW*	AATATTATTAATAATTAACTT
BPUM_0827	*perR*	BPUM_1077	*spxA*	ACTTATTTAGAACAATTTTAA
BPUM_0827	*perR*	BPUM_3712	*katX1*	TCTATATAAGACTGATTAGAC
BPUM_0827	*perR*	BPUM_3021		ATTTTTTTCAAAATATAATAC
BPUM_0827	*perR*	BPUM_3010		GACATGAATAAGAATATTCTC
BPUM_0881	*glpP*	BPUM_0882	*glpF*	GATGGAGAAACGGAGATCCA
BPUM_0881	*glpP*	BPUM_0182	*glpT*	GCTGATGAAATGAAAAAACA
BPUM_0902	*sigM*	BPUM_0902	*sigM*	GCACGTTGAAATACGGAAAGAATACGCACATATTGTATCT
BPUM_0902	*sigM*	BPUM_1077	*spxA*	TCCATACGTATAAAGACGAAGAATGGTATGGGCAGTCGTA

BPUM_0944	*hpr*	BPUM_0343		ACAAAATCTTATCAAAATAT
BPUM_0944	*hpr*	BPUM_0972	*aprE1*	GATAAAATAATATTTTTTTA
BPUM_0944	*hpr*	BPUM_0233	*epr*	AAATATACTTAAAGTTAAAG
BPUM_0978	*comK*	BPUM_3475	*rapE*	TTTTAAAGAAAAACCCA
BPUM_0978	*comK*	BPUM_2308	*rapA4*	TTTTAGTAGAAAAAGCT
BPUM_0978	*comK*	BPUM_2205	*comGA*	TTTTGGTGCAAAAGGTA
BPUM_0978	*comK*	BPUM_1651		TTTTACTATTATATACA
BPUM_0978	*comK*	BPUM_1615		GTAGAAAAATCGTTTTG
BPUM_0978	*comK*	BPUM_2096	*rapA3*	TTTTACTAGAAAACGTT
BPUM_0978	*comK*	BPUM_1598	*recA*	TTTTCTCACAAAAAAGA
BPUM_0978	*comK*	BPUM_1509	*smf*	TGACAAAACCTGTTGTT
BPUM_0978	*comK*	BPUM_0993	*addB*	TTTTATCGATAAAGGGA
BPUM_0978	*comK*	BPUM_0978	*comK*	GATAAAAAAATGATTTT
BPUM_0978	*comK*	BPUM_1209	*rapA2*	TATGAAACAGAGTTTTT
BPUM_0978	*comK*	BPUM_1141	*rapA1*	CTTCAGTAAAATAGAAG
BPUM_0978	*comK*	BPUM_1132	*rapH*	TTTTACTCCAAATGACT
BPUM_0978	*comK*	BPUM_3733	*trmE*	TTTTACGGTAAAAATGT
BPUM_0978	*comK*	BPUM_3721	*yyaF*	TTTAAACAGAAAACATA
BPUM_0978	*comK*	BPUM_1321	*rok*	TTTTAATCAAAAGATTA
BPUM_1227		BPUM_1814	*gltA*	TAGATTTAATATATA
BPUM_1227		BPUM_1227		TAGTGTTAAACACTA
BPUM_1237	*sigI*	BPUM_1237	*sigI*	GGGGAGATACCCCCTTATTTCTTTGCTTTTAGACGAAACATTGTAAAGTAGATG
BPUM_1310	*ccpC*	BPUM_2556	*citZ*	TGTATTAATTTATAT
BPUM_1310	*ccpC*	BPUM_1848	*mmgD*	CATATCGTCTTATCA
BPUM_1310	*ccpC*	BPUM_1699	*citB*	GTTATTTACTTATGT
BPUM_1426	*sigE*	BPUM_0284	*ycgF*	TTGTACTAAACCACTAAGCGTTTGAATAATATGTACAAGT
BPUM_1426	*sigE*	BPUM_0256	*cwlJ*	GCAGTCATCTACACAGAAACCCTACATAAAATAATAGAGA
BPUM_1426	*sigE*	BPUM_0148	*ybaN*	CTTTAATAATATAACATTTCGCTTCATCTTATGTCAACAA
BPUM_1426	*sigE*	BPUM_3725	*yyaD*	AATCATTGTCATATGCCTTTGTTCGAATTAAGATATGGAT
BPUM_1426	*sigE*	BPUM_3689	*yycF*	TTATTATTTTTATCTTTGCCACGTAAAACGTTATAATAGA
BPUM_1426	*sigE*	BPUM_3320	*spoIID*	CTGTCATATTAGCTTGTCCTTCCCCATAGGATAAAAGAGA
BPUM_1426	*sigE*	BPUM_2685	*asnB*	TAATTTTTTTTAATATGTTTTTCGCTTTTTTACACTTTGT
BPUM_1426	*sigE*	BPUM_2559	*ytvI*	CTTTTCATATTACTAATCTTTTTGAATAAAATGAAGTATC
BPUM_1426	*sigE*	BPUM_2553	*phoP*	TTGTTCTCTTGTTAAGCTTTCGTTAAAATGAAAGAGAAAG
BPUM_1426	*sigE*	BPUM_2540	*ytxC*	GGCGTCTATTTTTTATGTTCTCCCCATATATATGTAACAG
BPUM_1426	*sigE*	BPUM_2492	*gerM*	AACACTTCTCAAGAAGCGCCTGTTCATACATTTTATACAA
BPUM_1426	*sigE*	BPUM_2452	*spoVID*	CAATCATATTTTCACTGCCTCTCACATACATCTGTATTGA
BPUM_1426	*sigE*	BPUM_2438	*spoIVFA*	GTTGCATATCAGATGAGGTTGCCACATAAAATGTACAAAC
BPUM_1426	*sigE*	BPUM_2408	*spoVB*	AATAAGAGATTCATACTGGTATTAAGGAGATTTAGATGTA
BPUM_1426	*sigE*	BPUM_2286	*spoIIP*	CAGTTCTACTTTCTCTAGCTCGCTCATAGTGTAATTACTA
BPUM_1426	*sigE*	BPUM_1786	*yngJ*	TTGGCATGAAGCAGCCACTTGATTAATAAATATGATAGAA
BPUM_1426	*sigE*	BPUM_2175	*spoIIIAA*	CAAGACTACTTCCAGCTAGACAAGCATATAGTGTAAAAAA
BPUM_1426	*sigE*	BPUM_1633	*spoVK*	TTGTCACGATTACAGCTCCACCCGAATAGAATAAAGAGAG
BPUM_1426	*sigE*	BPUM_1607	*cotE*	AAGGCAAGTTTATGGGCGACCATGCATACACTGAAACAGA
BPUM_1426	*sigE*	BPUM_2085	*spoIIM*	ACTGTCATGTTTCATAATCTCTCTCATAGAATCTAGTAAC
BPUM_1426	*sigE*	BPUM_2052	*dacB*	GGCATCATAACTTTTTCAGCCGCGCATACACTTGTACAAA
BPUM_1426	*sigE*	BPUM_2011	*spoIVA*	GTGGACTACCGCAGTGATTTTGCTCATATTTTAAAGGATA
BPUM_1426	*sigE*	BPUM_0929	*yhaX*	ATGTTCTAAAGCACCAAGCTAGGGCATATCCTCATCGTAA
BPUM_1426	*sigE*	BPUM_1396	*ylbJ*	GGAGGAGAAAAGATACGTATATATGCTTAGATAACCAGAT
BPUM_1426	*sigE*	BPUM_1379	*ctaA*	AACGTAAATTCGTTACAACTCTATCATACAAGATAAACAG
BPUM_1426	*sigE*	BPUM_0896	*phoA*	AAATCAGGTAAGATACATTTCACCAGGAAATGTTAAAATT
BPUM_1426	*sigE*	BPUM_0895	*spoVR*	ACATCATGTTTTATTAGATGCGCTCATATAGTAGGAATAA
BPUM_1426	*sigE*	BPUM_0856	*prkA*	AAAGCATGTAATAGGACGCCTCCGCATAGATTGTAAAACA
BPUM_1426	*sigE*	BPUM_1273		TCAAAATATTTTCAGGACTTGTCTCATAGAATGTGAAAAG

BPUM_1426	*sigE*	BPUM_1262	*ykvI*	TCCTTCTTTTCAAGGAACTTGTCTCATAGGATAAAAAAAG
BPUM_1426	*sigE*	BPUM_0641	*phoB*	TATTCAAATAAAATACAAATTATGAAATAAATGATAAAAG
BPUM_1426	*sigE*	BPUM_1010	*asnO*	AAATCAACTATCAAGGAGACGATGCATAAACTAAGAAAGG
BPUM_1426	*sigE*	BPUM_0436	*ydcC*	TCTGCATATTGTTTGTCCCCCCTTCATATATTGGGTAGTG
BPUM_1426	*sigE*	BPUM_0434	*ydcA*	GTGATGGGTTATATACTTCCCCCCTGTTTGTTATACGTCT
BPUM_1427	*sigG*	BPUM_0868	*yhcQ*	GTAAATTAAGTACACCTTTACTAACTTTCTTTCGGA
BPUM_1427	*sigG*	BPUM_1283	*splB*	TTGTTTGAATAAGGGAGATTTTAGAAAGGATAAAAG
BPUM_1427	*sigG*	BPUM_0385	*acpD*	AAGTAGAAAAAAAGTACTCTGCAGAAATGTTTGAGA
BPUM_1427	*sigG*	BPUM_0206	*ybeC*	AAACATAAAATTAGCTTTCCACAATACTTGTACTTA
BPUM_1427	*sigG*	BPUM_0146	*gerD*	TATGTATAAATTTACGTGAAACATTCATATTAAAAG
BPUM_1427	*sigG*	BPUM_2959	*gerAA*	TTTTATCTTTATTTTAATTTTAGTTCAATATAAACG
BPUM_1427	*sigG*	BPUM_3388	*ywhE*	TTGTTTAAAACGGTTTTCTTTGTTCCATACTAATAA
BPUM_1427	*sigG*	BPUM_3260	*gerKA*	GACGAATAAAATCGGACTCTCCGCTCACACTATGGT
BPUM_1427	*sigG*	BPUM_3248	*gerBA*	TTGCATAATTCTCCACCATCCGAGTCACTCTATAAA
BPUM_1427	*sigG*	BPUM_2722	*dksA*	ATGAATAAAAACAAATATAGAAGTGTCTATAACAGC
BPUM_1427	*sigG*	BPUM_3166		TTACATGAAATCCCTTCTTCTGGCATATAATAACTC
BPUM_1427	*sigG*	BPUM_1964	*ponA*	GCAAATAATAAACGGGCGAAGTCTAGAGAAAAAGAA
BPUM_1427	*sigG*	BPUM_1880	*ctpA*	CCGAATAGAAAAGTTTCTCTTGAAGCTAGGAAACCA
BPUM_1427	*sigG*	BPUM_2287	*gpr*	TCGAATGTTTCTTTCAAGAGATGGAAAGACTACTGT
BPUM_1427	*sigG*	BPUM_1794	*gerA*	AACGATGTCAATGCAAATTTTAGATCATAAAATGAT
BPUM_1427	*sigG*	BPUM_2155	*spoIVB*	GGTTATAAATCAAAGCGAAGAAGGCAAAAGTAAAGA
BPUM_1427	*sigG*	BPUM_2079	*dacF*	CTGAATAAAACAGCCTGGATGAGGAAAAAATAGAAA
BPUM_1427	*sigG*	BPUM_2075	*spoVAA*	TCGGTATGAGAACCATTTTTCACCACATACTATTTT
BPUM_1427	*sigG*	BPUM_2026	*sleB*	AATGTATAAAAATAGGGATTGGAAGAAATATATATT
BPUM_1427	*sigG*	BPUM_0957	*pbpF*	AACTATCATATAGTTTTCTTACTATATATTAAGAAA
BPUM_1446	*pyrR*	BPUM_1446	*pyrR*	TCAAAGATTCTTTAAAACAGTCCAGAGAGGCTGAGAAGGATAACG GATCAATTT
BPUM_1487	*ylpC*	BPUM_1487	*ylpC*	ATTTATCATCTATGAGATTATATACTACTATTAGTACCTAGTCATAA AT
BPUM_1487	*ylpC*	BPUM_0903	*plsC*	AAAATCATGGTCCATTATTTTTAAACAATATTATCTTTGTTTAGGAA GT
BPUM_1487	*ylpC*	BPUM_1101	*fabI*	ATAATACTAGTCCATGATTATGATACAGAACTTTTTTATCCTCCGAA AT
BPUM_1487	*ylpC*	BPUM_1057	*fabHA*	TATTGCCAAGGAGAAAAAAATAAGGTAAAATTAGTACCAGATACT AATA
BPUM_1515	*codY*	BPUM_2472	*ilvB*	AAAGATGTCCATAAGACGGCAGCTG
BPUM_1515	*codY*	BPUM_0978	*comK*	ACAAGAATTTATAATTTTTAGACAA
BPUM_1515	*codY*	BPUM_1184	*dppA*	ACAAAATAAACAATTTTTTTAGGAA
BPUM_1515	*codY*	BPUM_1152	*hag5*	TTAGATATCGTTGATAATGCAATCA
BPUM_1515	*codY*	BPUM_1151	*hag4*	ACTCAAGTGTTTCTTGTTGTAAGAA
BPUM_1515	*codY*	BPUM_1150	*hag3*	AAATCTATTTAAAACTAATTGGTGA
BPUM_1515	*codY*	BPUM_1149	*hag2*	AAAAATTATTAAAATTAGGATAAAC
BPUM_1515	*codY*	BPUM_0636		CGACGTCGTTCACTAGCCATAAAAA
BPUM_1515	*codY*	BPUM_0629		ACATTTTATCTCAATTATTTACAAA
BPUM_1515	*codY*	BPUM_0319		AATATAGTTCTTAAGATTTAAGAAA
BPUM_1515	*codY*	BPUM_0150	*hag1*	ACAGGAAAAACAAAATGAATTATAA
BPUM_1515	*codY*	BPUM_3271	*alsS*	TTTTATATAAAAGTATTTTTTAAAA
BPUM_1546	*sigD*	BPUM_3698	*ybdO*	AACTAACAAATTTTAACTTAATTGCCGATATATTTAATAA
BPUM_1546	*sigD*	BPUM_3233	*lytD*	TGTTAAAATAATTAGGTGATTCTGTCGATAAAGAGATTAG
BPUM_1546	*sigD*	BPUM_2790		AACATGAAAATAATATGAACCTGTCCGATATATAACATAT
BPUM_1546	*sigD*	BPUM_2788		GTCTTGAAAAATAATTCATATTACCTAAGAATAGAAGGGA
BPUM_1546	*sigD*	BPUM_2735	*gbsA*	TTAAATTTTTATTTAACAAACTTACCAAAAAGAATCAGAG
BPUM_1546	*sigD*	BPUM_1857	*dhaS*	ATGATTTAAAAACTCCTTTTTTCATCTATACATTTATTGT
BPUM_1546	*sigD*	BPUM_2253	*sigA*	TAAAATAAGGATAAACGAGTAGACAAACTATTTCTCTCTT
BPUM_1546	*sigD*	BPUM_1776		ATGTCAAAATATAGTCGACTTTTGTCTTATAACTTACTTA
BPUM_1546	*sigD*	BPUM_1749	*mmsA*	ATATTATGATTTTTCTAAAGTTAACCAAAAGTTAATCAAA
BPUM_1546	*sigD*	BPUM_0974		TCCCTTTAAAAAATCCAATATTATCCGATAAATAATATAA
BPUM_1546	*sigD*	BPUM_0972	*aprE1*	ATGATAAAATAATATTTTTTTATATCGAAATTCGAAATAG
BPUM_1546	*sigD*	BPUM_0897	*lytE*	AACCCTAAAAAAAAATAAATTTTGCCGAATTAATAAGCAT
BPUM_1546	*sigD*	BPUM_1296	*cheV*	AACGTCAACTTTTTACCTATGCTGCCGATATAAAAAGTAC

BPUM_1546	*sigD*	BPUM_1284	*mcpC*	CTATTGAATGCGGTATCAATTATGCCGATATATACAATAA
BPUM_1546	*sigD*	BPUM_1260	*motA*	AGACTAAAACCCTTCTAGAAGACACCGATATTAACTATAG
BPUM_1546	*sigD*	BPUM_1152	*hag5*	TTGTTAAACAAATTCCATTAGAGTTCGATAAATGGATTGT
BPUM_1546	*sigD*	BPUM_1151	*hag4*	ATGTTAAACTTCCTATAAAATAAACCGATAATACCCTTGT
BPUM_1546	*sigD*	BPUM_1150	*hag3*	GTGTTAAACTTCCTATGAAATAATCCGATAAAAGAACTGT
BPUM_1546	*sigD*	BPUM_1149	*hag2*	CTGCTAAACATCCTATAACCACGTCCGATAATAAAATTGT
BPUM_1546	*sigD*	BPUM_0636		TGCATAGAGAAAAGCTTAATCTACTGAACTTTTATATTCA
BPUM_1546	*sigD*	BPUM_0629		ATACTAAATTATACAAGCATCTTTACAATTGATATACTTT
BPUM_1546	*sigD*	BPUM_0319		ATAATCAATAAAAGCCATTTATAGCATTTAATAATACACT
BPUM_1546	*sigD*	BPUM_0311	*tlpC*	GGTATTTCAAACATTTTTTTCATGCCGATAATCCTTAAGG
BPUM_1546	*sigD*	BPUM_0233	*epr*	CTTCTAACCATCCTTCCCACTTCTCCGATATAATAGAGAA
BPUM_1546	*sigD*	BPUM_0150	*hag1*	CAAGTAATTTAAAGCAGTAGGGATATACGAACAATTTACA
BPUM_1636	*glnR*	BPUM_1811	*nasD*	ACATTAAAATGAATGAA
BPUM_1686	*lexA*	BPUM_2566	*dnaE*	AAGAAAGAACATTTGTTTCTCA
BPUM_1686	*lexA*	BPUM_2506	*uvrC*	ATATTTTGTTTACAAGCACTAT
BPUM_1686	*lexA*	BPUM_2416	*ruvA*	CTCGCTTGCATACAATTGAAAA
BPUM_1686	*lexA*	BPUM_1709	*parE*	CTATCTTGTAAACAAGACAAAA
BPUM_1686	*lexA*	BPUM_2102	*polY2*	AGAACAGAACATACGTTCTAAG
BPUM_1686	*lexA*	BPUM_1686	*lexA*	AAATACGAACAAACGTTTCTGT
BPUM_1686	*lexA*	BPUM_1616	*aprX*	TCCCGTAGTGTACAAACCTAAA
BPUM_1686	*lexA*	BPUM_1598	*recA*	TTAGCTTATACACAAGCAAAAA
BPUM_1686	*lexA*	BPUM_0990	*yhjE*	TTGATAGAACACACGTTCTTAT
BPUM_1686	*lexA*	BPUM_0934	*yhaO*	GTGATAGAACGTGCATTCGCAA
BPUM_1686	*lexA*	BPUM_0906	*yhdT*	GTTTCTTGCTAATTAACATTAG
BPUM_1686	*lexA*	BPUM_0904	*yhdP*	GATTACAATTAATCGTTCTTTG
BPUM_1686	*lexA*	BPUM_1157	*xkdA*	AAAATAGAACATACGTTTGTAT
BPUM_1686	*lexA*	BPUM_1114	*yjcD*	CTACGAAAAGAAAGGTTACAAA
BPUM_1686	*lexA*	BPUM_0625	*pcrA*	AAAATAGAACATAAGTTCGAAT
BPUM_1686	*lexA*	BPUM_3600	*yhaZ*	TATAGCGAACAAGCAGTCTGAT
BPUM_1686	*lexA*	BPUM_3455	*vpr*	TTATAAACAAAATAGTTCTTTA
BPUM_1686	*lexA*	BPUM_3148	*uvrB*	TCGGCTTGAAATCAAGCATAAA
BPUM_1815	*gltC*	BPUM_1814	*gltA*	ATCTCAAAAAGAGA
BPUM_1815	*gltC*	BPUM_1815	*gltC*	AGAGAAAAACTCTA
BPUM_1830	*xylR*	BPUM_1832	*xynP*	ATCTTAGTTTGTTTGGTCAATAAACTAAGATGAGT
BPUM_1830	*xylR*	BPUM_1829	*xylA*	AACTTATAATCAAACAACTTAGTTGAATGTTTCAC
BPUM_2042	*sigX*	BPUM_3287	*rapD1*	TAAAAAGTTGCAGAGAAGGCGTGTATACCATTTTT
BPUM_2042	*sigX*	BPUM_3217	*lytR*	TTGAAACCTTTTTTTTGTTACAACGTCTATATCTA
BPUM_2042	*sigX*	BPUM_3158		TTATAAACTATCCGTTGATTAATTCATCTATTAAA
BPUM_2042	*sigX*	BPUM_2113	*yqjL*	GAGAAACATCTCTGCCTTGTGTGCCGTCTTTTAGC
BPUM_2042	*sigX*	BPUM_2042	*sigX*	ATGTAACGTTTTGTAAGAGTCACTCGACAAAAATA
BPUM_2042	*sigX*	BPUM_0800	*csbB*	ATGTAATTTATGGATAATAAAAAAGGGTTTCGTTT
BPUM_2042	*sigX*	BPUM_0636		CTGTAGTTCTGATAGTTATACTTGTTTACCAAGTT
BPUM_2042	*sigX*	BPUM_0629		GATCTAATTTTTTACTTGTATAAATCTGTAAAATA
BPUM_2042	*sigX*	BPUM_1076	*yjbC*	GTGAAACTAGAGGAGACTTTTCTGCGTCTTACTAG
BPUM_2042	*sigX*	BPUM_0319		ACGCAACTATTATTAGTTATTTTCGGTAAATATCG
BPUM_2045	*resD*	BPUM_0894	*hmp*	TAACACATTTTAGACAA
BPUM_2045	*resD*	BPUM_3519	*cydA*	ATAAGTTTGTTGACTGT
BPUM_2045	*resD*	BPUM_1811	*nasD*	GAAAGATTTTTACTAGT
BPUM_2045	*resD*	BPUM_1628	*nrdI*	CTGTGTGTTTTAATTTT
BPUM_2076	*sigF*	BPUM_2959	*gerAA*	ACAGAATATTGTTTTTTCAAGCAGGTAATGCTAAAA
BPUM_2076	*sigF*	BPUM_3388	*ywhE*	TTTGTTTAAAACGGTTTTCTTTGTTCCATACTAATA
BPUM_2076	*sigF*	BPUM_3342	*spoIIR*	TTAAATAATTAGTTTTTCCTTTTCAGTTTTTTCTTT
BPUM_2076	*sigF*	BPUM_3302	*spoIIQ*	CATACATATTATTGATGCGATTGCGCAAAATAAAAA
BPUM_2076	*sigF*	BPUM_2869		ACCGCATAGTCTCTCTCGCTGCGGCTGATATTATTA
BPUM_2076	*sigF*	BPUM_3260	*gerKA*	TAAACAAAAATAGTTGTAAGTAGTAAAGAATTGTAA
BPUM_2076	*sigF*	BPUM_3248	*gerBA*	CAGGGATTCTCTTTTTTCTATTTGCATAATTCTCCA
BPUM_2076	*sigF*	BPUM_2697	*ytkD*	CATGATTAAAATGTGGTGAAGTTGGGGATGATAAAA
BPUM_2076	*sigF*	BPUM_1964	*ponA*	TATCCAAAAAGAAGCGCTGCTGTGATCAAAGAAGCC

BPUM_2076	*sigF*	BPUM_2287	*gpr*	AACCTAAACACAATTGTCTTAAGGTTAAATCTTCAC
BPUM_2076	*sigF*	BPUM_1794	*gerA*	CAACGATGTCAATGCAAATTTTAGATCATAAAATGA
BPUM_2076	*sigF*	BPUM_2181	*yqhQ*	TTCATCATACAGGTTTTACACTATTTTTTTTCCTAA
BPUM_2076	*sigF*	BPUM_2155	*spoIVB*	ATAGTTAAATAGGTTAGTAAAAGATTTCGACGGCCA
BPUM_2076	*sigF*	BPUM_2079	*dacF*	TCTGAATAAAACAGCCTGGATGAGGAAAAAATAGAA
BPUM_2076	*sigF*	BPUM_0975	*yhfW*	AATGTATGATTTTCTTCATTGTGGAAAAATTATACA
BPUM_2076	*sigF*	BPUM_0957	*pbpF*	ACTATCATATAGTTTTCTTACTATATATTAAGAAAC
BPUM_2076	*sigF*	BPUM_0892	*katX2*	AGTACAAACTAAGTTATATTAATCCCATATATCTAA
BPUM_2076	*sigF*	BPUM_3726	*yyaC*	GTGTTAGTAGTGGAGTATCCATATCTTAATTCGAAC
BPUM_2076	*sigF*	BPUM_3712	*katX1*	GCTCAAATAACGGTTCATCGATTGGAACAATTGGTT
BPUM_2084	*fur*	BPUM_2994		CGATATAATTGAAATTGAGAATCATTATCATATGAAC
BPUM_2084	*fur*	BPUM_2949		AAAATTAACTTTTACTAAAAGTTTTAGTGAATTTAGA
BPUM_2084	*fur*	BPUM_3379	*ywjA*	GGATTCCTCTATAATTGAAAATGATTATCAATTATAG
BPUM_2084	*fur*	BPUM_2873		TATGATATGATATAGTGATAACAATTTTCATTTGGGA
BPUM_2084	*fur*	BPUM_2353		CAGCATAACTTTTACTTATTGCTTTGGGTCGCTTTGC
BPUM_2084	*fur*	BPUM_1816	*yfiB*	TACACAAACACTTCCTAAAGAAAAGATTTTAGCTTGT
BPUM_2084	*fur*	BPUM_0984	*yclN*	CGAATTAACTATTACTATTAGTAAACACCGCCTTTCT
BPUM_2084	*fur*	BPUM_0918	*yheI*	GGATATGTATCTTTTTGTTAGAAAATTTCATTTATTG
BPUM_2084	*fur*	BPUM_1312		GAAATCATTTGACAATGAAAATCATTATCATTTAAAG
BPUM_2084	*fur*	BPUM_0818		AAAACCTACTCCTACTTATACGTCTAGTATTCTCCGT
BPUM_2084	*fur*	BPUM_0791	*yfhC*	TGTTTATAATGTGAATGAGAATCAATTTCAATTAAGA
BPUM_2084	*fur*	BPUM_0777	*ycgT*	AAGTATATATGATAATGAGAATCAATATCACTTATAA
BPUM_2084	*fur*	BPUM_0733	*yfkM*	ATAAAAAACTTCAAATAACATTTCTATTGTATTTAGT
BPUM_2084	*fur*	BPUM_3700	*ybbB*	TAACATTTTTGATATTGATAATCATTTTCACTTGTGG
BPUM_2154	*spo0A*	BPUM_2078	*spoIIAA*	ACTTAAACACTTTA
BPUM_2154	*spo0A*	BPUM_1425	*spoIIGA*	CCTCGACAAAAACA
BPUM_2154	*spo0A*	BPUM_0636		ATAAAAACAGTTAT
BPUM_2154	*spo0A*	BPUM_0629		TTTAGACATTTTAT
BPUM_2154	*spo0A*	BPUM_0319		TTTCGGTAAATATC
BPUM_2154	*spo0A*	BPUM_0048	*spoIIE*	TTTTGACAAAATTT
BPUM_2154	*spo0A*	BPUM_1290	*kinA*	ATTTTAACAACTAT
BPUM_2154	*spo0A*	BPUM_2843		TTTTGACCGATTCA
BPUM_2157	*ahrC*	BPUM_0362	*gabT*	AATAATAAAAAAGACA
BPUM_2157	*ahrC*	BPUM_2735	*gbsA*	AATTTAAAAATAAATT
BPUM_2157	*ahrC*	BPUM_3060	*rocD*	ATGTATCCAAATGAAA
BPUM_2157	*ahrC*	BPUM_1857	*dhaS*	TATGCAAATGTAAGTA
BPUM_2157	*ahrC*	BPUM_1776		ATGAGTAAAATGACGA
BPUM_2157	*ahrC*	BPUM_1749	*mmsA*	GTTAATCAAAAATACA
BPUM_2157	*ahrC*	BPUM_1042	*argC*	ATGAAAAAAAATGAAA
BPUM_2184		BPUM_0413	*mntH*	AAATTGCATCAAGGAAACA
BPUM_2193	*sinR*	BPUM_0972	*aprE1*	TTGTGCGTGAAAAAGAAAAATGTG
BPUM_2193	*sinR*	BPUM_0233	*epr*	ACATTGTAAGGAATTAGGCTTAAG
BPUM_2241	*zur*	BPUM_3311	*yciC*	GTTTAGCATTAGTAATGA
BPUM_2253	*sigA*	BPUM_3061		AGAGGAAACCGCAAAATTTTTCATGTAAAATAGAA
BPUM_2253	*sigA*	BPUM_3060	*rocD*	AAAATAATATTTGTTGGCATACAGGTAAGAGTAGT
BPUM_2253	*sigA*	BPUM_3058	*cggR*	AGTTGAAAATGCAAGATTATCCTGTTAAAATAATT
BPUM_2253	*sigA*	BPUM_3056	*pgk*	ACTTGGTTCTTTCCTAGAGACACGATATAATGAAG
BPUM_2253	*sigA*	BPUM_3043	*opuCA*	ACTTTTTATTTTATAAAGTTTGCACTATAATTAAT
BPUM_2253	*sigA*	BPUM_3021		TGTTTACATTGATACCCTACTAGGGTATAGTAAAT
BPUM_2253	*sigA*	BPUM_3019		GGTGGAAAATTTTCTAAAAAGTATTTTTTAAGTGA
BPUM_2253	*sigA*	BPUM_3010		AAAATAAGATTCGCATACAGGAGAAAAACATTTCT
BPUM_2253	*sigA*	BPUM_2588	*des*	CCTTTACTTTTTTATTTAAATTAAATAAGCTTACA
BPUM_2253	*sigA*	BPUM_2579	*ackA*	ACTTGAAATGCAAAGTTTTCAATGTTTAAATGAAA
BPUM_2253	*sigA*	BPUM_2556	*citZ*	GCTTTTGCAAAAAAACATAATTAAATATAATAAAA
BPUM_2253	*sigA*	BPUM_2553	*phoP*	AATTGTTCTCTTGTTAAGCTTTCGTTAAAATGAAA
BPUM_2253	*sigA*	BPUM_2547	*gapB*	TACTGGCAATTACATTCTAATGTGATATACTAATT
BPUM_2253	*sigA*	BPUM_2546	*speD*	GCTTGCAAAATGGGTCTAAACAAAGTATACTATTT

BPUM_2253	*sigA*	BPUM_2539	*thrS*	TCTTGATTTTCACGAAAAAACGTTTTATAATACGG
BPUM_2253	*sigA*	BPUM_2522	*pheS*	TGTTGCAACATGCGCTCATTTTCACTATAATGAAA
BPUM_2253	*sigA*	BPUM_2513	*lcfA2*	AACTGCCCTAAGGAAAAGTTTGCTATAATATAATA
BPUM_2253	*sigA*	BPUM_2512	*ysiA*	ACTTGACAAATCGTCCGTTTCTTTTTAACATAAAG
BPUM_2253	*sigA*	BPUM_2505	*lysC*	GATTGTACTTTTTCAAAAAGTCTGTTAAGATTGTG
BPUM_2253	*sigA*	BPUM_2503	*sdhC*	TCTTGACGCTTTTTTGACATAGGAGTAAAATGAAA
BPUM_2253	*sigA*	BPUM_2496		AATATCTTATATGCTTTTTTAGTTATAAAAGTAAT
BPUM_2253	*sigA*	BPUM_2472	*ilvB*	AATATACCAGATTTTAGAGCTAGAACTTATCTTTT
BPUM_2253	*sigA*	BPUM_2463	*clpX*	AATTGATTTTCTTGTAGAAAACCGTTAATATGGTG
BPUM_2253	*sigA*	BPUM_2458	*hemA*	TTAGTAATATTTATTATTTAAGATATAACATTAAT
BPUM_2253	*sigA*	BPUM_2449	*valS*	ATTGACGAAAACAAAAGGCATTTAGTAATATAAAT
BPUM_2253	*sigA*	BPUM_2433	*spo0B*	TGTTTTTCTATGCTGTCTTCTCTGTTATAATTCAG
BPUM_2253	*sigA*	BPUM_2427	*nadB*	TCTTGTATTTATTGTAGTGTTGTGTAATGATATGT
BPUM_2253	*sigA*	BPUM_1999	*trpE*	GATTTACAGTTGACAAAAAACAAGGAATTATAATA
BPUM_2253	*sigA*	BPUM_1987	*qcrA*	TGTTGACCTTGTGAATAGATTGATATATTATGAGG
BPUM_2253	*sigA*	BPUM_1964	*ponA*	ACTTGGATATTCGCCAAGAATTGATTATATTAAAG
BPUM_2253	*sigA*	BPUM_1944	*xpt*	GCTTTCCAAGCGTCTGAAATCTTGTTATAATTTGA
BPUM_2253	*sigA*	BPUM_2393	*yrzC*	AAATGAAAGTTGGCTAAAAGTATGATATAATTGCT
BPUM_2253	*sigA*	BPUM_2363	*yrrT*	AAGTTAAAATAGGAAATCTTTTAGGTTGACTTTTA
BPUM_2253	*sigA*	BPUM_2339	*wapA*	TATTGAAGTATCATTTCAAATGTATTATTATATAT
BPUM_2253	*sigA*	BPUM_2330		AGTGTTATATTATAAACATGCATGTATACAGTTAT
BPUM_2253	*sigA*	BPUM_1893		AAGATAATATAACTTCTAGTCTTTTTTTCCGCTAA
BPUM_2253	*sigA*	BPUM_1862	*sucA*	TGTAGAAAGTAATCCGAAACAGTGGTAGAATATTA
BPUM_2253	*sigA*	BPUM_1857	*dhaS*	TAAATAACATCGGTTAAAACGTACTTTACGTTACT
BPUM_2253	*sigA*	BPUM_1851	*yocH*	ATTCGTTTATTTATAGAAAAAGAGAAAAAATAAAG
BPUM_2253	*sigA*	BPUM_1848	*mmgD*	TATTTACAAGATGCAGTGCTTTGGATAAAATGGGA
BPUM_2253	*sigA*	BPUM_1832	*xynP*	TATTGAAAAGTCAGATCAGATTTTATAAAATGAAT
BPUM_2253	*sigA*	BPUM_1829	*xylA*	GAAATAATATTGCTTTTATAAAGATTTTTTGATTT
BPUM_2253	*sigA*	BPUM_1815	*gltC*	AGTTTTCAAATTAATATAAACAATATATAATTTAG
BPUM_2253	*sigA*	BPUM_1814	*gltA*	GATTTAATATATAACAAATATAATTAAACTTTTGA
BPUM_2253	*sigA*	BPUM_1811	*nasD*	TTTTATATCTATGTTTAAATGATGGTACTTTCTAA
BPUM_2253	*sigA*	BPUM_2284	*lepA*	GATTGAATCTTTACAGCCCTATTGATATAATCTAA
BPUM_2253	*sigA*	BPUM_2263	*cdd*	TTGTTATAATGATAAATAATTGTAAAAATCTTTAC
BPUM_2253	*sigA*	BPUM_2255	*yqxD*	TTTGTATTATAATAAAAAATTGTGATAAAATGATT
BPUM_2253	*sigA*	BPUM_2227	*pstS*	TTCATAATCTGTTTATGAAACTCGATTTTAGTTTT
BPUM_2253	*sigA*	BPUM_2205	*comGA*	TTTTGTCAGACAGATTCCTCCGACTTACAATATGG
BPUM_2253	*sigA*	BPUM_1777	*cysJ*	ATTTTACTTAATGGCAAACTTTCGGTAAAGTTCAA
BPUM_2253	*sigA*	BPUM_1776		CGCATAATATAGTTTTTGAAAAAGAAAACGTCAAA
BPUM_2253	*sigA*	BPUM_1749	*mmsA*	GATTGACTGAAAGCGTTTTTAATATTATGATTTTT
BPUM_2253	*sigA*	BPUM_1742		CATTCAAAATATTTTTGAGATGATATTAGATAAAT
BPUM_2253	*sigA*	BPUM_2193	*sinR*	TTCTATCAAACCTTTCTGAATGTGCTATAATATCG
BPUM_2253	*sigA*	BPUM_2154	*spo0A*	TCTTTCTTTCTCTGCCATCATGTTATAAAATAAGG
BPUM_2253	*sigA*	BPUM_2149	*rocR*	AAGGTAACAGATTAAGTTTTTTATAAAACAGTTCT
BPUM_2253	*sigA*	BPUM_2124	*yqjI*	CTTGATAAAAGAGGTGATGTTGTGGTAATTTCAAA
BPUM_2253	*sigA*	BPUM_1699	*citB*	TATTTACTTATGTATGAATTTGTTTTAAGATGAAA
BPUM_2253	*sigA*	BPUM_1694	*ccdA*	AAGATAATATAGGTAAAATTCCAAATAAAAGTTTA
BPUM_2253	*sigA*	BPUM_1651		GAAATGATATAGTTTTGTGGGATTTAAAACGCCAA
BPUM_2253	*sigA*	BPUM_1628	*nrdI*	TTTTGAAAAAACATAGTATACTATATATAGTGAAT
BPUM_2253	*sigA*	BPUM_1616	*aprX*	TTTTCCACCTTTTTCTATTATTTTACAAAATGTAA
BPUM_2253	*sigA*	BPUM_1615		ATTTGCCTTAAAAACATAAATTATGTAAAATAATA
BPUM_2253	*sigA*	BPUM_2096	*rapA3*	ATTTCTTCCATTCGAGGAAATATGATAGAATACAA
BPUM_2253	*sigA*	BPUM_2084	*fur*	AATTTATTATTAGTTAAATAAATAGTTAAATTTTA
BPUM_2253	*sigA*	BPUM_2081	*drm*	GTTGTCAACCAGCTTGTAAACGGTTTATACTTGAG
BPUM_2253	*sigA*	BPUM_2066	*ppiB*	AAGGTAACATAGTTAGGATTCTTTTCTCCATTTAG
BPUM_2253	*sigA*	BPUM_2048	*resA*	AAATTAAAAGTGTATTTGAAGTTTTTTACTTCTTC
BPUM_2253	*sigA*	BPUM_2042	*sigX*	ACTTGTAATATATTTGTCATAAAGTTGTAATGTAA
BPUM_2253	*sigA*	BPUM_2029	*gudB*	AAATGTATTTTTAATCACTTTTTTCTATTATATTT

