School of Computing Science

Enhancing Systems Biology Models Through Semantic Data Integration

Allyson L. Lister

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

December 2011

Abstract

Studying and modelling biology at a systems level requires a large amount of data of different experimental types. Historically, each of these types is stored in its own distinct format, with its own internal structure for holding the data produced by those experiments. While the use of community data standards can reduce the need for specialised, independent formats by providing a common syntax, standards uptake is not universal and a single standard cannot yet describe all biological data. In the work described in this thesis, a variety of integrative methods have been developed to reuse and restructure already extant systems biology data.

SyMBA is a simple Web interface which stores experimental metadata in a published, common format. The creation of accurate quantitative SBML models is a time-intensive manual process. Modellers need to understand both the systems they are modelling and the intricacies of the SBML format. However, the amount of relevant data for even a relatively small and well-scoped model can be overwhelming. Saint is a Web application which accesses a number of external Web services and which provides suggested annotation for SBML and CellML models. MFO was developed to formalise all of the knowledge within the multiple SBML specification documents in a manner which is both human and computationally accessible. Rule-based mediation, a form of semantic data integration, is a useful way of reusing and re-purposing heterogeneous datasets which cannot, or are not, structured according to a common standard. This method of ontology-based integration is generic and can be used in any context, but has been implemented specifically to integrate systems biology data and to enrich systems biology models through the creation of new biological annotations.

The work described in this thesis is one step towards the formalisation of biological knowledge useful to systems biology. Experimental metadata has been transformed into common structures, a Web application has been created for the retrieval of data appropriate to the annotation of systems biology models and multiple data models have been formalised and made accessible to semantic integration techniques.

This thesis is dedicated to my parents, who encouraged me in all things, and is particularly dedicated to my husband and my son, without whose patience and support I could not have finished this research.

"Among those who have endeavoured to promote learning and rectify judgement, it has long been customary to complain of the abuse of words, which are often admitted to signify things so different, that, instead of assisting the understanding as vehicles of knowledge, they produce error, dissension, and perplexity...." Dr. Samuel Johnson, 1752, via Nature Structural & Molecular Biology 14, 681 (2007).

"Metadata is a love note to the future." (NYPL Labs, `http://twitpic.com/6ry6ar`, via @CameronNeylon @kissane)

Acknowledgements

Many thanks go to my supervisors Dr. Anil Wipat, Dr. Phillip Lord and Dr. Matthew Pocock. A special thanks goes to those people who found extra commas and other errors: Phoebe Boatright, Dr. Dagmar Waltemath, Lucy Slattery, Mélanie Courtot and Dr. Paul Williams, Jr. Past and present colleagues at Newcastle University have provided much support and inspiration, including Dr. Frank Gibson and Dr. Katherine James.

I gratefully acknowledge the support of the BBSRC and the EPSRC for funding CISBAN at Newcastle University. I also acknowledge the support of the Newcastle University Systems Biology Resource Centre, the Newcastle University Bioinformatics Support Unit and the North East regional e-Science centre.

Declaration

I declare that the following work embodies the result of my own work, that it has been composed by myself and does not include work forming part of a thesis presented successfully for a degree in this or another University.

Allyson Lister

During the course of this work, I have been involved both in the development of a number of standards efforts and in the publishing of a number of papers.

The list below describes the standards efforts to which I have contributed and the roles I have had within those efforts:

- a developer of UniProt/TrEMBL [1];
- a program developer and contributor to the Functional Genomics Experiment (FuGE) [2, 3, 4], a standard XML syntax for describing experiments;
- a core developer and coordinator of the Ontology of Biomedical Investigations (OBI) [5, 6, 7], a standard semantics for describing experiments;
- an early developer of the ISA-TAB [8] tab-delimited syntax for describing experiments;
- a developer of the minimal information checklist MIGS/MIMS for genomics and metagenomics information [9];
- an advisor for the Systems Biology Ontology (SBO), the Kinetic Simulation Algorithm Ontology (KiSAO) and the TErminology for the Description of DYnamics (TEDDY);
- a co-author of the Systems Biology Markup Language (SBML) Level 3 Annotation package [10];
- an advisor for the Cell Behavior Ontology[1]; and
- an advisor in the nascent synthetic biology standards[2].

I have co-authored 20 papers, specification documents, articles and technical reports during the course of this work:

1. Mélanie Courtot, Nick Juty, Christian Knupfer, Dagmar Waltemath, Anna Zhukova, Andreas Drager, Michel Dumontier, Andrew Finney, Martin Golebiewski, Janna Hastings, Stefan Hoops, Sarah Keating, Douglas B. Kell, Samuel Kerrien, James Lawson, Allyson Lister, James Lu, Rainer Machne, Pedro Mendes, Matthew Pocock, Nicolas Rodriguez, Alice Villeger, Darren J. Wilkinson, Sarala Wimalaratne, Camille Laibe, Michael Hucka, and Nicolas Le Novère. Controlled vocabularies and semantics in systems biology. *Molecular Systems Biology*, 7(1), October 2011.

2. Stephen G. Addinall, Eva-Maria Holstein, Conor Lawless, Min Yu, Kaye Chapman, A. Peter Banks, Hien-Ping Ngo, Laura Maringele, Morgan Taschuk, Alexander Young, Adam Ciesiolka, Allyson L. Lister, Anil Wipat, Darren J. Wilkinson, and David Lydall. Quantitative fitness analysis shows that NMD proteins and many other protein complexes suppress or enhance distinct telomere cap defects. *PLoS Genet*, 7(4):e1001362+, April 2011.

[1]`http://cbo.biocomplexity.indiana.edu`
[2]`http://www.sbolstandard.org/`

3. Mélanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. MIREOT: The minimum information to reference an external ontology term. *Applied Ontology*, 6(1):23–33, January 2011.

4. Allyson L. Lister, Phillip Lord, Matthew Pocock, and Anil Wipat. Annotation of SBML models through rule-based semantic integration. *Journal of biomedical semantics*, 1 Suppl 1(Suppl 1):S3+, 2010.

5. Andrew R. Jones and Allyson L. Lister. Managing experimental data using FuGE. Methods in molecular biology (Clifton, N.J.), 604:333–343, 2010.

6. Allyson L. Lister. Semantic integration in the life sciences. *Ontogenesis*, `http://ontogenesis.knowledgeblog.org/126`. January 2010.

7. Allyson L. Lister, Ruchira S. Datta, Oliver Hofmann, Roland Krause, Michael Kuhn, Bettina Roth, and Reinhard Schneider. Live coverage of intelligent systems for molecular Biology/European conference on computational biology (ISMB/ECCB) 2009. *PLoS Comput Biol*, 6(1):e1000640+, January 2010.

8. Allyson L. Lister, Ruchira S. Datta, Oliver Hofmann, Roland Krause, Michael Kuhn, Bettina Roth, and Reinhard Schneider. Live coverage of scientific conferences using Web technologies. *PLoS Comput Biol*, 6(1):e1000563+, January 2010.

9. Allyson Lister, Varodom Charoensawan, Subhajyoti De, Katherine James, Sarath Chandra C. Janga, and Julian Huppert. Interfacing systems biology and synthetic biology. *Genome biology*, 10(6):309+, 2009.

10. Allyson L. Lister, Matthew Pocock, Morgan Taschuk, and Anil Wipat. Saint: a lightweight integration environment for model annotation. *Bioinformatics*, 25(22):3026–3027, November 2009.

11. Mélanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. MIREOT: the minimum information to reference an external ontology term. In Barry Smith, editor, *International Conference on Biomedical Ontology*, pages 87–90. University at Buffalo College of Arts and Sciences, National Center for Ontological Research, National Center for Biomedical Ontology, July 2009.

12. Allyson L. Lister, Phillip Lord, Matthew Pocock, and Anil Wipat. Annotation of SBML models through Rule-Based semantic integration. In Phillip Lord, Susanna-Assunta Sansone, Nigam Shah, Susie Stephens, and Larisa Soldatova, editors, *The 12th Annual Bio-Ontologies Meeting, ISMB 2009*, pages 49+, June 2009.

13. The OBI Consortium. Modeling biomedical experimental processes with OBI. In Phillip Lord, Susanna-Assunta Sansone, Nigam Shah, Susie Stephens, and Larisa Soldatova, editors, *The 12th Annual Bio-Ontologies Meeting, ISMB 2009*, pages 41+, June 2009.

14. Andrew R. Jones, Allyson L. Lister, Leandro Hermida, Peter Wilkinson, Martin Eisenacher, Khalid Belhajjame, Frank Gibson, Phil Lord, Matthew Pocock, Heiko Rosenfelder, Javier Santoyo-Lopez, Anil Wipat, and Norman W. W. Paton. Modeling and managing experimental data using FuGE. *OMICS: A Journal of Integrative Biology*, 13(3):239–251, June 2009.

15. Mélanie Courtot, William Bug, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. The OWL of biomedical investigations. In *OWLED 2008*, October 2008.

16. Susanna-Assunta Sansone, Philippe Rocca-Serra, Marco Brandizi, Alvis Brazma, Dawn Field, Jennifer Fostel, Andrew G. Garrow, Jack Gilbert, Federico Goodsaid, Nigel Hardy, Phil Jones, Allyson Lister, Michael Miller, Norman Morrison, Tim Rayner, Nataliya Sklyar, Chris Taylor, Weida Tong, Guy Warner, and Stefan Wiemann. The first RSBI (ISA-TAB) workshop: " can a simple format work for complex studies?". *OMICS: A Journal of Integrative Biology*, 12(2):143–149, June 2008.

17. Dawn Field, George Garrity, Tanya Gray, Norman Morrison, Jeremy Selengut, Peter Sterk, Tatiana Tatusova, Nicholas Thomson, Michael J. Allen, Samuel V. Angiuoli, Michael Ashburner, Nelson Axelrod, Sandra Baldauf, Stuart Ballard, Jeffrey Boore, Guy Cochrane, James Cole, Peter Dawyndt, Paul De Vos, Claude dePamphilis, Robert Edwards, Nadeem Faruque, Robert Feldman, Jack Gilbert, Paul Gilna, Frank O. Glockner, Philip Goldstein, Robert Guralnick, Dan Haft, David Hancock, Henning Hermjakob, Christiane Hertz-Fowler, Phil Hugenholtz, Ian Joint, Leonid Kagan, Matthew Kane, Jessie Kennedy, George Kowalchuk, Renzo Kottmann, Eugene Kolker, Saul Kravitz, Nikos Kyrpides, Jim Leebens-Mack, Suzanna E. Lewis, Kelvin Li, Allyson L. Lister, Phillip Lord, Natalia Maltsev, Victor Markowitz, Jennifer Martiny, Barbara Methe, Ilene Mizrachi, Richard Moxon, Karen Nelson, Julian Parkhill, Lita Proctor, Owen White, Susanna-Assunta Sansone, Andrew Spiers, Robert Stevens, Paul Swift, Chris Taylor, Yoshio Tateno, Adrian Tett, Sarah Turner, David Ussery, Bob Vaughan, Naomi Ward, Trish Whetzel, Ingio San Gil, Gareth Wilson, and Anil Wipat. The minimum information about a genome sequence (MIGS) specification. *Nature Biotechnology*, 26(5):541–547, May 2008.

18. A. L. Lister, M. Pocock, and A. Wipat. Integration of constraints documented in SBML, SBO, and the SBML manual facilitates validation of biological models. *Journal of Integrative Bioinformatics*, 4(3):80+, 2007.

19. Dawn Field, George Garrity, Tanya Gray, Jeremy Selengut, Peter Sterk, Nick Thomson, Tatiana Tatusova, Guy Cochrane, Frank O. Glöckner, Renzo Kottmann, Allyson L. Lister, Yoshio Tateno, and Robert Vaughan. eGenomics: Cataloguing our complete genome collection III. *Comparative and Functional Genomics*, 2007.

20. A. L. Lister, A. R. Jones, M. Pocock, O. Shaw, and A. Wipat. CS-TR number 1016: Implementing the FuGE object model: a systems biology data portal and integrator. Technical report, Newcastle University, April 2007.

Contents

Abbreviations

BFO	Basic Formal Ontology
BGLaV	BYU Global-Local as View
CID	CISBAN Interactomes Database
CISBAN	Centre for Integrated Systems Biology of Ageing and Nutrition
DL	Description Logic
EBI	European Bioinformatics Institute
EC	Enzyme Classification
EKoM	Entrez Knowledge Model
FuGE	Functional Genomics Experiment
GO	Gene Ontology
GWT	Google Web Toolkit
KiSAO	Kinetic Simulation Algorithm Ontology
LSID	Life Science Identifier
MFO	Model Format OWL
MIASE	Minimum Information About a Simulation Experiment
MIBBI	Minimum Information for Biological and Biomedical Investigations
MIRIAM	Minimum Information Requested In the Annotation of Biochemical Models
OBI	Ontology of Biomedical Investigations
OBO	Open Biomedical Ontologies
OWL	Web Ontology Language
OWL-DL	Web Ontology Language constrained by Description Logics
PFIN	Probabilistic Functional Integrated Network
PDF	Portable Document Format
RDBMS	Relational Database Management System
RDF	Resource Description Framework
RO	Relations Ontology
SBO	Systems Biology Ontology
S.cerevisiae	Saccharomyces cerevisiae
SBGN	Systems Biology Graphical Notation

SBML	Systems Biology Markup Language
SGD	Saccharomyces Genome Database
SPARQL	SPARQL Protocol and RDF Query Language
SQWRL	Semantic Query-Enhanced Web Rule Language
SRS	Sequence Retrieval System
SWRL	Semantic Web Rule Language
SyMBA	Systems and Molecular Biology Data and Metadata Archive
TEDDY	TErminology for the Description of DYnamics
W3C	World-Wide-Web Consortium
XSD	XML Schema Definition

Chapter 1

Background

1.1 Overview

Studying biological systems requires a large amount of data of different experimental types. Historically, each of these types is stored in its own distinct format, with its own internal structure for holding the data produced by those experiments. The use of community data standards can reduce the need for specialised, independent formats by providing a common syntax to make data retrieval and manipulation easier. However, standards uptake is not universal and the disparate data types required by systems biologists creates data that is not, or cannot, be completely described by a single standard. If existing data does not share a standard structure, theoretically any heterogeneous data could be reproduced in a single format by rerunning experiments. Because in practice such a method would be expensive and time consuming, integrative methods which reuse existing data should be explored.

Though it is possible for the biology represented by any given data format to be completely orthogonal with other experimental types, more commonly, portions of the biology—but not necessarily the formats describing them—overlap. These common aspects of biological data representations create theoretical integration points among the representations, allowing information reuse and repurposing. However, shared biological concepts do not necessarily result in shared definitions of the biology. Whereas differences in format result in *syntactic heterogeneity*, the differences in meaning of seemingly identical biological concepts across different formats results in *semantic heterogeneity*. A portion of the work presented in this thesis addresses syntactic heterogeneity both through the use of a common experimental metadata standard and by systematically integrating data for the purposes of systems biology model annotation. While syntactic heterogeneity can be resolved through the alignment of common structures, semantic heterogeneity is a more complex challenge. Once the meaning of the underlying biological concepts of interest for all data sources has been made explicit, the semantic heterogeneity can be identified. Further, if the semantics of a data format are accessible to machines, computational reasoning can be applied to find inferences and logical inconsistencies. The work described in this thesis includes the conversion of a systems biology standard specification—in multiple documents—into a single semantically aware model of that specification.

A wide variety of integration methodologies addressing various aspects of syntactic and semantic heterogeneity are available, often optimised for different situations. Many of these methods do not address semantic heterogeneity in systems biology data and of those that do, very few use existing technologies, address syntactic and semantic heterogeneity and make use both of simple syntactic conversions of non-semantic formats and semantically-meaningful models of the biological domain of interest. The work described in this thesis includes a method of semantic data integration called *rule-based mediation* which provides these features and which was developed as an aid to systems biology model annotation. Integrating resources with rule-based mediation accommodates multiple formats with different semantics and provides richly-modelled biological knowledge suitable for annotation of systems biology models.

Within this introductory chapter, Section 1.2 provides an overview of systems biology and the challenge presented when multiple formats are used. Section 1.3 describes how differences in format are not simply the result of different types of experiments, but are also due to the variety of ways systems are modelled. Section 1.4 describes the content, syntax and semantics standards relevant for systems biology. Data heterogeneity is an issue not limited to systems biology, and as such there is a large amount of previous work on data integration (see Section 1.6). As described in Section 1.5, existing technologies such as ontologies, rules and reasoning can bring together heterogeneous data in a homogeneous, meaning-rich, computationally-amenable manner.

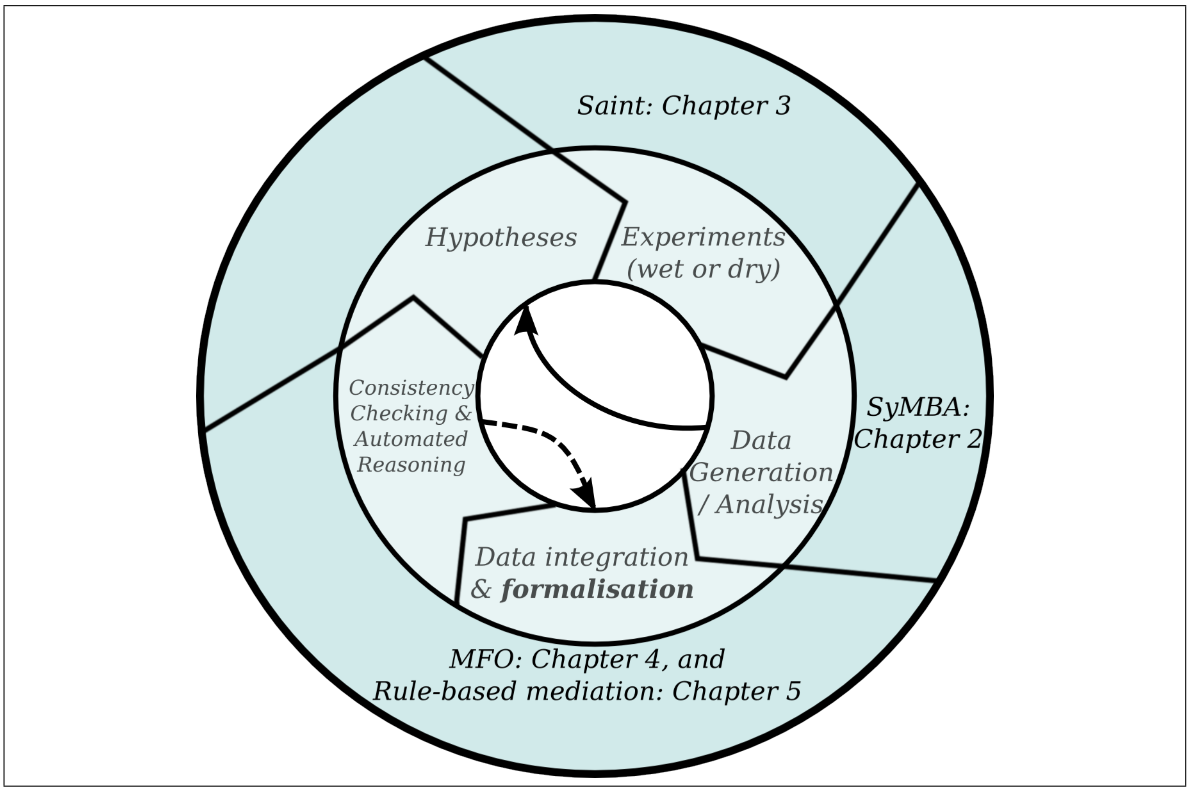

Saint: Chapter 3
Hypotheses
Experiments (wet or dry)
SyMBA: Chapter 2
Consistency Checking & Automated Reasoning
Data Generation / Analysis
Data integration & **formalisation**
MFO: Chapter 4, and
Rule-based mediation: Chapter 5

Figure 1.1: How the work described in this thesis relates to the semantic systems biology life cycle described further in Section 1.2, Figure 1.4. SyMBA provides a common structure for experimental metadata and a archive for experimental data of any type, thus aiding data storage and analysis. Saint helps systems biology modellers annotate models with biological information in a standard way, thus enhancing the quality of models used in *in silico* experiments. MFO and rule-based mediation use Semantic Web technologies to formalise systems biology data and perform automated reasoning over that data. Rule-based mediation also semantically integrates information relevant to systems biology.

Figure 1.1 provides a summary of the work described in the chapters of thesis in the context of the semantic systems biology life cycle originally described by Antezana and colleagues [11, Fig. 2]. Retrieval and storage of systems biology experimental metadata using a common syntax becomes easier with applications like the collaboratively-developed SyMBA (Chapter 2). The Saint Web application (Chapter 3) provides syntactic integration of systems biology data as well as a simple interface for viewing, manipulating and exporting new biological annotation to existing systems biology models. Saint is useful both in its own right and as a test case for determining the data sources, capabilities and requirements for the implementation of rule-based mediation. New data sources can easily be added, and therefore Saint has the capacity to provide access to data integrated via rule-based mediation.

In some cases, community standards and syntactic integration are not enough. Rules and restrictions on the use of a standard syntax are not always confined to the syntax itself; extra information can be present in human-readable documentation such as Word or Portable Document Format (PDF) documents, but not directly accessible to computers. Therefore, even if the data is in a common syntax, there are limits on its computational accessibility. This problem is resolved for Systems Biology Markup Language (SBML) models through the use of Model Format OWL (MFO) (Chapter 4), an ontology which holds SBML data as well as rules and restrictions on the SBML structure. MFO provides a format through which reasoning can be applied to SBML models, and stands both on its own and as part of the semantic data integration methodology described in Chapter 5.

Semantic data integration via rule-based mediation, described in Chapter 5, is a useful way of reusing and re-purposing heterogeneous datasets which cannot, or are not, structured according to a common standard. This method of integration is generic and can be used in any context, but has been implemented specifically to integrate systems biology data and to enrich systems biology models through the creation of new biological annotations. Syntactic heterogeneity is resolved through the conversion to a computationally-accessible syntactic ontology, and semantic heterogeneity is resolved by mapping the syntactic ontology to a biological domain of interest which is itself modelled using an ontology.

Chapter 6 discusses future possibilities for data use and reuse. Will data integration become a thing of the past? An increase in the uptake of standards will likely occur as communities mature. Experiments and their outputs could become better organised, cheaper and more open. Data usage would be much easier if integration were not required at all. Ultimately, however, science moves faster than standards and there are always new questions and new experiments. Hopefully, integration will become seamless and transparent to the user as semantic methods, combined with use of large, open resources on the Semantic Web, serve heterogeneous data via a homogeneous view.

1.2 What does systems biology data look like?

Properties of a system exist that are more than just the sum of their parts; systems that contain these emergent properties are said to be irreducible[1]. Though reductionist methods of research can provide a large amount of detail for specific biological entities, a more holistic systems approach is required to understand emergent systems properties [12]. Such a top-down approach creates a life cycle of systems biology research. Beginning with the Hodgkin–Huxley model of squid axons in 1952 [13], hypotheses have been tested both in the laboratory and through simulations of mathematical models. Data from the laboratory informs these models, which can then be used to inform further experimentation and validate or invalidate hypotheses.

Systems biology focuses on the study of systems as a whole rather than on the examination of individual constituent parts. Data useful to systems biology tends to be large and heterogeneous both in dimensionality and in structure, with modern high-throughput techniques collecting vast amounts of relevant information [14]. It is standard practice to take data points from a sample not just once, but across space, time, geographical location, organisational or even spectral range [15]. The wide variety of experimental types leads to a correspondingly large number of data representations, analysis methods and modelling strategies [16]. The reconciliation of disparate systems biology data, and the concomitant organisation and management of biological data sources into an exploitable "resourceome", is of great importance to researchers requiring access to existing data [11].

With the maturation of research methods, interpretations of the systems biology life cycle have become correspondingly more complex. Kitano detailed a relatively simple systems biology life cycle in 2002 which is summarised in Figure 1.2. By 2006, Philippi and colleagues had incorporated a data integration step as described in Figure 1.3. By 2009, semantics had become important enough to systems biology research that Antezana and colleagues had added formalisation of knowledge and reasoning to the cycle (see Figure 1.4).

Kitano's life cycle does not mention databases or integration of generated data with other data sources. Philippi and colleagues' modified life cycle has these additions as well as the acknowledgement that "dry" *in silico* experiments produce useful data independently of "wet" experiments. Historically data integration in bioinformatics consisted of cross references between databases or links out via URLs (see Section 1.6.3 for more information). More complex linking became common as ontologies such as the Gene Ontology (GO) [18] made it possible to reference community-wide hierarchies of descriptive biological terms.

[1] `http://www.systemsbiology.org/Intro_to_Systems_Biology/Why_Systems_Matter`, accessed December 2011.

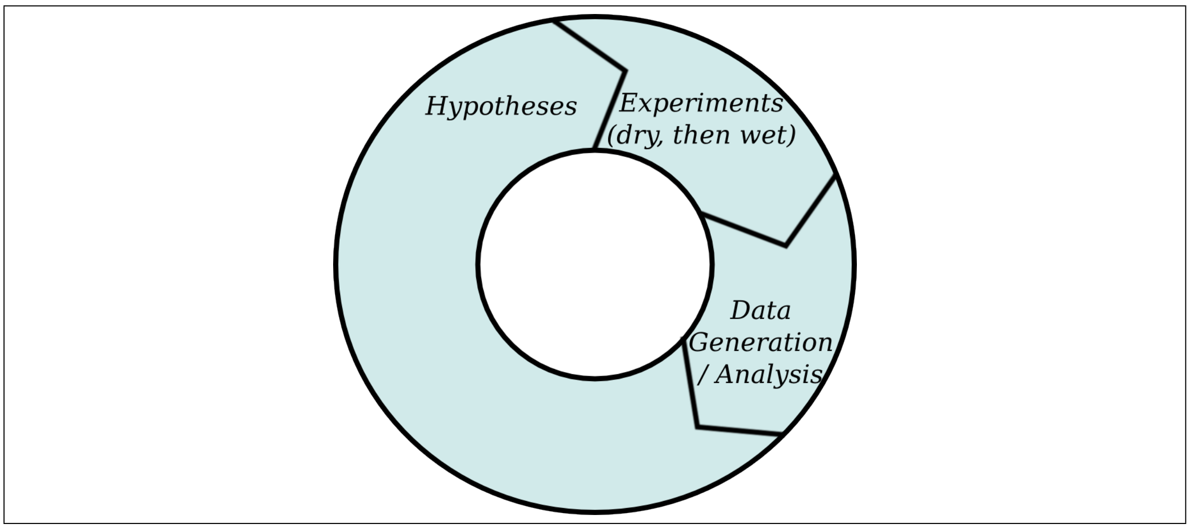


Figure 1.2: The systems biology life cycle in 2002, based on Kitano [14, Fig. 1]. “Dry”, *in silico* modelling and simulation experiments inform “wet” experiments, which in turn generate data used to create and further inform hypotheses.

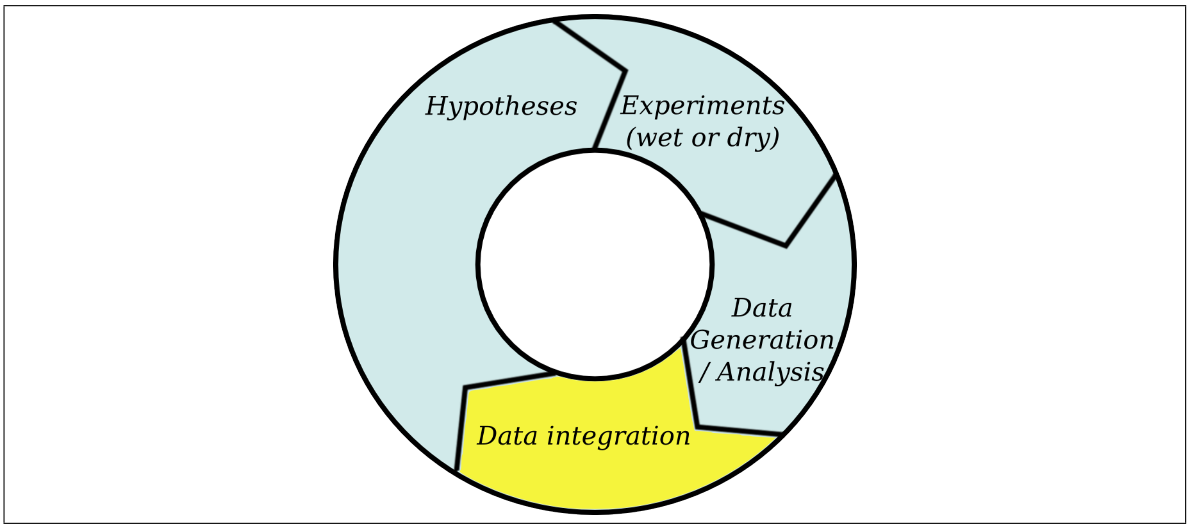


Figure 1.3: The systems biology life cycle in 2006, based on Philippi and colleagues [17, Fig. 1b]. Four years after the Kitano [14] life cycle was published, data integration methodologies, highlighted in yellow, were common enough to be added. Further, the entire cycle could be completed with either wet or dry experiments, or a combination of both.

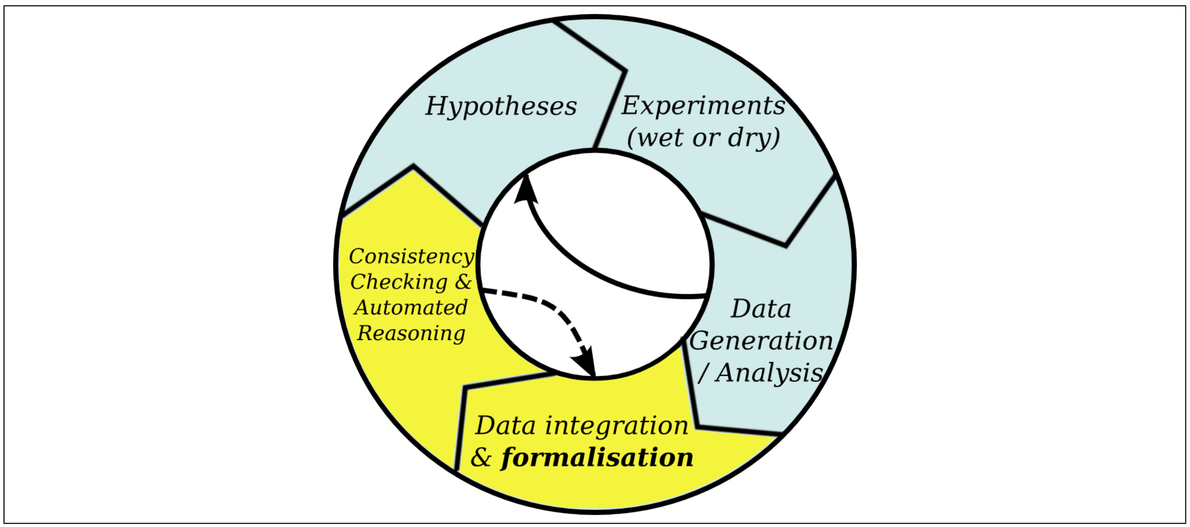

Hypotheses
Experiments (wet or dry)
Data Generation / Analysis
Data integration & **formalisation**
Consistency Checking & Automated Reasoning

Figure 1.4: The semantic systems biology life cycle in 2009, based on Antezana and colleagues [11, Fig. 2]. The new methods of integration and the addition of a reasoning step are highlighted in yellow. The semantic phase is iterative, shown with an arrow back to the integration and formalism step. The continued importance of the original Kitano life cycle is described with an arrow bypassing the semantic phase. While the original figure by Antezana and colleagues did not explicitly include a reference to *in silico* research, the experiments described in the paper could have been either dry or wet.

Very recently, with an increase in the use of Semantic Web[2] technologies such as ontologies, semantic data integration has become an important tool in systems biology research [11] (see Section 1.6). Figure 1.4 shows an interesting progression in the perception of researchers with regard to the systems biology life cycle with the addition of semantic techniques. By 2009, semantics and ontologies were becoming a bigger part of systems biology research. As such, Antezana and colleagues added the formalisation of data to the integration step, allowing data to be viewed in a semantically uniform way. The semantic data then becomes accessible to computational methods, allowing reasoning and consistency checking of the data. Even so, the research described in this thesis is one of only a handful of projects focusing on semantic data integration in systems biology.

There are four main areas of study in systems biology research: (i) the *structure* (e.g. interactions and pathways) of a system; (ii) how a system behaves over time, or its *dynamics*; (iii) the method of controlling and *modulating the system*; and (iv) the *design* method, or the deliberate progress using well defined design principles [14]. These four properties are strongly tied to the quantitative modelling aspect of systems biology, and illustrate the importance of such models. However, models are of limited use to either people or computers if they do not have structured biological annotations to provide context [19]. For instance, until Systems Biology Markup Language (SBML) [20] models are annotated by the BioModels team, elements often contain short-hand, biologically irrelevant

[2]http://www.w3.org/2001/sw/

names and descriptions in computationally incompatible free text [21]. While attaching additional biological knowledge to quantitative models is not a requirement for their simulation, without such annotations model sharing, interpretation of simulation results, integration and reuse becomes nearly impossible [19]. Therefore the addition of biologically relevant, computationally accessible metadata will not only enhance the semantics of a model but provide a method of unambiguously identifying its elements.

The majority of systems biology research projects can ultimately be interpreted to produce interconnected data such as gene networks, protein networks and metabolic networks [22]. The level of granularity of these networks of information can vary from large-scale omics networks with thousands of nodes to precisely calibrated quantitative models of specific molecular interactions. The integration of networks and models presents a challenge to systems biology, increasing the importance of bioinformatics techniques to the life science community, a result in opposition to early predictions [23]. In Section 1.3, the description of biological systems is examined through the use of networks and models.

1.3 Modelling biological systems

Le Novère described the 1952 Hodgkin–Huxley model of the squid giant nerve fibre [13] as the beginning of computational systems biology [24]. Since that time, systems biology models have been available in a variety of representations and with a large number of corresponding syntaxes. This variation is mainly a result of the different approaches used to model pathways and interactions in systems biology. Biological networks are generally qualitative and large-scale, and are created to provide a high level of granularity. Most networks do not yet have the information required to run successful mathematical simulations of the interactions under study; their remit is much broader and they are often composed of transitive binary interactions discovered in high-throughput experiments rather than complex biology-based pathways.

However, many experiments also produce quantitative data. For instance, high-throughput experimentation creates both qualitative and quantitative signalling pathway data important for the understanding of cellular communication [25]. Therefore modelling at the level of quantitative biological pathways is common in systems biology. Quantitative simulatable models are mainly used to enhance dynamic analyses and study biological pathways, and include specific details on parametrisation of those pathways. The quantitative description of a biological pathway or behaviour is essential for high-quality systems biology, informing hypotheses and creating an iterative cycle of prediction and experimental verification [14, 17, 25]. This section describes the basics of both networks and quantitative models in systems biology.

1.3.1 Networks in systems biology

To the majority of scientists, networks are perceived as views over sections of an *in vivo* cellular network [26]. There are five main types of network in systems biology: (i) transcription factor-binding networks, (ii) protein-protein interaction networks, (iii) protein phosphorylation networks, (iv) metabolic interaction networks and (v) genetic and small molecule interaction networks [27]. While such networks are just a conceptualisation of biological reality, they remain a useful virtual organisation of biological entities. For instance, network-mediated integration methodologies make use of the inherent graph-based organisation of networks to add many different datasets.

The CISBAN Interactomes Database (CID) integrates interaction data for multiple organisms into a weighted Probabilistic Functional Integrated Network (PFIN)[3]. CID allows users to determine the reliability of a particular connection between two interactors. Additional work has made use

[3]`http://cisban-silico.cs.ncl.ac.uk/cid.html`

of the inherent bias of a dataset to generate PFIN networks which include a relevance score [28]. Biases are exploited rather than eliminated; they can be introduced by an experiment being designed for a particular biological process or by choosing the final published data because it reflects the process of interest [28]. By adding a relevance score to the existing integrated confidence scores of a network, biased networks perform better than unbiased networks at gene function assignment and identification of important sets of interactors [28].

Genome-scale metabolic networks attempt to add new annotation as well as reconcile often-contradictory information present in the original networks and have been created for yeast [29, 30, 31] and human [32, 33]. The ultimate goal is to generate a view spanning an entire cellular network rather than sections of it.

Common formats for large-scale network data include linked data via the Resource Description Framework (RDF) [34] or semantically structured data via BioPAX [35]. RDF is a format which uses *triples* to create a directed, labelled graph of data and is the underpinning of the Semantic Web. A triple is similar to a sentence comprising a *Subject*, a *Predicate* and an *Object* [34]. A collection of RDF data can also be visualised as a graph where the Subjects and Objects are nodes linked together by Predicates, which form the edges between nodes.

Linked Life Data, one of the biggest networks of life science data, uses RDF to store and link its entities[4]. Linked Life Data contains over one billion entities and over five billion statements[5]. ONDEX [36] is a tool for data visualisation and integration which has been used to generate [37, 38, 39] and visualise [38] biological networks. While ONDEX can export data as RDF, internally it uses a labelled graph with a similar level of expressivity to RDF. BioPAX describes pathways in great qualitative detail, but does not have the capacity to store parameters or other quantitative data about pathways and interactions.

1.3.2 From networks to mathematical models

Although networks are generally qualitative, they can aid in the creation of quantitative models; indeed, mathematical modelling is one way of studying the complex behaviour of networks [25]. Some researchers are even blurring the line between networks and models, attempting to create genome-scale models with partial quantisation of data. These integrated consensus networks sit on the border between qualitative large-scale networks and quantitative small-scale pathway models. Herrgård and colleagues [29] created the “Yeast 1.0” consensus network by re-formatting existing

[4] `http://linkedlifedata.com`

[5] as of December 2011, `http://linkedlifedata.com/sources`

yeast interaction data in SBML [20], a structure more commonly used for quantitative modelling. Although Yeast 1.0 is represented in SBML, it does not yet contain enough quantitative data to be simulated.

Irrespective of a computational model's size reactions, effectors, kinetic rate equations and parameters of those equations need to be added for it to be fully functional [29]. Of those requirements, the original version of Yeast 1.0 only contained known reactions. Updates to the consensus network have vastly increased the connectivity of the nodes as well as the number of metabolites and enzymes, but have not yet increased its quantitative information [40]. Once complete, such network-scale models will benefit both from the large amount of data contained within them and from the ability to run *in silico* modelling experiments normally only accessible to smaller quantitative models.

Formats such as SBML are able to describe pathways quantitatively, and were created to provide machine-readable formats for model simulation. Other resources such as the BioPAX ontology were created to describe pathways qualitatively. Even with this logical division in purpose between qualitative and quantitative descriptions of systems biology, some projects have begun to bridge the divide. The network-scale Yeast 1.0 SBML model does not yet contain quantitative information. Rather than producing Yeast 1.0 for the purposes of simulation, there is a commitment to realistic representation and high quality selection of reactions [40]. Additionally, research into adding quantitative information to the qualitative pathway ontology BioPAX via SBPAX is progressing [41].

Work by Smallbone and colleagues [42, 43] uses flux balance analysis to create kinetic models of metabolism using only information regarding the reaction stoichiometries. Even though there is little experimental data for the variables, the dynamics concerning the concentrations of cellular metabolites can be inferred. Work on genome-scale kinetic models has progressed by adding the information from kinetic models stored within BioModels to this flux balance analysis method [43].

1.3.3 Quantitative modelling

Processes are the fundamental unit of systems biology, and the biological entities and associated quantitative data such as concentrations and rate constants are of prime importance for systems biology research [26]. The simulation and analysis of computational models that describe the dynamics of the interactions between biological entities is a vital facet of systems biology research [17]. Models and experiments are typically refined iteratively, as described in Section 1.2; models provide useful feedback to experimentalists, and experimental results help in turn to improve the models. The creation of systems biology models, such as those written in SBML or CellML [44], is primarily performed manually. When faced with such a time-consuming process, many researchers will not represent the biological context of a pathway or make use of many of the data sources and formats relevant to model development. While a small number of core databases can be used to retrieve a large amount of relevant biological information, accessing the "long tail" of information stored in other resources is a more complicated process. However, computational aids could help modellers retrieve new biological information easily and quickly.

Formats such as SBML and CellML provide machine-readable interpretations of biological pathways, complex formation and fundamental processes such as transcription and translation [45]. They are intended to make the task of understanding models easy for the programs that process them. The success of computational models in systems biology is not just shown by their prevalence in literature, but also by the 231[6] programs and applications making use of them. These programs allow the creation, simulation, analysis, annotation, and storage of SBML in a way that hides the underlying machine-readable format, making the model information accessible to humans. Further information on systems biology formats is available in Section 1.4.

1.3.4 Annotation of systems biology models

Systems biology models may contain both the quantitative information required to run a simulation of a biological system and biologically meaningful annotation describing the entities in the system. Annotation provides a description of how a model has been generated and defines the biology of its components in a computationally accessible fashion. However, while the mathematical information necessary for simulation models must be included, syntactically valid models capable of simulation are not required to contain explicit information about the biological context. Therefore, even though the presence of biological annotation aids efficient exchange, reuse, and integration of models, such information is often limited or lacking [21]. As a result, model usefulness is often limited to the person who created it; ambiguity in naming schemes and a lack of biological context hinders model reuse as an input in other computational tasks and as a reference for researchers [19, 46].

BioModels is a database of SBML models divided into curated and non-curated branches [21]. In the curated section, Minimum Information Requested In the Annotation of Biochemical Models (MIRIAM) [19] compliance is assured and biological semantics have been added. BioModels curators regularly add biological annotations to an entry prior to promotion to the curated branch. These annotations resolve ambiguity of identification through links to external resources using stable URIs

[6]as of December 2011, `http://sbml.org/SBML_Software_Guide`

as provided by the MIRIAM Registry[7]. However, model annotation either by BioModels curators or the modellers themselves is complex [21]. Programmatic methods to add such annotation would enrich publicly available models and therefore improve their quality and reusability.

In SBML, biological annotation is structured according to the MIRIAM specification [19]. There are three parts to MIRIAM: (i) a recommended URI-based structure for compliant annotations, (ii) a set of resources to generate and interpret those URIs and (iii) a checklist of minimal information requested in the annotation of biological models. While other annotations are allowed within the specification, MIRIAM annotations are the most relevant to the work presented here. MIRIAM annotations are added to models in a standardised way, and link external resources such as ontologies or data sources to a model. MIRIAM provides a standard structure for explicit links between the mathematical and biological aspects of a model. Aids to model annotation exist [47, 48, 41, 49, 50, 51], but rely extensively on the expert knowledge of the modeller for identification of appropriate additions. More information on such tools is available in Sections 3.6 and 5.1. Ultimately, there is a need for computational approaches that automate the integration of multiple sources to decrease the annotation burden on the modeller.

[7]`http://www.ebi.ac.uk/miriam`

1.4 Standards as a shared structure for data

As described in Section 1.3, systems biology benefits from the use of common standards for describing data and from unified naming schemes to ensure precise identification of entities. The research community must make use of standardised methods to increase the annotation of data to a point where it matches the rate of data generation [52]. Researchers must adapt, documenting and managing their data "with as much professionalism as they devote to their experiments" [53]. However, the level of standards support for consistent adoption and deployment is a difficult commitment for individuals [53]. Careful deployment by bioinformaticians can make standards use transparent to the researchers generating the data. This section provides an overview of the standards specific to systems biology as well as those useful both to systems biologists and the wider life sciences community.

Virtually every life science community has at least one proposed or accepted standard, including transcriptomics [54, 55], proteomics [56], genomics [57], flow cytometry [58] and systems biology [20, 59]. The Minimum Information for Biological and Biomedical Investigations (MIBBI) Registry alone contains 32 minimal information checklists for various experimental types[8]. Standards are difficult to create and maintain in terms of manpower, money, and consensus. Standards begin with islands of initial researchers in a field, who gradually develop into a nascent community. With new scientific developments, 'just-enough' strategies for storing and structuring data become 'nowhere-near-enough'. Communication with peers and among machines becomes more important and standards become a critical requirement.

The best solution may seem to be the creation of an elegant or complex ontology which models a domain in a richly described and logically rigorous way. However, the most practical solutions are generally easy to use and intuitive to understand, characteristics which do not always apply to more complex ontologies. Some solutions involve relying on realist upper level ontologies such as the Basic Formal Ontology (BFO) [60], but even experienced researchers find limitations and difficulties in such highly philosophical solutions [61, 62, 63, 64]. In addition to the complexity of the solution, the practicalities of creating a community standard require a large amount of effort and continued goodwill. Even with the best of intentions, with every participant working towards the same goal, it can take months—or years—of meetings, document revisions and conference calls to derive a working standard. For instance, Ontology of Biomedical Investigations (OBI) [65] has been in development for seven years and is yet to be officially published. Despite the time it takes to

[8]as of December 2011, http://mibbi.org/index.php/MIBBI_portal

reach consensus, even the best structure or semantics will not guarantee usage if the work has been developed without the input of the wider community.

While it may seem that only one standard is needed for each community, there are in fact two axes along which multiple standards can be developed. The first is the axis of *scope* and the second is the axis of *type*. Even though a single community may wish to describe a single type of information (for instance, a systems biology model), the amount and scope of the data might be too large to feasibly fit into just one standard. There are three types of standard which must be considered: the minimal descriptive *content*, or metadata, a piece of data must include; the standard *syntax* to which that data must adhere; and a standard *semantics* for describing the meaning of the data in a way understandable to both computers and humans. Unlike the scope of a standard, which changes depending on the community, standards types are consistent across communities. Table 1.1 summarises these axes with the use of existing systems biology community standards. The standards relevant to systems biology and to the work presented in this thesis are described in Sections 1.4.1 and 1.4.2.

		Scope		
		Representation	**Simulation/Analysis**	**Behaviour**
Type	**Content**	MIRIAM	MIASE	-
	Syntax	SBML, SBGN, CellML, VCML, BioPAX*	SED-ML	-
	Semantics	SBO, BioPAX*	KiSAO	TEDDY

Table 1.1: The two axes of standards development, illustrated using systems biology community standards. The horizontal axis is the scope of the standard and the vertical axis is the type of standard. The content and syntax standards for system behaviour are empty as these standards have not yet been finalised. *BioPAX is both a syntax standard and a semantic standard. Written in OWL, it provides both the structure and the meaning for representing pathways qualitatively. Adapted from [66], first presented in [67].

Some researchers may wish to describe or visualise a mathematical model while others might be more interested in storing the simulation details or the results of multiple simulation runs. Though mathematical descriptions, visualisations and results storage are all separate activities, they fall within a single community and are related via the computational model itself. These activities create the scope of the various systems biology standards: one representing the model, another describing a simulation and a third structuring the results of a simulation. These divisions are present within the SBML community, and at least three other similar – but not identical – representation standards have also been developed in the systems biology community as a whole [35, 68, 69]. The columns in Table 1.1 sort the most common systems biology standards according to scope.

Discovering where to delineate areas within a single community's standards can be difficult. Further, the scope of a standard might overlap with another community's, requiring cross-community participation. Such efforts result in higher-level standards which many different communities can utilise. Resources such as OBI and the Functional Genomics Experiment (FuGE) [2] standards have a very broad scope and a correspondingly high level of granularity, enabling the description of virtually any experiment. Individual communities are meant to extend these upper-level standards to provide terminology and structure to meet their specific needs at a lower level of granularity. By sharing a common upper-level standard, integration and reuse of the data it organises becomes much easier. The interconnectedness of standards is a task in itself, and one for which dedicated organisations such as BioSharing [70] and the MIBBI Registry [71] were created.

1.4.1 Systems biology modelling standards

This section describes a number of standards important in systems biology according the type of the standard, as described in Table 1.1. Greater detail is provided for those standards used extensively in the work described in this thesis. In particular, a comparison of the main systems biology formats is provided in Figure 1.5. Other cross-community standards commonly used in systems biology are described as they are introduced.

1.4.1.1 Content standards

Content standards provide a checklist for the minimal descriptive content an experimental type must include. The results and conclusions of a scientific investigation are dependent upon the investigation's context, such as the methods and other metadata describing an experiment. Defining a common set of metadata to guide researchers in reporting scientific context is increasingly gaining favour with data repositories, journals and funders [71]. Checklists outline the minimal information required to evaluate, interpret and disseminate an experiment; such guidelines effectively define what is contained within a scientific dataset and how the set was generated. Currently, the MIBBI Registry provides a list of these guidelines[9].

MIRIAM is a content standard which provides a minimal checklist for interpreting a model correctly, a controlled method of providing annotation through URIs [19] and, via the MIRIAM Registry, Web services for correctly resolving these URIs and for listing supported data types [46]. MIRIAM annotations are URIs which are added to SBML in a standardised way and link external resources such as ontologies or data sources to a model. MIRIAM provides a standard scheme for unambiguously identifying biological entities in networks and models; without such a scheme, the quality of the data

[9]http://www.mibbi.org

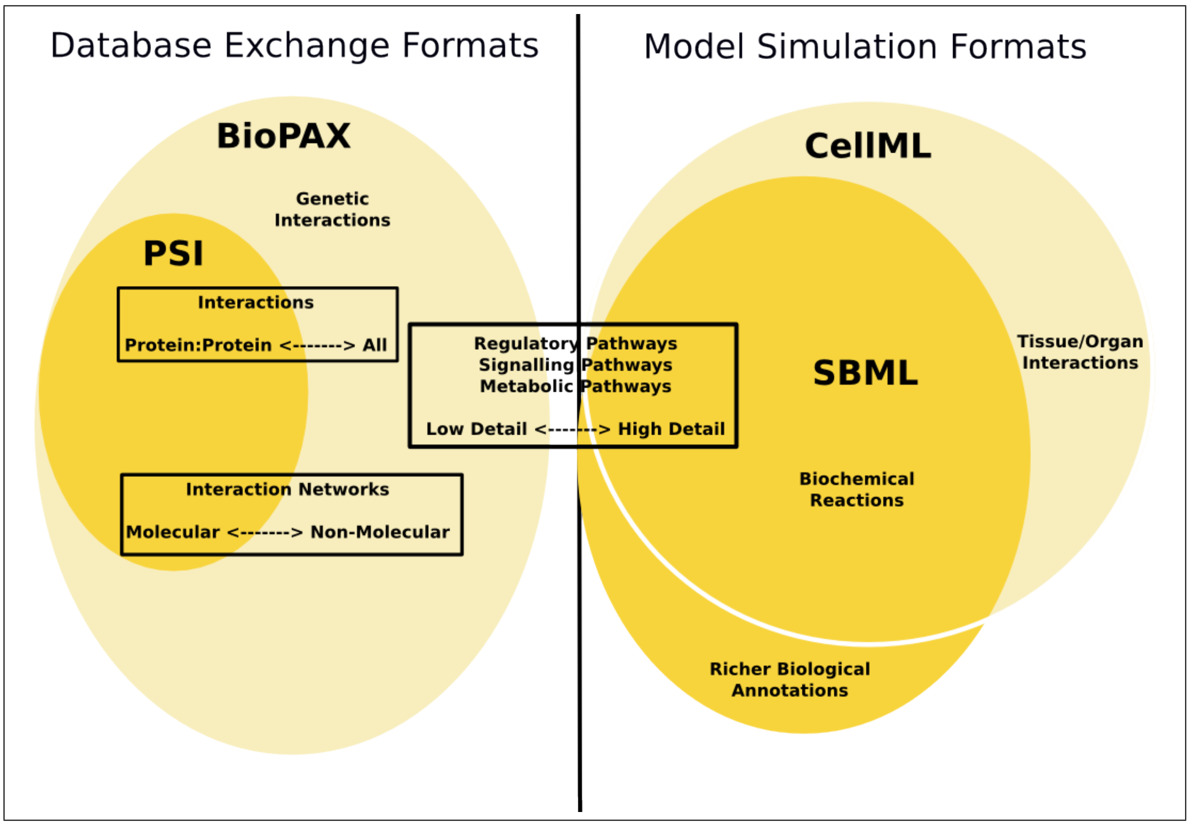

Database Exchange Formats
Model Simulation Formats
BioPAX
Genetic Interactions
PSI
Interactions
Protein:Protein <-------> All
Interaction Networks
Molecular <-------> Non-Molecular
Regulatory Pathways
Signalling Pathways
Metabolic Pathways
Low Detail <-------> High Detail
CellML
SBML
Tissue/Organ Interactions
Biochemical Reactions
Richer Biological Annotations

Figure 1.5: A graphical comparison of the capabilities of BioPAX, SBML, PSI-MIF and CellML. While CellML and SBML can model many of the same concepts, CellML is also capable of modelling tissue and organ interactions, and SBML currently provides richer biological annotations. While PSI and BioPAX can both be used to model interactions, BioPAX is able to model more types of interaction as well as pathways. The figure shows BioPAX as capable of modelling pathways in low detail because BioPAX does not capture quantitative information. Modified from [72].

suffers [40]. By providing an integration methodology which enhances MIRIAM annotations, this thesis aids collaboration and data reuse in systems biology.

While models complying with MIRIAM provide consistently named annotations about the model to allow for its correct interpretation and reuse, the MIRIAM checklist does not cover the reproduction of simulation results. For this, the Minimum Information About a Simulation Experiment (MIASE) was created [73]. This checklist describes the models to use, the modifications made to those models, the order in which all simulation procedures were applied, how the raw results were processed and a description of the final output [73].

1.4.1.2 Syntax standards

While the difficulty inherent in translating native data schemas to a unified format is just one of the integration challenges facing researchers [17], it can also be one of the most easily remedied if there is a strong focus within a community for creating a common syntax. Choosing a structure for the data such as XML, RDF or even structured flat files creates a single format and eases data sharing and reuse. Syntax standards aim to provide a common structure for all data of a given experimental type.

SBML is a syntax standard which allows the exchange and reuse of quantitative models for systems biology [20]. SBML is primarily an XML format for describing computational models in systems biology. It is supported by a large community and a wide range of tools, allowing model generation, analysis and curation in any one of the many independently maintained software applications[10]. While the majority of models written in this format describe relatively small and well contained pathways, SBML is capable of storing larger-scale views of entire metabolic networks [29]. SBML models tend to remain small due to limitations in both the simulation environments and in the kinetic data available for parametrising the models. Herrgård and colleagues [29] sidestep this problem by currently providing only qualitative information for their large SBML metabolic network, without parametrisation.

Metadata, or extra information about any component of an SBML model, is stored in two ways: (i) as a link to an Systems Biology Ontology (SBO) [74] term and (ii) as an RDF triple structured according to the MIRIAM specification. In the SBML data model, each element inherits from the SBase top-level class which has an optional attribute sboTerm for referencing a specific term in SBO [75]. Further biological annotations can be added to the annotation element of any SBase class. Although the annotation element may contain any RDF, annotation complying with the MIRIAM specification must be in the form of valid MIRIAM identifiers. Resolution of these identifiers is available from `http://www.identifiers.org`. An example of annotation within SBML is shown in Figure 1.6. A detailed description of the components and constraints on an SBML model is available in the specification document [76].

The Systems Biology Graphical Notation (SBGN) is the first community standard for the graphical representation of systems biology models [77]. Though many different notations were available prior to SBGN, those efforts dealt mainly with notation proposals and software implementations without seeking the backing of the entire community and without addressing the many biological and technical needs of the users. Specifically, SBGN was created for the following purposes: to be semantically, syntactically and visually concise and unambiguous; to be free of copyright restrictions; to minimise the number of possible symbols; to support modularity as well as many different biological entities; and to support the automated generation of diagrams based on simulatable models [77]. Conversion between SBGN and a descriptive format such as SBML is only possible through the shared semantics

[10]`http://sbml.org/SBML_Software_Guide`

```
<species metaid="_000003" id="BLL" name="BasalACh2"
     compartment="comp1" initialAmount="0"
     sboTerm="SBO:0000297">
  [...]
  <annotation>
    <rdf:RDF>
      <rdf:Description rdf:about="#_000003">
        <bqbiol:isVersionOf>
          <rdf:Bag>
            <rdf:li rdf:resource="urn:miriam:interpro:IPR002394"/>
            <rdf:li rdf:resource="urn:miriam:obo.go:GO%3A0005892"/>
          </rdf:Bag>
        </bqbiol:isVersionOf>
      </rdf:Description>
    </rdf:RDF>
  </annotation>
</species>
```

Figure 1.6: How SBO and MIRIAM annotation are used within an SBML element. MIRIAM-compliant RDF is present within the annotation element and via the sboTerm attribute linking to the SBO term SBO:0000297, “protein complex”. RDF namespaces of the rdf element and some children of the species element have been removed to aid readability of the figure. The identifiers used in this example are URNs rather than URIs; BioModels is in the process of converting all URNs to identifiers.org-based URIs. XML taken from the “BLL” species of BioModels entry BIOMD0000000001.

such as those provided by SBO [75].

SED-ML is the format counterpart to the MIASE checklist and provides an XML structure for describing simulations that are independent of both the model encoding and the software used to perform the simulations [78]. SED-ML tasks are used to link a model to a particular simulation setting and a DataGenerator is used to structure the post-processing steps used on the simulation result before final output [78].

Like SBML, CellML is an XML format used to encode and exchange quantitative systems biology models. Unlike SBML, CellML is able to describe a wider range of mathematical expressions [44]. Additionally, CellML has a component-based structure, allowing reuse of individual modules of a model in a way currently impossible for SBML[11]. However, SBML user support and software functionality is much higher than that provided by CellML, and SBML has a much more active user community. While historically SBML was able to provide richer biological annotations, recent developments within CellML have all but resolved this limitation[12].

The Physiome project stores models of integrative functions of cells, organs and organisms [79]. This project was expressly developed for modelling at both a cellular and at an organ level, and the models can vary from non-simulatable diagrammatic schema to fully quantitative computational models[13]. A final example is the Virtual Cell project, which provides not just the VCML XML format for describing mathematical and biological models for simulation, but also an entire software infrastructure and user interfaces for performing creation, simulation and analysis of those models [80].

1.4.1.3 Semantic standards

Describing the meaning, or semantics, of data and its associated metadata is a complex and difficult problem being addressed in the life sciences through the use of controlled vocabularies and ontologies [81]. The use of ontologies for describing data has a twofold purpose. Firstly, ontologies help ensure consistent annotation, such as the spellings and choice of terms. Consistent naming of terms allows the use of a single word-definition pair to describe a single concept. Secondly, ontologies can add human- and computationally amenable semantics to the data. The curation of datasets with common ontology terms minimises querying and integration errors due to semantic ambiguity by providing a method of consistent annotation; for instance, GO is a community-driven ontology in widespread use, and its presence in many datasets creates semantic links between

[11] However, an SBML package is being developed to allow hierarchical composition: http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Hierarchical_Model_Composition.

[12] http://www.cellml.org/specifications/metadata/mcdraft

[13] http://www.physiome.org/About/index.html

them [82, 83, 84, 85, 81]. In addition to ontologies, RDF can be used to organise life science data in a generic, high-level fashion. RDF can provide a simple, single format for scientific data[14], but cannot provide a biologically meaningful semantic layer.

SBO is a semantic standard initially developed for the addition of a biologically meaningful semantic layer on quantitative systems biology models [86]. SBO provides unambiguous terms for biological annotation [74], therefore increasing MIRIAM compliance. Initially, only SBML models supported SBO's use; currently many other formats such as CellML use SBO. Further, SBO is capable of more than simple entity annotation:

- every SBGN glyph corresponds exactly to an SBO term, allowing model conversion from a simulation format such as SBML to the graphical SBGN notation;
- SBO aids conversion between pathway formats utilising its terms;
- SBO-annotated models can be translated between continuous deterministic frameworks and discrete stochastic frameworks; and
- models can be merged more cleanly and precisely when model entities are unambiguously defined with SBO [75].

While SBO is natively stored as a relational database, it can be exported either in Open Biomedical Ontologies (OBO) [81] or OWL on demand[15]. The only property used within SBO is the subsumption, or "is a" relationship. A summary of the main classes and their children is available in Figure 1.7.

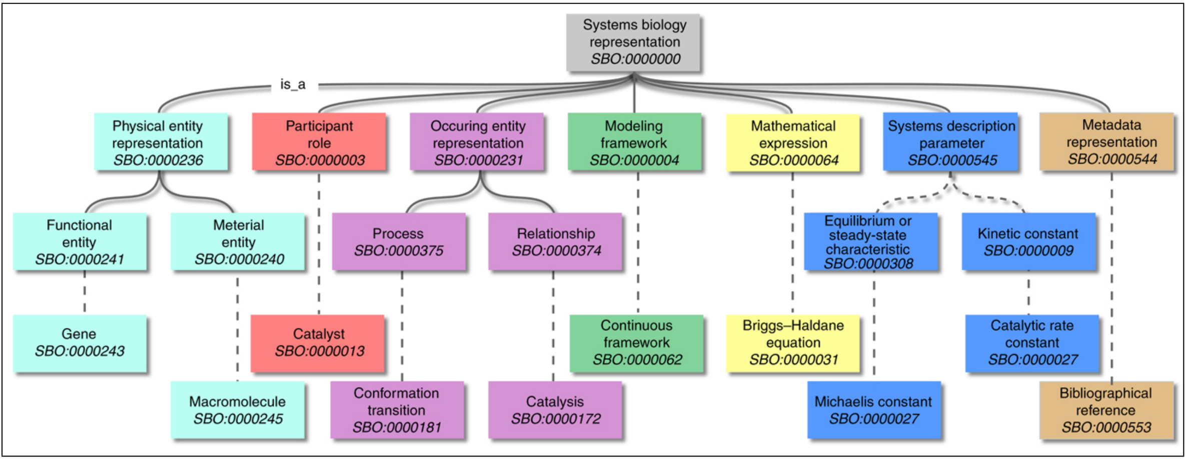

Systems biology representation SBO:0000000
is_a
Physical entity representation SBO:0000236
Participant role SBO:0000003
Occuring entity representation SBO:0000231
Modeling framework SBO:0000004
Mathematical expression SBO:0000064
Systems description parameter SBO:0000545
Metadata representation SBO:0000544
Functional entity SBO:0000241
Meterial entity SBO:0000240
Process SBO:0000375
Relationship SBO:0000374
Equilibrium or steady-state characteristic SBO:0000308
Kinetic constant SBO:0000009
Gene SBO:0000243
Catalyst SBO:0000013
Continuous framework SBO:0000062
Briggs–Haldane equation SBO:0000031
Catalytic rate constant SBO:0000027
Macromolecule SBO:0000245
Conformation transition SBO:0000181
Catalysis SBO:0000172
Michaelis constant SBO:0000027
Bibliographical reference SBO:0000553

Figure 1.7: A summary of the SBO subsumption hierarchy, taken from [74, Box 1]. The seven orthogonal branches of the SBO hierarchy each have their own colour. Dashed lines indicate that intermediate terms have been removed from the summary for readability.

All information other than the SBO class names, identifiers, synonyms and subsumption hierarchy

[14]e.g. Linked Life Data (http://biolod.org/) and BioLOD (http://linkedlifedata.com)
[15]http://www.ebi.ac.uk/sbo/main/download.do

is contained either within human-readable annotations on each SBO term or within MathML annotations. Human-readable annotations include general comments, history of the term and a textual definition. Any constraints on the usage of an SBO term is not defined in the ontology itself, but rather in the other formats (such as SBML) where the constraint is applied.

The MIASE checklist requires that the applied algorithm and the initial set-up for a model simulation are described. However, as some algorithms are proprietary and others are not well documented, repeatability can be difficult. Kinetic Simulation Algorithm Ontology (KiSAO) is an ontology which describes algorithms, unambiguously identifying those which are similar enough to perform a particular simulation task [74][16]. Unlike SBO, the native encoding of KiSAO is OWL. KiSAO can be used in conjunction with SED-ML to allow software to automatically choose the best algorithm for a simulation.

TErminology for the Description of DYnamics (TEDDY) is an ontology which models the dynamic behaviour, observable phenomena and control elements of systems and synthetic biology models [74][17]. It is still at an early stage of development, but the ontology and some limited documentation are available. Like KiSAO, its native format is OWL.

BioPAX is an OWL ontology created to qualitatively describe biological pathway information [35]. It uses GO and the cell type ontology [87] to describe compartments and locations as well as the NCBI taxonomy database for organisms [88]. In contrast to other standards such as SBML which provide a syntactic representation, BioPAX provides a semantic representation of pathways. An example of BioPAX properties and classes is available in Figure 1.8 and a summary of classes in BioPAX Level 3 is available in Figure 1.9[18].

While SBML is primarily intended for quantitative encoding of pathways for simulation and PSI-MIF is capable only of storing binary interactions, BioPAX is intended to store both levels of granularity equally well, albeit at a qualitative level [89]. For a graphical representation of the differences between BioPAX, SBML, PSI-MIF and CellML, see Figure 1.5. Each release, or level, of BioPAX has increased the expressivity of the ontology. The latest version of BioPAX is Level 3, which is capable of modelling signalling pathways, molecular state, gene regulation and genetic interactions [35]. As a comparison, the earliest release, Level 1, supported only metabolic pathways. However, most databases such as Pathway Commons still provide their data in BioPAX Level 2, which adds molecular interactions and post-translational modifications.

[16]http://biomodels.net/kisao
[17]http://biomodels.net/teddy/
[18]The entities shown in this figure are very similar between Level 2 and Level 3.

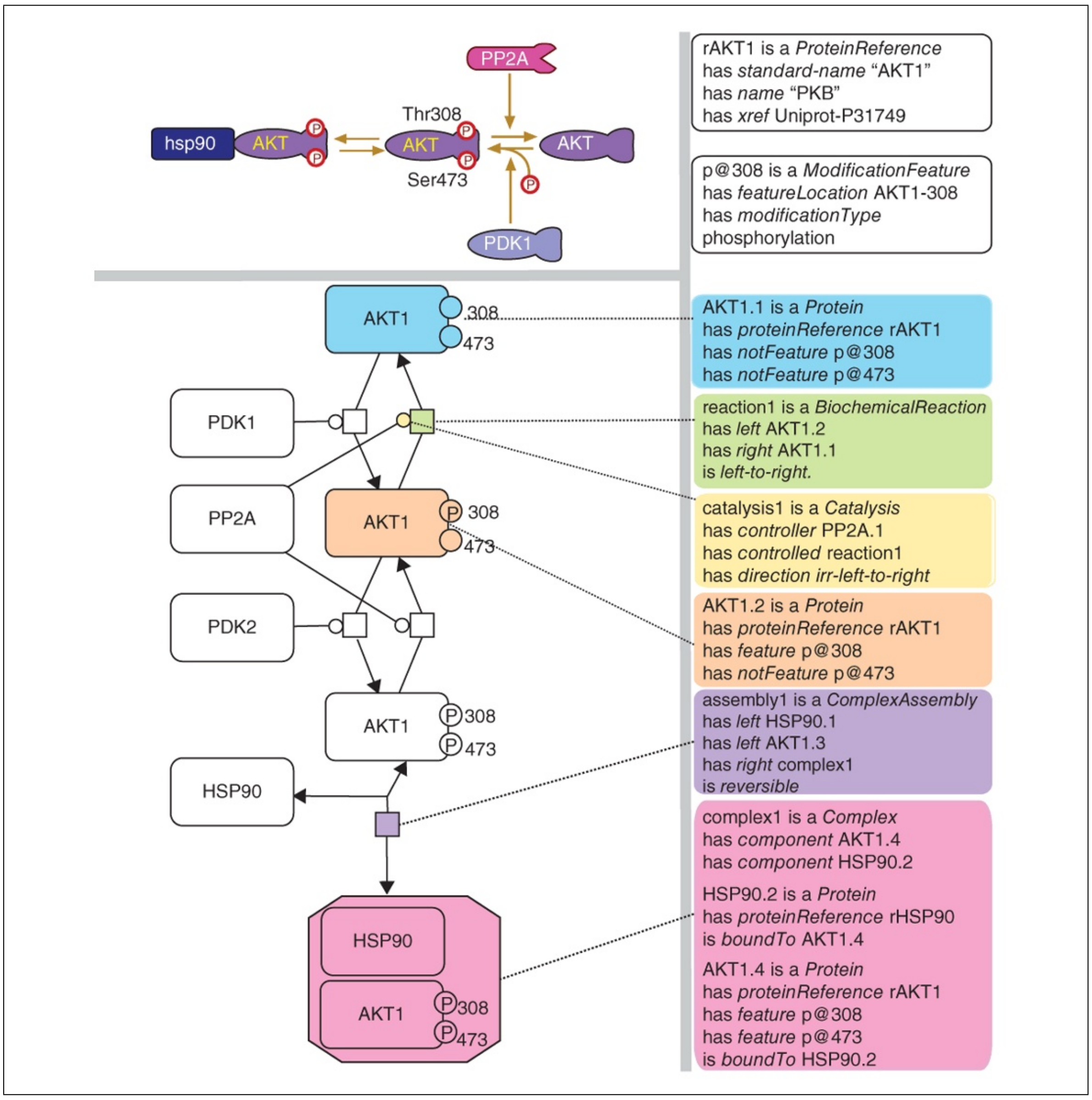

PP2A
hsp90
AKT
Thr308
AKT
Ser473
AKT
PDK1
rAKT1 is a ProteinReference has standard-name "AKT1" has name "PKB" has xref Uniprot-P31749
p@308 is a ModificationFeature has featureLocation AKT1-308 has modificationType phosphorylation
AKT1
308
473
PDK1
PP2A
AKT1
308
473
PDK2
AKT1
308
473
HSP90
HSP90
AKT1
308
473
AKT1.1 is a Protein has proteinReference rAKT1 has notFeature p@308 has notFeature p@473
reaction1 is a BiochemicalReaction has left AKT1.2 has right AKT1.1 is left-to-right.
catalysis1 is a Catalysis has controller PP2A.1 has controlled reaction1 has direction irr-left-to-right
AKT1.2 is a Protein has proteinReference rAKT1 has feature p@308 has notFeature p@473
assembly1 is a ComplexAssembly has left HSP90.1 has left AKT1.3 has right complex1 is reversible
complex1 is a Complex has component AKT1.4 has component HSP90.2
HSP90.2 is a Protein has proteinReference rHSP90 is boundTo AKT1.4
AKT1.4 is a Protein has proteinReference rAKT1 has feature p@308 has feature p@473 is boundTo HSP90.2

Figure 1.8: The AKT pathway shown graphically (left) and using BioPAX (right). Taken from [35, Figure 3].

Although BioPAX provides a detailed semantic representation of pathways and interactions, it has a number of limitations: there is no way of explicitly describing broader experimental metadata other than through simple cross-references [89, 90], no ability to represent mathematical relations other than providing chemical details about interactions [89, 90], and dynamic and quantitative aspects of processes are not supported [35]. Although there is a BioPAX export available for all entries in the BioModels database, such a conversion is not perfect. However, a new bridge between SBML and BioPAX in the form of a quantitative module for BioPAX, called SBPAX, is under development[19] [41].

[19]http://sbpax.org/

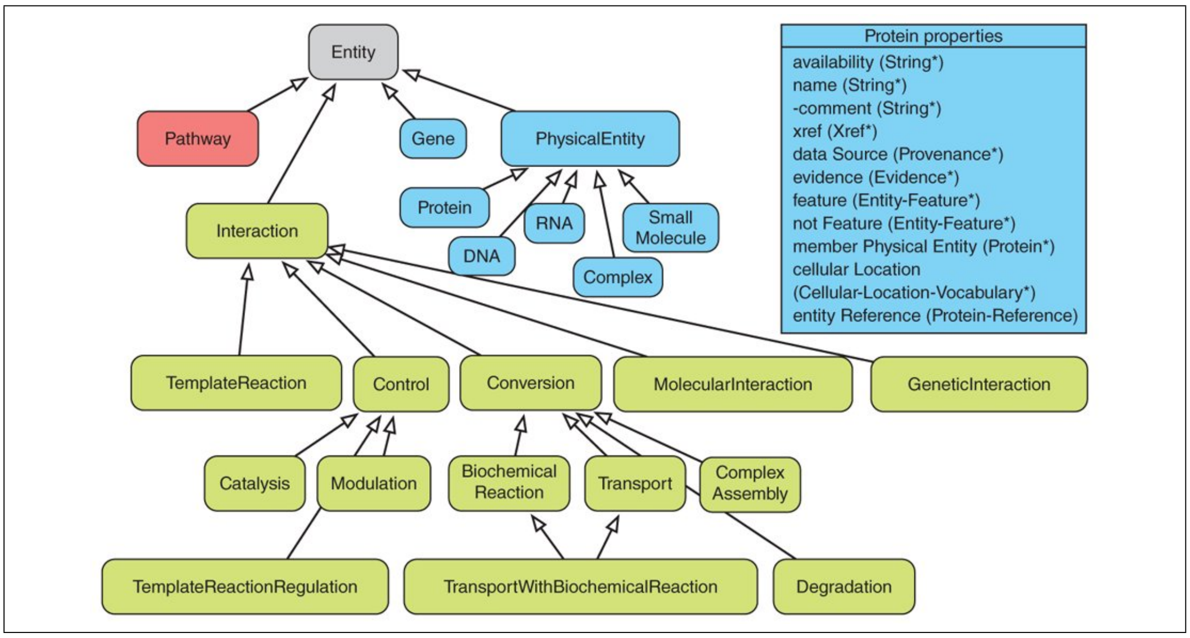


Figure 1.9: High level overview of BioPAX Level 3. Taken from [35, Figure 4].

1.4.2 Data Sources for Systems Biology

This section describes three commonly used data sources for the creation of systems biology models which have been used in the work described in this thesis (see Chapter 5). This list is not intended to be exhaustive, and many other databases are available[20]. For information on BioModels, a database for storing systems biology models, see Section 1.3.4.

1.4.2.1 BioGRID

BioGRID stores 24 different types of interactions and exports pairs of interacting entities in PSI-MIF 2.5 format [56]. As described in Figure 1.5, Pathway Commons and BioGRID store similar types of data, but have different underlying representations. As this thesis was being completed Pathway Commons began importing limited BioGRID data, allowing retrieval of some BioGRID data in BioPAX format and simplifying the integration process for that portion of BioGRID.

1.4.2.2 UniProtKB

The UniProtKB is a comprehensive public protein sequence and function database, consisting both of manually curated and automatically annotated data [1]. While some limited pathway and interaction data is provided, mainly within the comments and feature table sections of a UniProtKB entry,

[20]For example, see the "Metabolic and Signalling Pathways" subsection of the 2011 Nucleic Acids Research Database Summary Paper Category List: http://www.oxfordjournals.org/nar/database/c.

its main use in the creation of systems biology models is as a high quality reference for protein information. It also contains 144 cross references to other resources such as GO and IntAct [91], many of which are useful for model creation.

1.4.2.3 Pathway Commons

Pathway Commons is a database which provides pathway and interaction data in a number of formats, including BioPAX. The Pathway Commons binary interaction data has limitations not present in the BioPAX format itself. For binary interactions, Pathway Commons describes the participants in a reaction without providing any directionality. Specifically, the participant subtypes—product, reactant, and modifier—available within BioPAX are unused. Where pathway (rather than interaction) data is provided from Pathway Commons, more complete use of BioPAX is possible. Pathway Commons uses BioPAX Level 2, which has a number of limitations compared with BioPAX Level 3, as described in Section 1.4.1.3.

1.4.3 Data standards and data integration

Although data standards are key to data sharing and reuse, they do not provide a complete solution. By their very nature, standards cannot be implemented until a new experimental type has been available long enough to produce a list of standard requirements. Further, while orthogonality is important, in practice there remains some level of overlap among standards. As such, the integration of information stored in different representations remains important. To integrate systems biology data successfully, the data needs to be human readable and computationally accessible. Section 1.6 describes data integration methodologies and their use within systems biology.

1.5 What are ontologies?

1.5.1 Introduction

Controlled vocabularies, often just a list of agreed-upon terms, have been used to describe data since the early releases of the first biological databases such as EMBL [92] and Swiss-Prot [1]. As data handling in the life sciences matured, more complex methods of categorising terms and their relationships developed. Ontologies are one such method, and their use has become pervasive within the life sciences. For example, 3137 papers have cited the main GO [18] publication as of December 2011[21]. As their production is the result of scientific research and activity, ontologies have become accepted as "first-class citizens" of science, and their evaluation and maintenance are considered part of that activity [93]. Ontologies are mainly used in systems biology as a method of adding extra levels of information to core data. This extra information is called metadata, and describes the context of the data. For instance, author information, related publications and links to biological databases are all useful context for a systems biology model, but are not required for its simulation. While helpful even when used as a controlled vocabulary, an ontology can also be a primary structure for the data and, when created with the appropriate semantics, can be used in automated integration tasks [94]. The semantic data integration methodology described in this thesis uses ontologies, rules and reasoning to convert heterogeneous data into homogeneous knowledge.

Ontologies:

- define a *clear, logically consistent structure* for information that can be shared among people and is understandable to computers;
- enable *reuse of knowledge* both within a single domain and across multiple domains;
- *explicitly describe information* about a research domain;
- *separate* domain knowledge from the operational knowledge;
- can be reasoned over for the purposes of analysis and *drawing inferences from domain knowledge* that would otherwise remain hidden [95].

Within the life sciences, ontologists create ontologies both as reference works for a particular domain of interest and as a specific tool for a specific application. Reference ontologies such as GO provide terms commonly used for generic tagging of database entries, whereas application ontologies are designed for a specific purpose such as part of a bioinformatics tool. Section 1.5.2 provides an explanation of ontologies with respect to other classification schemes, while commonly used definitions of

[21]http://www.ncbi.nlm.nih.gov/pubmed?db=pubmed&cmd=link&linkname=pubmed_pubmed_citedin&uid=10802651, accessed 3 August 2011

an ontology and strategies for modelling are examined in Section 1.5.3. Ontology languages used in biology are described in Section 1.5.4. Section 1.5.5 describes the structural components of ontologies. Naming conventions used in this thesis are listed in Section 1.5.6, and the choice of ontology language and modelling strategy is explained in Section 1.5.7.

1.5.2 From tag clouds to ontologies

The simplest way to categorise objects is via social tagging, where people add whatever tag or tags they feel are best suited to describe an object, thus creating folksonomies [96]. There are no definitions for these tags and there is no way to relate tags to each other. Frequencies of words can be analysed, and tag clouds are often created to display relative frequencies. The benefit of folksonomies is that they are free-form, with no constraints on the words people use to describe their objects, making them highly accessible to the general public. However, free tagging can lead to a large amount of near-identical tags for the same concept, making searches difficult. Folksonomies and ontologies are different, rather than opposing, technologies; where ontologies are an enabling technology for sharing information, folksonomies are emergent properties of shared information [97].

Controlled vocabularies are lists of terms that have been enumerated explicitly, of which there are four main types: simple lists, synonym rings, taxonomies and thesauri [98]. A list is a set of terms whose membership is limited, while synonym rings are used to broaden search queries according to predetermined lists of synonyms. Taxonomies are hierarchical controlled vocabularies where the hierarchy is determined by a single named relationship [98]. Thesauri are taxonomies that have additional standardised relationship indicators such as “equal to” or “related to”. Ontologies provide an extra level of expressivity and complexity by allowing specialised relationships beyond those available with any controlled vocabulary, such as one entity being a part of another. The W3C perceives ontologies as critical for searching and merging information [99].

1.5.3 Definitions

Ontologies are models of domains of interest, typically defining concepts using formal logic-based representations as well as relationships between, and constraints on, those concepts. In one of the most widely cited computer science definitions of an ontology, Thomas Gruber states that an ontology is “an explicit specification of a conceptualization” [100]. In other words, an ontology explicitly specifies, or defines, a concept of interest. Within the same paper, Gruber provides an extension to his original definition: “a specification of a representational vocabulary for a shared domain of discourse" [100]. This definition extends the first, stating that a particular vocabulary should be the method by which the shared domain of interest is described.

The definitions above are abstract, restricting the concepts used to describe a domain without specifying the syntax of that description. Concrete definitions of ontologies are also common. Such definitions make use of the structural aspects of an ontology. Gruber's concrete definition of an ontology is "a set of representational primitives with which to model a domain of knowledge or discourse" [101]. In this definition, ontologies describe a topic, or domain, of interest in a formalised way using representational primitives. These primitives, or ontology entities, should include information about their meaning as well as any constraints on their use [101].

Bijan Parsia, one of the developers of the Pellet reasoner [102], states that a good first approximation of a concrete definition of an ontology is simply a set of axioms. Very generally, axioms are closed, well formed[22] statements declaring what is true about the domain of interest [103]. The set of axioms is the main component of an ontology, and is built using ontological entities such as classes, properties, and instances (see Figure 1.10). Parsia's definition is only a first approximation because an ontology can be defined not just as the set of its axioms, but be wholly characterised through the union of its

- name/identifier;
- set of annotations;
- set of imported ontologies; and
- set of axioms [103].

1.5.3.1 Philosophical versus computer science ontologies

In addition to the definitions described above, there are two categories of modelling strategies for building ontologies. Ontologists with a computer science background tend to perceive ontology development very differently from those with a philosophical background [104]. The role of an ontology to a philosopher is to accurately model reality, while computer scientists perceive an ontology's role as shared communication and computational readability. The contrasting points of view of computer scientists and philosophical ontologists produce two strategies for ontology development which can be distinguished based on the intent, or purpose, of the ontologist [104].

Philosophical ontologies are created with the intent of representing reality as closely as possible [100]. The description of the nature of reality hinges upon how universals are defined [105]. Differences

[22]http://www.nabble.com/Help-Required-td24715216.html

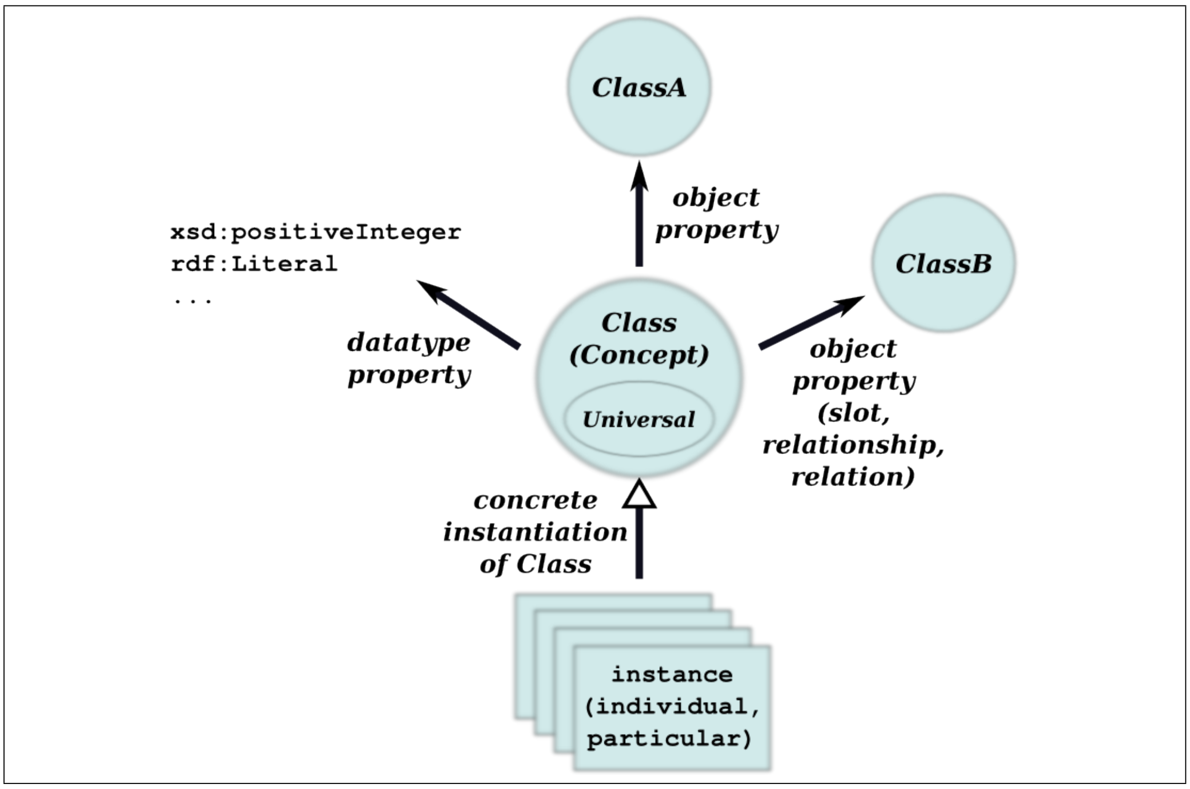

ClassA
object property
xsd:positiveInteger
rdf:Literal
...
ClassB
datatype property
Class (Concept)
Universal
object property (slot, relationship, relation)
concrete instantiation of Class
instance (individual, particular)

Figure 1.10: In each case where there is a single term followed by a set of terms in parentheses, those in parentheses are synonyms or near-synonyms of the first term. Further, the first term is the choice for this thesis. Casing conventions are indicated. The circles in the centre class show that while all universals are classes, not all classes are universals. Classes are abstract conceptualisations which describe common sets of properties. *Concept* is commonly used as a synonym for class. Concrete instantiations of these classes are called instances, and are constrained by the properties declared on that class. Commonly used synonyms for instances include *individual* and *particular*. Collectively, classes, properties, rules and instances are called *entities* in this thesis. The properties used to define membership in a class can define the relationship between two instances, or can link an instance to a data property such as a string or integer value. Synonyms and near-synonyms for property include *slot*, *axiom*, *relation* or *relationship*. See Section 1.5.6 for more information.

in the nature of universals result in three main strategies: realism, conceptualism and nominalism. Universals are the key to understanding the differences in these strategies.

Universals qualitatively identify and describe similarities among instances according to their common properties [106]. Universals are similar, but not identical, to ontology classes: all universals are generally modelled as classes, but all classes are not universals. Universals describe concepts and patterns which are repeatable in nature, such as black labradors. Other type of classes include arbitrary patterns that are useful to group together, but which are not repeatable in nature. One example of such an arbitrary, non-universal class is the union of (*Cheeseburger Monkey Planet*)[23]. While there may be some reason to represent the conjunction of these classes as a single class, they do not share properties other than being present in the set itself.

In realism, universals are considered both to exist, just as the particular instances which belong to a universal exist, and to be independent from any one person's beliefs or language [107]. Conceptualists believe universals exist only as abstract concepts, not as real entities [106]. Nominalism does not make use of universals at all, not even as abstract concepts. Instead, nominalists believe that only instances exist and that the problems surrounding the nature of universals can be resolved through careful thinking about instances alone [106].

The "mathematics" perspective of Rzhetsky and Evans is broadly synonymous with this type of ontology; here, computationally-accessible realist ontologies could ultimately merge into a single ontology for all of the life sciences [108]. The main deviation from the definition of philosophical ontologies presented in this section is their statement that this perspective is a view prevalent among computer scientists, which is the exact opposite of what the other articles referenced in this section describe for computer science ontologies.

In computer science, an ontology only becomes useful to scientists if it successfully models a domain of interest. In other words, the purpose of an ontology is to fulfil a role and accomplish a task. In a very general sense, ontologies in computer science provide a representation of common areas of interest which can be useful both for humans and computers [104]. While a computer science ontology may be viewed in the context of a philosophical ontology, such an alignment is not the driving force in its creation. Computer science ontologies are intended to explicitly describe the shared background knowledge of a community in a way that facilitates communication and, as a consequence, integration of that knowledge [94]. Rzhetsky and Evans describe an "Esperanto" perspective which closely aligns with this definition of computer science ontologies by presenting on the one hand a

[23]http://groups.google.com/group/information-ontology/browse_thread/thread/db9385cbc0f04700?pli=1

social, collaborative view of ontology development and on the other hand a practical acceptance that multiple ontologies will persist within the life science community and that integrative measures will be required to manage the situation [108].

In computer science strategies, data is defined as a collection of facts, and knowledge is defined as the union of data and an interpretation of the data's meaning [11]. As there can be many interpretations of a dataset, there can be many ontologies which validly describe that dataset. This is in contrast with the purpose of a philosophical ontology, which is to closely model existence. A philosophical ontologist might argue that there is only one true ontology to which a particular dataset or format can be applied, and that all others are not true interpretations of reality. As Gruber states, while philosophy is concerned with existence, those building computer science ontologies define that which exists as exactly that which is represented in their ontology [100]. Computer science ontologies focus on communicating a shared understanding of the domain of interest in a way accessible both to computers and humans; how closely reality is modelled is irrelevant in such a strategy.

In order for a computer science ontology to fulfil its role, certain structural constraints are often imposed. For instance, while Noy and McGuinness agree with Gruber that an ontology defines a shared domain of interest, they also require that ontologies be interpretable by machines [95]. Barkmeyer and Obrst go one step further, stating that written knowledge cannot be an ontology unless it is both understandable to computers and suitable for automated reasoning [109]. Barkmeyer and Obrst believe that an ontology without access to automated reasoning is incomplete and not fit for purpose. This requirement of a format being suitable for automated reasoning is at odds with many philosophical definitions of ontologies.

Rzhetsky and Evans add a third type of ontology to the two already mentioned; their "CS"[24] perspective, which describes a one-ontology-per-tool method of ontology development intended to be in direct competition with the mathematics perspective [108]. While on the surface this perspective shares some aspects with the already-described computer science ontologies, particularly the fit-for-purpose practicality of ontology development, in fact this perspective requires that each ontology is made and kept in isolation, with no attempt at reconciliation. The definition of computer science ontologies used in this section has as its primary requirement the facilitation of communication of a domain of interest both for humans and computers, which is why it most closely matches the Esperanto perspective.

[24]This ontological perspective is called "computer science" in the paper by Rzhetsky and Evans, however to avoid confusion with the definition of a computer science ontology used earlier in this section, the acronym *CS* is used.

1.5.4 Ontological formalisms, languages and formats

There are three initial considerations to take into account before construction of an ontology can begin: the *formalism*, or logical framework available to software systems; the knowledge representation *language*, which is a human-readable interpretation of a formalism; and finally, the *syntax* used for storing the ontology, or its format. Ontology languages may have multiple formal interpretations, or formalisms. However, there needs to be a tight link between a language and its formalism, as a human-readable definition in a language needs to be accurately represented in its formal definition[25]. The choices of ontology language, and even of the format, are independent from the choice of expressivity level and therefore from the choice of formalism. Ontology languages and formalisms are abstract concepts and, as such, require formats with which to create, share, and analyse the ontology itself. A single ontology language may also be implemented in more than one format.

The two most commonly used ontology languages in the life sciences community are OWL [103] and OBO [81], and this section describes the languages themselves as well as their formalisms and formats.

1.5.4.1 OWL

Description Logics Description Logics (DLs) are formalisms for knowledge representation characterised by various levels of expressivity. Knowledge-based systems created through the use of DLs are capable of finding implicit consequences of explicitly represented knowledge [110, pg. 2]. Historically, DLs have been used in many domains: software engineering, configuration, medicine, digital libraries, Web-based information systems (e.g. the Semantic Web), data mining, natural language processing and data integration [110, pg. 23-29]. The more expressive a DL is, the less tractable it is for reasoning purposes [110, pg 9]. Therefore, a language must be chosen that has an appropriate ratio of expressivity to tractability. DLs are widely used in the biomedical community via the OWL language, where Web Ontology Language constrained by Description Logics (OWL-DL) is one of a number of OWL 2 profiles [111]. The DL formalisms accessible through OWL have the ability to represent complex logic constructs and constraints such as number restrictions and property hierarchies [110, pg 8]. Editors such as Protégé [26] and libraries such as the OWL API [112] can determine in which profile a particular ontology is written.

[25] `http://sourceforge.net/mailarchive/forum.php?thread_name=F3AB4896-CE9A-44FD-9623-CFF0A30F8F8F%40gmail.com&forum_name=obi-devel`

[26] `http://protege.stanford.edu`

The OWL Language Semantic Web technologies such as RDF and OWL have been espoused as a unified framework for integrating and retrieving life science data [113, 114, 115], and are core components of the Semantic Web. OWL improves upon RDF by providing the ability to explicitly describe objects and make assertions about them [115]. Constraints on class membership such as disjointedness (e.g. stating that *BlackLabrador* cannot also be a *Person*) cannot be expressed in RDF [26]. Decidability and complexity go hand-in-hand; the more complex the logic is, the more challenging the reasoning [110, pg 44]. Therefore, a variety of tractable subsets of OWL, called profiles, have been developed. Each profile has optimisations for particular reasoning tasks. OWL-EL is a profile optimised for large numbers of properties and classes, while OWL-QL is aimed primarily at ontologies with large numbers of instances, where querying is the main concern [111]. Finally, OWL-RL is a profile which combines expressive power with scalable reasoning times and which is suitable for use with rule languages and rule-based reasoning [111]. However, OWL profiles are not limited to these three; for instance, OWL-DL and OWL-Lite are both valid OWL profiles.

OWL formats OWL can be expressed in a number of formats, including the Manchester OWL Syntax [116] and RDF/XML. The Manchester OWL Syntax is highly understandable to a human, while the triples of an RDF-based format are easily handled by standard RDF libraries available to many programmers. The OWL RDF/XML format is created by layering OWL on top of RDF, which can then be serialised into XML [113]. Presenting biological data in RDF-based formats allows unambiguous naming of entities via URIs, simple addition of data via graph structures, the use of the open world assumption and the addition of new data without invalidating or changing existing data [113].

1.5.4.2 OBO

OBO Formalism and Language The OBO Foundry [81], a consortium of ontology developers, does not explicitly state which formalism is required of its ontologies. However, as the Foundry recommends usage of their upper-level ontologies by all of its domain ontologies, these upper-level ontologies provide direction for Foundry domain ontologies. BFO, an OBO upper-level ontology for scientific research, was developed with interpretations in OWL, first-order logic and the native OBO. The Relations Ontology (RO) [117], an upper-level ontology concerned with high-level biology-specific relationships, is native to OBO but also automatically converted to OWL.

While commonly used in the life sciences, the OBO language is not well suited to semantic reasoning, inference and querying. Indeed, when Golbriech and colleagues created a mapping from OBO to OWL, thus making reasoning and strict semantics available to OBO ontologies, modelling errors that had previously gone unnoticed were discovered [118]. While DL languages such as OWL were developed to unambiguously describe ontological concepts, OBO contains ambiguous and informal descriptions of its concepts, and as such its reasoner can miss important inferences [118]. In contrast to the formal approach provided by OWL where the properties link instances of classes and therefore must be quantified, OBO is a terminology-based language whose properties link classes directly [119]. As such, within OBO existential qualifications such as *some*, *exactly* or *only* are not possible [119]. Additionally, OWL has the ability to represent more complex logic constructs, such as number restrictions and property hierarchies, and OWL reasoners are more powerful than their OBO counterparts [118]. Updates to bring OBO closer to OWL and provide lossless conversion from OBO to OWL address many of these limitations [120].

The OBO Format The OBO language currently only has a single format and therefore OBO can be considered both a language and a format. Like the Manchester OWL Syntax, OBO is highly readable to humans and is composed of stanzas describing entities and their locations within the hierarchy. Figure 1.11 shows a GO term and its stanza.

```
[Term]
id: GO:0001555
name: oocyte growth
is_a: GO:0016049 ! cell growth
relationship: part_of GO:0048601 ! oocyte morphogenesis
intersection_of: GO:0040007 ! growth
intersection_of: has_central_participant CL:0000023 ! oocyte
```

Figure 1.11: The GO term `oocyte growth` and its stanza. This figure illustrates the native OBO format.

The OBO Foundry has created a set of naming conventions and other principles that, while not part of the format, are important restrictions on how the ontologies are constructed[27]. If utilised by the OBO Foundry ontologies, Foundry principles such as naming conventions aid social collaboration as well as the semantic alignment of multiple ontologies [121].

1.5.5 Structural components of an ontology

An ontology can be broadly defined by two types of components: entities and descriptors [122]. Entities are the main body of the ontology and include classes, properties, instances and rules. Figure 1.10

[27]http://www.obofoundry.org/crit.shtml

shows each of these entities and how they are related. Each class is an abstract concept representing a group of instances [122]. In DL, only one definition for each class is allowed, and that definition must not be cyclic [110, pg. 13]. Properties describe the links between instances [110, pg. 46], and can be expressed between classes and applying to all instances of that class [122]. Rules can be written in OWL or OWL-based rule languages such as Semantic Web Rule Language (SWRL) [123], which provides the capability to write *if-then* statements about an ontology. Descriptors are vital for human understanding of an ontology, and include documentation (such as English definitions of entities) and annotation (such as notes and other metadata) [122]. Imports of other ontologies are not entities, but neither are they strictly descriptors. Import statements are vital for completeness of the ontology and for its successful reasoning, as they provide a method of including additional ontological resources.

The components of an ontology are commonly grouped into the *TBox* and *ABox*; classes reside in the TBox and instances in the ABox. Sometimes a third grouping, the *RBox*, is defined to specify the location of the rules within an ontology. The TBox is the component of an ontology describing intensional knowledge, or the properties of an entity required to identify it, and is considered to be unchanging knowledge [110, pg. 12]. The ABox is similar to extensional knowledge, and describes knowledge specific to the domain of interest, or to a particular problem [110, pg. 13]. ABox knowledge is considered to be dependent upon a set of circumstances, and therefore changeable [110, pg. 13]. Some ontologies contain a set of rule axioms; such a set is called the RBox [124]. More information about rules in ontologies is available in Section 5.1.4.

There are a number of commonly used metrics available for summarising the components of an ontology (see Table 4.1 in Chapter 4 for examples). Class metrics include a count of the classes themselves as well as various metrics describing their axioms. If an instance fulfils the *necessary and sufficient* conditions of a class, then a reasoner will infer that instance to be a member of that class. These conditions are also known as *equivalent class axioms*. Similarly, *necessary* conditions (also called *subclass axioms*) must hold true if an instance is asserted to be a member of the class, but such axioms are not sufficient to infer the placement of an instance under that class. If an instance is asserted to be a member of two *disjoint* classes, then the reasoner is able to deduce an inconsistency [125].

There are also a number of useful property-level metrics. Total numbers of object and data properties give a general idea of the possible connections between classes. *Functional* properties are those which have either zero or one value. In other words, no individual in the domain of a functional property may have more than one value in the range of that property.

1.5.6 Conventions

Due to the large number of nearly synonymous terms for the components of an ontology where multiple equivalent terms exist, one has been used throughout. Figure 1.10 graphically illustrates the conventions described here, and includes synonymous and near-synonymous terminology and how each component interrelates. Figure 1.12 provides a simple example of a partial ontology which matches the structure described in Figure 1.10. The design decisions made with respect to the available range of terminology are listed below.

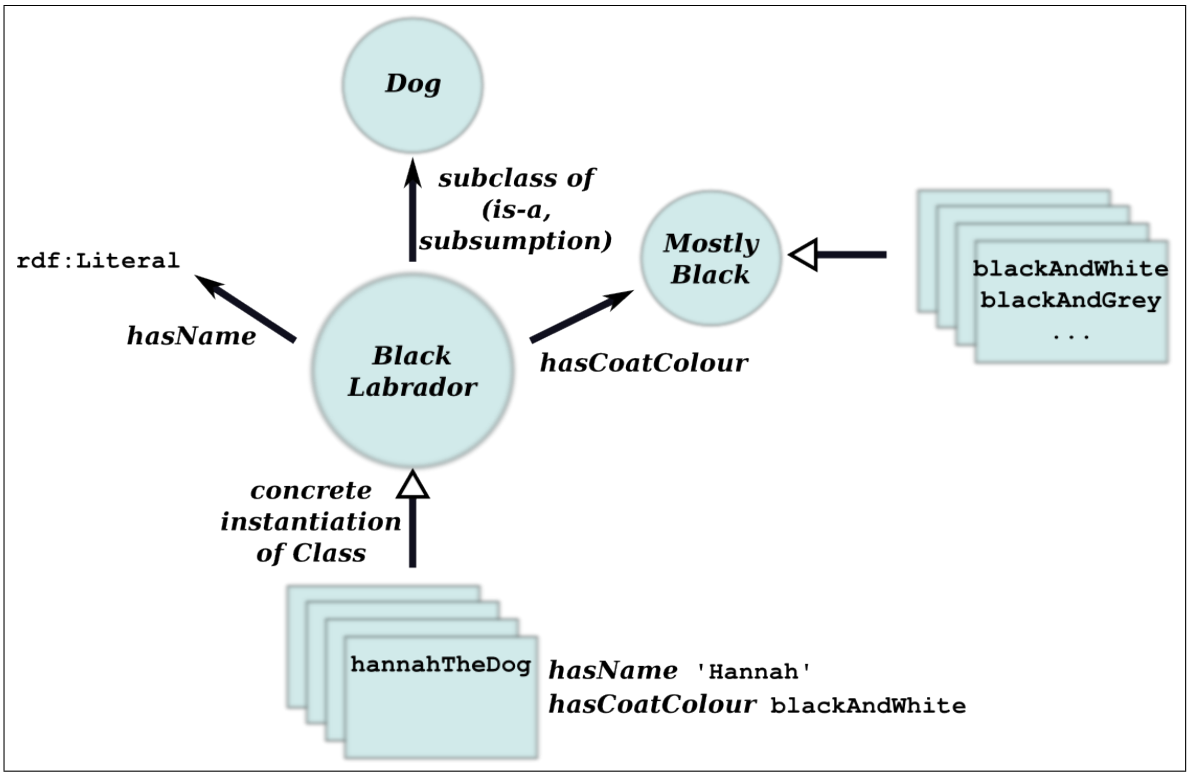


Figure 1.12: Using the structure described in Figure 1.10, this concrete example of a portion of an ontology shows how classes, properties and instances are used together to generate a model of a domain of interest. Here, the domain of interest is black labradors. See Section 1.5.6 for more information.

- *Class* and *concept* are often used interchangeably as a name for the entity which describes and constrains a group of instances. Because *concept* is often used as a more general term similar to *entity*, *class* is used to describe this ontology component within this thesis. A third term, *universal*, is similar but not equivalent to class. While all universals should be written as classes when creating an ontology, all classes are not universals (see Section 1.5.3.1).
- *Instance*, *individual* and *particular* are all used to refer to the members of a class which are specific instantiations of the class type. *Instance* is the term used throughout this thesis.
- *Property*, *slot*, *relation* and *relationship* are all used to describe the links between two instances

as well as links from an instance to a data type (e.g. a plain string or integer). *Property* has been chosen for this thesis.

- *Entity* is used in this thesis as a general term to define any object (e.g. class, property, instance, rules) within an ontology.
- The definitions of *computer science ontologies* and *philosophical ontologies* match the definitions provided by Stevens and colleagues [104].

The following typographical conventions were used to distinguish specific object types from standard text:

- *ClassName*: the name of a class is shown in an *italicised font*. Further, classes created in this research begin with a capital letter, although some third-party ontologies do not follow this convention.
- *propertyName*: the name of a property is shown in an *italicised font* and begins with a lower-case letter unless otherwise specified. In some third-party ontologies, underscores are used to separate words in the property name, while in this thesis CamelCase is preferred.
- `instanceName`: instances are shown in a `fixed-width font` and begin with a lower-case letter unless otherwise specified.
- XML element and attribute names are used when describing some XML-based data sources. Such names are shown in a sans serif font.

1.5.7 Ontological design decisions

The Semantic Web allows computers to go beyond performing numerical computations and provides a common structure for easily sharing, reusing and integrating information[28]. OWL, a Semantic Web standard for representing knowledge, enjoys strong tool support and is often used for capturing biological and medical knowledge. OWL ontologies in the life sciences include, amongst others, OBI, BioPax [35], EXPO [126], FMA [127] and GALEN [128]. Once the information about the domain has been modelled in OWL, a software application called a reasoner (such as Pellet [102], FaCT++ [129] or HerMIT [130]) can automatically infer all other facts that must logically follow as well as find inconsistencies between asserted facts. OWL reasoners perform four main tasks:

- *Consistency checking.* These checks ensure that all of the logical statements asserted in an ontology do not contradict each other. Inconsistent ontologies have no models which do *not* contradict the statements made in the ontology, and therefore no meaningful conclusions can

[28]`http://www.w3.org/2001/sw/`

be made [131].

- *Class satisfiability.* Satisfiability is whether or not a class can have any instances without making that ontology inconsistent. If a class is unsatisfiable, it is equivalent to the empty set, or *owl:Nothing*, and therefore no useful analysis can be done with that class. Such an error is generally due to a fundamental modelling error [132].
- *Classification.* During classification, the placement of all classes are checked and rearrangements to the hierarchy (based on the asserted logic) are made, creating an inferred hierarchy. Such inferred hierarchies are useful, especially when an ontology is normalised such that each class has only one asserted superclass, or parent [133]. Once classification has occurred, a class may be inferred to be a member of more than one parent class. Normalisation has a number of benefits, including aiding the creation of modular ontologies [134].
- *Realisation.* Realisation is performed on instances in an ontology. When realizing instances, all instances are placed under their most specific defining class or classes, if they belong in multiple locations[29].

Due to the more formal nature of the language, the use of existential quantification and the reasoning benefits listed above, OWL was chosen for this research. Specifically, OWL-DL was chosen because the DL profile ensures that ontologies are decidable and can be reasoned upon. While research domains are successfully modelled in OBO, this higher level of expressivity together with the logical inferences available when reasoning over a DL-based ontology make a DL format such as OWL the best choice. In the thesis, OWL is used as a shorthand for OWL 2 DL. With DL, the implicit knowledge that is present within an ontology—and which is not immediately obvious to a human—can be made explicit through inference and reasoning [110, p. 61]. Irrespective of the level of expressivity of an OWL ontology, OWL is more useful than other ontology languages such as OBO when reduction of the ambiguity of interpretation is paramount [135]. While any of the OWL syntaxes could be used, to ensure compatibility with the broadest range of third-party software, the RDF/XML format was chosen, although examples presented in this thesis make use of the more human readable Manchester OWL syntax [136].

Although the various strategies such as realism and nominalism are hotly debated in the philosophical ontology community, to a computer scientist, such differences are, to all intents and purposes, irrelevant. Irrespective of whether or not a *BlackLabrador* is a universal which exists in reality, it is a concept which a researcher may be interested in studying. Therefore, from a computer science perspective, it does not matter if *BlackLabrador* truly exists: it is enough that it is a concept which needs

[29]http://clarkparsia.com/pellet/features

to be modelled. Further, the outcome of using *BlackLabrador* in an ontology will be the same, irrespective of whether it was added from a realist or nominalist point of view. Ultimately, philosophical strategies require that a concept must exist in reality for it to be modelled. Robert Stevens describes scientific concepts such as the Higgs boson and Newtonian mechanics as "unicorns" because they are imaginary, conjecture or simply convenience entities [63]. Even though these entities are not "real" according to a philosophical modelling strategy, they are relevant to science and need to be modelled. Because these unicorns need to be modelled, and because computer science ontologies are based on a shared understanding of the domain as well as computational accessibility, a computing science modelling strategy was chosen for the work described in this thesis.

1.6 Data integration methodologies for systems biology

The amount of data available to systems biologists is constantly increasing. However, its integration remains an ongoing challenge due to the heterogeneous nature of biological information, the multitude of data sources and the variability of data formats both in syntax and semantics [137]. Very broadly, integration methodologies resolve this heterogeneity by loading multiple data sources into an integration interface and then providing access to that interface. Integrated data sources, whether distributed or centralised, allow querying of multiple data sources in a single search.

This section provides a review of those data integration structures and methodologies relevant to systems biology and to the work described in this thesis. Section 1.6.1 describes the three main obstacles facing researchers attempting to integrate systems biology data. Section 1.6.2 describes the difference between structuring the integration methodology syntactically and semantically. The methods of integration described in this section can be classified along two axes: firstly, the mapping type (Section 1.6.3) determines how the integration interface is connected to the underlying data sources; and secondly, the integration interface (Section 1.6.4) determines how the data is retrieved and structured for the end application or user. The implications for the work presented in this thesis and the design decisions made with regard to integration structure and methodology are described in Section 5.1.

There are a number of existing reviews of data integration methodologies in the life sciences as a whole. Joyce and Palsson [138] review the algorithms for the integration and generation of networks from omics data, and the benefit of combining different pairs of such datasets. Stein [139] and Sujansky [140] present a more global view of data integration in biology and biomedicine, respectively, without touching on the specific difficulties for systems biology. Philippi and Koehler [17] discuss the practical, political and social hurdles to integrating high throughput and primary databases in systems biology, but do not evaluate specific tools. Harland and colleagues provide an up-to-date summary of the challenges facing both industry and academia in data integration, and how the use of controlled vocabularies can address these challenges [137].

1.6.1 Data integration challenges in the life sciences

Experimental data is noisy, incomplete, of different granularities, changeable, context dependent, does not always have a long lifetime and is not necessarily freely available. If data integration is to surmount these hurdles and be successful in the long term, three practical, social and political obstacles must be overcome.

- *The **format** challenge.* Representational heterogeneity occurs when the underlying databases have completely different methods of representing data [140]. Resolving representational heterogeneity through a unified representation is difficult, with misunderstandings possible if matching terms are not semantically equal [17]. Even if the underlying databases contain semantically identical knowledge, there is no guarantee that they share a common representation of that knowledge [140]. Further, the lack of attribution of annotation must be resolved [17].

- *The **availability** challenge.* The availability and lifetime of the data must be considered, and Web access should be provided for all data sources. Partial downloads of databases should be available to allow manageable offline use of data [17].

- *The **social** challenge.* Social and political factors can often be central to the success of a data source. Integration can be hampered by a lack of communication and understanding among database providers and biologists, and by restrictive licensing methods [17]. Local autonomy of data resources may result in a lack of knowledge about any integration efforts as well as complete independence from those efforts [140].

Although a detailed discussion of the availability and social challenges are beyond the scope of this thesis, mapping types such as hybrid mediator mapping (see also Section 1.6.3.3) enforcing the use of a common schema or ontology across multiple research groups minimise format and social challenges and allow researchers to concentrate on availability issues. In such cases, format-based obstacles are minimised as all data sources are part of a collaborative project. However, a common structure for data does not in any way ensure that the associated data repositories survive past the lifetime of their grant or project. In many cases, prospective users often find only broken links or an outdated database [141, 142, 143]. These problems are exemplified by the former Integrated Genome Database which, as Stein reported, collapsed due to database churn after about one year [139].

Most biological databases are neither based on identical schemas nor refer to a common ontology; it would be impractical for a nucleotide sequence database, for example, to have the same data model as a database that stores mass spectrometry results. Hybrid mediator mapping is therefore not always possible, especially when resolution of social challenges via a shared schema across research groups is simply not available. In such cases, format challenges must be resolved through the integration of heterogeneous databases. In systems biology, heterogeneity needs to be addressed not only for practical management purposes, but also to build a more systems-level view of the organisms under study [144]. The use of community-endorsed data standards for guiding the content, syntax and semantics of life-science data (see Section 1.4) provides partial solutions to format and social obstacles. However, there will always be new data formats and new experiment types, preventing format challenges from being fully resolvable through common data standards alone. Further, development of a common structure for related data at an early stage is often hampered by political and social factors.

When common data standards have been used as much as possible, and a hybrid mediator mapping approach is not feasible, other methods for integrating heterogeneous data sources must be chosen. The work described in this thesis directly addresses the format challenge in a number of ways. An archive for storing experimental metadata in a common format has been developed and a Web application integrating data from disparate sources for the purposes of systems biology model annotation has been created. Additionally, and of greatest relevance to this section, a semantic integration methodology has been developed to resolve representational heterogeneity. The end result is a unified ontology-based view of multiple data sources which can be used to add knowledge to systems biology models.

1.6.2 Structuring integrated data

Traditional methods of data integration map multiple schemas to an integration interface based on the common structures, such as table names or fields in database entries, within each format. Such methods tend to resolve syntactic heterogeneity but do not address semantic heterogeneity. The problems of, and historical approaches to, syntactic and semantic data integration have been well described [140, 145]. Whereas traditional methods of data integration rely on the mapping of component data models to a single data model, semantic data integration incorporates computationally-accessible meanings, or definitions of the data types to be integrated. In general, the integration structures used in syntactic integration are schemas, while those used in semantic integration are ontologies.

Syntactic heterogeneity is when data of interest is available in different formats such as in SBML or BioPAX, or XML as well as flat-file format. Data in multiple formats can be aligned to a global schema by linking structural units such as XML Schema Definition (XSD) components or table and row names. The mapping of syntaxes to a global schema in syntactic integration approaches tends to be hard-coded for the task at hand, and therefore not easily reusable in other projects. Further, concepts between the source and global schema are often linked based on syntactic similarity, which does not necessarily account for differences in the meanings of those concepts. For instance, a protein in BioPAX is strictly defined as having only one polypeptide chain, while a protein in UniProtKB can consist of multiple chains. Such semantic inequalities in syntactically identical terms (in this example, "protein") can result in errors in data integration, creating possible inconsistencies in how concepts are defined among those data sources [17].

A high level of semantic heterogeneity makes mapping difficult; to be successful, extra information about the entities of interest might be required. Semantic data integration resolves the syntactic heterogeneity present in multiple data models as well as the semantic heterogeneity among similar concepts across those data models. Semantic integration can make use of a richer description of biology than is possible with syntactic methods, and makes that description accessible to machines and humans.

The example of differing protein definitions described earlier in this section for BioPAX and UniProtKB can be further extended to illustrate the practical differences between syntactic and semantic integration. In traditional data integration methods, two database schemas may contain a "Protein" table, but if the developers' definitions of "Protein" are not identical, it is difficult to determine this difference programmatically. A syntactic integration project using these two schemas as data sources may erroneously mark them as equivalent tables. In semantic integration, if the two data sources model their *Protein* classes correctly, the differences in their meaning would be clear both programmatically and to a human. Identification of semantic differences is the first step in their resolution. For instance, one possible solution would be the creation of a *Protein* superclass that describes a protein in a high-level, generic way. The two source definitions could then be modelled as children of that *Protein* superclass.

Often, ontologies or other Semantic Web tools such as RDF [34] are used both to provide a richer model of biological data and to perform the integration. The Semantic Web is a network of data which acts as a framework for sharing, reuse and integration of that data. It fulfils its purpose via common formats for integration and via a common language for relating that data to real-life objects[30]. Ruttenberg and colleagues see the Semantic Web, of which both OWL and RDF are components, as having the potential to aid translational and systems biology research; any life science field where there are large amounts of data in distributed, disparate formats should benefit from Semantic Web technologies [146]. While RDF is a Semantic Web technology, making use of this format is not enough on its own to resolve semantic heterogeneity. Often OWL ontologies, either on their own or in conjunction with RDF, are used to address semantic differences in data.

Ontologies can be more generic, reusable and independent of the integrative applications they were created for when compared with traditional approaches [115]. Ontologies are enhanced logic-based

[30]http://www.w3.org/2001/sw/

formalisms where classes can be logically compared for equivalence and where classes and other entities can be integrated across domains [147]. Mappings between schemas in non-semantic approaches are specific to those schemas, and cannot be applied to other data sources; conversely, mappings between ontologies—and therefore data sources utilising those ontologies—can be used by any resource making use of those ontologies, and not just the original, intended, data sources [148]. Section 1.5 provides a detailed description of ontologies.

1.6.3 Mapping source data

There are two axes, the *mapping type* used and the *integration interface*, along which the four main integration categories can be classified (see Table 1.2). These four categories are data encapsulation, query encapsulation, data translation and query translation. In this section, the mapping types for linking native data to an integrated end result are described. Section 1.6.4 then relates these mapping types with the two most common integration interfaces for source data: warehouses and federation.

			Integration Interface	
			Stable ***Quick access***	***Lightweight*** ***Up-to-date***
			warehouse	*federation*
Mapping types	***Robust***	*multiple mediator*	data encapsulation	query encapsulation†
	Unified	*hybrid + single mediator*	data translation*	query translation

Table 1.2: Summary of integration methodologies and their main characteristics according to mapping type (rows) and integration interface (columns). Information linkage and direct mapping, the two mapping types which do not make use of an integration interface, are not included in this table as the lack of an integration interface makes them neither a warehouse nor a federated resource. (†) Query encapsulation is not a commonly used method, but is included for completeness. *As hybrid mediator mapping guarantees that the local data sources are aware of the global schema, it has no valid warehousing interface, as warehoused hybrid mapping would be equivalent to data translation.

The mapping methods described here were originally defined by Alonzo-Calvo and colleagues [145] for federated databases only. In the work described in this thesis, their definitions have been extended and corresponding changes to the names of the mapping types made. For example, Alonzo-Calvo and colleagues originally used a naming scheme containing the phrase "conceptual schema" (e.g. *multiple conceptual schema*). Here, "conceptual" has been removed, as it relates only to a federated method's conceptual data model as opposed to a warehouse's applied data model. Further, as "schema" is often shorthand for "database schema", "mediator" is used instead to incorporate both syntactic and semantic integration methods. In syntactic integration, the data sources are generally described by database schemas or other schemas which define the format of the data. Conversely, semantic integration uses ontologies or other similar semantic structures to model the data.

Figures 1.13-1.14 describe simple mapping methods which do not require an explicit integration interface. As such, these simple types are fully explained in this section and are not included in Section 1.6.4. The “integration interface” named in Figures 1.15–1.17 refers to either a warehouse or a federated solution, as described in Section 1.6.4.

1.6.3.1 Information linkage

The simplest method of linking data sources to a user is information linkage [145], where the data is presented without an integration step. This method is also known as link integration [139, 149], navigational integration [150] and portal [151] integration. Shown in Figure 1.13, information linkage provides the data directly to the user and is therefore more a measure of interconnectedness than of integration. This solution is lightweight for the program developer, as it only involves linking to related information. However, it puts a greater burden on the user, who must visit all links to determine which resources are useful and then manually pull the relevant information from those resources. It also forces the user to click through into a site of unknown quality [139].

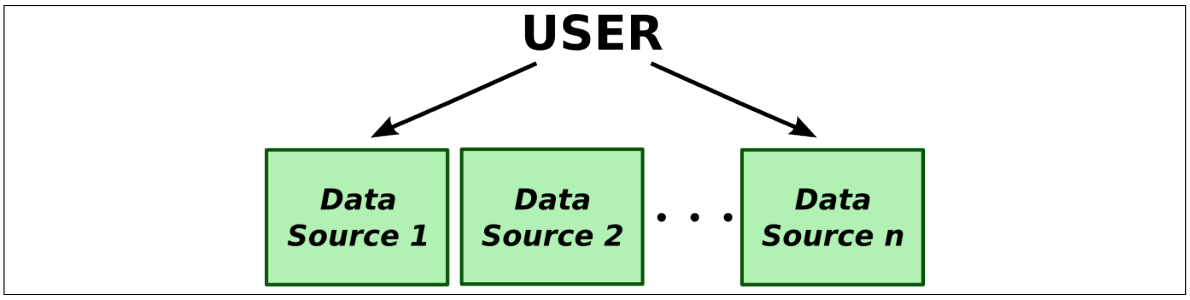


Figure 1.13: A visualisation of information linkage. Here, data is provided directly to the user, usually in the form of hyperlinks. There are no connections among data sources, only links from the end user’s interface (such as a Web page) to each data source. As such, information linkage is not strictly a mapping method, but rather a method of connecting the user to resources.

Links out Links out are information linkage in its simplest form. This method connects disparate datasets and analytical tools via hyperlinks or other explicit links, allowing the user to easily navigate from one data entry to another in a different dataset. Most Web-based database viewers which provide cross references fall into this category.