BPUM_2253	*sigA*	BPUM_2009	*folE*	ATGTTAAAATCTACGTGAATTTTTACCACTGTTTT
BPUM_2253	*sigA*	BPUM_1598	*recA*	TCTTGGCAAATGACGTAAAACAAGGTATAGTATAG
BPUM_2253	*sigA*	BPUM_1585	*spoIIIE*	CCGACTCATATGAATGATTCTATGCTATAATAAAG
BPUM_2253	*sigA*	BPUM_1579	*asd*	TGCTTCAAGCCTTGCTTTTCTATGGTAAAATAAAA
BPUM_2253	*sigA*	BPUM_1487	*ylpC*	GATTGATTTATCATCTATGAGATTATATACTACTA
BPUM_2253	*sigA*	BPUM_1456	*cysH*	GCTTTTTTTATTTGGTTAAAAATAGTAAAATAGAC
BPUM_2253	*sigA*	BPUM_1446	*pyrR*	GATTGACAGGCCGTTCTTTTTTTGAAATAATAAAC
BPUM_2253	*sigA*	BPUM_1442	*ileS*	CGTATATTATGGTAGTATTTAATTGTTCTCTATTT
BPUM_2253	*sigA*	BPUM_1425	*spoIIGA*	CATTAAATATCAATGAAAAGTAATTTAAACAAAAG
BPUM_2253	*sigA*	BPUM_1421	*ftsA*	CATTGTATTGTTGTTCAGCAAATAATAGAATAGAA
BPUM_2253	*sigA*	BPUM_0993	*addB*	AATAGCTATTTCCCTTTATTTTCGCTAGAATATGA
BPUM_2253	*sigA*	BPUM_0981	*ydeL*	TTAGTATTTTTCTGGTTTATTTTGCTTATATTTAC
BPUM_2253	*sigA*	BPUM_0978	*comK*	TAATTATCATAGATTTTATTGGTTTTTAATTGTGT
BPUM_2253	*sigA*	BPUM_0972	*aprE1*	TCTTTTTAAATGGCAAAAACAATGATAAAATAATA
BPUM_2253	*sigA*	BPUM_0969	*lcfA1*	CTTGTGAATTTTTTCGATTGTCTATTATAATAGAA
BPUM_2253	*sigA*	BPUM_0965	*gltT*	CTTTAATTCATATATTAAAACTTAATATAATGGGA
BPUM_2253	*sigA*	BPUM_0957	*pbpF*	ACTATCATATAGTTTTCTTACTATATATTAAGAAA
BPUM_2253	*sigA*	BPUM_0930	*hemZ*	CATTGCTTTTCCTACATAGATTTTGTAAAATGAAA
BPUM_2253	*sigA*	BPUM_0902	*sigM*	AAGATAATATTGTTTAAAAATAATGGACCATGATT
BPUM_2253	*sigA*	BPUM_1379	*ctaA*	GAGATAGTATGTTCTATTTGTCGACATTTTTTTGT
BPUM_2253	*sigA*	BPUM_1374	*gls*	TATTTAATATCTTCAAAAGCAGGTTAAACAGTGAA
BPUM_2253	*sigA*	BPUM_1355	*pdhA*	AATTGACGGACTATTTTTCTTGTTGTAAACTAGTC
BPUM_2253	*sigA*	BPUM_1337	*sipT*	TTGTGAATTTTTAAATGACATGTGATAAAGTAAAG
BPUM_2253	*sigA*	BPUM_1312		AATTTCACATGTATATTATTAACTTTTACTAATAG
BPUM_2253	*sigA*	BPUM_1310	*ccpC*	TTTTGCCTTCATAAGCAAAAATTTATATAATGGCA
BPUM_2253	*sigA*	BPUM_1303	*ykuF*	CCTTTATAAATAAAATTAAGTATAGCATCATATAT
BPUM_2253	*sigA*	BPUM_0897	*lytE*	AATGGACTAAAAAGAGAAATGAATGTATCATTTAT
BPUM_2253	*sigA*	BPUM_0896	*phoA*	TATTTACTATGTAAGTAAAGAGAAAAATCAGGTAA
BPUM_2253	*sigA*	BPUM_0894	*hmp*	ATGTGACAACATTTAAAACATGTATTATAAATATT
BPUM_2253	*sigA*	BPUM_0886	*yhcY*	CGAGTACTATATTACTCGTATTACTTTTCTCTTCC
BPUM_2253	*sigA*	BPUM_0882	*glpF*	TATTGACAACGTTTTCATCCTCTGATACTATAGCA
BPUM_2253	*sigA*	BPUM_0854		ATAATAAAAAAGTCTAAATAGATTTTTTACTAACT
BPUM_2253	*sigA*	BPUM_0840	*sipP*	AGACTAATAAAGATACCATTTAGTTTTCTATTTCT
BPUM_2253	*sigA*	BPUM_0827	*perR*	GGTACACTATTTATAAAGATTATAATATAAGAATA
BPUM_2253	*sigA*	BPUM_0812	*fabL*	ATTGCCCTTCTTCGTTTAAATAAGGTAAAATTTGT
BPUM_2253	*sigA*	BPUM_1279	*ptsG*	ATTGAAAAATGAATATCCTTTATGCTACAATACTG
BPUM_2253	*sigA*	BPUM_1275	*ykvW*	TAATTGAAAATTGATCAGAACCTGTTATAATGAAT
BPUM_2253	*sigA*	BPUM_1261	*clpE*	AGTTGAAAGTCAAAATAGGTCAGAGTATACTATAG
BPUM_2253	*sigA*	BPUM_1252	*mtnW*	CATTGACAACATGAAAAAACGGTGCGATAATTCAG
BPUM_2253	*sigA*	BPUM_1251	*mtnE*	CTTTACAATTCGTGAAACAATCCATTACAGTGTTT
BPUM_2253	*sigA*	BPUM_1250	*mtnK*	TTTGTGACATTACCTAACAAAGTGCTTAACATTTC
BPUM_2253	*sigA*	BPUM_1227		AGTTTACTTATTCTATTGGATCACGTAAAATCAAA
BPUM_2253	*sigA*	BPUM_1214	*guaD*	GTTTCCTTTTCTAGCATGATTCGTTTATACTAGAG
BPUM_2253	*sigA*	BPUM_1211	*ohrA*	CGTTGACTCTTGATTTTAAATTGTGTACAATTCAA
BPUM_2253	*sigA*	BPUM_1209	*rapA2*	TTTGTATTTCTCCGGTCAAGTATGATAAAATATGG
BPUM_2253	*sigA*	BPUM_0761	*thrZ*	TGTGTTTTTTTATTTTTAAAAAAAGAAAGGTGTGT
BPUM_2253	*sigA*	BPUM_0728	*treP*	TGTTGACTACATGTATATACAAATATAAAATAATA
BPUM_2253	*sigA*	BPUM_0720	*nagP*	TATAGATAACTAGATGTGTATATGATATATTTTTT
BPUM_2253	*sigA*	BPUM_0719	*yflG*	AATTAATTGATGTTGTACCAGCTGATAAAATAAAA
BPUM_2253	*sigA*	BPUM_0710	*citM*	TATTGCCCGAAAAAGAATTAAAATTCATAATATTT
BPUM_2253	*sigA*	BPUM_1184	*dppA*	TATTGAAAAATCAATCAAGTAATCATATAATTTTT
BPUM_2253	*sigA*	BPUM_1181	*htrA*	TGAATAAAATTGTTAAAGGGTGTTACAAAAGTAAA
BPUM_2253	*sigA*	BPUM_1141	*rapA1*	AAATGATAATATATGTCATTATAGGGAAAATAGAG
BPUM_2253	*sigA*	BPUM_1132	*rapH*	TTTTGTGATGAGAATGAAAATGAGGTTTACTGAGG
BPUM_2253	*sigA*	BPUM_1119	*yjcI*	TCTTGAAAAATGAGATTTAATGCGTTACAGTATTA
BPUM_2253	*sigA*	BPUM_0692	*yfmP*	CGTTTACGTTAAGGTAATAAAGGTGTATAATAAAT
BPUM_2253	*sigA*	BPUM_0667	*licH1*	AAAATCCCATGCAAGTTATTTAAGTATATTTTTCA

BPUM_2253	*sigA*	BPUM_0643	*opuE*	GAAATAGTATAGTTTAAAATGACAAGATTTCGTTG
BPUM_2253	*sigA*	BPUM_0641	*phoB*	GGTTGAACCATTGCTTTTATTCAAATAAAATACAA
BPUM_2253	*sigA*	BPUM_0636		CTTTTCGAATTAGATGACTTGAAAATATAAGTAAA
BPUM_2253	*sigA*	BPUM_0629		TATTTACAAATATACCATTGTATGATTTAATATGT
BPUM_2253	*sigA*	BPUM_1079	*mecA*	GTTTTTATTTCTTTTATTCATGTTTTATCATAAAA
BPUM_2253	*sigA*	BPUM_1077	*spxA*	TCTGTTCACACTTACTTTTTTATAGTATAATCACT
BPUM_2253	*sigA*	BPUM_1057	*fabHA*	TATTGCCAAGGAGAAAAAAATAAGGTAAAATTAGT
BPUM_2253	*sigA*	BPUM_1054	*med*	TCTTCCATCTTATCTTTTTTCAGTATAAAATAAAC
BPUM_2253	*sigA*	BPUM_1042	*argC*	CAATTGAATTAATTTTTATACATGTTATAATGTTA
BPUM_2253	*sigA*	BPUM_0596	*purE*	CGTTGACATCATGAACATCCGTTGTTAAGATAAAC
BPUM_2253	*sigA*	BPUM_0548	*guaA*	TTTTTTCTGTGCGTTTATAAAAAACTATATTGGAC
BPUM_2253	*sigA*	BPUM_0543	*gabP*	TAAGTCATACTATTTTAAATTTTATAAGATTTTTG
BPUM_2253	*sigA*	BPUM_0515	*yaaJ*	CCTTGAAAGCATGATCTTTAAGAGCTAAAATCTTA
BPUM_2253	*sigA*	BPUM_0501		TTAATAGAATTGTATAGCAAGAATAGTTCAGTTGT
BPUM_2253	*sigA*	BPUM_0470	*yccC*	CCTTTTTGAAGAATTTACTATATTTTATCATAAAT
BPUM_2253	*sigA*	BPUM_0463	*bglS*	CTAGTAAAATAAAGTATTGAAGGATACTTTATTCT
BPUM_2253	*sigA*	BPUM_0421		TGTTTACAATTTTCTAAAAATAAGGTATAAATAGT
BPUM_2253	*sigA*	BPUM_0419	*dctP*	GCTTCACAGCCTTTGTTGAAATCGCTATCATTTAG
BPUM_2253	*sigA*	BPUM_0362	*gabT*	AAGATAGTATGGTTTTTTATAAGTCTTATTATTTT
BPUM_2253	*sigA*	BPUM_0361	*gabR*	TTTTATTATTCTGAATATTTTTTGGTATGATAGAA
BPUM_2253	*sigA*	BPUM_0345	*yxeK*	GGTTGACAATCCTGTAAAAAGGGCATATAGTAATA
BPUM_2253	*sigA*	BPUM_0319		TATTAGTTATTTTCGGTAAATATCGTAAATTATTA
BPUM_2253	*sigA*	BPUM_0296	*ykrU*	TATTGGCAAAGAAGATTGAAACCTTTATGATAAAG
BPUM_2253	*sigA*	BPUM_0289	*nadE*	CAAAATGTTCTTCCTGAAAACATGATATAATGGAG
BPUM_2253	*sigA*	BPUM_0280	*yhjL*	TATTTAACATGTGTATAATTGTTTAGGAAATGTGT
BPUM_2253	*sigA*	BPUM_0277	*ldh*	CCTTGCAAAAGTTTGTGAAGTATTGCACAATAAGA
BPUM_2253	*sigA*	BPUM_0275	*amhX*	TTTTCAGAATAATCAAATGATGTGTTATATTATAA
BPUM_2253	*sigA*	BPUM_0270	*opuAA*	AAAAAATTATAAAATTAATTTTAAACTTATTTTTT
BPUM_2253	*sigA*	BPUM_0260	*yceC*	TCATGAATTTTGTCGAATTTTCATGTATGATGATT
BPUM_2253	*sigA*	BPUM_0244	*glsA1*	GGTGTCATATGTTCCTCTGAGAAAAGAAGTGTTAC
BPUM_2253	*sigA*	BPUM_0239	*ydaB*	ATTTTACTTTCAGCTGGTTTTATTTTAGAATGTGA
BPUM_2253	*sigA*	BPUM_0233	*epr*	GAGATAACAGATGATCAAAATATACTTAAAGTTAA
BPUM_2253	*sigA*	BPUM_0209		AAAATAAAACGGATTTTTGCATAAAAAAACTTTTT
BPUM_2253	*sigA*	BPUM_0182	*glpT*	AAATTGAAAAACTAAAAAAATCTGCGAAAATTAAT
BPUM_2253	*sigA*	BPUM_0180	*ybxG*	AATTGTTTTTTAAGAGTAGATTTATTATGATAATA
BPUM_2253	*sigA*	BPUM_0093	*rpoB*	TTTGTATAAATTATGCATACTTGGGAAAACTAATA
BPUM_2253	*sigA*	BPUM_0077	*gltX*	TTTTGATGAGAATGAGATTTGATGATAAAATACGT
BPUM_2253	*sigA*	BPUM_0068	*ctsR*	TAAATATCATTCTAATTTCAGTTTATATCAGTTTC
BPUM_2253	*sigA*	BPUM_0057	*cysK*	AATTGATAATTTAATATTAATACAATAAAATTACT
BPUM_2253	*sigA*	BPUM_0048	*spoIIE*	TATTGACGGACAAGCGATTCCTTTGTATTATAGAT
BPUM_2253	*sigA*	BPUM_0034	*glmU*	CCTTGAAATCAGTCCTTATTTAGGATATATTTTTC
BPUM_2253	*sigA*	BPUM_0001	*dnaA*	CATTGCGCCATCTTGTTTATTTTGATATTATATTT
BPUM_2253	*sigA*	BPUM_3735	*spoIIIJ*	TAATGAAAAAATGTTACCATACTGATTAAGTGAGT
BPUM_2253	*sigA*	BPUM_3717		TTAGAAAGTTTGTTTCTGAGCGTGGTAAAATTTTA
BPUM_2253	*sigA*	BPUM_3693	*purA*	CGTTGACTTTTTCGGTTTGCATTGATAAAATAAAT
BPUM_2253	*sigA*	BPUM_3689	*yycF*	TTTTTATCTTTGCCACGTAAAACGTTATAATAGAC
BPUM_2253	*sigA*	BPUM_3684	*yyxA*	CGTATGATTTTTATAATCGTATGATTAAGATAAGA
BPUM_2253	*sigA*	BPUM_3662	*bglA*	AAACTAATATCGTGTATGGCTAAAAAGAAACTATA
BPUM_2253	*sigA*	BPUM_3652	*aapA*	TAAATAATATGTACTTAATTTCCCCTAACTACTTC
BPUM_2253	*sigA*	BPUM_3644		TGTTGACGCTTTCATATAATCGATATATACTAAAA
BPUM_2253	*sigA*	BPUM_3641	*rapK*	CTGTTAATTTTTTTCTAAAATTAGTAAACATTTTT
BPUM_2253	*sigA*	BPUM_3618		AATATAAAATAGATTTAAAAAACTGTACAAAGTAG
BPUM_2253	*sigA*	BPUM_3611	*ywfK*	AAGATCATATCTCAGAAATTTTTTTAATCATTTAT
BPUM_2253	*sigA*	BPUM_3599		TTTTTTCATGCTTTTTGAAATATTTTACAGTTATA
BPUM_2253	*sigA*	BPUM_3587	*deoC*	TGTTTACAATTGTTCATCGGTCATATATGATGAAA
BPUM_2253	*sigA*	BPUM_3566		CGTTGACATCATAAAATTAACATGATACTTTATAA
BPUM_2253	*sigA*	BPUM_3548	*ansA*	GTTGTCATATAGTATAAAACACTATCTTACATGCG
BPUM_2253	*sigA*	BPUM_3541		TAACTAGTTTTTTAGAACGTGAAACTTAACTTTTT

BPUM_2253	*sigA*	BPUM_3519	*cydA*	ACTTTATTTTTGAAGCGTATGGGTGTAAAGTACAG
BPUM_2253	*sigA*	BPUM_3507		GTTGGCAATTTTTATTTCTTTTCAGTAGAATATGA
BPUM_2253	*sigA*	BPUM_3485	*ywbI*	TCAATCCATTTTAATTGAATAATTTTAAAATGAAG
BPUM_2253	*sigA*	BPUM_3475	*rapE*	AATGTATAATTACCTATTTATCCAAAAATAATCCT
BPUM_2253	*sigA*	BPUM_3452	*sacP*	TGATGAAAGCGTAATCAAAATTTGATATCGTGGGA
BPUM_2253	*sigA*	BPUM_3415	*eutD*	TATGGAAATTGATTTAATTAGTTTGAAAAATAAGA
BPUM_2253	*sigA*	BPUM_2987	*uxaC*	GATTGAAATCAACTTTGTAAACGATTACAATTAAA
BPUM_2253	*sigA*	BPUM_2957	*fumC*	TATTGAACTAAAATTAAAATAAAGATAAAATTTTA
BPUM_2253	*sigA*	BPUM_2953	*cssR*	AGTTTAATATATTATGAATAAGTATCTTTTTTTCG
BPUM_2253	*sigA*	BPUM_2952		GCTTTTTTTCTATGAATAAGTATTATATAATTTGA
BPUM_2253	*sigA*	BPUM_2942		TCTTGAACTGTTTCGGAAAATCTTTTAAGATAAAG
BPUM_2253	*sigA*	BPUM_2905		TTTGGAAAAGAAAACTAAATTTTGAAAAACTAGAC
BPUM_2253	*sigA*	BPUM_3388	*ywhE*	TTTTTCATATCACCTAAGTATAGTTTAACATTTCT
BPUM_2253	*sigA*	BPUM_3387	*speE*	TCTTTACAATTTGATATGAATCCACTATACTTTTT
BPUM_2253	*sigA*	BPUM_3374	*ywjF*	TCTTGTCGTTCTTTACAATAAAGAACATCATATGA
BPUM_2253	*sigA*	BPUM_3360	*pyrG*	TCTTATTTTTCAAGAGATTTAGGGATACACTCTAA
BPUM_2253	*sigA*	BPUM_3358	*spo0F*	AAAATATTATCTTATATTTCTCTTATAATCCTTTT
BPUM_2253	*sigA*	BPUM_3348	*ywkA*	CTTTAAATTAATAGTTCACTTTATTTATATTTTAG
BPUM_2253	*sigA*	BPUM_3335	*glyA*	TTTTACATTTCACTCAAAAACCGTGTACAATGAGA
BPUM_2253	*sigA*	BPUM_3311	*yciC*	GCTTGACATTCTTTCGTGATCATTTTAAAATACAA
BPUM_2253	*sigA*	BPUM_3300	*nrgA*	ATATGTTATATATGACTTTTCTCTCTATAATGAAG
BPUM_2253	*sigA*	BPUM_2843		ATGAACCATAAGCACAATCATTTGGTATACTGAAA
BPUM_2253	*sigA*	BPUM_2827		TTTACTTTTATTACACTTAATTTAGTTTAATAAAT
BPUM_2253	*sigA*	BPUM_3287	*rapD1*	ACTCATTGTTTTTTCACAAATATGGTAATATATAT
BPUM_2253	*sigA*	BPUM_3271	*alsS*	CATTTAGTATAGTACTTTTATATAAAAGTATTTTT
BPUM_2253	*sigA*	BPUM_3263	*rbsR*	TCTTTCTTTTCTTTTTTTATTTTTTTAAGATTGTC
BPUM_2253	*sigA*	BPUM_3252	*ywtG*	AAGTTATTATGCGTTTACTTTATATAAACAGTATT
BPUM_2253	*sigA*	BPUM_3245	*kdgR*	TCAATACAATACGAGAAAAATTCAATAAACTTTAG
BPUM_2253	*sigA*	BPUM_3235	*pmi*	GGATTGACTTCTCTTTAGATAGCGTCATAATAAAA
BPUM_2253	*sigA*	BPUM_3231	*tagA*	GGAGTAATATAGTGTTACGGATACTTTCCAATTCT
BPUM_2253	*sigA*	BPUM_3230	*tagD*	TCTTAACCTTTCATAGGCATTGTGATATAATGAGG
BPUM_2253	*sigA*	BPUM_3219	*gtaB*	AAAATAACATGATTTCTAGAGAAGCTTACTATTTG
BPUM_2253	*sigA*	BPUM_3217	*lytR*	ATTATAAAATTCCATTTAAATTCCAAAAAACTTTG
BPUM_2253	*sigA*	BPUM_2735	*gbsA*	CTTTGACAGAAAAATTTGAACATGATACGTTAATT
BPUM_2253	*sigA*	BPUM_2719	*menF*	TGTTTTCTTTACTAAGTTCATCTGTTAAGGTTTAT
BPUM_2253	*sigA*	BPUM_3199	*degS*	TTTAGAATTTTACAGCGTATTTTGTTATGATGATA
BPUM_2253	*sigA*	BPUM_3196	*comFA*	TATTTAGAATTTTTAGAGAAAAGAGAATCTTTTCC
BPUM_2253	*sigA*	BPUM_3178	*secA*	TCTTTGGAAATGAAGGAAGGTATGGTATGATAAAT
BPUM_2253	*sigA*	BPUM_3159		TAAATTTTTTATTTTAGAAACTATTTATCTGTTTA
BPUM_2253	*sigA*	BPUM_3102	*clpP*	GTTTGACCTTTATTGACCAAAAATGTATCATTGAA
BPUM_2253	*sigA*	BPUM_2697	*ytkD*	TTCTTCAGAGAAATGTGATCTGTGGTATGATGTTT
BPUM_2253	*sigA*	BPUM_2687	*pckA*	TTTATCATATCTGATAAGAATACTTACACAATATG
BPUM_2253	*sigA*	BPUM_2677	*ytrA*	TATTGACTTTTCGATTGAGTGTCTATATTGTATTA
BPUM_2253	*sigA*	BPUM_2643	*ytkP*	ACTATCATATTGATTTCTTTACGCCTTACTCTTCT
BPUM_2253	*sigA*	BPUM_2635	*malS*	TTTCGCCTTTCACACAAAATAACGACAAAATGAAC
BPUM_2253	*sigA*	BPUM_2627	*ytpT*	GCTTTTCTGCATGAAAAAGCAATGGTATACTAAAA
BPUM_2253	*sigA*	BPUM_2617	*acuA*	GTTTTAATTCTTGCAGGTTTTAAGATAAAATAGAG
BPUM_2253	*sigA*	BPUM_2616	*acsA1*	GAGATAAAATAGAATTTTGGACGTTCTTAATTTTG
BPUM_2253	*sigA*	BPUM_2615	*tyrS*	TAATTAAAAAAGTAGTCAACTGAACTATAGTTTTT
BPUM_2253	*sigA*	BPUM_2610	*rpsD*	AAAATAAAAAGTGTTTATTTTTGTTTATTCCTCCT
BPUM_2253	*sigA*	BPUM_2606	*ezrA*	AATGTACCATAGTATTATTGTTACTTTTGCTGTCG
BPUM_2253	*sigA*	BPUM_2600	*ytcI*	TAAATAAGATTTTATTTTTAAAGTATTATTAATGG
BPUM_2257	*yqzB*	BPUM_2687	*pckA*	AATATTTAAATAGTATAGACTATTCTTATGAATGTGTTATACTAATC GAAACAAAAC
BPUM_2257	*yqzB*	BPUM_2547	*gapB*	ATATGACCGTTAATGTAAGATTACACTATATGATTAAAGTATTTATT TAAATAACTC
BPUM_2330		BPUM_3252	*ywtG*	TTTCTTATGGCTGATATT
BPUM_2330		BPUM_2330		TATTTGTACGTACATATG
BPUM_2393	*yrzC*	BPUM_0345	*yxeK*	TATTGATAAAACCGATTAATTCAATCAGAAAAATGTAT

BPUM_2393	*yrzC*	BPUM_0057	*cysK*	ATTATAATTATGTTATTTTAATGAGCCTCTATCTCCAC
BPUM_2393	*yrzC*	BPUM_2643	*ytkP*	CTATACTTAGGGTTATTTTATTACTGCCAAGAGACTGT
BPUM_2393	*yrzC*	BPUM_2363	*yrrT*	ACGGATTAAAACATACTATACGCATAGGAATAAATAGA
BPUM_2393	*yrzC*	BPUM_2341	*ydbM*	GTGTAATTTAGGATATTCATTTCACACAAAATAAATAA
BPUM_2429	*yrxA*	BPUM_2427	*nadB*	ATTTGGTTAGGTAGGTAGAGGATACTCTTTTAAAAAAGAACATAAA TAACATCACAACACATTACTATACACAGTTCTGTCCACATTTACTGT CCTCCCCGTTACCAC
BPUM_2512	*ysiA*	BPUM_0239	*ydaB*	AGAATGTGAATGAGATAAAATCTGCCAACCCATTCACTG
BPUM_2512	*ysiA*	BPUM_2942		TTTTTTTTGCACAAAAAATGAATGACTGTTCATTCAAAA
BPUM_2512	*ysiA*	BPUM_3374	*ywjF*	ATAACTGACTTATGAGTAAGTAATAAATACTATCTCCAT
BPUM_2512	*ysiA*	BPUM_2513	*lcfA2*	CTTACCGATATAGGAGTAAGTTATACTTGCTCCTTAAGT
BPUM_2512	*ysiA*	BPUM_2512	*ysiA*	CTTTTTAACATAAAGTTATGAATGAATACTCATTCAATT
BPUM_2512	*ysiA*	BPUM_0969	*lcfA1*	GAATGTTTGGCCATAAAATGAATGGTTATTCATTCATTG
BPUM_2512	*ysiA*	BPUM_1303	*ykuF*	TGATATGATAGGGAATAATGAATGAATAGTCATTCAAAA
BPUM_2553	*phoP*	BPUM_3689	*yycF*	AAAAAACGAGTTTACTTTTTCCA
BPUM_2553	*phoP*	BPUM_2553	*phoP*	TCGCTTTTCACATAAGATACAAA
BPUM_2553	*phoP*	BPUM_2227	*pstS*	AAAACCTTAACATTCCATTTAAA
BPUM_2553	*phoP*	BPUM_2048	*resA*	TTAATTTTCACATAAACTTCAAA
BPUM_2553	*phoP*	BPUM_0896	*phoA*	ACAATTTTAAAGGGATTTTTACG
BPUM_2553	*phoP*	BPUM_0732	*yfkN*	AACATTTCTACAATAGGTATTGT
BPUM_2553	*phoP*	BPUM_3231	*tagA*	TAAATTTCCAAACATTTCAATAA
BPUM_2553	*phoP*	BPUM_3230	*tagD*	AATAACTTTACAAACCTTTAAAT
BPUM_2553	*phoP*	BPUM_0641	*phoB*	AAATTATGAAATAAATGATAAAA
BPUM_2590	*desR*	BPUM_2588	*des*	TCATATTGAATACATGA
BPUM_2620	*ccpA*	BPUM_3271	*alsS*	TGAATTCGCGTACA
BPUM_2620	*ccpA*	BPUM_2579	*ackA*	ACATTCGCCATTGT
BPUM_2620	*ccpA*	BPUM_2472	*ilvB*	CCTTTGGCCACAAT
BPUM_2620	*ccpA*	BPUM_0501		GGAAAATTCTTAAA
BPUM_2620	*ccpA*	BPUM_0463	*bglS*	AGAAAACGCTTTCA
BPUM_2620	*ccpA*	BPUM_0451	*acoA*	TAATTAGCCTTAGT
BPUM_2620	*ccpA*	BPUM_0421		TGAACCTGTTTACA
BPUM_2620	*ccpA*	BPUM_0419	*dctP*	ACTTTAGCGATAGT
BPUM_2620	*ccpA*	BPUM_0239	*ydaB*	AATAAACCCTTTCA
BPUM_2620	*ccpA*	BPUM_0209		TCTTTCGCAACTGT
BPUM_2620	*ccpA*	BPUM_3644		TGTTGACGCTTTCA
BPUM_2620	*ccpA*	BPUM_3618		ACTTTCGCCAACGG
BPUM_2620	*ccpA*	BPUM_3587	*deoC*	TAACTGTGTTTACA
BPUM_2620	*ccpA*	BPUM_3566		TCTTTCGCAACTGT
BPUM_2620	*ccpA*	BPUM_3519	*cydA*	AGTAAACCATTTCA
BPUM_2620	*ccpA*	BPUM_3452	*sacP*	ACTTTCGCATTAGT
BPUM_2620	*ccpA*	BPUM_3415	*eutD*	ACATTCGCGATATT
BPUM_2620	*ccpA*	BPUM_3263	*rbsR*	ACATTTGCCAATGT
BPUM_2620	*ccpA*	BPUM_3252	*ywtG*	TTTAAACGCTTTTT
BPUM_2620	*ccpA*	BPUM_2735	*gbsA*	AGTTTTGCAATTTA
BPUM_2620	*ccpA*	BPUM_2617	*acuA*	TGAAACCGTTTTAA
BPUM_2620	*ccpA*	BPUM_2616	*acsA1*	AATTTTGCCAAAGT
BPUM_2620	*ccpA*	BPUM_2600	*ytcI*	ACTTAAGCCACATT
BPUM_2620	*ccpA*	BPUM_3063	*sigL*	TGAATACGCTTCAA
BPUM_2620	*ccpA*	BPUM_3061		TTTTAGCGCTTCAA
BPUM_2620	*ccpA*	BPUM_3019		TCTTTCGCAATAGT
BPUM_2620	*ccpA*	BPUM_2556	*citZ*	GGATAGCGCTTTCA
BPUM_2620	*ccpA*	BPUM_2513	*lcfA2*	ACTTCAGTAAATGA
BPUM_2620	*ccpA*	BPUM_2496		TCTTTCGCGATTGT
BPUM_2620	*ccpA*	BPUM_1893		TCTATACGATTTCA
BPUM_2620	*ccpA*	BPUM_1857	*dhaS*	TGTATACGTTTACA
BPUM_2620	*ccpA*	BPUM_1848	*mmgD*	AATTTCGCGTATCT
BPUM_2620	*ccpA*	BPUM_1832	*xynP*	ACTTTCGCGAAATA
BPUM_2620	*ccpA*	BPUM_1829	*xylA*	TGAATCAACTTACA
BPUM_2620	*ccpA*	BPUM_1776		ACTTTTGTCTTATA

BPUM_2620	*ccpA*	BPUM_1749	*mmsA*	ACTTTCGCAAAAAT
BPUM_2620	*ccpA*	BPUM_1742		TGTAAACGCTCAAA
BPUM_2620	*ccpA*	BPUM_2149	*rocR*	ACCTTTGCCACGTT
BPUM_2620	*ccpA*	BPUM_0969	*lcfA1*	TGAAAGCGATTTAC
BPUM_2620	*ccpA*	BPUM_0965	*gltT*	TGGAAATGATTTAA
BPUM_2620	*ccpA*	BPUM_0922	*msmX*	CACTTCGCGAAAGT
BPUM_2620	*ccpA*	BPUM_0882	*glpF*	TGACAACGTTTTCA
BPUM_2620	*ccpA*	BPUM_0728	*treP*	ACTTTCGCGATGTA
BPUM_2620	*ccpA*	BPUM_0710	*citM*	ACTTTCGCATATGT
BPUM_2620	*ccpA*	BPUM_1171		TGAAAACGCTTAAA
BPUM_2620	*ccpA*	BPUM_0667	*licH1*	ATAAAGCGCTTTCT
BPUM_2821		BPUM_3348	*ywkA*	ATGAATTTATTTAATTTATTGCGATGCTTTTTTAATAGATTAAAAGTTGCG
BPUM_2821		BPUM_2827		AAACTCCTGTTTATTTTACTTTTATTACACTTAATTTAGTTTAATAAATAG
BPUM_2821		BPUM_2635	*malS*	CATCTGCATGTTTTTTCGCCTTTCACACAAAATAACGACAAAATGAACTTG
BPUM_2837	*comA*	BPUM_1141	*rapA1*	TGAGGAAAACAACAA
BPUM_2837	*comA*	BPUM_1132	*rapH*	TAAGGGATGGAGAAA
BPUM_2837	*comA*	BPUM_0629		AACTTAGTAAGACGT
BPUM_2837	*comA*	BPUM_0319		TAAGCCGTATTGCGT
BPUM_2837	*comA*	BPUM_3475	*rapE*	TGCGTCATGCCGAAA
BPUM_2837	*comA*	BPUM_2096	*rapA3*	TGCGGGATACCGCAA
BPUM_2903		BPUM_1214	*guaD*	CCTTTTCTAGCATGATTCGTTTATACT
BPUM_2952		BPUM_3684	*yyxA*	CAGTCAGCCAATACATTTGTATTTGAGAGCAATCATGATGTCGGTATGCTGCAAATGGGTAGATACCCGTGGAGCATTAAGCGCAGAATTCTGAGCGATGTCGGACATGTTTC
BPUM_2952		BPUM_2952		GTCATTATAACGGGATACAAAGGCGGCATGTCAACAATCACACGCTTTTTTCACAATTTCACATATTCTTCTCATCATTATCCCATAATGTTTGTTTATGATAAGCTCAAGGG
BPUM_2952		BPUM_1181	*htrA*	TGTCTCCACAGTTTAATCTCCTCTATTTTGCTTAGATTCACTACTTATTTTAACAATTTCCCACAATGTTTTCATTTTTATCCCACACTTTTTTCTTACGATCAAACCACACA
BPUM_2963		BPUM_0886	*yhcY*	AAAAAGAGGTAGTTTTCCTACTCTTTT
BPUM_3058	*cggR*	BPUM_3058	*cggR*	CTGTGGCGGGACGTTTTTTGTCACAGTGGGACTTTTTTAGTCCAAC
BPUM_3061		BPUM_1776		TAGTTTTTGAAAAAGAAAA
BPUM_3061		BPUM_1749	*mmsA*	GATGCTCCCGCATTTTGCA
BPUM_3061		BPUM_2029	*gudB*	GATTCAAATGTATTTTTAA
BPUM_3061		BPUM_0362	*gabT*	ATTTCACCTTTAATTTTCG
BPUM_3061		BPUM_2735	*gbsA*	TTTTTTTAGAAATACCTGA
BPUM_3061		BPUM_3060	*rocD*	CGTGCAAAGAAATTTTGCT
BPUM_3061		BPUM_1857	*dhaS*	TCACCAAAATCATTCAGCA
BPUM_3061		BPUM_2149	*rocR*	AACGGAAAAATATGGTGTG
BPUM_3061		BPUM_3061		TCGTTTTAAAGAAACGTGC
BPUM_3063	*sigL*	BPUM_3060	*rocD*	AGAAAGTTGGCACGATCCTTGCATAACATATGGATG
BPUM_3063	*sigL*	BPUM_1857	*dhaS*	TGAACAAAATTGCAAACTCGGCAAACGGTCAAAACT
BPUM_3063	*sigL*	BPUM_1776		TCAGATAATGGTAATTTACTATGTACGGGCGGCCAA
BPUM_3063	*sigL*	BPUM_1749	*mmsA*	TGTTAGTTTGTGTAGGATATTCGTACTGTGTAAAAA
BPUM_3063	*sigL*	BPUM_2029	*gudB*	TATCAGCTGATACCGTCATTTCCATTTTAGCGGAGT
BPUM_3063	*sigL*	BPUM_1171		TTTGCGAATTTTTACTTTCTCTTTACGAACTGAAGT
BPUM_3063	*sigL*	BPUM_0451	*acoA*	GTGATGTTGGCACGCTTTTTGCTTGTATAGAAGTGC
BPUM_3063	*sigL*	BPUM_0362	*gabT*	GGAAATTAAAAGCACATTTAAACTACTATTTATTTA
BPUM_3198	*degU*	BPUM_0233	*epr*	TTTCAGGTTACTTGTAAAAA
BPUM_3198	*degU*	BPUM_3452	*sacP*	GGTCAATAGATTCGAAAAAA
BPUM_3198	*degU*	BPUM_0978	*comK*	TCTAAAATAACCAAAAATTA
BPUM_3198	*degU*	BPUM_0972	*aprE1*	AAGAATTTCTTGGACAATAT
BPUM_3198	*degU*	BPUM_1337	*sipT*	TTTTAAATGACATGTGATAA
BPUM_3198	*degU*	BPUM_0840	*sipP*	TAAAGATACCATTTAGTTTT
BPUM_3198	*degU*	BPUM_0728	*treP*	TATAAAATAATAATCAAACA
BPUM_3453	*sacT*	BPUM_3452	*sacP*	CGTGGGATTGTGACTGTCAGGCAGGCAAGACCTAAAATT
BPUM_3453	*sacT*	BPUM_0728	*treP*	TAATAGACCGAAACCGTTTGGAGACGGATTTCTACTGCG
BPUM_3588	*deoR*	BPUM_3587	*deoC*	GATGAACAAAATTTCATAACTGTGTTTACAATTGTTCAT

BPUM_3611	*ywfK*	BPUM_1777	*cysJ*	ATCATGACAATCAATATTATATATTT
BPUM_3611	*ywfK*	BPUM_3611	*ywfK*	CCTAAATTTTATTTATAGAAGTTCGA
BPUM_3689	*yycF*	BPUM_3230	*tagD*	GTCTTTACATTTTCTTAACCTTTCATA
BPUM_3689	*yycF*	BPUM_1851	*yocH*	ACAAAGTACATTTCCTTGACATTAGAT

Table 1: Binding sequence predictions for *Bacillus thuringiensis* Al Hakam. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BALH_0001	*dnaA*	BALH_0001	*dnaA*	TAGAATAGGTGTTAAAC
BALH_0041	*purR*	BALH_1417		ACAAAACTCGAACGAAAATGATGTTTTTATAAAATTTGTACGTGTTTTATG
BALH_0041	*purR*	BALH_0685		CAATTTTTTGCTTATAATATATATTGACAACACTTTTAATAAGCATTTGTC
BALH_0041	*purR*	BALH_0633		GAACATCTCACTTGCTACAAACTTACGCAAATGATTTATAAATTGATAAAG
BALH_0041	*purR*	BALH_4963	*guaC*	CCTTTGGACGAACATTTTTAAACAAACTTAATAAAATATTCGTATTTTAGG
BALH_0041	*purR*	BALH_4816	*glyA*	TGAAAACATGAAGGTTCGAAAGAATTACAACCTGAATTTTCGTTTTTTCTG
BALH_0041	*purR*	BALH_0282	*purE*	TGATTTTGTGCTTGCAATAACTAATCATTATTTTTTTGTAAGCTAATAAAC
BALH_0041	*purR*	BALH_0264		TCAAATTATGCTTGCTTTAATATTTTCAATTATTTTTACAAGCATAAATTT
BALH_0041	*purR*	BALH_3701		GCTAAAAATGCATAAATATTTATTAAGGTAAATATTCATTTGCAATAGTAA
BALH_0041	*purR*	BALH_3522		CATAAGAAAACTGAGATACTGACAACAGTTAAATAATGATCCTTTAACACA
BALH_0093	*sigH*	BALH_3886	*dnaG*	TAAAGGAATTTTTTAATTTATATCGAATACAAAATAC
BALH_0093	*sigH*	BALH_3780	*spo0A*	AAAGGGAATTTTCACACAATTGTCGAACAATTCATGT
BALH_0093	*sigH*	BALH_3537		AAAAGGGAAACGTATAAAGACGTTGAATATTTTTCTT
BALH_0093	*sigH*	BALH_1553	*fumC*	GAGGAATTTTTTTAAATAAATATAGAATATCTATATA
BALH_0093	*sigH*	BALH_0548	*aspA*	GAAGGAAAAGTGCTTATTAGTGAAGTGAATGATTTAT
BALH_0093	*sigH*	BALH_4685	*yfiA*	GGAAGAGTGAAATGGTTTAATGCAGAAAAGGGATTTG
BALH_0158	*gntR*	BALH_0158	*gntR*	ATACTTGTATACAAGTATACTT
BALH_0205		BALH_0463	*gltB*	ATATCGATATTACAT
BALH_0205		BALH_0205		TAGTGGAAAACATTA
BALH_0205		BALH_3149		ATATAAAATAAAGAT
BALH_0248		BALH_1252	*alsS*	AGCAAATAATCTTCTTAAGGCTTTCAATCAATA
BALH_0248		BALH_0784		AATTAATGGGTGTTCCAATATTTACAAATATAT
BALH_0248		BALH_1170		AGGAAAAACACTTTGTAACGTGTTTGAATAACT
BALH_0248		BALH_4532	*ldh*	AACATATACAGTTATTTATGTATTATTTATAAA
BALH_0248		BALH_4435	*ldh*	GTATTAAACACTTCATTAAGTGTTATTATGCAC
BALH_0248		BALH_4363	*cydA*	AATAAGATATTTTCAGATAATCGACATAATTCT
BALH_0248		BALH_1722	*cydA*	TTTGTGATGACTTTATCACTTTTCCATTTAAAA
BALH_0248		BALH_1702	*ldh*	ACAGATACATTTTCAAATACTTGAATATCTTTT
BALH_0318		BALH_1015	*rocD*	AATGATATAATATTTAGAG
BALH_0318		BALH_0317	*gabT*	AGTGAAAAGTTATTTTCCC
BALH_0318		BALH_0301	*rocA*	TCGTTTTTGCAAAACCTTA
BALH_0318		BALH_3195		GTGTTTACTTTAAACTTAA
BALH_0318		BALH_1347	*gdhA*	AATACACAATTACTTTGTG
BALH_0318		BALH_0771		GAGTAGTGTAAAAAGGTGG
BALH_0318		BALH_0427	*rocR*	TCGTTTTTTTAAAACGCCA
BALH_0318		BALH_3777		TATGCGTAAATATTTTGAT
BALH_0318		BALH_2657		ACGTTTTTTTTCAATTTCT
BALH_0318		BALH_2545		ACATTTTTATAAAATTTAA
BALH_0318		BALH_2482	*acoR*	AAAACAAAATTATATTTTA
BALH_0318		BALH_2061	*rocR*	ACGTCTTCTTAATCGTCGA
BALH_0469	*perR*	BALH_1231	*dltA*	TACTTCAATAATCTTGTTTAA
BALH_0469	*perR*	BALH_0766		AATACTAAAAATGTATGTTTT
BALH_0469	*perR*	BALH_1049	*spxA*	TCTATTGTATAATGATTATAG
BALH_0469	*perR*	BALH_0469	*perR*	AATATTCTTAATATTTTATTA
BALH_0469	*perR*	BALH_0410		TATCTTATTAATGTTGTATTA

BALH_0469	*perR*	BALH_0336	*ahpC*	AATCTTAATAATATCAATTAA
BALH_0469	*perR*	BALH_3709		ACTTTATGAGAATAATTATCA
BALH_0469	*perR*	BALH_4059	*hemA*	ATTAATTTATAATCATTATAA
BALH_0469	*perR*	BALH_2710		ACAATTTTAGGATGATTATAT
BALH_0567		BALH_0568	*treP*	CAACTGATTGAATATATATGCTCAATTATACA
BALH_0832	*citT*	BALH_0501	*citN*	TTTGGAAAAAGACTTAAAAATAAAATAAAACTTTCGCATTAGTAAAAACTC
BALH_0889		BALH_3195		TAAGAGAATGTAATGTATAAAAAATGTTAACATTGTGT
BALH_0889		BALH_2332		TAGTGTTGTTATATGGGGAAAACCAAAGAGAAATTTAG
BALH_0889		BALH_2256		AAAAGAAGTAATATGTGTGAACTAAGATTTAACTTTTG
BALH_0889		BALH_1783		AAAGGTTTATGTAGGGGTTTATTTATAAACAATTTAAG
BALH_0889		BALH_1773		GGATTAATAGATTTATTGAAAGAATAACAATTACTAAA
BALH_0889		BALH_1767	*nadE*	ATTTTAAGGATTAAGTGCGAAAGAGCAAAAATTTTTGT
BALH_0889		BALH_2184	*relA*	GTGAAAATAAATAAAGTATTAATTTTATTAAATTTTCC
BALH_0889		BALH_2120		GTTTTTATTTTTTCTAAAAAAGGAGAATACAGATGGAG
BALH_0889		BALH_1252	*alsS*	GATCTGAGTAAATAGGGCGAAATTACCCGTCATTCTGG
BALH_0889		BALH_0784		TTTTTTGTGTATTTTTAGAAGAGGTATAAAAATATTAT
BALH_0889		BALH_0771		AAAAGTGTTTATAAAGTAAACTTTTTACAATAGTTTGG
BALH_0889		BALH_0766		ATGTTTACGTCCTAATAAAAAGGGTATCTATTAATTAA
BALH_0889		BALH_0686		GTTTTTATTATGCACATTATAAGAAAAAGGAGTTGAAA
BALH_0889		BALH_1094		TAAAATAAAAAGAGAGGAAAGTTATAATATAACTGTTT
BALH_0889		BALH_4973		TTTGTGTAATTTTACAAAATTGAGTAGTTATTTTAAAA
BALH_0889		BALH_4936		TCATTACAATTATTAGCTTTTGGATATTATTAAAATTC
BALH_0889		BALH_0579		ATATTCTTGATAAAAAGGAGGGGGAATATTGGAAGCTT
BALH_0889		BALH_4779	*gtaB*	GTATTTAATCATTAGCGGGAAAGCAATTTATAAAGAAG
BALH_0889		BALH_0301	*rocA*	AAAAATAATATAAAATTTTTAATTAATTAAAAATTTTA
BALH_0889		BALH_4685	*yfiA*	AATATAAGCAATAAGTGCTACAATTATCCTATTTTATT
BALH_0889		BALH_4642	*clpP*	GATGTTGAAGATACATAAGCAGGGAATTTCAAAAACAT
BALH_0889		BALH_4593		ATCATACATGTAAACGATTTTTCTCCTCAGTACTTTGG
BALH_0889		BALH_4592		GTTTTTTTATTGTAGTAAATGATGAAAAGAGTGAGAGG
BALH_0889		BALH_4583		TTTTTTAAAATGTAAATAATTCTAAAAATATTATTATG
BALH_0889		BALH_0084		CGTTTTGATTAGCGAAACAATGGTTAATAATGAGTAGG
BALH_0889		BALH_3988	*relA*	TGTTTTAACAATCCATAAAGATGCAATTAAGCCAGGCC
BALH_0889		BALH_4390		TTCTTTCCATTACAAAAAACAGGGAGAGATGAAAATGA
BALH_0889		BALH_3807		TGTTTAAATACTGCGAAATTTGTCTATAAAAAGAATAA
BALH_0889		BALH_4194		AAAAAAAGAAAATCGGTGACGTAATTTAAAAGATTAGA
BALH_0889		BALH_4173	*phoP*	GGTTTCAAGTCTATAACTTGGAGGAAAAGAATGAACAA
BALH_0889		BALH_2819		TGATTTTATATAGATTATGATAGATAAATTTTGGCATG
BALH_0889		BALH_3280		GCTTATTTTTATTAAAAAAATAGGTATTTTATTTTCTG
BALH_0889		BALH_3279		ATTTAGTGCAAAATAGGGTGAAATTACCCGTCATTTGG
BALH_0889		BALH_2710		GCTATAAGTAAAAAGGTATTAAAGAAGATAAAGTTGTC
BALH_0900		BALH_1152		TTTTCAATAATACGACGCTACTAGCACATAGGTTACCCG
BALH_0916	*glpP*	BALH_0917	*glpF*	GACGGAGAAAAGTAGAGATC
BALH_0916	*glpP*	BALH_0602	*glpT*	CACGGAGAATCGGAGTACAC
BALH_0933		BALH_0628		ATTCATTTTAAAGAAAAAAG
BALH_0933		BALH_0551	*ptr*	ATAAAGGTATATTAAGATAA
BALH_0933		BALH_0539	*nprE*	GATATAAGCATTTTGAAGAA
BALH_0933		BALH_4578	*nprB*	ATAATGTTTAAACCTAAAAT
BALH_0933		BALH_4378		ATATATATAAATAAATATAT
BALH_0933		BALH_2915	*npr*	ATATAGAGTTATTAAAATAT
BALH_1296		BALH_4912		TTATATCAATGTTATGAATAAGTACAA
BALH_1333	*resD*	BALH_4363	*cydA*	GAAGGTGTTTAAAGAGT
BALH_1333	*resD*	BALH_1907	*nirB*	AAGTGTTTTTTATATGT
BALH_1333	*resD*	BALH_1892		GTAAGTTTTTGAAAAGT
BALH_1333	*resD*	BALH_1722	*cydA*	TTTGAATTTTTAAAATA
BALH_1333	*resD*	BALH_1304		TTAGATATTATTGTAAA
BALH_1584		BALH_0436	*glsA*	GTAGTTTTTATTTTCGAATTTTTTATGGATAAATCTCCTATTTC
BALH_1672	*deoR*	BALH_1673		AATGAACAAAATTTCAAAAGTGTATTTACATTTGTTCAA
BALH_1892		BALH_1901	*narK*	GAGGATGACATCATTCAAACT

BALH_1892		BALH_1887		AACTGTGCGAAATATCACATC
BALH_2255		BALH_1252	*alsS*	TCTTTCGCAACTCT
BALH_2255		BALH_0784		ACATTCGCATAAGA
BALH_2255		BALH_4215	*ackA*	TACTTTGCAAAAGT
BALH_2255		BALH_1288	*gltT*	TCCTTAGCAAGAGT
BALH_2255		BALH_1243	*gltT*	TCTTTTGTAACAGT
BALH_2255		BALH_0790	*fadD*	TCTTTCATAAATGT
BALH_2255		BALH_0771		TCTTTAGCAAAAGT
BALH_2255		BALH_1147	*potA*	AGTGAACGCTTGCT
BALH_2255		BALH_0568	*treP*	ACATTTGCCATAGT
BALH_2255		BALH_0501	*citN*	ACTTTCGCATTAGT
BALH_2255		BALH_4886	*eutD*	AGACAAGGCTTAAA
BALH_2255		BALH_0427	*rocR*	TATAACCGTCTTCA
BALH_2255		BALH_0301	*rocA*	ACATTTGCAAAAAG
BALH_2255		BALH_4634	*rpoN*	ACCTTACCCACAGA
BALH_2255		BALH_0158	*gntR*	TAAAAGCGCTTGCA
BALH_2255		BALH_4418	*menE*	TCTTTCACAAAAAA
BALH_2255		BALH_4363	*cydA*	GGATATCGCTTTAT
BALH_2255		BALH_4243	*acuA*	TCCTTCGCAAAGAT
BALH_2255		BALH_4242	*acsA*	TAGAAACGCTTCCT
BALH_2255		BALH_4223	*acsA*	ACTTTTTAAACAGT
BALH_2255		BALH_3777		ATGTTCGCCAGAGT
BALH_2255		BALH_4178	*citZ*	ACCCTCTTAAAAGT
BALH_2255		BALH_4110	*lcfA*	ACCTTCACGATAAT
BALH_2255		BALH_3642	*fadA*	ACATTCGCCCCTTA
BALH_2255		BALH_3636		ACTTTTGCAAAATA
BALH_2255		BALH_3225	*dctA*	ACCTTCGCTAAAAT
BALH_2255		BALH_3195		AATAAACGCTGTCT
BALH_2255		BALH_2657		CATTTTGCATTTGA
BALH_2255		BALH_2656	*caiC*	GGAATCCGCTGAAA
BALH_2255		BALH_3074		ACCTTCGCGAACAT
BALH_2255		BALH_2545		AATTTAGGCAATGA
BALH_2255		BALH_2494		ACTTTCTAAATAGA
BALH_2255		BALH_2482	*acoR*	TGAAAACGCTTTTA
BALH_2255		BALH_1984		ACTTTAGCAAAAAA
BALH_2255		BALH_1722	*cydA*	TGGAAACGCTCGAA
BALH_2255		BALH_1673		AGAATACGTTTTAT
BALH_2255		BALH_2093	*mmgD*	TGTAAAGGCTTACA
BALH_2255		BALH_2061	*rocR*	AACTTCACATAGGT
BALH_2255		BALH_1581	*gltT*	ACTTTCGCAAGCGT
BALH_2255		BALH_1523		ACTTTCGTCCAATT
BALH_2255		BALH_0970	*fadD*	ATTTTACGCATACA
BALH_2255		BALH_0917	*glpF*	TGACACCGCTTTCA
BALH_2657		BALH_2494		TCAAATTAGATGAACAAAAGTGGACAAAACGAGACTAATGTCTCA TTTTGTCCACTTTTTTATT
BALH_3081		BALH_3081		AGAGAAAATCGGCAGGAAATAGTGAGAACAGCAGATATTCTTACA TATAATGAA
BALH_3328		BALH_4187	*dnaE*	TCTTTTTTCATACAAGAGAATG
BALH_3328		BALH_4104	*uvrC*	TTTTCTTGCATACAACCCACAA
BALH_3328		BALH_4000	*ruvA*	ATCTCTTGTAAACAACCCCATC
BALH_3328		BALH_3407	*recA*	AAAATCGAATAAATGTTCGATT
BALH_3328		BALH_2929		AAATAAGAACAAATGTTCTCAT
BALH_3328		BALH_3328		CAAGCTTGAATACAAACATCTT
BALH_3328		BALH_3238		TATCTTTGTATACAAGACTAAA
BALH_3328		BALH_3041		TTTGCTTGCTAACTAATAATAT
BALH_3328		BALH_1820		TTTAAACAACAATCATACGCTT
BALH_3328		BALH_2051		TAAGATTGTAAACCAGATTTAG
BALH_3328		BALH_0903		GTATGTTAGGAAAAAGAAAATT
BALH_3328		BALH_1085		TTACCTTGCGTAAACGCATGAT

BALH_3328		BALH_0575	*vpr*	TTTTCTTGATAGCACAAAGAAA
BALH_3328		BALH_0547		AACGTTAATAAACAAGCTTAAT
BALH_3328		BALH_4665	*uvrB*	ATGGCTTATAGACAAGCAAAAA
BALH_3328		BALH_0297	*pcrA*	ATGGCTATAAAATAAATCTTAT
BALH_3328		BALH_3942	*vpr*	ATCGGTTGTAAGCAAAAATATA
BALH_3459		BALH_1252	*alsS*	ACAACTTTTTATAACATAAGAAAGC
BALH_3459		BALH_1231	*dltA*	ATTAACGATTCAATCCCTAAAAGAA
BALH_3459		BALH_0784		CAAGGTACTGTAAACATTTTGGTAC
BALH_3485		BALH_1603		AATCTATATTGACATTTTTATAACCTGATATTAGTATCAGGTTATAAAA
BALH_3485		BALH_1036		TCTAGACAAAAATACATTTATGTTTTAAAATTAGTACCAAGTCATAATA
BALH_3485		BALH_0207	*plsC*	TACACTTTTGTAAGAATTGAAATGTAACAATTAATACTTAAATTTATTA
BALH_3522		BALH_3522		ATAATGATCCTTTAACACAGCCCCGTGAGGTTGAGAAGGTAACGGTTTGAAATA
BALH_3534		BALH_3336	*spoVK*	TAGTCTTTTGTACAAAGTGTAGCCCATATATTACTGGAAG
BALH_3534		BALH_3248		TAATCATAATACAGTGACATCAAAAATAAAATAAAAATAG
BALH_3534		BALH_2786		TAGTTGTACAAACTTTAACCACTGAATAGATTTAATATAG
BALH_3534		BALH_2766		TAATCATATTACTTAAAAATTATGCGTAAGAATATAGTAA
BALH_3534		BALH_2686		ATAAATTATAGAATAATTAAATGGTACTTTAAATAACAAA
BALH_3534		BALH_1984		TGGAAGTAATTTAGATGGTACGAGAATAAAAAAAGAAGAA
BALH_3534		BALH_2384	*pbpB*	CAAAAGTTTTACATACGTAAGCATCGCTTTGCTTAAGAAA
BALH_3534		BALH_1827		ACGGTCTATTCTCTAGAGCTTGTACATAACTTGAAATAGA
BALH_3534		BALH_2288	*bcd*	TAGGAATTTTTTTGTGCATTAGTACAGAATCTGTTTTAAA
BALH_3534		BALH_1782		TACGTCTAATTGTCCAAGCTCATACATATGCATAATAGTA
BALH_3534		BALH_2096	*mmgC*	TTATCATTTCATTTACGTGGAAATGAACGCTTATACTGAC
BALH_3534		BALH_2013		AGTGCCTAGTTTTCATGTTTTCATAAAAAATTGTTAAAAA
BALH_3534		BALH_1539		ATCACATGTAAAATAAAGCAATTGGATGATAAATTTTAAT
BALH_3534		BALH_1363		AAAATTATCAGTATACTTTAACCTGTTTTAATATACTATA
BALH_3534		BALH_0787		TTGTACTATCTCCTTTACTAAGTGTATATGTATATAGTAA
BALH_3534		BALH_0701	*spoVR*	TAAGCACACCTGCACAAATTGTTGTATATATAAATGAAGA
BALH_3534		BALH_1017		GTAATGGCTGACATACTCGCGCCCCGACCAATATATTTTA
BALH_3534		BALH_4973		ATTATAGGAAAAATAAACACATTAAAATGTTTTAACTCAT
BALH_3534		BALH_0490		AATAAAAATGTTATACGTATACTATCTCATTGGTTGGTAA
BALH_3534		BALH_0489	*prkA*	AAAGAATGTTTCAGGACATCCTAGCATACCTTTTATAATA
BALH_3534		BALH_4791	*spoIID*	TTTAAGATATGAATACGTGCATGGTCACTTTTATAAATCT
BALH_3534		BALH_0147	*cda*	TTGTTCTCTTTTTTATATTTACAGCATATATGTAACTGAA
BALH_3534		BALH_0143	*cwlD*	CAAGCATAAACTTTCCTCTTGTCCCATATAAGTAATTAGA
BALH_3534		BALH_4432	*glgB*	TTAAACATTTTTAAGAAAAAAACGTATATTTTTTATAAAT
BALH_3534		BALH_3994	*spoVB*	CTTGTCATTCTTGTCTCGTACGTGCATATAAATTATTGTA
BALH_3534		BALH_3935		TATACAAATTATAACTGATATTTACATAAGTTAACAAAAA
BALH_3534		BALH_3801	*spoIIIAA*	TACTTCTGAAATAGACAACCTCTGCATATGGTTGTACAAA
BALH_3534		BALH_4180		TTTCACCATCCAAATGTAAGCATGTATACATTTTAAAAAA
BALH_3534		BALH_4173	*phoP*	GAACAAAAGATCATACTACTCTTATTGCCCCTTTTATGAT
BALH_3534		BALH_3642	*fadA*	GATAAAAATAATATAATTTATAAAACTAAAAAATCATACA
BALH_3534		BALH_4074	*gerM*	GTGGCAATTTTGAAAGTGAAATGGCATATGAAGTTGTAAG
BALH_3534		BALH_3574		GCGCAAAATAAGATATGTATTACCTTACTTTTATAAAGCG
BALH_3534		BALH_3557		TGGTCTAAATTCACTACGTATCCACGTATACTGAAATGAA
BALH_3604		BALH_2093	*mmgD*	TTTATTTATTTAAAA
BALH_3604		BALH_4178	*citZ*	TGTATTTTGTTAAAA
BALH_3604		BALH_3253		ATTATTTACTTATTT
BALH_3690		BALH_1310	*pbp2A*	AGTAATTGTAAAGTCTTTTAAATAGAAAAAAACTAA
BALH_3690		BALH_0766		AATACTAAAAATGTATGTTTTTGCTATTATGTACAA
BALH_3690		BALH_0698		CAAGTATAAATTCCCGTCTTTCCCAAAAACTAACAC
BALH_3690		BALH_0644	*gerLA*	TTGTATATATTTTCTTCTATTAGCGGAATCTAACTA
BALH_3690		BALH_0572	*gerKA*	AAGCATAAATTTCTCACAAAAAGCAAAAATTAACAG
BALH_3690		BALH_4826	*spoIIR*	ACACCTTACTCTATTTTTTTGTGGAAAAATTCATAT
BALH_3690		BALH_4787	*spoIIQ*	AATGTCTAAAACAGCTTTTCATTAATCAATATGCAA

BALH_3690		BALH_0355		AAAGTTATTTCTTTTTTGTGTATAAAATATTCCAAA
BALH_3690		BALH_0200	*glcU*	GAATTTTAACTTTTATTTAAATATACATTCTATTTG
BALH_3690		BALH_4462		ACAAAATTAGAGGTTACGTTAACCTCTAATAAAAAA
BALH_3690		BALH_3910	*gpr*	AAGAATGACCTTATATAACTTTGGGAAATCTAACGA
BALH_3690		BALH_4359		AACATTAACTACTTTTGTTATCTTTAATTTTCCTAC
BALH_3690		BALH_4307	*gerHA*	AATGAATAATATACAAAATTAGCCACAAAATAGCAC
BALH_3690		BALH_3877	*nfo*	TCATATCAGACAAATTTATCAGCATATTTAAATACA
BALH_3690		BALH_3810	*splB*	CCGAAAATTACAAGGAAGTTAGTTAAAAAGAAAAAC
BALH_3690		BALH_3807		CATGTTTAAATACTGCGAAATTTGTCTATAAAAAGA
BALH_3690		BALH_3693	*dacF*	TGGTATTAAAAAATAATAAATTGGAAAAAATAGGTT
BALH_3690		BALH_3687		AATAATCAAACTTTTTTACCTTACGTGTTTATGTAA
BALH_3690		BALH_2801	*gerIA*	TTTAATTATATAAGAAAATGACTTTAAAAATTCTTT
BALH_3690		BALH_3213	*gerKA*	ACACTATAATTATAGCGTTAAAGTTAAAAATATACA
BALH_3690		BALH_2792		ATTAATAATAAAGTATTAAACATCGTATAATTTTTG
BALH_3690		BALH_2710		CATGTATAAATTCATTTTTAAGATTGAAATTTTAAT
BALH_3690		BALH_2473	*sleB*	GAAAATAGGAAAAACATATAACAAATGTATAAAAAA
BALH_3690		BALH_1760		CATGTATTTAAATTAAAATTTAACACAATTTATATA
BALH_3690		BALH_2087	*pbp1A*	AAATATTAGACACTTAATCCTTTTATAACTTACGAG
BALH_3690		BALH_2003	*yvaB*	AAACAAAATAAATCATTAACTTACGAAAAAAAAGTA
BALH_3690		BALH_1401		ATTATTAAATTTACACTTTTTGATGAACTTTATTTA
BALH_3690		BALH_0949	*pbpF*	ACATAAAAGAAATATTTTGTATAGAAAAAGTAATAA
BALH_3709		BALH_3317		TTAACGAACAAATGTTAAAAATGATGATGATGCAAAT
BALH_3709		BALH_1731		TTTTAAATTAGTTATTTATAATTCACATAGAATAATA
BALH_3709		BALH_2172		ACATATAGATGAAAAACCGTATTCTTTTCATTAAGAA
BALH_3709		BALH_2107		TTTTTACATCGATAATGATAATCATTATCATTATTTG
BALH_3709		BALH_0772		TGATTGTATGGGGTTTGATAACATTTTTCAAATGAAA
BALH_3709		BALH_0554	*fecC*	TATGCCAACTTTTATTAAGAGTAAAGGCAACTGTTAC
BALH_3709		BALH_0343	*trxB*	TTGTTTTACTATTACTAATAGTGATAACTAAGAAATT
BALH_3709		BALH_4675		TTAATAAACTATTACCAATAGTAAATCTCTCAGAAAA
BALH_3709		BALH_4659		GATAATAACTAATACCATTACTGAAACAATTTCCACG
BALH_3709		BALH_4465	*trxB*	TTTATATAATAGTAATAAAAAATAATACCAAGTATCA
BALH_3709		BALH_4042		AAGAAAAAGAAGAAAAAATAAACATTTTTATAAAGAG
BALH_3709		BALH_3365	*feuA*	ATATATACTTAGTAACGATAATCATTATCAATGAAAA
BALH_3777		BALH_3776	*ptb*	TTATACTCACACGTTTTTTATCGTACGTTATTTA
BALH_3780	*spo0A*	BALH_0060	*spoIIE*	TTTTGACAAAAATC
BALH_3780	*spo0A*	BALH_3535		TCTAATACAGATCT
BALH_3780	*spo0A*	BALH_1231	*dltA*	TTTTGACTTATTTT
BALH_3783	*argR*	BALH_3195		TCCTAAAAAATATTAA
BALH_3783	*argR*	BALH_0771		TTGAATAAAAATATCT
BALH_3783	*argR*	BALH_1015	*rocD*	AAGAATAAAATTAAAC
BALH_3783	*argR*	BALH_0317	*gabT*	AAAGTAAATCTACGTA
BALH_3783	*argR*	BALH_0301	*rocA*	AACGTAAAAATAATTA
BALH_3783	*argR*	BALH_3747	*argC*	AATTAAAAAGTAAGTA
BALH_3809		BALH_1657		AATTTGCACTAAGGAAACA
BALH_3885	*sigA*	BALH_1095		TGTGTATAAAAAATTAAATAGATGAAATCGTTTAA
BALH_3885	*sigA*	BALH_1094		TATGGATTTTTCAGAATAATCATGATAGATTATAA
BALH_3885	*sigA*	BALH_1051		ATTTCCATTCTGGAGAATCTTATCATAAAATGAGA
BALH_3885	*sigA*	BALH_1049	*spxA*	TATAGACATGAATGAGAATGTTTGTTAAAATAAGA
BALH_3885	*sigA*	BALH_1036		TCTAGACAAAAATACATTTATGTTTTAAAATTAGT
BALH_3885	*sigA*	BALH_1032		TATTGTATAAAAAAGTCAGTTAGCGTATAATCAAA
BALH_3885	*sigA*	BALH_1020		AGTTGATTTTTCTTTTCACTTCGAATATGATATGA
BALH_3885	*sigA*	BALH_1015	*rocD*	GTTTCGCTTTTTTACATTTAAATGATATAATATTT
BALH_3885	*sigA*	BALH_4973		ATTTTACAAAATTGAGTAGTTATTTTAAAATAGAA
BALH_3885	*sigA*	BALH_4968		GTATTGTTATTATCTTTCAATCGAGAATAATGTCT
BALH_3885	*sigA*	BALH_4958	*galE*	GATGTGCATTGAGATTATAAAATTTTATAATTAAT
BALH_3885	*sigA*	BALH_0568	*treP*	TGTTGACTAACTTATATATACGAGTTAATATGTAA
BALH_3885	*sigA*	BALH_4912		GTGGTAATATAGTTACAATACTTATTCATGTTTCA
BALH_3885	*sigA*	BALH_4907	*cysD*	AATTGACATTAGGAAATTATTGCTATAAAATCAAC

BALH_3885	*sigA*	BALH_0548	*aspA*	GATTTTCTATAAAAGTTATATATAGAAAATTAAAT
BALH_3885	*sigA*	BALH_0532	*ald*	CGTTTTCATAACTTTTAAGTTATGCTAGAATAAGG
BALH_3885	*sigA*	BALH_0523		TTTGTTTCTTTATCAGTAAGAAAGATAAAATAAAT
BALH_3885	*sigA*	BALH_0501	*citN*	TATTTACCATTCAAATAGATGCTTTTATGATAAAC
BALH_3885	*sigA*	BALH_4886	*eutD*	GCTTTATTTCTATGCTGAATTTTATGAAAATATAA
BALH_3885	*sigA*	BALH_4867	*speE*	ACTATACATTTTGGATGTTTTGTTTTATTTTTGTT
BALH_3885	*sigA*	BALH_4854	*map*	CTTTTTTATGTTTAAATGTTTTAGAAACACTATTT
BALH_3885	*sigA*	BALH_4844	*fdx*	CGACTAAAATAGCAAAAAATGTTACTTCTTGTTTT
BALH_3885	*sigA*	BALH_4836	*pyrG*	AATTTTCTTTTTTGTTTTTTGTTTAGAAAATAAAA
BALH_3885	*sigA*	BALH_0469	*perR*	AGTACACTCTTTATAAGAATTATAAAATAATAATC
BALH_3885	*sigA*	BALH_0463	*gltB*	TATTTCATAAATTCCTTATTAATAGTATAATTATT
BALH_3885	*sigA*	BALH_4816	*glyA*	TCTTTTATGATATAGAAAGTCATGTTAGAATGTAG
BALH_3885	*sigA*	BALH_0438	*nagE*	TATTTATTATTGTACTTTTTCACCTTACTATTTTG
BALH_3885	*sigA*	BALH_0436	*glsA*	GTTTTATCATTCCACTTTTTCATGTTATTATTTAT
BALH_3885	*sigA*	BALH_0427	*rocR*	TTTTGCGGTTTTATAAGAATTTTAATAAAATTATA
BALH_3885	*sigA*	BALH_0410		GATATGATATCTTATTAATGTTGTATTATTGACAA
BALH_3885	*sigA*	BALH_4779	*gtaB*	TTTTGTTATTGTTACATATAAATAAAAGGATATAG
BALH_3885	*sigA*	BALH_4768	*lytR*	TTATCATTTATTAGTATTGCTATAATATAATGAAG
BALH_3885	*sigA*	BALH_4766	*galE*	AAAATATTATTCTCTCACTTATTCTTTAGGATTAT
BALH_3885	*sigA*	BALH_4765	*manR*	TTTATGTAAAAATGAACAAATTCGAAACAATAAAA
BALH_3885	*sigA*	BALH_0337	*mtrK*	AAGATAAAATGTTTCTTAACACACTATTTATTTTA
BALH_3885	*sigA*	BALH_0336	*ahpC*	ATTTTATTTATCACACAATTCTTTGTAAAATAGAA
BALH_3885	*sigA*	BALH_0333	*amhX*	AATTGACTTTTTTCTTAAAATTCAATATGTTCATA
BALH_3885	*sigA*	BALH_0317	*gabT*	TTAATAAAAATTTTGTCATAGTTAATCTCAGAAAA
BALH_3885	*sigA*	BALH_0301	*rocA*	AAAATAATATAAAATTTTTAATTAATTAAAAATTT
BALH_3885	*sigA*	BALH_4688	*comFA*	TTTTTACGAGATTATAAGAAAGGTTTATAGTAATG
BALH_3885	*sigA*	BALH_4684	*secA*	AAAATAAAATTATTTTTCTCCTCGCATAAAGGATA
BALH_3885	*sigA*	BALH_4670		AATGTAATATTCTCTCAAAGATATTATTCAAATTA
BALH_3885	*sigA*	BALH_4642	*clpP*	GTTTGACCTTTATTGACCATAATTGTATTATAAGA
BALH_3885	*sigA*	BALH_0282	*purE*	ATTTCTACTACAAGTATATATTTTAAATAATGAGA
BALH_3885	*sigA*	BALH_4632	*cggR*	AGTTGAATAAGTTTCTATCTCCTGTTACAATAATA
BALH_3885	*sigA*	BALH_0254	*guaA*	TTTTAGAAAAGGAAATGAATTTCTGTAGAATTTTG
BALH_3885	*sigA*	BALH_4603	*tyrS*	TGTTGACAATTTGTTTTCAATTCAATATAATGGCT
BALH_3885	*sigA*	BALH_0205		ATTCAAAATGATATTAACTATCATTTATAATGATA
BALH_3885	*sigA*	BALH_4547	*fadB*	TATTGCGAAAATTTAAACAGTATCATATATTTATT
BALH_3885	*sigA*	BALH_4532	*ldh*	TATGTATTATTTATAAAAAGGTCAACATTGTTTAT
BALH_3885	*sigA*	BALH_0162	*gntZ*	TATATGATATTATGTCTCATTGTATGTACCTTTCC
BALH_3885	*sigA*	BALH_0158	*gntR*	GTAATAATATTTTTAGTATAATTTTTTGAAGTCCT
BALH_3885	*sigA*	BALH_0100	*rpoB*	CCTATAATATAGTATTTTAGTTTTTTTGCAACTGC
BALH_3885	*sigA*	BALH_4499	*tipA*	TAGATAATACGTACTTTATAAAATATTTCTTTTAC
BALH_3885	*sigA*	BALH_0093	*sigH*	AAAATGACATATTATAAAGATTTATTTATCGCCAG
BALH_3885	*sigA*	BALH_4435	*ldh*	ATCATAAAATTAATTAATGAATTTGTTAGTATTTT
BALH_3885	*sigA*	BALH_4422	*menF*	TGATTCACATCTTTTTCGTAGTAAATCACAGTTGC
BALH_3885	*sigA*	BALH_0060	*spoIIE*	TGTTGACTTTAAATTATATCTGAGGTAAGATACTA
BALH_3885	*sigA*	BALH_4419	*menB*	ATGTGGAACAACCGGAAAAGTTTGATACAATAGTA
BALH_3885	*sigA*	BALH_4418	*menE*	ACTACAGTATAGTTCAGAGAAAGAAAGTACTTTCC
BALH_3885	*sigA*	BALH_0044	*glmU*	GTAAAACATCTTATAAGAATTATGGTAAAATTTAA
BALH_3885	*sigA*	BALH_0018		AAGATAAAAATTTATCGGATTATTTCTTAAATTTC
BALH_3885	*sigA*	BALH_0001	*dnaA*	CATTGCTATAGCTACTTTTTTTTGATATTATAGTT
BALH_3885	*sigA*	BALH_3980		AATTTCTTATAATACTATATTATGCTATAATTTCA
BALH_3885	*sigA*	BALH_3960		GATATCACATTGTTTCTTTTCTTCCTTTATGGTAC
BALH_3885	*sigA*	BALH_3935		TGTTTAATATTGACTATAAATGTATTCAATTGTTT
BALH_3885	*sigA*	BALH_3908	*lepA*	CATTGAATCTTTGCTGCTCTATTGATATAATCAGT
BALH_3885	*sigA*	BALH_3906	*hrcA*	GTTGACAATTGAATGTCAATTTTGGTAAGTTATTA
BALH_3885	*sigA*	BALH_4378		ATAATTTTATAGTTTAACTTGATAATTTTTTTTAC
BALH_3885	*sigA*	BALH_4363	*cydA*	AAAGTGATATTTGATTAACATCTATTACATTTTTT
BALH_3885	*sigA*	BALH_4359		GAAATAAAATGTATTAAAACATTAACTACTTTTGT
BALH_3885	*sigA*	BALH_4341	*pckA*	TTTATCATATGTAATAAATATATTAACACAATATG