Integrated analysis environments Integrated analysis environments are specialised cases of information linkage. While having the same capabilities as links out, an integrated analysis environment also locally stores user data for analysis. The main goal of an integrated analysis environment is to provide an environment specifically for analysing user-provided raw data such as high throughput omics datasets. One example of this mapping type is Gaggle, which provides access to a variety of analysis tools and databases using external data sources as well as locally stored copies of user datasets [152]. Gaggle's main purpose is as a single environment for accessing Web-based resources and programs. Its controller (the Gaggle "boss") directly accesses remote or local sources, each via their own interface, or "goose" [152]. VANTED is similar to Gaggle in that it accepts various types of expression data in Microsoft Excel format then allows the mapping of this data onto either user-defined or publicly-accessible networks [153]. It also provides various analytical tools and network-based viewers. However, the environment is centralised into a single application, rather than the boss-geese application set of Gaggle.

1.6.3.2 Direct mapping

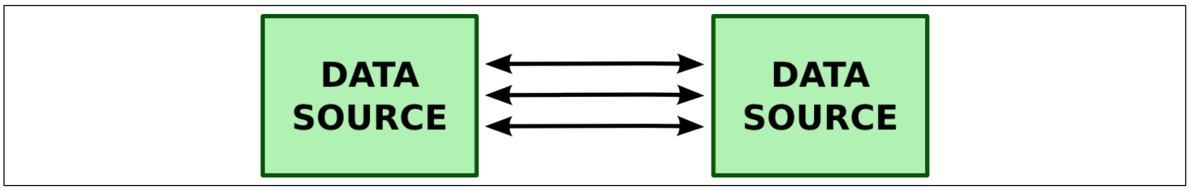


Figure 1.14: *Direct mapping* between two schemas or ontologies. Though not often used in syntactic integration, schema mapping would directly link specific columns in one database with appropriate columns in another. Within semantic integration, ontology alignment and ontology mapping allow the mapping of a source class from one ontology to a target class in another ontology.

To present a more integrated view of data to the user than is available with information linkage, some level of mapping between resources must occur. The simplest type of mapping is *direct mapping*. As shown in Figure 1.14, this method maps one data source directly onto a second data source. Though possible in syntactic integration, direct mapping is more commonly used for performing ontology mapping or alignment in semantic integration. Ontology mapping and alignment creates associations between classes in two or more ontologies [154]. While in the literature alignment and mapping are often used interchangeably, a clear separation between the terms is made in this thesis. Ontology mapping is the term used when the existing ontologies cannot or should not be changed and does not necessarily result in a true merging of the original ontologies, while ontology alignment is used when the integrated result must be a single ontology created from merging source ontologies together.

Direct mapping via ontology alignment was performed by Lomax and colleagues between GO and UMLS [83]. These two mature ontologies were linked such that data annotated with one ontology could also be associated with the second. Ontology mapping rather than alignment was performed when Entrez Gene/HomoloGene was integrated with pathway resources [155]. Here, the Entrez Knowledge Model (EKoM) OWL ontology of gene resources was directly mapped to BioPAX [155]. To integrate the two ontologies, BioPAX was imported into EKoM and relationships were added to link the two ontologies together. These ontologies were then populated directly from the data sources, and SPARQL Protocol and RDF Query Language (SPARQL) was used to query over the integrated knowledge base [155]. Direct mapping is not easily extensible, nor is it easy to update the integrated ontology when either source ontology changes.

Other mapping approaches make use of a bridging ontology between two resources. Often, the resources linked via a bridging ontology are not required to be ontologies themselves. One example is the alignment of the UMLS semantic network with the upper-level ontology BioTop by Schulz and colleagues [156]. Schulz and colleagues created a bridging ontology which imports both the UMLS semantic network and BioTop in order to improve the quality and richness of BioTop. While the UMLS semantic network is a hierarchical set of classes and relationships, it is not a formal ontology but rather an information legacy system [156]. SBPAX is another bridging ontology between the SBML format and the BioPAX ontology. SBPAX was created to integrate the information available in both formats to aid systems biology model integration [41]. In this approach, data can be either converted from one format to another, or merged together and then saved in a target format. Only concepts common to both formats are modelled within SBPAX [41]. While SBPAX provides a useful approach for conversion and merging of model and pathway data, it is not a generic solution. Further, later releases of SBPAX (discussed in Section 5.1) are moving away from integration and towards the creation of an extension to the BioPAX ontology which includes quantitative information.

1.6.3.3 Mapping types

The two axes of classification for integration methods defined earlier are the mapping type and the integration interface. In this section, the various mapping types are described. The single common feature of all of these mapping types is the presences of a mediator. Mediators can be created in different ways and using different technologies, and when combined with a data storage solution become the integration interface for a particular methodology. Integration interfaces are further described in the next section, 1.6.4.

Rather than trying to map three or more data sources directly to each other, mediator-based approaches individually map data sources to one or more mediator schemas or ontologies. The mediator-based mapping types are represented in the rows of Table 1.2. Each of the mappings illustrated in Figures 1.15–1.17 uses integration interfaces to mediate between the user and the source data. The first of these types, multiple mediator mapping, has an integration interface for each data source or group of similar data sources. This makes it robust with respect to changes in the data source, but produces a complex integration interface. Single mediator mapping and hybrid mediator mapping utilise a single integration interface, which creates a stable, unified global schema but which requires changes to that schema when underlying data sources are modified. These two single interface mapping types have three further subtypes. First developed for database schemas, global-as-view, local-as-view and hybrid strategies are also applicable for ontology-based integration techniques where more than two ontologies are required.

Global-as-view mapping defines a mediator ontology or schema as a function of the data sources rather than as a semantically rich description of the research domain in its own right, though the level of dependence of the mediator ontology can vary [157, 158, 159]. With local-as-view mapping the mediator is independent of the sources and the sources themselves are described as views of the mediator. Local-as-view mapping can be used to automatically generate queries over the data source ontologies [158].

In addition to pure global-as-view and local-as-view subtypes hybrid approaches, such as that used in this thesis [160, 161], are also available. Though these approaches generate mappings between sources and the mediator, unlike traditional approaches, the mediator is completely independent of any of the sources. Such approaches allow both the straightforward addition of new sources as well as the maintenance of the mediator as an independent entity. More information on ontology-based integration and these mapping subtypes is available in Section 5.1.

Multiple mediator mapping With multiple mediator mapping, a schema or ontology for each data source or grouping of similar data sources is created, as shown in Figure 1.15. While any schema, controlled vocabulary or ontology may be used as the integration interface, there is no guarantee that the underlying data sources will have any knowledge of that interface [145]/. This type of mapping allows new data sources to be added without disruption to existing resources or existing code. However, the complexity of the integrative interface is much greater, as many schemas or ontologies need to be queried.

Single mediator mapping The complexity problems of multiple mediators are solved in single mediator mapping by modelling all incoming data sources using a unified global schema, as shown in Figure 1.16. This is the second of the mediator-based approaches and may be realised via a number of routes, including schema matching and ontology mapping [145]. This method presents a unified

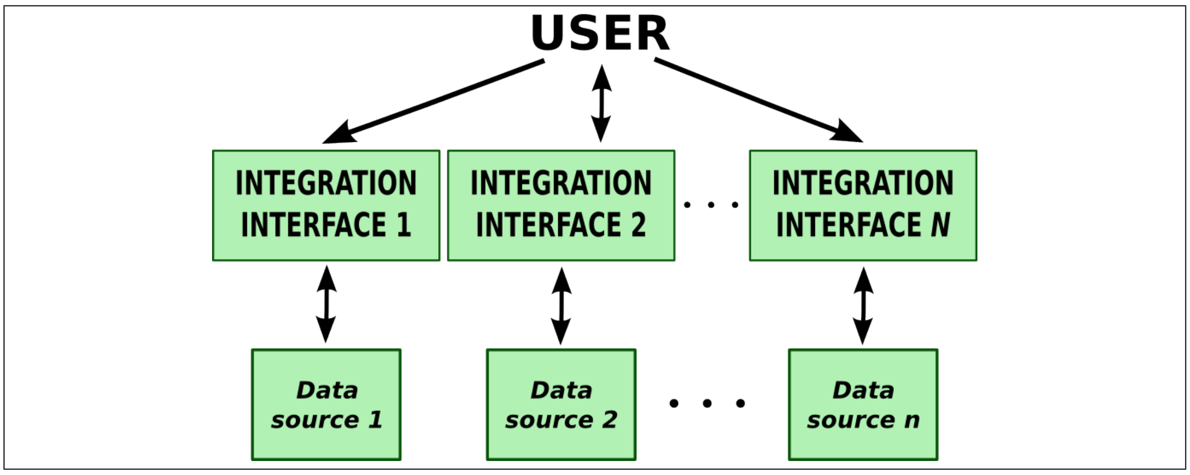


Figure 1.15: Multiple mediator mapping for data integration. Each data format is assigned its own mediator within the integration interface.

view of the data to the user and does not force the underlying data sources to subscribe to that unified view.

Global-as-view and hybrid ontology mapping subtypes are possible with single mediator mapping as they do not require the data sources to have any knowledge of the mediator ontology. Global-as-view implementations must change their mediator ontology with any change to any underlying data source, which may affect the entire integration method. The local-as-view mapping subtype is not available for single mediator mapping, as data source schemas or ontologies are a view of the global ontology and therefore must have knowledge of the global mediator ontology.

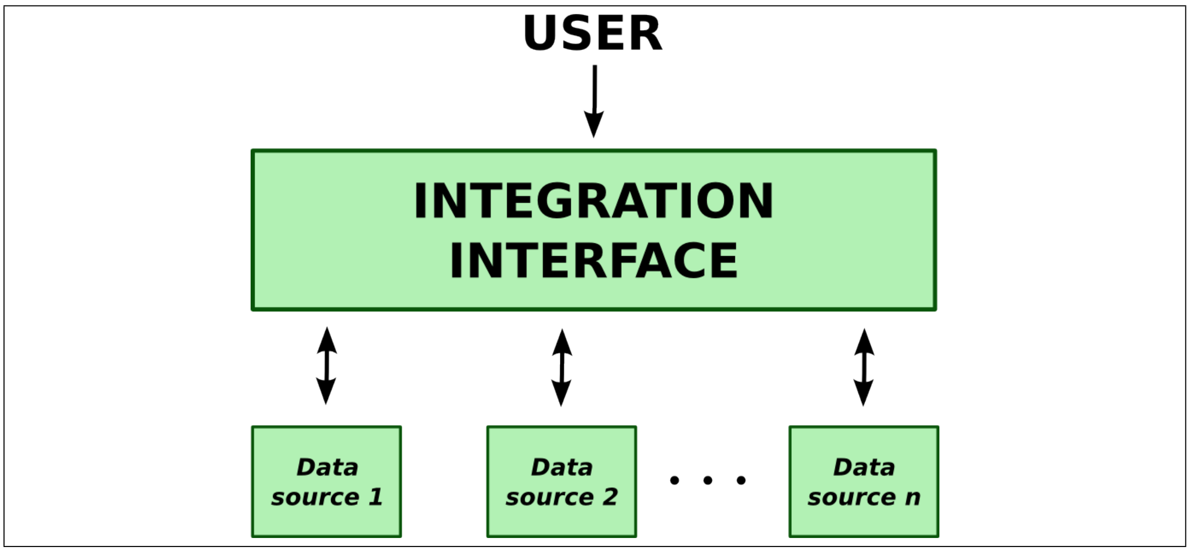


Figure 1.16: Single schema mapping provides a single view over multiple data sources.

Hybrid mediator mapping Hybrid mediator mapping utilises multiple schemas that either inherit from a global mediator or map to each other via a common mediator. Such a mapping must guarantee that all queries be resolved correctly for all appropriate data sources [145]. While data traditionally comes from multiple, autonomous database sources, the use of hybrid mediator mapping from the start of a multi-group project could simplify integration and ease the data storage burden on any one group. However, the difficulty of enforcing such a social and political restriction necessarily limits the environments in which this mapping method can be utilised. In suitable systems, related but non-intersecting data is stored in disparate locations using a common schema and viewed from a common interface [140]. Local-as-view mapping as well as the other mappings available to single mediator methods can be used with hybrid mediator systems, as the data sources are aware of the integration interface.

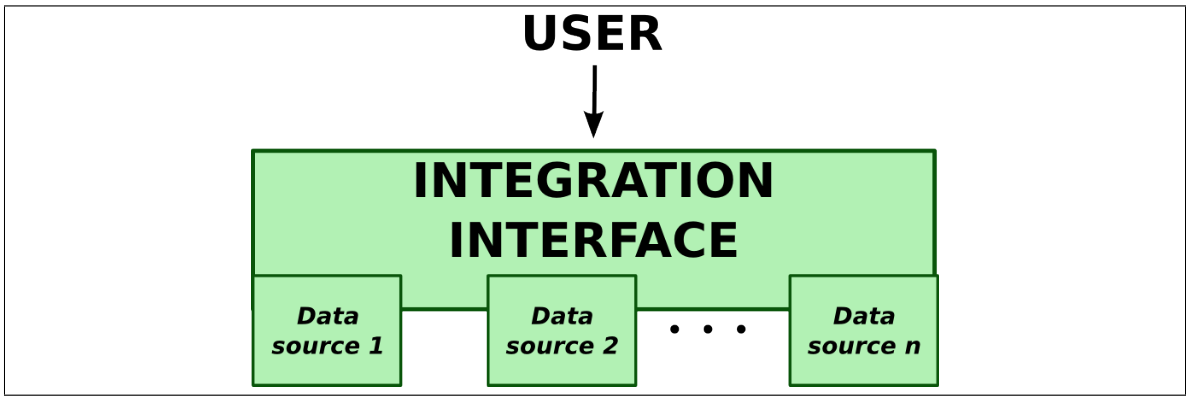


Figure 1.17: Hybrid mediator mapping provides high quality integration at the cost of independence of the data sources. This mapping type requires that all data sources know and implement the common schema or ontology used in the integration layer.

1.6.4 Integration interfaces

Irrespective of whether syntactic or semantic structures are used, integration interfaces for the mediator-based mapping types described in Section 1.6.3 can be divided into two broad categories: data warehousing using applied schemas or ontologies and federated data access using conceptual schemas or ontologies. If only two sources need to be reconciled, simpler methods without a mediator—such as direct mapping—are an effective strategy, as only two conceptualisations need to be compared. However, the complexity increases as more schemas or ontologies are added, rapidly limiting direct approaches. The use of a mediator allows an arbitrary number of resources without a large increase in complexity of the integration. Each integration type has its own benefits and constraints, and an integration project's particular requirements will influence which method is used. Warehousing provides a stable resource under the control of the integration project which can be much quicker at retrieving data than federated methods. However, data federation ensures that the data being queried is up-to-date and the integration infrastructure lightweight.

1.6.4.1 Data warehousing

Data warehousing is when a single resource partially or wholly centralises multiple data sources into one integrated view [139, 151, 150, 149]. Data warehousing techniques can be useful, especially if there is relatively easy access to the data sources, but can be problematic due to high maintenance requirements. Whenever underlying databases change, the integrated database needs to be updated [145], and any format changes require the appropriate parsers to be modified as well. This "database churn" was identified by Stein to be a major limiting factor in establishing a successful data warehouse [139].

When applied to syntactic integration methodologies, the data warehouse is normally a relational database. When used for semantic integration, RDF-based triple stores are often used. *Linked data* is a way of publishing information which Tim Berners-Lee has defined as a collection of URIs structured using RDF for the purpose of making links that are accessible and searchable [162]. Linked open data is a World-Wide-Web Consortium (W3C) community project which publishes freely available linked data as RDF datasets on the Web and which links data among the different sources [31]. Some linked data systems are enhanced with OWL as well as RDF. Shared ontologies can be used to build connections between RDF data files, building upon existing connections among datasets.

Data encapsulation Data encapsulation is the name chosen in this thesis for the subtype of data warehousing where multiple mediator mappings are stored in a single database. These data sources may be queried and presented as a completely integrated view, but the underlying data sources remain distinct. The querying side of data encapsulation is identical to query translation in that, for both methods, the API resolves differences in data structure as and when the data is requested. However, as a warehousing technique, data encapsulation stores information locally, and may pre-process incoming data before storage.

Though the Sequence Retrieval System (SRS) [163] is frequently described as an example of information linkage, it is more correctly a form of information linkage via data encapsulation. For instance, Stein [139] and Birkland [151] classify SRS as using information linkage, albeit one where fields can be structured, and two fields in different data sources can be explicitly connected. However, information linkage is commonly defined as a decentralised method of relating data items where hyperlinks are provided as cross-references between data entries in different databases [149]. Therefore, SRS is more precisely defined as information linkage through data encapsulation. SRS provides

[31] http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

linking between data entries, but stores all databases locally in a flat-file format which is then indexed. While not a relational implementation, it is a data warehouse with powerful local indexing, searching and analysis tools. All data sources are centralised, and all analysis tools are run on the server rather than the client side.

The Atlas data warehouse is a data encapsulation scheme which stores multiple data sources accessible through SQL queries and a higher level API [164]. Each individual database is imported into its own section of the global Atlas schema, retaining its own identity within the larger warehouse [164]. Atlas relies on its API to resolve differences and present a unified view. While it makes limited use of ontologies, a semantically identical protein present in two different databases will not be given the same identifier.

The majority of the data warehousing projects utilising RDF use data encapsulation. RDF databases provide datasets in a syntactically similar way using this Semantic Web technology, but the use of RDF does not necessarily imply resolution of semantic heterogeneity. For instance, while the linked data resource Bio2RDF [165, 166] stores millions of RDF triples, queries must still trace a path against existing resources rather than have those resources linked via a shared ontology or ontologies [147]. Similarly with linked open data, warehousing is common as conversion to RDF must occur even when the triples remain organised according to the original data structure. However, it is more common for ontology-based semantic integration systems using mediators to be federated systems, as warehousing creates large ontologies and consequent slow reasoning times.

Neurocommons is a large RDF resource which can be accessed through standard RDF methods such as SPARQL. Neurocommons converts the structure of selected data sources to OWL and then passes data instances via these OWL representations into an RDF database [167]. Neurocommons stores one named graph per data source and provides a SPARQL endpoint for querying. Each one of these named graphs acts as a mediator in the data encapsulation process. The intention of the Neurocommons authors is to create a Semantic Web for science [167].

Other linked open data projects include BioLOD[32] and OpenFlyData [168]. Some projects, such as Linked Life Data[33], provide both open and copyright-restricted data. Other RDF databases in the life sciences include BioGateway [85], YeastHub [169], LinkHub [170] and S3DB [171]. Many others are listed in Antezana and colleagues [11, Table 1].

[32]http://biolod.org
[33]http://linkedlifedata.com/

Data translation Data translation [140, 145] is the phrase commonly used to describe a data warehouse which transforms data from multiple native formats to a single unified schema. This method allows seamless integration in a controlled environment and greater control over the data than federated approaches (see Section 1.6.4.2). However, it is unclear if what seems to be the same actually is semantically identical data. Additionally, frequent update cycles are required to maintain usability, and political obstacles such as licensing or access restrictions prevent redistribution of databases. Licensing restrictions force both data translation and data encapsulation implementations to limit availability to either a download of the software on its own, or to a limited implementation of a working installation containing only unrestricted resources [17].

The Genome-based Modelling System [172] is a database of genome-scale networks generated by linking a variety of databases via Enzyme Classification (EC) number. Enzyme information from UniProt/Swiss-Prot, KEGG [173], WIT [174] and BRENDA [175] are integrated into the GEM database together with all EC-associated genes from EMBL/GenBank/DDBJ [92] whole bacterial genomes. All genomes are available for viewing as qualitative models however very few quantitative SBML models are available. Further, while the Genome-based Modelling System is successful at presenting a genomic view of known pathways, it does not suggest any novel ones.

Another integration system using data translation is BioWarehouse [176], which uses a relational database for storage. Its public implementation, PublicHouse, contains only openly accessible databases for which republication is legally allowed. Biozon [151] integrates data from protein sequence, nucleotide sequence, pathway and interaction databases, as well as a variety of novel information derived from these databases. Unlike BioWarehouse, Biozon uses graphs as its internal data structure rather than a relational format [151]. Each native schema is translated into a graph and then overlaid on the Biozon graph schema. When the user interface reads from this schema the results view itself is not unified, but instead returns a disjointed result set where the query term is shown in each of the data sources separately.

PathSys [177] is another graph-based data translation application and was the original back end to the BiologicalNetworks [178] client application. BiologicalNetworks presents the user with a much more unified view than that provided by Biozon. For each data source in PathSys, two schemas are loaded prior to data upload: the native schema of the data source for initial reading of the data, and the mapping schema for conversion of the native data into the global PathSys schema. As the BiologicalNetworks resource has matured, it has replaced its original syntactic data translation system PathSys with a new back end called IntegromeDB [179]. This new system uses OWL and RDF for integrating and formatting the data. Like Biozon and BiologicalNetworks, ONDEX [36] makes use of a graph-based storage structure and stores its information with a similar level of expressivity to RDF. ONDEX uses a set of 15 ontology and pathway database parsers to create a graph-based data structure for generating, viewing and analysing networks which can be exported as RDF [36].

1.6.4.2 Data federation

Also known as *view integration* [139, 149] and *mediator* systems [151, 150], *data federation* translates a unified query into native queries for each underlying data source. The retrieved results are combined and presented to the user. Integration via data federation is easier to maintain than data translation as it does not rely on local, and possibly outdated, copies of data sources. Federation has a lower maintenance cost compared to data warehousing methods and provides the most up-to-date versions of the underlying data. Space requirements are also lower, as it does not require local installation of all source databases. However, the processing of a single query into the sub-queries to be sent to each data source is a complex process which can itself have a high overhead. Additionally, the integration method's query engine is limited by the querying abilities of each native resource and of the quality of the internet connection to each resource. Finally, the querying of remote databases is often slower than equivalent warehousing options, and will not be any faster than the slowest data source [149].

While query encapsulation is possible in theory, in practice such methods are rarely used due to their impracticality. A hypothetical example of such a system would be similar to SRS or some other Web-based information retrieval service. Rather than presenting a unified query interface, this system would show the available data sources and provide query forms for each source. Such methods are impractical because a lot of work is done for very little gain over simply visiting the data sources directly. Therefore, in a very real sense, data federation is almost entirely composed of query translation methods.

Query translation [140, 145] is where a mediator schema or ontology is used to distribute a single query across all data sources known to the integration system [151]. Sujansky divides query translation into two broad categories: *declarative mapping*[140] and a hybrid of query and data translation called *procedural mapping*. Procedural mapping methods fetch data from underlying data sources and import the results into global objects, which can then be further manipulated before presentation. Querying and integration occur within the same procedure. Declarative mappings separate the global and native data models from any code that performs the queries. This separation results in a lower level of maintenance, as changes to a native data model only require updating of the global schema, and the query layer only has to work within a single global model.

ArrayXPath II [180] is a query translation system which accepts user input in the form of gene expression clusters, and visualises those clusters in the context of a variety of major pathway databases, providing a level of statistical significance for how well each expression cluster matches each node of the best-fit pathways. ArrayXPath II then layers the expression levels, if provided, over the pathway. Another example of query translation is the DiscoveryLink system by IBM [181]. DiscoveryLink presents the user with an SQL-based API which connects to a single mediator mapping scheme. The mediator is a middleware engine which breaks down the original query into sub-queries for the external data sources.

Query translation systems utilising ontologies include the TAMBIS ontology-based query system [182] and ComparaGRID[34]. TAMBIS allows the user to create a query according to its own global ontology. Each query is transformed into concrete searches specific to each data source. The results are then sent back to the user and displayed within the global ontology [182]. ComparaGRID uses local-as-view mapping to create application ontologies which are trimmed down versions of the semantically rich domain ontology describing the biology of interest. This method of creating on-the-fly application views allows ComparaGRID to reach a wide variety of users quickly and easily, but means it also suffers from a lack of efficiency [183].

OntoFusion uses the local-as-view method of query translation to build virtual local schemas—one per data format—that describe those formats using semantically identical concepts from the shared domain ontology [145]. While this allows for automated alignment of different data types because their virtual schemas share a common core ontology, the method relies on query translation rather than data warehousing, which can be costly.

1.6.5 Rule-based mediation

The purpose of the work described in this thesis is to annotate systems biology models using data from multiple data sources stored in multiple formats. Ultimately, while current methodologies have a variety of benefits as described in this section, a new semantic integration methodology was designed. Semantic approaches were chosen over syntactic integration methods to allow rich models of the biological domain to be created. Semantic structuring of the data is vital for making the knowledge accessible to humans and machines. Non-mediator mapping types were discarded due to a lack of flexibility. Single mediator mapping was chosen for its ability to present a unified view to both users and applications accessing the data. Further, the hybrid mapping subtype was chosen as

[34]http://bioinf.ncl.ac.uk/comparagrid/

it allows the global domain ontology to be completely independent from the source formats. This ensures that the biological domain ontology remains free from any reference to the structure of the underlying data sources. Finally, pure warehousing options were discarded as the large amount of data would be incompatible with reasoning and other semantic tasks. Therefore, the integration interface normally functions as a federated resource to minimise reasoning times, but can also store data and build queries over longer time scales.

As a result, *rule-based mediation* was developed using single mediator mapping with modified query translation. Ultimately, rule-based mediation builds tailored views of systems biology data accessible to computer programs and to users, allowing both automated and manual model annotation. Further details of rule-based mediation, including a comparison with the methodologies presented in this section, are available in Chapter 5.

Chapter 2

SyMBA: a Web-based metadata and data archive for biological experiments

2.1 Introduction

With the development and adoption of high throughput experimentation techniques in the life sciences, vast quantities of heterogeneous data are being produced. Consequently, there is a need to integrate these diverse datasets to build a more system-level view of organisms under study. These datasets must be archived, curated with appropriate metadata, and made available in a suitable form for subsequent analysis and integration [144].

There is also an urgent need to provide tools and resources that allow the digital curation and management of scientific data in a fashion amenable to standardisation and capable of facilitating data sharing and knowledge discovery [184]. It is desirable that this data storage and annotation process is implemented as early as possible in the data management cycle to prevent data loss and ensure traceability throughout.

This chapter describes the development of the Systems and Molecular Biology Data and Metadata Archive (SyMBA), a data and metadata archive for scientific data [185]. SyMBA addresses the need for data standardisation described by Paton [184] by implementing a method of experimental metadata integration for systems biology. SyMBA is an open source software system providing a mechanism to collate the content, syntax and semantics of scientific data, regardless of type. It can be used to review, share and download metadata and data associated with stored experiments through a simple Web interface. SyMBA accommodates both the representation of community-endorsed data standards and the development of bespoke representations to suit users' digital curation needs.

Section 2.2 contains a summary of life-science data standards and how they are utilised by SyMBA. A detailed description of how SyMBA was implemented follows in Section 2.3, including initial requirements, architecture, data storage and retrieval techniques, and versioning methodology. Section 2.4 explains how SyMBA can be used, while Section 2.5 describes similar work and discusses the benefits of SyMBA as part of a larger life-science research life cycle.

2.1.1 Availability

A sandbox installation of SyMBA is freely available to test at `http://symba.ncl.ac.uk`. Researchers wishing to use SyMBA as a local data and metadata archive may evaluate it by downloading the system and customising it for their needs. The Web application is implemented in Google Web Toolkit (GWT)[1], with all major browsers supported. SyMBA is built with Apache Maven 2[2], a soft-

[1] `http://code.google.com/webtoolkit/`
[2] `http://maven.apache.org/`

ware project management and build system similar to but more comprehensive than other build tools such as Apache Ant or GNU Make. SyMBA has a Sourceforge.net project website[3] and a Newcastle University project page[4]. The Sourceforge.net project website provides access to the Subversion repository[5] as well as various explanatory documents for programmers[6].

SyMBA is licensed by Allyson Lister and Newcastle University under the LGPL[7]. For more information including licensing for third-party libraries used in the application, see LICENSE.txt in the SyMBA Subversion project. Installation and running of SyMBA has been tested on 32-bit Ubuntu 8.10 and higher. Development questions can be directed to symba-devel@lists.sourceforge.net.

2.2 Data Standards

To ensure maximum reuse of published data, Lynch [186] has stated that scientists should act as data stewards in three ways:

- honour established disciplinary data standards;
- record appropriate metadata to make future interpretation of the data possible; and
- define metadata such as provenance and parameters, ideally at the time of data capture [186].

SyMBA makes use of a number of standards, emerging standards, and common formats to save time downstream of data generation by ensuring compatibility with other centres of research and journals complying with the same standards. SyMBA can help research groups follow all of the stewardship methods described above and can limit tedious and repetitive data entry. A brief overview of content, syntax and semantic data standards that are or can be integrated within SyMBA is provided in this section. A more detailed description of these standards is available in Section 1.4.

2.2.1 Data Content

The results and conclusions of scientific investigations are dependent upon the context, methods and data of the associated experiments. Defining a list of commonly required metadata to guide researchers in reporting the context of these experiments is increasingly gaining favour with data repositories, journals and funders [71]. These metadata reporting checklists outline the important or

[3]http://symba.sourceforge.net
[4]http://cisban-silico.cs.ncl.ac.uk/symba.html
[5]http://symba.svn.sourceforge.net/viewvc/symba/
[6]http://symba.sourceforge.net/symba-books/general-information/index.html, http://symba.sourceforge.net/symba-books/installation/index.html
[7]http://www.gnu.org/copyleft/lesser.html

minimal information integral to the evaluation, interpretation and dissemination of an experiment; such guidelines effectively define what is contained within a scientific dataset and how the set was generated. The MIBBI Registry[8] provides a listing of these checklists. SyMBA is specifically designed to allow users to store and annotate their data according to any content standard such as those registered with MIBBI. Users can create and save standards templates for guidelines such as MIAME [55] and MINI electrophysiology [187]. Saved templates are then accessible to all users of SyMBA.

2.2.2 Syntax

Digital curation in the life sciences is predominately database centric [52]. While repositories have been developed for single datatype 'omics' experiments [54, 188, 91, 189], until recently the lack of suitable data standards has hampered the development of a data storage system capable of storing *multiple* data types in combination with a uniform set of experimental metadata.

Recently, two related projects have been created to address the need for a common syntax to describe experimental metadata: FuGE and ISA-TAB [190]. The FuGE project was formed with the aim of standardising the experimental metadata for a range of omics experiments. The FuGE standard contains a model of experimental objects such as samples, protocols, instruments, and software, and provides extension points for the creation of technology-specific data standards [2]. The FuGE project's native format is UML, with the FuGE-ML XML export as its most commonly used format. The availability of FuGE makes the development of a generic data and metadata storage portal feasible not only to perform data capture, but also for the purpose of integrating metadata from a range of experimental datasets. Among others, the proteomics[9], metabolomics[10], microarray[11] and flow cytometry[12] standards groups have adopted the FuGE standard either wholly or for certain aspects of their formats. The structure of SyMBA is based on version 1.0 of FuGE-ML and accepts input and provides output in that format.

The ISA-TAB project is a standard for describing experiments using a tabular format, and is ideal for researchers primarily using spreadsheets. ISA-TAB has been developed alongside FuGE and can be converted into FuGE if required [191]. While spreadsheet formats are useful for data entry and record keeping on a small scale, an XML format—and the associated benefits such as XPath and the

[8]http://mibbi.org
[9]Proteomics Standards Initiative, http://www.psidev.info
[10]Metabolomics Standards Initiative, http://msi-workgroups.sourceforge.net
[11]FGED Society, http://www.mged.org
[12]Flow Informatics and Computational Cytometry Society, http://www.ficcs.org

large library of tools for the XML format—was deemed more suitable for use within SyMBA.

2.2.3 Semantics

Describing the meaning, or semantics, of data and its associated metadata can be a complex and difficult process currently being addressed in the life sciences through the use of controlled vocabularies or ontologies [81]. The use of ontologies for describing data has a twofold purpose. Firstly, ontologies help ensure consistent annotation, such as the spellings and choice of terms, such that a single word-definition pair is used to describe a single entity. Secondly, ontologies can add human- and computationally-amenable semantics to the data. Related datasets can be integrated and queried via a common ontology, for example linking experimental results with associated publications. In SyMBA 1.0, users could pre-load controlled vocabularies and portions of ontologies to limit the descriptor choices made available to users. However, it was a complex feature to implement and was not heavily used. In SyMBA 2.0 the functionality was modified such that term choices previously saved by users are offered as auto-complete suggestions in some text fields. When an interface better suited to the addition of subsets of ontologies and controlled vocabularies has been designed, this feature may be reintroduced.

2.3 Implementation

One of the major requirements of representing scientific data is to be able to store raw, unprocessed data together with information about the equipment, processes, people and conditions under which they were generated [184]. Data arises from a variety of sources and is generated by many different protocols and experimental procedures. Metadata, or information about how the data was generated, should also be captured to ensure that recommended minimal information standards are met. Further there should be methods, whether directly provided from data producers or indirectly via public repositories, to export the data in standard formats. Users also need to be able to retrieve their data and track modifications to their metadata through the assignment of appropriate unique identifiers to ensure traceability and data provenance. Finally, and perhaps most importantly to users of a bioinformatics application, there is a need for an effective user interface. The rest of this section describes the tools and techniques used to meet these requirements.

2.3.1 Toolkit capabilities

As shown in Figure 2.1, the SyMBA toolkit provides a number of features:

- the choice of a FuGE-structured Relational Database Management System (RDBMS) or in-memory back-end;
- an object-relational persistence and query layer using Hibernate[13];
- a set of plain Java classes representing FuGE-ML entities which also connect to the database via hyperjaxb3[14];
- unit tests of all queries and version retrieval functionality; and
- a user-friendly Web interface that interacts with the chosen back-end and a remote file store for raw data outputted from experimental assays.

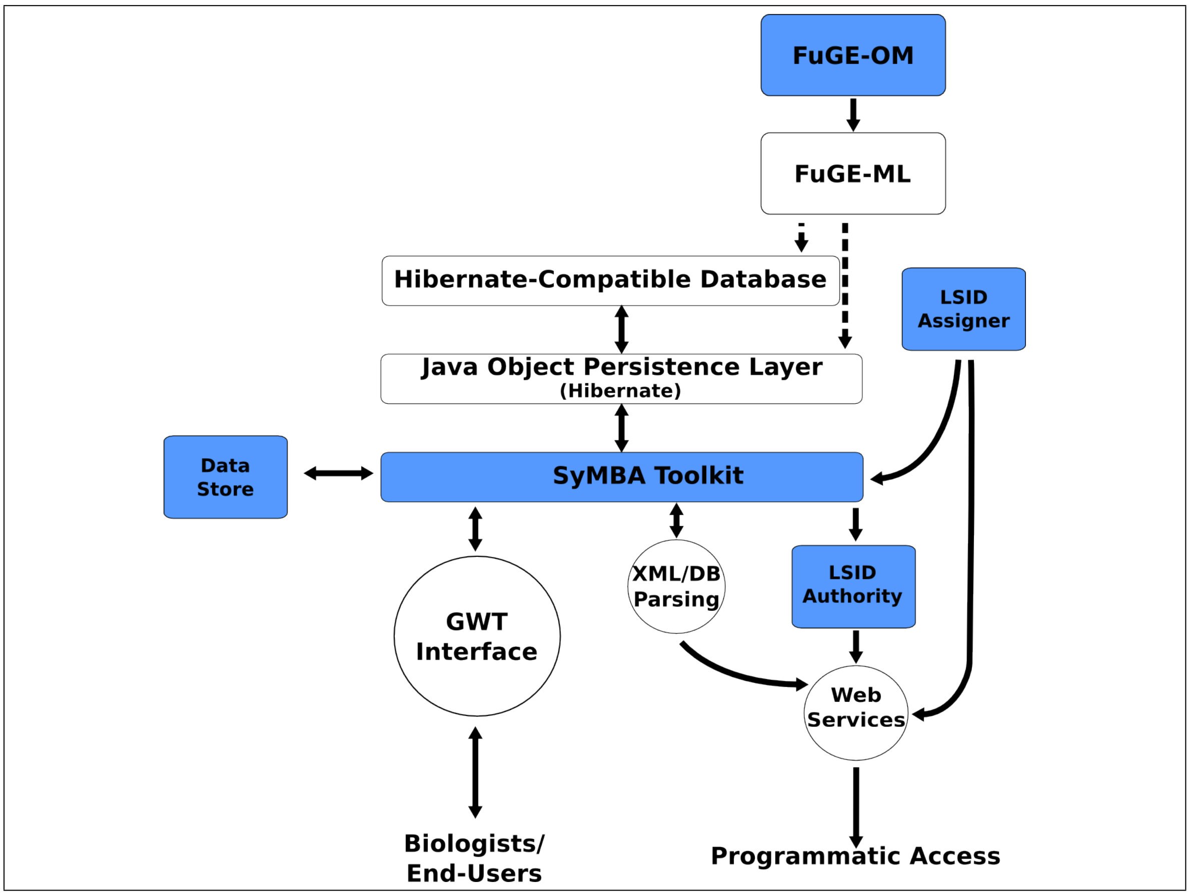


Figure 2.1: An architectural overview of SyMBA. Blue shading signifies hand-written code, while unshaded areas designate automatically-generated code. For the back-end, automatically-generated code is created with hyperjaxb3 (shown with dashed arrows). GWT automatically converts the front-end Java code into a compiled Web archive. Circles mark agent access points; solid arrows indicate direction of data flow.

[13]`http://www.hibernate.org`
[14]`http://confluence.highsource.org/display/HJ3/Home`

FuGE provides all of the structures required to adequately describe an experimental protocol, including data, analyses, and information about the structure of the experiment. At a basic level, information in FuGE-ML may be stored simply as a collection of XML documents or as records in an XML database. However, the use of a RDBMS allows the storage of metadata in a format amenable to searching via technology that is trusted, scalable and familiar to the bioinformatics community. SyMBA uses hyperjaxb3 to create a Hibernate persistence layer and database as well as an automatically-generated FuGE Java API based on the FuGE XSD. Hibernate is a service that can be used with any compatible RDBMS and which allows the creation of an object/relational database persistence and query layer, thus abstracting the low-level database code away from the programmer interface.

In addition to the database back-end, SyMBA also implements a memory-only back-end, which facilitates the testing and initial set-up of SyMBA without needing to first install a database. The in-memory version of SyMBA has all of the capabilities of the database implementation except the ability to store the entered metadata beyond the lifetime of the Web application itself. If an in-memory version of SyMBA is restarted, all data previously uploaded is deleted. This makes the memory implementation ideal for test or sandbox installations.

2.3.2 Architecture

On the client side, SyMBA is a simple Web application for describing experiments and storing experimental data. Behind the user interface, SyMBA is implemented in GWT and hosted on a Tomcat[15] server. GWT was chosen because it abstracts away the details of the user interface and allows the programmer to create a working Web application quickly and without needing to know about differences between Web browsers. Figure 2.1 shows an architectural overview of SyMBA and how GWT relates to the application as a whole, while Figure 2.2 shows how GWT can be used to create a Web application.

With GWT, the programmer writes using standard Java. The Java code is separated into client-side and server-side classes. During testing, all of the code is compiled using a Java compiler and run in *hosted mode*, a development environment for GWT where a dedicated browser runs the Web application and any errors in the code are reported with standard Java exceptions. In this way, hosted mode makes debugging the application easier. When ready for production, server-side classes are compiled with a Java compiler and the client-side classes are converted into JavaScript using the

[15] http://tomcat.apache.org/

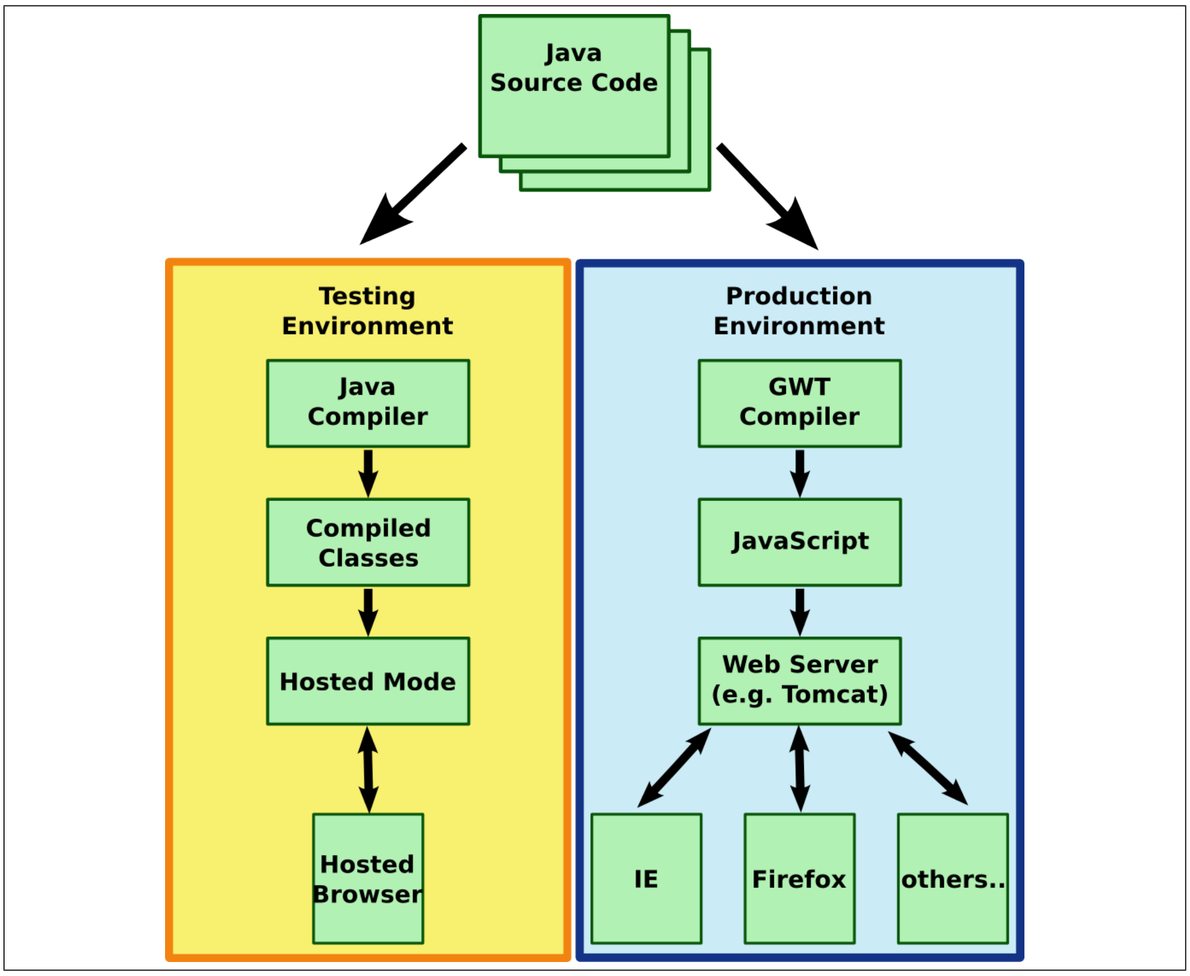


Figure 2.2: A simplified overview of how Google Web Toolkit creates a Web application. Hosted mode is a development environment for Google Web Toolkit applications, which allows more detailed debugging than is available when running the application using a standard Web server.

GWT compiler. This results in JavaScript created specifically for each of the common browser types. This creation of multiple versions of the application allows the programmer to ignore the details of cross-browser compatibility. At the end of the production compilation, a Web archive is produced which can be deployed on an appropriate server such as Tomcat.

SyMBA 1.0 made use of Java Server Pages, which have the benefit of being simple to learn and quick to implement. However, much of the processing code and display code was mixed within the pages, and the application as a whole was not highly configurable. Templates for new experimental methods had to be created by hand, or through a convoluted method of building template FuGE-ML files which would be converted on-the-fly by the Web front end. SyMBA 2.0 allows all users to generate templates via the Web interface rather than by hand-coding XML or Java Server Pages. The new template generation functionality opens up the use of SyMBA templates to a larger community of biologists rather than just bioinformaticians.

2.3.3 Identifiers

All objects in FuGE-ML inherit from the Describable complex type. This entity allows basic information such as audit trails, additional ontology terms and textual descriptions to be added to all objects. Not all FuGE objects require unique identification, and as such entities that inherit only from Describable have no requirement for identifiers. Most objects in FuGE which describe entities rather than collections of those entities also extend Identifiable, which adds a unique identifier and human-readable name to objects of this type.

SyMBA takes this level of identification one step further, and ensures that all identifiers created within SyMBA are based on Life Science Identifiers (LSIDs) and therefore globally unique. To facilitate versioning (described further in Section 2.3.4), SyMBA introduced the abstract Endurant entity as a new child of Describable and sibling of Identifiable, as shown in Figure 2.3. Both Endurant and Identifiable objects are identified within SyMBA using LSIDs.

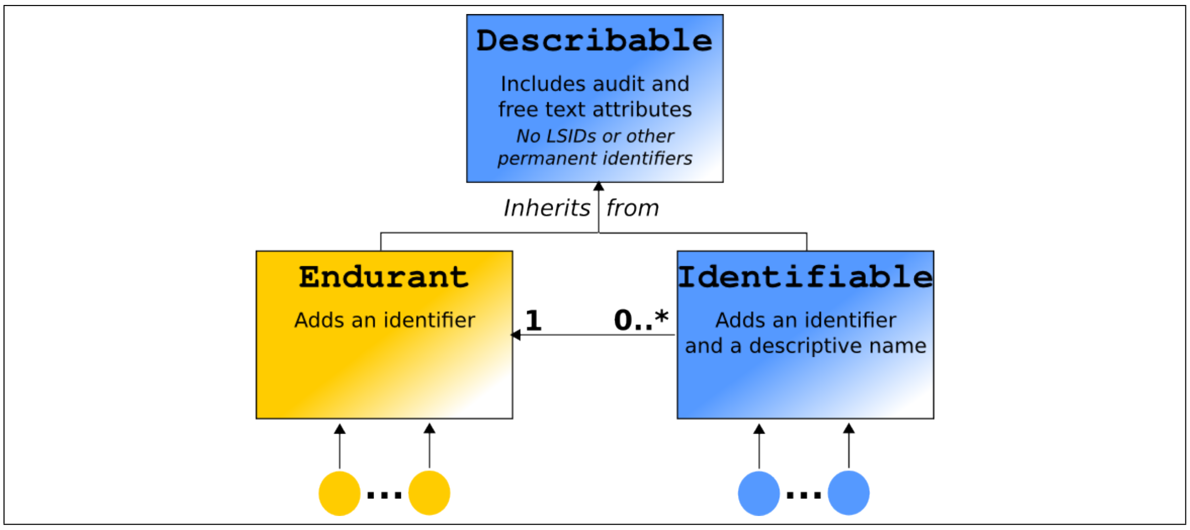

Describable
Includes audit and free text attributes
No LSIDs or other permanent identifiers
Inherits from
Endurant
Adds an identifier
1
0..*
Identifiable
Adds an identifier and a descriptive name
...
...

Figure 2.3: The relationship of the SyMBA-specific Endurant element to existing FuGE structures. Abstract entities are represented as rectangles, and concrete entities as circles. The Endurant element and its children (shown in yellow) were added to the FuGE XSD to provide all Identifiable elements with a mechanism to group versions together. The Describable and Identifiable elements are shown in blue and are key elements of the original FuGE XSD. Both Endurant and Identifiable entities are children of Describable, and both are identified in SyMBA with an LSID.

In order to describe how the identifiers are created within SyMBA, the LSID specification must be examined more closely. LSIDs contain both metadata and data. LSID metadata is information about the LSID itself, while LSID data is the information of interest associated with that LSID. An important requirement of LSIDs is that the returned data for a particular LSID must always be identical to the data retrieved in past queries of that LSID. Unlike its data, an LSID's metadata can change at any time and is not required to always return the same object. LSID metadata includes an expiration timestamp which, if null, means the LSID does not expire, as well as any other information that the supplier of the LSID wishes to provide. In SyMBA, the timestamp will always be null, and the provided metadata is an RDF-formatted object with a title and a value. The metadata value is the LSID of the most recent Identifiable object in that version group. More information on these methods is available in Section 2.4.6 and Table 2.1.

The use of the terms *data* and *metadata* within the LSID specification should not be confused with their definitions within the FuGE standard. FuGE is a format standard for describing the metadata associated with an investigation. Some raw data can be defined within the FuGE structure, although in SyMBA associated data files are always stored separately from the FuGE metadata.

Over the past five years there has been a large amount of controversy on whether LSIDs are sustainable in the long term, or if HTTP URIs are a better solution[16]. Recently, a number of high-profile biology standards such as MIRIAM have developed non-LSID schemes such as `identifiers.org`, which uses URIs to provide resolvable, permanent identifiers for many life sciences databases. As SyMBA is not intended to be a public large-scale warehouse of experimental data but rather an application for individual research groups to store their data locally and in a standard format, URIs based on the HTTP protocol are not required. Whether or not LSIDs become a heavily-used standard, as URNs they can be easily converted to any other identifier system which does gain widespread acceptance.

2.3.4 Versioning

The requirement for maintaining data provenance means that modification histories for all metadata must be recorded. The issue of versioning is therefore important. The core FuGE structure can record simple audit information but does not store histories and details of changes to experimental objects. To allow access to previous versions of objects, each versioned object has a stable and globally-unique identifier. Reflecting its more permanent nature, versioning is used when running the database implementation of SyMBA but not with the in-memory implementation.

As shown in Figure 2.3, the three main classes involved in versioning in SyMBA are Describable, Endurant and Identifiable. All three are abstract, and may not be directly instantiated. Endurant and its child classes were created to add versioning to the FuGE model. Endurant objects do not change their intrinsic value over time, and are identified with an LSID. It is useful to distinguish between a modification to an actual experimental object, such as the addition of a new camera to

[16]`http://lists.w3.org/Archives/Public/www-tag/2006Jul/0041`, `http://i9606.blogspot.com/2007/06/main-problem-with-lsids.html`, `http://iphylo.blogspot.com/2007/07/lsid-wars.html`

a robot, and a modification of the stored metadata, such as fixing a spelling mistake in the name of the camera (see Figure 2.4). However, while the SyMBA back end allows these two types of versioning, the front end will currently always attach the new Identifiable to the existing Endurant. The difference between the two types of versioning can be difficult for users to grasp and therefore at this time only one method has been added to the user interface.

Endurant objects point to one or more versions of non-enduring Identifiable objects and are unresolvable by the LSID Web services. This restriction prevents the LSID Authority from breaking the LSID specification, as the authority must not allow the same LSID to resolve to different objects. All other objects identified with an LSID are children of the Identifiable entity, and identify a particular version of the object associated with the Endurant. As such, Identifiable LSIDs will always point to the same, single, version of the object and are resolvable.

The Endurant classes allow every Identifiable in FuGE to represent a single revision in the history of the Endurant: new objects are added to the database with every change, however minute. The state of an object at any point in its history is easily retrievable, as is the latest version of each object. Error-correction or addition of information, such as fixing a typo, will create an additional Identifiable associated with the same Endurant.

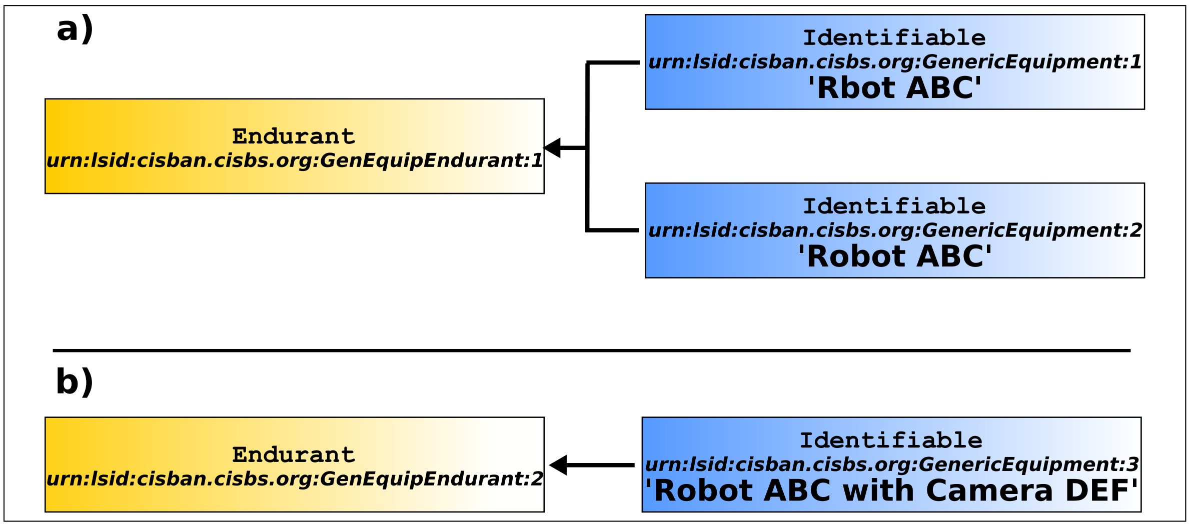

a)
Identifiable
urn:lsid:cisban.cisbs.org:GenericEquipment:1
'Rbot ABC'
Endurant
urn:lsid:cisban.cisbs.org:GenEquipEndurant:1
Identifiable
urn:lsid:cisban.cisbs.org:GenericEquipment:2
'Robot ABC'
b)
Endurant
urn:lsid:cisban.cisbs.org:GenEquipEndurant:2
Identifiable
urn:lsid:cisban.cisbs.org:GenericEquipment:3
'Robot ABC with Camera DEF'

Figure 2.4: The implementation of Endurant objects within SyMBA. In a), a superficial change to the spelling of an equipment name creates a new Identifiable assigned to the existing Endurant. In b), modifications to the type of equipment used requires the creation of a new Endurant as well as a new Identifiable object.

Both the addition of new objects and updates to existing objects have the same cost within SyMBA, as both actions result in the creation of a new object; no cascade event or propagation of the new object is required. Such a model is suitable for an archival database as it is during data retrieval, rather than data storage, when new versions of each object are queried for and retrieved. This produces an application that is quick to update, but not as quick to retrieve.

2.3.5 Investigations and Experiments

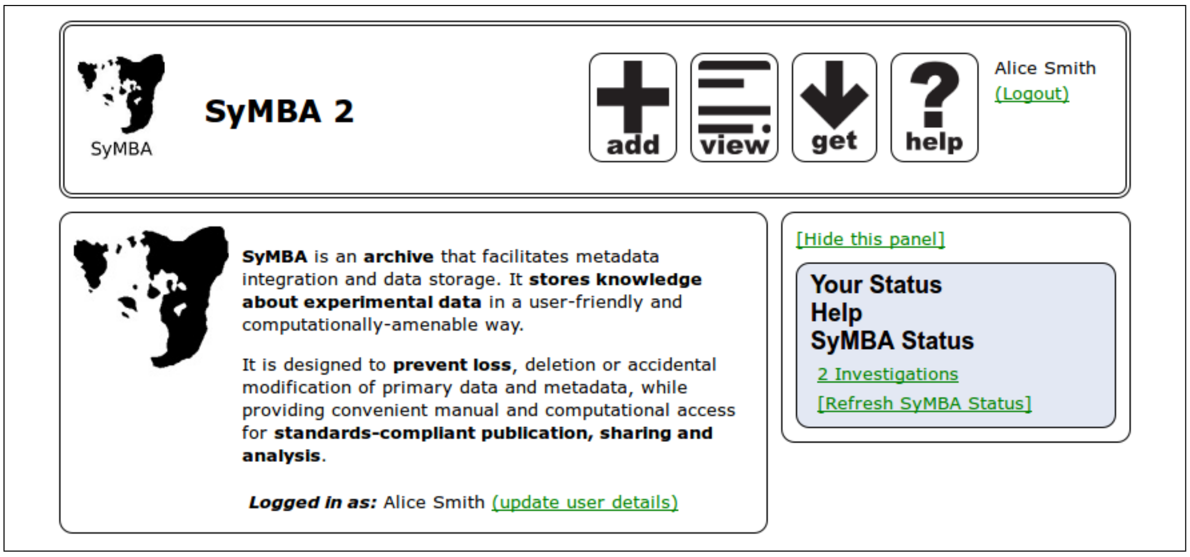
SyMBA uses a specific terminology for describing experiments based on the FuGE standard. An investigation is a high-level concept which contains hypotheses and conclusions as well as a set of one or more experiments. As such, every experiment in SyMBA must be contained within a top-level investigation. Experiments can be arranged in an ordered hierarchy of steps and may have data files associated with them (see Figure 2.8 for an example).

2.4 Data and metadata storage and retrieval

Traditionally, laboratories stored raw data and metadata in disparate, interconnected resources such as spreadsheets, file stores, local databases and lab books. While these are useful tools for a conventional approach to biological research, they do not meet the needs of an environment where the generation of high throughput datasets is commonplace. Further downstream analysis of the raw data requires data to be adequately annotated and available in a standard format. In many cases, the researcher carrying out the analysis will not be the same person who generated the data. SyMBA has been developed to meet the need for a centralised raw data repository that collates and stores data and captures sufficient metadata in an integrated fashion to facilitate downstream analysis, export and publication. This section describes how SyMBA is utilised by individual users to store metadata, create templates and generate FuGE-ML for download.

2.4.1 Welcome and creating an account

The main entry point for end users is a Web-based graphical user interface that displays a simplified view of the underlying data structure and allows upload of data and description of experimental metadata. The welcome screen is shown in Figure 2.5. There are three main sections to the SyMBA Web application which are named hereafter using compass directions for their positions on the page. The header along the top of the page, or north panel, contains the SyMBA logo and name on the left, a variety of icons on the right, and the name of the user on the far right. The SyMBA logo will always take the user back to this welcome page. Named from left to right, the four icons on the right-hand side of the header are shortcuts to *add* a new experiment, *view* existing experiments, *get* data files or metadata FuGE-ML for download, and browse the *help* topics. If the user is not yet logged in, only the links to the welcome and help pages are active.



SyMBA is an **archive** that facilitates metadata integration and data storage. It **stores knowledge about experimental data** in a user-friendly and computationally-amenable way.

It is designed to **prevent loss**, deletion or accidental modification of primary data and metadata, while providing convenient manual and computational access for **standards-compliant publication, sharing and analysis**.

Logged in as: Alice Smith (update user details)

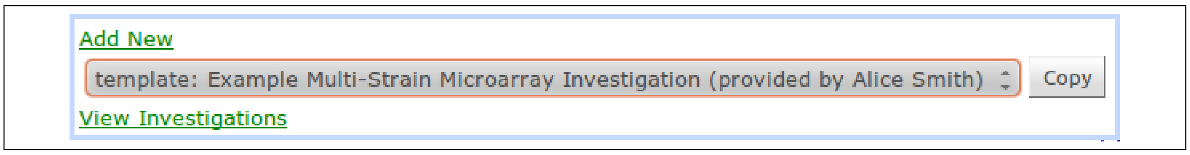
Figure 2.5: The welcome screen for the SyMBA Web application.

To the west is the main SyMBA usage panel. Initially, the west panel contains a short summary of SyMBA and a login form. Once the user begins work within SyMBA, the west panel displays summaries of experiments, the experimental metadata and much more. In the east is an optional status panel providing both contextual help and status messages. The status panel can be hidden at any time by clicking on "Hide this panel".

If SyMBA is being used in a sandbox or test installation, the user may prefer to login with a prepared guest account from the default installation of the application. A guest logs in by choosing "Guest Account" from the pull-down menu of users in the west panel, and clicking *Login*. Otherwise, a new user may be created by clicking on "add new user and log in" directly to the right of the login button. The user then fills out the contact form that appears and clicks *Save Contact*. The contact is saved and the user is automatically logged in. A named account, rather than a guest account, allows the stored metadata to be identified with a particular user.

2.4.2 Add and view buttons

As described earlier, the right side of the north panel contains a number of quick link buttons. Clicking on the *add* button results in the appearance of a pop-up window for creating new investigations as shown in Figure 2.6. This pop-up presents the user with three options. Choosing "Add New" will take the user directly to an empty investigation. The second option is a pull-down menu of existing investigations. Selecting one of these and pressing the *Copy* button will take the user to the standard investigation form pre-populated with a copy of the selected investigation. Finally, the "View Investigations" link allows the user to opt out of the *add* menu and instead view a list of existing investigations. This link works in a way identical to the *view* button from the north panel.