BALH_3885	*sigA*	BALH_3852	*metC*	GAAATAATAAAATAAATACATCGATTTTCCTTTAC
BALH_3885	*sigA*	BALH_3843		ATTTACAATAGTAAGAAAAATAAACTATAATGAAG
BALH_3885	*sigA*	BALH_3841	*comGA*	AAAGTCGTATACTTGTAGAAGATTAAATCTATTTA
BALH_3885	*sigA*	BALH_4264	*ftsK*	ATATGAAAACAATTGTAAATTAGAATATTATGGAT
BALH_3885	*sigA*	BALH_4243	*acuA*	TAGATAATATTATTGTAATAAATTTTAAAAGTCAT
BALH_3885	*sigA*	BALH_4242	*acsA*	TACTGAAAATTTTAAATAATGTTATTATAATAGAT
BALH_3885	*sigA*	BALH_4238	*tyrS*	TTTGACACGCCTATATAAATGATGGTATAAAGAAA
BALH_3885	*sigA*	BALH_4233	*rpsD*	GGTAAAAGATATAATAGCGTTTGTGTAAAATAAAA
BALH_3885	*sigA*	BALH_4232	*megL*	AAAATAAAATGTGTTTGCGATAATATAGAAAATGG
BALH_3885	*sigA*	BALH_4227	*ezrA*	AGGTCAAAAACAAGCCATTACATGGTATGATTAGT
BALH_3885	*sigA*	BALH_4226		TGTGTAATTTTGCAGGCGATGTTGCTATAATGAAT
BALH_3885	*sigA*	BALH_4223	*acsA*	GGTTCAAAATAGACAGAAAATTCTTTATTATATAA
BALH_3885	*sigA*	BALH_4215	*ackA*	AATATACTATACGGTATAAAAATTTATATAGGTTC
BALH_3885	*sigA*	BALH_4204	*ald*	TTTGTCAAATATTGAGATATTATGTTAAAATATAA
BALH_3885	*sigA*	BALH_3780	*spo0A*	GCTCCGCAAGAGGTGTTTTTTGTTGTAAAATAAAA
BALH_3885	*sigA*	BALH_3777		TATTTTGTTTATTTTAATGATGTTATATGAAAAGG
BALH_3885	*sigA*	BALH_3747	*argC*	AATTGACTCTATAAATATACAATATTATAATCATA
BALH_3885	*sigA*	BALH_3709		ATTGGATTGCCACTTATTTTTATTTTATTATTTGC
BALH_3885	*sigA*	BALH_3705	*deoB*	TATTAAAAAGAAAGCGTAAACATGTTATGATATCA
BALH_3885	*sigA*	BALH_4178	*citZ*	TGTTCACAAAAGTCAGATAATTGTTTATAATAGGA
BALH_3885	*sigA*	BALH_4173	*phoP*	AAAATGATATAGCACTAAACTATTAAATTCAGTTA
BALH_3885	*sigA*	BALH_4166	*gap*	CTTGTAATAACTGATTTTTTAACACTTTAATTACT
BALH_3885	*sigA*	BALH_4165	*speD*	TGTTGCAAAATGAAAATAAACAAAGTATACTAACA
BALH_3885	*sigA*	BALH_4160	*thrS*	CATATAATTACCTACATAAATGAAATATATTGAAA
BALH_3885	*sigA*	BALH_4159	*infC*	TATTGCAAGTATGTTAGTTGTTTGTTACAATATTT
BALH_3885	*sigA*	BALH_4142	*pheS*	GGTTGCGAACCAATAAAAAACTTCTTATAATAAGT
BALH_3885	*sigA*	BALH_4110	*lcfA*	GATTGTAACGTTATGGTGACTATAATATAATGAAA
BALH_3885	*sigA*	BALH_4102	*sdhC*	CTAATAATATATTGTAAAGGGTAAATATTTTTGTC
BALH_3885	*sigA*	BALH_3666	*ptsG*	CATTGAATCGCTTACAATAGTTATGTATAATAACT
BALH_3885	*sigA*	BALH_3653	*mtnW*	AATTGACAACATGAAAAATATCTGACAAAATTCAG
BALH_3885	*sigA*	BALH_3652		TCTTTACAAAATACTGGAAAGATTATATCATTTGT
BALH_3885	*sigA*	BALH_3651		TGTTTACTATATTAGAAAGGTCATAAAACATTTCT
BALH_3885	*sigA*	BALH_3650	*mtnK*	TTTTTAAGATTTTTTAATATATTGAGAATAGTTCT
BALH_3885	*sigA*	BALH_3627	*kinB*	GTTTACATTCATTTCGAAAATCTGTAACAATTATT
BALH_3885	*sigA*	BALH_3611		TAAAGATTTCTATTAAATAAAATGATAAAATGAAT
BALH_3885	*sigA*	BALH_4059	*hemA*	TTAGTTATATTGCTTTTTATTTATGTTACAGTTTA
BALH_3885	*sigA*	BALH_4051	*valS*	GTGATAACATATATTATTAATTGTTAGTATATTTA
BALH_3885	*sigA*	BALH_4012	*nifS*	TGGATATAAATGTATCAAAATTTTATTACTGTTCT
BALH_3885	*sigA*	BALH_4011	*nadB*	TCTTGTCATTATTTTAAAACTATGTAAATATAGGT
BALH_3885	*sigA*	BALH_3595		AATGTATAATAAGATTCTCTTTTTTGTTCAATTTT
BALH_3885	*sigA*	BALH_3574		CTTTGCTATTTTATTATTTTTTATATATCATAGGA
BALH_3885	*sigA*	BALH_3537		TTGTTCCATATATTGAGTTGTATGGTATAAATAAA
BALH_3885	*sigA*	BALH_3535		TTTATATGTAAGTAAAAAAATATAGCATTCTATGA
BALH_3885	*sigA*	BALH_3526	*ileS*	GCTTGACGTTTTTCTCATTATTACATATAATTTTA
BALH_3885	*sigA*	BALH_3522		TTTATAAAATGCTTTTATCATAACTGTTTAGTATT
BALH_3885	*sigA*	BALH_3429		CCATCATTCTTACTGTAGAATATGATAAAATTAAC
BALH_3885	*sigA*	BALH_3422	*ftsK*	GAAGGATTTTGTTTAGAAGCTATGGTATAATATAA
BALH_3885	*sigA*	BALH_3407	*recA*	GTTGGCAAATTGAATTGAAAATAGGTATAATAAGA
BALH_3885	*sigA*	BALH_2995		CAAATATAATCATGGTAAATATTACTTAAAAGTTT
BALH_3885	*sigA*	BALH_2925		TATTTTTTTCTCATAAAAAGTCTGATAAATTAGCA
BALH_3885	*sigA*	BALH_3368	*adaA*	AAAGTACGATAGTGTTTTATCTTTTTGTTAGTATA
BALH_3885	*sigA*	BALH_3365	*feuA*	TCTTGAAATAATTGCATATTAAATATATACTTAGT
BALH_3885	*sigA*	BALH_3350	*gabP*	AATTGTTGAATTCAGAATATTTTGATATATTTTAA
BALH_3885	*sigA*	BALH_3300		ATTTGAGAAACTATGTTACCTTTGTTATGTTAATA
BALH_3885	*sigA*	BALH_2868		AATAAAAAAACGTTCCATATTTAAATTACAGTTAT
BALH_3885	*sigA*	BALH_3280		ACTTGAAAGAACAAAAAATGTTTGTTAAGGTAGGG
BALH_3885	*sigA*	BALH_3253		TATTTACTTATTTAGAAAGTTGTAATAAGATATTA

BALH_3885	*sigA*	BALH_3225	*dctA*	TATTTTCAAAAATAATATAATAGTGTAACATAAAG
BALH_3885	*sigA*	BALH_2797		AGATTAAAATAATTTAATTAATATTGTATCTTTTT
BALH_3885	*sigA*	BALH_2747	*lysP*	AATTGCCATATATTAAAAATCTTAGCAATATATAA
BALH_3885	*sigA*	BALH_3195		ATTTGAATTTTCAGTATTTGTTAATTATGATTTAA
BALH_3885	*sigA*	BALH_3152	*glpQ*	TATTCACAAATTTTAATAGGTTTGGTATGATTAAT
BALH_3885	*sigA*	BALH_3149		AAGGTAACATACTTTCAAGACTATAGTATATTTTA
BALH_3885	*sigA*	BALH_3112		TACATAAAATAACAATTAAGTCAATTTTTAATTTG
BALH_3885	*sigA*	BALH_2686		ATAATTAAATGGTACTTTAAATAACAAATTGTTAT
BALH_3885	*sigA*	BALH_2685		AATATCATATATTATTATAAATAATGTAGATGAAA
BALH_3885	*sigA*	BALH_2683	*rocC*	ATAGTTAAATTGTTTTTAAGCTTTACTTTAGTTAG
BALH_3885	*sigA*	BALH_2657		TTTATAAATTTTACTTATATATTATGTACGCTTTT
BALH_3885	*sigA*	BALH_2656	*caiC*	TATCGAAAAATAGAAAAATATGAAATATCATGAAG
BALH_3885	*sigA*	BALH_2614	*ssuB*	AGTTGACTTATCGGAAATATTTGTATAATATGTAG
BALH_3885	*sigA*	BALH_2602	*aspB*	CATTTTAATTTTCATAAAAAAGTGATAAAATGTAA
BALH_3885	*sigA*	BALH_3074		TTTAGTAAATTAATATTTAATTTGGTGAAATAAAA
BALH_3885	*sigA*	BALH_2545		ATTTTAAATTTATCGTAAATAAGTGTATTATTTTT
BALH_3885	*sigA*	BALH_2503	*opuAA*	AAACTAAAATTCTGTTATAGTAGAAATATCGCTAA
BALH_3885	*sigA*	BALH_2482	*acoR*	TATTGAAAACGCTTTTATTCAGTTTTATAATAAAG
BALH_3885	*sigA*	BALH_1940	*ileS*	ATATTTATTTTATATAAAACTTAATCAACTGTTAA
BALH_3885	*sigA*	BALH_1907	*nirB*	GGCTTAAAATTGTTAAACACTTAGTAAAGTTTTCA
BALH_3885	*sigA*	BALH_1901	*narK*	ATGATAATAAACGTGTAAATGTTCTTCTTCATTCA
BALH_3885	*sigA*	BALH_1892		AAAGTAAAAAAATAAACGGAAATAATATTTTTTCT
BALH_3885	*sigA*	BALH_1887		AAAGTATTATATGAGGAAAATAATAATAGAATTTT
BALH_3885	*sigA*	BALH_1856		TATTTATAAAATTTTCTAAATGTTATATCATTATT
BALH_3885	*sigA*	BALH_1767	*nadE*	TATTGTATATTCGTTAATTTTGCGAAATAATAAAG
BALH_3885	*sigA*	BALH_1722	*cydA*	AATATAATTTTTTTATGTAAAATCATTACTTTTTT
BALH_3885	*sigA*	BALH_1702	*ldh*	TTTTCAAATACTTGAATATCTTTTATAAAATGTAA
BALH_3885	*sigA*	BALH_2166		ATTTTCCAAATGATTAAATTCCTTATAAAATGAAA
BALH_3885	*sigA*	BALH_2125	*thrS*	AAAATAATACTTTACCTCTCTACTTTTTACTTTCT
BALH_3885	*sigA*	BALH_2107		TATATAATATAGGCTTTATGTATAATTTAGAATTT
BALH_3885	*sigA*	BALH_1673		TATTTACATTTGTTCAACTAAGAGTTAGAATGAAG
BALH_3885	*sigA*	BALH_1603		CATTATCAATTTTAGACACTTCTCGTATATTATTT
BALH_3885	*sigA*	BALH_2093	*mmgD*	ATAGTAATATTTTATTTAACTTTAGAAGATTTTTG
BALH_3885	*sigA*	BALH_2087	*pbp1A*	TATTAAAATTTCACTTTTTACATTCTATGCTTAAA
BALH_3885	*sigA*	BALH_2061	*rocR*	TTTATGAAAAGTTTATTAATTAAGTTAGAAGAAGA
BALH_3885	*sigA*	BALH_2040		AAATGTATGTGCGTGAATATGCTGCTATAATGAAA
BALH_3885	*sigA*	BALH_1589		TTTTGCTATATTTTTCACAAACTTTTAAGATAAAA
BALH_3885	*sigA*	BALH_1581	*gltT*	AAACTAATATGCTTTGTTAGACTAATGAATAGTTC
BALH_3885	*sigA*	BALH_1553	*fumC*	TCTTATCATTTAAGAGAAAGTCTGTTATTATAAGA
BALH_3885	*sigA*	BALH_1547		TTTGAATGAGCCTTCTTACAATTGGTATAATGACA
BALH_3885	*sigA*	BALH_1457		TAAAATACATAGCATAATTATCTGCAAAATTTTTC
BALH_3885	*sigA*	BALH_1417		TTAGTAAAATTGCTTCAATATTAAACCTTACTTTC
BALH_3885	*sigA*	BALH_1401		AATTTACACTTTTTGATGAACTTTATTTATTAGAT
BALH_3885	*sigA*	BALH_0972	*wapA*	TATTTGAAATAAAGTTAAATCTTCTCTCTTATTTC
BALH_3885	*sigA*	BALH_0970	*fadD*	ACTTGCCACTTTAATTTTCTTTTAATATAATTAAT
BALH_3885	*sigA*	BALH_0949	*pbpF*	TATTTTCTTTATAAAACATATCTTTTTCATTATTT
BALH_3885	*sigA*	BALH_0917	*glpF*	TGTTGACACCGCTTTCATTAACGGTTAACATTATA
BALH_3885	*sigA*	BALH_1376		ATAGTGTCCTAGTTTTTATTTTTTATATAATTTAT
BALH_3885	*sigA*	BALH_1347	*gdhA*	ACTTTTCTAAAAGCAATAAAAGGACTATAATGATA
BALH_3885	*sigA*	BALH_1333	*resD*	TTTTGGAAAAAAATTGAACAAACTTTATTATTCTT
BALH_3885	*sigA*	BALH_1310	*pbp2A*	TTTGTAAAATAGCTTTTCAAATGTTTTTGAATTGT
BALH_3885	*sigA*	BALH_1304		ATAATAACATTTATTTATATCTCCACTTTGCTTGT
BALH_3885	*sigA*	BALH_1288	*gltT*	CAAGGATTTATTTTTAGATTTTTTGTAAAATAAAG
BALH_3885	*sigA*	BALH_1276		GGATTGAAAAAAATCAGAATTGTGATAAAATACAA
BALH_3885	*sigA*	BALH_1252	*alsS*	AATAAAGTTGGATAACAACTTTTTATAACATAAGA
BALH_3885	*sigA*	BALH_1243	*gltT*	ATTGTCAAAATAGGGATTTTTTCGGTACAATTAAT
BALH_3885	*sigA*	BALH_1231	*dltA*	GATTTTCTTTTTTAGTATTTAGTGCTAATATCAGA
BALH_3885	*sigA*	BALH_0795		AAAGTTATATATTAAAGATATGTATAATGTAGTAT

BALH_3885	*sigA*	BALH_0790	*fadD*	GCTGATATATTTTGAAAATCGCCGATATAATTCAA
BALH_3885	*sigA*	BALH_0789	*hemN*	GATCTAACATTGTTCTCATATAAGATTACAGATAT
BALH_3885	*sigA*	BALH_0784		TGTTGGAAAAATAATGCATTTGTACTATAATGATA
BALH_3885	*sigA*	BALH_0771		CATTTATTTTCCATTTATAATCTTATAACTTTATT
BALH_3885	*sigA*	BALH_0761	*blT*	CAAGTAAAAACTCCTTTTGACTTACTATCAGTAAG
BALH_3885	*sigA*	BALH_0718	*licT*	TTCGGATTTACATTAAAGAAATAGATAAAATAAAA
BALH_3885	*sigA*	BALH_1170		AAGATAATATGATAAAAAACATTCGCAAAAGAGCT
BALH_3885	*sigA*	BALH_1152		GTTTTTTGTTTGCAGGAAGATGTGATACAGTATAA
BALH_3885	*sigA*	BALH_1113	*sucA*	TTTGGTATTATGTAAATTATTGTAATAACATAAGA
BALH_3885	*sigA*	BALH_0650	*pstS*	TGTTCCAATTTATTTAGTTGTACTAAAAAATATTA
BALH_3885	*sigA*	BALH_0632	*proY*	TGTAGCATAACAGATGGAAATGTTTTATAATTTGA
BALH_3885	*sigA*	BALH_0602	*glpT*	CGTTTACATTTTTGTCACAAAGTTGTATAATTTTC
BALH_3888	*tlyC*	BALH_4341	*pckA*	AAGAAATAAATAGTATACATTATTTATATAATTGTGTTATACTCTTG TTGGGGATTA
BALH_3888	*tlyC*	BALH_4166	*gap*	AGAAATGGAACATTATTGACTAAAAAATTGTGAAATTAATGAAAAT GCTTAGTGACA
BALH_3906	*hrcA*	BALH_3906	*hrcA*	AATCGTGAGCTTCTGTTTCTCACGATT
BALH_3980		BALH_3960		AAATAATAATAGCTAAACAAGTGATTAAAAAGCAAGAT
BALH_3980		BALH_3843		TTCCGGAATTCCCTAGTTGACTTATAGGTTTTATTGAT
BALH_3980		BALH_2769	*acdA*	TTAGTCAAAAACATTGTATTTTTATCGGAATAGTGAAA
BALH_3980		BALH_2614	*ssuB*	TTTTAATAAAACCTAGTTGACTTATCGGAAATATTTGT
BALH_4013		BALH_4012	*nifS*	TGTTGCCTCCTACTTTACACTTTTCAACTGTCTTGACACCTATATTTA CATAGTTTTAAAATAATGACAAGAGTTTTTTTTATAATTCTTATTTTC AACACCATTTAAA
BALH_4013		BALH_4011	*nadB*	AAATTTACCACAACTTTTATTCTTAATATTTTTTTGAGAACAGTAAT AAAATTTTGATACATTTATATCCACAGTTCTGTCAACTTTTCACATT TCATCCTCCGTTGT
BALH_4109		BALH_0790	*fadD*	TTATATGTCTCCTGTTTGACTTTTAAAATCTTCTCGATA
BALH_4109		BALH_4844	*fdx*	ACAACTGACTTACGAGTGAGTAATATATTACTTCTATCC
BALH_4109		BALH_4547	*fadB*	CGACTTATGTCGAAAAATTGAATGAGCATTCATTCAAGA
BALH_4109		BALH_4418	*menE*	GAAATTTTAGCAAACAGCCCAATGGCACTTCGTTTCCTA
BALH_4109		BALH_4110	*lcfA*	TCAACTTATTATTGAGAGAATGAGAAAATTCTAATCCTC
BALH_4109		BALH_3611		TTAAATAAAATGATAAAATGAATATGTATTCATTTTTTG
BALH_4109		BALH_2656	*caiC*	GCGATGAATGTTATAGAAGGAATGATTATTCAAGTGCAT
BALH_4109		BALH_3074		ATTCACTAAGTGTAATCTTTAGTAAATTAATATTTAATT
BALH_4109		BALH_0970	*fadD*	AATTTTCTTTTAATATAATTAATTTAGAATTTTTAAATA
BALH_4173	*phoP*	BALH_3152	*glpQ*	ACTTTTTTATTATAAGTGTTTAA
BALH_4173	*phoP*	BALH_0650	*pstS*	ACATTGTTAAGAGTAAATTTACA
BALH_4173	*phoP*	BALH_4973		ATAAACACATTAAAATGTTTTAA
BALH_4173	*phoP*	BALH_3739		ACAAATACTTTTCAAATGTTATT
BALH_4173	*phoP*	BALH_4173	*phoP*	AAATATATACCTAAAATTCTAAT
BALH_4173	*phoP*	BALH_3325	*cpdC*	AATCTTTACACATTCTATACACT
BALH_4173	*phoP*	BALH_2686		TATTGTTTAACAATAAATTTTTA
BALH_4173	*phoP*	BALH_1333	*resD*	ACGTTAGTAATACAAATTGAAAT
BALH_4173	*phoP*	BALH_3935		AAATGTTAAATAAGAGATTTACG
BALH_4499	*tipA*	BALH_4499	*tipA*	GACTCTAACGTTGCGTCATA
BALH_4632	*cggR*	BALH_4632	*cggR*	CCTATGTGGGACACAAAATGTCACTACGGGACGTAAAGTGACCAC G
BALH_4634	*rpoN*	BALH_3776	*ptb*	AAAAAGTTGGCACGGTATTTGCTTAATAAAAAGACG
BALH_4634	*rpoN*	BALH_3195		ATTTTAATTATTAACGTTCAGGTTCTGGTTAATTGC
BALH_4634	*rpoN*	BALH_2494		TAGAATTTGGCACGGTACTTGCAATATAAAAGATGA
BALH_4634	*rpoN*	BALH_1347	*gdhA*	TTTATAAATTTATAAATTCATAAAATGAAAAGATTT
BALH_4634	*rpoN*	BALH_0771		AAATCGTTTTCACAAATATTTCATTTGAAAAATGTT
BALH_4634	*rpoN*	BALH_1015	*rocD*	AGAGTATTTACAAACTTCTTGTAATAACAAAAGGGG
BALH_4634	*rpoN*	BALH_0317	*gabT*	TATGTATGAGCACCGTTCCTATATGAAAATGACGGC
BALH_4634	*rpoN*	BALH_0301	*rocA*	CAGGATCCTTTTTAATTTTTGCATTTTTATTAATTG
BALH_4670		BALH_2868		TTCATTGTAATATATTATTTTTTTGC
BALH_4670		BALH_4670		AGAGATAGTAGTCTTTATGAATACTA
BALH_4850		BALH_2685		TTAATTAGGAGGAGTGG
BALH_4968		BALH_4968		GCAAAGTGTACTACGCCGTCTCGGCTACAAGATACGAAAAGTATTG TTATTATCTTTCAATCGAGAATAATGTCTATGCCCTATGCACTCACT GGCATACTTTCCACAATAAT

Table 1: Binding sequence predictions for *Bacillus tusciae* DSM 2912. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	Target Gene Locus Tag	Binding Sequence
Btus_0001	Btus_0001	ACAGTTGTGGATAACTG
Btus_0066	Btus_3298	TTTAACCCGAACGATTGACGGCGTATGATGGGGAAACGTTCGGGCTTCCGT
Btus_0134	Btus_0782	GGAGGATTTTTTGAGGAAAAGACGTAAATATGTAGAG
Btus_0748	Btus_0092	CCTTTTACAGCCTT
Btus_0779	Btus_2447	GCTCATCTCCATCCACGGGCAAATAGGGTGTTACATTTTGCCGAATTCACATGTCAC
Btus_0779	Btus_1896	GCCTTTACACTGTATGAAAAAGTTCCGAGCACTTTTTGTCATATCTTGTTAGACCAC
Btus_0783	Btus_2893	GGTTTGCGATGACGACTACATGCTATATTTTAAAA
Btus_0783	Btus_2865	GCTTGACTTTCGATTCACTTGCGAAAAGAATTAAA
Btus_0783	Btus_3278	CATTGCCCAAAAGTGAGGGGACGGTTAAAATGAAT
Btus_0783	Btus_2759	GGTTGATCCCCGGACGATACCTGGGTAAGATGGAA
Btus_0783	Btus_2743	CGCGAACAGAAGAGAGGGTTTGTGCTAAGATGAAA
Btus_0783	Btus_2742	AAAGTAGAATCGTGTTTGGGAGAGAAGACAAGCGC
Btus_0783	Btus_3123	ATGTTGAAATTGTTGTTTAACGAGTTTTTTGGAAA
Btus_0783	Btus_2669	TTTGGCTCCAGATGTGAAACAATGGTATATGAGAA
Btus_0783	Btus_2642	GATTTCGAAACGGCGTTCCGTTTGGTATATTGGTA
Btus_0783	Btus_3089	TATTTAAAAATCTTGTTAATTTCGCAAAAAATACG
Btus_0783	Btus_3086	TTTTGACTGAATAAGTAGATATATGTATAAAAGAA
Btus_0783	Btus_3060	TTTTCTCGGAATTCGAGTTGTGGTGTAGAATGAGA
Btus_0783	Btus_2539	CGTTCACGAAAAAGAGGAAATGCGCTACACTATAG
Btus_0783	Btus_2533	AAAATCAGATAGCTTGTTTGCATTTATCTAGTACT
Btus_0783	Btus_2470	CCATCCCCAGTCAGCCGAAATTTAGTATAATAGTG
Btus_0783	Btus_2447	TATTGCGTCCCAATCCCCACCTCGGTACAATAAGA
Btus_0783	Btus_2400	TATTGGAGCCAGACGTAAAACCTGTTATAATTGTG
Btus_0783	Btus_1982	TCTTGAGTGTGCTGTTAGGTCACGTTATACTGTAA
Btus_0783	Btus_1936	CAAGTAATATGGCCCTTAGGACAGCTTATTCGTTC
Btus_0783	Btus_2374	CTTGGAATTGAACCCGGCGATCCGGTAGAATGGTG
Btus_0783	Btus_2353	AGTTGCAAAGCCGGGAGGAAAGCCTTATACTATCT
Btus_0783	Btus_2309	AATTGACATCCGGCAGCAGGATGTTTATAATACGG
Btus_0783	Btus_2301	CCTTTCTTTCACAGCTGGAATTTCGTATAATGAAG
Btus_0783	Btus_1898	TTTTGACCTTTGGTGTTCGATATGGCATAATGGGG
Btus_0783	Btus_1896	TATAGAACAATCTGGTGAAACCATGTACAATGGAA
Btus_0783	Btus_1892	TGCTGTCAAATCTGTCAATGTTTGATAGAAGGTGA
Btus_0783	Btus_2296	TCACAAGAACGCACCTTGGTTTTGTTATAATAAAT
Btus_0783	Btus_2272	GATTGAATGCTTATACATATCGTTGTATAATCATA
Btus_0783	Btus_1793	TGTTGTGTTTCACGAGTCAATGTAGTATAGTATAA
Btus_0783	Btus_1790	CTTGGATTTTTTCTGTTTCTTATGTTATAGTGGTT
Btus_0783	Btus_2159	AAGACAATATGATGTGAACTCACCTTTTACTTACT
Btus_0783	Btus_2142	AGTTCCGTCCCGCTGCAAAATCTGTTAGGATTAGA
Btus_0783	Btus_1668	TTCTTATAATCGTATCATTTTACCGTAAGTATTCT
Btus_0783	Btus_2084	TCTCGACAAAAAGATCGTTCCATGATATGGTGTAT
Btus_0783	Btus_2022	ATTATAATATATAGTATTATGCAGTCATTCAATCT
Btus_0783	Btus_1583	TGTTACTAAAAAATTTTACGTTTATTACATTTTTG
Btus_0783	Btus_1558	TATTTACTGTACCACTGGAAACCGGTATAATGGTC
Btus_0783	Btus_1487	GCAGGATAATGCTGCGGCGACGCTGTATAATATAG
Btus_0783	Btus_1472	AAAGTAATACCAAGACTCGAACAGTGTACAGTTCT
Btus_0783	Btus_1382	CACGCGCATGAGTGGACGGATCTGGTAAAATTTAA
Btus_0783	Btus_1375	GGTCCAAAAGTGCACCAAAATGGTATATAATCGAA
Btus_0783	Btus_1307	ACATGGAGCCCCGTGGAATGACTGATATAATGAAA
Btus_0783	Btus_1300	TCTTTACATAGTCCCTCGGATGTGGTATGTTAGGG
Btus_0783	Btus_0898	AAGAGACATTGCGTTGGAATTCGACTATAATATAG

Btus_0783	Btus_0847	GTATTAATATTGTCTTAAGTATATGGCAGCTAGCG
Btus_0783	Btus_0837	CCTTTGTTTGCCCGGAAACCTGTGCTAAGATGAGG
Btus_0783	Btus_0827	GCTTCTGTGGAAATGAAAGTTCAAGTATAATAAGG
Btus_0783	Btus_0758	CAGGTAATATTAGAGACATTAGGCTAATTATATGA
Btus_0783	Btus_0748	CAAGTGGTATGTATTTCCTTGTGGACGCCAGTTGG
Btus_0783	Btus_1162	TATTGTCATTAAACATAACAGATTATATAATATAA
Btus_0783	Btus_1118	TATATTCCCTGGTTTACTTATCGTCTATAATGTTA
Btus_0783	Btus_0647	GTTTTACACGGCGGCCACATTCCTTTAGAATAGGA
Btus_0783	Btus_0635	CCTTGACATGTCCACCGGGAAACTTTAAAATTTCT
Btus_0783	Btus_0632	CTATACACAATCCCCGAATCTATGATATACTTTGT
Btus_0783	Btus_0613	TTCGGACTCGACTTTTAGTATTTCATAGAATCATG
Btus_0783	Btus_1056	GTTCGTTGGTTTTCGACGTTTCTGTTACAATAGTA
Btus_0783	Btus_1055	ATGATAACATTGTCTTTGCAGCTTTTGGTTGCTTG
Btus_0783	Btus_0598	GCGGTAATATGAACTTGGCGACTTTCTCGATTTCT
Btus_0783	Btus_0597	TCTTTAGCTCTTTCAGCGGTTCAAGTATAATGGCG
Btus_0783	Btus_0430	TTTTGTCGTCGGGTAACCCTTTTGCTATACTGGCT
Btus_0783	Btus_0369	ACTGAGCGATCAAAGAAATCCCGTGCAAAATATAT
Btus_0783	Btus_0315	GATTGGCAATCGTGCGCAGGAATGCTACAATCATG
Btus_0783	Btus_0310	GATATGATATTGTCTAAACTCCAACATATATTGTG
Btus_0783	Btus_0295	GGTTCTTGTATGCTAAAAACAGTTGTACAATATGA
Btus_0783	Btus_0193	CCAATAATATAGGGGTATAAGGTTAATTTGTATAA
Btus_0783	Btus_0145	TCTTGACCCGGTTGAGCGTCTGTGATATCTTTATA
Btus_0783	Btus_0069	GGTTGAATCCCCTATAGTGATGAGTTATAGTGTTT
Btus_0783	Btus_0040	ATTTTGAGATGTTATTCTCGGTACTTACTAGTTTC
Btus_0898	Btus_1241	GATGCAATTTTACCGGGCG
Btus_0898	Btus_0369	ACGTTTTATATACGGTAGA
Btus_0898	Btus_3089	GGGTTTTGCTTGGACGTAG
Btus_0898	Btus_1982	ATCGCAAACCTAAATTTCA
Btus_0898	Btus_2389	GCACTTGATGCCTTACAATATCAAACACGCAGGCAAAAACTACTAAGCGGCTAGTAGCCAGGGC
Btus_0898	Btus_2374	AGGCCATCTTACCACGTAA
Btus_0898	Btus_1781	CTTTTTTCCTAAAACGTCG
Btus_0898	Btus_0898	TTGTTTTATTTCAGCTTGT
Btus_0898	Btus_0837	AACGGAAAAATAATTCGGG
Btus_0898	Btus_1123	GCGTTTTAGGACAACCCAA
Btus_0898	Btus_0363	GGTGAAAACAGCGTTTGCG
Btus_0898	Btus_0315	AATGCTACAATCATGTGTA
Btus_0898	Btus_0295	GATTTTTGTCAACATGTTA
Btus_0898	Btus_2893	GTTTTTTACCTAAAAGTAG
Btus_0898	Btus_3060	GGACATAAATTCTTTTACA
Btus_1644	Btus_2179	CAGGCGTGCCAACAGGCCAAAA
Btus_1644	Btus_1644	GCCGTTTATATGCAAACAAAAA
Btus_1644	Btus_0302	ACTGCCTGTAAATATTAATTAA
Btus_1813	Btus_1162	TAACAGATTATATAATA
Btus_1813	Btus_2232	GAGTGTTTTACAGGAGT
Btus_1846	Btus_0076	TATTGTTAAAATCCCTTCGGGTGTACACCCTAATGT
Btus_1846	Btus_1932	TTCCATTCTATTCCTCCGGACATCGCATGATATTCT
Btus_1846	Btus_1929	GTGTATAAAAGATTAAGGTCAAGGGCCGCTAAGGAA
Btus_1846	Btus_2296	TCACAAGAACGCACCTTGGTTTTGTTATAATAAATG
Btus_1846	Btus_2251	GATGATGAGACAAAAGCGCAGTGGACAAACGAAAAG
Btus_1846	Btus_1791	AGATTGGAATAAGTTGGTATACTCCCGAATTCCCGG
Btus_1846	Btus_0809	TCTGCATGGACCCTTTTCCATGGTACACACTAATTC
Btus_1846	Btus_0747	TGGGTTTAAGACGCGTTTCACTCTAAACACATCCAA
Btus_1847	Btus_0991	CGGGACTATTCCCGGACAAACCCGCATAGGGTGAAAAAGG
Btus_1847	Btus_1278	TTCTGATGATTGTTATCCAGGATGCATTGGATGAAGATTA
Btus_1847	Btus_1243	GAAAAAAGACCCAAACGCCCCCAGCGCACTTGTAACCGTT
Btus_1847	Btus_0525	AAGGAAAGAAGTATAGCTTGGTTATGTTTTCCTTTTAGAA
Btus_1847	Btus_0474	TGAGAAAAAAAACTACGAGTATCCACCCCATATCTTTCTT
Btus_1847	Btus_0439	GCAAAAAGATAGATAAGTCGCTTCCTCTCTTGCTGTTACC

Btus_1847	Btus_3213	TCGGTATACTGCAGCGGGCCGATGCATAAAAACCAACATG
Btus_1847	Btus_2533	TGAACCGAATGTGAACCTTGGGTAAATACAATTTTACAGA
Btus_1847	Btus_2501	GGTGCTATCAGACTAACGGACATGAATATATTTTACAAAA
Btus_1847	Btus_2452	TCTTTCTCAAACGACCGTTTGGATGATATACTCAACATAG
Btus_1847	Btus_1764	GCGGTCATATTTCTCCTTCGGGTTCATAGATATGTACCGT
Btus_2021	Btus_2022	ACCAAACTCTTTTAGCTATTATAGGCC
Btus_2145	Btus_0847	CAGCCCCTGGCAAGAATCTTGCTTGGTAAGAAAACG
Btus_2145	Btus_0600	GGCGGCAATATTTTTCTGCGTCTCACGGCCCTAGGT
Btus_2145	Btus_0369	GTTTCTTTAGGGCACGTTTTATATACGGTAGAAACG
Btus_2145	Btus_3089	GGCGCGTACTATTACTATCCAAGTCTGCAAAAGTCC
Btus_2145	Btus_2389	CATAGGCCGGCATGTTTTTTGCATATATTGAGGAAG
Btus_2145	Btus_2341	CGAGAAGGCGGATAAGAGTTGGAGATTGATTCGAGG
Btus_2533	Btus_2533	CAATTTTACAGACCATCTATACA

Table 1: Binding sequence predictions for *Bacillus weihenstephanensis* KBAB4. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
BcerKBAB4_0001	*dnaA*	BcerKBAB4_0001	*dnaA*	GATTTTGTGGATAACTT
BcerKBAB4_0041		BcerKBAB4_1490		ACGTTTTGAGCTTGCTTTTATCATAAAAGTATTTTAAGC ATGCACAAAAAA
BcerKBAB4_0041		BcerKBAB4_0666		CAATTTTTTGCTTATAATATATATTGACAACACTTTTAA TAAGCATTTTTC
BcerKBAB4_0041		BcerKBAB4_0609		ACTCAATGGATTTATAAATTGATAAAGTTTAATATTTT GTAAAGATAGACA
BcerKBAB4_0041		BcerKBAB4_5260		ATTAAGAATGCTTGCTTTTGCAAAAAACTATTTTTTTAC AAGCGAAAAACC
BcerKBAB4_0041		BcerKBAB4_5249		CCTATGGACGAACATTTTCAATCAATAAAAATAAAATA TTCGTATTTTAGG
BcerKBAB4_0041		BcerKBAB4_5114	*glyA*	TGAAAACATGAAGGTTCGAAAGAATTACAACCTGAAT TTTCGTTTTTTCTG
BcerKBAB4_0041		BcerKBAB4_0269		CTAAAACACGAACGTTATCGATGACTAATAAAAAAAC ATTCGATTATTTAT
BcerKBAB4_0041		BcerKBAB4_0248		GGGAATTATGCTTGCTTTTATATTTTCAAATATTTTTAC AAGCAAAAATTT
BcerKBAB4_0041		BcerKBAB4_3914		GTTAAAAATGCATAAATATTTATTTGAGTAAATATTCA TTGGCAATAGTAA
BcerKBAB4_0041		BcerKBAB4_3718		CATAAGAAAACTGAGATACTGACAACAGTTAAATAAT GATCCTTTAACACA
BcerKBAB4_0088		BcerKBAB4_4984		AGAAGGAGTTTTTAAGCCAATGTCGAAATTATAGTAC
BcerKBAB4_0088		BcerKBAB4_0523	*aspA*	AAAAGGATATGCATCAAATTAAGCGAGTTGAAATAAA
BcerKBAB4_0088		BcerKBAB4_3734		AAAAGGGAAACGTACAAAGACGTTGAATATTTTTCTT
BcerKBAB4_0088		BcerKBAB4_4145	*dnaG*	TAAAGGAATCTTTTAATTTATATCGAATACAAAATAC
BcerKBAB4_0088		BcerKBAB4_4023		AAAGGGAATTTTCACACAATTGTCGAACAATTCATGT
BcerKBAB4_0242		BcerKBAB4_1223		AGGAAAAACACTTTATAACGTGTTTGAATAACT
BcerKBAB4_0242		BcerKBAB4_0770		AATTACACCACTTCATATATTTTAATTTTAGAA
BcerKBAB4_0242		BcerKBAB4_4826	*ldh*	ACAAAAATTTTGCTCAATATTTCACAAAGTATC
BcerKBAB4_0242		BcerKBAB4_4713	*ldh*	GTATTAAACACTTCCTTAAGTGTTATTAGGCAC
BcerKBAB4_0242		BcerKBAB4_4629		AATAATTTTTACATCACCGTTACGCATATTCAT
BcerKBAB4_0242		BcerKBAB4_1809		TTTGTGATGACTTTATCACTTTTCCATTTAGCA
BcerKBAB4_0242		BcerKBAB4_1798	*ldh*	ATTTTAAATACTTTATCGCTCCTTAAAACTTTA
BcerKBAB4_0242		BcerKBAB4_1319		AGCAAATAATCTTCTTAAGGCATTTAATCAACA
BcerKBAB4_0306		BcerKBAB4_3257		GCTTTTACTTTAAACTTAA
BcerKBAB4_0306		BcerKBAB4_3173		ACTTTTTCTTAATATTCTA
BcerKBAB4_0306		BcerKBAB4_2245		CGTGAAAGATTGTTTTACT
BcerKBAB4_0306		BcerKBAB4_2108		TCGTTTTACAAAAAAACTA
BcerKBAB4_0306		BcerKBAB4_1413		AATATACAATTACTTTGTG
BcerKBAB4_0306		BcerKBAB4_1276		TATGTATGATTCTTTTGAA
BcerKBAB4_0306		BcerKBAB4_0756		TTATTTTATTAGAAAATAG
BcerKBAB4_0306		BcerKBAB4_1053	*rocD*	AATGCTATAATATTTAGAG
BcerKBAB4_0306		BcerKBAB4_0305		ATTGTAGAATCGTTTTGTG
BcerKBAB4_0306		BcerKBAB4_0290		ACATCTTATTACAACATTT
BcerKBAB4_0306		BcerKBAB4_2639		ACATTTTTATAAAATTTAA
BcerKBAB4_0306		BcerKBAB4_2563		AAAACAAAATTATATTTTA
BcerKBAB4_0306		BcerKBAB4_2117		TATTTTTATATATATGTAA
BcerKBAB4_0306		BcerKBAB4_1275		AGAGGAACAATATTGGGCA
BcerKBAB4_0306		BcerKBAB4_0409		TCGTTTTTTTAAAACGCTA
BcerKBAB4_0306		BcerKBAB4_0306		ATTGCAAGCTTGTTATGAG
BcerKBAB4_0306		BcerKBAB4_4008		TATGCGTAAATATTTTGAT
BcerKBAB4_0453		BcerKBAB4_1097	*spxA*	ACTATTGTATAATGATTATAG
BcerKBAB4_0453		BcerKBAB4_1058		ATTAGTTTATAATAATTTTAA
BcerKBAB4_0453		BcerKBAB4_0453		AATATTCTTAATATTTTATTA
BcerKBAB4_0453		BcerKBAB4_0390		TATCTTATTAATGTTGTATTA

BcerKBAB4_0453		BcerKBAB4_0325		AATCTTAATAATATCAATTAA
BcerKBAB4_0453		BcerKBAB4_4605		ATGGTAAATAATATTATTTAC
BcerKBAB4_0453		BcerKBAB4_3922		ACTTTTTGAGAATAATTATCA
BcerKBAB4_0453		BcerKBAB4_4311	*hemA*	AATATTAATAATATTTCATCT
BcerKBAB4_0453		BcerKBAB4_2820		TTTTTAAGATTGTAATTTTAA
BcerKBAB4_0543		BcerKBAB4_0544		CAACTGATTGAATATATATGCTCAATTATACA
BcerKBAB4_0812		BcerKBAB4_0474		TGTACGAAGGCATTTATCTTTTTAAAAATTCAAAATTACCTCATGAACAAA
BcerKBAB4_0903		BcerKBAB4_4740		GGTTTAACAGTCGTTGCAGTTGAGGATTATAAAGAAAG
BcerKBAB4_0903		BcerKBAB4_4652		CATTTATCGCTTCAGTTGTAACGGAATAATATAATAAA
BcerKBAB4_0903		BcerKBAB4_0290		AAAAATAATATAAAATTTTTAAATAATTAAAAATTTTA
BcerKBAB4_0903		BcerKBAB4_4605		ACGTTCGAAGGTATAGGGAAATACGGTTAAAAGATTTC
BcerKBAB4_0903		BcerKBAB4_5071		TTTTTTATTATTAATACATATAGATAAAGCGATATTTT
BcerKBAB4_0903		BcerKBAB4_4442		CAAAAAAAGAAAATCGGTAACGTAATTTAAAAGATTAG
BcerKBAB4_0903		BcerKBAB4_0079		CGTTTTGATTAGCAAAACAATGGTTAATAATGAGTAGG
BcerKBAB4_0903		BcerKBAB4_4419		GGTTTCAAGTCTATAACTTGGAGGAAAAGAATGAACAA
BcerKBAB4_0903		BcerKBAB4_4254		AAATAATAACCAACGGGTCAAAGAATACGTAATCTTCA
BcerKBAB4_0903		BcerKBAB4_4054		TGTTTAAATCCTGCAAAATTTGTCTATAAAAAGAATAA
BcerKBAB4_0903		BcerKBAB4_3331		TGTTTCCTACTAGTAAATAGAAGATAAGCTATGTTTAA
BcerKBAB4_0903		BcerKBAB4_3330		TTTTTAAAATGAAATAAAAAGGGGCGGAGAAAATGGAA
BcerKBAB4_0903		BcerKBAB4_2820		GCTATAAGTAAAAAGGTATTAAAGAAGATAAAGTTGTC
BcerKBAB4_0903		BcerKBAB4_3257		TTTTTACAATTGTAACGAAAATGAAATTTGAATTTTCA
BcerKBAB4_0903		BcerKBAB4_3173		AGTGTTTATTACTGTTGATGGGGAAAAGTATCATATTA
BcerKBAB4_0903		BcerKBAB4_1875		AAAGGTTTATGTAGGGGTTTATTTATAAATAATTTAAG
BcerKBAB4_0903		BcerKBAB4_1866		GGATTAATAGATTTATTAAAAGAATAGCAAAAAATAAA
BcerKBAB4_0903		BcerKBAB4_1861	*nadE*	GTTTTCAAAAACATTCCCATTGAATATAAAATAAAGAA
BcerKBAB4_0903		BcerKBAB4_2260		TTATATCAAATTAGAGTTATAGAAATAAAAAAATGTGT
BcerKBAB4_0903		BcerKBAB4_1775		GTTATAAAAAACTGAAAATACAGAAAAATAACATCTTA
BcerKBAB4_0903		BcerKBAB4_2187		GTTTTTATTTTTTCTAAAAAAGGAGAATACAGATGGAG
BcerKBAB4_0903		BcerKBAB4_2108		TTTTTTGATGTATTAGGGCTATGATACAATTACACAAG
BcerKBAB4_0903		BcerKBAB4_1319		TTTATAAATGTTTTAAAAAAGGGGGCTTATAAGAAAAA
BcerKBAB4_0903		BcerKBAB4_1276		TATCAGTCAAATAGGGCTATAAACACCCGTCATTTTGA
BcerKBAB4_0903		BcerKBAB4_0770		TTTTTCGTGCATTTTTAGAGGAGATATAAAAATATTAT
BcerKBAB4_0903		BcerKBAB4_0756		AAAAGTGTTTGTAAAGTAAACTTTTTATAATAGTTTGG
BcerKBAB4_0903		BcerKBAB4_1142		GGATTTAAAAAAGTGTTCAGTTATGAAAAAAATTTTTT
BcerKBAB4_0903		BcerKBAB4_0667		AATATATATTATAAGCAAAAAATTGATAAAACGTTATC
BcerKBAB4_0903		BcerKBAB4_1058		AGTTTATAATAATTTTAAATAAATAATAATTACTATGA
BcerKBAB4_0903		BcerKBAB4_4984		TCGTTATTCACGATGTTAATAGGATAAAATAAAGATAG
BcerKBAB4_0903		BcerKBAB4_4944	*clpP*	AGAATTTATTTCATTGGATAGACAATTTTCTTATTTTT
BcerKBAB4_0903		BcerKBAB4_0551		GAATATATAATACAGATATAAGGGAACAAAGTATATAA
BcerKBAB4_0903		BcerKBAB4_4886		GTTTTAGTATGTACATTTAAAGGGAGGAGAGTCATAAA
BcerKBAB4_0903		BcerKBAB4_4885		GATTTACCGTATTGCGGTAGTGGGAATGGTATTATTCG
BcerKBAB4_0903		BcerKBAB4_4874		CTTTTCTTATTATGAAAAAAAGGAAGTGTATTATATGG
BcerKBAB4_0903		BcerKBAB4_5259		TTGATCATATTTTTATAAGGAGGAAATAGACGATGATG
BcerKBAB4_0919		BcerKBAB4_1206		TATAGCTTCCCAAGCGGACATGCAATGCTATCTATTATA
BcerKBAB4_0946		BcerKBAB4_0947		GACGGAGAAAAGAAGAGATC
BcerKBAB4_0960		BcerKBAB4_0604		GATATAATAAAACCTAACAA
BcerKBAB4_0960		BcerKBAB4_0528		ATAAAGGTATATTAAGATAA
BcerKBAB4_0960		BcerKBAB4_2868		ATATGAACATATTATAGTTT
BcerKBAB4_0960		BcerKBAB4_0514		GATATAAATATTTTGAAGAA
BcerKBAB4_0960		BcerKBAB4_4867		AATAAGTCTTATAAATATTT
BcerKBAB4_1363		BcerKBAB4_5206		GGCGTTACGACTAATGCCTTTGGAACA
BcerKBAB4_1363		BcerKBAB4_2086		TTGGATATGAATTTTGTATTACAGGTT
BcerKBAB4_1399		BcerKBAB4_1990		TTTTAATTTTCTATAAT

BcerKBAB4_1399		BcerKBAB4_1975		GTATGATTTTAAAGAAT
BcerKBAB4_1399		BcerKBAB4_1809		AACTGTTTTTTGATCGT
BcerKBAB4_1399		BcerKBAB4_1372		TAATAATTTCTTGAATT
BcerKBAB4_1399		BcerKBAB4_4629		ACATGGTTTTAAACTTT
BcerKBAB4_1667		BcerKBAB4_0416		CGAAATCTTTTGCGAAAGATAATGTATTGGTATTATTG TATATC
BcerKBAB4_1750		BcerKBAB4_1752		AATGAACAAAAGTTCAAAAGTGTATTTACATTTGTTCA A
BcerKBAB4_1975		BcerKBAB4_1981		GAGGATGACATTATTCAAACT
BcerKBAB4_1975		BcerKBAB4_1971		AGTGTAGACATAAAATGTTGG
BcerKBAB4_2639		BcerKBAB4_2578		TCAAATTAGGTGAACAAAAGTGGACAAAACGAGACAG ATGTCTCATTTTGTCCACTTTTTTATT
BcerKBAB4_3122		BcerKBAB4_3122		CTGAAAATTATGTTAATATGTAAATTTATTTTTCCAAAT GTTTCCTTAATGGTA
BcerKBAB4_3213		BcerKBAB4_0446		ATTGAAGATTACGAT
BcerKBAB4_3213		BcerKBAB4_3213		ATCAAAAAAATTGAT
BcerKBAB4_3213		BcerKBAB4_0188		AACGAAAAATAAAAT
BcerKBAB4_3287		BcerKBAB4_3287		CGCCATCCTCATAGTCTTCTTTTATGATACAGAAATCG ACCTTATCTTCACTTAGCTGATAGATCTTGTCAAAATCA TCTACCACTTCTTTAATATAACTGTGTATTCCTCAA
BcerKBAB4_3287		BcerKBAB4_5254		ACAAAGAGTACTACGCCGCCTTGGCTACAAAATGCGA AAAGTATTGTTATTATCTTTCAATCGAGAATAATGTCT ATGCCCGATGCATTCACTGGCATACTTTCCACAATAAT
BcerKBAB4_3377		BcerKBAB4_4959		ATGGCTTATAGACAAGCAAAAA
BcerKBAB4_3377		BcerKBAB4_0522		AACGTTAATAAACAAGCTTAAT
BcerKBAB4_3377		BcerKBAB4_4434	*dnaE*	TCTTTTTTCATACAAGAGAATG
BcerKBAB4_3377		BcerKBAB4_4348	*uvrC*	TTTTCTTGCATACAACCCACAA
BcerKBAB4_3377		BcerKBAB4_4268	*ruvA*	AAATGTTGAAAACAAAAGCCAA
BcerKBAB4_3377		BcerKBAB4_4203		TTTAGCCAACATCCGTTCCTAT
BcerKBAB4_3377		BcerKBAB4_3550	*recA*	AAAATCGAATAAATGTTCGATT
BcerKBAB4_3377		BcerKBAB4_3377		CAAGCTTGAATACAAACATCTT
BcerKBAB4_3377		BcerKBAB4_3285		TATCTTTGTATACAAGACTAAA
BcerKBAB4_3377		BcerKBAB4_3095		TTTGCTTGCTAACTAATAATAT
BcerKBAB4_3377		BcerKBAB4_1910		TAAACAAAACACGTATACTTTA
BcerKBAB4_3377		BcerKBAB4_2206		TGATTATACAAATAAGAAATAA
BcerKBAB4_3377		BcerKBAB4_2107		TACTGGTATAAATAAAAAAAAT
BcerKBAB4_3377		BcerKBAB4_0925		CAATCTTGCATATGAGGGTACT
BcerKBAB4_3377		BcerKBAB4_1133		ATAAAAGAATTATAATTTTGTA
BcerKBAB4_3651		BcerKBAB4_1319		AAGAATTCCGTAAATTAGTTGTAGA
BcerKBAB4_3651		BcerKBAB4_0770		TATAATCTTATTAATACTGTGATTA
BcerKBAB4_3677		BcerKBAB4_0192		GAATTGAAATTTAACAATAAATACTTATTTTTATTACC AACTCCTAAAA
BcerKBAB4_3677		BcerKBAB4_1082		TCTAGACAAAAGTACATTTATGTTTTAAAATTAGTACC AAGTCATAATA
BcerKBAB4_3718		BcerKBAB4_3718		ATAATGATCCTTTAACACAGCCCCGTGAGGTTGAGAAG GTAACGGTTTGAAATA
BcerKBAB4_3730		BcerKBAB4_1851		AAAAATAAAAGTTAATTAACCATACCTATATAACAC
BcerKBAB4_3730		BcerKBAB4_2153		ACATATTAGACACTTAACCCTTTTATAACTTACGAG
BcerKBAB4_3730		BcerKBAB4_2064		AAATAATATAACTTATTAACTTACGAAAAAAAAGTA
BcerKBAB4_3730		BcerKBAB4_1474		ATTATTAAATTTACACTTTTTGATGAATTTTATTTA
BcerKBAB4_3730		BcerKBAB4_0988		AGCAATTCTACTGTATACATAGCAATGAATTACCCG
BcerKBAB4_3730		BcerKBAB4_1378		TAATATTATTTAAATTTTCCTCCATATAAGAAGGAA
BcerKBAB4_3730		BcerKBAB4_0677		CAAGTATAAATTCCCGTCTTTCCCAAAAACTAACAC
BcerKBAB4_3730		BcerKBAB4_0624		CAAGAAATGTATCTGTAAAGACCGACAATATAATAG
BcerKBAB4_3730		BcerKBAB4_0548		AAGCATAATTTTTTCATAAAAAGCAAAAATTAACAG
BcerKBAB4_3730		BcerKBAB4_4700		TCTCATAATTCCTACTCTCAACGCTCATACTATGTA
BcerKBAB4_3730		BcerKBAB4_4563		AATGAATAATATACAAAATTAGCCACAAAATAGCAC
BcerKBAB4_3730		BcerKBAB4_0184		ACGAAAAATAAAGGGCCTTTTATTTTACTAAACTAA
BcerKBAB4_3730		BcerKBAB4_3906		TGGTATTAAAAATGTATAAATTGGAAAAAATAAGTT
BcerKBAB4_3730		BcerKBAB4_3900		TGCAAATTTATTTAAGAAAAAGGCATAAACTAAAAA
BcerKBAB4_3730		BcerKBAB4_4172		TATCAATGAAATTTCACGTAAAGGAAAGAATGACCT
BcerKBAB4_3730		BcerKBAB4_4137		TTTAATACGATAGTAATATAGATAACATATAACTTC
BcerKBAB4_3730		BcerKBAB4_4058		GTTATTCAAAGAGAAAAACGTACGCATTGTTATTTC

BcerKBAB4_3730		BcerKBAB4_2553		GAAAATAAGAAAAACATATAACAAATGTATAAAGAA
BcerKBAB4_3730		BcerKBAB4_2341		AATGAATAAAATTCAAACATATAAAAATAATAACTA
BcerKBAB4_3731		BcerKBAB4_0145		TTGTTCTCTTTTTTAATTTTAAAGCATATATGTAACTGAA
BcerKBAB4_3731		BcerKBAB4_0141		CAAGCATAAACTTTCCTCTTGTCCCATATAAGTAATTAGA
BcerKBAB4_3731		BcerKBAB4_4426		AGGAAAGAAAAGATAATTCTTTGTCCATTTTATTTTTCTT
BcerKBAB4_3731		BcerKBAB4_4419		GAATGATATAAAAGTAGCGTTAGTCATATTCTTATTAAGG
BcerKBAB4_3731		BcerKBAB4_4389		GGAGAATAAAAAATACTATTTTTATAACTTTATTTATCTA
BcerKBAB4_3731		BcerKBAB4_4326		GAACAAGATTTTGTACTTTGTCGAGCATATGTATTATCAT
BcerKBAB4_3731		BcerKBAB4_4260		CTTGTCATTCTTGTCTCATACGTGCATATAAATTATTGTA
BcerKBAB4_3731		BcerKBAB4_3772		GTGCAAAATAAGATATGTATTACCTTACTTTTATAAAGCG
BcerKBAB4_3731		BcerKBAB4_3755		TGGTCTAAATTCACCACATATCCACGTATACTGAAATGAA
BcerKBAB4_3731		BcerKBAB4_4194		TATACAAATTATAACTTAAATTTACATAAGTTAACAAAAA
BcerKBAB4_3731		BcerKBAB4_3475		TAGTCTTTTGTACAAAGTGTTGCCCATATATTACTGGAAG
BcerKBAB4_3731		BcerKBAB4_3294		TAATCATAATACAGTGGAATCAAAAATAAAATAAAAACAG
BcerKBAB4_3731		BcerKBAB4_3057		TGATAAATTTAACCGAACAATTTTCATACCATTGTTTATT
BcerKBAB4_3731		BcerKBAB4_2488		TAAAAGTTTTACATAAGTGATCATCGTTTTGCTCAAGGAA
BcerKBAB4_3731		BcerKBAB4_1925		CGAAGAAATTACCTTCGTCAGATACAGAAAAAGAATAATA
BcerKBAB4_3731		BcerKBAB4_1917		AAAAATATAAACATTATTACATTCTGTTTTCATTCATATT
BcerKBAB4_3731		BcerKBAB4_2355		TAGGAATTTTTTTGTGCATTGGTACAGAATCTGTTTTAAA
BcerKBAB4_3731		BcerKBAB4_1874		TATGTCTAATTGTCCAAGCTCACACATATGCATAATAGTA
BcerKBAB4_3731		BcerKBAB4_2159		TGATCGTTTCACTTACGTGGAAATGATCGTTTATACTGAC
BcerKBAB4_3731		BcerKBAB4_1625		AGAAAAAATAAAAAACATACATTACCTTATGAAAAGGAAT
BcerKBAB4_3731		BcerKBAB4_2074		AGTGCCTAGTTTTCATGTTTTCATAAAAAAATGTTAAAAA
BcerKBAB4_3731		BcerKBAB4_2046		GGGAAATATTGTATACTGCTACTTTAGCAAAAACAAATTT
BcerKBAB4_3731		BcerKBAB4_1433		AAATTTATCAGTATACTTTAACCTGTTTTAATATACTATA
BcerKBAB4_3731		BcerKBAB4_0775		TCGTTCTATTTACTACATTTATTTCATAAATTATGGATAG
BcerKBAB4_3731		BcerKBAB4_0680		TCCTCATGAAAGCTTGTTCCTCATCATACACTTTCTTAAG
BcerKBAB4_3731		BcerKBAB4_1056		TAGTTATTCTTGTTATGGACAGTTCGTACGTTCTTTATAA
BcerKBAB4_3731		BcerKBAB4_4833		TAGATAGGTTCAATAAGCACTTGAACCAGATCTACTTCTT
BcerKBAB4_3731		BcerKBAB4_0466		AATAAAAATGTTATACGTATACTATCTCATTGGTTGGTAA
BcerKBAB4_3731		BcerKBAB4_0465		AAAGAATGTTTAAGGACATCCTAGCATACCTTTTATAATA
BcerKBAB4_3731		BcerKBAB4_5259		CCAATATGTTTGAGGATTATTGATCATATTTTTATAAGGA
BcerKBAB4_3731		BcerKBAB4_4710		TAAGTGAAATTTAAGAAAAAAACGAATATTTTTTAAAAAA
BcerKBAB4_3731		BcerKBAB4_5084		ATGTTCTAAAAGATCCCGTATTTCCATAAGAATGAAAGGA
BcerKBAB4_3809		BcerKBAB4_4424		TGTATTTTGTTAAAA
BcerKBAB4_3809		BcerKBAB4_3302		ATTATTTACTTATTT
BcerKBAB4_3809		BcerKBAB4_2156		TTTATTTATTTAAAA
BcerKBAB4_3903		BcerKBAB4_1474		CCAAATAAAAAGCACTATCTCAGTAGTGATTCCGAG
BcerKBAB4_3903		BcerKBAB4_0988		ACGAGTCAAAGGGTGTAGCAATTCTACTGTATACAT
BcerKBAB4_3903		BcerKBAB4_1378		TACTTGTAGAATGTCCTTTTCTTGCAAAAAGTATGA

BcerKBAB4_3903		BcerKBAB4_0677		AATGATTAAATTAGTGAAGTCTATTAAAGAAAAGAA
BcerKBAB4_3903		BcerKBAB4_0624		ACTTGTTGAAAAGGAAGTTAGGTGGGGAAATTGTTA
BcerKBAB4_3903		BcerKBAB4_1058		AATAGTTGACAAGGATTTAATTTTGAATAATAATTA
BcerKBAB4_3903		BcerKBAB4_0548		CGTATTAAAAAAGTATTTTTCGTTTTTAATTGTCAC
BcerKBAB4_3903		BcerKBAB4_4744		ACAAAATTAGAGGTTAGTTAAACCTCTAATAAAAAA
BcerKBAB4_3903		BcerKBAB4_0345		CATACATGAATTCCATATAATATGGAAATATTAAAA
BcerKBAB4_3903		BcerKBAB4_5125		AAGCCTTACTCTATTTTTTGTATGAAAAATTCGTAA
BcerKBAB4_3903		BcerKBAB4_4620		AACTTTAAATACGTTTACTGCTTCTCTAATGCCTCA
BcerKBAB4_3903		BcerKBAB4_4605		TAGCGTTTTTCTTTTCTTCTTTAGCTGATAGAAAAC
BcerKBAB4_3903		BcerKBAB4_5080		AATGTCTAAAACAGCTTTTCATTAATCAATATCCAA
BcerKBAB4_3903		BcerKBAB4_4563		CGTGATATAATGTTTTAGTTTCTCCCTTATATATAC
BcerKBAB4_3903		BcerKBAB4_3906		ATGGTATTAAAAATGTATAAATTGGAAAAAATAAGT
BcerKBAB4_3903		BcerKBAB4_4172		AAGAATGTATCTTGTTCACTATAAAATAGTTTGCCA
BcerKBAB4_3903		BcerKBAB4_4054		CATGTTTAAATCCTGCAAAATTTGTCTATAAAAAGA
BcerKBAB4_3903		BcerKBAB4_2820		ATAAGTAAAAAGGTATTAAAGAAGATAAAGTTGTCC
BcerKBAB4_3903		BcerKBAB4_2341		AAGTGTAAAACAGGGATTTGTTTTAAACAATCGCAC
BcerKBAB4_3903		BcerKBAB4_2153		GTCATAATAATCTTTTTTATCTTCACTTTTATCAAA
BcerKBAB4_3922		BcerKBAB4_2432		CTTCCAAATAGTCACTCATATTCATGGTAATTGAAAA
BcerKBAB4_3922		BcerKBAB4_1823		ATAATTTACATAGAATAATAATAAGTACCAGGTAATA
BcerKBAB4_3922		BcerKBAB4_2249		TTTTTATTAAGTTAAAGAGTATTCACATCAACTGGCA
BcerKBAB4_3922		BcerKBAB4_2175		TGTAAGAATATTAAATGAAAAACATTATCAATGAATA
BcerKBAB4_3922		BcerKBAB4_0757		AATTTCAAAGTGAAATAAAAGGAACAATGATATTTTT
BcerKBAB4_3922		BcerKBAB4_0753		GTAATATAGGTTAACTATGACTTTTTGCTTACTTAGT
BcerKBAB4_3922		BcerKBAB4_4974		TTAATAAACTATTACCAATAGTAAATCTCTCAGAAAA
BcerKBAB4_3922		BcerKBAB4_0530		TGTAATGACTTTTATCAAGAGTAATAGCAACTGTTAC
BcerKBAB4_3922		BcerKBAB4_4748		TTTATATAATAGTAATAAAAAATAATACCAAGTACTA
BcerKBAB4_3922		BcerKBAB4_0332		TTGTTTAACTATTACTAATAGTGATAACTAAGAAATT
BcerKBAB4_3922		BcerKBAB4_3506		ATATATACTTAGTAACGATAATCATTATCATTGAAAA
BcerKBAB4_3922		BcerKBAB4_3366		TTTTTCTTTTGGTTATAAAACATATTTAAAACTTTTA
BcerKBAB4_4008		BcerKBAB4_4007		TCATACTGACACGTTTTTTATCGTACGTTATTTA
BcerKBAB4_4023		BcerKBAB4_0057		TTTTGACAAAAAAA
BcerKBAB4_4023		BcerKBAB4_3732		TCTAATACAGATCT
BcerKBAB4_4027		BcerKBAB4_1276		ATGATTAAAGTTTCAC
BcerKBAB4_4027		BcerKBAB4_0756		AATGTTGAAATAACTC
BcerKBAB4_4027		BcerKBAB4_1053	*rocD*	AAAATAAAAATAAACT
BcerKBAB4_4027		BcerKBAB4_0305		AAATTAAATCTCTGTA
BcerKBAB4_4027		BcerKBAB4_0290		AACGTAAAAATAATTA
BcerKBAB4_4027		BcerKBAB4_3257		TCGTAAAAAATAAATA
BcerKBAB4_4027		BcerKBAB4_3173		ATGAATAAGATTAATA
BcerKBAB4_4027		BcerKBAB4_2245		AAGTAAAACATTATTA
BcerKBAB4_4027		BcerKBAB4_2108		ATGAAAAAACATTTAT
BcerKBAB4_4027		BcerKBAB4_3965	*argC*	AATTAAAAAATAAGTA
BcerKBAB4_4057		BcerKBAB4_1735		AATTTGCATTAAGGAAACT
BcerKBAB4_4144		BcerKBAB4_0496		ATTTGATTTTTTATGTGTGACATGATAAAAGGAAT
BcerKBAB4_4144		BcerKBAB4_4835		TATTGCGAAAATTCAAACTGTATCATATATTTATT
BcerKBAB4_4144		BcerKBAB4_0474		TATTTACCATTCGAATTGAACCTTTTATTATAAAT
BcerKBAB4_4144		BcerKBAB4_4826	*ldh*	GGTTGTAACAAATATGATGAAAATGTAATATAACA
BcerKBAB4_4144		BcerKBAB4_0453		TGTTCTTTCTTATATAAAATAAATCTATATTAATG
BcerKBAB4_4144		BcerKBAB4_0446		TATTTCATAAATCCCACATTAATAGTATAATTATT
BcerKBAB4_4144		BcerKBAB4_0418		TATTTATTATTGTACTTTTTCACCTTACTATTTTG
BcerKBAB4_4144		BcerKBAB4_0416		GTTTTATCATTCCACTTTTTCATGTTATTATTTAT
BcerKBAB4_4144		BcerKBAB4_0409		TTTTGCGATTTTATAAGAATTTTAATAAAATTATA
BcerKBAB4_4144		BcerKBAB4_5260		AGAATATTATTCTACACAAAGAAAAAAAAAATATA
BcerKBAB4_4144		BcerKBAB4_5259		GTTTTACAAATGTGGGTAGTTATTTTACAATAGAA
BcerKBAB4_4144		BcerKBAB4_5254		ATTTAGCACATTTAAGCTTAGATAATAACATGAAA
BcerKBAB4_4144		BcerKBAB4_5244		TAAACATATGATAAGAAATATGTGATATACTTTGA
BcerKBAB4_4144		BcerKBAB4_5206		GTGGTAATATAGTTACAATGCTTATTCATGTTTCA
BcerKBAB4_4144		BcerKBAB4_4791		TATATTTTTTACGGAAATAAACGAGTACAATAATT