Add New

template: Example Multi-Strain Microarray Investigation (provided by Alice Smith)

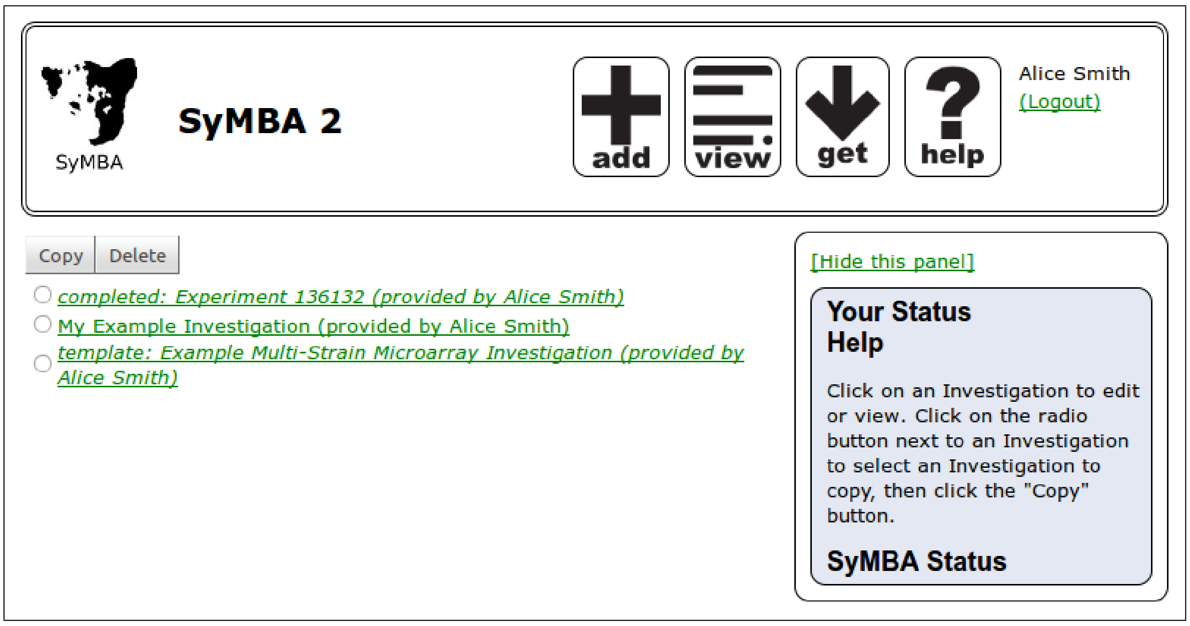
Copy

View Investigations

Figure 2.6: The pop-up window which appears when the *add* button from the north panel of SyMBA is pressed.

The *view* button present in the north panel takes the user to a summary of all investigations stored within SyMBA. Figure 2.7 shows an example summary page containing template, completed and standard investigations. Standard, modifiable investigations are shown in plain text, while read-only investigations are displayed in italics. Further, if an investigation is either completed or a template, this information is displayed in front of the investigation name. Both template and completed investigations are described in detail in the next section.

As explained in the contextual help section of the east panel, the user may either click on an investigation name to view its details or click on the radio button next to its name and then either *Copy* or *Delete* the investigation. SyMBA can be modified to disable or completely remove the *Delete* button.



completed: Experiment 136132 (provided by Alice Smith)

My Example Investigation (provided by Alice Smith)

template: Example Multi-Strain Microarray Investigation (provided by Alice Smith)

Your Status Help

Click on an Investigation to edit or view. Click on the radio button next to an Investigation to select an Investigation to copy, then click the "Copy" button.

SyMBA Status

Figure 2.7: The summary of investigations stored within SyMBA shown when the *view* button from the north panel is pressed.

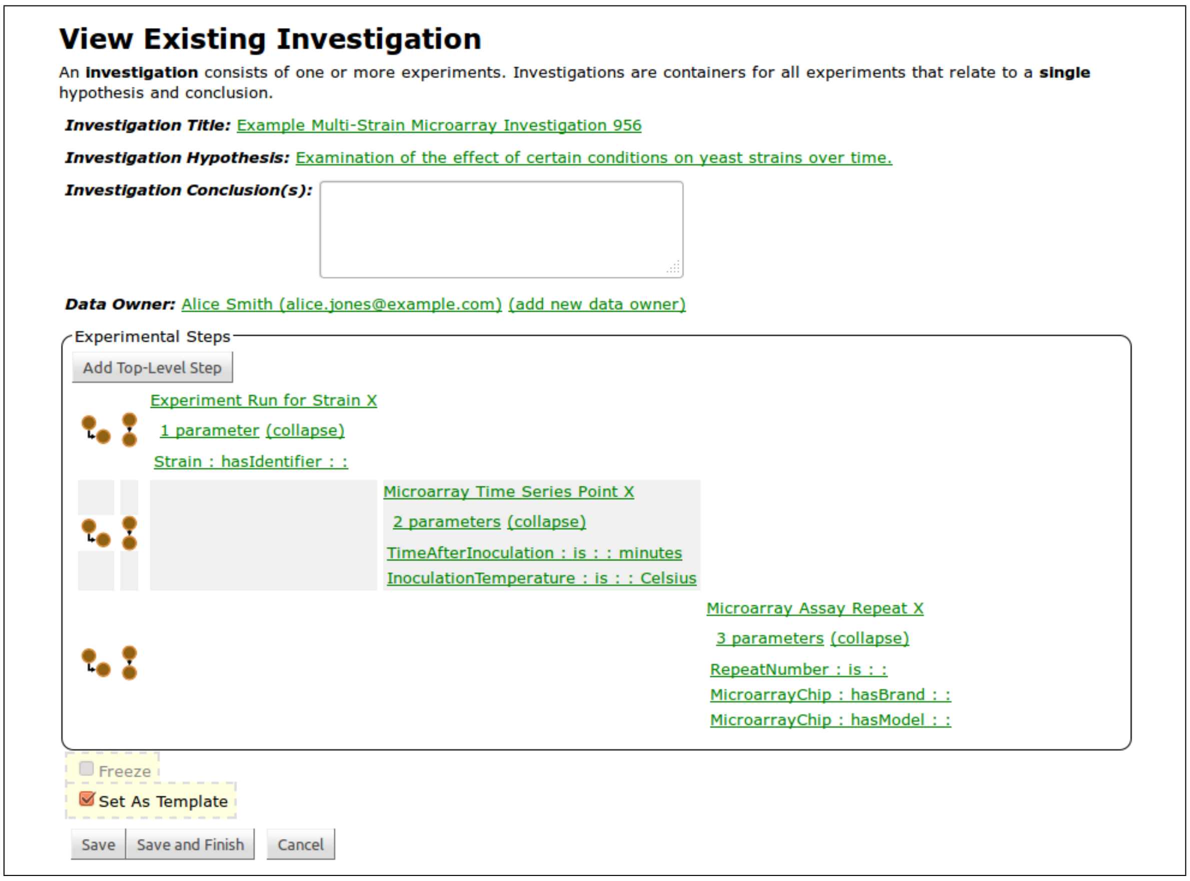
2.4.3 Templates and completed investigations

If a user wishes to save a series of experimental steps for anyone to copy and use, a template can be created. There are a number of benefits to the use of templates.

- *Content standards.* SyMBA templates can be built to community standards such as those present in the MIBBI registry.
- *Conformance.* If there are standard protocols for certain types of experiments within a research group, each protocol can get their own template in SyMBA. As each protocol is realised in a particular experiment, users can copy the template and fill in specific experimental details.
- *Reuse.* Templates make it easier to add information to experiments which are repeated multiple times. Rather than creating a completely new experiment description and introducing potential errors due to mistakes and missing information a template, for instance based on a completed description, can be used as many times as required to ensure the same information is filled in each time.

A template can either be begun with a blank investigation or based on an existing investigation. Once a set of experimental steps has been created, saving the investigation as a template simply requires the selection of the "Set As Template" checkbox, as shown in Figure 2.8. If an existing investigation is to be converted to a template a copy should be made of the investigation first. As the conversion removes all links to files and marks the investigation as read-only the copy, rather than the original, should be the one converted to a template. All information contained within each experimental step other than the file links will be retained in the template. The retained information includes parameters, descriptions, names, input materials and output materials. For parameters, the name of the parameter and the relationship the parameter name has with its value will be marked as read-only. Therefore, when a user starts working with an investigation based on that template wishes to fill in that parameter, they must retain the name and relationship of each templated parameter already provided. More information on parameters is available in the next section.

If the description of an investigation within SyMBA has been completed and all parameters and associated metadata have been filled in, the investigation can be frozen, which marks it as completed and prevents further modification. Freezing an investigation is achieved by selecting the "Freeze" checkbox visible in Figure 2.8. Italics are used to mark read-only investigations in the summary of experiments as shown in Figure 2.7.


View Existing Investigation

An **investigation** consists of one or more experiments. Investigations are containers for all experiments that relate to a **single** hypothesis and conclusion.

Investigation Title: Example Multi-Strain Microarray Investigation 956

Investigation Hypothesis: Examination of the effect of certain conditions on yeast strains over time.

Investigation Conclusion(s):

Data Owner: Alice Smith (alice.jones@example.com) (add new data owner)

Experimental Steps

Add Top-Level Step

Experiment Run for Strain X
1 parameter (collapse)
Strain : hasIdentifier : :

Microarray Time Series Point X
2 parameters (collapse)
TimeAfterInoculation : is : : minutes
InoculationTemperature : is : : Celsius

Microarray Assay Repeat X
3 parameters (collapse)
RepeatNumber : is : :
MicroarrayChip : hasBrand : :
MicroarrayChip : hasModel : :

☐ Freeze
☑ Set As Template

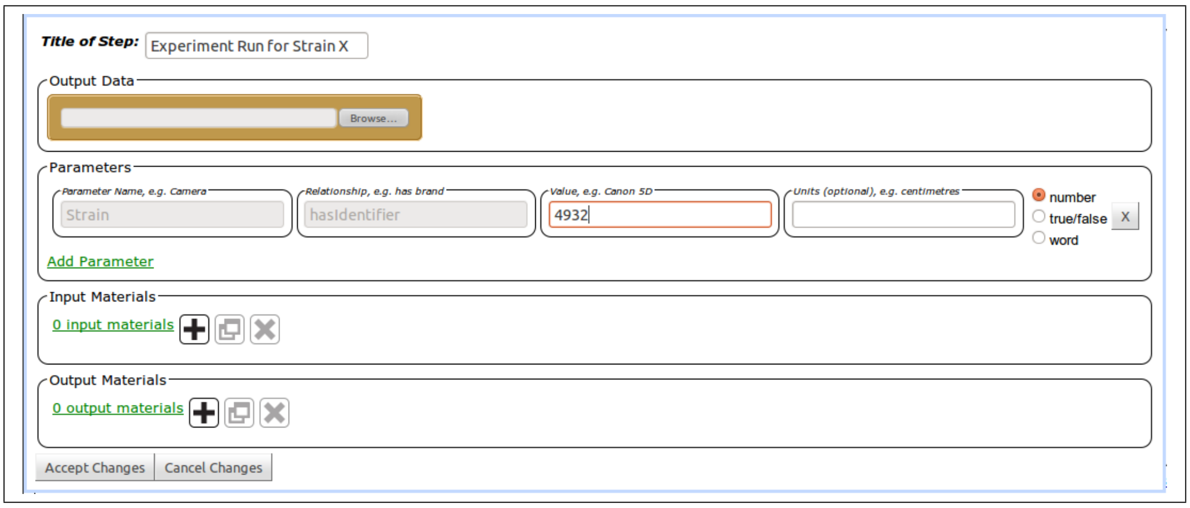
Save | Save and Finish | Cancel

Figure 2.8: The “View Existing Investigation” screen in SyMBA. This investigation has had some experimental steps and parameters added in preparation for saving it as a template. The user has selected the “Set As Template” checkbox. Clicking on *Save* or *Save and Finish* will result in this investigation being saved as a new template within SyMBA.

2.4.4 Storing experimental metadata and data

When a user is ready to start entering metadata and uploading data files, an existing template can be copied or a brand new experiment description can be created as described in Section 2.4.2. The act of copying or creation brings the user to the detailed investigation view where SyMBA users can create a new experiment, fill in information about the experiment and upload associated raw data files.

Figure 2.8 shows an overview of the investigation detail page. Each experimental step can have zero or more parameters, input materials and output materials. Parameters are structured according to a statement of three required and one optional attributes, as shown in Figure 2.9. These attributes are structured as *Name-Relationship-Value-Unit*, with the *Unit* being optional. In templates the *Value* can also be missing, but in standard investigations the *Value* attribute is mandatory. At any time, parameter statements can be deleted. While the *Name-Relationship-Value* portion of the parameter statement is similar to the *Subject-Predicate-Object* of RDF triples, RDF is not used within SyMBA as all metadata can be captured within the existing FuGE structure.


Title of Step: Experiment Run for Strain X

Output Data

Browse...

Parameters

Parameter Name, e.g. Camera	Relationship, e.g. has brand	Value, e.g. Canon 5D	Units (optional), e.g. centimetres	
Strain	hasIdentifier	4932		[✓ number] true/false word X

Add Parameter

Input Materials

0 input materials

Output Materials

0 output materials

Accept Changes Cancel Changes

Figure 2.9: Pop-up window allowing user input of parameters and materials.

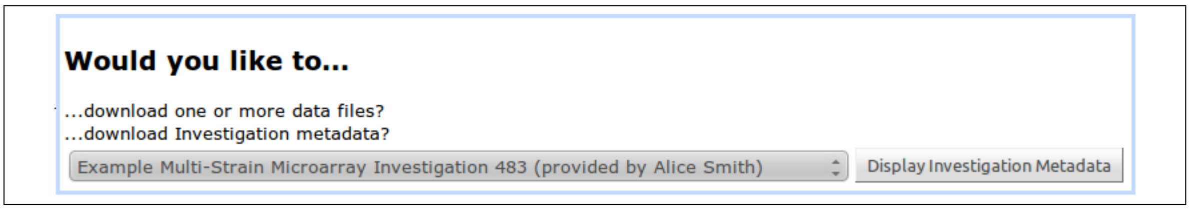
FuGE parameters have three types: number, boolean and string literal. These types can be manually specified by the user by selecting the appropriate radio button at the end of the parameter input line, as shown in Figure 2.9. However, these parameter types are relatively easy to programmatically ascertain, and once a *Value* is provided SyMBA will automatically select the radio button corresponding to its best guess. This selection can be overridden by the user at any time.

Parameters are highly configurable and can be built to describe virtually any statements required by the user. Concentrations of reagents, types of equipment, taxonomic information and repeat numbers are just some of the vital pieces of experimental information which can be modelled with parameters. In theory, materials could also be stored as parameters, but adding materials separately allows sets of commonly-used materials to be created and reused across investigations.

The name and relationship attributes within parameters are suitable candidates for restriction using ontology terms. Although this functionality was partly implemented in SyMBA version 1.0, it has not yet been added to version 2.0 as the best display of this information has not yet been determined. For instance, OBI could be used for equipment and protocol names, while RO could be used for some of the relationship attributes. In future, equipment will be organised as materials are now, but are currently modelled using parameters.

2.4.5 Retrieving FuGE-ML

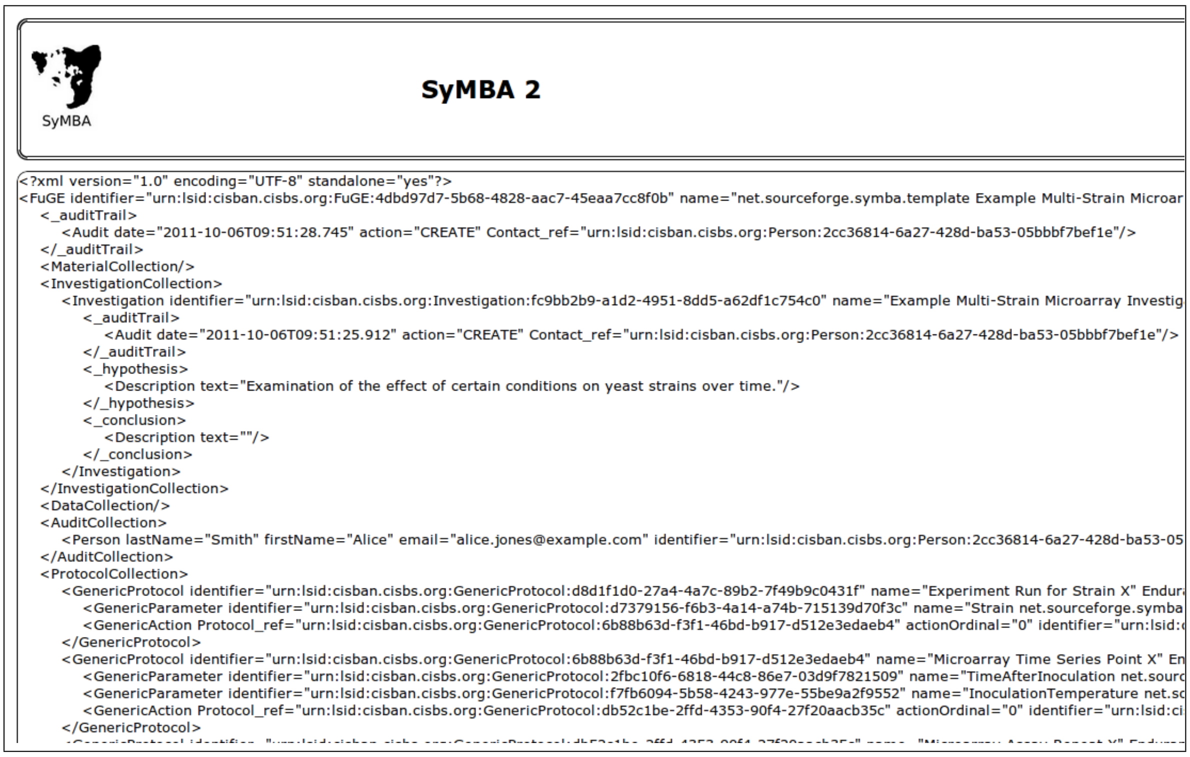
SyMBA uses and manipulates Java objects built from the FuGE XSD, converting upon request between those objects and FuGE-ML for download and batch upload. The *get* button is used to view and download the metadata for a given investigation in FuGE-ML. Figure 2.10 shows the pop-up menu which appears when the *get* button is pressed, while Figure 2.11 shows a partial screenshot of the resulting FuGE-ML.


Would you like to...

...download one or more data files?
...download Investigation metadata?

Example Multi-Strain Microarray Investigation 483 (provided by Alice Smith)
Display Investigation Metadata

Figure 2.10: The pop-up download menu which appears as a result of pressing the get button in the north panel of SyMBA.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FuGE identifier="urn:lsid:cisban.cisbs.org:FuGE:4dbd97d7-5b68-4828-aac7-45eaa7cc8f0b" name="net.sourceforge.symba.template Example Multi-Strain Microar
  <_auditTrail>
    <Audit date="2011-10-06T09:51:28.745" action="CREATE" Contact_ref="urn:lsid:cisban.cisbs.org:Person:2cc36814-6a27-428d-ba53-05bbbf7bef1e"/>
  </_auditTrail>
  <MaterialCollection/>
  <InvestigationCollection>
    <Investigation identifier="urn:lsid:cisban.cisbs.org:Investigation:fc9bb2b9-a1d2-4951-8dd5-a62df1c754c0" name="Example Multi-Strain Microarray Investig
      <_auditTrail>
        <Audit date="2011-10-06T09:51:25.912" action="CREATE" Contact_ref="urn:lsid:cisban.cisbs.org:Person:2cc36814-6a27-428d-ba53-05bbbf7bef1e"/>
      </_auditTrail>
      <_hypothesis>
        <Description text="Examination of the effect of certain conditions on yeast strains over time."/>
      </_hypothesis>
      <_conclusion>
        <Description text=""/>
      </_conclusion>
    </Investigation>
  </InvestigationCollection>
  <DataCollection/>
  <AuditCollection>
    <Person lastName="Smith" firstName="Alice" email="alice.jones@example.com" identifier="urn:lsid:cisban.cisbs.org:Person:2cc36814-6a27-428d-ba53-05
  </AuditCollection>
  <ProtocolCollection>
    <GenericProtocol identifier="urn:lsid:cisban.cisbs.org:GenericProtocol:d8d1f1d0-27a4-4a7c-89b2-7f49b9c0431f" name="Experiment Run for Strain X" Endur
      <GenericParameter identifier="urn:lsid:cisban.cisbs.org:GenericProtocol:d7379156-f6b3-4a14-a74b-715139d70f3c" name="Strain net.sourceforge.symba
      <GenericAction Protocol_ref="urn:lsid:cisban.cisbs.org:GenericProtocol:6b88b63d-f3f1-46bd-b917-d512e3edaeb4" actionOrdinal="0" identifier="urn:lsid:
    </GenericProtocol>
    <GenericProtocol identifier="urn:lsid:cisban.cisbs.org:GenericProtocol:6b88b63d-f3f1-46bd-b917-d512e3edaeb4" name="Microarray Time Series Point X" En
      <GenericParameter identifier="urn:lsid:cisban.cisbs.org:GenericProtocol:2fbc10f6-6818-44c8-86e7-03d9f7821509" name="TimeAfterInoculation net.sourc
      <GenericParameter identifier="urn:lsid:cisban.cisbs.org:GenericProtocol:f7fb6094-5b58-4243-977e-55be9a2f9552" name="InoculationTemperature net.sc
      <GenericAction Protocol_ref="urn:lsid:cisban.cisbs.org:GenericProtocol:db52c1be-2ffd-4353-90f4-27f20aacb35c" actionOrdinal="0" identifier="urn:lsid:ci
    </GenericProtocol>
```

Figure 2.11: Example FuGE-ML retrieved when an investigation is selected for retrieval in SyMBA.

2.4.6 SyMBA Web Services

Access to SyMBA is available via a Web interface and through the SyMBA Web services. Web services provide straightforward access to applications via the Web[17]. SyMBA makes use of Crossfire[18] via a Maven 2 plugin to package Web services into a Web archive which is then loaded into a Tomcat server.

Currently, SyMBA has three services containing a total of four methods. All of these services relate to the LSIDs contained within SyMBA:

[17]For more information see http://www.w3schools.com/webservices/ws_intro.asp
[18]http://cxf.apache.org/

- The *LsidAssigner* service contains one method, `assignLsid()`. This method creates a new LSID based on the default values within the application.
- The *LsidResolver* service contains one method, `getAvailableServices()`. This method checks that the available Web services can deal with the provided LSID.
- The *LsidDataRetriever* service provides the `getMetadata()` and `getData()` methods from the LSID specification. The former retrieves the LSID of the latest Identifiable object associated with the provided LSID. The latter returns the appropriate FuGE-ML or data file associated with the provided LSID. This may be a full document or XML snippet, depending on the LSID. A `RuntimeException` is returned if the LSID is not stored within SyMBA. LSIDs based on Endurants will also return a `RuntimeException`, as there is no way to be sure which version of the corresponding Identifiable objects have been requested.

The LSID Web services currently implement only those parts of the specification[19] in active use within SyMBA. In particular, these Web services implement LSIDs as Java strings rather than as complex objects. Table 2.1 shows how the LSID specifications' `getMetadata()` and `getData()` methods are handled. To retrieve FuGE metadata for a given LSID, call the `getData()` method in the LSID Data Retrieval Web service. To retrieve the data file itself, the getData() method is passed the LSID that points to the file of interest. Only those methods from the LSID specification that are useful within SyMBA have been implemented as it is unclear how complete the long-term uptake of LSIDs will be, and it is important to keep the SyMBA implementation as straightforward as possible.

	`getMetadata()`	`getData()`
Endurant	LSID of latest Identifiable	empty array of bytes
Identifiable	LSID of latest version	FuGE-ML for that object
Data File Identifiable	LSID of latest holding investigation	original file + SyMBA identifier
Timestamped	LSID of latest version	FuGE-ML as at the stated time

Table 2.1: Each column describes a different LSID-specified method, and the rows display the behaviour of each method when that object type is passed as a parameter. An Endurant does not point to a specific object, and therefore will never return anything in the getData() call, though it will return the LSID of the latest Identifiable associated with it via `getMetadata()`. Any time-stamped LSID, whether Endurant or Identifiable, will always return the FuGE-ML that was present at the requested time.

[19] `http://lsids.sourceforge.net/quick-links/lsid-spec/`

2.5 Discussion

With the development and application of high-throughput methodologies and experimentation in the life sciences, researchers are seeing what is often referred to as the life-science data deluge [192, 82]. To cope with the data deluge and preserve and discover the knowledge contained within it, there needs to be support for both public centralised archiving, such as that provided by the European Bioinformatics Institute (EBI)[20] and the National Center for Biotechnology Information[21], and smaller-scale local archiving such as that possible with a local SyMBA installation.

Data and metadata archiving that utilises content, syntax and semantic standards facilitates data longevity, provenance and interchange. SyMBA has been designed to meet this requirement for research groups and is capable of archiving numerous types of life-science data using FuGE, a community developed and endorsed data standard. SyMBA provides a straightforward capture method for essential experimental metadata and a safe location to store original, unmodified data. Metadata capture is centralised, enabling users to access and view their data in a uniform fashion. SyMBA is the first complete implementation of FuGE Version 1.0 from database back-end through to Web front-end, and includes a full library of code to manipulate and export/import information from FuGE-ML input files and the FuGE-ML-based database. Other applications such as the ISA Software Suite have been built around sister standards such as ISA-TAB.

Within Centre for Integrated Systems Biology of Ageing and Nutrition (CISBAN), SyMBA was the official archiving platform, and is used as one part of a larger data integration infrastructure. CISBAN has generated metadata and data from approximately 400 data files requiring 300 Gb of space, and covering high-throughput and large-scale transcriptomics, proteomics and time-lapse, multi-dimension, live-cell imaging experiments. SyMBA 1.0 was designed to be a required archival step in the data processing pipeline for all wet lab researchers within CISBAN. However, while this initial implementation had a back end database and support services which were suitable for the task, users found the front end difficult to manage. In practice, each researcher needed the help of a bioinformatician to work through the front end metadata deposition forms. This resulted in low usage of the services, and requests for refinements and improvements to the front end. These changes resulted in SyMBA 2.0, which was primarily a front end user interface change, although streamlining was also performed for non-user-facing parts of the architecture such as the persistence layer. However, the upgrade to SyMBA 2.0 was ultimately never put into production, as CISBAN reached the end of the lifetime of the grant. It remains a useful tool for interested research groups to

[20] http://www.ebi.ac.uk
[21] http://www.ncbi.nlm.nih.gov/

download and install, allowing flexible metadata entry as well as storage and export of that metadata in a published, standardised format.

In the spirit of the community standards that SyMBA can accommodate, the SyMBA code base is open source. New users and developers are encouraged to interact and contribute to the SyMBA system in an open and collaborative manner, to suit their specific needs. A wider application and contribution to SyMBA will benefit and help solve standards based data management issues in the life-sciences. These features make SyMBA an ideal infrastructural component for enabling interoperability, integration and analysis across many different types of experimental data, and for supporting data management and knowledge discovery throughout the research life cycle. SyMBA aids data integration by providing a common format for multi-omics experimental metadata, minimising format challenges as described in Section 1.6.1.

SyMBA provides a browsable history of experimental data and metadata while allowing users to compare aspects of studies such as controls and treatments. The unified and simple interface for data capture makes it easier for researchers to archive and store important information about their experiment. Easy copying of the structure and metadata from one investigation to another improves repeatability of experiments. SyMBA also provides improved metadata organization through the use of a nascent standard (FuGE), making preparations for publication and data interchange simpler. Additionally, SyMBA provides a computationally amenable starting point for data analysis pipelines via SyMBA's ability to store disparate data outputs of laboratories in a single repository with a single metadata format.

Chapter 3

Saint: a Web application for model annotation utilising syntactic integration

3.1 Introduction

Quantitative modelling is at the heart of systems biology. Model description languages such as SBML [20] and CellML [44] allow the relationships between biological entities to be captured and the dynamics of these interactions to be described mathematically. Without a biological context to these dynamic models, they are only easily understandable by their creators. Interested third parties must rely on extra documentation such as publications or possibly-incomplete descriptions of species and reactions included in an element's name or identifier to determine the relevance of a model to their work. Many dynamic models include only the mathematical information required to run simulations, and do not explicitly contain the full biological context. However, the efficient exchange, reuse and integration of models is aided by the presence of biological information in the model. If biological information about a model is added consistently and thoroughly, the model becomes useful not just for simulation, but also as an input in other computational tasks and as a reference for researchers.

Saint, an automated annotation integration environment, aids the modeller and reduces development time by providing extra information about a model in an easy-to-use interface [47]. Saint is a lightweight Web application which enables modellers to rapidly mark up models with biological information derived from a range of data sources. It supports the addition of basic annotation to SBML and CellML entities and identifies new reactions which may be valuable for extending a model. While the addition of biological annotation does not modify the behaviour of the model, the incorporation of new reactions or species adds new features that can later be built upon to potentially change the model's output. Saint is a syntactic integration solution to the annotation of systems biology models, developed both to explore the data sources relevant to modellers and to provide a practical examination of data integration for the purpose of model annotation.

Section 3.2 describes the importance of linking biology to models, and why the syntax of those links is vital to their usefulness. Section 3.3 describes the technology used to implement Saint, while Section 3.4 focuses on how Saint retrieves model annotation and how it can be used to pick and choose the annotation to keep. Section 3.5 provides practical examples of annotating models using Saint. Finally, Section 3.6 describes the future plans for Saint, the collaborations that have been developed, and how Saint can be used with rule-based mediation (see Chapter 5).

3.1.1 Availability

Saint is freely available for use on the Web at http://saint.ncl.ac.uk. The Web application is implemented in GWT, with all major browsers supported. Saint has a Sourceforge.net project website[1] and wiki[2]. There is also a Newcastle University project page[3]. The Sourceforge.net project website provides access to the Subversion repository[4] as well as various documents for programmers. The Newcastle University website and the Sourceforge.net wiki are both user-oriented, and contain general information about the project, associated publications, screenshots and instructions for use.

Saint is licensed by Allyson Lister, Morgan Taschuk and Newcastle University under the LGPL[5]. For more information, see LICENSE.txt in the Saint Subversion project. Installing Saint on a Web server requires an installation of LibSBML [193] and the CellML API [59]. Installation and running of Saint has been tested on 32-bit Ubuntu 8.10 and higher.

3.2 Linking biology to models

Figure 3.1 is one of a set of examples provided by SBGN to demonstrate the standard. Specifically, it is a diagrammatic representation of glycolysis. There are two layers to systems biology models such as this one: the mathematical model, where a term like “hexokinase” is the name of a variable in a model to be simulated; and the real, underlying biological system where hexokinase is a molecule inside a biological compartment which physically interacts with other entities. The mathematical model is self contained, and will work perfectly without information regarding hexokinase’s biological importance. However, mathematical models are created to aid the understanding of biology therefore links to the biology of the model should be created and maintained. In systems biology models, these links are added via annotations.

Model annotations are necessary to describe how the model has been generated and to define the meaning of the components that make up the model in a computationally-accessible fashion. The addition of biological annotations to a model is usually a manual, time-consuming process; there is no single resource encompassing all suitable data sources. A modeller usually has to visit many websites, applications and interfaces in order to identify relevant information, and may not be aware

[1] http://saint-annotate.sourceforge.net
[2] http://sourceforge.net/apps/mediawiki/saint-annotate/index.php?title=Main_Page
[3] http://cisban-silico.cs.ncl.ac.uk/saint.html
[4] http://saint-annotate.svn.sourceforge.net/viewvc/saint-annotate/
[5] http://www.gnu.org/copyleft/lesser.html

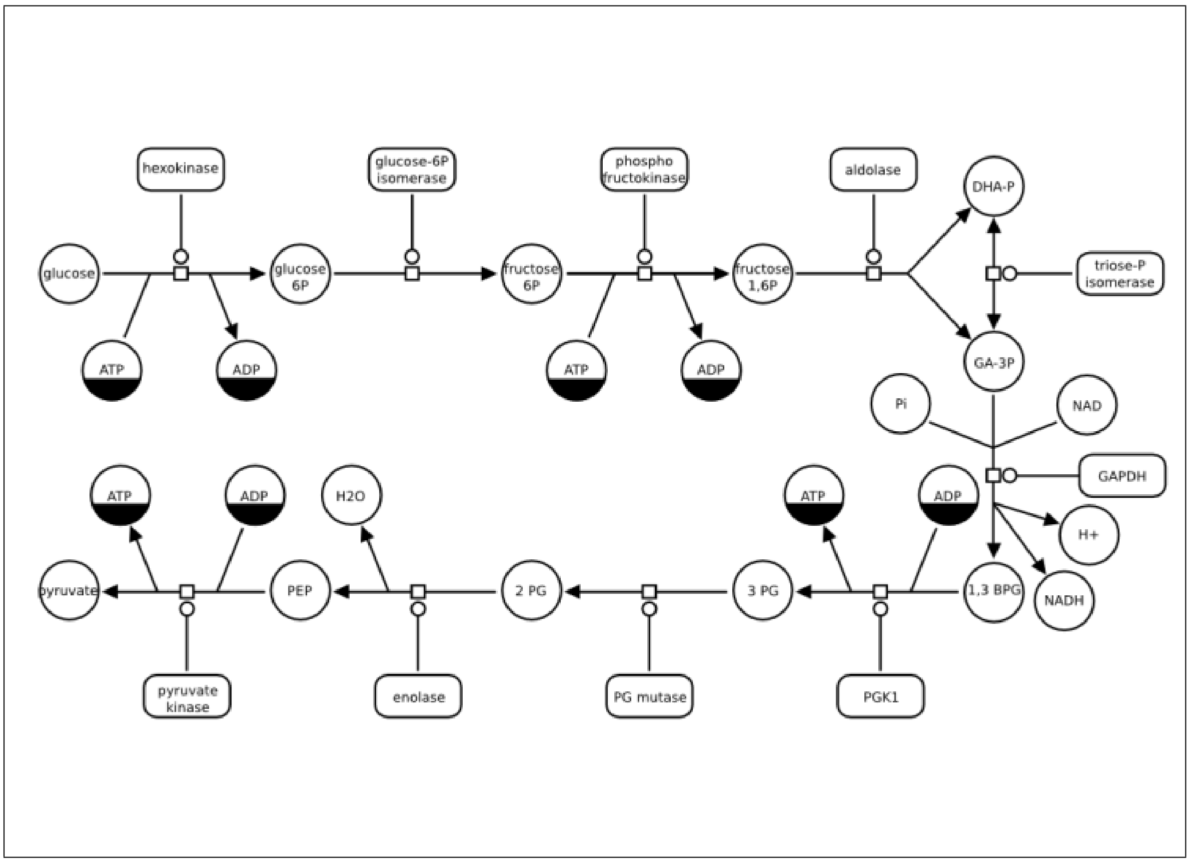


Figure 3.1: Diagram of glycolysis using the SBGN Process Description language. Taken from http://www.sbgn.org/Documents/PD_L1_Examples.

of all potentially useful databases. Because modellers add information manually, it is very difficult to annotate exhaustively.

However, without computationally-amenable biological annotations, it is difficult to create software tools that determine the biological pathway or reaction being modelled. The manual curation of such semantic information to non-curated models is a complex process. For instance, the advice for the curation of new BioModels database entries includes checks that all entities are what they say they are (e.g. that a species is a species and not a parameter), the creation of meaningful names for SBML entities and the association of the model with at least one publication[6]. For many submitted models, the quality of the biological annotation is low, making it difficult to programmatically determine what biological pathway or pathways a model is describing.

While annotations are vital for model sharing and reuse, they do not contribute to the mathematical content of a model and are not critical to its successful functioning. The addition of biological knowledge must be performed quickly and easily in order to make annotation worthwhile to a mod-

[6]http://biomodels.caltech.edu/curationtips

eller. A large number of tools[7] are available for the construction, manipulation and simulation of models, but there is currently a limited number of tools to facilitate rapid and systematic model annotation. While websites and applications specialising in integrating disparate data sources exist such as BioMart [194] and Pathway Commons, few are designed to put information directly into a model.

SBML and CellML can store both the mathematical model and the links to the biological information necessary to interpret the model. The difficulty lies in retrieving appropriate biological information quickly and with a minimal amount of effort. In the glycolysis example, UniProtKB [1] states that hexokinase is a protein with an enzymatic role in glycolysis[8]. STRING [195] and Pathway Commons provide a list of interactions and pathways in which hexokinase is involved. MIRIAM, SBO [86] and GO [18] provide Saint with the vocabulary needed to label hexokinase and describe its role in the model.

MIRIAM annotations use vocabularies and ontologies to document the biological knowledge in a systems biology model. In SBML, biological annotation is added to annotation elements according to the MIRIAM specification [19]. MIRIAM annotations link external resources such as ontologies and data sources to the model in a standardised way. MIRIAM also defines an annotation scheme, accessible via Web services, which specifies the format and set of standard data types which should be used for these URIs [46]. The use of the MIRIAM format provides a standard structure for explicit links between the mathematical and biological aspects of an SBML model.

3.3 Implementation

3.3.1 Integration methodology

Saint makes it easy to find information related to the domain of interest and save that information back to SBML. The biological annotation of models is facilitated by using query translation to present an integrated view of data sources and suggested ontological terms. As described in Section 1.6.4.2, query translation uses a mediator to distribute a single query across known data sources. When the results are retrieved, they are formatted according to the set of global Saint objects and sent to the user interface.

Within Saint, the query for each species is translated into a set of queries over Web services provided by each data source. The results are imported into global objects, which can then be further ma-

[7]http://sbml.org/SBML_Software_Guide
[8]http://www.uniprot.org/uniprot/P19367

nipulated before presentation: querying and integration occur together. This method is suitable for a single-purpose application that runs pre-specified queries and then stores the query results generically. The combined query results are then displayed in the Web browser.

Query translation removes the need for a local database of information, and allows the easy addition of new data sources. However, changes to the source Web services or data structures require equivalent changes in the Saint query system. As changes to services are relatively infrequent and the size of the available datasets very large, query translation is the best option for this type of integration.

Saint performs syntactic integration by matching data to a particular identifier, name or MIRIAM URI through syntactic equivalence of the query term and the external data source. No semantic integration occurs with the standard version of Saint, although Section 3.6 discusses how Saint can be connected to the rule-based mediation knowledgebase, thus adding a semantically-aware data source.

3.3.2 Saint back-end

On the client side, Saint is a simple Web application which can be used immediately, without creating an account or requesting access. Behind the user interface, Saint is implemented in GWT[9] and hosted on a Tomcat[10] server. For more information on GWT, see Section 2.3.2. GWT was chosen for a number of reasons:

- *Asynchronous data transfer*. Remote procedure calls can be used to query the server-side of the GWT application. As soon as the query is dispatched, control is handed back to the client. A notification is sent back to the client once the query to the server completes. This allows for multiple calls to be performed simultaneously and asynchronously. For instance, new annotation can be retrieved for many SBML species at the same time.
- *Abstraction of the details of creating a user interface*. Although the programmer writes code describing the look and feel of the panels and forms, the specific details of the user interface are hidden. The default skin of GWT Web applications also creates a clean and lightweight feel with very little cost in development time.
- *Speed of development*. GWT only requires a knowledge of standard packages and libraries in Java, so there is a low initial overhead for creating a working Web application. The majority of bioinformaticians are not experts in the creation of user interfaces, and by using GWT more time can be spent on functionality rather than the user interface.

[9]http://code.google.com/webtoolkit/
[10]http://tomcat.apache.org/

3.4 Data retrieval and model annotation

This section describes both the structure of Saint and its annotation methodology using the curated BioModels entry BIOMD0000000087 [196] by Proctor and colleagues as an example, hereafter referred to as the *telomere model*. This model describes telomere uncapping in *Saccharomyces cerevisiae* (*S.cerevisiae*), a process implicated in ageing.

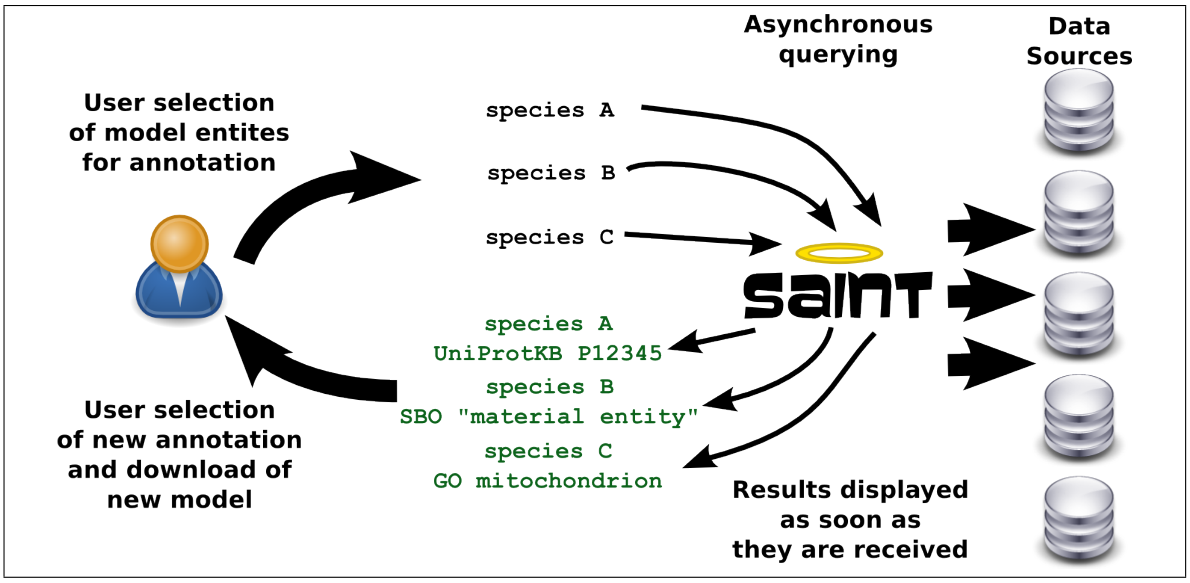

Asynchronous querying
Data Sources
User selection of model entites for annotation
species A
species B
species C
saint
species A
UniProtKB P12345
species B
SBO "material entity"
species C
GO mitochondrion
User selection of new annotation and download of new model
Results displayed as soon as they are received

Figure 3.2: An overview of how Saint queries remote data sources and retrieves new annotation. The image of the database cylinders is LGPL (Image source: http://commons.wikimedia.org/wiki/File:Crystal_Clear_app_database.png). The image of the user is licensed under the CC-BY-SA 2.5 (Image source: http://commons.wikimedia.org/wiki/File:User_icon_2.svg).

Figure 3.2 shows an overview of the Saint integration infrastructure. As soon as the user session begins, the full complement of official MIRIAM URIs are downloaded via EBI Web services. These official URIs are used within MIRIAM-compliant models to provide cross-references to many different data sources. By retrieving the URIs for each data source at the beginning of a session, Saint will remain up-to-date with respect to any changes made by MIRIAM developers.

A model is uploaded by copying either a CellML or SBML model into the input box and pressing the "Load Model" button. Saint then validates the model with LibSBML for SBML or the CellML API for CellML documents. If the model is valid the SBML species or the CellML variables are retrieved and displayed, together with any MIRIAM annotation already present. Obsolete MIRIAM annotations are updated, and Saint then uses a table to display the elements available for annotation (see Figure 3.3).

It is possible to restrict which types of annotation are retrieved by Saint. The "Advanced Annotation

Options" shown at the bottom of Figure 3.3 can be disabled as necessary prior to starting the annotation process. For instance, a modeller who has a skeleton model might be interested in adding all types of annotation, while one who has already defined the reactions of interest might want to deliberately exclude the addition of new reactions. A model that has already been submitted to a database such as BioModels will already have annotation for each species. However, a modeller might want to extend such a model, and therefore might only be interested in new reactions. Additionally, modellers might not add GO annotation to a species if the species is already annotated with a UniProtKB accession, as that information can be found in the UniProtKB entry itself. However, if the curated SBO term is quite high level, providing a GO term with a more specific description could be helpful.

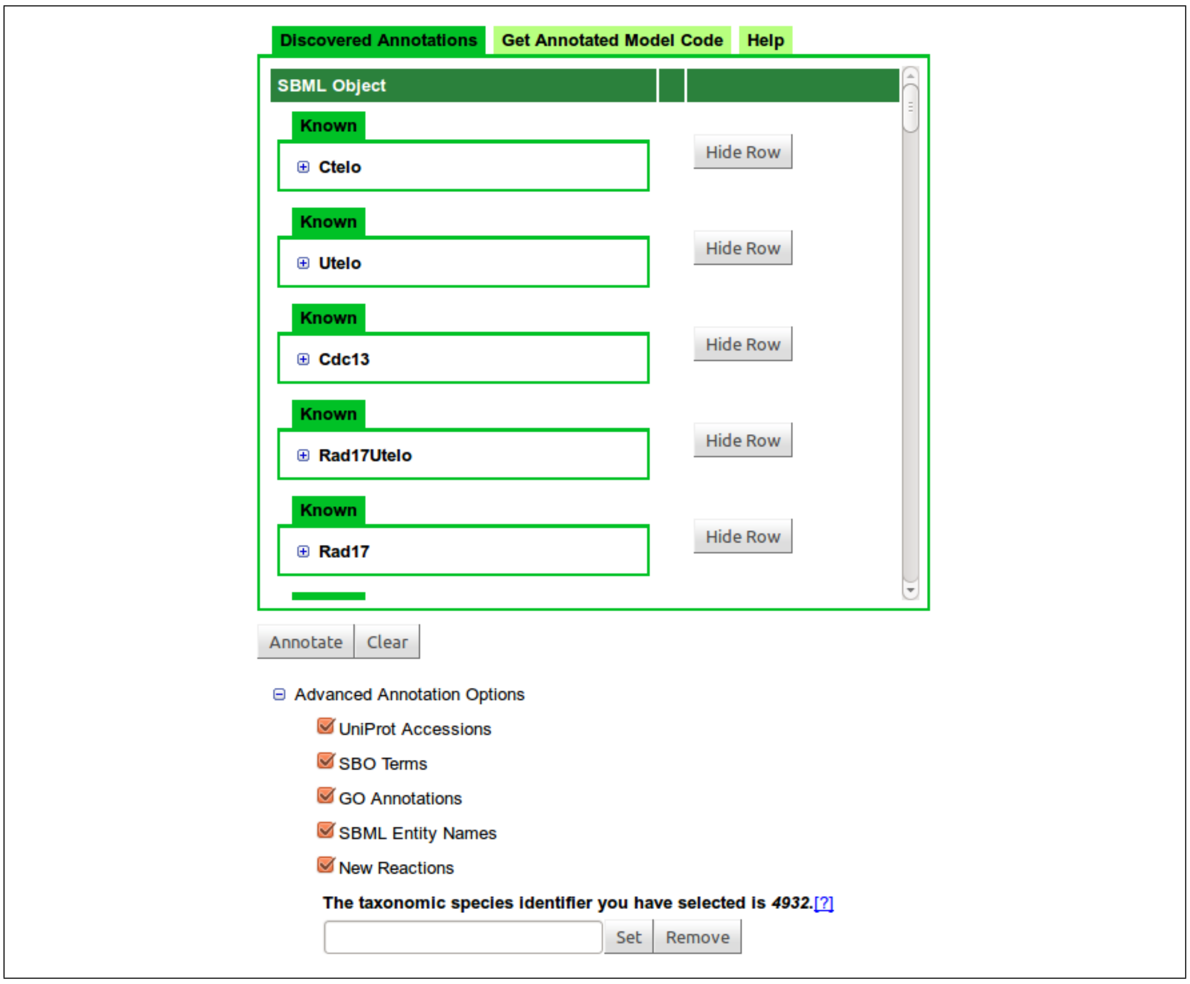
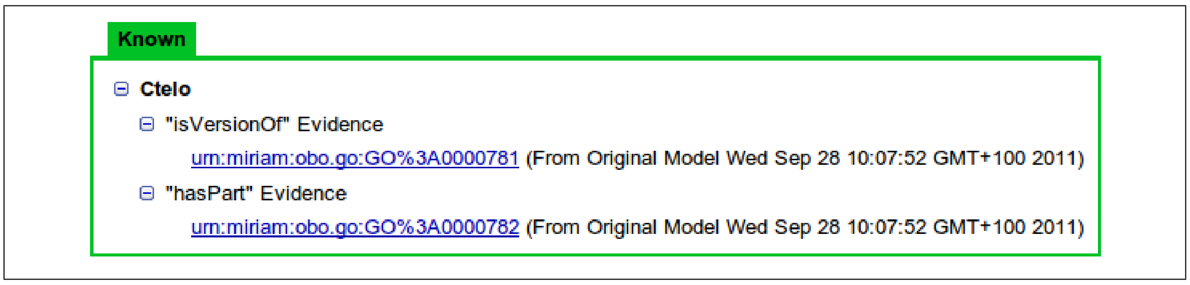


Figure 3.3: SBML species and CellML variables are loaded into a tabular structure, allowing the user to pick entities of interest and delete irrelevant entities. This screenshot shows how the telomere model is displayed.

Both SBML and CellML models can be read by Saint, although the CellML export is not yet implemented as the method used to store MIRIAM annotation within CellML has not yet been finalised. The user can then remove any items they are not interested in annotating. For instance, some terms such as “sink” are modelling artefacts and do not correspond to genes or proteins. Therefore, the user would normally wish to delete this species from the search space to prevent any possible matches with actual biological species of a similar name. Deletion of a row in the table is accomplished by clicking on the “Hide Row” button for the appropriate row. Figure 3.3 shows a portion of the main table in Saint after the telomere model has been loaded.

Each of the species from the telomere model is added as a new row to the table, although the screenshot does not show the entire list. Those that already contain annotation will have a collapsible list allowing the user to browse the pre-existing annotation. For instance, Figure 3.4 shows how the `Ctelo` species has two items of evidence, the GO terms *chromosome, telomeric region* (GO:0000781) and *telomere cap complex* (GO:0000782). At this stage, the only tab visible within this row is the “Known” tab containing the pre-existing items of evidence.


Known

⊟ Ctelo
⊟ "isVersionOf" Evidence
urn:miriam:obo.go:GO%3A0000781 (From Original Model Wed Sep 28 10:07:52 GMT+100 2011)
⊟ "hasPart" Evidence
urn:miriam:obo.go:GO%3A0000782 (From Original Model Wed Sep 28 10:07:52 GMT+100 2011)

Figure 3.4: The expanded display of the `Ctelo` species row in Saint, showing both the id of the species and its pre-existing MIRIAM annotation. This species corresponds to the first line of the table shown in Figure 3.3.

Once the user is satisfied with the list of items to be annotated, the model is submitted using the “Annotate” button at the bottom of the table. Pressing “Clear” will instead unload the current model and return the user to the Saint homepage. As annotation queries complete, summaries are added to the main table. Each row of the table is submitted for annotation separately, which allows the query response to be sent back to the user as soon as it is received, even if other rows have not yet returned information.

In the example, the focus is on the annotation of `Ctelo`, `Cdc13` and `Mec1`. After all other species are hidden from view, the “Annotate” button is pressed. Once annotation is added, a status message appears in the second column of the table, and an “Inferred” tab is added next to “Known” (see Figure 3.5). Clicking on the Inferred tab allows the user to view and modify the newly-suggested annotation. When hovering over a new item of annotation with the mouse, the evidence for that piece of annotation will appear. Information on each of the data sources where the annotation was found is displayed. The results can be customised by the user, who can examine the evidence and then discard any individual piece of inappropriate annotation. The new model is kept in sync with the choices being made, and can be exported at any time.

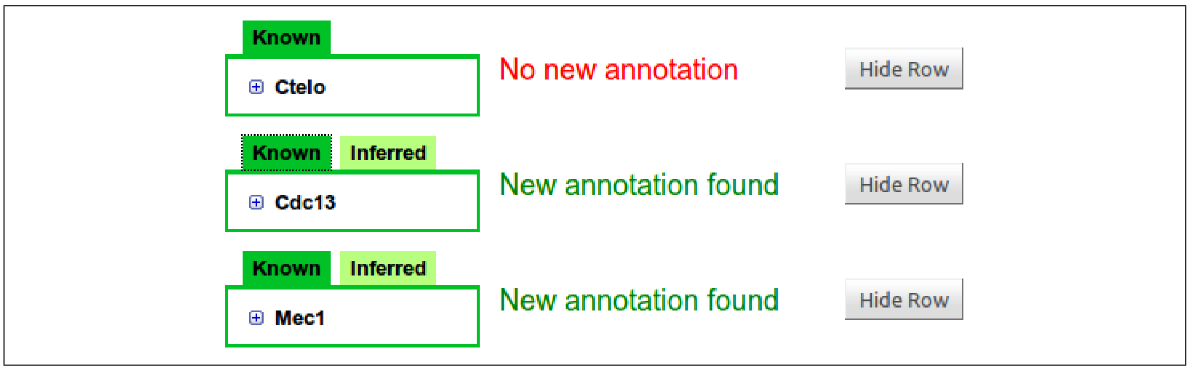
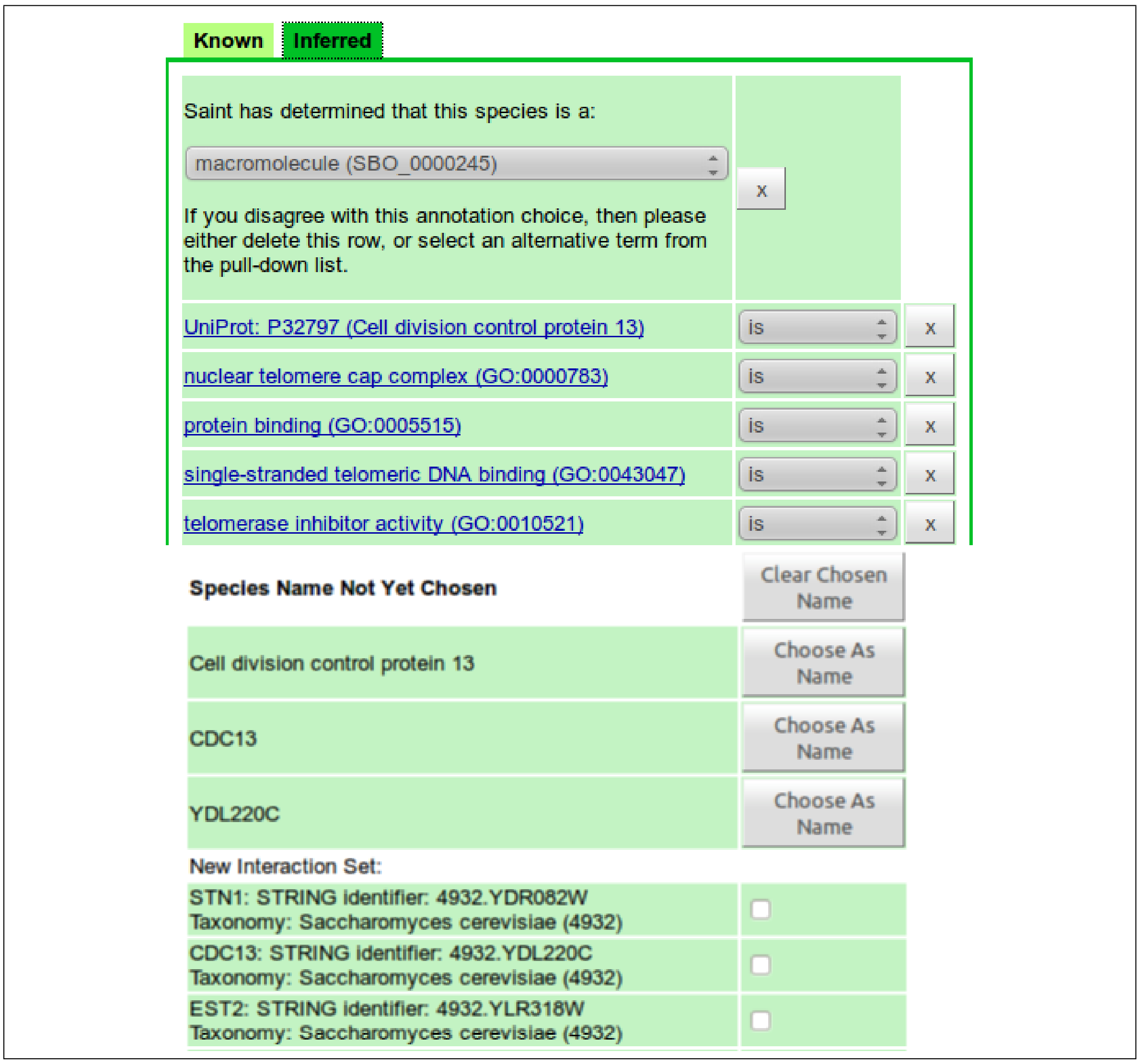


Figure 3.5: How Saint looks immediately after querying for information on the chosen species. No annotation is found for `Ctelo`, but annotation has been identified for the other two species. If new annotation is provided by Saint, a new “Inferred” tab containing the suggestions is made visible.

In systems biology models, separate species are often used to describe the various states of a single biological entity. For instance, in the telomere model the enzyme `Exo1` is represented by `Exo1A` for the active state and `Exo1I` for the inactive. For the modeller, it is useful to name multiple species in this way to allow for differing amounts of protein present in each state. Such naming schemes caused difficulties for Saint because the provided name or identifier does not match the canonical name.

Naming ambiguities have been partially solved by parsing commonly-used systematic naming schemes into more likely query candidates. Systematic naming schemes are conventions used by researchers for naming species in alternative states. If no results are retrieved for the original name or identifier, Saint removes any appended “_I”, “_A”, “active” or “inactive” literals and tries the query again. In some cases, species names or identifiers are created as a hybrid of two protein names, e.g. “proteinA_proteinB”. In such cases, Saint suggests annotation based around both proteins. Having a name attribute as well as the id attribute helps, as generally the name used by modellers is a more precise identification of the biological entity.

As shown in Figure 3.5, in the telomere model example Saint found new annotation for `Cdc13` and `Mec1` but not for `Ctelo`. Figures 3.6 and 3.7 show a selected subset of the suggested annotation for `Cdc13` and `Mec1`, respectively. `Ctelo` is the modeller’s private name for the capped state of a telomere, just as `Utelo` represents the uncapped telomere. As such, no hits are found in the source data. However, both `Cdc13` and `Mec1` are correctly identified, and the new annotation is displayed to the user. The first section of the results shows which ontology terms are suggested for the species. For SBO, a suggested term is provided as the default value in a pull-down list, allowing the user to either accept, change or delete the suggestion. Saint suggests the SBO term ‘macromolecule’ (SBO:0000245) as the best SBO match to a protein. This term was suggested both because ‘cdc13’


Known

Inferred

Saint has determined that this species is a:

macromolecule (SBO_0000245)

If you disagree with this annotation choice, then please either delete this row, or select an alternative term from the pull-down list.

UniProt: P32797 (Cell division control protein 13)	is	x
nuclear telomere cap complex (GO:0000783)	is	x
protein binding (GO:0005515)	is	x
single-stranded telomeric DNA binding (GO:0043047)	is	x
telomerase inhibitor activity (GO:0010521)	is	x

Species Name Not Yet Chosen

Cell division control protein 13	Choose As Name
CDC13	Choose As Name
YDL220C	Choose As Name

New Interaction Set:

STN1: STRING identifier: 4932.YDR082W Taxonomy: Saccharomyces cerevisiae (4932)	☐
CDC13: STRING identifier: 4932.YDL220C Taxonomy: Saccharomyces cerevisiae (4932)	☐
EST2: STRING identifier: 4932.YLR318W Taxonomy: Saccharomyces cerevisiae (4932)	☐

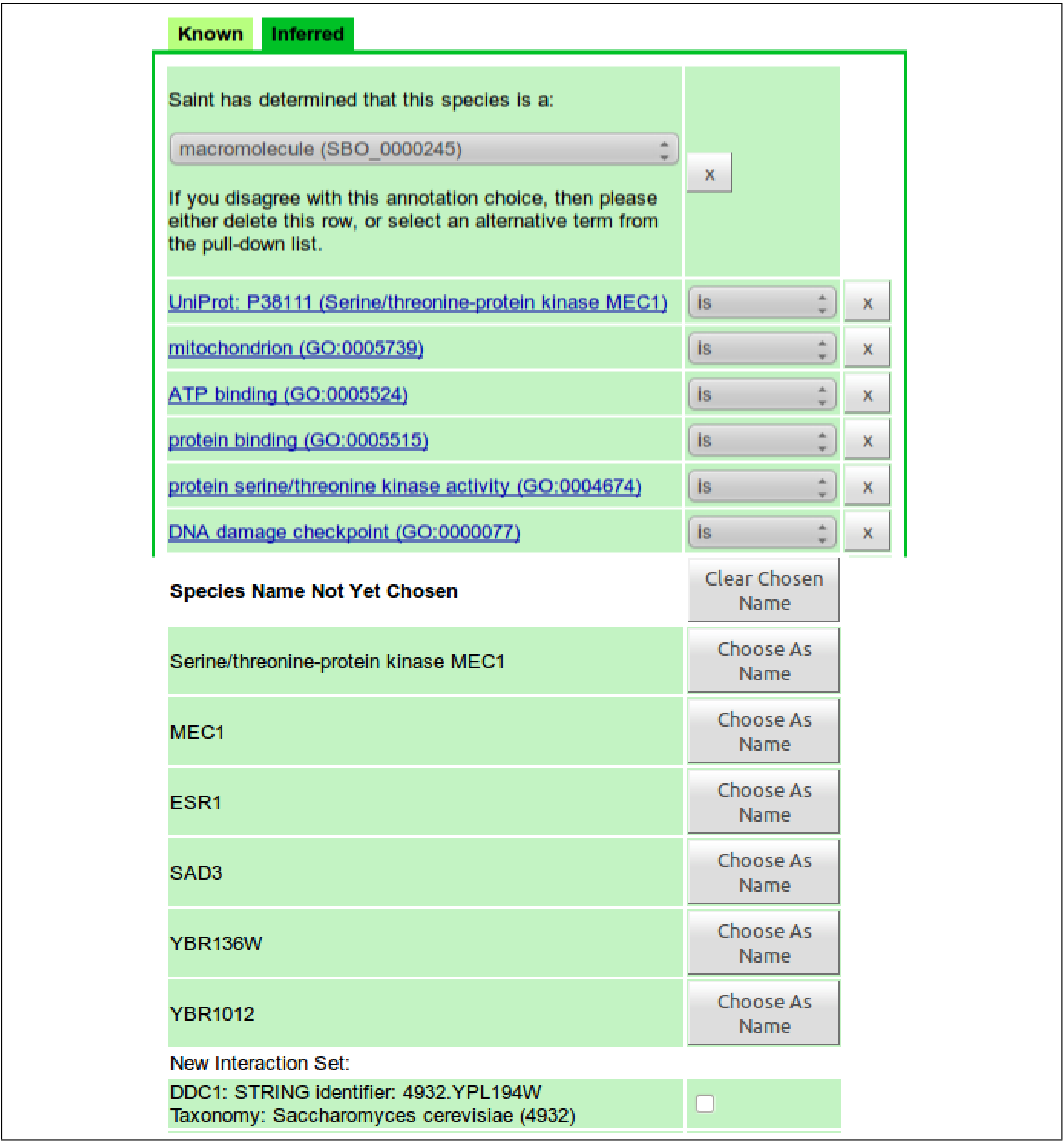
Figure 3.6: A selected subset of the annotation suggested by Saint for `Cdc13`. SBO and GO terms are available, and as the name attribute was missing, a variety of alternative names are listed. Finally, interactions are available to select and will be added as skeleton reactions to the new model.

was found within UniProtKB and because the interaction set identified the species as a protein.

Because multiple MIRIAM annotations such as GO and UniProtKB cross-references are allowed within the annotation element, each annotation has its own row and may be individually removed. Each of these rows also shows the relationships available to link the annotation to the model. While SBO terms are added directly to SBML elements, GO terms and UniProtKB accessions are added to the MIRIAM annotation section.

As well as annotations, extensions to the model are also suggested. Binary interactions retrieved from both Pathway Commons and STRING can be made into basic SBML reactions. Also, alternative names for the species or variables are presented to the user by parsing the description lines of all retrieved UniProtKB entries. For each row, the depth of the green background is dependent upon the number of sources of evidence; the greater the sources, the darker the green. Reactions and associated species are added directly to the SBML model, while the majority of the other annotation is added as MIRIAM-compliant URIs.

Once the new annotation has been approved by the user, the model can be saved as a new SBML document. Clicking on the "Get Annotated Model Code" tab in Saint adds all approved annotation and presents the new SBML model for viewing and download.



Saint has determined that this species is a:

macromolecule (SBO_0000245)

If you disagree with this annotation choice, then please either delete this row, or select an alternative term from the pull-down list.

UniProt: P38111 (Serine/threonine-protein kinase MEC1)	is	x
mitochondrion (GO:0005739)	is	x
ATP binding (GO:0005524)	is	x
protein binding (GO:0005515)	is	x
protein serine/threonine kinase activity (GO:0004674)	is	x
DNA damage checkpoint (GO:0000077)	is	x

Species Name Not Yet Chosen	Clear Chosen Name
Serine/threonine-protein kinase MEC1	Choose As Name
MEC1	Choose As Name
ESR1	Choose As Name
SAD3	Choose As Name
YBR136W	Choose As Name
YBR1012	Choose As Name

New Interaction Set:

DDC1: STRING identifier: 4932.YPL194W
Taxonomy: Saccharomyces cerevisiae (4932) ☐

Figure 3.7: A selected subset of the annotation suggested by Saint for `Mec1`. SBO and GO terms are available, and as the name attribute was missing, a variety of alternative names are listed. Finally, interactions are available to select and will be added as skeleton reactions to the new model.

3.5 Comparing Saint-annotated models with pre-existing annotation

Saint uses the name and identifier of an element as well as the existing resource URIs as starting points for adding annotation to models. Saint can provide new MIRIAM-compliant annotation with even a minimal amount of data, making it useful both for early-stage and already-annotated models. This section compares two Saint-annotated models against their original models as well as against stripped variants of those original models. In the first example, Saint annotates versions of the original model stripped of all biological annotation. The stripped model is then compared with the original model to asses the performance of Saint when presented with minimal information. In the second example, Saint annotates the original models. This comparison determines the extra annotation possible when Saint is provided with all available information.

3.5.1 Initial set-up of the comparisons

The models used for these comparisons are the telomere model as defined in Section 3.4, BIOMD0000000189 [197] and BIOMD0000000032 [198]. The telomere model describes telomere uncapping in *S.cerevisiae*. Proctor and Gray created BIOMD0000000189, hereafter referred to as the *p53-Mdm2 model*, to model oscillations and variability in the p53-Mdm2 system in humans. Finally, in BIOMD0000000032 Kofahl and Klipp model the dynamics of the pheromone pathway in *S.cerevisiae*. For readability, BIOMD0000000032 is hereafter referred to as the *pheromone model*.

As well as making use of the intact, original version of each model, stripped versions were created by removing their biological annotation. Specifically, after each model is downloaded from BioModels, the sboTerm and the annotation element of each species is removed. Even though SBO terms, species names and annotation elements are added by Saint in these comparisons, names and identifiers were deliberately not removed. Without these fields, Saint would lose those strings as query terms. Generally, modellers use either descriptive id attributes without names, or descriptive name attributes together with an arbitrary identifier scheme. Though *all* URIs within the annotation elements are stripped from the model, only a proportion of those URIs are currently supported by Saint and therefore only those supported types can be added back by Saint. These stripped models are then used as a starting point for Saint annotation for the first set of comparisons.

The reactions added by Saint are new and as such cannot be usefully compared against the original models. This functionality of Saint is therefore not compared. Additionally, species which describe a composite or complex formed from multiple proteins will not necessarily behave in the same way as its constituents. As such, only the SBO terms and UniProtKB cross-references suggested by Saint will be retained for comparison in those instances. An expert modeller would be able to correctly choose the appropriate name and GO terms, but for the purposes of this example they were simply removed from consideration.

To perform the comparison, the stripped models are loaded into Saint and the new annotation is retrieved. After the new annotation is displayed, basic editing of the annotation is performed, including the choosing of sensible species names from the list and the removal of obvious false positive annotation. These choices mimic the expected behaviour of the modeller using the application. Then the newly-annotated model is downloaded and compared to the original, intact model.

Two phrases are used throughout these comparisons: *useful species* and *Saint-supported URIs*. Species with informative id or name attributes in the original model are termed *useful species*. Informative identifiers and names closely match actual protein or gene names, and are therefore suitable for querying external resources. Sometimes, if an explicit statement must be made as to which of either the identifier or name is useful, the terms *useful identifier* or *useful name* will be used. There is a limit to how much information Saint can pull from non-informative identifiers and names. In these cases, it is important to use any MIRIAM information already stored in the annotation elements. This style of annotation is covered Section 3.5.3. *Saint-supported URIs* are MIRIAM URIs, such as those for GO and UniProtKB, which are parsed by Saint as well as added during the annotation process. Saint-supported URIs are therefore, by definition, those which can be compared both before and after new annotation has been added.

All original, stripped and re-annotated versions of the models used in this example are available from the Saint Subversion repository[11].

3.5.2 Annotating stripped models

The first comparison illustrates the benefits of searching based on the id or name attributes of an SBML species element. The stripped versions of the telomere model and the p53-Mdm2 model, both of which contained a number of useful species, were annotated by Saint and then compared against their originals. Even within a single model, some names and identifiers can be useful to Saint while others are not.

[11] http://saint-annotate.svn.sourceforge.net/viewvc/saint-annotate/trunk/docs/supplementary-data

3.5.2.1 Annotation of the stripped telomere model

Basic information on the annotation of both the original and re-annotated the telomere model is available in Table 3.1, together with a summary of the comparison between the two models. A detailed listing of the modifications made to the model is available in Table 3.2.

Species	55
Species with any annotation†	51/55
Useful species	15
Species with names	0
Species with SBO terms	0
Saint-supported URIs in useful species	14/15
Species with names after re-annotation	**14**
Species with SBO terms after re-annotation	**15**
URIs after re-annotation	**163**
URIs recovered from useful species	**14/14**

Table 3.1: Basic information on the annotation present in the original version of the telomere model and **in bold for the re-annotated version**. This model had no species name attributes, and therefore Saint relies on the id attribute. There were also no SBO terms in the original model. Although there are 15 useful species, only 14 contain either a Saint-supported URI, a name or an SBO term; the fifteenth (the "RPA" species) has a useful identifier but contains only a PIRSF URI unsupported by Saint. †*Any annotation* refers to MIRIAM URIs, names and SBO terms.

The total number of Saint-supported URIs has increased from 14 to 163, with further addition or subtraction of URIs possible if required by the modeller. In addition, Saint recovered all 14 of the URIs from the useful species. The original model contained no SBO terms or species names, and Saint has added 15 SBO Terms and 14 species names, which is almost complete coverage of the useful species.

The species "ssDNA" is an example of incorrect annotation by Saint due to a lack of available information. "ssDNA" is difficult for Saint to filter: it is not a protein, but can be used as a phrase within protein names. Currently, the Saint user must spot these errors and remove the suggested annotation using the Saint interface prior to downloading the model. The one useful species which was not completely annotated with SBO term, species name *and* URIs was "RPA". A large amount of annotation was suggested by Saint, but all except the SBO term was manually rejected. In the original annotation, the "RPA" annotation is a single URI which points to the PIR family (PIRSF) "replication protein A, RPA70 subunit"[12]. This URI is marked as having an "isVersionOf" relationship the RPA species, rather than as an exact "is" match, as the PIRSF URI is linked to mouse, and not yeast. A search for an equivalent for yeast in UniProtKB shows a number of yeast subunits. In

[12] `http://pir.georgetown.edu/cgi-bin/ipcSF?id=PIRSF002091`

theory, an appropriate choice would be made by the modeller. In PIRSF002091, the equivalent yeast UniProtKB accession is listed as P22336, which is " Replication factor A protein 1", a.k.a. "RFA1". This leads to the conclusion that the species in the model was named after the mouse protein and not the yeast protein. As "RFA" is not a name that matched the original ID ("RPA"), there is no way to retrieve the correct information from the data sources.

In conclusion, in comparing the annotation on the useful species for this model, all Saint-supported URIs were recovered, and new URIs, SBO terms and species names were added. A detailed listing of all new and recovered annotation present in the useful species are shown in Table 3.2.

SPECIES ID	MATCH	NEW
Cdc13	urn:miriam:uniprot:P32797	CDC13 SBO:0000245
Rad17	urn:miriam:uniprot:P48581	RAD17 SBO:0000245
Rad24	urn:miriam:uniprot:P32641	RAD24 SBO:0000245
RPA		SBO:0000245
Mec1	urn:miriam:uniprot:P38111	MEC1 SBO:0000245
Exo1I	urn:miriam:uniprot:P39875	EXO1 SBO:0000245
Exo1A	urn:miriam:uniprot:P39875	EXO1 SBO:0000245
Rad9I	urn:miriam:uniprot:P14737	RAD9 SBO:0000245
Rad9A	urn:miriam:uniprot:P14737	RAD9 SBO:0000245
Rad53I	urn:miriam:uniprot:P22216	RAD53 SBO:0000245
Rad53A	urn:miriam:uniprot:P22216	RAD53 SBO:0000245
Chk1I	urn:miriam:uniprot:P38147	CHK1 SBO:0000245
Chk1A	urn:miriam:uniprot:P38147	CHK1 SBO:0000245
Dun1I	urn:miriam:uniprot:P39009	DUN1 SBO:0000245
Dun1A	urn:miriam:uniprot:P39009	DUN1 SBO:0000245

Table 3.2: A detailed breakdown of recovered (in the "MATCH" column) and new (in the "NEW" column) annotation by Saint for the telomere model. Only names, SBO terms and Saint-supported URIs are shown in this table. Additionally, the large number of new GO terms added as MIRIAM URIs are omitted for brevity.

3.5.2.2 Annotation of the stripped p53-Mdm2 model

Basic information on the annotation of both the original and re-annotated p53-Mdm2 model is presented in Table 3.3, together with a summary of the comparison between the two models. A detailed listing of the modifications made to the model is available in Table 3.4. The total number of Saint-supported URIs has increased from 7 to 89, and of the original seven Saint-supported URIs present in the original useful species, five were recovered. The other two URIs were from related species which were incorrectly identified, and are described below. The 84 remaining URIs added to the model are GO URIs. The original model contained only two SBO terms, both of which were recovered. Additionally, two new SBO terms were added. Finally, there were no species names in the original model, and two new names were added with Saint.

In contrast to the telomere model, the p53-Mdm2 model contains species whose id attribute was created as a mixture of two protein names, e.g. "Mdm2_p53". Of the six useful species, four have identifiers formed in this way. For each of these four, Saint suggested annotation based around both of their associated proteins. However, as a composite of two proteins will not necessarily behave in the same way as its constituents, all annotation except for SBO terms and UniProtKB URIs was manually removed. An expert modeller would be able to correctly choose the appropriate name and GO terms. The UniProtKB URIs were kept, but with a relationship of *hasPart*.

Two species, "ARF" and "ARF_Mdm2", were incorrectly identified. In the original model, the both species have an exact "is" match with UniProtKB Q8N726, whose names and synonyms are "Cyclin-dependent kinase inhibitor 2A, isoform 4", "p14ARF" and "p19ARF". The annotation procedure by Saint only had the id to use, however, as the MIRIAM URIs were removed in the stripped models. As such, the correct entry could not be returned due to the mismatch of the query term with the UniProtKB names. However, a number of alternative suggestions were provided by Saint.

3.5.3 Annotating an intact model

Two annotation runs are performed on the pheromone model in this example. First, a stripped version is re-annotated and then the original, intact model is annotated. This example illustrates the benefits of searching based on the resource URI of a species element. Comparison using URIs is suitable primarily for those models whose identifiers and names are uninformative with respect to their biologically-relevant names or descriptions.

As in the previous examples, species composed of more than one protein had all of their suggested GO URIs removed. The authors of the model would themselves be able to choose the correct terms,

Species	18
Species with any annotation†	13
Useful species	6
Species with names	0
Species with SBO terms	2
Saint-supported URIs in useful species	7
Species with names after re-annotation	**2**
SBO terms after re-annotation	**6**
SBO terms recovered from useful species	**2/2**
URIs after re-annotation	**89**
URIs recovered from useful species	**5/7**

Table 3.3: Basic information on the annotation present in the original version of the p53-Mdm2 model and **in bold for the re-annotated version**. This model has no species name attributes, and therefore Saint relies on the id attribute. The two unrecoverable URIs are for the "ARF" species and are described in the main text. †*Any annotation* refers to MIRIAM URIs, names and SBO terms.

SPECIES ID	MATCH	CHANGED	
		Old	New
Mdm2	SBO:0000245 urn:miriam:uniprot:Q00987		E3 ubiquitin-protein ligase Mdm2
p53	SBO:0000245 urn:miriam:uniprot:P04637		Cellular tumor antigen p53
Mdm2_p53	urn:miriam:uniprot:P04637 urn:miriam:uniprot:Q00987		SBO:0000245
Mdm2_mRNA			SBO:0000245 urn:miriam:uniprot:Q00987
ARF		urn:miriam:uniprot:Q8N726	SBO:0000245
ARF_Mdm2	urn:miriam:uniprot:Q00987	urn:miriam:uniprot:Q8N726	SBO:0000245

Table 3.4: A detailed breakdown of annotation applied by Saint for the p53-Mdm2 model. Recovered annotation is shown in the MATCH column. If an item of annotation is new, it will only be present as a "New" annotation. If an item of annotation has been removed, it will only be present in as an "Old" annotation. Only names, SBO terms and Saint-supported URIs are shown in this table. Additionally, the large number of new GO terms are omitted for brevity.

but for simplicity GO URIs were only kept when there was only a single UniProtKB URI per species to ensure no false positives were retained.

3.5.3.1 Annotation of the stripped pheromone model

This section summarises the annotation received by the stripped version of this model. Basic information on the annotation of both the original and re-annotated pheromone model model as well as a summary of the comparison between them is presented in Table 3.5. A detailed listing of the modifications made to the model is available in Table 3.6.

The total number of Saint-supported URIs has increased from 14 to 174. Saint recovered all 14 of the URIs from the useful species, as shown in Table 3.5. 15 new SBO terms were added where none had been present previously. The number of species names increased from 10 to 21 when the new annotation was applied. Of the ten original names, four names were changed to be more descriptive and six were unmodified.

With some species such as "α factor" it was unclear which α factor was intended from the original model, as no further annotation was provided. Therefore one of the most likely candidates was chosen, and only those GO URIs which had the most sources of evidence were kept. Names for complexes such as "complexB" were completely uninformative, and did not retrieve any results.

Species	37
Species with any annotation†	36
Useful species	15
Species with names	10
Species with SBO terms	0
Saint-supported URIs in useful species	14
Species with names after re-annotation	**21**
Species names changed	**4/10**
New species names	**11**
SBO terms after re-annotation	**15**
URIs after re-annotation	**174**
URIs recovered from useful species	**14/14**

Table 3.5: Basic information on the annotation present in the stripped version of the pheromone model, and **in bold for the re-annotated version**. This model has no SBO terms in its original form, although it did contain 10 names, 6 of which were recovered and 4 of which were modified. †*Any annotation* refers to MIRIAM URIs, names and SBO terms.

Table 3.6 shows that a number of UniProtKB accessions were not recovered when re-annotating the stripped model. This is likely due to the nature of the annotations themselves. For instance, `Gabc` has three UniProtKB URIs in the original model, each of which were associated with the species via the *hasPart* MIRIAM qualifier. These three proteins are subunits of a larger complex, and neither the complex nor any of the subunits have protein names matching the string literal "Gabc". As such, when the stripped version was annotated, there was no way for the species to be reconnected to these three proteins. The lack of similarity of the other identifiers to their protein names caused a similar loss of UniProtKB URIs for their species.

SPECIES ID	MATCH	CHANGED	
		Old	New
alpha		α-factor	Mating factor alpha-1 SBO:0000245 urn:miriam:uniprot:P01149
Ste2	urn:miriam:uniprot:P06842		Pheromone alpha factor receptor SBO:0000245
Ste2a	urn:miriam:uniprot:P06842	Ste2active	Pheromone alpha factor receptor SBO:0000245
Ste5	urn:miriam:uniprot:P32917		Protein STE5 SBO:0000245
Ste11	urn:miriam:uniprot:P23561		Serine/threonine-protein kinase STE11 SBO:0000245
Ste7	urn:miriam:uniprot:P06784		Serine/threonine-protein kinase STE7 SBO:0000245
Fus3	urn:miriam:uniprot:P16892		Mitogen-activated protein kinase FUS3 SBO:0000245
Ste20	urn:miriam:uniprot:Q03497		Serine/threonine-protein kinase STE20 SBO:0000245
Ste12	urn:miriam:uniprot:P13574		Protein STE12 SBO:0000245
Ste12a	urn:miriam:uniprot:P13574	Ste12active	Protein STE12 SBO:0000245
Bar1	urn:miriam:uniprot:P12630		Barrierpepsin SBO:0000245
Bar1a	urn:miriam:uniprot:P12630	Bar1active	Barrierpepsin SBO:0000245
Far1	urn:miriam:uniprot:P21268		Cyclin-dependent kinase inhibitor FAR1 SBO:0000245
Cdc28	urn:miriam:uniprot:P00546		Cell division control protein 28 SBO:0000245
Sst2	urn:miriam:uniprot:P11972		Protein SST2 SBO:0000245
Gabc		urn:miriam:uniprot:P08539 urn:miriam:uniprot:P18851 urn:miriam:uniprot:P18852	
GaGTP		urn:miriam:uniprot:P08539	
Gbc		urn:miriam:uniprot:P18851 urn:miriam:uniprot:P18852 urn:miriam:bind:50058	
GaGDP		urn:miriam:uniprot:P08539	
Fus3PP		urn:miriam:uniprot:P16892	
Bar1aex		urn:miriam:uniprot:P12630	
Far1PP		urn:miriam:uniprot:P21268	
Far1U		urn:miriam:uniprot:P21268	

Table 3.6: A detailed breakdown of Saint annotation for the pheromone model, starting from a stripped model. Recovered annotation is shown in the MATCH column. New and removed annotation are in the “New” and “Old” columns, respectively. Modified annotation items have their original values in the “Old” column and their new values in the “New” column. Only names, SBO terms and Saint-supported URIs are shown in this table, with GO terms omitted for brevity. The choice among possible species name is at the discretion of the user.

3.5.3.2 Annotation of the intact pheromone model

This section summarises the improvements in annotation gained by making use of the MIRIAM URIs already present in the original model. When annotating the stripped model, there were a number of species (e.g. `Fus3PP`) which had no annotation retrieved as the identifier and name were unsuitable. In these cases, making use of the URIs present in the intact, original model increases annotation coverage. For instance, UniProtKB URIs can unambiguously identify species with uninformative identifiers such as `GaGTP` and `complexC`, allowing Saint to suggest appropriate species names, GO URIs and SBO terms.

Basic information on the annotation of the intact pheromone model by Saint is presented in Table 3.7, together with a summary of how the new annotation compares both with the original model and the re-annotated stripped model previously described (see Section 3.5.3.1). A detailed listing of the modifications made to the model is available in Table 3.8.

Overall, Saint produced more annotation when starting from the intact model versus the stripped model. The total number of Saint-supported URIs increased from 101 to 325, as compared with 174 URIs added to the stripped version of the model. While the original model did not contain SBO terms and the re-annotation of the stripped model added only 15, the annotation of the intact model produced a total of 21 terms. Further, while the re-annotation of the stripped model was able to add 11 more species names, a further two names were added when starting with an intact model.

	Original model	Stripped model annotated by Saint	Original model annotated by Saint
Species with names	10	21	**23**
SBO terms	0	15	**21**
Saint-supported URIs	101	174	**325**

Table 3.7: The amount of annotation added by Saint when annotating the original version of the pheromone model and when annotating its stripped form. More annotation is added by Saint when the original model is annotated as compared to annotating the stripped model.

3.6 Discussion

Saint is a syntactic data integration application for the enhancement of systems biology models. It was developed as an interactive Web tool to annotate models with new MIRIAM resources and reactions, keeping track of data provenance so the modeller can make an informed decision about the

SPECIES ID	MATCH	CHANGED	
		Old	New
GaGTP	urn:miriam:uniprot:P08539	GαGTP	Guanine nucleotide-binding protein alpha-1 subunit SBO:0000245
GaGDP	urn:miriam:uniprot:P08539	GαGDP	Guanine nucleotide-binding protein alpha-1 subunit SBO:0000245
Fus3PP	urn:miriam:uniprot:P16892		Mitogen-activated protein kinase FUS3 SBO:0000245
Bar1aex	urn:miriam:uniprot:P12630	Bar1activeEx	Barrierpepsin SBO:0000245
Far1PP	urn:miriam:uniprot:P21268		Cyclin-dependent kinase inhibitor FAR1 SBO:0000245
Far1U	urn:miriam:uniprot:P21268	Far1ubiquitin	Cyclin-dependent kinase inhibitor FAR1 SBO:0000245

Table 3.8: A detailed breakdown of recovered (in the MATCH column) and new or modified annotation (in the CHANGED columns) by Saint for the pheromone model, starting from an intact model. Saint discovers all annotation found in Table 3.6 plus the annotation shown in this table. Only names, SBO terms and Saint-supported URIs are shown in this table. Additionally, the large number of new GO terms are omitted for brevity. Whether the original or new species name is used is entirely at the discretion of the user, and these particular examples are not necessarily the only choice a user could make.

quality of the suggested annotation. The system makes it easy for modellers to add explicit biological knowledge to their models, increasing a model's usefulness both as a reference for other researchers and as an input for further computational analysis.

While Saint allows the user to see which data source provided the prospective annotation, annotation accepted by the user does not itself have any provenance information within the completed model. At the moment, both SBML and CellML are unable to make statements about annotations. The proposed annotation package for SBML Level 3 will address this and other issues [10], while the updated metadata specification for CellML will also allow such provenance statements[13].

Saint can save time as well as add information that might not otherwise be introduced without the help of specialists like BioModels curators. Saint can be useful either as a first-pass annotation tool or as a way of adding useful cross-references and possible new reactions. Queries in Saint are sent to the data sources asynchronously, letting the user work on each result as it is returned, without having to wait for a complete response. With Saint, the user examine retrieved annotation, chooses the required annotations, and then moves on to the next task. In general, Saint returns a large amount of prospective annotation for the modeller to examine. Not all of it will be useful; many suggested reactions will be beyond the scope of the uploaded model, and suggested ontology terms might be

[13]`http://www.cellml.org/specifications/metadata/mcdraft`

too general to be useful. However, Saint provides modeller-directed annotation, where some level of filtering is expected.

By examining how Saint behaves with both stripped and intact models, we have isolated the two main query types used within the application: id / name querying and querying based on pre-existing MIRIAM resources. The use of URIs to retrieve information is an effective method of bypassing the variable quality of identifiers and names.

It is difficult for automated procedures such as Saint to disambiguate the identifiers and names used to identify species. While a number of checks and automated modifications can and are performed, many different naming conventions are used. Saint performs well with a minimal amount of data, thus allowing annotation of models that are still at an early stage of development. However, the use of a common naming scheme would improve the coverage of new annotations.

3.6.1 Similar work

While there are a number of tools available for manipulating and validating SBML (e.g. LibSBML), simulating SBML models, analysing simulations and running modelling workflows (e.g. Taverna), Saint is the first to provide basic automatic annotation of SBML models in an easy-to-use GUI. The purpose of Saint is to aid the researcher in the difficult task of information discovery by seamlessly querying multiple databases and providing the results of that query within the SBML model itself. By providing a modelling interface to existing data sources, modellers are able to add valuable information to models quickly and simply.

A few related applications are available for automating the retrieval and integration of data for the annotation of SBML models. SemanticSBML [51] provides MIRIAM annotations via a combination of data warehousing and query translation via Web services as part of a larger application. The Java library libAnnotationSBML [49] uses query translation to provide annotation functionality with a minimal user interface. Unlike libAnnotationSBML, Saint is accessible through an easy-to-use Web interface and unlike both tools is unique in its ability to add new reactions and associated species.

3.6.2 Collaborations and future plans

Saint is under active development. Future enhancements will include the addition of new data sources and ontologies, annotation of elements other than species and reactions and complete support for other modelling formalisms such as CellML. Since early 2009, the CellML developer and curator community has been in active collaboration with the Saint group. A CellML developer has recently been assigned to complete the CellML integration with Saint. Additionally, the collaboration is helping to finalise the way MIRIAM annotations are described in CellML and the addition of helper code for such annotations in the CellML API. They have provided suggestions on how the Saint interface could be modified, and these and other requirements have been formalised in a Saint requirements document[14]. This requirements document contains a full, itemised list of all of the future plans for Saint.

3.6.3 Rule-based mediation and Saint

Saint is a useful, lightweight syntactic data integration environment for systems biology. It provides not only a simple user interface to allow the modeller to integrate data and export it into their models, but is also a successful method of enhancing systems biology models through data integration. However, the work described in this thesis goes further, investigating semantic as well as syntactic methods of integration for the annotation of systems biology models.

Chapter 5 describes the application of semantic data integration to enhance systems biology model annotation. This integration methodology results in the creation of a domain ontology populated with data from multiple data sources. In future, Saint could be modified to use this populated ontology as a data source and query it in a manner similar to the already existing data sources. This would combine the user-friendly interface of Saint with a semantically-integrated knowledgebase.

[14]http://saint-annotate.svn.sourceforge.net/viewvc/saint-annotate/trunk/docs/SaintRequirements.pdf

Chapter 4

Model Format OWL integrates multiple SBML specification documents

Introduction

The creation of quantitative SBML models that simulate the system under study is a time-consuming manual process. Currently, the rules and constraints of model creation, curation, and annotation are distributed over at least three separate specification documents: the SBML XSD, SBO, and the SBML Language Specification for each SBML level (known hereafter as the SBML Manual). Although the XSD and SBO are computationally accessible, the majority of the SBML usage constraints are contained within the SBML Manual, which is not amenable to computational processing.

Model Format OWL (MFO), a syntactic conversion of the SBML standard to OWL, was created to integrate the three specification documents and contains a large amount of the information contained within them [199]. MFO is a a fit-for-purpose ontology performing both structural and constraint integration; it can also be reasoned over and used for model validation. SBML models are represented as instances of OWL classes, resulting in a single machine-readable resource for model checking. Knowledge that was only accessible to humans is now explicit and directly available computationally. The integration of all structural knowledge into a single resource creates a new style of model development and checking. Inconsistencies in the computationally inaccessible SBML Manual were discovered while creating MFO, and non-conformance of SBML models to the specification were identified while reasoning over the models.

Section 4.1 describes the rules and constraints of the SBML standard, while Section 4.2 provides a detailed description of MFO as well as how each section of the SBML standard was modelled. Large-scale conversion of SBML entries as well as the reasoning times for the populated ontologies are described in Section 4.3. Finally, a discussion of similar work and further applicability is in Section 4.4.

Availability

The MFO website is `http://cisban-silico.cs.ncl.ac.uk/MFO/`, and MFO itself is available for download[1]. The website contains general information about the project, links to the Subversion repository, associated publications and instructions for use. MFO is licensed by Allyson Lister and Newcastle University under a Creative Commons Attribution 2.0 UK: England & Wales License[2].

[1]MFO is available from `http://metagenome.ncl.ac.uk/subversion/MFO/trunk/src/main/resources/owl/MFO.owl`. In order for the import statements to work, a copy of SBO must be in the same directory. A release of SBO guaranteed to work with MFO is available from `http://metagenome.ncl.ac.uk/subversion/MFO/trunk/src/main/resources/owl/SBO_OWL.owl`

[2]`http://creativecommons.org/licenses/by/2.0/uk/`

The MFO Software Toolset is covered under the LGPL[3]. For more information, see LICENSE.txt in the MFO Subversion project[4].

4.1 The SBML standard

Both biological and structural knowledge are stored within an SBML document. Biological knowledge describes and identifies the biological entities themselves, while the structural information defines the capturing of biological knowledge in well-formed documents suitable for processing by machines. The structural knowledge required to create an SBML model is tied up in three main locations:

- The SBML [20] XSD, describing the range of XML documents considered to be in SBML syntax;
- SBO [86], containing a hierarchy of unambiguous terms useful as a biologically-meaningful semantic layer of annotation for systems biology models; and
- The SBML Manual [76], detailing the many restrictions and constraints upon SBML documents, the context within which SBO terms can be used, and information on the correct interpretation of documents conforming to the SBML specification.

The SBML specification is in separate documents because each document has been created to match a different type of consumption. The portion of the knowledge codified in the XSD transmits only the information needed to parametrise and run a computational simulation of the system. The knowledge in SBO is intended to aid human understanding of the model and to aid conversion between formats. Finally, the SBML Manual is aimed at modellers as well as tool developers who need to ensure their developed software is fully compliant with the specification.

Only two of the three documents are directly computationally amenable. Firstly, the SBML XSD describes the range of legal XML documents, the required and optional elements and attributes, and the constraints on certain values within those entities. Secondly, SBO provides a term hierarchy of human-readable descriptions and labels together with machine-readable identifiers and inheritance information. Unfortunately, these two documents contain little information on how they interact, the usage of the XML entities or SBO terms, or what a conforming SBML document means to an end user. The majority of such compliance information is in the SBML Manual in human-readable English. With traditional programming techniques, the implementation of these English descriptions

[3]`http://www.gnu.org/copyleft/lesser.html`
[4]`http://metagenome.ncl.ac.uk/subversion/MFO/trunk/`

must be manually hard-coded.

Many libraries are available for the manipulation and validation of SBML models, with libSBML [193] being the most prominent example. LibSBML is an extensive library for manipulating SBML. It is also a powerful constraint validator, reporting any errors in an SBML document by incorporating the constraints found in all three SBML specification documents. The SBML website provides an online XML validator[5], and the SBML Toolbox [200] is also capable of XML validation via libSBML. The online validator uses only the SBML XSD to check the consistency of a model against the schema. The purpose of the SBML Toolbox is to integrate SBML models with the MATLAB environment, but it does have some validation features that check for the correct combination of components. The SBML Test Suite[6] provides a set of test-cases meant to be used in software development, and can also be used to compare different software applications.

However, such manual encoding provides scope for misinterpretation of the specification, and has the potential to produce code that accepts or creates non-compliant documents due to silent bugs. In practice, these problems are ameliorated by regular SBML Hackathons[7] and the widespread use of libSBML. However a formal, machine-readable description of the information contained in the SBML Manual will only become more relevant as the community grows beyond the point where developers and users can be adequately served by face-to-face meetings and informal representations of constraints.

4.2 Model Format OWL

While OWL provides the vocabulary and structure required to describe entities in a semantically-rich and computationally-expressive way, such usage is not mandated by the language. Simpler ontologies can be created in OWL which do not make full use of its semantics, trading expressivity for faster reasoning. A reduction in complexity does not necessarily mean a concomitant reduction in clarity; irrespective of expressivity level, OWL has a number of features unavailable to OBO such as access to reasoners and complex reasoner-based querying [135]. Additionally, when using fit-for-purpose modelling to create computer science ontologies, a rich ontology is often not needed and would generate a high overhead for little gain (see Section 1.5.3.1). Semantically simpler ontologies include those adhering to OWL profiles and those performing direct, syntactic conversion to OWL from a non-semantic format. The three official subsets, or profiles, of OWL are optimised for large numbers

[5]http://sbml.org/Facilities/Validator/
[6]http://sbml.org/Software/SBML_Test_Suite
[7]http://sbml.org/Events

of classes (OWL-EL), large numbers of instances (OWL-QL), or rules usage (OWL-RL) [111]. Syntactic conversion of a data format into OWL allows reconciliation of the format in preparation for the application of more advanced data integration methods.

MFO is within the OWL 2 DL profile, but makes use of a number of expressions which prevent it from conforming to any of the other OWL profiles. MFO captures the entirety of the SBML structure and the SBO hierarchy plus a selection of the human-readable structural rules and semantic constraints from the SBML Manual. This method of constraint integration improves the accessibility of the specification for machines. Since MFO was originally published [199], reasoning and inference times have been dramatically reduced, and more aspects of the three main sources of rules and constraints in SBML have been incorporated.

Detailed descriptions of the modelling of these documents is available in Sections 4.2.3-4.2.5. These sections use the progression of SBase from XSD complex type to OWL class as a desciption of how MFO models information from each of the three SBML specification documents.

The mapping between SBML documents and the OWL representation is bi-directional: information can be parsed as OWL individuals from an SBML document, manipulated and studied, and then serialised back out again as SBML. Standard Semantic Web tools can be used to validate and reason over SBML models stored in MFO, checking their conformance and deriving any conclusions that follow from the facts stated in the document, all without manual intervention. In Chapter 5, we use MFO as the basis for a method of automatically improving the annotation of SBML models with rich biological knowledge.

4.2.1 Comments and annotations in MFO

Comments contained within the SBML XSD are converted to OWL along with the rest of the document. Such comments are identified within the XSD by the use of the xsd:documentation element, and are copied to OWL and identified using the `rdfs:comment` field. This field is also used provide attribution for other facts using the phrases below.

- *Structural Element of SBML*: The classes with this comment have a direct association to a specific element or attribute of the SBML format, and generally have identical names to the original XML.
- *Structural Element of XML*: Classes with this comment have a direct association to a specific part of an XML document. The only OWL class currently containing this statement is *XML-Namespace*, which holds any namespace information from a given SBML document.

- *Documented Constraint*: The comments which begin with this phrase describe documented constraints on their containing classes. They are generally taken from the SBML Manual, and include the version of the specification supplying the constraint.

SBO classes do not contain these comment types because they are neither based on a structural XML element nor drawn from the SBML Manual. However, many MFO classes have more than one of these comments. The most common combination is “Structural Element of SBML” combined with “Documented Constraint”, indicating that a specific SBML component is defined not only in the XSD, but also in the SBML Manual. One example is the MFO class *Reaction*, which restricts its SBO terms to those found in the *occurring entity representation* hierarchy. This documented constraint is found in Table 6, Section 5.2.2 of Level 2 Version 3 Release 1 of the SBML Manual, while the XML element itself is defined in the XSD.

4.2.2 Metrics

Table 4.1 contains selected metrics for MFO and SBO. Definitions of commonly-used metrics are available in Section 1.5.5. As an ontology whose primary purpose is as a controlled vocabulary to facilitate the annotation and reuse of systems biology models [86], SBO is high in classes and in high in annotations. It is also low in instance data and contains only the transitive object property *part of* in addition to the standard subsumption property *is a*. This lack of complex properties and instances combined with a high number of classes and high level of annotation reflects its purpose as a vocabulary.

	Model Format OWL (MFO)	**Systems Biology Ontology (SBO)**
Classes	88	583
Equivalent classes	13	0
Subclass axioms	264	633
Disjoint statements	692	0
Object properties	55	1
Functional object properties	35	0
Data properties	44	0
Functional data properties	26	0
Instances	35	0
Annotation axioms	63	1319

Table 4.1: Selected ontology metrics for MFO and SBO. The first column of data is the metrics for MFO alone, although SBO is a direct import within MFO. A summary of metrics for SBO is provided in the second column. Metrics created using Protégé 4.

In contrast, MFO only needs enough classes to model the various entities present in the SBML XSD and their set of constraints. Therefore the number of classes is smaller in comparison to a vocabulary-based ontology such as SBO. Due to the interconnectedness of the entities and constraints within the SBML specification documents, the total number of class axioms and property types is high relative to the number of classes. Even though MFO is primarily a syntactic ontology and not a model of a biological domain, the syntax of SBML combined with the constraints present in the specification create an interconnected structure.

4.2.3 Modelling the SBML XML Schema

SBML is a good candidate for representation in OWL as the large amount of rules and constraints are described in detail, and the interlocking nature of those rules makes integrating them worthwhile. The XMLTab plugin [201] for Protégé 3.4 reads in either an XSD or XML file and converts that file into a representation in OWL. More information on the use of the XMLTab plugin is available in Section 5.2.2. This plugin was applied to the SBML XSD, resulting in the creation of an initial set of classes and relations. This creation step was virtually instantaneous, and did not require human intervention. However, limitations of the plugin required some manual additions. For instance, although the enumeration of allowed UnitSId values is available within the XSD, it is not converted by the XMLTab plugin. Instead, it was manually created as a set of 35 instances (shown in Table 4.1) of the corresponding *UnitSId* class within MFO.

In the creation of MFO, SBML elements became OWL classes and SBML attributes became OWL properties (either data or object properties, as appropriate). When specific SBML models are loaded, their data is stored as instances. An example of the conversion is shown in Figure 4.1, using a small portion of the SBML SBase complex type. SBase is useful as an example because nearly every SBML type inherits from this complex type.

```
<xsd:complexType name="SBase" abstract="true">
  <xsd:element name="notes" minOccurs="0">
  <xsd:element name="annotation" minOccurs="0">
 <xsd:attribute name="sboTerm" type="SBOTerm" use="optional"/>
</xsd:complexType>
```

Figure 4.1: A portion of the SBML SBase element for Level 2 Version 4, as defined by the XSD. For brevity, the complete complex type has not been shown. Source: http://sbml.org/Special/xml-schemas/sbml-l2v4-schema/sbml.xsd

The XSD provides information on the characteristics of each complex type. Such information includes the optionality of attributes, how often attributes can be used and the typing of attributes (e.g.

boolean). While not easy for humans to read, XML is easily parseable by computers. Figure 4.2 shows how constraints within the XSD for SBase are fully converted into OWL.

```
Class: SBase

    Annotations:
        comment "Structural Element of SBML"

    SubClassOf:
        Thing,
        annotation max 1 Literal,
        notes max 1 Literal,
        sboTerm only SBOTerm,
        sboTerm max 1 Thing
```

Figure 4.2: A portion of the MFO *SBase* class when converted directly from the XSD using the XMLTab plugin. Each of the elements and attributes shown in Figure 4.1 are shown converted into OWL axioms. The Manchester OWL Syntax has been used for readability. For brevity, the complete axiom list has not been shown.

A number of XSD constraints have direct equivalents in OWL. For instance, if an attribute is optional and can only be present once, the `max 1` construct is used, as with the *sboTerm* property:

```
sboTerm max 1 Thing
```

Using an axiom from the *Species* class as an example, the conversion of required attributes makes use of the `exactly` construct:

```
compartment_id exactly 1 SId
```

If the attribute can only be filled with a particular type of entity, then that can be specified as well:

```
sboTerm only SBOTerm
```

While the OWL shown in Figure 4.2 is a direct conversion of the SBML XSD, closer inspection reveals the limitations of XSDs, as very little rule and constraint information is contained within it. For example, the relationship between *SBase* and *SBOTerm* cannot be adequately described without SBO itself, and some usages of the *SId* class within *Species* cannot be correctly modelled without access to the additional specification information found in the SBML Manual. These and other examples of how MFO is able to model aspects of the SBML specification documents are described in detail in the next two sections.

4.2.4 Modelling SBO

SBO is a hierarchy of terms developed by the systems biology modelling community to assist compliance with MIRIAM, ensure unambiguous understanding of the meaning of the annotated entities and foster mapping between annotated elements from multiple formats making use of the ontology [74]. By adding SBO terms, modellers take an essential step towards such compliance [86]. Each term in SBO is related to its direct parent with an *is a* subsumption relationship such as *catalyst is a modifier*. For a full description of SBO itself and its relationship to other systems biology standards, see Section 1.4.1.3.

While SBO is natively stored as a relational database, it can be exported either as OWL or OBO[8]. The OWL export of SBO is regularly downloaded and imported into MFO. The current version of SBO in MFO is the auto-generated OWL export downloaded on 13 December 2011. While the import could directly reference the online version of SBO from the EBI servers, a local copy ensures that MFO works offline and that unexpected changes in SBO do not cause any adverse effects.

In SBML, SBase provides an optional sboTerm attribute which creates a link to a specific term in SBO. Most elements in SBML inherit from SBase and therefore have access to the sboTerm attribute. In MFO, this is represented as the object property *sboTerm*. Figure 4.3 provides examples of *sboTerm* usage within MFO. The SBOTerm simple type, which is the type for the sboTerm attribute used within SBase, defines a string pattern restriction of "`(SBO:)([0-9]7)`" rather than referencing SBO directly. As such, the XSD constraint is only an approximation rather than a complete representation of the correct values of the sboTerm attribute. MFO improves upon this situation by linking the *sboTerm* attribute directly to SBO itself. Figure 4.3 shows how the *material entity* and *physical entity representation* classes from SBO are linked directly to the MFO classes *SpeciesAtAndAfterL2V4* and *SpeciesPreL2V4*, respectively.

There are both formal and informal restrictions on how SBO terms should be used within an SBML model. The formal restrictions are listed in the SBML Manual and covered in the next section. The informal restrictions are concerned with those entities which inherit from *SBase* but which do not have any specific rules governing how SBO should be used. For instance, while the ListOf__ elements inherit from SBase, the SBML Manual does not provide the range (if any) of allowed SBO terms for these elements [76], making it difficult to model them correctly in OWL. Therefore in MFO, *ListOf__* classes have no restrictions on the use of SBO terms. While this is an exact reproduction of the rules and constraints in the specification, a lack of clarity within the specification

[8] http://www.ebi.ac.uk/sbo/main/download.do

```
Class: SpeciesPreL2V4

    Annotations:
        comment "Documented Constraint: Table 6, Section 5.2.2
         of L2V3 Release 1, Structures and Facilities for Model
         Definitions (for sboTerm restriction)."

    EquivalentTo:
        Species
         and ( inverse (species) only ( inverse (listOfSpecies) only
           ( inverse (model) only
            (SbmlL2V1
             or SbmlL2V2
             or SbmlL2V3))))

    SubClassOf:
        sboTerm only 'physical entity representation'

Class: SpeciesAtAndAfterL2V4

    SubClassOf:
        sboTerm only 'material entity'
```

Figure 4.3: The *SpeciesPreL2V4* and *SpeciesAtAndAfterL2V4* classes whose union is the full necessary and sufficient definition of *Species*, as shown in Figure 4.4. These classes allow the reasoner to check that the appropriate SBO terms are used for the appropriate SBML model versions. The Manchester OWL Syntax has been used for readability, and only part of the *SpeciesAtAndAfterL2V4* class is shown for brevity.

leads to possibly incomplete modelling.

4.2.5 Modelling the SBML Manual

The SBML Manual contains the majority of the rules and constraints to which valid models must conform. For instance, depending on the value of the constant and boundaryCondition attributes of the species element, corresponding speciesReference elements may or may not be products or reactants in reactions [76, Section 4.8.6, Table 5].

A new document is released for each new version of SBML. New versions of the manual may create new rules, invalidate old ones or publish changes to existing rules. For example, an explicit statement of the SBO terms allowed for modifierSpeciesReference was removed in later versions of the SBML Manual. The Level 2 Version 4 Manual states that modifierSpeciesReference elements should have SBO terms from the *modifier* hierarchy [202, Section 4.13.2]. In Level 3 Version 1 this constraint is no longer explicitly stated, though it is implied [76, Section 4.11.4]. Determining the correct restriction to use in cases such as this becomes impossible without direct consultation with the developers of the specification. Such a lack of precision is a serious pitfall of specification documents which are only human readable.

In addition to the inexact wording possible with an English specification, new versions of the SBML Manual also change which SBO term hierarchies can be used for particular elements [202, 76, Table 6]. For example, as shown in Figure 4.3, *material entity* and its children are the only SBO terms that can be used with *Species* classes at or after Level 2 Version 4. However, *Species* classes have different constraints on their SBO terms if they are before Level 2 Version 4.

To solve this problem and allow the reasoner access to this logic, the *Species* class was given two additional children, *SpeciesAtAndAfterL2V4* and *SpeciesPreL2V4*. These child classes are together marked as equivalent to the *Species* class (Figure 4.4), and are fully modelled based on the level and version type of the SBML document (Figure 4.3). These are some of the many constraints placed on the use of SBO terms documented within the SBML Manual, and not captured in either the SBML XSD or the SBO OBO file.

The *SId* class models all identifiers internal to the SBML model, and is heavily used in the XML to provide links from one part of the document to another. It and its child class, *UnitSId*, constrain the type of identifier allowed for entities within SBML [76, Section 3.1.7]. The conversion to OWL from the XSD does create an *SId* class, but a lack of further information means that the relevant parts of

```
Class: Species

    Annotations:
        comment "Documented Constraint: 4.8.3 of L2V3 Release 2 and L2V4
         Release 1, Structures and Facilities for Model Definitions (for
         the non-optional link to Compartment via hasCompartmentId). Though
         this information is also present in the XSD, the XSD does not say
         what that SId must belong to."^^string,
        comment "Documented Constraint: 4.8.2 of L2V3 Release 2 and L2V4
         Release 1, Structures and Facilities for Model Definitions (for
         the optional link to SpeciesType)"^^string

    EquivalentTo:
        SpeciesAtAndAfterL2V4
         or SpeciesPreL2V4

    SubClassOf:
        speciesType_id only (id_of only SpeciesType),
        speciesType_id max 1 Thing,
        compartment_id only (id_of only Compartment),
        compartment_id exactly 1 Thing,
```

Figure 4.4: Example constraints added to MFO based on information contained in the SBML Manual. The constraints listed here are in addition to those already added by XMLTab. As some constraints are dependent upon the level and version of the SBML model, subclasses of *Species* were created (see also Figure 4.3). Additionally, the constraints present in the SBML Manual allowed the correct linking of properties such as *speciesType_id* and *compartment_id* to their owning classes via a restriction on the *SId* used in those properties.

the SBML Manual must also be used to correctly model *SIds* within MFO. Simple statements using the *id* object property require no changes to the axioms created by the conversion process:

```
id only SId
```

The statement above, taken from the *Species* class, states that the *Species* identifier must be of type *SId*. This matches the behaviour defined in the SBML Manual, and no further modifications to the OWL are required. However, other cases where *SIds* are used within the *Species* class are not completely defined in the XSD. These cases are when a *SId* is used as the link between *Species* and another class. For instance, *compartment_id* is the object property which, ultimately, links a given *Species* to its containing *Compartment*, but the XSD does not specify this, so the conversion to OWL results in:

```
compartment_id exactly 1 SId
```

While this statement tells us that *compartment_id* must be of type *SId*, it does not let us know that it must connect to an *SId* belonging to a *Compartment*. Therefore, the addition of this information from the SBML Manual results in the following modification to the statement (also shown in Figure 4.4):

```
compartment_id only (id_of only Compartment),
compartment_id exactly 1 Thing
```

On its own, the conversion of SBase into OWL results in the creation of a generic *annotation* data property which is filled by a string literal (Figure 4.2). However, a major part of the annotation of an SBML entity is its MIRIAM annotation. Therefore to align more closely with the SBML Manual and to aid further integration work in this thesis, classes describing MIRIAM annotation in SBML have been added. Figure 4.5 shows the *MiriamAnnotation* class and the *miriamResource* data property in the context of *SBase*, while Figure 4.6 shows the MIRIAM qualifiers within MFO. The MIRIAM biological qualifiers are held within the *bqb* property hierarchy, and the model qualifiers are listed within the *bqm* hierarchy. MIRIAM-compliant annotation elements are important to rule-based mediation (Chapter 5), as these elements contain information such as cross references and taxonomic information.

```
Class: MiriamAnnotation

    Annotations:
        comment "Documented Constraint: Section 6 of L2V4 Release 1,
        Structures and Facilities for Model Definitions (for the
        structure of this property)."^^string

    SubClassOf:
        Thing,
        miriamResource exactly 1 Literal

    DisjointWith:
        XMLNamespace, SId, SBOTerm, SBase


Class: SBase

        miriamAnnotation only MiriamAnnotation
```

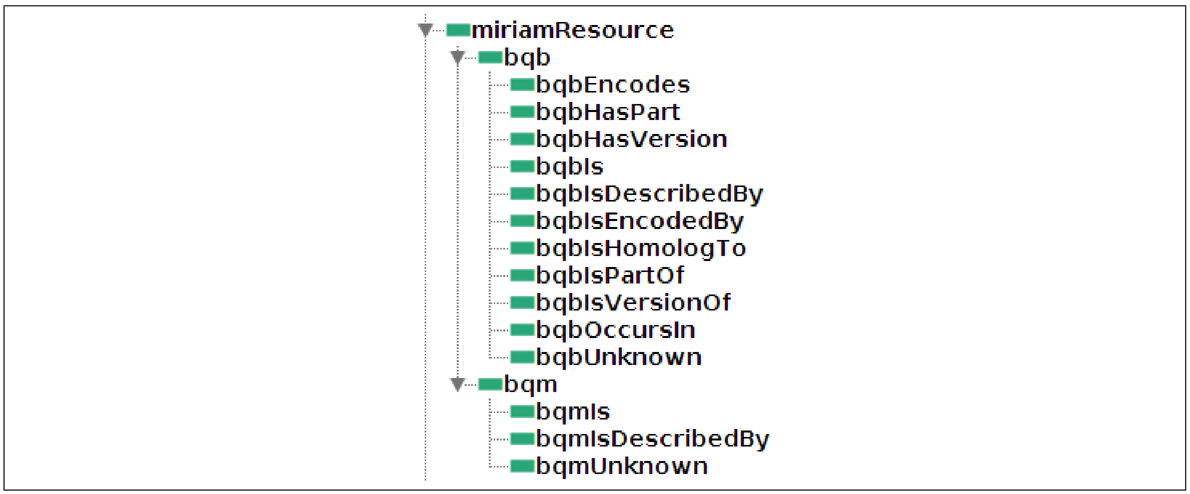
Figure 4.5: Example constraints added to MFO based on information contained in the SBML Manual. A new class called *MiriamAnnotation* was created and linked to *SBase*, allowing the full and complete modelling of the MIRIAM structure. The constraints listed here are in addition to those already added by XMLTab.

4.3 Working with MFO

On its own, MFO integrates the rules and constraints present in the three SBML specification documents, making those rules available to machines and presenting them in a single view for humans. Further, by adding data from SBML models themselves, the information stored within the model is exposed to reasoners and semantic querying. This section describes the population and exporting of MFO using the entirety of the BioModels database as an example, and shows the benefits of reasoning over even a relatively simple syntactic ontology such as MFO. Chapter 5 then describes the larger semantic integration project and how MFO can be used in such a situation.

4.3.1 Creating instance data

In Figure 4.7, an SBML species is shown in its original XML format, while its corresponding instance data in MFO is shown in Figure 4.8. Each attribute of *Species* is filled with the data provided from the XML. Data properties point directly to the value for that property. For example, the *name* data property points to "DesensitisedACh". Object properties can be followed to the instances being referenced. For example, a *miriamAnnotation* object property points to `BIOMD0000000001_000012_bqbIsVersionOf_0_0`, an instance of the class *MiriamAnnotation*.



```
miriamResource
    bqb
        bqbEncodes
        bqbHasPart
        bqbHasVersion
        bqbIs
        bqbIsDescribedBy
        bqbIsEncodedBy
        bqbIsHomologTo
        bqbIsPartOf
        bqbIsVersionOf
        bqbOccursIn
        bqbUnknown
    bqm
        bqmIs
        bqmIsDescribedBy
        bqmUnknown
```

Figure 4.6: The hierarchy of MIRIAM qualifiers in OWL, as shown in Protégé 4. *miriamResource* is the top-level data property, and the biology qualifier (modelled here as *bqb*) and model qualifier (modelled here as *bqm*) hierarchies follow those found in the SBML Manual.