BcerKBAB4_4144		BcerKBAB4_0390		GATATGATATCTTATTAATGTTGTATTATTGATAA
BcerKBAB4_4144		BcerKBAB4_4740		ATTAAAGAAGATATTATGAATCGTTTATAATAGAA
BcerKBAB4_4144		BcerKBAB4_0379		CTTTTTCTCATTAGATAAATAATGCTACAATATAA
BcerKBAB4_4144		BcerKBAB4_4713	*ldh*	ATCATAAAATTAATTAATGAATTTGTTATTATTTT
BcerKBAB4_4144		BcerKBAB4_0326		AAGATAAAATGTTTCTTAACACGTTATTTATTTTG
BcerKBAB4_4144		BcerKBAB4_0325		GTTTTATTTATTGCACAATTCTTTGTAAAATAGAA
BcerKBAB4_4144		BcerKBAB4_0322		AATTGACTTTTTTCTTAAAATTCAATATGTTCATA
BcerKBAB4_4144		BcerKBAB4_0306		TATATAAAATGCAGAAAATTCACTTTATAATAAAG
BcerKBAB4_4144		BcerKBAB4_0305		AACGTTATATTTCTCTTTTGTAACATATCTTTTCC
BcerKBAB4_4144		BcerKBAB4_5199		TATGGAATATATTTCTTTTATTGTTTATATTAAAT
BcerKBAB4_4144		BcerKBAB4_5180	*eutD*	GCTTTATTTCTATGCTGAATTTTATGAAAATATAA
BcerKBAB4_4144		BcerKBAB4_5164		TTTTCTTAATTGCACTATTCTTCGTTAAAATTATG
BcerKBAB4_4144		BcerKBAB4_5154		AGTAGCAGTACAAAAACAAACATGATACAATCGGA
BcerKBAB4_4144		BcerKBAB4_5143		GAAATAATTTTTTACACAACAACATATTGTTTTTT
BcerKBAB4_4144		BcerKBAB4_5136	*pyrG*	AATTTTCTTTTTTGTTTTTTGTTTAGAAAATAAAA
BcerKBAB4_4144		BcerKBAB4_5114	*glyA*	TCTTTTATGATATAGAAAGTCATGTTAGAATGTAG
BcerKBAB4_4144		BcerKBAB4_4698		TAATTCACATCTTTTTCGTAGTAAATCACAGTTGC
BcerKBAB4_4144		BcerKBAB4_4695		ATGTGGAACAACCGGAAAAGTTTGATACAATAGTA
BcerKBAB4_4144		BcerKBAB4_4694		ACTACAGTATAGTCCAGAGAAAGAAAGTACTTTCC
BcerKBAB4_4144		BcerKBAB4_0290		AAAATAATATAAAATTTTTAAATAATTAAAAATTT
BcerKBAB4_4144		BcerKBAB4_4629		TATATCACATGGTTTTAAACTTTTTGAGTTTTTCC
BcerKBAB4_4144		BcerKBAB4_0269		TATTCCTCCTTTGAAATTAAATCGTTATAATATAG
BcerKBAB4_4144		BcerKBAB4_4620		GAAATAAAATGTATTAAAACATTAACTACTTTTGT
BcerKBAB4_4144		BcerKBAB4_0247	*guaA*	ACTTCATTGTTAGAATCTAGGAAGATAATATAAAA
BcerKBAB4_4144		BcerKBAB4_5071		ACTATGACTCTATAGGTGGTTTTTTTATTATTAAT
BcerKBAB4_4144		BcerKBAB4_5058		TAATTATTATTTTTCTTATGTCTCAATATATTTTT
BcerKBAB4_4144		BcerKBAB4_5056		GGTATACGAGCAATCGTATACCTTTTATAATAAGA
BcerKBAB4_4144		BcerKBAB4_4595		GTATACATTATTTATATAATTGTGTTATACTATTG
BcerKBAB4_4144		BcerKBAB4_0188		ATTCAAAATGATGTTAACTATCATTTATAATGATA
BcerKBAB4_4144		BcerKBAB4_4520		ATTAGAATATTATGGATAAAACTAATATGCTTAAT
BcerKBAB4_4144		BcerKBAB4_4494		TAGATAATATTACTGTAATAAATTTTAAAAGTCAT
BcerKBAB4_4144		BcerKBAB4_4493		TACTGAAAATTTTAAATAATGTCATTATAATAGAT
BcerKBAB4_4144		BcerKBAB4_4488		TTTGACACGCCTATATAAATGATGGTATAAAGAAA
BcerKBAB4_4144		BcerKBAB4_4485	*rpsD*	AAAATAAAATGCTTTTATTTTTGTAAAAATTTCCT
BcerKBAB4_4144		BcerKBAB4_4484		TCCTTTAAAAATGTTTTTATTTTCGTAAAATAAAA
BcerKBAB4_4144		BcerKBAB4_4480		TAATGGAATAAGTGGAATATTAAGAAATATTACAA
BcerKBAB4_4144		BcerKBAB4_4479		TGTGTAATTTTGAAGTCGATGTTGCTATAATGAAT
BcerKBAB4_4144		BcerKBAB4_4476		GGTTCAAAATTTACTGAAAATTCTTTATTATATAA
BcerKBAB4_4144		BcerKBAB4_4468		AACTGAATTTTTTTGAAATCTGTAATATATTTTTT
BcerKBAB4_4144		BcerKBAB4_4453		TTTGTCAAATATTGAGATATTATGTTAGAATATAA
BcerKBAB4_4144		BcerKBAB4_0097	*rpoB*	CCTATAATATAGTATTTTAGTTTTTTCGCAACTGC
BcerKBAB4_4144		BcerKBAB4_0088		ATTGACGCTCTTTATCTTTTTACTGTATAATATTG
BcerKBAB4_4144		BcerKBAB4_4424		TGTTCACAAAAGTCAGATAATTGTTTATAATAGGA
BcerKBAB4_4144		BcerKBAB4_0064		GAGTGAATGACTAGAAAACTTCATGTAAAATAAGA
BcerKBAB4_4144		BcerKBAB4_4419		TTTTAAAAAATGAAAGTTAGAATGATATAAAAGTA
BcerKBAB4_4144		BcerKBAB4_4412		TGTTGCAAAATGAAAATAAACAAAGTATACTAACA
BcerKBAB4_4144		BcerKBAB4_0057		TATTGACTTTAAAATATATCTGATGTAATATACTA
BcerKBAB4_4144		BcerKBAB4_4407	*thrS*	CATATAATTACCTACATAAATGAAATATATTGAAA
BcerKBAB4_4144		BcerKBAB4_4406	*infC*	TATTGCAAGTATGCTGGTTGTTTGTTACAATATTT
BcerKBAB4_4144		BcerKBAB4_0044	*glmU*	GTAAAACATCTTATAAGAATTATGGTAAAATTTAA
BcerKBAB4_4144		BcerKBAB4_0017		AAATTAACATTTTACCCCAAGAAATATTAAAGTGT
BcerKBAB4_4144		BcerKBAB4_0001	*dnaA*	CATTGCTATAGCTGCTTTTTTTTGATATTATAGTT
BcerKBAB4_4144		BcerKBAB4_3965	*argC*	AATTGACTTTATAATTATGCAATATTATAATCAAA
BcerKBAB4_4144		BcerKBAB4_3922		TATTAACAAACAATAAATACTTTTTGAGAATAATT
BcerKBAB4_4144		BcerKBAB4_3918		TATTTTTATATTAGGAAATATGTTATAAAATGATT
BcerKBAB4_4144		BcerKBAB4_4392	*pheS*	AGTTGCGAATCGATAAAAAACTTCTTATAATAAGT
BcerKBAB4_4144		BcerKBAB4_4354		GATTGTAACGTTGTTATGACTATAATATAATGAAA
BcerKBAB4_4144		BcerKBAB4_4346		CTAATAATATATTGTAAAAAGTAAATATTTTTATC

BcerKBAB4_4144		BcerKBAB4_4311	*hemA*	TTGGTTATATTGCTTTTTCTTTATGTTACAGTTTA
BcerKBAB4_4144		BcerKBAB4_4303	*valS*	GTGATAACATATATTATTAATTGTTAGTGTATTTA
BcerKBAB4_4144		BcerKBAB4_3879		GATTGAATCGCTTACAATAGTTATGTATAATAACT
BcerKBAB4_4144		BcerKBAB4_3865	*mtnW*	AATTTAAAAAAGTTTTTCAGTAACTGTTGTGTTTT
BcerKBAB4_4144		BcerKBAB4_3864		TCTTTACAAAATACTGGAAAGATTATATCATTTGT
BcerKBAB4_4144		BcerKBAB4_3863		TGTTTACTATATTAGAAAGGTCATAAAACATTTCT
BcerKBAB4_4144		BcerKBAB4_3862	*mtnK*	GAAGTAGTATTTCAATTTTTAAGATTTTTAATATT
BcerKBAB4_4144		BcerKBAB4_3835		TAAATAATAATCAACAATTGTTTATTAATTGTTTC
BcerKBAB4_4144		BcerKBAB4_3819		TAAAGATTTCTATCAAATAAAGTGATAAAATGAAT
BcerKBAB4_4144		BcerKBAB4_3809		AAGATTACATATTTTCATACGTTAATTTAATTCCT
BcerKBAB4_4144		BcerKBAB4_4277		TGGATATAAATGTATCAAAATTTTATTACTGTTCT
BcerKBAB4_4144		BcerKBAB4_4276		TCTTGTCATTATTTTAAAACTATGTAAATATAGGT
BcerKBAB4_4144		BcerKBAB4_4245		AATTTCTTATAATACTATATTATGCTATAATTTCA
BcerKBAB4_4144		BcerKBAB4_4222		AATTGACATTTTTAGAAAGAGAGAATATCATACTG
BcerKBAB4_4144		BcerKBAB4_3799		AATGTATAATAAGATTCTCTTTTTTATTCAATTTT
BcerKBAB4_4144		BcerKBAB4_3772		CTTTGCTATTTTATTATTTTTTATATATCATAGGA
BcerKBAB4_4144		BcerKBAB4_3734		TTGTTCCATATATTGAGTTGTATGGTATAAATAAA
BcerKBAB4_4144		BcerKBAB4_3732		TATATATTAAGTTAATAATACATATGTTCATTTTT
BcerKBAB4_4144		BcerKBAB4_3722	*ileS*	GCTTGACGTTTTTCTCATTATTACATATAATTTTA
BcerKBAB4_4144		BcerKBAB4_3718		AAAAAAATATGGTAATGAGAGTGCGTTCTTTTTTT
BcerKBAB4_4144		BcerKBAB4_4194		TGTTTAATATTGAATTTAAATGTATTCAATTGTTT
BcerKBAB4_4144		BcerKBAB4_4170		CATTGAATCTTTGCTGCTCTATTGATATAATCAGT
BcerKBAB4_4144		BcerKBAB4_4110		GAAATAATAAAATAAATACATCGATTTTCCTTTAT
BcerKBAB4_4144		BcerKBAB4_4100		ATTTACAATAGTTAGAAAAATAAACTATAATGAAA
BcerKBAB4_4144		BcerKBAB4_4097		CTCTAATTTTTCTTGTGCATTTTTTTATGATATAT
BcerKBAB4_4144		BcerKBAB4_4023		TGTATAAAAAAACACTAAAATAGCCTTTCATTCTT
BcerKBAB4_4144		BcerKBAB4_4008		GTTTGTCAAGAGGTTGCTGAACGAGTAAGATAAGA
BcerKBAB4_4144		BcerKBAB4_3570		CGTTGCAATTTTTCTAAACTTGAGGAAAAATATAT
BcerKBAB4_4144		BcerKBAB4_3564		GAAGGATTTTGTTTAGGAGCTATGGTATAATATAA
BcerKBAB4_4144		BcerKBAB4_3550	*recA*	GTTGGCAAATTGAATTGAAAATAGGTATAATAAGA
BcerKBAB4_4144		BcerKBAB4_3531		GATAGAAAAAAATCGAAACTTAAGCTATAATTACT
BcerKBAB4_4144		BcerKBAB4_3508		GTTTGGAGTTTGGTGTATATAACGTTATCATATAA
BcerKBAB4_4144		BcerKBAB4_3506		TCTTGAAATAATTGCATATTAAATATATACTTAGT
BcerKBAB4_4144		BcerKBAB4_3489		GATTGTTAAATTCAGAATATTTTGATATATTTTAA
BcerKBAB4_4144		BcerKBAB4_2936		AAAATTAAATCATAAAAGAGACGTTCAATATTTAG
BcerKBAB4_4144		BcerKBAB4_3348		TCATAATTCAATTGAAATTTTTCTATATAATAGAA
BcerKBAB4_4144		BcerKBAB4_3331		ACTTGAAAGAACAAAAAATGTTTGTTAAGGTAGGG
BcerKBAB4_4144		BcerKBAB4_3302		TATTTACTTATTTAGAAAGTTGTAATAAGATATTA
BcerKBAB4_4144		BcerKBAB4_2856		TTAATATAAAAGACTTAAAAGAAAATTAAAATTTC
BcerKBAB4_4144		BcerKBAB4_3287		TCTTGTCTACAATGGTAGATAGGAGTATTTTTTTA
BcerKBAB4_4144		BcerKBAB4_3257		ATTTGAATTTTCAGCATTTTTTATTTATGATTTTA
BcerKBAB4_4144		BcerKBAB4_3213		AAAGTAACATACTTCCAAGACTATAGTACATTTTA
BcerKBAB4_4144		BcerKBAB4_2792		AAACTTATATAGTATCTAAACTTTAATTTATTTGC
BcerKBAB4_4144		BcerKBAB4_2705		CGTTTTAATTTTCATAAAAAAATGATAAAATGTAA
BcerKBAB4_4144		BcerKBAB4_3173		TAAGTTACTTATTCTAATTATTACAACTTCTTTAC
BcerKBAB4_4144		BcerKBAB4_3168		TAGTTAAAAATTAATCAAACTAATATATAATAATT
BcerKBAB4_4144		BcerKBAB4_3117		TTTAGTAAATTAATAATTAATCTGGTAAAATAAAA
BcerKBAB4_4144		BcerKBAB4_2639		ATTTATATAAAATAAATTTATTTGTAAAAATATTT
BcerKBAB4_4144		BcerKBAB4_3057		ACCATCCTTTTTTAAATGTTTCCCATAAAATAAAG
BcerKBAB4_4144		BcerKBAB4_2591		ATTTTTAAAGAAATCTTATATATGTCATTTTTATG
BcerKBAB4_4144		BcerKBAB4_2563		TATTGAAAACGCTTTTATTCAGTTTTATAATAAAA
BcerKBAB4_4144		BcerKBAB4_1990		CTTTTTTAATTTTCTATAATGGAAATATAATTCTT
BcerKBAB4_4144		BcerKBAB4_1981		ATGATAATAAACGTGTAAACGTTCTTCTTCGTTCA
BcerKBAB4_4144		BcerKBAB4_1975		TCTTAAACTAAAAAACTTATATTCGTATAATATAA
BcerKBAB4_4144		BcerKBAB4_1971		AAGTAAATATTTGAATAGAATGCTAAATCCGTTTA
BcerKBAB4_4144		BcerKBAB4_1937		TATTTATTAAATTTTCTAAATGTTGTATCATCATT
BcerKBAB4_4144		BcerKBAB4_1861	*nadE*	TATTGTATATTCGTTAGTTTTGCGAAATAATAAAG
BcerKBAB4_4144		BcerKBAB4_1809		CATATAATATGTTAAGCTAATTGCTTATTTTGTGA

BcerKBAB4_4144		BcerKBAB4_2245		ATAGTAATAATATATTTATTAACTTAACTTGTTAC
BcerKBAB4_4144		BcerKBAB4_2241		ATTTTTCAAATCATTAAATTTCTTTTAAAATTAAG
BcerKBAB4_4144		BcerKBAB4_1798	*ldh*	TATTTAATATAGACTTCTTCAAAATAATTATTTTT
BcerKBAB4_4144		BcerKBAB4_1752		TATTTACATTTGTTCAACTAAGGGTTAGAATGAAG
BcerKBAB4_4144		BcerKBAB4_2193		AAAATAATACTTTACCTCTCTACTTTTTACTTTCT
BcerKBAB4_4144		BcerKBAB4_2175		TATATAATATAAGCTTTATGAATAATTTAGAATTT
BcerKBAB4_4144		BcerKBAB4_2156		AAATTAATACATTTTATTATTAAAATAATTTTTAT
BcerKBAB4_4144		BcerKBAB4_2153		AAAATAAAGTGGAAACTAAATATTTATTCTTTTAT
BcerKBAB4_4144		BcerKBAB4_2117		CAATAAAAATATATATACATTTCATCAAAATAATA
BcerKBAB4_4144		BcerKBAB4_2108		AAATGTTTTTTTGATGTATTAGGGCTATGATACAA
BcerKBAB4_4144		BcerKBAB4_1675		ACTTGATTCCCTGAAAAATTTAAATTATTATAAAA
BcerKBAB4_4144		BcerKBAB4_1664		AATAAAATAAAGAAAAAATATTTTTTCACAGTTAC
BcerKBAB4_4144		BcerKBAB4_2099		ATTTGAAATGAATGCTATATTCCATCAAAATAAAG
BcerKBAB4_4144		BcerKBAB4_2086		GTAGTATTTTGAAATTTCTTAATGTTATAATGGTA
BcerKBAB4_4144		BcerKBAB4_2014	*ileS*	GTTGACAATTAATTGTTTAATGGCATAAAATAACT
BcerKBAB4_4144		BcerKBAB4_1532		TATTAAAAAAGATATGTTAATTTTTCATTATGAAG
BcerKBAB4_4144		BcerKBAB4_1490		GCTTTCCAAAATTTAAAAAGTGCGTTATAATCCTT
BcerKBAB4_4144		BcerKBAB4_1474		AATTTACACTTTTTGATGAATTTTATTTATTAGAT
BcerKBAB4_4144		BcerKBAB4_1447		TATTGACCTATGCAAGAGGGTATTATATCATGATA
BcerKBAB4_4144		BcerKBAB4_1413		ACTTTTCTAAAAGCAATAAAAGGACTATAATAATA
BcerKBAB4_4144		BcerKBAB4_0988		TCTTTATAAAACATATCTTTTTCGTTATTTTTTGC
BcerKBAB4_4144		BcerKBAB4_0947		TAAGTAAAATGGAGATAGTGAAAAGCTTTCGTAAA
BcerKBAB4_4144		BcerKBAB4_1399		AAAATAAAAACAGTTTAAAAGAAGAAAAAACTCTT
BcerKBAB4_4144		BcerKBAB4_1378		TTTGTAAAATAGCTTTTCAAATGTTTTTGAATTGT
BcerKBAB4_4144		BcerKBAB4_1372		TTTTTTATATGTATTTCTTTATTTTTTACTTTTTT
BcerKBAB4_4144		BcerKBAB4_1355		TGTTGTCAGATTTTAAATTTTGATTTATGATTTTT
BcerKBAB4_4144		BcerKBAB4_1344		GGATTGAAAAAAAACAGAATTGTGATAAAATACAA
BcerKBAB4_4144		BcerKBAB4_1319		AAAATAAAATATTTACAAAATTTTTTCCCCCGAAT
BcerKBAB4_4144		BcerKBAB4_1311		ATTGTCAAAATAGGTATTTTTTCGGTATAATTAAT
BcerKBAB4_4144		BcerKBAB4_1276		AAAGTGAAATACATACTAAGAAAACTTAAAATTTA
BcerKBAB4_4144		BcerKBAB4_1275		AATTAAAAAAATCTGTATTTAATGATACAATACAA
BcerKBAB4_4144		BcerKBAB4_1223		TTTTTAATTGTAGGTTGGGTTGGTGTAAAATTAAC
BcerKBAB4_4144		BcerKBAB4_1206		GTTTTTTGTTTATGGGGAGATGTGATAAAGTATAA
BcerKBAB4_4144		BcerKBAB4_0779		TAGATGTTATAACAATAATACCTTCTTCATGTTAG
BcerKBAB4_4144		BcerKBAB4_0778		TATTTAAAATTGTTCTCTTTTAGTACTACAGATAT
BcerKBAB4_4144		BcerKBAB4_0770		CGTTGGAAAAATAATGCATTTGTACTATAATGATA
BcerKBAB4_4144		BcerKBAB4_0756		GAAATCTTATTTTATTAGAAAATAGTGTTTTTTAC
BcerKBAB4_4144		BcerKBAB4_0747		GATTTTTTAACACATGCACTACATATAAAGTGAAA
BcerKBAB4_4144		BcerKBAB4_0701		ATCTTGATATGCTATGTTGTTGTCTAAATCTTTCT
BcerKBAB4_4144		BcerKBAB4_1166	*sucA*	TTTGGTATTATGTAAATTATTGTAATAACATAAGA
BcerKBAB4_4144		BcerKBAB4_1144		TGTGTACAAAAAATTAAATATGTTAAATCGTTTAA
BcerKBAB4_4144		BcerKBAB4_1142		TATGTAAATATAATAAATACTCTATTAAAATTAAG
BcerKBAB4_4144		BcerKBAB4_0631		TAACTTAAATGTAACAATTCTAATTTAAATGTTAT
BcerKBAB4_4144		BcerKBAB4_0608		TGTAGCATAACAGATAGAAATGTTTTATAATTTGA
BcerKBAB4_4144		BcerKBAB4_1099		ATTTCCCTTCTGGAGAATCTTATCATAAAATGAGA
BcerKBAB4_4144		BcerKBAB4_1097	*spxA*	AAAATAACATACTCTATTTACCATTTTTTCTTTCT
BcerKBAB4_4144		BcerKBAB4_1082		TCTAGACAAAAGTACATTTATGTTTTAAAATTAGT
BcerKBAB4_4144		BcerKBAB4_1077		TATTGTATAAAAAAGTCAGTTGGCGTATAATCAAA
BcerKBAB4_4144		BcerKBAB4_1059		AGTTGATTTTTCTTTTTATTTCGAATATGATATGA
BcerKBAB4_4144		BcerKBAB4_1053	*rocD*	GTTTCGCTTTTTTATATTTAAATGCTATAATATTT
BcerKBAB4_4144		BcerKBAB4_1008		ACTTGCTACTTTAATTTTCTTTTAATATAATTAAT
BcerKBAB4_4144		BcerKBAB4_4987		TTTTGACTGTAGTGGATTTATTTCTTACATTTATG
BcerKBAB4_4144		BcerKBAB4_4982		TTTTTAATTAGGAGAAAAATTGGCATAACATAAAT
BcerKBAB4_4144		BcerKBAB4_4969		AATGTAATATTCTCTCAAAATTATTATTCAAATTA
BcerKBAB4_4144		BcerKBAB4_4944	*clpP*	GTTTGACCTTCATTGACCATAATTGTATTATAAGA
BcerKBAB4_4144		BcerKBAB4_4932		AGTTGAATAAGTTTCTGTCTCCTGTTACAATAATA
BcerKBAB4_4144		BcerKBAB4_0575		CGTTTACATTCTTGTCACAAAGTTGTATAATTTTC
BcerKBAB4_4144		BcerKBAB4_0544		TGTTGACTAACTTATATATACGAGTTAATATGTAA

BcerKBAB4_4144		BcerKBAB4_0523	*aspA*	AAGATAATATATTTGAAATTATTTTTCATATATAT
BcerKBAB4_4144		BcerKBAB4_0506		CGTTTTCATAATTTTTAAGTTATGCTAGAATAAGG
BcerKBAB4_4144		BcerKBAB4_4898		TGTTGACAATTTGATTTGGATTCAATATAATGGCT
BcerKBAB4_4148		BcerKBAB4_4595		AATAAATAAATAGTATACATTATTTATATAATTGTGTTATACTATTGATGGGGATTA
BcerKBAB4_4245		BcerKBAB4_4222		TACTGATAAAACCGATAAGAAAAAGGGGAATTTTTATG
BcerKBAB4_4245		BcerKBAB4_4100		TTCCGCAATTCCCTAGTTGACTTATAGGTTTTATTACC
BcerKBAB4_4245		BcerKBAB4_2881		TTGTTCAAAAGCATTGTATTTTTATCGGAATAGTGAAA
BcerKBAB4_4278		BcerKBAB4_4479		GTCATACGGACGTTTACCGGCCCCTTTTACATGTGAAAATATATTTTTGCCTTGGAAACTCATCAGAATCGCATTAATGTGCATAAACACTATACTAATATACTACAC
BcerKBAB4_4278		BcerKBAB4_4277		TGTTGCCTCCCACTTTACACTTTCCAACTGTCTTGACACCTATATTTACATAGTTTTAAAATAATGACAAGAGTTTTTTTACAATTCTTATTTTCACTACCATTCTAA
BcerKBAB4_4278		BcerKBAB4_4276		AGTACATTAAAGGAGAGAAGTCATTCTCACTGTAAACAAAAGCAAAAATCTTACCATCACTTTTATTCTTAACATTTTTTTGAGAACAGTAATAAAATTTTGATACAT
BcerKBAB4_4353		BcerKBAB4_0779		TTATATGTCTTCTATTTGACTTTTAAAATCTTCTCGATA
BcerKBAB4_4353		BcerKBAB4_1008		AATTTTCTTTTAATATAATTAATTTAGAATTTTTAAATA
BcerKBAB4_4353		BcerKBAB4_4835		CGACTTATGTCGAAAAATTGAATGAGCATTCATTCAAGA
BcerKBAB4_4353		BcerKBAB4_5143		ACAACTGACTTACGAGTGAGTAATATATTACCCTTATCC
BcerKBAB4_4353		BcerKBAB4_4694		GAAATTTTAGCAAACAGCCCAATGGCACTTCGTTTCCTA
BcerKBAB4_4353		BcerKBAB4_4354		AGTACTTACTTGTAAGTAAATAAAAAATACATAAACTTC
BcerKBAB4_4353		BcerKBAB4_3819		TCAAATAAAGTGATAAAATGAATGTGTATTCATTTTTTG
BcerKBAB4_4353		BcerKBAB4_3117		ACTACTTATTCCTCAAAAGATTATATATTCCTCTTTATC
BcerKBAB4_4419		BcerKBAB4_3374		AATCTTTACACATTCTATACATT
BcerKBAB4_4419		BcerKBAB4_0631		ACATTGTTAAGATTAAATTTACA
BcerKBAB4_4419		BcerKBAB4_5259		AGTTATTTTACAATAGAATGAGG
BcerKBAB4_4419		BcerKBAB4_4419		AAAAGCATTAGATAATTTTTAAA
BcerKBAB4_4419		BcerKBAB4_3954		TTCTTTTCAAAATTAGAATAAGA
BcerKBAB4_4419		BcerKBAB4_3057		AATTTTCATACCATTGTTTATTA
BcerKBAB4_4419		BcerKBAB4_1399		ACGTTAGTAATACAAATTGAAAT
BcerKBAB4_4419		BcerKBAB4_4194		AAAATTCGATAACAAATTGTTTT
BcerKBAB4_4506		BcerKBAB4_1319		TATTTCGCAACTAT
BcerKBAB4_4506		BcerKBAB4_0770		ACATTCGCATAAGG
BcerKBAB4_4506		BcerKBAB4_4468		TACTTTGCAAAAGT
BcerKBAB4_4506		BcerKBAB4_0947		TGACACCGCTTTCA
BcerKBAB4_4506		BcerKBAB4_1355		TCCTTAGCTAAAGT
BcerKBAB4_4506		BcerKBAB4_1311		TCTTTCGTTAAAAT
BcerKBAB4_4506		BcerKBAB4_1276		TGATTTTGCTTGCA
BcerKBAB4_4506		BcerKBAB4_1275		AGTTTCGAAAAAGA
BcerKBAB4_4506		BcerKBAB4_1201		AAAAAGTGCTTTAA
BcerKBAB4_4506		BcerKBAB4_0756		TCTTTAGCAAAAGT
BcerKBAB4_4506		BcerKBAB4_1008		ATTATACGCATACA
BcerKBAB4_4506		BcerKBAB4_4934		AAATACGCAATAAT
BcerKBAB4_4506		BcerKBAB4_0544		ACATTTGCCATAGT
BcerKBAB4_4506		BcerKBAB4_0474		ACTTTCGCGTTAGT
BcerKBAB4_4506		BcerKBAB4_0409		TATAACCGTCTTCA
BcerKBAB4_4506		BcerKBAB4_5180	*eutD*	AGATAAGGCTTAAA
BcerKBAB4_4506		BcerKBAB4_4694		TCTTTCACAAAAAA
BcerKBAB4_4506		BcerKBAB4_0290		ACATTTGCAAAAAG
BcerKBAB4_4506		BcerKBAB4_4629		AAATTTGCAAAAAA
BcerKBAB4_4506		BcerKBAB4_4494		TCCTTCGCAAAGGT
BcerKBAB4_4506		BcerKBAB4_4493		TGGAAACGCTTCCT
BcerKBAB4_4506		BcerKBAB4_4476		TGAAAATTCTTTAT
BcerKBAB4_4506		BcerKBAB4_4424		ACCCTCTTAAAAGT
BcerKBAB4_4506		BcerKBAB4_4354		TTTTAACGATTATA
BcerKBAB4_4506		BcerKBAB4_3844		ACTTTTGCAAAATA

BcerKBAB4_4506		BcerKBAB4_4008		ACTTTTGAAACAAA
BcerKBAB4_4506		BcerKBAB4_2796		TCCTTTGCAATGCT
BcerKBAB4_4506		BcerKBAB4_3173		TGAATGCGCTTTCT
BcerKBAB4_4506		BcerKBAB4_2578		ACTTTCACTACATT
BcerKBAB4_4506		BcerKBAB4_2563		TGAAAACGCTTTTA
BcerKBAB4_4506		BcerKBAB4_1809		TGGAAACGCTCGAA
BcerKBAB4_4506		BcerKBAB4_1752		TCTTTCGGAAAATA
BcerKBAB4_4506		BcerKBAB4_2156		TGTAAAGGCTTACA
BcerKBAB4_4506		BcerKBAB4_2117		ACTTTCTAAATAGT
BcerKBAB4_4506		BcerKBAB4_2108		CGAAAACGATTTTA
BcerKBAB4_4506		BcerKBAB4_1664		ACTTTCGCAAGCGT
BcerKBAB4_4506		BcerKBAB4_1610		ACTTTCGTCCAATT
BcerKBAB4_4506		BcerKBAB4_2046		ACTTTAGCAAAAAC
BcerKBAB4_4791		BcerKBAB4_4791		GACTCTAACGTTGCGTCATA
BcerKBAB4_4932		BcerKBAB4_4932		CCTAAGTGGGACACAAAATGTCACTACGGGACATAAA GTGACCACG
BcerKBAB4_4934		BcerKBAB4_0305		AGGAAGTTGGCATACATCTTGCAATATAAAGAGAAA
BcerKBAB4_4934		BcerKBAB4_0290		CAGGATCCTTTTTAATTTTTGCATTTTTATTAATTG
BcerKBAB4_4934		BcerKBAB4_4007		TTAAAGTTGGCACGGTATTTGCTTAATGAATAGACG
BcerKBAB4_4934		BcerKBAB4_3257		CAAAAACGCGCAGGATTTTTTGTAATTATATAGAAT
BcerKBAB4_4934		BcerKBAB4_3173		TTGCAAATTTTTTATGTTTACAATTCTGTCATCTAA
BcerKBAB4_4934		BcerKBAB4_2578		TAGAATTTGGCACAGTACTTGCAATATAAAAGATGA
BcerKBAB4_4934		BcerKBAB4_2245		TTAATCATAGGATATACCTATCATTATTATATAAAT
BcerKBAB4_4934		BcerKBAB4_1413		AATGAAAAGATTTTCGTTATTTTCCTGATATTATTA
BcerKBAB4_4934		BcerKBAB4_1276		TAAATATTGGCATGGAAATTGCTTATAAATATATAA
BcerKBAB4_4934		BcerKBAB4_0756		AAATCGTTTTCACAAACATTTCATTTGAAAAATATT
BcerKBAB4_4934		BcerKBAB4_1053	*rocD*	AGAGTATTTACAAACTTCTTGTAATAACAAAAGGGG
BcerKBAB4_4969		BcerKBAB4_2936		GTTAAAGACTATTTGTAATATTATTA
BcerKBAB4_4969		BcerKBAB4_4969		AGGGATAGTAGTCTTTATGAATACTA
BcerKBAB4_5150		BcerKBAB4_2792		TAATTTGTAAGGAATGA

Table 1: Binding sequence predictions for *Geobacillus kaustophilus* HTA426. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
GK0001	*dnaA*	GK0001	*dnaA*	TACAATGGGTGTCTACC
GK0040		GK0257		AAAAAACACGAACATTTGCTTTGGTGAAATACTAAATGTTCGATTTTTTCC
GK0040		GK0255		CTTATTTTTGCTTGTTTTCTGCTTGTAAAGGTTTCTTTACAAGCAGAAAAA
GK0040		GK0040		TTAAACACGTACATTTTTGTTAAAATATATTTATAATATTCGGTTTCAGGA
GK0040		GK3475		AACTATTTGCTTATTTTTTCTGCTATAAAAGAAAATTACAAGCACAAAGTA
GK0089		GK3109		AAACGGACAGTAGCTATATTTACATAACTTAGGGAAA
GK0089		GK2483	*dnaG*	AAAGGAAAATCGGCTTGCTGCCGAGAATAAATACGAT
GK0089		GK2387	*spo0A*	ATTAAAAATAAAAAACTTTTTATTTTCCTAAAAAACG
GK0089		GK1124	*ftsA*	ACAGGGAAAGTGTTTTCGCGTGTTGAATTATATAGGG
GK0089		GK0250	*fumC*	CCGGGTAGCAAAGAAGTAACAGCTTGGTAAGGGGAAA
GK0150		GK0150		AAATAAGCGCTTTTTCCTTAGTTTTAAATGTGCTA
GK0150		GK2791		GAGTTTCCAATTTTTCGCCGCTTCCACCATATGAT
GK0242		GK2661		AGCAGGTACTGTTCACCTATCGCTTTTTCGCAA
GK0242		GK0475	*ldh*	TAAAATAAAATGTGAATATATTCACAAAAAGGA
GK0242		GK0373		GGCGTAAATACTTGTTGCGTACTTATACTACCA
GK0332		GK3469		CTCGGTTCTATTGTTATACTTTCTCGACCGCGCGTACTGGCACTGCGTTCACAAC CTTGTCTTTCCGAATCGCCAGCCGCGGGCGAAAGCAAACATACTATGGCTAGGC GCAG
GK0332		GK0332		AGGAACAAAAATAGTATGGTTTGTAGGCACTTTTTTGCCACTTTTTCGACACGCG AACCGAAGCGAGCGATTACCTTCCCTTCCTTAAAGTGGTAAAAAAGTGAAAGGT ACCA
GK0478		GK0478		AATATTTATAATATTTCACTA
GK0478		GK2647	*hemA*	CCCTTTTGGACATGTTTATGA
GK0478		GK2575		AATTTTAATAATATTTAATTG
GK0478		GK0902		AGTATTATACCTATTTTTTTA
GK0478		GK0817	*spxA*	ATCATGTTAAAGTCATTTCAA
GK0652		GK2020		ATAAATTTATTATAAAAAAG
GK0729		GK1951		CATGAACCAATCTTTTGAA
GK0729		GK2235		AGCGATGACATATTTTACG
GK0729		GK2037		AGCGCTTCATCATCTTGAA
GK0729		GK2000		AATGCAAAAGCGTGTTTGA
GK0729		GK1426		ACCTCCTATTAAAACACAT
GK0729		GK1411		GGGGCAAACATTTTTTCCA
GK0729		GK0710		TACCTATTTACCTTATAAGCCATAACTGACAGGGTAAAACATAGTTTCACATCGC TGTTTCACT
GK0729		GK0199		CCCTTTTACTAACTGATAA
GK0729		GK0188	*rocD*	GATGCAAAGGGCTTCTTCA
GK0729		GK2889		GATGGAGAAGTAGTTTGCC
GK0729		GK3198		AAATATTATAAGAAAGGAA
GK0729		GK1953		TTGTCTTCCTAAAAAGGTA
GK0729		GK2383		AAGGCATTTTTTGTTTGCC
GK0729		GK0196		AATGGAGATGTATTTTTCT
GK0729		GK0185		TATGCAAAATTATTTTTCA
GK0891		GK1409		ATAATTTATATTTATTTTACTAATGT
GK0891		GK1408		TGTAATCATTTTATTTATATTTAATA
GK0891		GK0891		TATTTCTACTTTCGTTAGTGTTACTT
GK1127	*sigE*	GK0148		TTTCATAACCGCCATGGCTGTCCGCATAAGATGTTATTAG
GK1127	*sigE*	GK0014		AGTAGTGGTAGTATCGTCCCTCTCTCAGGAAAAAAGTGCT
GK1127	*sigE*	GK2981		GCAGGCACTTCTTTTTCCATGCTTCATACATTGGAGTATG
GK1127	*sigE*	GK3340	*spoIID*	CTGTCATATATACTTGTCCGAGCCCATAACTATAGTAACG
GK1127	*sigE*	GK2732		GAAAAAAGAAAAAAACAGGCTTAGTACCGCAGTTAACACC