```
<species metaid="_000012" id="DL" name="DesensitisedACh"
 compartment="comp1" initialAmount="0" sboTerm="SBO:0000297">
<annotation>
    <bqbiol:isVersionOf>
        <rdf:li rdf:resource="urn:miriam:interpro:IPR002394"/>
        <rdf:li rdf:resource="urn:miriam:obo.go:GO%3A0005892"/>
    </bqbiol:isVersionOf>
 </annotation>
</species>
```

Figure 4.7: The species “DL” from BioModels entry BIOMD0000000001 in the native XML format. Please note that multiple sections of the XML have been removed to aid readability.

This instance has a *bqbIsVersionOf* data property whose value is “urn:miriam:interpro:IPR002394”. Additionally, the *sboTerm* attribute points, not to a string literal as described in the XSD, but to an actual *SBOTerm*, as required by the SBML Manual.

4.3.2 Populating MFO

When converting data from SBML to MFO, first the XML is parsed and then corresponding OWL is created. Each of these steps is aided by existing libraries coordinated with the MFO Software Toolset. LibSBML 5.0.0 [193] converts the XML structures into plain Java objects. Each Java object is then assigned to a particular MFO entity and saved as OWL using the OWL API version 3.2.3 [112]. After conversion back to SBML, roundtrip comparisons of the XML both before and after processing were performed to ensure that all data is retained.

The BioModels [21] database release of 15 April 2011 contains 326 curated and 373 non-curated

```
Individual: BIOMD0000000001_DL

    Types:
        Species

    Facts:
     id  BIOMD0000000001_DL_SId,
     miriamAnnotation  BIOMD0000000001_000012_bqbIsVersionOf_0_0,
     miriamAnnotation  BIOMD0000000001_000012_bqbIsVersionOf_0_1,
     sboTerm  sbo_0000297,
     name  "DesensitisedACh"^^string

Individual: BIOMD0000000001_DL_SId

    Types:
        SId

    Facts:
     id_of  BIOMD0000000001_DL,
     id_value  "DL"^^string

Individual: BIOMD0000000001_000012_bqbIsVersionOf_0_0

    Types:
        MiriamAnnotation

    Facts:
     bqbIsVersionOf  "urn:miriam:interpro:IPR002394"^^string

Individual: sbo_0000297

    Types:
        'protein complex'
```

Figure 4.8: The `BIOMD0000000001_DL` instance of the MFO *Species* class from BioModels entry BIOMD0000000001. In addition to `BIOMD0000000001_DL` itself, a small number of object properties are expanded to show their actual values.

models. Within these two datasets, a total of six models could not be loaded into libSBML at all, and a further nine did not pass the roundtrip test from XML to MFO and back again. There was only a single curated model which failed loading into libSBML: BIOMD0000000326 contains a notes element which is in an incorrect format and was rejected. An additional five entries from the non-curated set failed loading for the same reason[9]. Table 4.2 shows the number of instances and property axioms created when both the curated and non-curated BioModels datasets were converted into MFO.

	Curated BioModels	Non-Curated BioModels
Instances	149,912	1,217,067
Object Property Assertions	246,679	2,225,935
Data Property Assertions	221,738	1,884,250

Table 4.2: The number of created instances and related axioms when the BioModels database is converted into MFO. Curated and non-curated datasets are shown separately.

As SBML models are standalone and cannot be imported into each other, when downloading the BioModels database, each entry is stored in its own file. Figure 4.9 summarises individual creation times for MFO models of all parseable curated and uncurated entries in the BioModels database. The conversion to MFO generally takes less than one second, although uncurated entries take longer, on average, to be converted.[10]. Conversion times were longer for larger SBML models such as the yeast molecular interaction network model [203] present in the non-curated dataset[11].

When roundtrip testing was performed, two curated models (BIOMD0000000169 and BIOMD0000000195) had errors in their original MIRIAM annotation resulting in libSBML dropping some of the MIRIAM URIs. These have been confirmed as errors in the SBML model by the BioModels curators [12]. A third curated model, BIOMD0000000228, gains a superfluous sbml namespace on most elements when converting from MFO to SBML, causing an error within libSBML. A total of six non-curated models failed the roundtrip process. Four of these models[13] had segmentation faults from libSBML's C++ core during the comparison of the original and the newly-converted SBML, and the two biggest non-curated models[14] could not feasibly be compared in memory.

Although each BioModels entry is stored within the MFO Subversion project in its own file to match

[9]MODEL0911120000, MODEL1011020000, MODEL1012110001, MODEL1150151512, MODEL6655501972
[10]All timings in this chapter were performed on a Dell Latitude E6400 computer, unless otherwise stated.
[11]MODEL3883569319
[12]http://sbml.org/Forums/index.php?t=tree&goto=7097&rid=0
[13]MODEL2463683119, MODEL4132046015, MODEL1009150002, MODEL1949107276
[14]MODEL3883569319, MODEL6399676120

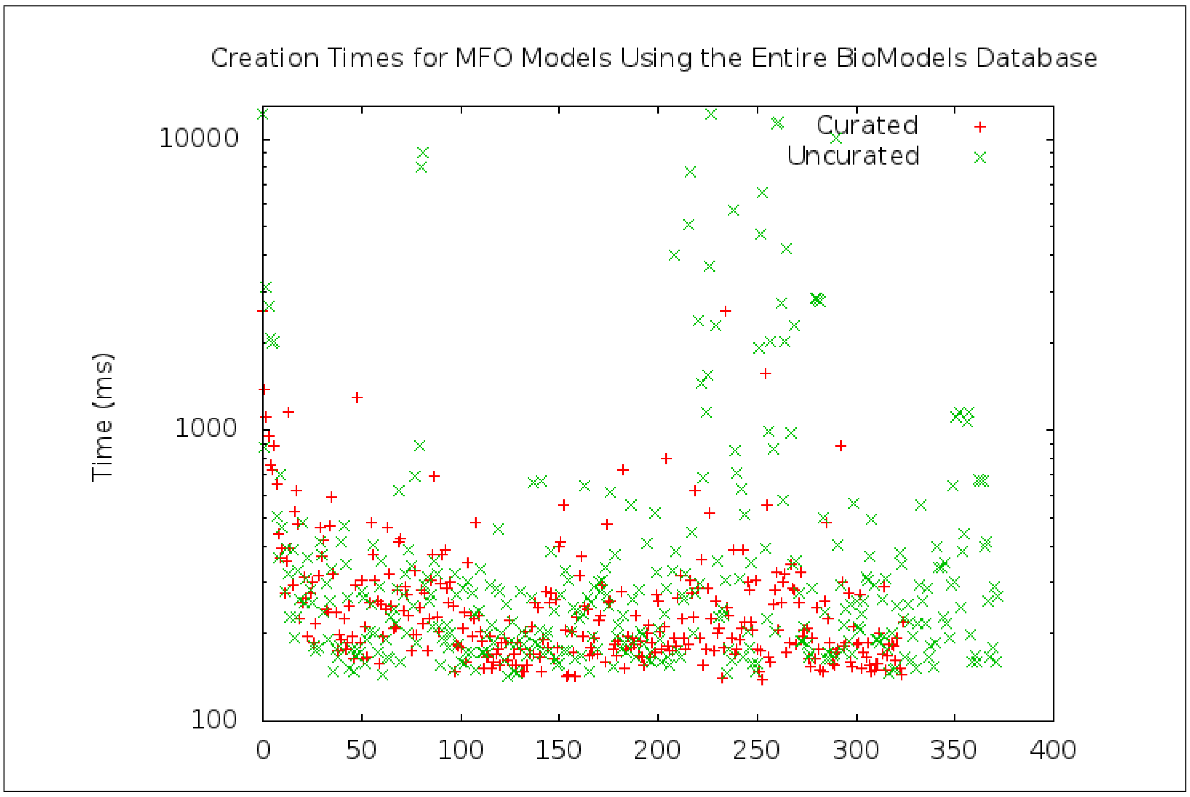


Figure 4.9: Summary of the conversion times for all MFO models stemming from the complete BioModels database. Each dot corresponds to a single BioModels entry, and the conversion times are plotted on a logarithmic scale. The extreme outlier, MODEL3883569319, is not included in this graph to aid readability (conversion time 2401 seconds).

how those database entries are provided, there is no reason why the entire database cannot be stored in a single OWL file. Within the MFO Subversion project, OWL files are available for download both singly and as a concatenation of all curated data.

4.3.3 Reasoning over MFO

Reasoning over an ontology can provide a number of benefits. Inconsistencies in the logic or in the instance data can be identified, and inference of more precise locations for classes and instances can create new hierarchies and pull out otherwise hidden knowledge. By using MFO, constraints can be automatically enforced that currently are only described textually in the SBML Manual or hard-coded into libSBML.

Libraries such as libSBML capture the process of validating constraints, while MFO explicitly captures the constraints themselves, thus identifying where the SBML Manual is not explicit or appears to contradict itself. Examples of this constraint capture and verification were already described in Section 4.2.5, where a lack of precision in the English language definitions as well as changes in correct annotation for sboTerm attributes result in logical inconsistencies. These inconsistencies were discovered while MFO was being created, as the axioms for the relevant parts of the specification were being added. However, while it is not easy to manually spot such errors it is also not required that they be noticed at this stage. If two incompatible SBO term branches of the SBO hierarchy were both stated to be the only branches allowed for a particular MFO entity, the reasoner would catch this inconsistency even if the ontologist did not.

The ongoing process of incorporating constraints into MFO will improve the quality of the SBML Manual through identification of these weaknesses. Once captured in MFO, these constraints can be investigated systematically by running a reasoner. This allows many aspects of models to be validated without the need for hard-coded checking libraries. Furthermore, the addition of new constraints requires the release of a new version of a library such as libSBML, but would only require retrieval of the new MFO file; no changes are required to the supporting MFO Toolset.

Factors affecting reasoning time are complex, and are not just tied to ontology, ABox or TBox size. Such a discussion is beyond the scope of this thesis, but Motik and colleagues provide a thorough discussion and comparison of current reasoners [130]. Figure 4.10 shows a plot of the reasoning times for curated BioModels, providing a general overview of how long reasoning takes over MFO models. Although, very broadly, larger models took longer to reason over, Figure 4.10 shows that size is not a highly useful indicator of reasoning time. Only the 290 models which were both consistent and satisfiable were included in the plot. The vast majority of models in this set took less than one second to reason over. Two extreme outliers took much longer to reason over, most likely due to their sizes, as they are two of the largest models in the curated BioModels dataset[15].

As a syntactic ontology, the simple structure allows fast reasoning, and the benefits gained include SBO validation and inference of more specific subtypes of classes. Of the 35 inconsistent curated BioModels entries, a sample were examined manually to determine the causes of the inconsistencies. For both BIOMD0000000055 and BIOMD0000000216, the inconsistencies stemmed from errors in the use of SBO terms. In BIOMD0000000055, the SBO term *messenger RNA* was used within a species. As this model is Level 2 Version 4, and *messenger RNA* is not part of the *material entity* hierarchy (see Section 4.2.5), the use of this term makes the ontology inconsistent. Similarly, in BIOMD0000000216 the incorrect SBO term *simple chemical* is used within a *parameter*. Therefore, inconsistencies in models represented in MFO can be traced to errors in the underlying SBML documents.

Reasoners can also place instances in more specific sub-classes than the class in which they were

[15]BIOMD0000000235 and BIOMD0000000255

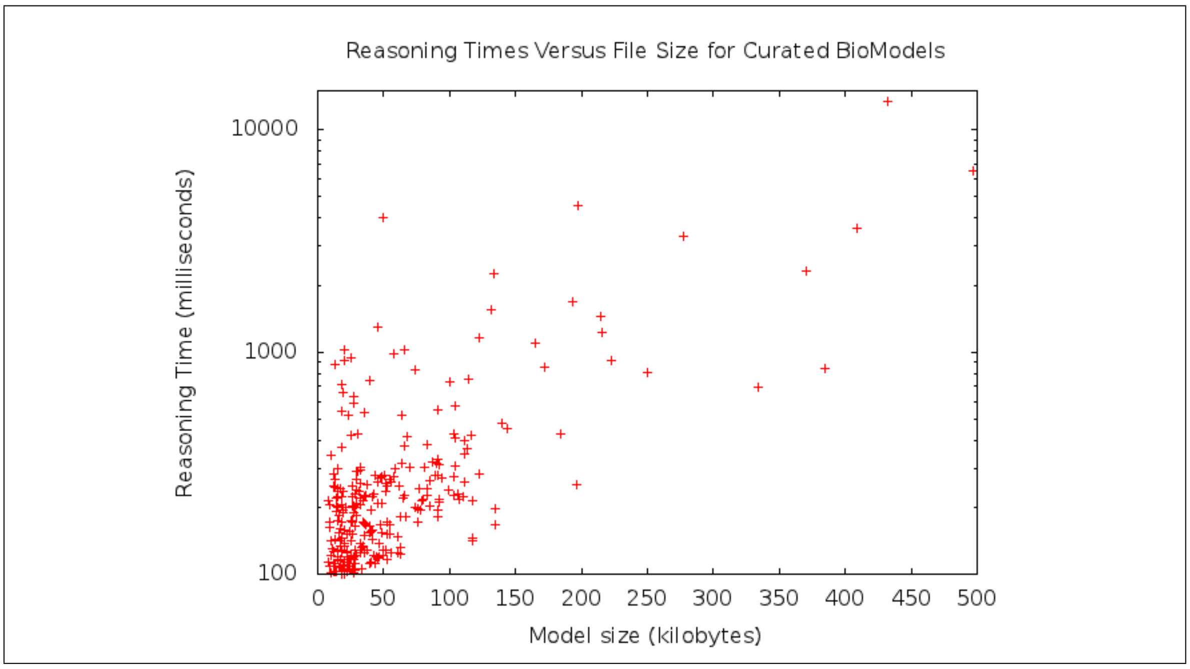


Figure 4.10: Reasoning times over curated BioModels modelled in MFO. Each dot corresponds to a single BioModels entry, and the reasoning times are plotted on a logarithmic scale. The reasoning time is plotted against the size of each model. Two extreme outliers (BIOMD0000000235: 924 seconds, BIOMD0000000255: 370 seconds) have not been included in the graph. HermiT [130] version 1.3.3 was used as the reasoner.

originally asserted. For instance, how a species' quantity can be changed depends upon the values of the boundaryCondition and constant attributes [76, Table 5]. If both are true, the quantity of the species cannot be altered by the reaction system and therefore its value will not change during a simulation run. MFO models all four possible combinations of these booleans as four distinct sub-types of *Species*. The placement in the *Species* hierarchy of a given instance can be inferred using the values of these properties. As an example, if an instance is asserted to be a *Species* and both *boundaryCondition* and its *constant* are true, reasoners will correctly classify this individual as a *ConstantBoundarySpecies*.

4.4 Discussion

4.4.1 Similar work

The CellML Ontological Framework [204] has parallels with MFO in that it aims to represent CellML models in OWL. This project was published two years after MFO, and uses, among other things, graph reduction algorithms to represent CellML models in a biological "view" which shows the biology of the model while hiding the complexity of the full CellML document. However, the purpose of the CellML Ontological Framework is to aid readability of CellML models, not to integrate or validate.

SBML Harvester is a tool similar to MFO, allowing the conversion of SBML models into OWL [205]. Like MFO, multiple SBML documents can be loaded into a single OWL file, reasoned over, and queried. SBML Harvester can be exported into the OWL-EL profile, and incorporates a minimal, custom-made upper-level ontology which itself makes use of the Relation Ontology [117]. Unlike MFO, which models only the SBML specification and not the underlying biology, SBML Harvester incorporates both biological concepts and the model concepts into the same ontology. This conflates two completely different domains into a single ontology. Further, SBML Harvester is not a complete model of the SBML specification. It currently only covers a limited subset of the SBML document: compartments, species, reactions and the model element itself. This means that it is not capable of performing a full round-trip of the SBML documents.

In theory BioPAX could be used to model all SBML documents, as there are tools available for converting from one format to another. However, this solution is not feasible for a number of reasons. Most importantly, BioPAX does not store quantitative information or the logical restrictions and axioms that are otherwise only stored in either SBO or the SBML Manual. As such, BioPAX cannot be easily used to perform the tasks possible with MFO. Le Novère stated that current converters result in "pathological SBML or BioPAX, and the round-tripping is near impossible" [16]. Few modellers in the target group for MFO use BioPAX natively. Their simulators accept SBML, and they develop and test their models in SBML. For those familiar with SBML, MFO is likely to be a much more accessible view of models than BioPAX. Finally, the export of data from the integration project described in Chapter 5 needs the SBML format, which can be exported losslessly with MFO but not with BioPAX.

4.4.2 Applicability

MFO represents the information required to build and validate an SBML model in a way accessible to machines. The three separate sources of this information meet the needs of distinct user groups and, while optimal for the existing range of human-driven applications, this disparate arrangement is not suitable for computational approaches to model construction and annotation. This is primarily because much of the knowledge about what constitutes valid and meaningful SBML models is tied up in a non-machine-readable format. Once available to machines, the model is exposed to reasoners which can be used to check for internal consistency and correct structure.

[16] https://groups.google.com/d/topic/biopax-discuss/X2wkwvLJEhg/discussion

MFO is not intended to be a replacement for any of the APIs or software programs available to the SBML community. Rather, MFO is a complementary approach that facilitates the development of the rule-based mediation integration strategy (Chapter 5) for model improvement and manipulation by drawing upon internal consistency and inference capabilities. With MFO, inconsistencies both in the specification documents and in the models themselves can be identified. It addresses the very specific need of a sub-community within SBML that wishes to be able to express their models in OWL for the purpose of reasoning, validation and querying.

The scope of MFO does not extend to mapping other domains or integrating other data sources. These integrative tasks have a logically distinct purpose. MFO can be used on its own or as one of the source ontologies integrated via rule-based mediation. Chapter 5 goes beyond the standalone usage described here to explain how rule-based mediation utilises MFO with other source ontologies to integrate and query systems biology data. In rule-based mediation, MFO has a dual role: firstly, it acts as the format representation for SBML models such as those stored within the BioModels database, and secondly it exports query results as new annotation on SBML documents.

Chapter 5

Rule-based mediation for the semantic integration of systems biology data

Introduction

The creation of accurate quantitative SBML models is a time-intensive manual process. Modellers need to know and understand both the systems they are modelling and the intricacies of the SBML format. However, the amount of relevant data for even a relatively small and well-scoped model can be overwhelming. Systems biology model annotation in particular can be time-consuming; the data required to perform the annotation is represented in many different formats and present in many different locations. The modeller may not even be aware of all suitable resources.

There is a need for computational approaches that automate the integration of multiple sources to enable the model annotation process. Ideally, the retrieval and integration of biological knowledge for model annotation should be performed quickly, precisely, and with a minimum of manual effort. If information is added manually, it is very difficult for the modeller to annotate exhaustively. Current systems biology model annotation tools rely mainly on syntactic data integration methods such as query translation. A review of data integration methodologies in the life sciences is available in Section 1.6, and the Saint syntactic integration system for model annotation is described in detail in Chapter 3.

While large amounts of data can be gathered and queried with syntactic integration, the integration process is limited to resolving formatting differences without attempting to address semantic heterogeneity, which occurs when a concept in one data source is matched to a concept in another data source which is not exactly equivalent. Semantic data integration methods can be used to resolve differences in the meaning of the data. This chapter describes *rule-based mediation*, a novel methodology for semantic data integration. While rule-based mediation can be used for the integration of any domain of interest, in this instance it has been applied to systems biology model annotation. Off-the-shelf Semantic Web technologies have been used to provide novel annotation and interactions for systems biology models. Existing tools and technology provide a framework around which the system is built, reducing development time and increasing usability.

There are three main parts to rule-based mediation: the source ontologies for describing the disparate data formats, the mapping rules for linking the source ontologies to a biologically-relevant core, and the core ontology for describing and manipulating semantically-homogeneous data (see Figure 5.1). In rule-based mediation, the data is materialised in the core ontology by using a set of rules to translate the data from the source ontologies to the core ontology, avoiding the need for complex query translation algorithms.

Rule-based mediation has a number of defining features:

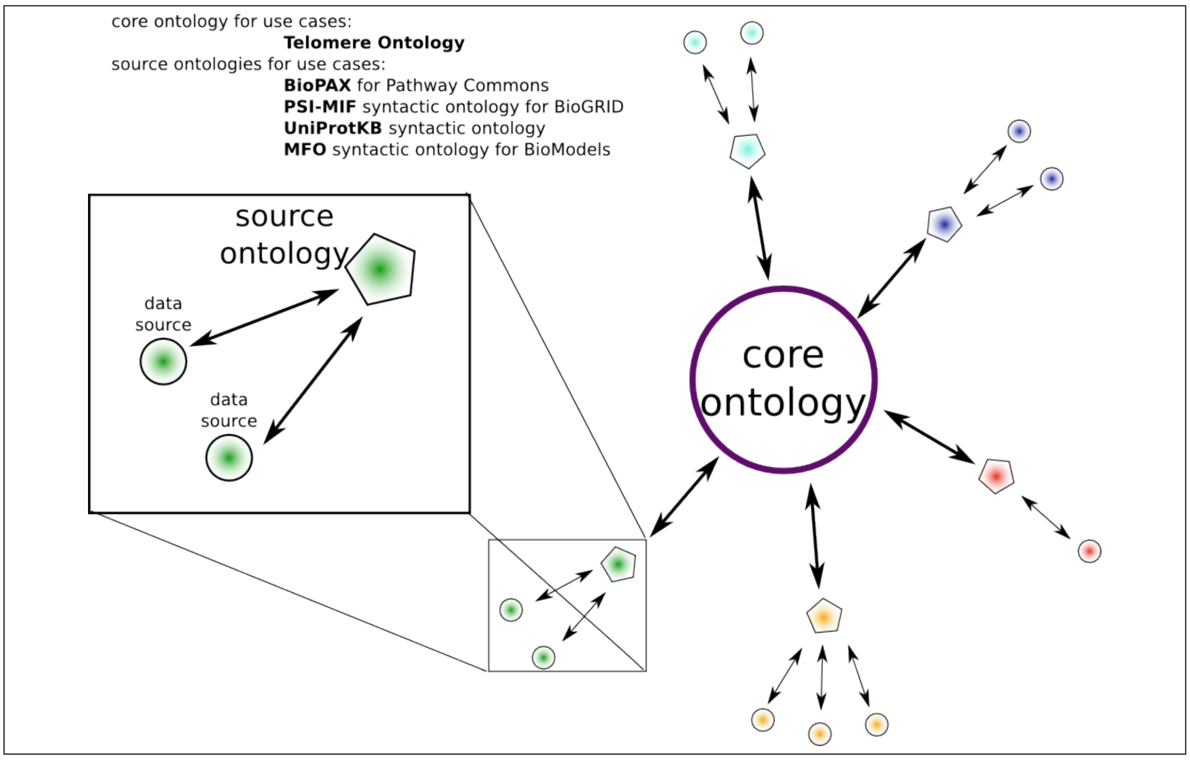

core ontology for use cases:
Telomere Ontology
source ontologies for use cases:
BioPAX for Pathway Commons
PSI-MIF syntactic ontology for BioGRID
UniProtKB syntactic ontology
MFO syntactic ontology for BioModels
source ontology
data source
data source
core ontology

Figure 5.1: An overview of rule-based mediation. One or more data sources sharing the same format (shown here as small circles) can be represented using a source ontology (shown as pentagons). Rules, or mappings, from the source ontology to the core ontology provide a route through which information from the data sources is mapped. The information within the core ontology can then be queried and reasoned over. Relevant knowledge for a particular systems biology model or models can then be exported from the core ontology via a chosen source ontology and ultimately saved in a underlying data source format such as SBML. In the upper left corner, the names of the core and source ontologies created or used for the research described in this thesis are listed.

- the data from syntactically and semantically different data sources are materialised within a core ontology for easy access and stable querying;
- the core ontology is a semantically-meaningful model of the biological domain of interest; and
- implementation can be performed using off-the-shelf tools and mapping languages.

Integrating resources in this way accommodates multiple formats with different semantics and provides richly-modelled biological knowledge suitable for annotation of systems biology models. This work establishes the feasibility of rule-based mediation as part of an automated systems biology model annotation system.

Section 5.1 describes the type of integration performed in rule-based mediation as compared with existing data integration methodologies. The first step in rule-based mediation is the syntactic conversion of heterogeneous data sources into ontologies, as detailed in Section 5.2. These source ontologies are then aligned to a small core ontology by applying a rule base (Section 5.3). This core ontology is a model of the biological domain of interest which can be queried and reasoned over (Section 5.4). As demonstrated in Section 5.5 with two use cases, selected results from the core domain ontology can then be applied to systems biology models as a variety of types of biological annotation. Section 5.6 describes possible future directions and applications for rule-based mediation, and Section 5.7 provides a final discussion and summary of the methodology.

Availability

The rule-based mediation website is `http://cisban-silico.cs.ncl.ac.uk/RBM/`, and the ontologies and software toolset is available from Subversion[1]. The ontologies stored within the rule-based mediation project are licensed by Allyson Lister and Newcastle University under a Creative Commons Attribution 2.0 UK: England & Wales License[2]. The rule-based mediation software toolset is covered under the LGPL[3]. For more information, see `LICENSE.txt` in the Subversion project.

The SWRL rules described in this chapter can be run programmatically or via the Protégé 3.4 editor[4]. Programmatic access requires the rule-based mediation software toolset, which in turn requires the OWL API. The SWRLTab plugin within Protégé 3.4 requires the installation of the Jess Rule Engine[5] for Java. All mapping rules are available for viewing within `swrl-mappings.owl` in the Subversion distribution. Instructions for viewing the telomere ontology, the source ontologies, the SWRL mappings, Semantic Query-Enhanced Web Rule Language (SQWRL) queries, and the SBML model before and after annotation in these use cases are available in the Subversion repository[6].

5.1 Design decisions

This section describes the design decisions made for rule-based mediation with respect to existing methodologies. An overview of this integration methodology is summarised in Figure 5.2 in the context of SBML model annotation. First, information is syntactically converted into OWL or retrieved, correctly formatted, in an pre-existing source ontology. Second, the information is mapped into a core ontology. Third, the core ontology is queried to answer specific biological questions. Finally, the new information is sent back to an external format, in this case SBML, via mappings from the core ontology to MFO. From a biological perspective, rule-based mediation produces an integrated view of information useful for modelling.

[1]`http://metagenome.ncl.ac.uk/subversion/rule-based-mediation/telomere/trunk`
[2]`http://creativecommons.org/licenses/by/2.0/uk/`
[3]`http://www.gnu.org/copyleft/lesser.html`
[4]`http://protege.stanford.edu`
[5]`http://www.jessrules.com/`
[6]`http://metagenome.ncl.ac.uk/subversion/rule-based-mediation/telomere/trunk/INSTALL.txt`

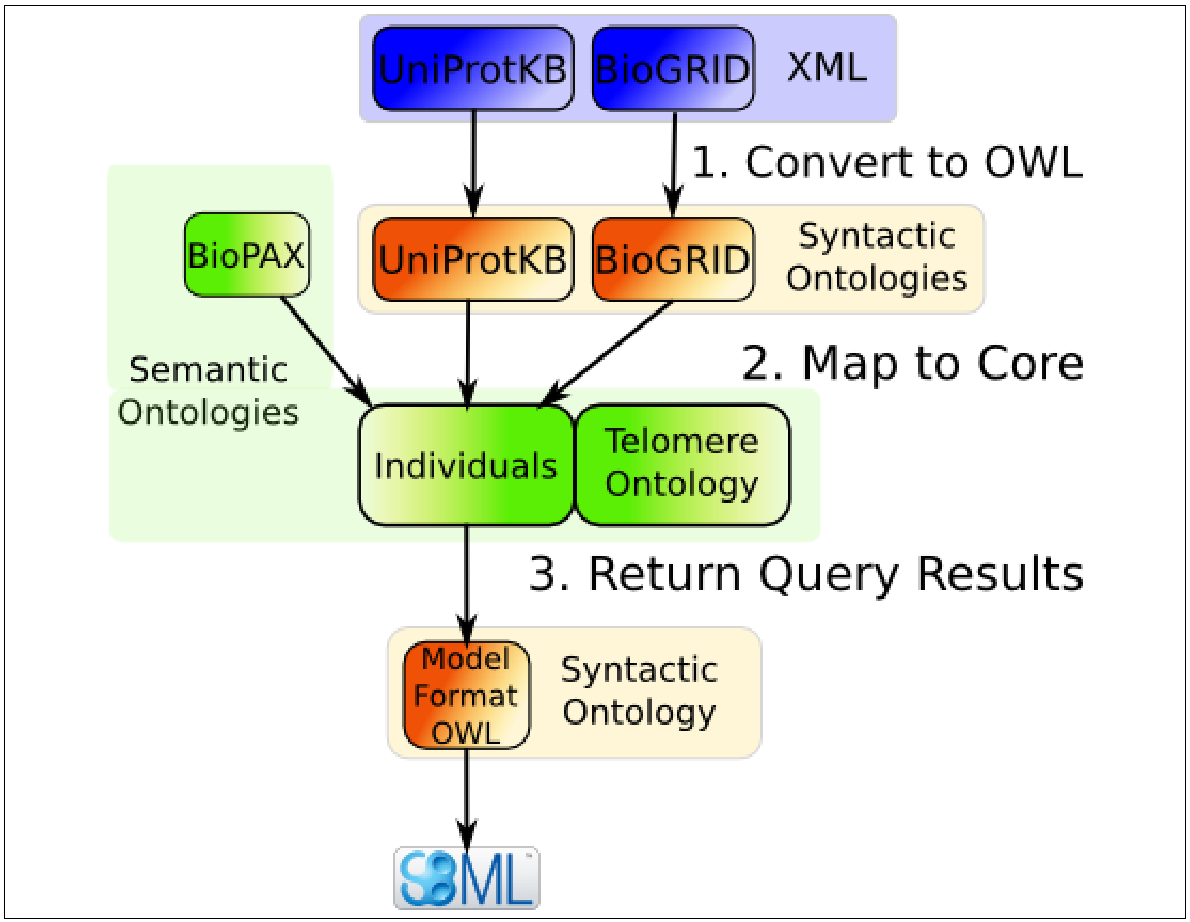


Figure 5.2: Rule-based mediation in the context of SBML model annotation. Non-OWL formats are first converted into syntactic ontologiesvia the XMLTab plugin for Protégé 3.4. Here, both UniProtKB and BioGRID data are in XML: UniProtKB has its own schema, while BioGRID uses PSI-MIF. BioPAX, the format used for Pathway Commons, is already in OWL and needs no conversion. Next, the instances present in the source ontologies are mapped to the core ontology using SWRL. Finally, querying is performed using SQWRL queries using only core ontology concepts. Information is then passed through a final syntactic ontology (MFO) into an SBML model.

5.1.1 Syntactic or semantic integration

The majority of systems biology model annotation tools such as those described in Section 1.3.4 rely on syntactic data integration methods. Taverna workflows can act as query translation services to pull annotation into models from data sources [48]. SemanticSBML [51] provides MIRIAM annotations, but does not create any other model elements. LibAnnotationSBML connects to and serves data from a large number of Web services useful for model annotation but has only a minimal user interface [49]. BioNetGen combines its own rules of molecular interactions with BioPAX-compliant data and the Virtual Cell editor to produce SBML [50]. However, BioNetGen requires some level of manual intervention and neither adds MIRIAM annotations nor integrates disparate data formats, instead accepting only BioPAX-formatted data. Chapter 3 describes Saint, which uses query translation and an intuitive user interface to pull information from multiple Web services and syntactically integrate the results, which the user can then select and apply to a pre-existing model [47].

However, while large amounts of data can be gathered and queried in this manner, the syntactic integration process is limited to resolving formatting differences without attempting to address semantic heterogeneity. While existing syntactic tools for model annotation are suitable for many tasks, semantic data integration methods can resolve differences in the meaning of the data and theoretically provide a more useful level of integration. A number of semantic integration approaches have been applied to many different research areas [206, Section 9], and are beginning to be used in the life sciences (see also Sections 1.6 and 1.5).

Formal ontologies are useful data integration tools, as they make the semantics of terms and relations explicit [94]. They also make knowledge accessible to both humans and machines, and provide access to automated reasoners. By modelling source data and the core domain of interest as ontologies, semantically-meaningful mapping can be used to link them. Such mappings are not possible with hard-coded syntactic conversions. The use of ontologies provides a rich integration environment, but can be costly with regards to the time taken to analyse and reason over the data. The primary cost is the scalability of ontology-based integration methods. As ontologies grow in size and complexity, the reasoning time rapidly increases.

5.1.2 Mapping type for the integration interface

As described in Section 1.6.3, neither information linkage nor direct mapping meets the needs of this project. Information linkage is not true integration, and direct mapping is not suitable for situations where more than two resources need to be integrated. EKoM [155] and SBPAX [41] are two semantic direct mapping approaches discussed in Section 1.6.3 which most closely match the goals of rule-based mediation. However, modification of either of these approaches is not feasible, as they were both developed to integrate only two resources.

The integrated data in EKoM [155] is presented in RDF rather than OWL. The high scalability of RDF compared with OWL comes at the expense of expressivity. Although large amounts of data can be efficiently stored and queried with RDF, the RDF query language SPARQL cannot make use of OWL, thus limiting the possible query space. There are other limitations to the EKoM method: the addition of new data in different formats would require the modification of the main ontology, and perhaps even the stored queries used to retrieve information from it; and there is no mention of any ability to export results using one of the import formats, as is possible with rule-based mediation.

SBPAX, originally developed to integrate SBML and BioPAX, seems to provide much of the required annotation functionality for systems biology models. Extension of this method to more than two ontologies by expansion of the role of the bridging ontology initially sounds feasible. However, SBPAX was not created to be a generic methodology and the latest release (SBPAX3) has a different purpose from previously published versions[7]. Instead of being created for purely integrative purposes, SBPAX3 has moved away from ontology mapping and is instead a proposed extension to BioPAX Level 4 to incorporate quantitative information into that standard. While SBPAX3 does allow closer alignment between the SBML XML format and the BioPAX OWL format, its primary purpose has moved from data integration to extension of BioPAX functionality, making this methodology highly unlikely to incorporate additional data sources in future.

Because of the limitations with non-mediator methods, a mediator methodology has been used for rule-based mediation. Within the three mediator types described in Section 1.6.3.3, multiple mediator mapping was rejected due to its complexity; this mapping subtype requires that the users, whether machine or human, be aware of the multiple data models. As a result semantic solutions using this approach, such as linked data or RDF triple stores, are not integrated enough to be useful for this project.

A single unified view makes examination, querying and ultimately viewing the integrated dataset easier. This leaves either single mediator mapping or hybrid mediator mapping. Hybrid mediator mapping was rejected as it requires that each source ontology be aware of the global ontology, thus immediately making it impossible to utilise ontologies with no knowledge of rule-based mediation such as pre-existing community ontologies. Therefore single mediator mapping, which has the benefit of being a straightforward mapping method with minimal pre-requisites, was chosen.

There are three subtypes within single mediator mapping: global-as-view, local-as-view, and hybrid approaches. While the global-as-view approach makes the passing of queries to the underlying data sources simple, it has other restrictions which make it unsuitable for model annotation. For instance, defining the global ontology merely as a model of a set of data sources creates an ontology which is brittle with respect to the addition of new data sources and new formats. However, choosing local-as-view can also be problematic, as query complexity increases quickly with local-as-view approaches, and data sources need to be forced into the structure of the global ontology [207].

Both methods are limited in their ability to describe the nuances of the biological concepts being studied. When one layer (either the global or source ontologies) is a view of the other, conflation of format and biological domain concepts into a single model can occur. For global-as-view, the source

[7]http://sbpax.org/sbpax3/index.html

ontologies are intended to model the data sources, information which is of no use in a global ontology intended to be a model of a biological domain of interest, as in rule-based mediation. For local-as-view, the source ontologies are merely views over a global ontology, restricting the description of the syntax of the sources.

Therefore, hybrid approaches which do not force one layer to be the view of another provide a more flexible solution. The BYU Global-Local as View (BGLaV) [207] project describes a hybrid mapping method where the global ontology is modelled completely separately from the source ontologies. However, its overall methodology depends upon complex query translation algorithms to pull information from the underlying sources [207]. Therefore rule-based mediation uses this hybrid mapping type but creates a new methodology using existing Semantic Web mapping strategies which does not have the query complexity of BGLaV.

5.1.3 The integration interface: warehousing or federation

The final refinement to the methodology is the integration interface itself, and the decision as to whether a warehousing or federated approach is appropriate for rule-based mediation. While warehousing methods (described fully in Section 1.6.4.1) are powerful and quick, they are not scalable and can create large resources which current ontological reasoning tools struggle to process. The Neurocommons integration methodology, which uses data encapsulation via multiple mediation mapping, partially solves these scalability issues [167]. While OWL is used to re-format the original data sources, Neurocommons minimises the reasoning cost by using plain RDF as the format for the main data store, which does not get reasoned over. Further, while Neurocommons makes use of Semantic Web technologies, they integrate at the level of the database entry itself [208] and do not try to model the biological concepts referenced by those entries. As the focus of rule-based mediation is on providing modellers with a method of targeted annotation of specific models, such a large triple store is not required, and a methodology generating a smaller, more semantically-rich dataset amenable to reasoning is required.

Another ontology-based model comparison system has recently been developed by Schulz and colleagues to integrate the semantic MIRIAM annotations in SBML across different models in the BioModels database [209]. While this method is useful for finding semantically similar entities in SBML, it is not intended as a large-scale, multiple data source integration methodology.

As described in Section 4.4, SBML Harvester allows the loading and querying of SBML models in OWL. Both rule-based mediation and SBML Harvester apply automated reasoning to enrich the knowledge available to an SBML model. SBML Harvester also makes use of an upper-level ontology that integrates *in silico* and *in vivo* information and includes a number of community ontologies [205]. However, the conflation of both structural and biological entities in a single ontology could make it more difficult to clearly isolate syntactic from semantic heterogeneity. Additionally, Hoehndorf and colleagues have stated that the existing biomedical ontologies used within the system often have a weak formalisation which limits the capabilities of SBML Harvester for verification and knowledge extraction [205]. The authors have expressed a desire to see if rule-based mediation and SBML Harvester can be combined in future [205].

Data federation (see Section 1.6.4.2) provides its own challenges. Federated approaches require constant calls to remote Web sources to retrieve the requested subset of information and are not guaranteed to be able to access source data. Older federated integration techniques such as TAMBIS use only a single ontology, mapping both the semantics and syntax of the underlying data sources into that ontology [182]. TAMBIS also assumes only one data source per format type. In contrast, rule-based mediation separates the tasks of resolving syntactic and semantic heterogeneity through the use of source ontologies as well as a core ontology.

Query translation systems such as ComparaGRID suffer from inefficiency during data retrieval. With local-as-view methods such as OntoFusion, input data can only be modelled as a view over the global ontology. As the global ontology is independent of the data sources, it may not have the expressivity to model all of the information in every format of interest. In PISCEL [158], class and query rewriting were used to pull information from each data source at the time of the query. However, in rule-based mediation the core ontology is a tightly-scoped ontology which benefits from having all of the relevant information stored directly within it, thus obviating the need for a query rewriting step.

Large-scale ontology alignment has very recently been performed using a variety of OBO Foundry ontologies by aligning each ontology to a upper-level philosophical ontology [94]. While this method has found numerous modelling errors, inconsistencies, and contradictions in the aligned ontologies, integration is limited to the knowledge within the ontologies themselves and is not attempted for any instance-level data or biological databases. It provides valuable error checking across ontologies which are intended to be orthogonal according to OBO Foundry principles[8], but is a methodology tailored to ontologies without instance data, making it unsuitable for this thesis.

The integrated data in rule-based mediation needs stability to ensure effective querying, but also needs to be lightweight enough that reasoning can be performed. As such, rule-based mediation has aspects of both warehousing and federation. In rule-based mediation, instance data is stored in

[8]http://www.obofoundry.org/wiki/index.php/FP_005_delineated_content

the core ontology for as long as a user wishes, creating a warehouse of integrated data. However, the user selects only a subset of each data source to be loaded into the core ontology. The query results are then filtered through the source ontologies and mapped to the core. This query step can be performed repeatedly, adding or removing information from the global ontology as required, thus providing a federated quality to the methodology. Because the core ontology in rule-based mediation is a tightly-scoped ontology modelling a specific biological domain of interest, only the information required for a particular use-case needs to be added.

5.1.4 Ontology mapping rules

SWRL [123] is a Semantic Web rule language that extends OWL with the addition of a new *if-then* conditional and a number of convenience methods called *built-ins* [210]. The DL-safe version of SWRL is used within rule-based mediation to create mappings, also called rules, from one entity to another. SWRL has mature libraries and is interoperable with the existing OWL API. Although it is possible to hard code rules using a library such as the OWL API, SWRL rules allow the logical separation of the knowledgebase and the rulebase from any lower-level code, make knowledge shareable across ontologies, and make the knowledge easier to maintain [211].

SWRL rules are divided into a *antecedent* and a *consequent*. In the simple case below, both the antecedent ($classA(?X)$) and the consequent ($classB(?X)$) are classes, though SWRL is often used for more complex assertions involving properties, classes and even instances:

$$classA(?X) \Rightarrow classB(?X)$$

In SWRL, whenever the antecedent of the rule is true, the conditions in the consequent must also hold. The above mapping in plain English is "if X is a member of *classA*, then X is a member of *classB*". With this mapping, all instances $?X$ of *classA* are also asserted as members of *classB*. *classA* and *classB* could be from the same or different ontologies. SWRL rules provide only *instance-to-instance* mapping, which results in the assignment of instance data to membership in another class. No classes or properties are ever modified by SWRL.

SWRL is not the only way to create rules in OWL. Two or more OWL ontologies can be linked together with a number of methods. In ontology alignments, classes can be marked as equivalent or joined together via other, more complex, axioms (see Section 1.6.3 for a discussion of ontology alignment and merging). This method does not require any languages other than OWL itself, and is easily visible within the asserted hierarchy. However, each aligned ontology must be imported into the same file; imports can cause problems if the ontologies are large or numerous, and it is impractical if only part of the ontology needs to be aligned. DL rules [212] are pure OWL and are a fragment of SWRL. However, the use of complete rule languages such as SWRL is often easier than OWL rules and can sometimes be a requirement for expressing particular mappings [9].

Although multiple implementations for ontology mapping exist, ultimately only the SWRLTab [213] plugin for Protégé 3.4 and the OWL API were used within rule-based mediation. SWRLTab is a Protégé 3.4 plugin which can perform ontology mapping as well as query relational data sources or XML, and was used within rule-based mediation to create and execute SWRL rules. SWRL is used for the mapping rules and SQWRL [214] for querying those rules. While SWRL rules are written using OWL and are therefore easy to understand, there are drawbacks to this implementation. SWRLTab requires that each additional ontology to be mapped must be imported directly into the main ontology. Therefore when mapping many source ontologies to a single core ontology, the amount of imports could become unwieldy. Further, importing via the normal OWL import mechanism means that multiple dissimilar ontologies are combined in one logical area. However, once the SWRL rules have been run, a cleaner version of the final ontology—without the source ontologies or the rule file—may be viewed if desired.

Other ontology mapping applications were investigated and ultimately deemed inappropriate for rule-based mediation. Snoggle[10] is a graphical, SWRL-based ontology tool for aiding alignment of OWL ontologies. Ontology alignment is mainly used for merging two ontologies and as such is not ideally suited to rule-based mediation, which retains the independence of each ontology. Similarly, Chimera[11] is intended for alignment and manipulation, but not mapping. COMA++[12] currently only supports OWL-Lite. While capable of limited OWL functionality, COBrA [215] is primarily for OBO files, specifically GO. Falconer and colleagues provide an overview of a number of other ontology mapping tools [216].

Closer to the requirements of this rule-based mediation are PROMPT [206] and Fluxion[13]. PROMPT is a Protégé plugin which performs automatic mapping and allows user-provided manual mappings. It is a mature application with the ability to map classes, relations, and instances. However, PROMPT could not provide any automated mappings between the BioGRID syntactic ontology and the telom-

[9]http://answers.semanticWeb.com/questions/1354/why-do-we-need-swrl-and-rif-in-an-owl2-world, http://answers.semanticWeb.com/questions/2329/rifspin-needed-or-not-as-when-owl2-seems-sufficient
[10]http://snoggle.semWebcentral.org/
[11]http://www-ksl.stanford.edu/software/chimaera/
[12]http://dbs.uni-leipzig.de/Research/coma.html
[13]http://bioinf.ncl.ac.uk/comparagrid/

ere ontology. Fluxion is a standalone ontology mapping tool which has been used within the ComparaGrid system, and is a service-oriented architecture for federated semantic data integration which uses translation rules to map between the source ontologies and the core ontology[14]. However, Fluxion is not currently in development and cannot be easily integrated with the overall structure of rule-based mediation.

5.1.4.1 Querying an ontology

The simplest method of ontology querying is the creation of defined classes which function as queries over the instance data, as with Luciano and colleagues [26]. Alternatively, SPARQL queries can be constructed for ontologies which have an underlying RDF. However, both methods have their limitations. Defined classes created as queries are difficult to add on-the-fly, and have similar limitations the use of plain OWL does for the creation of rules. SPARQL is only compatible with RDF-formatted ontologies. SQWRL is a query language based on SWRL which is used for querying OWL ontologies and formatting the query response. Unlike SWRL, SQWRL does not modify the underlying ontology. Therefore, SQWRL queries can be useful for testing SWRL mappings as well as running queries that should not be stored in the ontology.

In general, queries over mapped ontologies can be performed either directly on data materialised and stored in the core ontology, or by reformulating a core ontology query as a set of queries over the data stored in the source ontologies. Within rule-based mediation, SQWRL queries are executed when subsets of the knowledge stored within the core ontology need to be extracted.

5.2 Source Ontologies

In rule-based mediation, all ontologies which move data in and out of the core ontology are called *source ontologies*. Information flow between source and core ontologies is bi-directional and can be transitive, as translation between data sources via the core ontology is also possible. This section describes the source ontologies used to implement rule-based mediation for the model annotation use cases in Section 5.5.

A source ontology can be either a purpose-built *syntactic ontology* or a pre-existing ontology. Pre-existing source ontologies have no expressivity restrictions; community developed or other pre-existing ontologies which represent a field only tangentially related to the domain of interest can

[14]http://bioinf.ncl.ac.uk/comparagrid/

be used without modification. Mapping an existing ontology requires only a set of SWRL rules to connect it to the core ontology.

Information from one or more non-OWL data sources sharing the same format can be recast as a syntactic ontology, isolating syntactic variability so the core ontology is free to model the semantics of the relevant biology. Most syntactic ontologies are intended to be a useful representation of a format in OWL. Such a representation should be semantically lightweight and simple and quick to build. Strictly, it might be argued that syntactic ontologies are not true ontologies, being more precisely defined as a list of axioms. This is because syntactic ontologies are built as a simple conversion of a non-OWL format into OWL, with no additional semantics. There is little reason to enrich the syntactic ontologies when the majority of the inference, reasoning and querying will take place within the core ontology. Syntactic ontologies are a type of application ontology intended, not to be a reference for a particular biological domain of interest, but to facilitate a larger bioinformatics application.

Four data sources were used for the implementation of rule-based mediation presented in this chapter: BioGRID [217], exported in PSI-MIF [56]; the Pathway Commons [218] BioPAX Level 2 export; BioModels [21], in SBML format; and UniProtKB [1] in XML format. For each data format, a suitable source ontology was identified or syntactic ontology was created. BioModels was the data source for the SBML models to be annotated, while the other three resources were used as general data inputs to the integration system. Basic information on the data formats as well as the numbers of classes, relations and mapping rules in their source ontologies is available in Table 5.1.

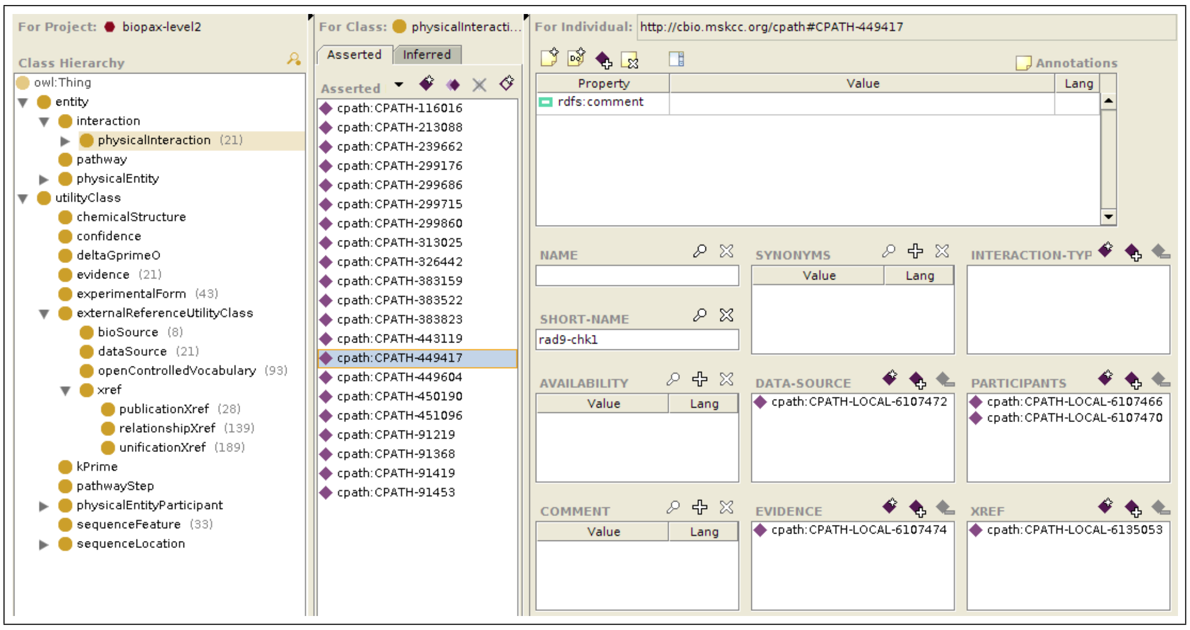
Data Source	Data Format	Classes	Object Properties	Data Properties	DL Expressivity	Rules
UniProtKB	UniProtKB XML	27	24	34	$\mathcal{ALEN}(\mathcal{D})$	11
BioGRID	PSI-MIF	26	24	15	$\mathcal{ALEN}(\mathcal{D})$	17
SBML	SBML	648	56	44	$\mathcal{SHOIN}(\mathcal{D})$	14*
Pathway Commons	BioPAX	41	33	37	$\mathcal{ALCHN}(\mathcal{D})$	11

Table 5.1: Basic information about the source ontologies and their mapping rules to the core ontology. Pathway Commons differs from the other sources as an OWL format is already available. For the SBML syntactic ontology (MFO), the numbers are a summary of the metrics for both SBO and MFO. The *Rules* column lists the number of SWRL mapping statements used to link each source ontology with the core ontology. MFO was used for export only, and therefore its rules are marked with "*" to show that these are export rules. Statistics generated using Protégé 4.

5.2.1 Pathway Commons and BioPAX

Two data sources, UniProtKB and Pathway Commons, provide a data export in a format accessible to SWRL and OWL. For BioPAX, the creation of a specific syntactic ontology was not required. For the UniProtKB, XML rather than RDF was chosen for the reasons outlined in Section 5.2.2.3 below.

Pathway Commons integrates nine pathway and interaction databases[15]. Information on network neighbours of any given entity can be accessed as a BioPAX Level 2 document, written in OWL. Pathway Commons provides both binary interaction data and pathway data. BioPAX can store information about which species are reactants, products or modifiers, but the data coming from Pathway Commons does not always provide this information. As this information is available in BioPAX, no additional syntactic ontology needs to be created; the populated BioPAX ontology returned from the nearest neighbour query *is* the source ontology.



```
owl:Thing
▼ entity
    ▼ interaction
        ▶ physicalInteraction (21)
        pathway
    ▶ physicalEntity
▼ utilityClass
    chemicalStructure
    confidence
    deltaGprimeO
    evidence (21)
    experimentalForm (43)
    ▼ externalReferenceUtilityClass
        bioSource (8)
        dataSource (21)
        openControlledVocabulary (93)
        ▼ xref
            publicationXref (28)
            relationshipXref (139)
            unificationXref (189)
    kPrime
    pathwayStep
    ▶ physicalEntityParticipant
    sequenceFeature (33)
    ▶ sequenceLocation
```