GK1127	*sigE*	GK2667		AGGAAATCGCAGGTACGCGATTTCCTAAAAAAATAAGATA
GK1127	*sigE*	GK2584		TGTCATGCTTGGCCATGGGCCTTGCATATATTGTATAAAA
GK1127	*sigE*	GK2510	*spoIIP*	CAGTTCTATCTTTTCCCGCTTGGCCATACGTTAAACTGAG
GK1127	*sigE*	GK2418		TTTTACATAAAAGGCAAAAGCCAAAATATAGTGGTAAAAG
GK1127	*sigE*	GK2286		TTCATCATATTTGTCCCATCGAGGCATACACTTGTACAAA
GK1127	*sigE*	GK2217	*spoIVA*	GAGTCATCTTCCCTTCTCTAGGAAAATACATTCTAGAGAA
GK1127	*sigE*	GK1654		GTATTGTACTTTTCAGAACATTCGAATATAATAAAGATAA
GK1127	*sigE*	GK2077		ATTGCTTTTTTAATGATCAATTTGCATGAAAAATCAACGA
GK1127	*sigE*	GK1597		CTTACTAAAACCGACTAGTCGGTTCAAATGATAGAACATT
GK1127	*sigE*	GK0970		CTCGTCTTTCTTGTCCGCTGCGGACATATATATGGAAGTA
GK1127	*sigE*	GK0906	*ansB*	TTTAAGTGTGAAAGAGGTATAAAATAGCAATTATAATACT
GK1127	*sigE*	GK1314	*spoVK*	GATGCGTAAACCGCCGCTTTTGTGAATAAACATATACAAA
GK1127	*sigE*	GK1305		AAGGCACATCGAACGGAGCAGGCGCATACAATGAACTATA
GK1127	*sigE*	GK0700		AATTCATCGAAATGACGGGGGGTGCATAAAATGGAATCAT
GK1127	*sigE*	GK1114		TTTGCATATTGAACCGAACGGGGAAAAACGAATGAAAGCA
GK1127	*sigE*	GK1100		TGGTCTAAAACGCCCCCCTATGCTCGTATATTAGGTTATA
GK1127	*sigE*	GK0641		GTGTTCTTTTTTTGTCCGTTTCGTCATAGTCATGAAAATG
GK1127	*sigE*	GK0603		TGAAAATAAAATTTTATCTTTGTTTAGATATTATGAAAGA
GK1127	*sigE*	GK1080	*ctaA*	TCCTCATCATGTGAACTTTTTTTGAATAATTTGTGTCAAA
GK1127	*sigE*	GK0574		AACGTATCAAAACGCCGTTTCCCTCATACATTGTGATAAG
GK1127	*sigE*	GK0487		TTGGTCAATTATTTTGAATTTTTGCATACGATAAAATAAA
GK1127	*sigE*	GK0486		GGCGAATGATACGGGACATCCTCGCATAAATTGTAATAAT
GK1128	*sigG*	GK3428	*acpD*	GATCTTTATATAATTATCTCTTGTTAGAGATATTTG
GK1128	*sigG*	GK3406		ATAAAAAATTTTTTTTCATACTGGTAAGAAAAAGCG
GK1128	*sigG*	GK2388	*spoIVB*	CGATTATATTTTCCCCGGTTCAGGCAAAATTAAAAC
GK1128	*sigG*	GK2311		AAAAAACAAAAAACCTTTTACTCGACAAAAAACATA
GK1128	*sigG*	GK2307		CCGTAAAAAACGGGTAAGAAAAGATAAAAATACTTT
GK1128	*sigG*	GK1624		TCCAATCACAAACGATCGTTCTTCCTTTGCTATGCG
GK1128	*sigG*	GK0972		ATGCATAAAAATATCGATCGATTGGAAGACTATGCC
GK1128	*sigG*	GK0937		AGTAATAAAACAGGCTAGGCGTTAACAAATGCGACG
GK1128	*sigG*	GK1128	*sigG*	TGTGCATATTTTTCCCTTCCAAGGAGATACTGAAAA
GK1128	*sigG*	GK0049	*spoVT*	CACGTATAGAACGTGGATTGTGAAGGATACTAATGA
GK1147		GK1148	*pyrP*	AAGCACCCTTTTAAGCGCAGTCCCGTGAGGCTGCAAAAGGGGCGGAATCGTTTC
GK1147		GK1147		ACAGGAACCTTTAAGTTCAGTCCTGTGAGGCTGAAAAGGGGTCGGAATGACAA A
GK1187		GK0834		TTAATTGTGGACCACGATTCTTATTTACTATTAACCCTCCTACTTGCTT
GK1187		GK0804		TATTGCGCATCCACGCTCTTTCGGCTAAAATTAGTACCAAGTAATAATA
GK1187		GK1187		CATTTCGTTTTTTGATATTATATATTACTTTTAGTACCTAGTCATAAAA
GK1215		GK0373		AGAGACTTCGAGAACTGGGCATCGA
GK1215		GK3131	*hag*	CCAAACCTATTAAACTTTTAAAAAA
GK1246	*sigD*	GK2889		CATAGATAAACTGCCCTTTCAAACCACATTCTTAACTTCT
GK1246	*sigD*	GK2772		TATGTTCAATATATTTTTCATAAGCCGGAAGCGAAAATTA
GK1246	*sigD*	GK3198		TATAACATAAGTAACCTTTCTTAAAAATATTATAAGAAAG
GK1246	*sigD*	GK3131	*hag*	CCTATTAAACTTTTAAAAAACAAACCGATATAAAAAGTGA
GK1246	*sigD*	GK2530	*motA*	GCAAAAATTACCAGCCGTGCAAATTTGACTTTTTTTATAC
GK1246	*sigD*	GK1951		AGAAAACTTTCTAGCTGATCTTTCCACCAAAGTAGGTACT
GK1246	*sigD*	GK2000		TTTCCGTATTATTATGTTAAACTGACGATATTAAATAAAA
GK1246	*sigD*	GK1426		TACTTTTTATATTACCTTTGAACAACCTTTGCAATTTCCA
GK1246	*sigD*	GK1411		TTATATAATTCACAAAATATTTCAACTATATTATAGTAGT
GK1246	*sigD*	GK0730		TATCGCTAAAAAAGCCGTAACAGACAAAAACCAAATCCTC
GK1246	*sigD*	GK1020		TGTCCTTTTATAATGAAACATAGTTCGACAAAAAAGAGGA
GK1246	*sigD*	GK0199		ATCTATCCAGAAGTACAACACCAGCCGACATTTCAAGAAA
GK1328		GK0005	*gyrB*	AAAAAAGTGCAAACTTTTGTCA
GK1328		GK2744		TCTTCCGGCGAAAAAGCAAGTA
GK1328		GK2675	*uvrC*	AAATGAAAACGTATCGTTCCAG
GK1328		GK3086		TGGTACGAATATGAGTTCGCAT
GK1328		GK2592	*ruvA*	CGGTATGAACAAGTGATCGAAA
GK1328		GK1751		TATCTTTGTCCACAAGAACAAA
GK1328		GK1328		TGAGCTTGGATACAAACAAAAA

GK1328		GK0564		TACCTTTGAAAACACACGTCCT
GK1328		GK0275		AAAATAGAACATATGTTTTGTT
GK1430	*gltC*	GK1431		ATCGCAAAATGAGA
GK1430	*gltC*	GK1430	*gltC*	AGAGTAAAACGCTA
GK1745	*treR*	GK3444		CAAAATCCAGAAAAAACAAAAAGTAAAGTGGT
GK1745	*treR*	GK1747		CAACTGTTTGAACATATATGTTCTATTTTACT
GK1907	*araR*	GK1907	*araR*	ATTACTATGCCTGTTTAT
GK2154		GK1359		GGTGGAGAAGCGGAGACCCA
GK2154		GK2153		TCAGGAGACAGGGAGAGACC
GK2279		GK1868		AACAATTTTCACACAAT
GK2279		GK1737		ATAAACTTTATAACAAA
GK2279		GK0788		AAATGCATTCCAACACT
GK2279		GK0768	*fnr*	AAATCTTTTTTAACACT
GK2308	*sigF*	GK3487		CATTCTTTTATTTTCCGTATACTGTTAATACTTTTG
GK2308	*sigF*	GK3463		CAGTTTGATTCCCTTTTTCTATTGTACCATCTTTCC
GK2308	*sigF*	GK3406		CAAGTTTAAAATAAAAAATTTTTTTTCATACTGGTA
GK2308	*sigF*	GK2949		AATGAATATTTCGATTTTGGATGGAGATAAATAGAG
GK2308	*sigF*	GK3376	*spoIIR*	CGCCGCTATGCTTTCCCCGGTCGTCAAAAACTATGA
GK2308	*sigF*	GK3339	*spoIIQ*	TGTGTATATCTTTTCCTCTTTCTGTTCAAAATGGTT
GK2308	*sigF*	GK2546		AATGCTTGTGAAAATCGTGGCTGTGAAATAAGAAGC
GK2308	*sigF*	GK2511		GCGCTAAAAACGGCCGACCGCCGCAAAAATACCTAT
GK2308	*sigF*	GK2413		AAGAACATCTTATTTTATTATTTGTATTTTTTATCC
GK2308	*sigF*	GK2388	*spoIVB*	AATATAAAAGGGGCCAAGTCCGTTTTAATTTTGACA
GK2308	*sigF*	GK2311		AAAAACAAAAAACCTTTTACTCGACAAAAAACATAT
GK2308	*sigF*	GK1624		GATGAATACAACCTCCATGTGCGTCAATGGCTAGAA
GK2308	*sigF*	GK0972		GATGCATAAAAATATCGATCGATTGGAAGACTATGC
GK2308	*sigF*	GK1128	*sigG*	GACGCAAAAAGACGTCGCTGATTTGCTCGGTATTTC
GK2317		GK1630		ACGAAGACGTTTACGTAAGAGTAACAGTTTATTTATA
GK2317		GK1459		ATATAGTAATAAAAGTGAGAATCATTGTCAACTATAA
GK2317		GK1335		AAATCAATCAGATGATGAAAATGATGAACAATCAACT
GK2317		GK0194		AACAATTACTATCACTAAAAGTAAAAGAATCAGGTTT
GK2317		GK2954		AAAAGCAAACCTTACTAAAAGCAGTGGCATAGAACCT
GK2387	*spo0A*	GK1126	*spoIIGA*	TTTTGACAAAACAT
GK2387	*spo0A*	GK0057	*spoIIE*	TTTTGACAAATTTC
GK2390		GK0199		TTCAAGAAAAATGGAA
GK2390		GK2889		TTTGATCAAAATAAAA
GK2390		GK2772		ACGCATGAAAATGGGA
GK2390		GK3198		TACTATAACATAAGTA
GK2390		GK1919		ATGAAACAAATAAACA
GK2390		GK2000		AAAAGTAAAAAAAGTC
GK2390		GK1426		ATGAACAAAGATAGGA
GK2390		GK1411		ATGTAAAAAAGTTGCA
GK2390		GK0730		ATTAGAAAAAATTAAC
GK2390		GK0790	*argC*	AAAATAAAAATATGTC
GK2422		GK1875		CACTTTGTTTATATGATAGACAAACAAACATGAAA
GK2482	*sigA*	GK3475		CATTGACTTTATGGCTGAAAACTGGTACACTGACA
GK2482	*sigA*	GK3474		GGATGATTTCATGTTCATAACATGGTAAATTAGTA
GK2482	*sigA*	GK3464		CATTTGCCGATGGCGCATTTCGTGATAAGATGAAA
GK2482	*sigA*	GK3461		TAAGTCTAATAATTTATATATTATTGAACTTTTGT
GK2482	*sigA*	GK3452		GTAAAAATATTTTATTAAAAAATACATATTTTTCT
GK2482	*sigA*	GK3444		CGTTGACAAAAACAGCAGGTTGCCGTATAATGCAA
GK2482	*sigA*	GK3415	*eutD*	AACATGTCATGGTGATATAGTATCAAACCGGTTTT
GK2482	*sigA*	GK3406		AAGTTTAAAATAAAAAATTTTTTTTCATACTGGTA
GK2482	*sigA*	GK3405		ATGGTCATACTTTTTTTTAAAAAATAAAATTTGAA
GK2482	*sigA*	GK3398		TATTGACTGAATGCTCATTCATTTGTACAATAGAC
GK2482	*sigA*	GK3389	*pyrG*	GCTTGACTTTGCCAGGGGAACCCGGTACAATGTGT
GK2482	*sigA*	GK3324		AGTTGTCTTCTTTTCGAAAACGGTCTATAATCGAA

GK2482	*sigA*	GK3320		GAAATAATATAGTATGTGTTTTCTTTCCCCCCTTG
GK2482	*sigA*	GK2889		TTGGTGAAAAGTTCCCTAAACTAGTTTTATTTTAC
GK2482	*sigA*	GK2876		ATAGATAAATGATTTCCTATTATATTATAATGGGA
GK2482	*sigA*	GK2850		GTATGGACTATTAATAAAAATGTGTTATACTATTT
GK2482	*sigA*	GK2807		AAGATAATATCAGCTAACTTTTGGTAAAGAGTTAA
GK2482	*sigA*	GK2806		AATTGAGAAATGGTTTTCAATCGACTATAATAGAA
GK2482	*sigA*	GK2805		AAGAGTATACTTTATATATGTAAGAATAATGGTAT
GK2482	*sigA*	GK2803	*tyrS*	TATGGTAATAAGAATGTATATATTTCATATGAGAA
GK2482	*sigA*	GK2802	*rpsD*	TGTTGACTTTATCCTCTAAAAATCATATAATAGCC
GK2482	*sigA*	GK3269		TTCTTTTCCGTTCAACAGGATTTGGTATAATAAAA
GK2482	*sigA*	GK3252		ATTTGAAAATTTCCTCTGTTTCCATTATAATTAAG
GK2482	*sigA*	GK3231		GCTTGATGGCGCACAAAAAATAGGATATACTATGT
GK2482	*sigA*	GK3218		GATATCTGATAGTATTTAGAACATCCAACCTTTTA
GK2482	*sigA*	GK3214		CCTTTACTTTTTTCGCCAAACACGGTTTATTGGTG
GK2482	*sigA*	GK2798		CTAAGAACATTTTTTTTATTTTTAAAACCAGTTTT
GK2482	*sigA*	GK2796		TAACTAATAGGTAATAAAATGGATTTTTTGTGTCT
GK2482	*sigA*	GK2793		TATTCAAAATAGAACGACTATTTTGTATAATAGAG
GK2482	*sigA*	GK2785		CAACTAAAACGTTTTCTTTCCCGGCTTGCAGTTTT
GK2482	*sigA*	GK2772		CAAGTTATATAAAAAGTATTCGGCCTTCGCTTTTA
GK2482	*sigA*	GK2759		TATTGAATTTTATTTTTATTTCAATTATTATTGAT
GK2482	*sigA*	GK2736		AGCTGACAAAAGTCTGATAATTGTTTATAATAATA
GK2482	*sigA*	GK2732		GTCCGAATCATGGCGTCAATTGTGGTATCATAAAG
GK2482	*sigA*	GK2726		TTTATACTATATGATAAATATGTTTATATTCTTTT
GK2482	*sigA*	GK2725		TGTTGCAAAAACCCCCTAAACAAAGTATACTATGT
GK2482	*sigA*	GK2719	*thrS*	GCTTGATTTTTCCTGCTGTTTTCGCTATAATGACG
GK2482	*sigA*	GK2718	*infC*	TCTTGCCAATTGAATGTTGATCCGCTATAATAAAG
GK2482	*sigA*	GK2707	*pheS*	CCTTGCCTTTGCCCACTCATTTGCGTATAATGAAT
GK2482	*sigA*	GK3198		TATTGTATTCATTGGAAAGAATTTTTATAATATTC
GK2482	*sigA*	GK3170		GAAATACAATGGTATATGCTACTCATTTAGTTAGG
GK2482	*sigA*	GK3151		ATAATAATATTTAGCTGTTGAAAAAAGGAAGTTTG
GK2482	*sigA*	GK3147	*comFA*	ATTTTTCACCATCTTTCTAACTTTTTACGAAAAAG
GK2482	*sigA*	GK3107	*secA*	AGTTGTCACCTATCCCTGTAAATGATATGATGAAT
GK2482	*sigA*	GK2690		TGATGAAAAATTAGAAAAAGTAAAATAAAATAGAA
GK2482	*sigA*	GK2673	*sdhC*	TCTTGACGCTTGTACAATGTGGGAGTACAATGAAT
GK2482	*sigA*	GK2647	*hemA*	GTTTCCCTTTTGGACATGTTTATGATAGAATGATC
GK2482	*sigA*	GK2638	*valS*	ACTTGGCAAAGATGATGGACAATGGTAGTATAATA
GK2482	*sigA*	GK2602		TAGATATAAATGTGTATAAATTTGATTACCGTTCT
GK2482	*sigA*	GK2601		TCTTGCCATTAGTTTAAATATGTGTAAATATAGAT
GK2482	*sigA*	GK3062	*clpP*	GTTTGACCTTCATTGACCATTCGTGTATGATTAAG
GK2482	*sigA*	GK3059		AGTTGAACGGCTATCGTCCTCCTGCTATAATGAAA
GK2482	*sigA*	GK3008		TATTGCGAAAATTATCAACATACTATATATTTAGT
GK2482	*sigA*	GK2575		TTTTTTATTTTCGGCGAAAAATGGGAAAGATGAAA
GK2482	*sigA*	GK2543		GTTGGAACAAGGGAAAAACATATGCTACAGTAAGA
GK2482	*sigA*	GK2541		TTTTTACAATTTCGAAATTTAACGTAGATATATAT
GK2482	*sigA*	GK2534		TATTGACAGATTTTATTTTTAAAAGTAAAATAATT
GK2482	*sigA*	GK2508		CATTGAATCTTCTTCGCGCTGTTGCTATAATCAAA
GK2482	*sigA*	GK2463		TATTTCATATGTATGCTTGTCCAGCGAAAATATGT
GK2482	*sigA*	GK2435	*comGA*	TTTTGCCATATATTCATAAATATGAAATCTTGTCG
GK2482	*sigA*	GK1953		GTTTGAGACCATTTGGTGTATATGATAAAAAGATG
GK2482	*sigA*	GK1919		AAAATAGTATGTTGTTTTTGTATCCTCTTCCTCCT
GK2482	*sigA*	GK1907	*araR*	AATGATACGGACAAATAAAAACTGTTATAATAAAA
GK2482	*sigA*	GK2387	*spo0A*	TTTTTGTTATAATTTTTATTTTTTGAAAAATAAAA
GK2482	*sigA*	GK2383		TTTGTCAAGGAAAGAGCAAATCAAGTATGATAAAA
GK2482	*sigA*	GK2337		GATATAATATGATATAAAATGACTTAGATAGGATA
GK2482	*sigA*	GK2314		GGTTGACACGGCAAGGATTTTGGAAAATTTTAAAG
GK2482	*sigA*	GK1875		GATAGACAAACAAACATGAAAGCGTTATAATGAAA
GK2482	*sigA*	GK1868		ATTTTTCATGCGATTATTCATGTTTTTTGTTAAAA
GK2482	*sigA*	GK1865		AAGGTAAAATTGTATTGTTTGTGAGACACGGTTCC

GK2482	*sigA*	GK2235		CCTTTGCATCCGGCCACAGAAGGTGTATACTATGA
GK2482	*sigA*	GK2204	*trpE*	GTTGACAAAAAAAATAGAAATGTAATATAGTGAAA
GK2482	*sigA*	GK1779		ATTTTCATATATGGTTGTTTTCGTTTTTCTTTTAC
GK2482	*sigA*	GK1747		TGTTGACAAACTTGTATATACAAGATAAAATGAAT
GK2482	*sigA*	GK1740		GTTTGCCACAAGACGAAGAATATATTAAGATTAGA
GK2482	*sigA*	GK1737		TATTGTTTTTTCGCAGTGATCATGGTTGAATAAAA
GK2482	*sigA*	GK2153		GCTGGTCAAATCGGCCGCAGTCTGCTACAATAAAG
GK2482	*sigA*	GK2136		GCTTTAATAAAGTATACCTACTTCGTCTCCTGTAC
GK2482	*sigA*	GK1694		TGAATAATATAAGTTTATTATTAACGCTATCTGGA
GK2482	*sigA*	GK1652		CGTTGTATGGCTCTTTCTTTTTTGTTATCATGTTC
GK2482	*sigA*	GK2000		ATAATAATACAATTTGACTGCTATAATTTATTTTG
GK2482	*sigA*	GK1598		GGTTGAAAGACCAAGGGAAACCGTTTACAAAAGAA
GK2482	*sigA*	GK1525		TTTGAACGAAAAAAGAAAAATCATTTATAATGAAT
GK2482	*sigA*	GK1505		TATTGAAAATATTTATGAAAGCGTGTACAATAAAA
GK2482	*sigA*	GK1491		CGTTGACTTTTACGTAAACGTCAATTAAAATAAAA
GK2482	*sigA*	GK1431		GATTTAATATATAACAAATATAATTAAAACTTTTG
GK2482	*sigA*	GK1430	*gltC*	GTTTTCAAAATTAATATAAACAATATATAATTTAG
GK2482	*sigA*	GK1426		ACATTGTTATGTTTTATAAATCTAGTACTTGTTTC
GK2482	*sigA*	GK1419		AAAGTAATATGTGAAACAATTTATTATCTTTTGTA
GK2482	*sigA*	GK1411		AATTCACAAAATATTTCAACTATATTATAGTAGTG
GK2482	*sigA*	GK1409		ATTTTACTAATGTTTTGTTTTGCGGTAAAGTGTAA
GK2482	*sigA*	GK1408		AATGTGAAATGGCGTTTTGTTTTGTAATCATTTTA
GK2482	*sigA*	GK0999		GGAGTAGTATTGGCTTTGGATCTTTTTCGCGGTTC
GK2482	*sigA*	GK0991		GATATTAGATCGTACCTTCAACCAGTTTCAGTACG
GK2482	*sigA*	GK0968		ACTTGAAAGTCAAAAAAGGTCAAAGTATAATAAAG
GK2482	*sigA*	GK0953	*mtnW*	GATTGTTCGAAAAAAATTTTTGAAAAATGATTGAC
GK2482	*sigA*	GK0952		GCTTTACATTTTGCTTTTTACGATATATGATGTTG
GK2482	*sigA*	GK0951		GTTGTAGTATATAGCATTTTTCGTTTTACATTTCG
GK2482	*sigA*	GK0950	*mtnK*	GATTGACAACATCGTGAAAATTGCAAAAACTGAAT
GK2482	*sigA*	GK0924		CTTATAATATTACTTATATTGGTACAAAAAAGATA
GK2482	*sigA*	GK0902		CTTTTCGATCTTGATAACGATACTGTAAAATAAAA
GK2482	*sigA*	GK0900		AAAATAAAATGTCATAGCAATAGTTCTAGCTTTTC
GK2482	*sigA*	GK1359		GGTTGACAGCGTTTTCAACAGATGATATGCTAAAA
GK2482	*sigA*	GK1347	*citB*	TTTCGACTTTTGTGCGAAAAAATAGTATAGTATTA
GK2482	*sigA*	GK1339		ATGATAAAATAGTCACTTGCGTTTTTACCTTTTAT
GK2482	*sigA*	GK0891		GATTGGCGTTAACCGAAACGCATGATAAACTAAGC
GK2482	*sigA*	GK0866		AATGTAAAAATAATTTAAATTTTTTTAACTTATGC
GK2482	*sigA*	GK0819		ATTTCCCTTCTTTCGCTTTTTATCATAAAATAAAA
GK2482	*sigA*	GK0817	*spxA*	TATATCTTTCTTGGCCGATGTATTGTATAATATAT
GK2482	*sigA*	GK0804		TATTGCGCATCCACGCTCTTTCGGCTAAAATTAGT
GK2482	*sigA*	GK1281	*spoIIIE*	TAATGAATTGTTCAGCCGTTCATGATATAATAGTA
GK2482	*sigA*	GK1260	*nusA*	TATATAGTTTTGCATACTCGGGTAACTTCCTCTTT
GK2482	*sigA*	GK1212	*xerC*	AATAGGCATCGGATACAATTCAGAATTTTATAAAA
GK2482	*sigA*	GK0799		ATTTGCCGTGCGCTTTGCGATCCGCTATAATGATT
GK2482	*sigA*	GK0790	*argC*	ATTTGATTTTATTTTTATACAGTATTATAATGAGA
GK2482	*sigA*	GK0788		AGAGTAATATTATGACATATTTTTATTTTAGTTTA
GK2482	*sigA*	GK0782		AAATAATTTGAAAAAGTGATTCTGAAATAATGTGC
GK2482	*sigA*	GK0768	*fnr*	AGTGTAACATTGTATCTTTTTCCTTTTTGGTTAAA
GK2482	*sigA*	GK0738		GCATTCCATTCTTCACAATGTATAATATAATCAAA
GK2482	*sigA*	GK0730		TTTTTGACACTGTATTAAATGGCGTTACGATAAAA
GK2482	*sigA*	GK1187		TTTGTGCTTGCATTTCGTTTTTTGATATTATATAT
GK2482	*sigA*	GK1147		GATTGACAAACGAACACGCCCCCTTTACAATGAAA
GK2482	*sigA*	GK1136	*ileS*	CTTGACGAACCGAAGACGTTTCGGTTATAATAATT
GK2482	*sigA*	GK1126	*spoIIGA*	TATATAACATAGAGAGAATTTCTTTCTGCTAGAAA
GK2482	*sigA*	GK1124	*ftsA*	TTTTCGCGTGTTGAATTATATAGGGTAAAACAAAA
GK2482	*sigA*	GK0693		CTTTTTCTTTATGCCATCTTTGTTATATAATACAA
GK2482	*sigA*	GK0681		TCTTCCCAAAAGCGAACATACGTGCTAAAATAAAA

GK2482	*sigA*	GK0668		CCTTGCCCTTTGCTGTTCTTTTTTTTATAATAAAT
GK2482	*sigA*	GK1080	*ctaA*	TTTTTCCTCATCATGTGAACTTTTTTTGAATAATT
GK2482	*sigA*	GK1058		TTTTTTATTTCTCTTTCATTTTGGGTACATTAATA
GK2482	*sigA*	GK1044		GTTCGAATATTCCACTATTTTATGATATGATAAAA
GK2482	*sigA*	GK1023	*sucA*	TGCTAAAATAAACAGTGAAAACGTTTATACTGTTA
GK2482	*sigA*	GK0478		CTTTGTAGACTAATTATAAATATTATAAAGTGATA
GK2482	*sigA*	GK0475	*ldh*	TCTGTATCGTTAGCATGTTGTGGCGTAAAATAAAA
GK2482	*sigA*	GK0465		CCCTCCCTCATCTATGGAATTCGGCTATATTATAT
GK2482	*sigA*	GK0419		TTCTGATGAAAAACGGATGATGTGGTAAAATGTAG
GK2482	*sigA*	GK0416		TGATTGACATTATCGTCAATTGTGCTAAAATTTTA
GK2482	*sigA*	GK0381		GCTTGACAAAGAAAACGTTTTAGAGTTAGATGATG
GK2482	*sigA*	GK0372		GGAATAATAAAATTTTTATATATTAAAAATTTTTT
GK2482	*sigA*	GK0332		AAAATAGTATGGTTTGTAGGCACTTTTTTGCCACT
GK2482	*sigA*	GK0284		CGTTGTGATCGAATGATAGATTTTTTATAAGAAAA
GK2482	*sigA*	GK0257		TTTTTAACAACGGTTACAAAACAGGTTAGCCTTTT
GK2482	*sigA*	GK0254	*guaA*	TCTTGACCGTATGCCGGCAAGTTGATAGAATGAAC
GK2482	*sigA*	GK0250	*fumC*	GAAGTAACAGCTTGGTAAGGGGAAAAAACTTATAT
GK2482	*sigA*	GK0199		GATTGACTATTCATAACATTTTGGTCATTTTTACA
GK2482	*sigA*	GK0196		AGAATAATATTGCTTTAAGAATTTTTTCTGTTACT
GK2482	*sigA*	GK0188	*rocD*	GTTTTAATACCTTATATAGCTCTAGCCTTTTGTCC
GK2482	*sigA*	GK0186		TATTGAAAATAGAAAAAAATAATTTCATAATAAAG
GK2482	*sigA*	GK0185		GAAATAATACTTTAATAAAAAAAGATAAAAGTTAT
GK2482	*sigA*	GK0098	*rpoB*	TGTTGACACCGTTTTTTTATTGTGGTAGCATTATA
GK2482	*sigA*	GK0089		GATTTACTATCTTAACGACTTTAAAAACTTTTTAC
GK2482	*sigA*	GK0083	*gltX*	TTTTGAACTTTGCCGCGATAAACGGTACAATAATA
GK2482	*sigA*	GK0065		ACTCGCAAAAAAAGAATGAAAACAGTAAAATGATT
GK2482	*sigA*	GK0057	*spoIIE*	TATTGACAATCGTCATCTCCTCCGTTATGATATAA
GK2482	*sigA*	GK0043	*glmU*	ATTTTTAAATATTTTTTAAAAAGCTTTTTTCTTCC
GK2482	*sigA*	GK0016		AGTTGCGTCTTTATTCTCCATATTCTATGATGAAA
GK2482	*sigA*	GK0001	*dnaA*	CGTTGCAACAAGCCGCTTTATTTGCTATTATTATT
GK2485		GK2850		CCCACACAATTAGTATGGACTATTAATAAAAATGTGTTATACTATTTTCAAGAGTGA
GK2485		GK2726		ATATGATATACTATTTATACAAATATAAGAAAATAGTATAACATATATGAGAGGAGA
GK2603		GK2796		AATAATGTAAATGACGGTGTGTTTTCGTTGTCGATTGGCGCCTGTTTTGGTTAAGATGGACGATGTACAGGGATATTTTGCCGACAATGAGCGGAGGATGGAAGCGAT
GK2603		GK2602		CACTCTCGGACGCCCCTTTCTTTTCACCTGTCTTGACATCTATATTTACACATATTTAAACTAATGGCAAGAGGTTTGGCGAAAAAAATGATGCAAAGGCGGGGGAAT
GK2603		GK2601		TAAGGGGGCGGAAACGTAGTAAAAAAAGCGGTTTGGAGAACGGTAATCAAATTTATACACATTTATATCTACAGTTCTGTCCACTTTTCTTTCCCCGCAGGCTCTCAC
GK2689		GK3398		ATAACTGACTTACGAGTAAGTAAACATGTTATCTGTATT
GK2689		GK2690		TTTTTTTTTGATGAAAAATTAGAAAAAGTAAAATAAAAT
GK2689		GK3008		ATTTTTTTCATCAAAAAATGAATGACCATTCATTCAATG
GK2689		GK2136		CGCTTTTTATGACTGGTAAGTCATAAGACAGCTTCCTAC
GK2689		GK1598		TTTACGGACGTATAGGTATTGTTACTTAAGCCTGTTTCC
GK2689		GK1505		AATTATTGAAAATATTTATGAAAGCGTGTACAATAAAAA
GK2689		GK1491		ATCTTTTCCCTACCGGTACGTGATAAAGTGGTAAAGGCT
GK2689		GK1315		TTTCTTTCTCCAGAAGGAAAGAAAAAACACGGCGTTTTT
GK2689		GK0693		GAAACAATATATTATGTTAACACTTAAAAGATTTTGTAT
GK2689		GK0668		ATTTTTTTACTTATAAAATGAATGGTCATTCATTCATTT
GK2689		GK1044		TTTATGATATGATAAAAATGAATGAGTAATCAGTCATAG
GK2732		GK2732		AAAACTGGAGTACAACTTTCGTC
GK2732		GK3182		AAACAAAAAAGATAGTTTTAAAA
GK2732		GK2463		AGAAACTTAACAACTGCTTTATA
GK2810		GK2785		TAAAAACTTTTTCT
GK2810		GK2661		TTTTTCGCAACTGC
GK2810		GK3444		GCTTTCGCAACTGT
GK2810		GK3415	*eutD*	TGTAAAGGCTTACC
GK2810		GK2889		AATTTCGCTAGTTT
GK2810		GK2872		TGACAACGATGCCA
GK2810		GK2807		AGAAAACTCTTTCT

GK2810		GK2806		ACATTAGTGAATGT
GK2810		GK3231		TGTAACCGGTTACA
GK2810		GK3212		GGATGGCGCTTCCA
GK2810		GK2772		TGACAATGATTTCG
GK2810		GK2759		CGTCAACGATTTAT
GK2810		GK2736		ACATTCGTGAAAGC
GK2810		GK3198		ACATTCGCTTATGT
GK2810		GK2534		TACTTCGCTAAAGT
GK2810		GK1953		TGAATCCGCTTTAA
GK2810		GK1951		TGAAACAGCTTTGC
GK2810		GK1919		ACCTTCCTAAAAGT
GK2810		GK2383		AGAAAACGCCTTAA
GK2810		GK1875		ACTTTCGCAATATT
GK2810		GK1747		TGAAAGCGATTTAA
GK2810		GK2136		TCTTTCGCAAAAGA
GK2810		GK2037		TGAAAGCGCTTCAT
GK2810		GK2000		ACTTTCACCTTTGC
GK2810		GK1598		GGAAACCGTTTACA
GK2810		GK1505		ACTTTCGCACATGT
GK2810		GK1491		ACCTTCGCGACGAA
GK2810		GK1426		ACCTTTGCAATTTC
GK2810		GK1411		TGTAGCCGCATTCA
GK2810		GK1359		ACTGTCGCAAAAGT
GK2810		GK0730		ACCTTCGCCATGGT
GK2810		GK0723		ACCTTCGACCTAGT
GK2810		GK0710		ACTTTCGCCATGGT
GK2810		GK0693		ACTTTTGCCTCTTT
GK2810		GK0668		AGGAAACGCTTTCC
GK2810		GK0199		ACATTCGGAAATGT
GK2810		GK0185		CGCAATCGCTTACA
GK3059		GK3059		GCACTGCGGGACATAATATGTCATACCGGGACATAAAATGTCCCTA
GK3061		GK0199		AAAGAAAGGGAAGGTTGCTAGAATGAGGGAAAATGA
GK3061		GK0188	*rocD*	AAGAGGTATACGTTCGCTTTTGGCATAGCCTTTACA
GK3061		GK2889		TTAGCATAAACTACAGTTCTTTACTTGCTGGAAACC
GK3061		GK2772		TACGTGGACGCATGAAAATGGGAAATTGAAAATTGT
GK3061		GK3198		AGAATTTTTATAATATTCTTTCCTTTGTAAGCGAAT
GK3061		GK1951		CGGCGGCTGGGATGGCGATTGCAGAATGCATGACAG
GK3061		GK2235		AAGGTATAGAAAGGCTCCTTGAAATGGACAAGGAGT
GK3061		GK2037		AAGAGAGCCTTTTCCGCTCAAAGCGAGGGCGCAAAT
GK3061		GK1411		CGTTGAATATATTAAGTGTTTTATAAAGTTGATATA
GK3061		GK0730		GATGAAATATTCTGCGTTACCGGTATTCTCATATAG
GK3061		GK0710		CAAGCGTTGGAACGATTTTTGCTTTATAGAAATGAT
GK3150		GK3444		CATTTCACCATCATTCAGAA
GK3150		GK1747		TAAACGGACTAGTAAAGTCG
GK3445		GK3444		ACAAAATCCAGAAAAAACAAAAAGTAAAGTGGTAGTAAG
GK3445		GK1747		CCTCGGTTGTGCGTGGGACGGCAATCGATCCGGCGAACT

Table 1: Binding sequence predictions for *Geobacillus thermodenitrificans* NG80-2. Columns show transcription factors (TFs), their regulated genes and the binding sequences for these genes.

TF Locus Tag	TF Name	Target Gene Locus Tag	Target Gene Name	Binding Sequence
GTNG_0001	*dnaA*	GTNG_0001	*dnaA*	ACATTTGTGGATAGGAT
GTNG_0040		GTNG_1397	*xpt*	CCCATTTTCGCTTATTAACAGTAAAAAATAGGTAGTTACAAGCAGAAAAAA
GTNG_0040		GTNG_0237	*purE*	GAAAAACACGAACATTCCTGTTTTTATAATATAGAATGTTCGATTTTACCT
GTNG_0040		GTNG_0234		CTATTTTTTGCTTACTTTCCACTTGTAAAGGAAACCTTACAAGCAAACAAA
GTNG_0040		GTNG_3416	*purA*	CCATTATTTGCTTATTTTATCAAATATTAACAAAAATTACAAGCACAAAGT
GTNG_0040		GTNG_3355		TTCTTTTTTGCTTATAATAATATAAACAAACATTATAACAAGCAAAAATCC
GTNG_0040		GTNG_3314	*glyA*	ACCTTTTAGGCTTACTAAGGACATGAAAGATCAATCAAATAAGTGAAAAAG
GTNG_0089		GTNG_3049		TAAGGGAAAATGTATCTCTATGACTATATTTTTCTAT
GTNG_0089		GTNG_2420	*dnaG*	AGAAGGATTTTTTAGAACGGTGTAGAATAATAGGGGG
GTNG_0089		GTNG_2317		GAAGCAGGAATCGATTTATATGGCAAACAAAGAAAAG
GTNG_0089		GTNG_0989	*ftsA*	AAAACAAAATATGGTGTTTTTGCGGAATTTATAAACT
GTNG_0089		GTNG_0226	*fumC*	TTTCCAAGAAAAGTGACAAAAACTGACTTAACAGACG
GTNG_0147		GTNG_0147		CAGAAGAAATCATTTCGTGCATTTTGTCGCCTTAT
GTNG_0147		GTNG_2799		GCTATTTCATCTTTTACGTCTTTTGCAAGGTTGAA
GTNG_0147		GTNG_2694		ATGCTTAAACCGATGTACAATGAATATTGTACCAC
GTNG_0217		GTNG_0651		TCACATCCGTTGTTTAAAATGTAAGAAAATTAA
GTNG_0217		GTNG_0487	*ldh*	TAAAATAATTTGTGAATGTATTCACAATAATAA
GTNG_0286	*hhoA*	GTNG_0286	*hhoA*	AGTAAAGAAAATAGTATTGTTTCTACGCACTTTTTTTCCACTTTTAGTCTTCCCTTCCCTCCTCAGTGAGGCTTCTCTGACCAAAGAAAGTGATAAAAAAGTGAAAAGAACTG
GTNG_0490		GTNG_2576	*hemA*	CACGTTACTACTTTTTTTTCT
GTNG_0490		GTNG_2506		AATTTTAGTAAAATTAAATTT
GTNG_0490		GTNG_0697	*spxA*	ATCATGTTAAAGTCATTTCAA
GTNG_0490		GTNG_0490		AATATTTATAATATTTTACTA
GTNG_0561		GTNG_1910		AAATAGTTTTATCATAATAA
GTNG_0668		GTNG_1712	*narG2*	AAATGTGACAAAATTAACATC
GTNG_0819		GTNG_0819		CTTGAATGACGGCCATATTTCCCTTACGATATACGAAATAGTGTATAAGGAAGC
GTNG_0910		GTNG_1206		AGTAACCTCTTAAAA
GTNG_0910		GTNG_2660		AGGAATTGCTTATGT
GTNG_0992		GTNG_3415	*yycF*	GGAAAAAGTAACAAAAGCGCGGCCTCTAGTGTAAAAGCTT
GTNG_0992		GTNG_3288	*spoIID*	CTGTCATATATACTTGTCCGAGCCCATAACTATAGTAACG
GTNG_0992		GTNG_2782		TAATAATTATTAATTTATATCTATTATAATATAAAAAGTG
GTNG_0992		GTNG_2662		TCTTTTTGCTTCTTCGTAAAGAAACATGTAATATATCAGT
GTNG_0992		GTNG_2656		GCACTCAATACCATACTATTTTTACTCTTTGTAACCTGCT
GTNG_0992		GTNG_2595	*gerM*	TATTCTATTTTGGCAACCAGCTCGTATACATAGTAGTACA
GTNG_0992		GTNG_2515		CTAGCATATATTGTATAAAAGATTAAGGCATTTGAGAAGG
GTNG_0992		GTNG_2352		TTGAAATAGAAATAAACATTTACGTATATATTCAAAAGAA
GTNG_0992		GTNG_2150		GAGTCATCATCCCTTCTCTAGGAAAATACATTCTAGAGAA
GTNG_0992		GTNG_1503		GTATTGTACTTTTCAGAACATTCGAATATAATGAAGGTAA
GTNG_0992		GTNG_1446		AAAAATATATCAATAAGAAGTTTACCCCCTCTCCCTCTTA
GTNG_0992		GTNG_0965		TGGTCTAAAATGTCCCCCTATGCTCGTATATTAGGTTATA
GTNG_0992		GTNG_0945		AAAAAAAGTTTAGTACACTTGAAAAAAACTTATTAAACAC
GTNG_0992		GTNG_0858		TTCGTCTTTCTTGTCCACGTTAGACATATATATAGCAATA
GTNG_0992		GTNG_0790		CCTTTAAATTTCAAATCAGCTTGTAATAAAATGAAAGTGT
GTNG_0992		GTNG_1169		AAAAAGTAAAGGCTACGCATTTGCTGGCGACTAGACTTAT
GTNG_0992		GTNG_1159		TTGTGCACATTAGCGGAGATCACGCATAGGATGAAATGAA
GTNG_0992		GTNG_0607		AATTCATCGAAATCAACGATAGTGCATAAAATGGAATCAT

GTNG_0992		GTNG_0552		GTGTTCTTTTTTTGTCCGTTTTGTCATAGTCATGAGAATG
GTNG_0992		GTNG_0517		AACGTATCAAAACGCCTTTTTCCTCATACATTGTGATAAG
GTNG_0992		GTNG_0498		TCTGTCAATTATTTCGAATTTTTGCATAGGATAAAATAAA
GTNG_0992		GTNG_0497		GGCGAATGATACGGGACATCCTCGCATAAATTGTAATAAT
GTNG_0992		GTNG_0363		CAACATACGTCCTCCGGTTTGGAGCATATAGTAAAGGATA
GTNG_0992		GTNG_0204		TCTGCATATTCTAGACAGTTGTCTCATATATTGTAGTGCA
GTNG_0992		GTNG_0144		TTTCATAACGATAATGGCTGTCTGCATAAGATGTACATAG
GTNG_0992		GTNG_0014		CGGTCATTTCTCTTCTCTTTCCATCATAAAATATGGAGAA
GTNG_0993		GTNG_0049		CGCGTATAGAAAGTGGATTGTGAAGGATACTAATGA
GTNG_0993		GTNG_2965		AGTTAAATTACAAGTTTTTTCCAACCATGTTCGAGC
GTNG_0993		GTNG_2939		GGAAAAAATAGTTAATATCGTAAAATAAATTTTAAA
GTNG_0993		GTNG_3374		GAATATATTAATGGAACATAAGCTCTATAAACTAAA
GTNG_0993		GTNG_3351		AAGTTTAAAATAAAAAAATTTTTTTCATACTAGTAA
GTNG_0993		GTNG_2710		CGATATGAAATATATATACTAAACGGTAATATCCCC
GTNG_0993		GTNG_2447		GAGGCTTCATTCATTTCATTAAGGAAGCAAAAAAAC
GTNG_0993		GTNG_2318		TAATATAAAAGGGGACAAGTCCGTTTTAATTTTGAC
GTNG_0993		GTNG_2242		GTTTTTTATTTTGGTTCATTTGTATAAAAATATCTC
GTNG_0993		GTNG_2238		CTGCATGATCCTTTTTTCATCCAAACAAACTAAAGA
GTNG_0993		GTNG_2167		AGTAATCCAAAACGCTGCAATCTCTTGTAAAATATG
GTNG_0993		GTNG_1471	*gerKA*	ACGAATCACAAGTGGCCGATCCCTCCTTACTACGTG
GTNG_0993		GTNG_0860		ATGCATAAAAATGTTCTCACATTGGAAGACTATGCC
GTNG_1004		GTNG_1004		ACAGGAACCTTTAAATTCAGTCCTGTGAGGCTGAGAAGGGGTCG GATGTAAAAA
GTNG_1040		GTNG_0714	*fabL*	TTAATTGTGGACCACGATTATAATTTATTATTAATCCTCCTACTTG CTT
GTNG_1040		GTNG_0684	*fabH*	TATTGCGCACATGCGCTCTTTCCGATAAAATTAGTACCAAGTAAT AATA
GTNG_1040		GTNG_1040		CATTCCGGTTTTTGATATTATATATTACTTTTAGTACCTAGTCATA AAA
GTNG_1068	*codY*	GTNG_3056		TTAAACATTTAAAAAAATAAACCGA
GTNG_1068	*codY*	GTNG_2590		AAAAGGTTTTTTATTATGGATGACA
GTNG_1068	*codY*	GTNG_0651		AGTGAAGGAATAAACGTGGCTAAAA
GTNG_1100		GTNG_0884		GATATAAGTTAAAGCTTATGTATCGGAAGTACAAGCTAAC
GTNG_1100		GTNG_1271		ATTTTTTGTAAAACCAGTATCATCATACTATCACTTTAGT
GTNG_1100		GTNG_0293		TCATAACAAGATATAAAACTGTTGCAGATAGGGAGTAGAT
GTNG_1100		GTNG_3117		TTCCTTTAATTGAAAACGTATACAACAATAAAATGAAAGG
GTNG_1100		GTNG_3056		CTATTAAACATTTAAAAAAATAAACCGATATAAAAGGTGA
GTNG_1100		GTNG_2466		TATACTATTTATACCAAAGAAGTATTTTGTTTTCTTTGTA
GTNG_1100		GTNG_1498		TTAGAGAAATACCTTGTTGCATTGCCTATTTGATTGACAT
GTNG_1183		GTNG_0188		AAAAGAAAGCAAATTTTCATGT
GTNG_1183		GTNG_2668		TTTTCCGGTGAAAAAGCAAGTA
GTNG_1183		GTNG_2604	*uvrC*	ATATGGGAAGATACGATTATAT
GTNG_1183		GTNG_3036	*uvrB*	TGGTACGAATATCAGTTCGCAT
GTNG_1183		GTNG_2522	*ruvA*	TGCCTTTGCCAACAAAGCAATT
GTNG_1183		GTNG_1647		TGTCTTTGTCTACAAGAACAAA
GTNG_1183		GTNG_2069		TAAATAGAACATACGTTCTTAT
GTNG_1183		GTNG_1183		TGAGCTTGAATACAAACAAAAA
GTNG_1183		GTNG_1149		AAAAACGAATAAATGTTCGACT
GTNG_1183		GTNG_0507		CAACTTTGAAAATAAGCGTCCT
GTNG_1268		GTNG_1269		AATCTTTATAATTATCAGAATTATTA
GTNG_1268		GTNG_1268		TGTAATCATTTTATTTATATTTAATA
GTNG_1291		GTNG_1292		ATCGAAAAATGAGA
GTNG_1291		GTNG_1291		AGAGTAAAAAGCTA
GTNG_1642		GTNG_1644		CAACTGTTTGAACATATATGTTCTATCTTACT
GTNG_1798	*araR*	GTNG_1798	*araR*	TTATCTATGCTTGTTTAC
GTNG_1829		GTNG_1828		AGTGAAAAACCCTTTTTTC
GTNG_1829		GTNG_2171		AGCGATGACATATTTTACG
GTNG_1829		GTNG_1271		ATTTAAAAAACATTTTGGT
GTNG_1829		GTNG_0293		TCGTTAACGATAAACCTAA
GTNG_1829		GTNG_2312		AACTTTTGCTCGAACGTGA

GTNG_1829		GTNG_1852		CTTTTTTCCTAAAACCAGA
GTNG_1829		GTNG_1829		CTTTTTTCCCAAAAAGTGA
GTNG_1829		GTNG_0177		AGTGCAAAAGGCTTTTTGG
GTNG_1829		GTNG_0167		CCGGCAAAATTTTTTTTCG
GTNG_1829		GTNG_2708		AATGGTATAATATTGTCCA
GTNG_2090		GTNG_2089	*glpD*	TCAGGAGACATGGAGAGACC
GTNG_2090		GTNG_1215		GGCGGAGAAACGGAGACCCA
GTNG_2207		GTNG_1634		GGACAATTTTGCTTAAA
GTNG_2207		GTNG_0668		ACAAGTTGTTAAAAAGT
GTNG_2207		GTNG_0657		AAATCTTTTTAAACACT
GTNG_2239		GTNG_1471	*gerKA*	CTCGTATATACTTTTGCCCGATTGCCTATAATGGAG
GTNG_2239		GTNG_0860		AATGCATAAAAATGTTCTCACATTGGAAGACTATGC
GTNG_2239		GTNG_3433		AAAAATAAAATTGTTCTAGCTAGCAACTAAACATCA
GTNG_2239		GTNG_2965		ACTTGTAAAAATGTCTTTGCGTTTTAACGGTCGCAC
GTNG_2239		GTNG_2901		AATGAATATTTCGATTTTGGATGGAGATAAATAGAG
GTNG_2239		GTNG_3397		ACATATAAATAAGTAAGGGTTAGAAGGAAATTGTAA
GTNG_2239		GTNG_3351		CAAGTTTAAAATAAAAAAATTTTTTTCATACTAGTA
GTNG_2239		GTNG_3287	*spoIIQ*	CGTGTATATCTTTTCCTCTTTCTGTTCAAAATGGTT
GTNG_2239		GTNG_2447		GCGCTAAAAACGGCCGATCGCCGTAAAAATACCTAT
GTNG_2239		GTNG_2344		GAGGTTTCCGAACGTTTCCTGTAGAATAAAATAATA
GTNG_2239		GTNG_2318		AATATAAAAGGGGACAAGTCCGTTTTAATTTTGACA
GTNG_2239		GTNG_2242		ATTTGTATAAAAATATCTCCATTGGAAAAAATGGCA
GTNG_2248		GTNG_1477		AAATAAGTTCTCAAGGCGTCATCATTGTCAAAGAAAT
GTNG_2248		GTNG_1318		TATTTTCGCTATTAGTAACAGTTATATTTTAATTTTT
GTNG_2248		GTNG_1272		GAAACCTATTGACAATGAGAATTTTTATCAATTAAAA
GTNG_2248		GTNG_0175	*fhuC*	TATTATAACTATAACTAAGAGTAAAAGGATCAGGTTT
GTNG_2248		GTNG_2905		GAGTATTACTATCAGTACGAACAAGTGATTGCATTAT
GTNG_2317		GTNG_0991		CTTTGACAAAACAT
GTNG_2317		GTNG_0057		TTTTGACAAATTTC
GTNG_2320		GTNG_1271		CTGAATTTAAAAAACA
GTNG_2320		GTNG_0293		TCCCTCACAATAAGTA
GTNG_2320		GTNG_3117		TTCCTTAAAAAATTTA
GTNG_2320		GTNG_1828		CTGAAGAAAAAGTGAT
GTNG_2320		GTNG_0670	*argC*	AAAATAAAAATACGTC
GTNG_2419	*rpoD*	GTNG_3442		TTCATAGCCTAGTTTTTTTTGCTCCTTAAAGTTCT
GTNG_2419	*rpoD*	GTNG_3416	*purA*	CATTGACTTTATGCCTGAAAACTGTTACACTTACA
GTNG_2419	*rpoD*	GTNG_3415	*yycF*	GGATGATTTCATTTTCATAACATGATAAATTAGTA
GTNG_2419	*rpoD*	GTNG_3410		TTAAATAAATCGGGTAAATTCGGTTCTATTGTTAT
GTNG_2419	*rpoD*	GTNG_2960		TATTGCGAAAATTAATAACATACTATATATTTAAG
GTNG_2419	*rpoD*	GTNG_3398	*manA*	CGTTTGCCGACGGTGTATTTCATGGTAAGATGAAG
GTNG_2419	*rpoD*	GTNG_3393		AGGGTAACATGTTTTATATTCCGTTCCTCCGTTAA
GTNG_2419	*rpoD*	GTNG_3384		ATAGTATGATAGTTATAAAAACACTAATACGTTAT
GTNG_2419	*rpoD*	GTNG_3361	*eutD*	ATAATGATATGGTATCAAACGGGTTTTTTCGCCTG
GTNG_2419	*rpoD*	GTNG_3351		AAGTTTAAAATAAAAAAATTTTTTTCATACTAGTA
GTNG_2419	*rpoD*	GTNG_3350		ATGATCATACTTTTTTTAAAAAAATAAAATTTGAA
GTNG_2419	*rpoD*	GTNG_3343		TGTTGACTGAATGCTCATTCATTTGTAAAATAAAC
GTNG_2419	*rpoD*	GTNG_3334	*pyrG*	GCTTGACTTTACCGAGGGAACTCGGTACAATGTGT
GTNG_2419	*rpoD*	GTNG_3314	*glyA*	TTTGGATTTGTCCGATGAAAGCTGATAAAATAAAA
GTNG_2419	*rpoD*	GTNG_3266		TTATTAGTATTGTATCATCTAAACAGTATAGTTAG
GTNG_2419	*rpoD*	GTNG_3264	*gtaB*	TTAGTAAAATAAGGAAACAACCAGACAAAAGGTAA
GTNG_2419	*rpoD*	GTNG_2774		GATAGATAAATGATTTCCTATTATATATAATGAAA
GTNG_2419	*rpoD*	GTNG_2751		GTATGGACTATTAATAAAAATGTGTTATACTATTT
GTNG_2419	*rpoD*	GTNG_2712	*acuA*	TGAATAATATCATCTATTATTAACTAAAGAGTTAA
GTNG_2419	*rpoD*	GTNG_2711		AATTGAGAAATCAATTATTATCTACTATAATAAGT
GTNG_2419	*rpoD*	GTNG_2710		GCTATACTTTATATATATGATTTGCCATTATAGGG
GTNG_2419	*rpoD*	GTNG_2709		GGGATATTACCGTTTAGTATATATATTTCATATCG
GTNG_2419	*rpoD*	GTNG_2708		TTAATATTATTACTAAAATAAATGGTATAATATTG
GTNG_2419	*rpoD*	GTNG_2704	*rpsD*	AAAATAATATAGTATATTCGTCGTTTCTTCCTTTA

GTNG_2419	*rpoD*	GTNG_2700		CTAAGAACATTTTTTTTATTTTTAAAACCAGTTTT
GTNG_2419	*rpoD*	GTNG_3186		TATTTGAAAATTTCGTTTAATACGTTATAATTAAG
GTNG_2419	*rpoD*	GTNG_3175		GAAATAATATTGGTATGCACGCGTTTTAGTGTTAT
GTNG_2419	*rpoD*	GTNG_3139		GATATATGATAGTATTTTGAACATCCAACTTTTAA
GTNG_2419	*rpoD*	GTNG_3135		CCTTTACTTTTTTCGTCAAACGCGGTTTATTGGTG
GTNG_2419	*rpoD*	GTNG_3117		CTTTAATTGAAAACGTATACAACAATAAAATGAAA
GTNG_2419	*rpoD*	GTNG_2696		TATTCAAAATATAATAATTATTTTGTATAATAAAA
GTNG_2419	*rpoD*	GTNG_2686		TATTGAATTTTATATATTTTTCTATTATTATTAAT
GTNG_2419	*rpoD*	GTNG_2660		CGTTCACAAAAATCTGATAATTGTTTATAATAATA
GTNG_2419	*rpoD*	GTNG_2656		TGCCGAGTAAAAGCGTGAGTTATGGTATGATAAAA
GTNG_2419	*rpoD*	GTNG_2651		TTTATACAGATTGCGATAAATAGTATAACATATAT
GTNG_2419	*rpoD*	GTNG_2649		TGTTGCAAAAAACCCCTAAACAAAGTATACTATGT
GTNG_2419	*rpoD*	GTNG_2643	*thrS*	TCTTGATTTTTCCTGTTGTTTTCGCTATAATGGCG
GTNG_2419	*rpoD*	GTNG_2633	*pheS*	CCTTGCCTTTGTCTCTTGGTTTGCGTATAATGAAT
GTNG_2419	*rpoD*	GTNG_2617		TTTTTTGATGAAAAATTAAAAATAATAAAATAAAA
GTNG_2419	*rpoD*	GTNG_2601		TCTTGACGCTTGTACGATGTGGGAGTACAATGAAT
GTNG_2419	*rpoD*	GTNG_3081		CATTTCGTATAGATTATAAATTTGGTATAATAGAT
GTNG_2419	*rpoD*	GTNG_3076		ATGATAATATTTAGCTGTTGAAAAAAGGAAGTTTA
GTNG_2419	*rpoD*	GTNG_3072	*comFA*	GGTGTTCTTATTGGAAAATGTTTTTTATGCTGCAT
GTNG_2419	*rpoD*	GTNG_3048	*secA*	TATGATTAATAAGTAAGTCCATTGAAAAAATAAGG
GTNG_2419	*rpoD*	GTNG_3011	*clpP*	AAGATAATATCGTCCTAGTTTTTTTTTACGTTCTC
GTNG_2419	*rpoD*	GTNG_2581	*clpX*	CGTTAACAAGGGGTGAATAACATGTTTAAATTTAA
GTNG_2419	*rpoD*	GTNG_2576	*hemA*	GTTTCCCATTTAGACATGTTTATGATAGAATGTTC
GTNG_2419	*rpoD*	GTNG_2567	*valS*	ACTTGGCAAAGATGATGGACAATGGTAGTATAATA
GTNG_2419	*rpoD*	GTNG_2532		TAGATATAAATGTGTATAAATTTGATTACTGTTCT
GTNG_2419	*rpoD*	GTNG_2531		TCTTGTCATTAGTTTAAATATGTGTAAATATAGAT
GTNG_2419	*rpoD*	GTNG_2506		TTTATAGCATAATTTTAGTAAAATTAAATTTCTGA
GTNG_2419	*rpoD*	GTNG_2476		CATTGAACAAAAGAAAAACATATGCTACATTGGGA
GTNG_2419	*rpoD*	GTNG_2469		CATTGACAAGTTTTGTTTTTAAAAGTAAAATAATT
GTNG_2419	*rpoD*	GTNG_2444	*lepA*	CATTGAATCTTCTTCGCACTGTTGCTATAATCAAA
GTNG_2419	*rpoD*	GTNG_2435		GCTAGTTTTTGTTGCAAGGACATGGTATGATGGTA
GTNG_2419	*rpoD*	GTNG_2374		CACATAAAATCGGTACATAAGCATTTATACTTTGG
GTNG_2419	*rpoD*	GTNG_2351		CCTTGTTAAGGCACGAAATGTAAGATACAATAACA
GTNG_2419	*rpoD*	GTNG_2317		AATTTTTATTTTTTGCTAATTGATGAAAAATAAAA
GTNG_2419	*rpoD*	GTNG_2312		GTTTGTCAAGGAAAGAAAAAACAAGTATGATAAAA
GTNG_2419	*rpoD*	GTNG_1852		TTCTAAATAAAGAGTTATATAAGTTATGCGTTTAT
GTNG_2419	*rpoD*	GTNG_1829		TTTTGTCTTTTTTATATTTTCAGTCAATATTCTAT
GTNG_2419	*rpoD*	GTNG_1828		TATCTTATAACTGACTTTTATATTTTTTCTGTTTT
GTNG_2419	*rpoD*	GTNG_2289	*gntZ*	AGTTTAATACTATACAACAAACCACTTACTTGTTT
GTNG_2419	*rpoD*	GTNG_2266		CCTTGCTGCGTCATCAACTATGGGATATATTTTTG
GTNG_2419	*rpoD*	GTNG_1798	*araR*	TACGAACAAATGAGAATTGCTATAATAAAATTAAT
GTNG_2419	*rpoD*	GTNG_1712	*narG2*	GGGGTAATATGGTATTGTTTCTTATAAACAAGCGA
GTNG_2419	*rpoD*	GTNG_2171		CCTTTGCATCCGCCATCAGAAGGTGTATACTATGA
GTNG_2419	*rpoD*	GTNG_2138		GTTGACAAAAAAGACAGAAATGTAATATAGTTAAA
GTNG_2419	*rpoD*	GTNG_1674		GGTATCATATATGGTTCTTTTTGTTTTTCCTTTTC
GTNG_2419	*rpoD*	GTNG_1644		CGTTGACAAACTTGTATATACAAGATAGAATGAGT
GTNG_2419	*rpoD*	GTNG_1637		GTTTGCCACATTTCGCATAATATATTAAGATTAGA
GTNG_2419	*rpoD*	GTNG_1634		TATTGCTTTTCATTATCGCTCATGGTTGAATAAAA
GTNG_2419	*rpoD*	GTNG_2089	*glpD*	GCTAGCCAAACGGGCGGCAATCTGCTATAATAAAA
GTNG_2419	*rpoD*	GTNG_2024		AGGATAATATATTCTTTTTATCCCTCTTTAGTTAC
GTNG_2419	*rpoD*	GTNG_1539		CATTGACAGAAAAACATAACTATTATATTCTAAAA
GTNG_2419	*rpoD*	GTNG_1498		TTTGGAAAACTCTAAACATTAAGGTTTATCTATTA
GTNG_2419	*rpoD*	GTNG_0991		GGAGTGAAACTGTTTTGTAGGAAAAAAAGTGGTAG
GTNG_2419	*rpoD*	GTNG_0989	*ftsA*	TTTTGGCATGTTGAATTATATGGAGTAAAACAAAA
GTNG_2419	*rpoD*	GTNG_0945		ATTGTAAGATAGGTTATGCTTTTTTCTTGTTTTTC
GTNG_2419	*rpoD*	GTNG_0922		GGTGGTCATATGATTTATATTTTTTCAAAATATAA
GTNG_2419	*rpoD*	GTNG_0910		TTGATAAAATAAAAACAATATTGCTATTCTTTTAC
GTNG_2419	*rpoD*	GTNG_0904		GAAATAATATTGCTTTTGTTTTTCCCTTCTTGTTA

GTNG_2419	*rpoD*	GTNG_1397	*xpt*	GTTTGAACAAATGGTGACGATTGTGAAAAATGTAT
GTNG_2419	*rpoD*	GTNG_1373		TTTGAACAAAAAAAGAAAAATCATGTATAATGAAG
GTNG_2419	*rpoD*	GTNG_1356		TATTGAAATTATCTCTATGACTGATTACAATAAAA
GTNG_2419	*rpoD*	GTNG_0885	*sucA*	CGTCGATTTTAGTGAATAATAGTGCTAAAATAAAC
GTNG_2419	*rpoD*	GTNG_0878		GATATTAGATCGTACCTTCAATCAATTTCAGTACG
GTNG_2419	*rpoD*	GTNG_0856		ACTTGAAAGTCAAAAAAGGTCAAAGTATAATAAAA
GTNG_2419	*rpoD*	GTNG_0841	*mtnW*	GTTTTAACAAACTGTTCAAGTATTTATGCGTTACT
GTNG_2419	*rpoD*	GTNG_0840		ACTTTACATTTTGCTTTTTACGATATATGATTTTG
GTNG_2419	*rpoD*	GTNG_0839		GTTTTAGTATATAGCATTTTTCGTTTTACATTTCA
GTNG_2419	*rpoD*	GTNG_0838	*mtnK*	AATTGCCAACATCACAAAAATTCCAAAAACTAAAT
GTNG_2419	*rpoD*	GTNG_1292		GATTTAATATATAACAAATATATTTAAAACTTTTG
GTNG_2419	*rpoD*	GTNG_1291		GTTTTCAAAATTTATATAAACAATATATAATTTAG
GTNG_2419	*rpoD*	GTNG_1284		ATTATTCTATTGCAGATAACTGTTTAACAGGTTAT
GTNG_2419	*rpoD*	GTNG_1271		AATTTAAAAAACATTTTGGTCATAGTAGTATGATA
GTNG_2419	*rpoD*	GTNG_1269		ATTTTACTAATGTTTTGTTTTGCGGTAAAGTGTAA
GTNG_2419	*rpoD*	GTNG_1268		AATGTGAAATGGCGTTTTGTTTTGTAATCATTTTA
GTNG_2419	*rpoD*	GTNG_1215		GTTGACAACGCTTACAAACTTGCGCTATCCTAAAG
GTNG_2419	*rpoD*	GTNG_1206		TCATTATTTACAACTTGTCCTGTTATATAATAAGA
GTNG_2419	*rpoD*	GTNG_0745		AATGTAAAAAAATATTTAATTGTTTTAACTTATAT
GTNG_2419	*rpoD*	GTNG_1194	*ccdA*	ATCATAAAATATGATAAAATGGTTACTTACGTTTT
GTNG_2419	*rpoD*	GTNG_1149		GCTTGGCAATGGTCGCAAAACGCGGTATAGTATAA
GTNG_2419	*rpoD*	GTNG_1135		TAATGAATTGTTCAGCGGTTCATGATATAATAATA
GTNG_2419	*rpoD*	GTNG_1129		TTCTCTCGTCCGCAGTGGAATGTGATAAAATAAAA
GTNG_2419	*rpoD*	GTNG_0699	*mecA*	ATTTCCCTTCTTTCGCTTTTTATCATAAAATATAA
GTNG_2419	*rpoD*	GTNG_0697	*spxA*	TATATCTTTTTCCGTGGATTTATTGTATAATATAT
GTNG_2419	*rpoD*	GTNG_0684	*fabH*	CTTTCCGATAAAATTAGTACCAAGTAATAATAATT
GTNG_2419	*rpoD*	GTNG_0678	*clpB*	ATTTGCCGTGCGCCGTGCAATCCGCTATAATGATT
GTNG_2419	*rpoD*	GTNG_0670	*argC*	ATTTGATTTTATTTTTATGCAGTATTATAATGAGA
GTNG_2419	*rpoD*	GTNG_0668		AGAGTAATATTATGACGTATTTTTATTTTAGTTTA
GTNG_2419	*rpoD*	GTNG_0657		ACTCTGTCTCTGCTGTGATTTTTTTTACAATAGAA
GTNG_2419	*rpoD*	GTNG_0653	*narG*	TTTTGCATAAGTGATTAGGCGTTTGGAAAATATAG
GTNG_2419	*rpoD*	GTNG_0651		AATTTTACATTCTTTTAATTTTTTCCTTCACCTTT
GTNG_2419	*rpoD*	GTNG_1040		ACTATAATATATAATGAAAATCATGGATCAGTATT
GTNG_2419	*rpoD*	GTNG_1004		AATTGACAAACGGAGGCGGCCGCTTTACAATGAAA
GTNG_2419	*rpoD*	GTNG_1001	*ileS*	ATTTGTCTCATTTATAATAAATGGCAAAAATCGTT
GTNG_2419	*rpoD*	GTNG_0588		GCTCGTCAAATACGAACATGTGTGCTAAAATGAAG
GTNG_2419	*rpoD*	GTNG_0575		TCTTGCTCTTTGCTGTTCTTTTTTTTATAATTGGT
GTNG_2419	*rpoD*	GTNG_0553		CCTTTTTTCTTGTCTATGCGCATGAAACAATATAT
GTNG_2419	*rpoD*	GTNG_0504		TATTTTTATTTGGCTGACTTTTCTTTATGATGTTG
GTNG_2419	*rpoD*	GTNG_0490		CCTTGTAGACTAATTATAAATATTATAAAATGATA
GTNG_2419	*rpoD*	GTNG_0487	*ldh*	GCGTAAAATAATTTGTGAATGTATTCACAATAATA
GTNG_2419	*rpoD*	GTNG_0470	*fabI*	GATATAATATGCATTTCCTCACCTTTTTGTACTCA
GTNG_2419	*rpoD*	GTNG_0391		ACCTGCTGAAAAACGGACAATGTAGTAAAATATAG
GTNG_2419	*rpoD*	GTNG_0389		AGATTGACACCAGCGAGAATTGTGCTAAAATTTTA
GTNG_2419	*rpoD*	GTNG_0347		TATTCTAATTATTTGAAATGAATAATAAAATGATT
GTNG_2419	*rpoD*	GTNG_0299		ACTTGACATATCGGTTTTTAATCATTATAGTAATT
GTNG_2419	*rpoD*	GTNG_0293		TGAGTAATATATGTCCGCGTAACTTTTTAGTTGTA
GTNG_2419	*rpoD*	GTNG_0286	*hhoA*	TAAAGAAAATAGTATTGTTTCTACGCACTTTTTTT
GTNG_2419	*rpoD*	GTNG_0264		AAAATAAAATAAAAAATCTGAAAATTTTTTTTATGT
GTNG_2419	*rpoD*	GTNG_0237	*purE*	CTTGACATTTTGTCAAAAATCCTGCTAATATAAAA
GTNG_2419	*rpoD*	GTNG_0231	*guaA*	ACTTGACCGACTGCTGGCAAGTTGATAGAATGGAT
GTNG_2419	*rpoD*	GTNG_0226	*fumC*	TAGATAAAATCGATATCAATAATATTGTCATTTTT
GTNG_2419	*rpoD*	GTNG_0188		ATACTCATATCGTTAGTTGCTTCCTTCCCCCTTCC
GTNG_2419	*rpoD*	GTNG_0177		GAAGTAATATTGCCTAAGGAATTTTTTCTGTTATC
GTNG_2419	*rpoD*	GTNG_0168		TTTTGCCAAATATATTGAAAATAGAAAAAATAAGC
GTNG_2419	*rpoD*	GTNG_0167		CGAATAAAAAAGATAAAAGTTATATAAACCGTTTT
GTNG_2419	*rpoD*	GTNG_0098	*rpoB*	TGTTGACACCGTTTTTTTATTGTGGTAGCATTATA
GTNG_2419	*rpoD*	GTNG_0083	*gltX*	TTTTGAACTTTGCCACGATAAACGGTACAATAATA

GTNG_2419	*rpoD*	GTNG_0057		TGTTGACAATCTCCAATTCCTCTATTATGATATTA
GTNG_2419	*rpoD*	GTNG_0015		AGTTGCGTCTTTATTCTCCATATTTTATGATGGAA
GTNG_2419	*rpoD*	GTNG_0001	*dnaA*	TATATAAAAAGGTGTTTAAATAAAAAAATTTGCTT
GTNG_2422		GTNG_2751		CCTACACAATTAGTATGGACTATTAATAAAAATGTGTTATACTATTTTCAAGAGGAA
GTNG_2422		GTNG_2651		AAAAAACAGTAAAAAGAGAGGTGTAAGAGCTTTTTATACTATATGATAAATATGTCT
GTNG_2533		GTNG_2532		CCCACAACCGCTCCCCTCTCTTTTCATCTGTCTTGACATCTATATTTACACATATTTAAACTAATGACAAGAGGTTGATTGAAAAAAGTGAAAGAAAAGGCGGGAGAA
GTNG_2533		GTNG_2531		AAGAGGGCGGAAAAGAAAGTGAAAAAAGTTAGTTGGAGAACAGTAATCAAATTTATACACATTTATATCTACAGTTCTGTCTACTTTTCTCTCCCCTCGCCAACACCC
GTNG_2616		GTNG_0904		ACCAAGCATAAGAAAAAATGAATGAACCATCAGTCATAG
GTNG_2616		GTNG_1356		GTACTTTTTATTCGGATTTGTCTGAATGTTATTGAAATT
GTNG_2616		GTNG_1170		AGGCGCTAGCTCTAAATCAGTAAATGAGAATTTTTCTTT
GTNG_2616		GTNG_0575		ATTTTTTTACTTATAAAATGAATGGGTATTCATTCATTT
GTNG_2616		GTNG_2960		ATTTTTTTCACCGAAAAATGAATGACGGCTCATTCAATG
GTNG_2616		GTNG_3343		ACAACTGACTTACGAGTAAGTAAACATTTTATTTGTATT
GTNG_2616		GTNG_2617		CTTTTTTTTGATGAAAAATTAAAAATAATAAAATAAAAT
GTNG_2656		GTNG_3415	*yycF*	TTCATTTTCATAACATGATAAAT
GTNG_2656		GTNG_2656		ATACTATTTTTACTCTTTGTAAC
GTNG_2708		GTNG_0617		CTGGAAATATCCGTGAATTACAAAATATAATTGAGCGAGTGCTGAATCACAGTACAAAGCCGAT
GTNG_2715		GTNG_0651		ATACAACGTTTCCA
GTNG_2715		GTNG_2590		TCCTTTGGAAAAGC
GTNG_2715		GTNG_1356		AGTAACAGTTTACA
GTNG_2715		GTNG_1271		TCTTTTGCACAAAA
GTNG_2715		GTNG_1215		TGACAACGCTTACA
GTNG_2715		GTNG_0623		TCGCAAAGCTTTCA
GTNG_2715		GTNG_0617		ACTTTCGCCATAGT
GTNG_2715		GTNG_0575		AGGAAACGCTTTCT
GTNG_2715		GTNG_0504		ACTTTCGCAACAGT
GTNG_2715		GTNG_0293		ACCTGCGCCATATA
GTNG_2715		GTNG_0177		AGAAATTGCTTAAA
GTNG_2715		GTNG_0167		CGCAACCGCTTACA
GTNG_2715		GTNG_3361	*eutD*	ACCTTCCCGAATGA
GTNG_2715		GTNG_2712	*acuA*	ACATTCACCAATGT
GTNG_2715		GTNG_2711		TGTAACCACTTACA
GTNG_2715		GTNG_2708		TGTAAATGGTTCCA
GTNG_2715		GTNG_3175		TGTAACCGGTTACA
GTNG_2715		GTNG_3133		ACATTCGCTGAAGT
GTNG_2715		GTNG_3117		ACTTTTGCATATGT
GTNG_2715		GTNG_2696		ACTTTTTCGCTAGT
GTNG_2715		GTNG_2686		TGACTTCCCTTCCA
GTNG_2715		GTNG_2660		ACATTCGTGAAAGA
GTNG_2715		GTNG_2617		ACTTTCCCGACAAT
GTNG_2715		GTNG_2469		CGTTTCGCCAAAGT
GTNG_2715		GTNG_2435		AGAATACGCCTCCC
GTNG_2715		GTNG_2351		TTTATCCGCTCAAA
GTNG_2715		GTNG_2312		GAAAAACGCCTTAA
GTNG_2715		GTNG_1644		TGAAAGCGATTTAA
GTNG_3010		GTNG_3117		CTAGGCTTGCTTGACGAGCTTCACAAGGTAGTGGAA
GTNG_3010		GTNG_1828		AAAAAGATGGCACGAAACTTGCGAATCTGGTCTATG
GTNG_3010		GTNG_2171		TGTTACAAACCTTACAAACCAGGCAAAGTCGGAAAA
GTNG_3010		GTNG_1271		AATGCCATTGAATAAATCTGAATTTAAAAAACATTT
GTNG_3010		GTNG_0617		CAAGCATTGGAACGATTTTTGCATTATGGGAAATTG
GTNG_3075		GTNG_1644		TGTAAATTCTTATCGAAAAA