```
cpath:CPATH-116016
cpath:CPATH-213088
cpath:CPATH-239662
cpath:CPATH-299176
cpath:CPATH-299686
cpath:CPATH-299715
cpath:CPATH-299860
cpath:CPATH-313025
cpath:CPATH-326442
cpath:CPATH-383159
cpath:CPATH-383522
cpath:CPATH-383823
cpath:CPATH-443119
cpath:CPATH-449417
cpath:CPATH-449604
cpath:CPATH-450190
cpath:CPATH-451096
cpath:CPATH-91219
cpath:CPATH-91368
cpath:CPATH-91419
cpath:CPATH-91453
```

For Individual: http://cbio.mskcc.org/cpath#CPATH-449417

Property	Value	Lang
rdfs:comment		

SHORT-NAME: rad9-chk1
DATA-SOURCE: cpath:CPATH-LOCAL-6107472
PARTICIPANTS: cpath:CPATH-LOCAL-6107466, cpath:CPATH-LOCAL-6107470
EVIDENCE: cpath:CPATH-LOCAL-6107474
XREF: cpath:CPATH-LOCAL-6135053

Figure 5.3: Screenshot of Protégé 3.4 showing part of a Pathway Commons interaction set in BioPAX Level 2 format.

5.2.2 Novel syntactic ontologies

BioModels and BioGRID do not export their data in a format amenable to the application of SWRL rules, and therefore syntactic ontologies were created based on the export format to which each data source was best suited. This required the use of MFO for BioModels entries and the creation of a syntactic ontology based on PSI-MIF. Additionally, the UniProtKB XML format needed to be converted to OWL as the RDF export was deemed inappropriate for the use cases.

[15]http://www.pathwaycommons.org/pc/dbSources.do

The use of existing tools to implement rule-based mediation increases its usability for other researchers as well as decreases the development time. Protégé 3.4 and 4 were used to edit and view all ontologies used in this project, while the syntactic ontologies were generated using the XMLTab plugin [201] for Protégé 3.4. Once the syntactic ontologies are created, further editing may be performed as needed. Specifically, MFO was modified beyond the simple conversion created by XMLTab, as described in Chapter 4.

The XMLTab plugin takes only a few seconds to create syntactic ontologies. Additionally, if an XML file is provided instead of an XSD, then both classes and instances are generated: the classes represent structural elements and attributes present in the XML file, while the actual data is created as instances within the ontology. The resulting OWL file is an exact duplicate of the XML structure, which is acceptable for the rule-based mediation methodology as semantic heterogeneity is resolved when the data is mapped from the syntactic ontology to the core ontology.

However, XMLTab does not scale well, and cannot load multiple XML files into a single syntactic ontology. Additionally, it is not in active development. However, tools such as the SWRLAPI's XMLMapper [219] or Krextor [220] may provide viable alternatives. Quality assurance and normalisation is not relevant with most syntactic ontologies, as they are direct syntax conversions and do not bear the greatest reasoning load in rule-based mediation. As such, the most important feature is that they are quick to generate and easy to produce, rather than being highly expressive or optimized for reasoning.

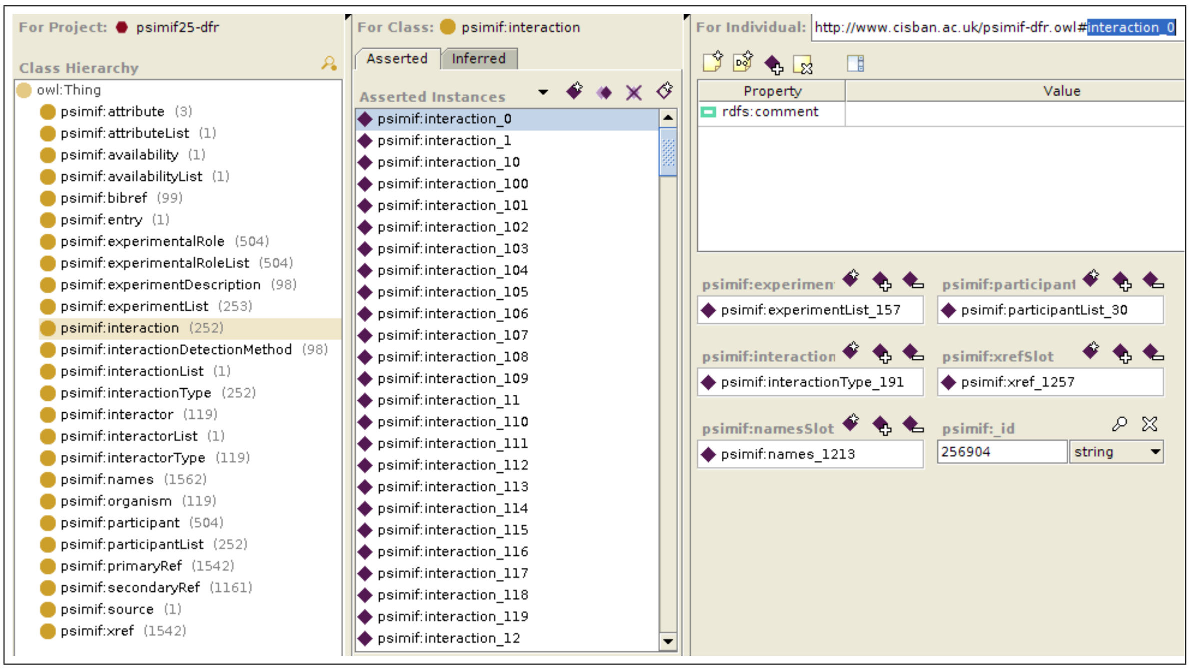
5.2.2.1 Model Format OWL

MFO is an ontology representing constraints from SBML, SBO and the SBML Manual [199] (see also Chapter 4). MFO is used as one of the source ontologies, and has a dual purpose for the presented use cases: firstly, it acts as the format representation for any data stored as SBML such as entries from the BioModels database, and secondly it is used to format query responses as SBML.

BioModels is an example of a single data source that may be loaded into a core ontology from more than one source ontology. BioModels entries are available in SBML format as well as in BioPAX and CellML. However, BioModels uses SBML as its native format, and conversion from SBML to other formats is not lossless. For example, quantitative data such as rate laws are not supported in BioPAX. Therefore when manipulating BioModels data, SBML—and therefore MFO—is the most suitable.

5.2.2.2 PSI-MIF

The PSI-MIF syntactic ontology was created from its native XML. BioGRID stores 24 different types of interactions. For the purposes of this work, its physical protein-protein interactions were used. BioGRID data is available in PSI-MI format (known as PSI-MIF) [56], and can easily be converted into a simple syntactic ontology. The PSI-MIF file exported by BioGRID was used to generate a simple syntactic ontology using XMLTab in Protégé 3.4. Figure 5.4 shows an overview of the classes created for the PSI-MIF syntactic ontology, while Figure 5.5 shows a selection of the axioms applied to the PSI-MIF *interaction* class.


For Project: psimif25-dfr

Class Hierarchy

- owl:Thing
  - psimif:attribute (3)
  - psimif:attributeList (1)
  - psimif:availability (1)
  - psimif:availabilityList (1)
  - psimif:bibref (99)
  - psimif:entry (1)
  - psimif:experimentalRole (504)
  - psimif:experimentalRoleList (504)
  - psimif:experimentDescription (98)
  - psimif:experimentList (253)
  - psimif:interaction (252)
  - psimif:interactionDetectionMethod (98)
  - psimif:interactionList (1)
  - psimif:interactionType (252)
  - psimif:interactor (119)
  - psimif:interactorList (1)
  - psimif:interactorType (119)
  - psimif:names (1562)
  - psimif:organism (119)
  - psimif:participant (504)
  - psimif:participantList (252)
  - psimif:primaryRef (1542)
  - psimif:secondaryRef (1161)
  - psimif:source (1)
  - psimif:xref (1542)

For Class: psimif:interaction

Asserted | Inferred

Asserted Instances

- psimif:interaction_0
- psimif:interaction_1
- psimif:interaction_10
- psimif:interaction_100
- psimif:interaction_101
- psimif:interaction_102
- psimif:interaction_103
- psimif:interaction_104
- psimif:interaction_105
- psimif:interaction_106
- psimif:interaction_107
- psimif:interaction_108
- psimif:interaction_109
- psimif:interaction_11
- psimif:interaction_110
- psimif:interaction_111
- psimif:interaction_112
- psimif:interaction_113
- psimif:interaction_114
- psimif:interaction_115
- psimif:interaction_116
- psimif:interaction_117
- psimif:interaction_118
- psimif:interaction_119
- psimif:interaction_12

For Individual: http://www.cisban.ac.uk/psimif-dfr.owl#interaction_0

Property	Value
rdfs:comment	

psimif:experimen: psimif:experimentList_157

psimif:participant: psimif:participantList_30

psimif:interaction: psimif:interactionType_191

psimif:xrefSlot: psimif:xref_1257

psimif:namesSlot: psimif:names_1213

psimif:_id: 256904 string

Figure 5.4: Overview of the PSI-MIF syntactic ontology. This syntactic ontology can be used for BioGRID data, and was created by the Protégé XMLTab plugin using the PSI-MIF-formatted result of a query over BioGRID.

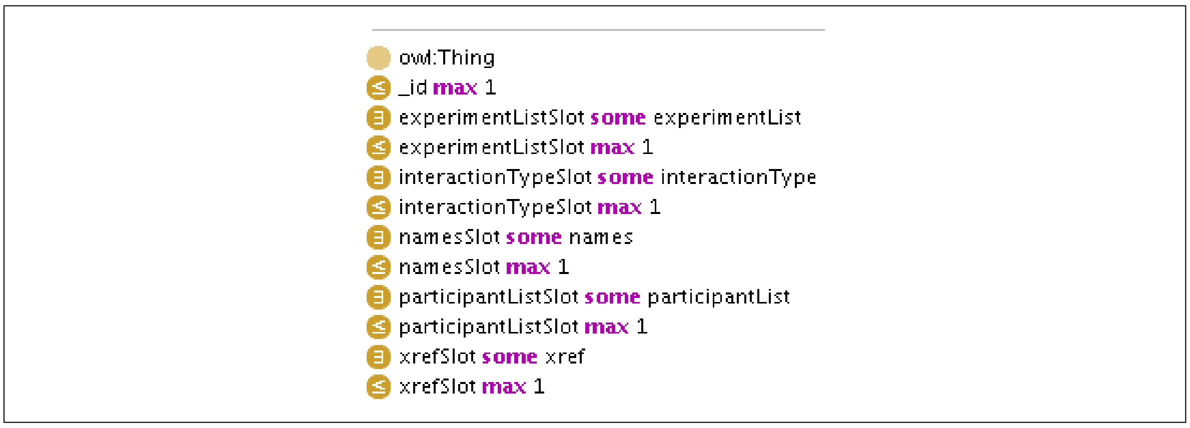

owl:Thing
_id **max** 1
experimentListSlot **some** experimentList
experimentListSlot **max** 1
interactionTypeSlot **some** interactionType
interactionTypeSlot **max** 1
namesSlot **some** names
namesSlot **max** 1
participantListSlot **some** participantList
participantListSlot **max** 1
xrefSlot **some** xref
xrefSlot **max** 1

Figure 5.5: Axioms created for the PSI-MIF syntactic ontology's *interaction* class.

As this work progressed, Pathway Commons began importing BioGRID data and can therefore can be used to retrieve BioGRID data in BioPAX format rather than requiring the PSI-MIF syntactic ontology. However, Pathway Commons does not store the genetic interactions available from BioGRID. The incorporation of BioGRID data into Pathway Commons does not reduce the effectiveness of the PSI-MIF syntactic ontology; the PSI-MIF syntactic ontology can be used to integrate any other PSI-MIF-based data sources, and it can also be used to integrate those portions of BioGRID which are not stored within Pathway Commons. The integration discussed in the use cases in Section 5.5 made use of the PSI-MIF syntactic ontology, while a Web application currently under development uses the BioPAX output from Pathway Commons.

5.2.2.3 UniProtKB

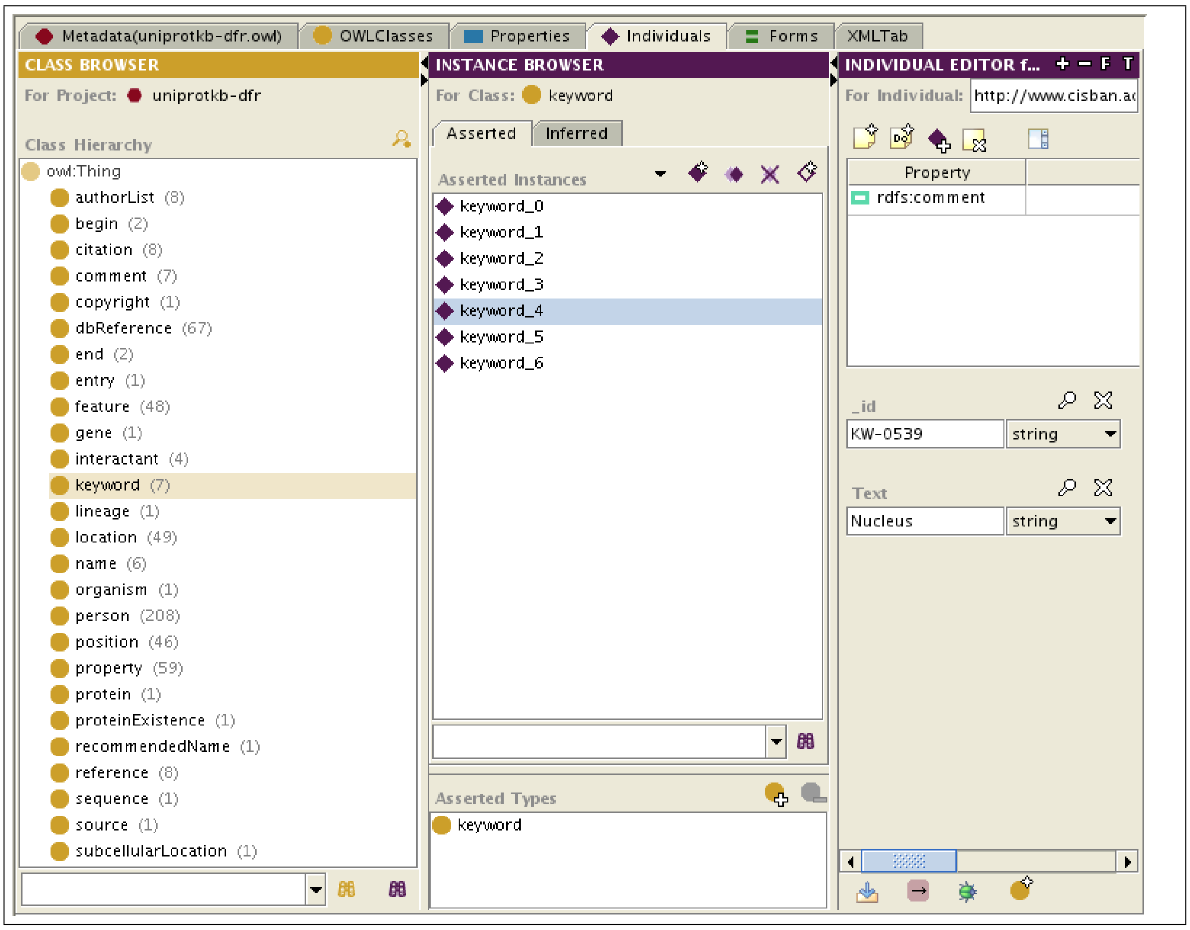
The UniProtKB is a comprehensive public protein sequence and function database, consisting both of manually-curated and automatically-annotated data. Adding UniProtKB as a data source provides access to a wealth of information about proteins, although the reaction data is more limited. As evidenced by the non-curated models stored within the BioModels database, new systems biology models are often created with only a small amount of biological information such as species names and skeleton reactions. In these cases, even the simple addition of cross-references to UniProtKB is useful. Unambiguous assignment of a UniProtKB primary accession to a new species in a systems biology model also provides the core ontology with high-quality knowledge of protein localization and identification.

While the original intent was to make use of the UniProtKB RDF format for rule-based mediation, reasoning over the RDF was slow and there was a large number of ontology and RDF files to import. Instead, the UniProtKB XML format was used for the initial work. At the beginning of this research, UniProtKB Release 14.8 XML files appropriate for the use cases were downloaded[16]. The UniProtKB syntactic ontology was created using the XMLTab plugin for Protégé 3.4. Figure 5.6 shows an excerpt of this ontology. The UniProtKB syntactic ontology was successfully used for the initial use cases detailed in Section 5.5.

A rule-based mediation Web application is being developed, where the set of relevant entries will be expanded from that required for the use cases to the entire UniProtKB database. As these larger sets of XML files can no longer easily be manually converted via the XMLTab plugin, the UniProtKB RDF syntax was revisited as a data source for the rule-based mediation Web application. The

[16]ftp://ftp.uniprot.org/pub/databases/uniprot/knowledgebase/

UniProtKB RDF format, together with the structure provided by the UniProtKB OWL ontology, is suitable for the application. The RDF syntax is accessible to SWRL rules without modification, and the slow reasoning times are irrelevant as reasoning over syntactic ontologies is not required with rule-based mediation. However, limitations remain in the UniProtKB RDF format.



For Project: uniprotkb-dfr

Class Hierarchy

- owl:Thing
  - authorList (8)
  - begin (2)
  - citation (8)
  - comment (7)
  - copyright (1)
  - dbReference (67)
  - end (2)
  - entry (1)
  - feature (48)
  - gene (1)
  - interactant (4)
  - keyword (7)
  - lineage (1)
  - location (49)
  - name (6)
  - organism (1)
  - person (208)
  - position (46)
  - property (59)
  - protein (1)
  - proteinExistence (1)
  - recommendedName (1)
  - reference (8)
  - sequence (1)
  - source (1)
  - subcellularLocation (1)

For Class: keyword

Asserted Instances: keyword_0, keyword_1, keyword_2, keyword_3, keyword_4, keyword_5, keyword_6

Asserted Types: keyword

For Individual: http://www.cisban.ac

Property: rdfs:comment

_id: KW-0539 string

Text: Nucleus string

Figure 5.6: Overview of the UniProtKB syntactic ontology. This syntactic ontology can be used for UniProtKB data, and was created by the Protégé XMLTab plugin using an XML file representing a UniProtKB entry. Screenshot taken from Protégé 3.4.

Firstly, the core UniProtKB OWL file is not guaranteed to be decidable. This is because its expressivity level is OWL Full and not one of the less expressive, decidable profiles such as OWL DL. If an ontology is not decidable, there is no guarantee that a reasoner can successfully be run over the ontology. The expressivity level, or the OWL profile, of the core UniProtKB OWL file[17] was tested using both the Manchester OWL Validator[18] and the WonderWeb OWL Ontology Validator[19]. The former stated that, while the UniProtKB OWL file conformed to OWL 2, it did not conform any decidable profiles of OWL 2. The WonderWeb validator produced a similar result, stating that the

[17] `ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/rdf/core.owl`, file downloaded and test performed on 16 October 2011.
[18] `http://owl.cs.manchester.ac.uk/validator/`
[19] `http://www.mygrid.org.uk/OWL/Validator`

file was OWL Full.

Secondly, the full model of the UniProtKB structure is stored in core file plus a further nine files, all ten of which must be imported to have a complete model[20]. The nine non-core files describe those parts of the UniProtKB structure which might be reused across multiple UniProtKB entries, and include topics such as citations, keywords and pathways. As each of these files contain *all* of the entities required by the entire UniProtKB dataset, many are quite large. This design requires the import of all ten files to completely view or process just one UniProtKB entry, creating a heavyweight method of modelling. For example, the RDF document containing all citations is 1.4 Gigabytes in size, making it difficult for many users to even load the document into memory for access by a reasoner.

5.3 Mapping between source and core ontologies

Once a source ontology has been created, it must be mapped to the core, biologically-relevant ontology. The core ontology (Section 5.4) is the mediator for rule-based mediation, and holds all of the biologically-meaningful statements that need to be made about the data to be integrated.

The mappings between the source ontologies and the core ontology are not as straightforward as the conversion from a data format to its syntactic ontology, and they are not created in the same way. The mapping in rule-based mediation is defined with the OWL-based SWRL rule language, which divides rules into a antecedent for describing the initial conditions of the rule and a consequent for describing the target of the rule (see Section 5.1.4). When mapping from the source ontologies to the core ontology, as a general rule the antecedent of the rule comes from a source ontology while the consequent corresponds to the mapped concept within the core ontology. The antecedent and consequent can be named classes or non-atomic classes such as those expressed by properties. In each of the rules presented in this section, the right arrow is a convenience rather than standard OWL notation, and the namespaces for each entity denote which ontology or library they belong to. For example, in the rule below, the antecedent is the *protein* class from the UniProtKB syntactic ontology, and the consequent is the *Protein* class from the telomere ontology[21]:

UP_00001 : *upkb:protein(?**protein**)* ⇒ *tuo:Protein(?**protein**)*

Many of these rules could be merged together, having one rule perform much of the mapping currently executed by many. However, that would reduce the flexibility now present in the system. With

[20] ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/rdf/README

[21] The "tuo" namespace for the telomere ontology is a result of an earlier naming scheme, where the telomere ontology was called the telomere uncapping ontology.

each rule performing just one part of the mapping, each mapping step may be run in isolation. Further, having each rule accomplish just one task aids understanding and readability of the rules for humans. Readability is also aided with the addition of comments for each rule within the SWRL mapping file.

Once instances have been integrated into the core ontology, the knowledge contained within the core ontology can be reasoned over and queried. SQWRL is used to query the core ontology, and the query response can then be formatted according to a suitable source ontology. Queries are used to illustrate interesting, but only indirectly-relevant queries without cluttering the core ontology. More precise mapping rules use SWRL to allow storage of the results within the core ontology. The mapping between source and core ontologies can be bi-directional, thus allowing output of information as well as input. The quality of the data exported to a source ontology and from there to a base format is dependent upon both the quality of the mappings to the source ontology and its scope: if either the mappings are incomplete, or the underlying data format has no way of describing the new data, the export will be lacking.

5.3.1 Rules for the UniProtKB syntactic ontology

This section describes the SWRL rules of the UniProtKB syntactic ontology and what sort of information those rules map to the telomere ontology. The UniProtKB rule set is different from the other data sources' rules. While UniProtKB contains a large amount of information, it is protein-centric rather than reaction-centric. More importantly for these use cases rich information about a protein, described in other databases with much less detail, is available to the telomere ontology via this data source. Specifically, information regarding protein localization within the cell as well as synonyms of gene and protein names are available. This rule set is simple, with many of the classes required for the use cases having direct equivalents in telomere ontology, and with many cardinalities of the properties exactly matching those in the telomere ontology itself. While the UniProtKB does contain some limited information on reactions via the comment and cross-reference sections, these are not currently modelled by the mapping rules for the UniProtKB syntactic ontology.

Protein identification

The first three rules in the UniProtKB syntactic ontology populate the *Protein* class with the protein instance itself as well as with a primary name and synonym:

UP_00001 :

*upkb:protein(?**protein**)* ⇒

*tuo:Protein(?**protein**)*

UP_00002 :

*upkb:recommendedNameSlot(**?protein**, ?name)* ∧

*upkb:fullName(?name, **?fullName**)* ⇒

*tuo:recommendedName(**?protein**, **?fullName**)*

UP_00003 :

upkb:entry(?entry) ∧

*upkb:proteinSlot(?entry, **?protein**)* ∧

upkb:geneSlot(?entry, ?gene) ∧

upkb:nameSlot(?gene, ?geneName) ∧

*upkb:Text(?geneName, ?**geneNameText**)* ⇒

*tuo:synonym(**?protein**, **?geneNameText**)*

As it is a primary data source for protein information, UniProtKB is the only source ontology which populates the *recommendedName* property within the telomere ontology. All other data sources are mapped to the *synonym* data property. In UP_00003, the first part of the rule contains only the statement *entry(?entry)*. From a computational point of view, this extra statement is superfluous, as the next part of the rule (*proteinSlot(?entry, ?protein)*) ensures the correct type of *?entry* instance is used. However, the extra statement in this and other rules aids readability of those rules.

Protein localisation

If the location of a protein is the nucleus, then the instance is mapped directly to the *Nucleus* class. Otherwise, the instance is mapped to the more generic *BiologicalLocalization* class:

UP_00004 :

*upkb:locationSlot(?subcellularLocation, **?location**)* ∧

upkb:Text(?location, ?locationText) ∧

swrlb:equal(?locationText, "Nucleus") ⇒

*tuo:Nucleus(**?location**)*

UP_00011 :

*upkb:locationSlot(?subcellularLocation, **?location**)* ∧

upkb:Text(?location, ?locationText) ∧

swrlb:notEqual(?locationText, "Nucleus") ⇒

*tuo:BiologicalLocalization(**?location**)*

The two rules above use SWRL built-ins. The *swrlb:equal* and *swrlb:notEqual* built-ins test whether two arguments are equivalent. The test for equality is satisfied if and only if the first argument and the second argument are identical, while the *swrlb:notEqual* is the simple negation of *swrlb:equal* [123].

As well as creating instances for each biological location, linking those locations to a specific protein also occurs. In UP_00006, the subcellular location of the protein is mapped directly onto the *isLocated* property within the telomere ontology.

UP_00006 :

upkb:entry(?entry) ∧

*upkb:proteinSlot(?entry, **?protein**)* ∧

upkb:commentSlot(?entry, ?comment) ∧

upkb:subcellularLocationSlot(?comment, ?subcellularLocation) ∧

*upkb:locationSlot(?subcellularLocation, **?location**)* ⇒

*tuo:isLocated(**?protein**, **?location**)*

Taxonomic identification and other database cross-references

The next three rules describe how the cross references of a protein are mapped from UniProtKB to the telomere ontology. UP_00005 creates an instance of *TaxonomicSpecies* with a specific taxonomic identifier. Conversely, UP_00007 is used to direct all other UniProtKB references to the more generic *DatabaseReference* from the telomere ontology. Both *swrlb:equal* and *swrlb:notEqual* have been used in these rules to determine whether or not a particular reference is a taxonomic identifier. At this stage, the information is not yet connected to a protein. UP_00008 provides this functionality, linking a particular protein with its taxonomic reference.

UP_00005 :

*upkb:dbReference(**?reference**)* ∧

upkb:_type(?reference, ?referenceType) ∧

swrlb:equal(?referenceType, "NCBI Taxonomy") ∧

*upkb:_id(?reference, **?id**)* ⇒

*tuo:TaxonomicSpecies(**?reference**)* ∧

*tuo:ncbiTaxId(**?reference**, **?id**)*

UP_00007 :

*upkb:dbReference(**?reference**)* ∧

*upkb:_type(?reference, **?referenceType**)* ∧

*upkb:_id(?reference, **?id**) swrlb:notEqual(?referenceType, "NCBI Taxonomy")* ⇒

*tuo:DatabaseReference(**?reference**)* ∧

*tuo:databaseName(?reference, **?referenceType**)* ∧

*tuo:accession(?reference, **?id**)*

UP_00008 :

upkb:entry(?entry) ∧

*upkb:proteinSlot(?entry, **?protein**)* ∧

upkb:organismSlot(?entry, ?organism) ∧

*upkb:dbReferenceSlot(?organism, **?reference**)* ⇒

*tuo:hasTaxon(**?protein**, **?reference**)*

UP_00009 has a structure virtually identical to UP_00008, but rather than mapping the organism references for a particular protein instance, it maps all associated *database* references to their equivalents in the telomere ontology.

UP_00009 :

upkb:entry(?entry) ∧

*upkb:proteinSlot(?entry, **?protein**)* ∧

*upkb:dbReferenceSlot(?entry, **?reference**)* ⇒

*tuo:hasDatabaseReference(**?protein**, **?reference**)*

UP_00010 uses a SWRL built-in called *swrlx:makeOWLThing* to create a new *DatabaseReference* for each UniProtKB entry. Rather than being part of the W3C Submission, built-ins with the namespace *swrlx* are experimental convenience methods. *swrlx:makeOWLThing* takes an empty variable as its first argument, and will create and bind a new instance to that variable for every instance of the second argument matched within the rule[22]. The SWRL built-in does *not* fill the value of the first argument with the actual instance or instances present in later arguments; it simply uses the second and later arguments as a statement of how many new instances to create. The associated UniProtKB

[22] http://protege.cim3.net/cgi-bin/wiki.pl?SWRLExtensionsBuiltIns

accession number is used as the value for the new instance's *accession* data property, while the database reference name is given as "UniProtKB".

> UP_00010 :
> *upkb:entry(?entry)* ∧
> *upkb:proteinSlot(?entry, ?protein)* ∧
> *upkb:accession(?entry,* ***?accessionValue****)* ∧
> *swrlx:makeOWLThing(****?reference****, ?entry)* ∧
> *swrlb:equal(****?referenceValue****, "UniProtKB")* ⇒
> *tuo:hasDatabaseReference(?protein, ?reference)* ∧
> *tuo:DatabaseReference(****?reference****)* ∧
> *tuo:accession(?reference,* ***?accessionValue****)* ∧
> *tuo:databaseName(?reference,* ***?referenceValue****)*

5.3.2 Rules for the PSI-MIF syntactic ontology

This section describes the SWRL rules of the PSI-MIF syntactic ontology, and what sort of information those rules map to the telomere ontology. This ontology is centred around binary interactions, with some extra information provided regarding taxonomy and database cross-references. Unlike BioPAX, BioGRID provides only binary interactions, and as such the information modelled in the BioGRID syntactic ontology is only of this interaction type.

One-to-one mappings

As with BioPAX, there is a high degree of convergence between the telomere ontology and BioGRID. BioPAX, BioGRID and the telomere ontology all model interactions in a similar way. This is exemplified by the five classes within the BioGRID syntactic ontology which are mapped directly to classes within the telomere ontology. PSIMIF_00001 is shown here, and PSIMIF_00003-6 are described in later subsections.

> PSIMIF_00001 :
> *psimif:interactor(****?interactor****)* ⇒
> *tuo:PhysicalEntity(****?interactor****)*

The convergence is further visible in rules PSIMIF_00002, PSIMIF_00010 and PSIMIF_00012-14 (see subsections below), where the classes and the properties described have direct one-to-one mappings with classes and properties in the telomere ontology. Although the one-to-one mapping rules could be accomplished with standard OWL axioms, it is useful to keep all cross-ontology mapping logically separate from the telomere ontology itself. Not only does this allow changes to the rules without changing the telomere ontology itself, but it also collects all mappings in one location for improved readability.

Identification

Two rules in the BioGRID syntactic ontology populate the *Protein* class with instance data as well as with synonyms. Unlike UniProtKB, which only stores protein entities, BioGRID can have a number of different types of interactor. Therefore, in order to only retrieve those interactors which are proteins, PSIMIF_00015 is restricted to those *interactor* instances which are of the "protein" type. Note that no *recommendedName* is populated in the telomere ontology via the BioGRID syntactic ontology; only the UniProtKB syntactic ontology provides information for that property. All other source ontologies provide information for the *synonym* data property only.

PSIMIF_00015 :
psimif:fullName(?name, ?nameText) ∧
swrlb:equal(?nameText, "protein") ∧
*psimif:interactor(**?interactor**)* ∧
psimif:interactorTypeSlot(?interactor, ?interactorType) ∧
psimif:namesSlot(?interactorType, ?name) ⇒
*tuo:Protein(**?interactor**)*

PSIMIF_00011 :
*psimif:interactor(**?interactor**)* ∧
psimif:namesSlot(?interactor, ?names) ∧
*psimif:shortLabel(?names, **?nameText**)* ⇒
*tuo:synonym(**?interactor**, **?nameText**)*

Taxonomic identification and other database cross-references

The two rules below assign the taxonomy instance from BioGRID to the telomere ontology *TaxonomicSpecies*, and then assign the mapped taxonomic instance to the appropriate interactor. In PSIMIF_00002, as in other rules, object and data properties for all PSI-MIF *interator* instances are mapped rather than just those for protein instances, as in PSIMIF_00015. While this will generate more properties within the telomere ontology then are required, no extraneous *PhysicalEntity* instances are created. General-purpose rules such as these are simpler, and do not contribute to a large increase in computational complexity.

PSIMIF_00002 :
*psimif:organism(**?organism**) ∧*
*psimif:_ncbiTaxId(?organism, **?taxId**) ⇒*
*tuo:TaxonomicSpecies(**?organism**) ∧*
*tuo:ncbiTaxId(?organism, **?taxId**)*

PSIMIF_00010 :
*psimif:interactor(**?interactor**) ∧*
*psimif:organismSlot(?interactor, **?organism**) ⇒*
*tuo:hasTaxon(?interactor, **?organism**)*

All non-taxonomic cross-references are identified in PSI-MIF with the *primaryRef* or *secondaryRef* classes. These, and the names of the databases themselves, are mapped to the telomere ontology's *DatabaseReference* class in rules PSIMIF_00005-6 and PSIMIF_00012. Finally, in PSIMIF_00013-14 the identifiers of each of the individual cross-references are mapped to the *accession* property in the telomere ontology.

PSIMIF_00005 :
*psimif:primaryRef(**?reference**) ⇒*
*tuo:DatabaseReference(**?reference**)*

PSIMIF_00006 :
*psimif:secondaryRef(**?reference**) ⇒*
*tuo:DatabaseReference(**?reference**)*

PSIMIF_00012 :
*psimif:_db(**?reference**, **?referenceName**) ⇒*
*tuo:databaseName(**?reference**, **?referenceName**)*

PSIMIF_00013 :
psimif:primaryRef(?primaryReference) ∧
*psimif:_id(**?primaryReference**, **?id**) ⇒*
*tuo:accession(**?primaryReference**, **?id**)*

PSIMIF_00014 :
psimif:secondaryRef(?secondaryReference) ∧
*psimif:_id(**?secondaryReference**, **?id**)* ⇒
*tuo:accession(**?secondaryReference**, **?id**)*

After the cross-references are mapped, they need to be tied to the individual *PhysicalEntity* instances which reference them. The link between a cross-reference and a PSI-MIF entity occurs with the *xrefSlot* property and either the *primaryRefSlot* or the *secondaryRefSlot*. These properties are simplified and mapped across to the *hasDatabaseReference* property in the telomere ontology.

PSIMIF_00016 :
*psimif:xrefSlot(**?entity**, ?crossRef)* ∧
*psimif:primaryRefSlot(?crossRef, **?primaryRef**)* ⇒
*tuo:hasDatabaseReference(**?entity**, **?primaryRef**)*

PSIMIF_00017 :
*psimif:xrefSlot(**?entity**, ?crossRef)* ∧
*psimif:secondaryRefSlot(?crossRef, **?secondaryRef**)* ⇒
*tuo:hasDatabaseReference(**?entity**, **?secondaryRef**)*

Binary interactions

Although less sophisticated than BioPAX, BioGRID defines interactions in a similar way. The participant in a binary interaction and the physical entity that is referenced by that participant are two separate entities. However, unlike BioPAX, BioGRID reactions are only expressed as binary interactions of varying types, with no concept of complex reactions containing multiple inputs or outputs. Additionally, there is no way to determine which entity is a product or modifier of the interaction. Therefore, PSIMIF_00003 maps all participants in the binary interactions to the telomere ontology's *Reactant* class. In PSIMIF_00004 all interactions, of whatever type, are mapped to the telomere ontology's *PhysicalProcess* class. In PSIMIF_00007 the instances of the "ReconstitutedComplex" interactions are then further mapped to the *ProteinComplexFormation* class in the telomere ontology.

PSIMIF_00003 :
*psimif:participant(**?participant**)* ⇒
*tuo:Reactant(**?participant**)*

PSIMIF_00004 :

*psimif:interaction(**?interaction**)* ⇒

*tuo:PhysicalProcess(**?interaction**)*

PSIMIF_00007 :

*psimif:interaction(**?interaction**)* ∧

psimif:interactionTypeSlot(?interaction, ?interactionType) ∧

psimif:namesSlot(?interactionType, ?interactionName) ∧

psimif:shortLabel(?interactionName, ?nameText) ∧

swrlb:equal(?nameText, "ReconstitutedComplex") ⇒

*tuo:ProteinComplexFormation(**?interaction**)*

The link between the interactor and its participant in BioGRID is modeled with the *interactorRef* property. In the telomere ontology, this connection is made with the *plays* property:

PSIMIF_00008 :

*psimif:interactor(**?interactor**)* ∧

psimif:_id(?interactor, ?id) ∧

*psimif:participant(**?participant**)* ∧

psimif:interactorRef(?participant, ?id) ⇒

*tuo:plays(**?interactor**, **?participant**)*

Finally, PSIMIF_00009 is used to link an interaction to its participants:

PSIMIF_00009 :

*psimif:participantListSlot(**?interaction**, ?list)* ∧

*psimif:participantSlot(?list, **?participant**)* ⇒

*tuo:hasReactant(**?interaction**, **?participant**)*

5.3.3 Rules for BioPAX Level 2

One-to-one mapping and interactions

As described in Section 5.2.1, Pathway Commons provides information in BioPAX Level 2 format. The simplicity of the mapping between BioPAX Level 2 and the telomere ontology is due to the similar modelling of physical processes within them. As described in Section 5.3.2 with respect to other simple mappings, rules such as BP_00001-5 could be written in pure OWL but are written in SWRL to keep rules separate from ontology modelling. These rules identify the direct equivalents in the telomere ontology for each relevant class and property within BioPAX. Specifically, BP_00001-5 provide mappings for BioPAX proteins, reaction participants and interactions.

BP_00001 :
*bp:protein(**?protein**)* ⇒
*tuo:Protein(**?protein**)*

BP_00002 :
*bp:physicalEntityParticipant(**?participant**)* ⇒
*tuo:Participant(**?participant**)*

BP_00004 :
*bp:physicalInteraction(**?interaction**)* ⇒
*tuo:PhysicalProcess(**?interaction**)*

BP_00003 :
*bp:PHYSICAL-ENTITY(**?participant**, **?physicalEntity**)* ⇒
*tuo:playedBy(**?participant**, **?physicalEntity**)*

BP_00005 :
*bp:PARTICIPANTS(**?interaction**, **?participant**)* ⇒
*tuo:hasReactant(**?interaction**, **?participant**)*

Protein identification

Rules BP_00008-9 map the *name* and *synonym* data properties present within a BioPAX *protein* to the telomere ontology *synonym* data property. As described in Section 5.3.1, only UniProtKB entries populate the *recommendedName* data property; all names and synonyms from other source ontologies map directly to the synonyms within the telomere ontology.

BP_00008 :
*bp:protein(**?protein**)* ∧
*bp:NAME(?protein, **?value**)* ⇒
*tuo:synonym(**?protein**, **?value**)*

BP_00009 :
*bp:protein(**?protein**)* ∧

*bp:SYNONYMS(?protein, **?value**) ⇒*

*tuo:synonym(**?protein**, **?value**)*

Taxonomic identification and other database cross-references

BP_00006 fills the *TaxonomicSpecies* class as well as the *ncbiTaxId* data property within the telomere ontology. In order to fill this value, only those *unificationXref* classes with a database whose name is “NCBI” are used. The identifier for this *unificationXref* is then used to fill the *ncbiTaxId* property. BP_00007 then links a *Protein* in the telomere ontology to its taxonomic species.

BP_00006 :
*bp:unificationXref(**?xref**) ∧*
bp:DB(?x, ?value) ∧
*bp:ID(?x, **?id**) ∧*
swrlb:equal(?value, “NCBI”) ⇒
*tuo:TaxonomicSpecies(**?xref**) ∧*
*tuo:ncbiTaxId(?xref, **?id**)*

BP_00007 :
*bp:protein(**?protein**) ∧*
bp:ORGANISM(?protein, ?organism) ∧
*bp:TAXON-XREF(?organism, **?ref**) ⇒*
*tuo:hasTaxon(**?protein**, **?ref**)*

BP_00010 fills the *DatabaseReference* class in the telomere ontology as well as the properties connecting the *DatabaseReference* to a database name and accession. Taxonomic identifiers are specifically not included in this rule, as they have already been parsed by BP_00006-7. BP_00011 then links the protein to the mapped *DatabaseReference* class via the *hasDatabaseReference* property. *unificationXref* instances are the only ones mapped from BioPAX, as they show equivalence between the BioPAX entity and the database reference specified. There are other reference types within BioPAX, however such cross references link to data that is related, but not identical, to the entity in question.

BP_00010 :
*bp:unificationXref(**?xref**) ∧*
*bp:DB(?xref, **?dbvalue**) ∧*

bp:ID(?xref, ***?idvalue****)* $\wedge$

swrlb:notEqual(?dbvalue, "NCBI") $\Rightarrow$

*tuo:DatabaseReference(****?xref****)* $\wedge$

tuo:databaseName(?xref, ***?dbvalue****)* $\wedge$

tuo:accession(?xref, ***?idvalue****)*

BP_00011 :

*bp:protein(****?protein****)* $\wedge$

bp:XREF(?protein, ***?xref****)* $\wedge$

bp:unificationXref(?xref) $\Rightarrow$

*tuo:hasDatabaseReference(****?protein****,* ***?xref****)*

5.3.4 Rules for MFO

There is a clear delineation in most systems biology formalisms between a protein as an abstract concept and a protein as a specific interactor in a reaction. Within SBML, this is defined using the speciesType or species elements for the concept of a protein, and speciesReference for the participant in the reaction. Within BioPAX Level 2, the biological entity / participant pairing is modelled with *physicalEntity* and *physicalEntityParticipant*. Within BioGRID, interactor and participant elements are used.

A similar dichotomy exists within the telomere ontology, and when converting the information retrieved in Use Case 1 back to SBML, these considerations need to be taken into account. One solution is to link the telomere ontology's *Protein* class (and its parent class *PhysicalEntity*) to the SBML speciesType element. However, most SBML models do not use speciesType, and future versions of SBML will have a different way of modelling such concepts [221]. Therefore, to align with future versions of SBML as well as to make the mapping simple, the instances of the telomere ontology *PhysicalEntity* class are instead linked to the appropriate attributes of an SBML species element.

Biological annotation

MIRIAM annotation is added to MFO by mapping a variety of database cross-references from the telomere ontology. The rules used to create these annotations introduce two additional SWRL built-ins, *swrlb:stringEqualIgnoreCase* and *swrlb:stringConcat*. The *swrlb:stringEqualIgnoreCase* built-in is satisfied if and only if the two arguments match in a case-insensitive manner, while the first argument in *swrlb:stringConcat* is filled with a concatenation of all further arguments [123].

In the first two rules for mapping information from the telomere ontology to MFO (TUO_MFO_00001 and TUO_MFO_00001_synonym), all *Rad9* instances are enhanced with two biological annotations: firstly, a MIRIAM URI for the UniProtKB entry associated with the *Rad9* protein and secondly, the addition of a link to the SBO "macromolecule" term (SBO_0000245)[23]. This association is double-checked by ensuring that the "rad9" species name is already present. This is a rule which returns information—required for the use cases—on the Rad9 protein only. While this is a highly specific rule, it can easily be generalised to run over *Protein*, the parent class of *Rad9*. In such a general case, shown later for TUO_MFO_00006 and TUO_MFO_00006_synonym, all of the references to specific MFO *species* instances have been removed.

TUO_MFO_00001 :
tuo:Rad9(?protein) ∧
tuo:hasDatabaseReference(?protein, ***?crossref****)* ∧
tuo:databaseName(?crossref, ?dbname) ∧
tuo:accession(?crossref, ?accession) ∧
swrlb:stringEqualIgnoreCase(?dbname, "uniprot") ∧
*swrlb:stringConcat(****?miriamIs****, "urn:miriam:uniprot:", ?accession)* ∧
*sbo:SBO_0000245(****?term****)* ∧
*mfo:Species(****?mfoSpecies****)* ∧
mfo:name(?mfoSpecies, ?nameText) ∧
swrlb:stringEqualIgnoreCase(?nameText, "rad9") ⇒
*mfo:MiriamAnnotation(****?crossref****)* ∧
*mfo:miriamAnnotation(****?mfoSpecies****, ?crossref)* ∧
mfo:bqbIs(?crossref, ***?miriamIs****)* ∧
mfo:sboTerm(?mfoSpecies, ***?term****)*

Additional MIRIAM annotations are added with TUO_MFO_00002, which adds any Saccharomyces Genome Database (SGD) cross-references to MFO for *Rad9* instances. As with TUO_MFO_00001 and TUO_MFO_00001_synonym, this rule could easily be made more generic to allow all MFO *species* to be annotated with their relevant cross-references. As MIRIAM URIs require a prefix

[23]For brevity, the second rule (TUO_MFO_00001_synonym) is not shown. It is identical to the first rule other than the *swrlb:stringEqualIgnoreCase* compares against "uniprotkb" rather than "uniprot". Different source datasets use different names for this database.

identifying the referenced database (e.g. “urn:miriam:uniprot:”), each database requires its own rule to correctly build the URI. Rules TUO_MFO_00003-4 (not shown) differ from TUO_MFO_00001-2 only in the database used to create the new MIRIAM annotation; TUO_MFO_00003 for Intact and TUO_MFO_00004 for Pathway Commons.

TUO_MFO_00002 :
tuo:Rad9(?protein) ∧
tuo:hasDatabaseReference(?protein, ***?crossref****)* ∧
tuo:databaseName(?crossref, ?dbname) ∧
tuo:accession(?crossref, ?accession) ∧
swrlb:stringEqualIgnoreCase(?dbname, “sgd”) ∧
*swrlb:stringConcat(****?miriamIs****, “urn:miriam:sgd:”, ?accession)* ∧
*mfo:Species(****?species****)* ∧
mfo:name(?species, ?nameText) ∧
swrlb:stringEqualIgnoreCase(?nameText, “rad9”) ⇒
*mfo:MiriamAnnotation(****?crossref****)* ∧
*mfo:miriamAnnotation(****?species****, ?crossref)* ∧
mfo:bqbIs(?crossref, ***?miriamIs****)*

In TUO_MFO_00006 and TUO_MFO_00006_synonym, for every *Protein* in the telomere ontology, a UniProtKB MIRIAM annotation is created. [24]. In addition to the UniProtKB MIRIAM annotations within MFO for each telomere ontology *Protein*, TUO_MFO_00007-9 add MIRIAM annotations for SGD, Intact and Pathway Commons. Due to their similarity with TUO_MFO_00006, for brevity they are not shown.

TUO_MFO_00006 :
*tuo:Protein(****?mfoSpecies****)* ∧
mfo:Species(?mfoSpecies) ∧
tuo:hasDatabaseReference(?mfoSpecies, ?crossref) ∧
tuo:databaseName(?crossref, ?dbname) ∧
tuo:accession(?crossref, ?accession) ∧
swrlb:stringEqualIgnoreCase(?dbname, “uniprot”) ∧
swrlb:stringConcat(?miriamIs, “urn:miriam:uniprot:”, ?accession) ∧

[24]TUO_MFO_00006_synonym is not shown in this section as it is identical to TUO_MFO_00006 except for the string comparison *swrlb:stringEqualIgnoreCase*, which instead uses the literal “uniprotkb”

sbo:SBO_0000245(?term) ⇒

*mfo:MiriamAnnotation(****?crossref****)* ∧

*mfo:miriamAnnotation(****?mfoSpecies****, ?crossref)* ∧

mfo:bqbIs(?crossref, ***?miriamIs****)* ∧

mfo:sboTerm(?mfoSpecies, ***?term****)*

TUO_MFO_00011-12 provide *name* properties for the *Species* instances within MFO. As SWRL rules will not overwrite existing name values, this rule will only modify the mapped *name* property if it is empty. This ensures that existing values are not overwritten, and further ensures—if the rules are run in sequence—that names from the *recommendedName* property are not overridden by those from the *synonym* property. If pre-existing names from the MFO model should be re-written, they can simply be deleted prior to or during conversion from SBML to MFO.

TUO_MFO_00011 :

*tuo:Protein(****?spec****)* ∧

mfo:Species(?spec) ∧

tuo:recommendedName(?spec, ***?recName****)* ⇒

*mfo:name(****?spec****,* ***?recName****)*

TUO_MFO_00012 :

*tuo:Protein(****?spec****)* ∧

mfo:Species(?spec) ∧

tuo:synonym(?spec, ***?synonym****)* ⇒

*mfo:name(****?spec****,* ***?synonym****)*

Rad9-Rad17 Interaction (TUO_MFO_00005)

Specific interactions relating to *Rad9* were retrieved from the telomere ontology and mapped to MFO. *Rad9-Rad17* is described in this section, and *Rad9-Mec1* is described in the next. TUO_MFO_00005 maps the *Rad9-Rad17* interaction, and is tailored to populating a single SBML model. Otherwise, if multiple models are stored within MFO, incorrect *listOfReactions* and *listOfSpecies* could be matched. Where necessary, this limitation can easily be resolved by extending the rule to specify the *ListOf__* instances associated with a particular systems biology model.

As TUO_MFO_00005 is a long and complex rule, it is presented in sections rather than all at once. Further, in contrast to the native ordering in the OWL files themselves, portions of the rule have been reordered slightly for increased readability. Firstly the reaction to be mapped, the reactants involved, and the proteins associated with those reactants within the telomere ontology are identified:

> TUO_MFO_00005 :
> *tuo:ProteinComplexFormation(?reaction)* ∧
> *tuo:Rad9Rad17Interaction(?reaction)* ∧
> *tuo:hasReactant(?reaction, ?reactant)* ∧
> *tuo:playedBy(?reactant, ?protein)* ∧

Next, the already-extant *ListOfReactions* and *ListOfSpecies* instances from MFO are stored in SWRL variables for later use:

> *mfo:ListOfReactions(?listOfReaction)* ∧
> *mfo:ListOfSpecies(?listOfSpecies)* ∧

SWRL built-ins are then called to create seven new variables. Firstly, *?nameVariable* is filled with the value "Rad9Rad17Complex" using *swrlb:stringConcat*; this is a string literal only, and is not an instance itself. *?nameVariable* is used later in the rule to create a value for the *name* data property within MFO. Six calls to *swrlx:makeOWLThing* are then run. These calls create the appropriate number of MFO instances for storing the information related to the reactions and proteins matched within this rule. For instance, if there are *X* instances of the *Protein* class matched within TUO_MFO_00005, then *X* new instances are created and bound to the variable *?speciesId*. As the *Rad9-Rad17* interaction is a protein complex formation, one product—the complex itself—is created for each reaction. The newly-created instance variable *?product* describes the new product of the reaction. The other new variables create additional parts of an SBML model which are not directly modelled within the telomere ontology.

> *swrlb:stringConcat(?nameVariable, "Rad9Rad17Complex")* ∧
> *swrlx:makeOWLThing(?reactantList, ?reaction)* ∧
> *swrlx:makeOWLThing(?reactionSId, ?reaction)* ∧
> *swrlx:makeOWLThing(?speciesSId, ?protein)* ∧
> *swrlx:makeOWLThing(?product, ?reaction)* ∧
> *swrlx:makeOWLThing(?prodSpecSid, ?reaction)* ∧
> *swrlx:makeOWLThing(?prodRef, ?reaction)* ∧
> *swrlx:makeOWLThing(?productList, ?reaction)*

These SWRL built-ins mark the end of the antecedent for TUO_MFO_00005, which has created or identified all variables needed to map the reaction data to MFO. The consequent begins with assignment of all of the identifiers to *SId* instances in MFO.

> $\Rightarrow$
>
> *mfo:SId(?reactionSId)* $\wedge$
>
> *mfo:SId(?speciesSId)* $\wedge$
>
> *mfo:SId(?prodSpecSid)* $\wedge$

Then, each of the links to those *SId* instances is built using the *id* object property. Additionally, the *name* property is filled for the *?product* instance:

> *mfo:id(?reaction, ?reactionSId)* $\wedge$
>
> *mfo:id(?protein, ?speciesSId)* $\wedge$
>
> *mfo:id(?product, ?prodSpecSid)* $\wedge$
>
> *mfo:name(?product, ?nameVariable)* $\wedge$

Next, each of the instances named in the first argument of the *id* property are mapped to their appropriate MFO classes. Additionally, the *Reaction* instance gets a mapped list of reactants and products and both the *Species* and *Reaction* instances are linked to their appropriate *ListOf__* instances.

> *mfo:Reaction(?reaction)* $\wedge$
>
> *mfo:Species(?protein)* $\wedge$
>
> *mfo:Species(?product)* $\wedge$
>
> *mfo:listOfReactants(?reaction, ?reactantList)* $\wedge$
>
> *mfo:listOfProducts(?reaction, ?productList)* $\wedge$
>
> *mfo:reaction(?listOfReaction, ?reaction)* $\wedge$
>
> *mfo:species(?listOfSpecies, ?protein)* $\wedge$
>
> *mfo:species(?listOfSpecies, ?product)* $\wedge$

The *ListOfSpeciesReferences* class in MFO is then populated with the appropriate new instances from the *swrlx:makeOWLThing* built-in, and each individual *SpeciesReference* is mapped and added to its list:

> *mfo:ListOfSpeciesReferences(?reactantList)* $\wedge$
>
> *mfo:ListOfSpeciesReferences(?productList)* $\wedge$

mfo:ProductSpeciesReference(?prodRef) ∧

mfo:ReactantSpeciesReference(?reactant) ∧

mfo:speciesReference(?reactantList, ?reactant) ∧

mfo:speciesReference(?productList, ?prodRef) ∧

Rad9-Mec1 Interaction (TUO_MFO_00010)

There are two interactions between *Rad9* and *Mec1* present within the telomere ontology. In this rule, one of these interactions has been arbitrarily chosen to send to MFO. This is a choice the modeller can make in future, after looking at the information contained in each interaction choice. Unlike TUO_MFO_00005, there is no extra information from the data sources which allows the interaction to be defined as being a protein complex formation. As such, a likely product for the reaction cannot be created. The lack of knowledge about the interaction means that TUO_MFO_00010 is a much simpler rule than TUO_MFO_00005. Except for those sections of the rule dealing with products and lists of products, TUO_MFO_00010 is the same as TUO_MFO_00005. As a result the reaction has reactants only, and no products.

TUO_MFO_00010 :
tuo:Rad9Mec1Interaction('psimif:interaction_143') ∧
tuo:hasReactant('psimif:interaction_143', ?react) ∧
tuo:playedBy(?react, ?protein) ∧
mfo:ListOfReactions(?listOfReaction) ∧
swrlx:makeOWLThing(?reactantList, 'psimif:interaction_143') ∧
swrlx:makeOWLThing(?reactSId, 'psimif:interaction_143') ∧
swrlx:makeOWLThing(?specSId, ?protein) ⇒
mfo:ListOfSpeciesReferences(?reactantList) ∧
mfo:SId(?reactSId) ∧
mfo:SId(?specSId) ∧
mfo:ReactantSpeciesReference(?react) ∧
mfo:speciesReference(?reactantList, ?react) ∧
mfo:Reaction('psimif:interaction_143') ∧
mfo:id('psimif:interaction_143', ?reactSId) ∧
mfo:reaction(?listOfReaction, 'psimif:interaction_143') ∧
mfo:Species(?protein) ∧

mfo:id(?protein, ?specSId) $\wedge$

mfo:listOfReactants('psimif:interaction_143', ?reactantList)

5.4 The core ontology

Often, mediator-based approaches build a core ontology as a union of source ontologies rather than as a semantically-rich description of the research domain in its own right [157, 158, 159]. While such ontologies thoroughly describe the associated data formats, a description of the structure of data is *not* a description of the biology of the data. A union of source ontologies would not produce a description of the biological domain of interest, but simply a description of the combined formats imported into the system. Further, if a core ontology is defined as merely an ontology which models a set of data sources, the core ontology becomes brittle with respect to the addition of new data sources and new formats.

In contrast, for rule-based mediation a core ontology should be a biologically-relevant, tightly-scoped, logically-rigorous description of the semantics of the research domain. Although the definition of a core ontology shares much with that of a reference ontology, a core ontology is not required to stand as a reference model for a biological domain of interest, although that may be one of its additional purposes. As originally written about BioPAX, a useful representation of the research domain should be detailed enough to express richness where required, but general enough to maintain the overall framework when few details are available [88]. The core ontology may be created for rule-based mediation or drawn from existing ontologies. However, pre-existing biomedical ontologies should be chosen with care, as many do not have sufficiently formalised semantics and as such may not be useful for automated reasoning tasks in a data integration context [94]. A core ontology is not intended to capture all of biology; instead, it is scoped tightly to its purpose, which is modelling the research domain of interest.

By creating an core ontology which is more than the entailment of a set of source ontologies, and which stands on its own as a semantically-rich model of a research domain, the core ontology becomes much more flexible with respect to changes in data sources. Because the core ontology is abstracted away from data formats, importing new data sources is made easier. Furthermore, the process of adding to the core ontology is simplified: each new mapping, class, or data import is incremental, without the need for large-scale changes. The richness of the core ontology depends on the type of biological questions that it has been created to answer and the amount of resources available; a detailed ontology may have higher coverage of the research domain, but may take longer to develop and reason over.

In contrast to global-as-view or local-as-view approaches, where either the target or source ontologies are entirely described using views of the other, the methods used in rule-based mediation provide a way to decouple syntactic and semantic heterogeneity and allow the core and source ontologies to remain distinct. Additionally, rule-based mediation maps and loads data directly into the core ontology, allowing simplified querying via the use of standard tools, mapping and querying languages. Being a semantically-rich model of a research domain, a core ontology could also be used as a standalone ontology for marking up data or as a reference model of its biological domain.

It is possible for a core ontology to incorporate other domain ontologies or upper-level ontologies. Hoehndorf and colleagues included a custom-built upper-level ontology to aid the integration of the *in vivo* and *in silico* aspects of SBML models [205]. However, as rule-based mediation uses the source and core ontologies to deliberately keep the biology and data structure separate, such an upper-level ontology is not required. Upper-level ontologies are often created to aid in the merging of ontologies with distantly-connected topics. As such, they are not as important for core ontologies, which are tied to one specific subject.

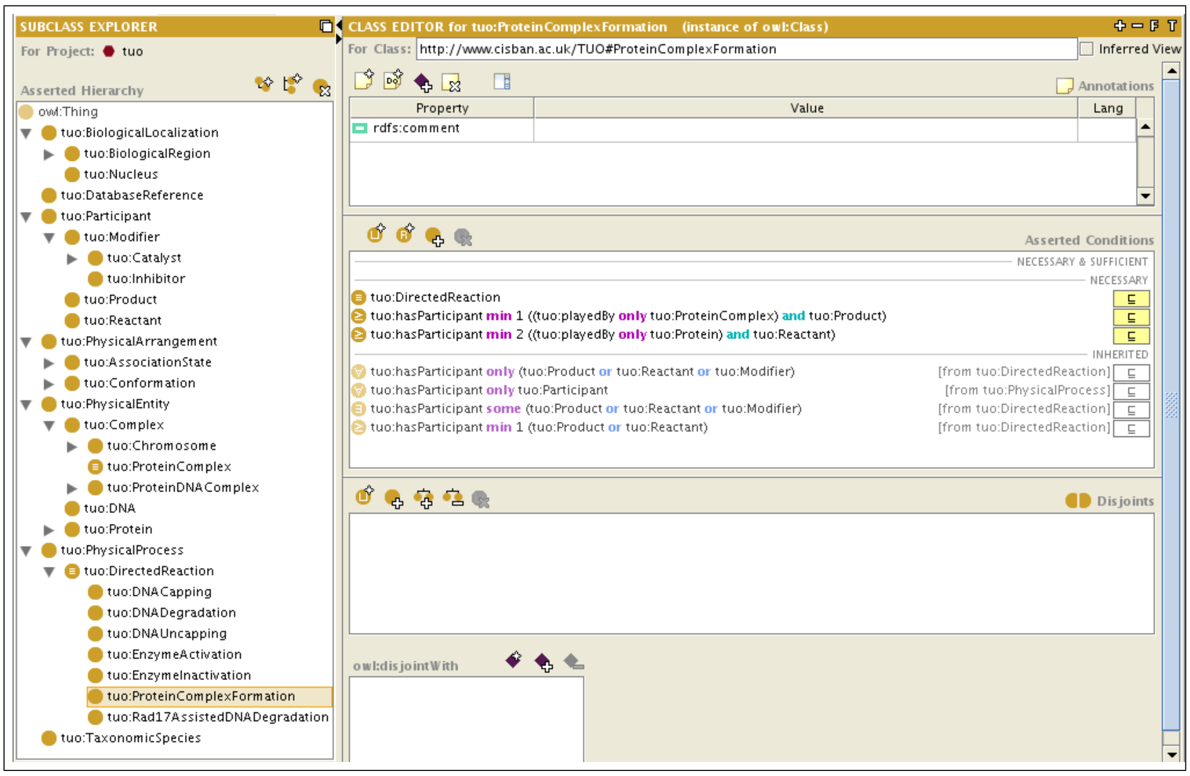
5.4.1 The Telomere Ontology

For the use cases presented in this chapter the core ontology, referred to as the telomere ontology, models a portion of the telomere binding process in *S.cerevisiae*. Specifically, it models selected parts of the biology relevant to the Proctor and colleagues model [196] of telomere uncapping. Figure 5.7 shows an overview of this ontology.

The majority of the reasoning and querying in rule-based mediation is run over the integrated data in the core ontology, allowing appropriate annotations to be queried for and selected. The results of those queries can then be exported, via a source ontology, to a non-OWL format for that data source. The use cases described in Section 5.5 illustrate the retrieval of knowledge from the telomere ontology and its application to systems biology model annotation via the MFO syntactic ontology.

5.4.2 Rules and stored queries within the telomere ontology

There are a number of rules where both the antecedent and the consequent contain entities only from the telomere ontology. These are generally convenience methods to make statements about the data coming into the ontology which otherwise would be difficult to do in plain OWL. For instance,


SUBCLASS EXPLORER

For Project: tuo

Asserted Hierarchy

```
owl:Thing
▼ tuo:BiologicalLocalization
    ▶ tuo:BiologicalRegion
      tuo:Nucleus
  tuo:DatabaseReference
▼ tuo:Participant
    ▼ tuo:Modifier
        ▶ tuo:Catalyst
          tuo:Inhibitor
      tuo:Product
      tuo:Reactant
▼ tuo:PhysicalArrangement
    ▶ tuo:AssociationState
    ▶ tuo:Conformation
▼ tuo:PhysicalEntity
    ▼ tuo:Complex
        ▶ tuo:Chromosome
          tuo:ProteinComplex
        ▶ tuo:ProteinDNAComplex
      tuo:DNA
    ▶ tuo:Protein
▼ tuo:PhysicalProcess
    ▼ tuo:DirectedReaction
          tuo:DNACapping
          tuo:DNADegradation
          tuo:DNAUncapping
          tuo:EnzymeActivation
          tuo:EnzymeInactivation
          tuo:ProteinComplexFormation
          tuo:Rad17AssistedDNADegradation
  tuo:TaxonomicSpecies
```

CLASS EDITOR for tuo:ProteinComplexFormation (instance of owl:Class)

For Class: http://www.cisban.ac.uk/TUO#ProteinComplexFormation   Inferred View

Annotations

Property	Value	Lang
rdfs:comment		

Asserted Conditions

NECESSARY & SUFFICIENT

NECESSARY

```
tuo:DirectedReaction                                                                        ⊑
tuo:hasParticipant min 1 ((tuo:playedBy only tuo:ProteinComplex) and tuo:Product)           ⊑
tuo:hasParticipant min 2 ((tuo:playedBy only tuo:Protein) and tuo:Reactant)                 ⊑
```

INHERITED

```
tuo:hasParticipant only (tuo:Product or tuo:Reactant or tuo:Modifier)   [from tuo:DirectedReaction]  ⊑
tuo:hasParticipant only tuo:Participant                                 [from tuo:PhysicalProcess]   ⊑
tuo:hasParticipant some (tuo:Product or tuo:Reactant or tuo:Modifier)   [from tuo:DirectedReaction]  ⊑
tuo:hasParticipant min 1 (tuo:Product or tuo:Reactant)                  [from tuo:DirectedReaction]  ⊑
```

Disjoints

owl:disjointWith

Figure 5.7: An overview of the telomere ontology. The telomere ontology is the core ontology for the use cases presented in this chapter.

TUO_00001-2 further specialise instances of *Protein* to be instances of *Rad9* if they have "rad9" somewhere in the recommended name or the synonym list. These rules are used within Use Case 1 to store the results of a query against the integrated dataset which places all RAD9 proteins within the *Rad9* class (see Section 5.5.1). While these rules work for relatively small datasets, the false positive rate might increase as the datasets grow. Further work on the telomere ontology could replace these rules with more precise axioms defining the classes they are modifying. Optionally, automated instance-to-class mapping (see Section 5.6.4) would eliminate the need for specific rules such as these, as would modifying the way proteins are handled such that each protein (e.g. *Rad9*) is not given its own class.

TUO_00001 :

*tuo:Protein(**?someEntity**)* ∧

tuo:synonym(?someEntity, ?synonym) ∧

swrlb:containsIgnoreCase(?synonym,"rad9") ⇒

*tuo:Rad9(**?someEntity**)*

TUO_00002 :

*tuo:Protein(**?someEntity**)* ∧

tuo:recommendedName(?someEntity, ?name) ∧
swrlb:containsIgnoreCase(?name, "rad9") ⇒
*tuo:Rad9(**?someEntity**)*

TUO_00004-11 are similar rules for four other subclasses of *Protein*: *Rad53*, *Chk1*, *Mec1* and *Rad17*. These include matching specific name such as "rad53" in TUO_00004, as well as other known synonym matches such as "YPL153C" in TUO_00004_Systematic. Only two of these rules are shown, with the others not included for brevity.

TUO_00004 :
*tuo:Protein(**?someEntity**)* ∧
tuo:synonym(?someEntity, ?synonym) ∧
swrlb:containsIgnoreCase(?synonym, "rad53") ⇒
*tuo:Rad53(**?someEntity**)*

TUO_00004_Systematic :
*tuo:Protein(**?someEntity**)* ∧
tuo:synonym(?someEntity, ?synonym) ∧
swrlb:containsIgnoreCase(?synonym, "YPL153C") ⇒
*tuo:Rad53(**?someEntity**)*

Other rules of this type were created specifically to perform the same task as the axioms present in the defined class referenced in the rules (see Use Case 2 in Section 5.5.2 for more information), thus allowing users of the ontology to see the results of the inference without needing to realise the ontology with a reasoner (see Section 1.5.7 for a description of realisation). An example of this is TUO_00012, which populates the *Rad9Rad17Interaction* and associated properties without needing to infer the placement of instances via a reasoner. Rules TUO_00013-15 (not shown) are virtually identical to TUO_00012, instead mapping the information required for *Rad9Rad53Interaction*, *Rad9Mec1Interaction* and *Rad9Chk1Interaction*.

TUO_00012 *tuo:Rad9(?rad9instance)* ∧
tuo:plays(?rad9instance, ?rad9participant) ∧
tuo:Rad17(?rad17instance) ∧
tuo:plays(?rad17instance, ?rad17participant) ∧
*tuo:hasParticipant(**?process**, ?rad9participant)* ∧

*tuo:hasParticipant(**?process**, ?rad17participant)* ⇒

*tuo:Rad9Rad17Interaction(**?process**)*

Finally, there are two stored SQWRL queries in the telomere ontology. TUO_SQWRL_00001 selects those entities with a taxonomic identifier of 4932 and which are instances of the *Rad9* class, which itself is filled via TUO_00001 and TUO_00002.

TUO_SQWRL_00001 :

*tuo:Rad9(**?someEntity**)* ∧

tuo:hasTaxon(?someEntity, ?x) ∧

tuo:ncbiTaxId(?x, ?y) ∧

swrlb:equal(?y, 4932) ⇒

*sqwrl:selectDistinct(**?someEntity**)*

TUO_SQWRL_00004 retrieves only those reactions that involve *Rad9*, and is used within Use Case 2 to retrieve specific interactions with *Rad9* instances (see Section 5.5.2).

TUO_SQWRL_00004 :

*tuo:Rad9(**?rad9instance**)* ∧

tuo:plays(?rad9instance, ?participant) ∧

*tuo:hasParticipant(**?process**, ?participant)* ⇒

*sqwrl:select(**?rad9instance**, **?process**)*

5.5 Two use cases demonstrate rule-based mediation

Rule-based mediation has been implemented for the two use cases presented in this section. A more generic Web implementation of rule-based mediation is currently under development. The use cases show how rule-based mediation extends a quantitative SBML model as a proof-of-principle for the methodology.

The context for the use cases is based on the Proctor and colleagues' *S.cerevisiae* telomere model previously described in this thesis [196]. In this way the integration results for the use cases can be compared against an already-extant, curated model. One of the key proteins within this model, RAD9, was chosen as the basis for the two use cases. In the use cases, various types of information about this query protein are retrieved. These examples highlight how rule-based mediation works in a systems biology setting and provide insight into how such a system might be extended in the future to provide automated model annotation.

Each use case shows a different aspect of the integration process. Use Case 1 shows how an existing entity within an SBML model can be augmented, and Use Case 2 describes the addition of near neighbours to the selected model species. The use cases begin with the same initial conditions. The telomere ontology described in Section 5.4.1 was used as the core ontology for the sources described in Section 5.2. Three standard Semantic Web tools (XMLTab [201], SWRLTab and SQWRLQueryTab [222]) were then used to implement the integration step via SWRL mappings. The information retrieved from the core ontology allowed the annotation of an SBML model, including the provisional identification of information not previously present in the model. These use cases demonstrate the feasibility of the approach, the applicability of the technology and its utility in gaining new knowledge.

The following subsections describe two methods for enriching the telomere model. In Use Case 1, a single SBML species is annotated with information relevant to the gene *RAD9*. Adding information to existing SBML elements at an early stage aids model development and extension prior to publication or deposition. In Use Case 2, possible protein-protein interactions are retrieved involving the RAD9 protein. This approach results in the identification of model elements as well as a putative match for an enzyme that was not identified in the original curated model. These examples show how rule-based mediation works in a systems biology setting. These use cases also show that rule-based mediation successfully integrates information from multiple sources using existing tools, and that it would be useful to expand the implementation of this method to additional biological questions and automated model annotation.

Prior to performing the queries for each use case, all mapping rules from the source ontologies to the telomere ontology were run (BP_*, PSIMIF_*, and UP_*, as described in Section 5.3). These rules populated the telomere ontology with the instance data from the source ontologies. Next, the rules to further classify the new instance data within the telomere ontology (TUO_000*) were executed. Once the execution of the SWRL rules was complete, the telomere ontology can be queried with SQWRL. To get the final results, the queries and rules specific to the use cases (TUO_MFO_* and TUO_SQWRL_*) are executed.

Protégé 3.4 and the built-in Pellet 1.5.2 reasoner were used to reason over the entire set of integrated ontologies as well as the SBML model within MFO. When running on a dual-CPU laptop running Ubuntu 11.10 with 4 Gigabytes of memory, consistency checks for the set of integrated ontologies take approximately 10-15 seconds, while computing inferred types requires approximately 9-12 minutes. Reasoning over just MFO with the annotated model from the use case takes less than 10 seconds for both consistency checks and the computation of inferred types. Finally, the MFO software toolset was used to convert the newly-annotated model from OWL to SBML.

Once instance data is loaded into telomere ontology from all of the source ontologies via the rules described in Section 5.3, reasoning as well as the application of telomere ontology specific rules can be performed for the merging of semantically identical instances, logical inconsistencies and the inference of instances to their most specific location in the hierarchy. For each use case, a query term and a taxonomic restriction has been provided. The query results are mapped from the telomere ontology to either a new or existing SBML model in MFO.

5.5.1 Use Case 1: Augmentation of existing model species

In Use Case 1, the SBML species 'rad9' is investigated. The only information provided by the modeller is the query term 'rad9' and the taxonomic species *S.cerevisiae*. In contrast to a taxonomic species, an SBML species is any physical entity that can be part of an SBML reaction. There are a number of motivations for such a query: many models are initially created as skeletons, with all species and reactions present but without accompanying data; alternatively, modellers might be searching for annotation at an even earlier stage, and want to see what is available for just one particular species. By adding information to SBML components at an early stage, modellers will not only have a more complete model, but such extra information could also aid them in developing and extending the model prior to publication or deposition in a database such as BioModels [21]. This use case showcases the ability of the methodology to add basic information to the species such as cross-references and other annotations.

Therefore in Use Case 1, the telomere ontology was queried for information about the *S.cerevisiae* gene *RAD9*. This use case demonstrates the addition of basic information to the species such as cross-references, SBO annotations, compartment localisations and a recommended name. The query for proteins containing 'rad9' as a name or synonym are stored within the telomere ontology as SWRL rules TUO_00001-2 (described initially in Section 5.4.2). Running the rules collects all matching telomere ontology *Protein* instances and stores them as instances of *Rad9*, a subclass of *Protein*.

The antecedent of TUO_00001 queries the telomere ontology *synonym* property within *Protein* for a string matching "rad9", and the consequent of the rule fills *Rad9* with any matched instances. TUO_00002 works in the same way, with *recommendedName* as the queried field. Running these SWRL rules places the following two instances under *Rad9*, each from a different data source:

```
cpath:CPATH-92332
upkb:protein_0
```

The next step is to discover equivalent instances elsewhere in the integrated schema and mark them as such. This is done by mimicking the behaviour of the OWL 2 construct `owl:hasKey` (see also Section 5.6). The end result of this procedure is that all instances with the same SGD accession will be marked as identical using the `owl:sameAs` construct. In future, this can happen automatically as the relation linking a telomere ontology *Protein* to its SGD accession number can be classed as the key for the *Protein* class in an `owl:hasKey` construct.

Until the `owl:hasKey` construct is used, those instances having the same SGD accession can be identified using a SQWRL query such as that shown below, and then manually adding the `owl:sameAs` assertions. A SQWRL is more appropriate, as SWRL rules modify the target ontology.

tuo:Protein(?someEntity) $\wedge$
tuo:hasDatabaseReference(?someEntity, ?dbref) $\wedge$
tuo:databaseName(?dbref, ?name) $\wedge$
swrlb:containsIgnoreCase(?name, "SGD") $\wedge$
tuo:accession(?dbref, ?acc) $\wedge$
swrlb:equal(?acc, "S000002625") $\Rightarrow$
sqwrl:selectDistinct(?someEntity)

There are already two instances of *Rad9* as a result of TUO_00001-2. After running the SQWRL query above and viewing the results (`upkb:protein_0` and `psimif:interactor_59`), the `owl:sameAs` construct can be applied between those two instances. The new instance, `psimif:interactor_59`, does not contain 'rad9' in its name or synonyms, but does contain a matching SGD accession. After inferring the placement of all individuals in the ontology using a reasoner, `psimif:interactor_59` is inferred as an instance of *Rad9*, bringing the total number of *Rad9* instances to three:

```
cpath:CPATH-92332
upkb:protein_0
psimif:interactor_59
```

The final missing `owl:sameAs` equivalences were then declared for these three instances, allowing the knowledge contained within each of them to be accessible as a single logical unit. The final step of this use case is to restrict the instances of *Rad9* to the specified NCBI taxonomic identifier for *S.cerevisiae*. The SQWRL query TUO_SQWRL_00001 (see Section 5.4.2) performs this function.

The information contained within these three instances is then sent out, via MFO, to a new version of the telomere model. In order to export the relevant information to MFO, a number of mapping rules with telomere ontology classes in the antecedent and MFO classes in the consequent were created (see Section 5.3.4). To create MIRIAM cross-references and SBO terms, rules TUO_MFO_00001 through TUO_MFO_00004 were run. For instance, in TUO_MFO_00001, SBO terms and all UniProtKB cross-references for *Rad9* were retrieved and reformatted according to MIRIAM guidelines, and added to the appropriate MFO *Rad9* species.

The original curated telomere model contained a single reference to UniProtKB, which was confirmed by the information contained within the telomere ontology. In addition, an SBO term and a number of other MIRIAM annotations were exported into the model for the two RAD9 species in the model. As described in Section 5.3.4, *Protein* classes from the telomere ontology are mapped directly to species elements in SBML. In Use Case 1, mapping from the telomere ontology back to MFO links annotation from the relevant *Rad9* instances back to MIRIAM annotations within the MFO *Species* class. The XML snippet below shows one of the Rad9 species after the results of this use case were exported:

```
<species metaid="_044468" id="Rad9A" name="Rad9" compartment="nucleus"
initialAmount="0" hasOnlySubstanceUnits="true" sboTerm="SBO:0000245">
  <annotation>
    <rdf:RDF>
        <bqbiol:is>
            <rdf:li rdf:resource="urn:miriam:uniprot:P14737"/>
            <rdf:li rdf:resource="urn:miriam:intact:EBI-14788"/>
            <rdf:li rdf:resource="urn:miriam:uniprot:Q04920"/>
            <rdf:li rdf:resource="urn:miriam:pathwaycommons:92332"/>
            <rdf:li rdf:resource="urn:miriam:sgd:S000002625"/>
            <rdf:li rdf:resource="urn:miriam:intact:P14737"/>
        </bqbiol:is>
    </rdf:RDF>
  </annotation>
</species>
```

All three data sources contributed to the set of information retrieved in Use Case 1. By successfully mapping data from the source ontologies into the core telomere ontology, information was retrieved regarding proteins with 'rad9' as one of their names. This information was then added to an SBML file by mapping from the telomere ontology back to MFO.

The use of the `owl:sameAs` construct for identifying equivalent instances is helpful, but caused some problems for this use case. For instance, a bug the SQWRL implementation used in this work prevented the use of any instances marked with `owl:sameAs` axioms, and these axioms were removed prior to running the queries. In addition, there is the potential for false positives in defining two instances as identical using `owl:sameAs`. For instance, BioPAX has multiple types of cross-references including *unificationXref* (for cross-references linking to semantically identical entries in the other data resources) and *relationshipXref*, which implies a less rigorous link. If the wrong type of key is created (e.g. using *relationshipXref* rather than *unificationXref*), or an imprecise rule is used, then two instances could be incorrectly marked as identical. Mapping rules must therefore be created carefully, or they could lead to false positives with respect to related, but not identical, instances.

5.5.2 Use Case 2: Augmentation by addition of near neighbours

Use Case 2 shows how information about possible protein-protein interactions involving RAD9 can be retrieved from the telomere ontology, confirming interactions already present in the curated model as well as discovering novel interactions. The retrieved information results in the proposal of new model elements as well as the putative identification of the enzyme responsible for activation of RAD9. The results of Use Case 1 are used as part of the input in this use case. TUO_SQWRL_00004 is the SQWRL query used to display, but not store, all interactions involving *Rad9* (see also Section 5.4.2).

In TUO_SQWRL_00004, the antecedent of the SQWRL query identifies interactions involving *Rad9* instances via the *Participant* class. The consequent displays the name of the instance, and the process in which it is involved. The use of SQWRL here rather than SWRL allows viewing of the more than 270 returned interactions. These results are not saved in the asserted ontology as there are too many, some of which are not meaningful for the model under study. If necessary, such unwanted results could be excluded with a small modification to the query or to the output mapping rules for MFO. Here, the interactions retrieved in TUO_SQWRL_00004 were filtered according to those species already present in the model, many of which did not have any prior involvement with RAD9. Before such relevant interactions can be found, the identity of the proteins involved must be determined. For example, just as *Rad9* instances were identified in Use Case 1, so should all other proteins of interest be identified. While these rules are currently manually generated, they could easily be automatically created and run on-the-fly. TUO_00001-11 are the SWRL rules which populates the class for these proteins of interest (see Section 5.4.2).

Discovered Interactor with RAD9 (P14737)	**Proctor *et al.***	**BioGRID**	**Pathway Commons**
Serine/threonine-protein kinase RAD53 (P22216)	✓	✓	✓
Serine/threonine-protein kinase CHK1 (P38147)	✓		✓
Serine/threonine-protein kinase MEC1 (P38111)		✓	
DNA damage checkpoint control RAD17 (P48581)		✓	
Rad9Kin (*)	✓		
ExoX (*)	✓		

Table 5.2: Summary of interactions retrieved for Use Case 2 against the telomere ontology. All discovered interactions already present in the curated model (RAD53 and CHK1) are shown, together with example interactions from the curated model that were not discovered (with Rad9Kin and ExoX) and discovered interactions that were *not* present in the model (with MEC1 and RAD17). SBML species shown with an asterisk (*) are those which are placeholder species, and therefore cannot have a match to a real protein. Some interactions are false positives inherent in the data source, while others are out of scope of the modelling domain of interest and should not be not included.

The combined results from queries TUO_00001-11 result in the appropriate proteins for this model being identified and classified. Once this is complete, specific interactions of interest can also be identified. Below are the OWL axioms on the telomere ontology defined class *Rad9Rad53Interaction*. This is one of four such defined classes which filter the *Rad9* interactions. When reasoning over instance data is performed, the appropriate *PhysicalProcess* instances which meet the criteria set out in these *necessary and sufficient* axioms are inferred to be children of *Rad9Rad53Interaction*.

```
Class: Rad9Rad53Interaction
    EquivalentTo:
        (hasParticipant some (playedBy some Rad53))
         and (hasParticipant some (playedBy some Rad9))
    SubClassOf:
        Rad9Interaction
```

Table 5.2 shows a summary of the inference results from the defined interaction classes *Rad9Rad53Interaction*, *Rad9Chk1Interaction*, *Rad9Mec1Interaction* and *Rad9Rad17Interaction*. The results may also be obtained by running TUO_00012-15 (see Section 5.4.2), or by loading the asserted ontology and re-running the reasoning. There are a total of four interactions involving RAD9 present in the curated model. Two of these, with RAD53 and CHK1, were confirmed in this use case. The RAD53 interaction was found in both BioGRID and in Pathway Commons, while the CHK1 interaction was only in Pathway Commons.

The other two RAD9 interactions are placeholder species, created by the modeller to describe an unknown protein which could not be matched further. In addition to confirming the RAD53 and

CHK1 interactions with RAD9, one of those placeholder species, marked as 'Rad9Kin' in the model, was provisionally identified as the protein MEC1 using data originating in BioGRID. In the curated model, the 'rad9Kin' species does not have a UniProtKB primary accession, as the model author did not know which protein activated RAD9. However, MEC1 is shown in the telomere ontology as interacting with RAD9 and is present elsewhere in the curated model reactions. Further, UniProtKB reports it as a kinase which phosphorylates RAD9. From this information, the model author now believes that MEC1 could be the correct protein to use as the activator of RAD9 [223]. RAD17 was added to Table 5.2 as another example of a protein present in the curated model, but not as an interacting partner of RAD9. This interaction may have been unknown to the modeller, and it may be that the curated model could be improved by the addition of a RAD9/RAD17 reaction.

After the information was retrieved within the telomere ontology, SWRL rules TUO_MFO_00005-12 were run to export the information from the telomere ontology into MFO and, ultimately, into the original SBML model. The XML snippet below shows the new SBML species added to the model as part of the new reaction with MEC1:

```
<species metaid="_3a9a2e27" id="interactor_24_id" name="YBR136W"
sboTerm="SBO:0000245">
  <annotation>
    <rdf:RDF>
      <rdf:Description rdf:about="#_3a9a2e27">
        <bqbiol:is>
          <rdf:Bag>
            <rdf:li rdf:resource="urn:miriam:sgd:S000000340"/>
            <rdf:li rdf:resource="urn:miriam:uniprot:P38111"/>
          </rdf:Bag>
        </bqbiol:is>
      </rdf:Description>
    </rdf:RDF>
  </annotation>
</species>
```

Here, the id is based on the name of the telomere ontology `Mec1` instance, the name is that instance's *recommendedName* or *synonym*, and the MIRIAM annotations are drawn from the list of *DatabaseReference* instances. MIRIAM annotations, names and SBO terms were added to the SBML species referenced in the use cases, and two example reactions were added: RAD9 with RAD17, and RAD9 with MEC1. The former was chosen as an example of how a *ProteinComplexFormation* reaction is mapped back to MFO, and the latter was chosen as an example of the mapping of a simpler interaction. Further, as MEC1 is a candidate for the value of the 'Rad9Kin' species in the curated model, creating a skeleton reaction in the annotated version of the curated model is the first step in updating the model. An excerpt of the RAD9/RAD17 reaction in SBML is included below. Further work by the modeller can flesh out the base reaction, species and annotation created by the integration method to include parameters, initial values and rate constants.

```
<reaction id="interaction_160_id">
  <listOfReactants>
    <speciesReference species="interactor_59_id"/>
    <speciesReference species="interactor_8_id"/>
  </listOfReactants>
  <listOfProducts>
    <speciesReference species="product_of_rad9rad17interaction"/>
  </listOfProducts>
</reaction>
[...]
<species metaid="_720ffc30" id="interactor_8_id" name="YOR368W"
sboTerm="SBO:0000245">
  <annotation>
    <rdf:RDF>
      <rdf:Description rdf:about="#_720ffc30">
        <bqbiol:is>
          <rdf:Bag>
            <rdf:li rdf:resource="urn:miriam:sgd:S000005895"/>
            <rdf:li rdf:resource="urn:miriam:uniprot:P48581"/>
          </rdf:Bag>
        </bqbiol:is>
      </rdf:Description>
    </rdf:RDF>
  </annotation>
</species>
<species id="product_of_rad9rad17interaction" name="Rad9Rad17Complex"/>
[...]
```

For Use Case 2, while BioGRID contained the most interactions, it was UniProtKB and Pathway Commons which together contributed the majority of the other types of information. Curated RAD9 reactions with RAD53 and CHK1 were recovered, while of the many other reactions, one (with RAD17) was with an existing species in the model and another (with MEC1) putatively identified a placeholder species in that model.

Within the telomere ontology, the product of a protein complex formation must be a protein complex as described in the *ProteinComplexFormation* class. Specifically the RAD9/RAD53 and RAD9/RAD17 reactions are part of a BioGRID ‘Reconstituted Complex’. This reaction type can be presented for RAD9/RAD53 as $rad9 + rad53 \leftrightarrow rad9rad53Complex$. While not identical to the curated reaction, which shows activation of RAD53 by RAD9 ($rad9Active + rad53Inactive \rightarrow rad9Active + rad53Active$), the proposed interaction is potentially informative.

Use Case 2 addresses a modeller's need to learn of possible new reactions involving the query species retrieved from Use Case 1. In this scenario, the modeller is only interested in protein-protein interactions. The query for this use case would be beneficial during model creation or extension of an already-extant model.

While BioGRID contained the most interactions for the second use case, it was UniProtKB and Pathway Commons which together contributed the majority of the information beyond those interactions. Combining data sources in a semantically-meaningful, rule-based mediation methodology is a useful way of retrieving useful information for SBML model annotation.

5.6 Future directions

5.6.1 Data provenance

There are limitations in describing data provenance using the SBML standard notation. When new MIRIAM annotations are added to an SBML model, there is no way of stating what procedure added the annotations, or the date they were added. At a minimum, the addition of the software type, version number and date on which the information was added should be included. At the moment, SBML is unable to make statements about annotations. The proposed annotation package for SBML Level 3 will address this issue, among others [10]. Until this issue is resolved, no provenance information is included in the newly-annotated models.

5.6.2 Mapping and the creation of syntactic ontologies

When mapping the telomere ontology to MFO, a number of limitations were encountered with the technology used. For instance, in order to map a telomere ontology *PhysicalProcess* such as *ProteinComplexFormation* correctly, a number of new instances were created when the rule was run (see rule in Section 5.3.4). Then, when multiple reactions make use of a single protein instance (such as, in these use cases, any *Rad9* instance), a new MFO identifier, or *SId*, is created. The ultimate result is that more than one *SId* was created for the same protein instance, which if left alone would result in an inconsistent ontology, as only one *SId* is allowed per MFO *Species*. Also, because SWRL rules do not add information to a data property slot unless that slot is already empty, running a rule which adds a telomere ontology *recommendedName* value to an MFO species will not work if a similar rule using the *synonym* value has previously been run. These are all limitations that can be ameliorated through more sophisticated usages of OWL and SWRL tools. One such tool already used within the MFO project is the OWL API [112], which allows finer control over ontologies and their mapping and reasoning. Greater use of the SWRLAPI [219] might also address these limitations.

While extensive use of pre-existing applications, plugins, and libraries has been made, in the very near future the limit of some of these technologies could be reached. Particularly, improvements will need to be made to the way the XML is converted into OWL. Further, if this methodology is to be used in a larger-scale semantic data integration project, the correspondingly large number of instances required will most likely necessitate the use of a database back-end for the ontologies. Some technologies are available for storing instances from ontologies within a database schema such as that followed by Sahoo and colleagues [155], the Protégé database back-end[25] or DBOWL [224].

5.6.3 OWL 2

The use cases presented in this chapter, while performed with existing tools, required manual intervention. Improvements can be made to the implementation of this methodology through the use of additional OWL 2 features. In order to assert equivalences between instance data from the various data sources, the `owl:sameAs` construct was manually applied. Tool and reasoning support for OWL 2, now a W3C Recommendation [99], is growing, and with OWL 2, some advances such as the `owl:hasKey` [225] construct allows more automation in this area. These keys, or inverse functional datatype properties, allow the definition of equivalence rules based on properties such as database accessions, which in the current system was achieved by manual mapping. For example, by applying the `owl:hasKey` axiom to a data property linking to UniProtKB accession numbers, any instances containing matching UniProtKB accessions will be marked as identical.

5.6.4 Classes versus instance modelling for biological entities

In ontologies such as BioPAX Level 3, a specific protein is modelled as an instance of a class rather than as a class in its own right. For instance, `RAD9` would be an instance of the BioPAX *Protein* class. Conversely in the telomere ontology, every protein—including *Rad9*—has been given its own class. This allows for expressive statements to be made about the protein and about particular interactions the protein is involved in. While this type of mapping allows greater expressivity, there are challenges

[25]http://protegewiki.stanford.edu/wiki/Working_with_the_Database_Backend_in_OWL

to the broader usefulness of such mapping. As Demir has highlighted[26], the these hurdles are social as well as technological in nature.

Technologically, the use of expressive classes for specific biological entities such as *Rad9* makes the information difficult for a "traditional" bioinformatician to use, as reasoners must be applied to get the most out of such ontologies. Reasoners are difficult for people unused to them, as their purpose is often unclear to new users and they tend to have esoteric error messages. From a social perspective, allowing users to create an unlimited number and type of entities such as *Rad9* can lead to a lack of standard methods of creating and using those entities.

However, these challenges are all associated with the creation of an ontology which is expressly developed as a community, or shared, ontology. Core ontologies in rule-based mediation may be used by multiple groups, but are primarily intended to address a specific and possibly short-term biological question, and as such do not need to fulfil such a wide remit. While the initial development of the source and core ontologies would require an ontologist, the populating of the resulting knowledgebase within the core ontology is intended to be accessible via a Web application currently under development and, in future, the querying of that knowledgebase via Saint (Chapter 3). Reasoners are an intrinsic part of rule-based mediation, and the number of ontologists building the core ontology would normally be small. Therefore, as the issues raised by Demir are valid for large-scale ontologies intended for public use, they do not apply to the primary role of rule-based mediation.

In the use cases described in this chapter, *instance-to-instance* SWRL mappings assign membership of instances from source ontologies to classes from the core ontology. For example, the `rad9A` instance from the MFO version of the telomere model becomes an instance of the *Rad9* class in the telomere ontology. It would be useful to be able to use mappings to automatically create core ontology classes such as *Rad9*. This *instance-to-class* mapping is theoretically possible, but no programmatically-accessible rule language has this feature. Such rules could instead be hard-coded using a library such as the OWL API. This would allow the automated population of a core ontology with all of the necessary classes for a particular pathway or systems biology model of interest.

5.7 Discussion

In rule-based mediation, a set of source ontologies are individually linked to a core ontology, which describes the domain of interest. Similar or even overlapping data sources and formats can be handled, as each format will have its own source ontology. Although some researchers question the

[26] `https://groups.google.com/forum/?hl=en#!topic/biopax2model/01XfxFfLxhA`

conversion from a closed-world syntax such as XML to a open-world language such as OWL [27], the use of syntactic ontologies allows the separation of the closed-world source syntax from the biological domain modelling which takes place in the core ontology. In essence, the OWL in a syntactic ontology is "abused" in order to allow the conversion of the underlying source data to the more expressive model present in the core ontology. Unlike previous integration methods where one ontology type is created as a view of the other, rule-based mediation allows both the straightforward addition of new source ontologies as well as the maintenance of the core ontology as an independent entity. Using a core ontology allows new data sources and data formats to be added without changing the way the core model of the biological domain is structured.

Rule-based mediation was shown to be a valid methodology for two use cases relevant to the biological modelling community. Two syntactic ontologies were created to represent PSI-MIF and UniProtKB XML entries in OWL format. BioPAX has also been adopted as one of the source ontologies, showing how pre-existing ontologies can successfully be used as a source ontology without modification. The telomere ontology, which describes the research domain surrounding the use cases, can be reasoned over and is able to store information mapped from multiple source ontologies. The source ontologies and telomere ontology are available for download, pre-filled with the use case instances. In future, a Web application will allow the user to query the source databases and retrieve a tailor-made dataset which is then automatically mapped to the telomere ontology. A semantic, rule-based data integration methodology has been created, and proof-of-principle for that methodology in the context of SBML model annotation has been shown.

[27]`https://groups.google.com/forum/?hl=en#!topic/biopax2model/01XfxFfLxhA`

Chapter 6

General discussion

As the structure, description and interconnectedness of life science data has become more sophisticated, a shared formalisation of all data for systems biology has become easier to imagine. Tools are available, such as those used by the Semantic Web community, to capture any area of biology in the form of a semantically defined model. While challenges such as achieving tractable reasoning over highly expressive ontologies still remain to be addressed, a future where vast amounts of data and metadata are modelled according to a shared formalisation is not just fantasy as shown by the work described in this thesis. The research presented here demonstrates the transformation of existing systems biology data models into more computationally amenable and more semantically aware structures.

The work presented also enables a more computationally amenable approach to systems biology, ultimately aiding the progression of systems biology metadata into a more formal structure. There is a general trend towards "big data", both with in-house private repositories and open data across all of the sciences, making maintenance and structuring of that data of primary importance [186, 52]. SyMBA (Chapter 2) aids the formalisation of systems biology knowledge by providing a common structure for experimental metadata and limiting the complexity of metadata input at the user interface. Traditionally, metadata input procedures are time consuming and often complex for large volumes of data. SyMBA was created to make it easy for researchers to add metadata even when large amounts of data are involved. In general the easier metadata entry becomes, the more metadata will be stored by users. By structuring systems biology metadata in a common format, it is made much more accessible to computational techniques and to the systems biology community as a whole.

Saint (Chapter 3) helps researchers find appropriate biological data for systems biology models by reusing existing data from disparate locations. Saint provides a method of integrating multiple Web-accessible resources using syntactic clues in the query model, providing prospective annotation to the modeller. This type of syntactic data integration is fast, providing a large amount of integrated data for relatively low cost and high scalability. However, syntactic integration does not resolve semantic differences in the data, nor does it allow a high level of expressivity between data sources or in the description of those data sources. The schema reconciliation in non-semantic approaches is generally hard-coded for the immediate integrative task, and not easily reused. Often, data is aligned by linking structural units such as XSD components or table and row names rather than the underlying biological components. Further, concepts between the source and target schema are often linked based on syntactic similarity, which does not necessarily account for possible differences in the meanings of those concepts.

Semantic data integration allows greater expressivity at the expense of scalability. Controlled vocabularies and ontologies are of primary importance in the resolution of heterogeneity via semantic means [137]. By using ontologies, entities can be integrated across domains according to their meaning. Ruttenberg and colleagues view the Semantic Web, of which both OWL and RDF are components, as having the potential to aid translational and systems biology research; indeed, any life science field where there are large amounts of data in distributed, disparate formats should benefit from Semantic Web technologies [146]. However, application of such techniques in bioinformatics is difficult, partly due to the bespoke nature of the majority of available tools [148]. MFO (Chapter 4) transforms existing SBML specification documents into a single model in OWL, allowing the storage both of SBML models and the restrictions on those models. While some of the SBML specification documents are already computationally accessible, one document is readable only by humans and each of the others are in different formats. MFO brings concepts from all of the SBML documents into a single computationally accessible format which can be used with a variety of Semantic Web technologies.

Rule-based mediation (Chapter 5) transforms multiple systems biology data models into a single, more computationally accessible model. This work examines the feasibility of semantic data integration within a systems biology context. To gain access to the underlying semantics of the data, a semantically-rich core ontology is utilised together with mappings to and from source ontologies representing the data sources of interest. New data sources can be easily inserted without modification of the biologically-relevant core ontology. Separation of syntactic integration and semantic description of the biology is robust to changes to both the source ontologies and the core ontology. The use of existing tools decreased development time and increased the applicability of this approach for future projects. While some hurdles–such as scalability and the creation of a simple user interface—remain, rule-based mediation has been shown to be an effective approach for model annotation.

The field of semantics and ontologies in the life sciences is continuing to mature. The traditional and highly useful role of the ontology as metadata and link provider is pervasive. In this role, a particular database entry structures the data according to a defined data format, and within that format links are provided to specific ontology terms. The data format is the primary structure, and the ontological terms are keywords and links out to more information and to other database entries sharing that term. Very recently, new research such as rule-based mediation has freed ontologies, reversing the roles of ontology and structured format [35, 65, 160, 205]. With semantic integration methodologies, ontologies—rather than the syntactic format—have become the backbone from which data hangs. Ontologies can therefore be used to describe the entirety of a data model or biological domain rather than just provide links to further information. The use of ontologies as a sophisticated semantic integration tool allows the creation of a methodology which not only follows cross-referenced links from one database entry to the next, but aligns information according to biological knowledge, limiting ambiguity and making that knowledge accessible to both humans and computers.

This work described in this thesis is one step towards the formalisation of biological knowledge useful to systems biology. Experimental metadata has been transformed into common structures, data appropriate to the annotation of systems biology models can be retrieved and presented to users and multiple data models have been formalised and made accessible to semantic integration techniques.

6.1 Simplicity versus complexity

When transforming data and metadata into a common structure or model, there are always compromises to be made between simplicity and complexity. Rather than there being a single correct answer, researchers investigating particular integration techniques choose points along this spectrum amenable to their requirements. The dichotomy of simplicity and complexity is appropriate for a range of topics relevant to systems biology data integration including metadata, simulatable model creation and expressivity. The question of expressivity versus scalability is discussed in more detail below as it has a direct effect on the performance of semantic data integration techniques such as rule-based mediation.

In general, researchers find it easier to create data than to annotate that data with large amounts of metadata; while simplistic metadata requirements allow quicker data deposition, complex metadata requirements can create a barrier to data entry. While standards are often developed once 'just-enough' data description strategies are no longer enough (as described in Section 1.4), as the amount of metadata asked of researchers grows, the chance of receiving complete metadata shrinks.

Systems biology models themselves are another example of choosing between simplicity and complexity. This is not simply a question of the granularity of such models, as whole organism models and single pathway models of a similar level of complexity can be produced. However, simpler models with fewer species and less complex interactions can be more easily understood by people and simulated by computers. While complex models present a more complete picture, the resources required to simulate and understand will be correspondingly greater.

As a final example, formal models such as ontologies can be either simple and scalable or complex and expressive; there is always a balance to be made between these two choices. Expressivity is a measure of what it is possible to say with a language, and as such has a direct bearing on the inferences possible with that language; too little expressivity results in a lack of "reasoning opportunities" in a language which would otherwise not provide any benefits over existing languages [226]. Simple formal models are usually scalable and can easily be reasoned over but do not, by definition, contain a high amount of expressivity. Complex models have a higher level of expressivity and are capable of modelling more biological context, for example. With such models a more precise description of the domain of interest is created at the cost of reasoning time.

OWL-DL is an ontology formalism which is guaranteed to be decidable. The decidable subsets of OWL, such as OWL-DL, are intended to allow both expressivity and tractability for reasoning purposes; these requirements are typically in opposition to each other, and as such it is a goal of OWL to find a suitable balance [226]. Specifically, there are no guarantees made about the level of scalability of any given ontology written in OWL-DL. The more complex an ontology the less amenable it is to reasoning; a change of even a single axiom can result in a reasoner completing in days rather than seconds. Detailed studies of the complexity of ontological reasoning and how that complexity increases with the expressiveness of an ontology are available [110, Chapter 3], and are beyond the scope of this thesis. However, expressivity versus tractability is of vital importance in semantic data integration in general and in rule-based mediation specifically, where the core ontology is intended to be small but highly expressive.

Decidability and expressiveness are vital considerations when choosing a language for rule-based mediation; languages such as OBO were not originally created with these challenges in mind. Not only is it important to choose a language which supports a balance between expressivity and tractability, but it is important to create ontologies with those considerations in mind. Even if a decidable language such as OWL-DL is used, an ontology can be easily created which cannot be reasoned over in sensible time scales. Currently, in semantic data integration methods such as rule-based mediation, scalability issues occur before more general integration issues such as database churn become important. Therefore for many in the life sciences, ontology engineering involves keeping ontologies as simple as possible for the stated requirements, adding detail and expressivity only where necessary and in a non-disruptive way [227].

6.2 The future of data integration

The use of confidence values for particular mappings between ontologies has been described [159], although research in this area is still limited. However, the ability to modify the set of mappings between source and core ontologies based on the perceived quality of the mappings would be an interesting area of further research. Rule-based mediation makes use of an expressive core ontology, which currently places limitations on its scalability. As brute computational power and technology such as reasoners improve, methodologies such as rule-based mediation could go from precise, small-scale integrative tasks to much larger integrative challenges. Additional automation is also a necessary step if semantic methods such as rule-based mediation are to be implemented on a wider scale. However, existing and future collaborations should provide scope for these and other possibilities for this research. A collaboration of the CellML project with the Saint project started in 2009 and has continued in 2011 and 2012 with additional programmer support to extend the functionality of Saint with respect to CellML, and further aid the integration of the MIRIAM annotation with the CellML format and API. Rule-based mediation has also been identified by Hoehndorf and co-workers as a possible source of collaboration [205].

A much broader perspective on the future of data integration is not just which integrative methodology is best suited for a task, but whether data integration will even have a place in biological research. If all life science data were expressed in a homogeneous manner, data integration would be unnecessary. The rest of this section explores some of the ways in which data might become accessible to all, without the need for interfaces and integration layers among formats.

If standards become fully utilised within the life sciences, every researcher would be publishing data in recognised, common formats. However, this does not necessarily mean a single format for the entirety of the life sciences; more likely, a large number of standards would remain, as described in Section 1.4. While it is true that many community standards are already in common use (e.g. SBML and BioPAX), standards efforts have a history of slow development (e.g. OBI) and slow uptake (e.g. FuGE). Additionally, new questions will continue to be asked and experimental types will continue to be invented. By definition, these new experiment types will be created faster than the corresponding standards can be developed. Even as experiment types and, ultimately, community standards mature, new standards or combinations of standards might be required. Through experience gained as a developer of many community standards it has become clear that, irrespective of both the number of experimental types and the quality of standardised data formats, there will likely always be some overlap in the data descriptions among the formats. If there is even a small amount of information in one community standard that is of use to another community, data integration will continue to play a vital role in the life sciences. As such even widespread, endemic use of standards will not remove the need for high-quality data integration. As quickly as standards are made, new data is created and new descriptions of data are written which require standardisation.

Alternatively, existing data integration methods might become useless as newer methodologies are investigated. For instance, semantic methodologies are able to capture more knowledge about the data than syntactic methods. More complete, less ambiguous models of the domain of interest leads to less confusion and fewer mapping errors during the integration process. But as has been described in Section 1.6, many syntactic data integration methods remain popular and are being successfully developed throughout the life sciences. Emerging technologies such as reasoner-accessible semantic methods have been proven to be useful, but are not yet supplanting existing methods. Scalability and tractability are core issues when reasoners are applied, and the very employment of the expressivity which make ontology-based techniques useful quickly exposes such issues.

There is a growing movement within science towards openness of data. Open Data movements such as the Panton Principles[1] describe appropriate licensing for open experimental data, while many others have advocated open models of publishing papers as well as data[2]. For instance arXiv[3], an e-print archive for topics including physics, mathematics, computer science, quantitative biology and statistics, provides open access to both data and papers. Conferences are also becoming more open, with the publishing of papers and presentations in open access journals and with the use of social networking to provide live coverage of conferences [228, 229]. It may seem as if this increased availability and openness of data and publications might decrease the importance of data integration. However, open data is a prime example of why data reuse and integration will remain important. Larger amounts of data do not lessen the need of integration methodologies, as open data is not guaranteed to be in the same format.

Finally, if there is a move away from small, tightly-scoped experiments towards large-scale experiments, large amounts of data could be produced in a single format. For example, there would be no need to integrate disparate experimental data files the research could be reproduced cheaply and quickly. While large-scale experimentation would ease the burden of integration within one community or experimental type, it will not solve the integration problems across multiple communities. As such, this would not make data integration redundant.

Although there are a number of ways in which data will become more homogeneous, none will render data integration obsolete. If all of systems biology was formalised and structured according to a common data model, transformations on that data model would still be required. For instance, the model could go through rounds of optimisation or updating during schema evolution. Syntactic

[1]http://pantonprinciples.org/

[2]More information can be found at http://cameronneylon.net/ and https://opencitations.wordpress.com/2011/08/04/the-plate-tectonics-of-research-data-publication/.

[3]http://arxiv.org/

methods will most likely play a pivotal role in data integration for the foreseeable future, until the limitations of semantic methods are ameliorated and ontologies and other Semantic Web technologies gain much higher coverage within the life sciences. The ability of Semantic Web formats such as RDF to describe any type of data, and of Semantic Web languages such as ontologies to model any type of biology will ensure that semantic data integration will only increase in importance as scaling issues are addressed and the time taken to perform the integration decreases.

6.3 Guidance for modellers and other researchers

If data generators and systems biology modellers followed just a few guidelines when describing their information, data integration tasks could become much more straightforward and many existing methods of integration would immediately become more powerful. Small changes could result in substantive improvements in the way data is described and ultimately, the quality of the integration that can be performed. For instance, unambiguous naming would result in better hits against external data sources using a systems biology annotation application such as Saint (Chapter 3). The list below describes these guidelines, and how their use would aid data integration in general as well as help the integration methods presented in this thesis.

1. *Informative naming.* In general, systems biology modellers name model entities using non-informative names and identifiers as well as non-intuitive naming schemes. A species might be called "A" or "species1", providing no clue as to the identity of the species for prospective users of the model. Further, while an individual modeller might conform to their own personal naming scheme, there is no guarantee that others will know that the "_p" added to a species name means it is the phosphorylated form; it could equally mean gene product, probable name, or even possibly unknown. An informative naming methodology is simple and quick to implement while also being a great help to others. Even if no further information is provided about a model entity, educated guesses as to the identity of that entity can be made both by humans and computers if commonly-used names are provided.

2. *Consider standards from the start of the research.* The life science community has many ways to store experimental data in a format recognisable to computers and to humans, and many helper applications to do most of the work for the researcher. SyMBA (Chapter 2) is an archival application capable of storing experimental data and metadata. The ISA Suite [190] assists with data management and standards-compliant storage while the experiments are being run,

and does so in a spreadsheet-style environment comfortable to many researchers. Ensuring that the correct information is stored while the data is being gathered saves time for the researcher and eases submission of that data to public repositories prior to publication.

3. *Think about the future.* As described by one twitter user, “metadata is a love note to the future”[4]. High quality metadata makes data integration easier, and enables greater reuse, sharing and re-purposing of the underlying information both for the creator of that information and for future interested researchers. The use of systems such as Saint (Chapter 3) and rule-based mediation (Chapter 5), which add annotation in a semi-automated way, can help with the addition of metadata, fostering understanding and allowing the research to reach a wider audience.

6.4 Conclusions

The early releases of life science databases such as EMBL/GenBank/DDBJ [57] and UniProtKB [1] provided data in a structured, text-based flat file format. As techniques, knowledge and computing power improved, the storage and exchange formats for these and other databases have become more complex. While EMBL/GenBank/DDBJ and UniProtKB can still be retrieved as flat files, internally the data is stored in relational databases, and UniProtKB provides both an XML and an RDF export. Newer life science databases such as BioModels [21] and Pathway Commons [218] never even produced a flat file format; instead, XML and OWL began to be used as the primary data format. The progression of the structure of data from text file to XML to OWL is a progression towards a formalisation of the representation of biology.

Semantic technologies are being heavily researched and used in the life sciences and in larger-scale commercial portals such as Google. However, not all semantic projects have been successful. For instance, the semantic database and publishing system Ambra[5], in use by the PLoS group of journals for a number of years[6], was dropped in November 2011 for a faster system called New Hope[7]. In contrast, both social networks and other large commercial portals have begun increasing their usage of semantic technologies. As of October 2011, Facebook has begun supporting linked data and RDF[8]. Google makes use of RDFa and microformats for their Rich Snippets[9] technology. The

[4]http://twitpic.com/6ry6ar
[5]http://www.topazproject.org/trac/wiki/Ambra
[6]http://blogs.plos.org/everyone/2009/05/13/all-plos-titles-now-on-the-same-publishing-platform/
[7]http://blogs.plos.org/plos/2011/12/new-hope-the-new-platform-for-the-plos-journal-websites/
[8]http://www.w3.org/wiki/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[9]http://support.google.com/webmasters/bin/answer.py?hl=en&answer=99170, http://www.readwriteweb.com/archives/google_semantic_web_push_rich_snippets_usage_grow.php

massive amounts of data stored by companies such as Google and Facebook, or to a lesser degree by semantically-aware life science databases such as Bio2RDF [166] may provide the volume of metadata required to increase awareness and usage of the Semantic Web[10]. As the amount of life science data grows, improved semantic structure of that data will aid integration and querying tasks.

The research presented in this thesis standardises experimental metadata (Chapter 2) and integrates (Chapter 3, Chapter 5) and formalises (Chapters 4-5) systems biology knowledge. As a result, new biological annotation can be added to systems biology models, and information useful for systems biology has been transformed into more computationally amenable and semantically aware structures.

This research demonstrates the advantages gained with an increase in the formalisation of biological knowledge. The use of richer semantics allows the automation of many aspects of the annotation process for systems biology models as well as computational access to those models to rigorously assess their correctness. While this comes with a cost of a greater formality and standardisation for experimental data, the cost can be partially alleviated with the use of appropriate applications to make the input of the biological descriptions easier. As the life sciences become ever more data rich, this formalisation becomes less of an option and more of a necessity to ensure that the multitude of data resources remain accessible and manageable to researchers and computers.

[10]http://semanticweb.com/the-evolution-of-search-at-google_b25042

Bibliography

[1] The UniProt Consortium. The Universal Protein Resource (UniProt). *Nucl. Acids Res.*, 36(suppl_1):D190–195, January 2008.

[2] Andrew R. Jones, Michael Miller, Ruedi Aebersold, Rolf Apweiler, Catherine A. Ball, Alvis Brazma, James DeGreef, Nigel Hardy, Henning Hermjakob, Simon J. Hubbard, Peter Hussey, Mark Igra, Helen Jenkins, Randall K. Julian, Kent Laursen, Stephen G. Oliver, Norman W. Paton, Susanna-Assunta Sansone, Ugis Sarkans, Christian J. Stoeckert, Chris F. Taylor, Patricia L. Whetzel, Joseph A. White, Paul Spellman, and Angel Pizarro. The Functional Genomics Experiment model (FuGE): an extensible framework for standards in functional genomics. *Nature Biotechnology*, 25(10):1127–1133, October 2007.

[3] Andrew R. Jones and Allyson L. Lister. Managing experimental data using FuGE. *Methods in molecular biology (Clifton, N.J.)*, 604:333–343, 2010.

[4] Andrew R. Jones, Allyson L. Lister, Leandro Hermida, Peter Wilkinson, Martin Eisenacher, Khalid Belhajjame, Frank Gibson, Phil Lord, Matthew Pocock, Heiko Rosenfelder, Javier Santoyo-Lopez, Anil Wipat, and Norman W. W. Paton. Modeling and managing experimental data using FuGE. *Omics : a journal of integrative biology*, 13(3):239–251, June 2009.

[5] Mélanie Courtot, William Bug, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. The OWL of Biomedical Investigations. In *OWLED 2008*, October 2008.

[6] The OBI Consortium. Modeling biomedical experimental processes with OBI. In Phillip Lord, Susanna-Assunta Sansone, Nigam Shah, Susie Stephens, and Larisa Soldatova, editors, *The 12th Annual Bio-Ontologies Meeting, ISMB 2009*, pages 41+, June 2009.

[7] Mélanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, and Alan Ruttenberg. MIREOT: The minimum information to reference an external ontology term. *Applied Ontology*, 6(1):23–33, January 2011.

[8] Susanna-Assunta Sansone, Philippe R. Serra, Marco Brandizi, Alvis Brazma, Dawn Field, Jennifer Fostel, Andrew G. Garrow, Jack Gilbert, Federico Goodsaid, Nigel Hardy, Phil Jones, Allyson Lister, Michael Miller, Norman Morrison, Tim Rayner, Nataliya Sklyar, Chris Taylor, Weida Tong, Guy Warner, and Stefan Wiemann. The First RSBI (ISA-TAB) Workshop: &#x201C;Can a Simple Format Work for Complex Studies?&#x201D;. *OMICS: A Journal of Integrative Biology*, 12(2):143–149, 2008.

[9] Dawn Field, George Garrity, Tanya Gray, Norman Morrison, Jeremy Selengut, Peter Sterk, Tatiana Tatusova, Nicholas Thomson, Michael J. Allen, Samuel V. Angiuoli, Michael Ashburner, Nelson Axelrod, Sandra Baldauf, Stuart Ballard, Jeffrey Boore, Guy Cochrane, James Cole, Peter Dawyndt, Paul De Vos, Claude dePamphilis, Robert Edwards, Nadeem Faruque,

Robert Feldman, Jack Gilbert, Paul Gilna, Frank O. Glockner, Philip Goldstein, Robert Guralnick, Dan Haft, David Hancock, Henning Hermjakob, Christiane Hertz-Fowler, Phil Hugenholtz, Ian Joint, Leonid Kagan, Matthew Kane, Jessie Kennedy, George Kowalchuk, Renzo Kottmann, Eugene Kolker, Saul Kravitz, Nikos Kyrpides, Jim Leebens-Mack, Suzanna E. Lewis, Kelvin Li, Allyson L. Lister, Phillip Lord, Natalia Maltsev, Victor Markowitz, Jennifer Martiny, Barbara Methe, Ilene Mizrachi, Richard Moxon, Karen Nelson, Julian Parkhill, Lita Proctor, Owen White, Susanna-Assunta Sansone, Andrew Spiers, Robert Stevens, Paul Swift, Chris Taylor, Yoshio Tateno, Adrian Tett, Sarah Turner, David Ussery, Bob Vaughan, Naomi Ward, Trish Whetzel, Ingio San Gil, Gareth Wilson, and Anil Wipat. The minimum information about a genome sequence (MIGS) specification. *Nature Biotechnology*, 26(5):541–547, May 2008.

[10] Dagmar Waltemath, Neil Swainston, Allyson Lister, Frank Bergmann, Ron Henkel, Stefan Hoops, Michael Hucka, Nick Juty, Sarah Keating, Christian Knuepfer, Falko Krause, Camille Laibe, Wolfram Liebermeister, Catherine Lloyd, Goksel Misirli, Marvin Schulz, Morgan Taschuk, and Nicolas Le Novère. SBML Level 3 Package Proposal: Annotation. *Nature Precedings*, (713), January 2011.

[11] Erick Antezana, Martin Kuiper, and Vladimir Mironov. Biological knowledge management: the emerging role of the Semantic Web technologies. *Briefings in Bioinformatics*, 10(4):392–407, July 2009.

[12] Uwe Sauer, Matthias Heinemann, and Nicola Zamboni. Getting Closer to the Whole Picture. *Science*, 316(5824):550–551, April 2007.

[13] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, August 1952.

[14] Hiroaki Kitano. Systems Biology: A Brief Overview. *Science*, 295(5560):1662–1664, March 2002.

[15] Jason R. Swedlow, Suzanna E. Lewis, and Ilya G. Goldberg. Modelling data across labs, genomes, space and time. *Nature Cell Biology*, 8(11):1190–1194, November 2006.

[16] Katrin Hübner, Sven Sahle, and Ursula Kummer. Applications and trends in systems biology in biochemistry. *FEBS Journal*, 278(16):2767–2857, August 2011.

[17] Stephan Philippi and Jacob Kohler. Addressing the problems with life-science databases for traditional uses and systems biology. *Nat Rev Genet*, 7(6):482–488, June 2006.

[18] Michael Ashburner, Catherine A. Ball, Judith A. Blake, David Botstein, Heather Butler, J. Michael Cherry, Allan P. Davis, Kara Dolinski, Selina S. Dwight, Janan T. Eppig, Midori A. Harris, David P. Hill, Laurie Issel-Tarver, Andrew Kasarskis, Suzanna Lewis, John C. Matese, Joel E. Richardson, Martin Ringwald, Gerald M. Rubin, and Gavin Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, May 2000.

[19] Nicolas L. Novere, Andrew Finney, Michael Hucka, Upinder S. Bhalla, Fabien Campagne, Julio Collado-Vides, Edmund J. Crampin, Matt Halstead, Edda Klipp, Pedro Mendes, Poul Nielsen, Herbert Sauro, Bruce Shapiro, Jacky L. Snoep, Hugh D. Spence, and Barry L. Wanner. Minimum information requested in the annotation of biochemical models (MIRIAM). *Nature Biotechnology*, 23(12):1509–1515, December 2005.

[20] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, , the rest of the SBML Forum:, 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*, 19(4):524–531, March 2003.

[21] Chen Li, Marco Donizelli, Nicolas Rodriguez, Harish Dharuri, Lukas Endler, Vijayalakshmi Chelliah, Lu Li, Enuo He, Arnaud Henry, Melanie Stefan, Jacky Snoep, Michael Hucka, Nicolas Le Novere, and Camille Laibe. BioModels Database: An enhanced, curated and annotated resource for published quantitative kinetic models. *BMC Systems Biology*, 4(1):92+, June 2010.

[22] James E. Ferrell. Q&A: systems biology. *Journal of biology*, 8(1):2+, January 2009.

[23] Lincoln D. Stein. Bioinformatics: alive and kicking. *Genome biology*, 9(12):114+, December 2008.

[24] Nicolas Le Novère. The long journey to a Systems Biology of neuronal function. *BMC systems biology*, 1(1):28+, 2007.

[25] Anna Bauer-Mehren, Laura I. Furlong, and Ferran Sanz. Pathway databases and tools for their exploitation: benefits, current limitations and challenges. *Molecular Systems Biology*, 5, July 2009.

[26] Joanne S. Luciano and Robert D. Stevens. e-Science and biological pathway semantics. *BMC bioinformatics*, 8 Suppl 3(Suppl 3):S3+, 2007.

[27] Xiaowei Zhu, Mark Gerstein, and Michael Snyder. Getting connected: analysis and principles of biological networks. *Genes & Development*, 21(9):1010–1024, May 2007.

[28] Katherine James, Anil Wipat, and Jennifer Hallinan. Integration of Full-Coverage Probabilistic Functional Networks with Relevance to Specific Biological Processes. In Norman Paton, Paolo Missier, and Cornelia Hedeler, editors, *Data Integration in the Life Sciences*, volume 5647 of *Lecture Notes in Computer Science*, chapter 4, pages 31–46. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2009.

[29] Markus J. Herrgard, Neil Swainston, Paul Dobson, Warwick B. Dunn, K. Yalcin Arga, Mikko Arvas, Nils Buthgen, Simon Borger, Roeland Costenoble, Matthias Heinemann, Michael Hucka, Nicolas Le Novere, Peter Li, Wolfram Liebermeister, Monica L. Mo, Ana P. Oliveira, Dina Petranovic, Stephen Pettifer, Evangelos Simeonidis, Kieran Smallbone, Irena Spasie, Dieter Weichart, Roger Brent, David S. Broomhead, Hans V. Westerhoff, Betul Kurdar, Merja Penttila, Edda Klipp, Bernhard O. Palsson, Uwe Sauer, Stephen G. Oliver, Pedro Mendes, Jens Nielsen, and Douglas B. Kell. A consensus yeast metabolic network reconstruction obtained from a community approach to systems biology. *Nature Biotechnology*, 26(10):1155–1160, October 2008.

[30] Natalie C. Duarte, Markus J. Herrgård, and Bernhard Palsson. Reconstruction and Validation of Saccharomyces cerevisiae iND750, a Fully Compartmentalized Genome-Scale Metabolic Model. *Genome Research*, 14(7):1298–1309, July 2004.

[31] Jochen Förster, Iman Famili, Patrick Fu, Bernhard Ø. Palsson, and Jens Nielsen. Genome-scale reconstruction of the Saccharomyces cerevisiae metabolic network. *Genome research*, 13(2):244–253, February 2003.

[32] Hongwu Ma, Anatoly Sorokin, Alexander Mazein, Alex Selkov, Evgeni Selkov, Oleg Demin, and Igor Goryanin. The Edinburgh human metabolic network reconstruction and its functional analysis. *Molecular systems biology*, 3, 2007.

[33] Natalie C. Duarte, Scott A. Becker, Neema Jamshidi, Ines Thiele, Monica L. Mo, Thuy D. Vo, Rohith Srivas, and Bernhard Ø. Palsson. Global reconstruction of the human metabolic network based on genomic and bibliomic data. *Proceedings of the National Academy of Sciences of the United States of America*, 104(6):1777–1782, February 2007.

[34] Dave Beckett. RDF/XML Syntax Specification (Revised). http://www.w3.org/TR/rdf-syntax-grammar/, February 2004.

[35] Emek Demir, Michael P. Cary, Suzanne Paley, Ken Fukuda, Christian Lemer, Imre Vastrik, Guanming Wu, Peter D'Eustachio, Carl Schaefer, Joanne Luciano, Frank Schacherer, Irma Martinez-Flores, Zhenjun Hu, Veronica Jimenez-Jacinto, Geeta Joshi-Tope, Kumaran Kandasamy, Alejandra C. Lopez-Fuentes, Huaiyu Mi, Elgar Pichler, Igor Rodchenkov, Andrea Splendiani, Sasha Tkachev, Jeremy Zucker, Gopal Gopinath, Harsha Rajasimha, Ranjani Ramakrishnan, Imran Shah, Mustafa Syed, Nadia Anwar, Ozgün Babur, Michael Blinov, Erik Brauner, Dan Corwin, Sylva Donaldson, Frank Gibbons, Robert Goldberg, Peter Hornbeck, Augustin Luna, Peter Murray-Rust, Eric Neumann, Oliver Reubenacker, Matthias Samwald, Martijn van Iersel, Sarala Wimalaratne, Keith Allen, Burk Braun, Michelle Whirl-Carrillo, Kei-Hoi H. Cheung, Kam Dahlquist, Andrew Finney, Marc Gillespie, Elizabeth Glass, Li Gong, Robin Haw, Michael Honig, Olivier Hubaut, David Kane, Shiva Krupa, Martina Kutmon, Julie Leonard, Debbie Marks, David Merberg, Victoria Petri, Alex Pico, Dean Ravenscroft, Liya Ren, Nigam Shah, Margot Sunshine, Rebecca Tang, Ryan Whaley, Stan Letovksy, Kenneth H. Buetow, Andrey Rzhetsky, Vincent Schachter, Bruno S. Sobral, Ugur Dogrusoz, Shannon McWeeney, Mirit Aladjem, Ewan Birney, Julio Collado-Vides, Susumu Goto, Michael Hucka, Nicolas Le Novère, Natalia Maltsev, Akhilesh Pandey, Paul Thomas, Edgar Wingender, Peter D. Karp, Chris Sander, and Gary D. Bader. The BioPAX community standard for pathway data sharing. *Nature biotechnology*, 28(9):935–942, September 2010.

[36] Jacob Köhler, Jan Baumbach, Jan Taubert, Michael Specht, Andre Skusa, Alexander Rüegg, Chris Rawlings, Paul Verrier, and Stephan Philippi. Graph-based analysis and visualization of experimental results with ONDEX. *Bioinformatics*, 22(11):1383–1390, June 2006.

[37] Artem Lysenko, Michael D. Platel, Keywan H. Pak, Jan Taubert, Charlie Hodgman, Christopher Rawlings, and Mansoor Saqi. Assessing the functional coherence of modules found in multiple-evidence networks from Arabidopsis. *BMC Bioinformatics*, 12(1):203+, 2011.

[38] Jochen Weile, Matthew Pocock, Simon J. Cockell, Phillip Lord, James M. Dewar, Eva-Maria Holstein, Darren Wilkinson, David Lydall, Jennifer Hallinan, and Anil Wipat. Customizable views on semantically integrated networks for systems biology. *Bioinformatics*, 27(9):1299–1306, May 2011.

[39] Simon J. Cockell, Jochen Weile, 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*, 7(3), 2010.

[40] Paul Dobson, Kieran Smallbone, Daniel Jameson, Evangelos Simeonidis, Karin Lanthaler, Pinar Pir, Chuan Lu, Neil Swainston, Warwick Dunn, Paul Fisher, Duncan Hull, Marie Brown,

Olusegun Oshota, Natalie Stanford, Douglas Kell, Ross King, Stephen Oliver, Robert Stevens, and Pedro Mendes. Further developments towards a genome-scale metabolic model of yeast. *BMC Systems Biology*, 4(1):145+, October 2010.

[41] O. Ruebenacker, I. I. Moraru, J. C. Schaff, and M. L. Blinov. Integrating BioPAX pathway knowledge with SBML models. *IET Systems Biology*, 3(5):317–328, 2009.

[42] Kieran Smallbone, Evangelos Simeonidis, David S. Broomhead, and Douglas B. Kell. Something from nothing - bridging the gap between constraint-based and kinetic modelling. *FEBS Journal*, 274(21):5576–5585, November 2007.

[43] Kieran Smallbone, Evangelos Simeonidis, Neil Swainston, and Pedro Mendes. Towards a genome-scale kinetic model of cellular metabolism. *BMC Systems Biology*, 4(1):6+, January 2010.

[44] Catherine M. Lloyd, James R. Lawson, Peter J. Hunter, and Poul F. Nielsen. The CellML Model Repository. *Bioinformatics*, 24(18):2122–2123, September 2008.

[45] M. Hucka, A. Finney, B. J. Bornstein, S. M. Keating, B. E. Shapiro, J. Matthews, B. L. Kovitz, M. J. Schilstra, A. Funahashi, J. C. Doyle, and H. Kitano. Evolving a lingua franca and associated software infrastructure for computational systems biology: the Systems Biology Markup Language (SBML) project. *Systems biology*, 1(1):41–53, June 2004.

[46] Camille Laibe and Nicolas Le Novere. MIRIAM Resources: tools to generate and resolve robust cross-references in Systems Biology. *BMC Systems Biology*, 1(1):58+, 2007.

[47] Allyson L. Lister, Matthew Pocock, Morgan Taschuk, and Anil Wipat. Saint: a lightweight integration environment for model annotation. *Bioinformatics*, 25(22):3026–3027, November 2009.

[48] Peter Li, Tom Oinn, Stian Soiland, and Douglas B. Kell. Automated manipulation of systems biology models using libSBML within Taverna workflows. *Bioinformatics (Oxford, England)*, 24(2):287–289, January 2008.

[49] Neil Swainston and Pedro Mendes. libAnnotationSBML: a library for exploiting SBML annotations. *Bioinformatics*, 25(17):2292–2293, September 2009.

[50] M. L. Blinov, O. Ruebenacker, and I. I. Moraru. Complexity and modularity of intracellular networks: a systematic approach for modelling and simulation. *IET systems biology*, 2(5):363–368, September 2008.

[51] Falko Krause, Jannis Uhlendorf, Timo Lubitz, Marvin Schulz, Edda Klipp, and Wolfram Liebermeister. Annotation and merging of SBML models with semanticSBML. *Bioinformatics*, 26(3):421–422, February 2010.

[52] Doug Howe, Maria Costanzo, Petra Fey, Takashi Gojobori, Linda Hannick, Winston Hide, David P. Hill, Renate Kania, Mary Schaeffer, Susan St Pierre, Simon Twigger, Owen White, and Seung Yon Y. Rhee. Big data: The future of biocuration. *Nature*, 455(7209):47–50, September 2008.

[53] Community cleverness required. *Nature*, 455(7209):1, September 2008.

[54] Ron Edgar and Tanya Barrett. NCBI GEO standards and services for microarray data. *Nature Biotechnology*, 24(12):1471–1472, December 2006.

[55] Alvis Brazma, Pascal Hingamp, John Quackenbush, Gavin Sherlock, Paul Spellman, Chris Stoeckert, John Aach, Wilhelm Ansorge, Catherine A. Ball, Helen C. Causton, Terry Gaasterland, Patrick Glenisson, Frank C. P. Holstege, Irene F. Kim, Victor Markowitz, John C. Matese, Helen Parkinson, Alan Robinson, Ugis Sarkans, Steffen Schulze-Kremer, Jason Stewart, Ronald Taylor, Jaak Vilo, and Martin Vingron. Minimum information about a microarray experiment (MIAME)—toward standards for microarray data. *Nature Genetics*, 29(4):365–371, December 2001.

[56] Henning Hermjakob, Luisa Montecchi-Palazzi, Gary Bader, Jérôme Wojcik, Lukasz Salwinski, Arnaud Ceol, Susan Moore, Sandra Orchard, Ugis Sarkans, Christian von Mering, Bernd Roechert, Sylvain Poux, Eva Jung, Henning Mersch, Paul Kersey, Michael Lappe, Yixue Li, Rong Zeng, Debashis Rana, Macha Nikolski, Holger Husi, Christine Brun, K. Shanker, Seth G. Grant, Chris Sander, Peer Bork, Weimin Zhu, Akhilesh Pandey, Alvis Brazma, Bernard Jacq, Marc Vidal, David Sherman, Pierre Legrain, Gianni Cesareni, Ioannis Xenarios, David Eisenberg, Boris Steipe, Chris Hogue, and Rolf Apweiler. The HUPO PSI's molecular interaction format–a community standard for the representation of protein interaction data. *Nature biotechnology*, 22(2):177–183, February 2004.

[57] Guy Cochrane, Ruth Akhtar, James Bonfield, Lawrence Bower, Fehmi Demiralp, Nadeem Faruque, Richard Gibson, Gemma Hoad, Tim Hubbard, Christopher Hunter, Mikyung Jang, Szilveszter Juhos, Rasko Leinonen, Steven Leonard, Quan Lin, Rodrigo Lopez, Dariusz Lorenc, Hamish McWilliam, Gaurab Mukherjee, Sheila Plaister, Rajesh Radhakrishnan, Stephen Robinson, Siamak Sobhany, Petra T. Hoopen, Robert Vaughan, Vadim Zalunin, and Ewan Birney. Petabyte-scale innovations at the European Nucleotide Archive. *Nucleic Acids Research*, 37(suppl 1):D19–D25, January 2009.

[58] Yu Qian, Olga Tchuvatkina, Josef Spidlen, Peter Wilkinson, Maura Gasparetto, Andrew Jones, Frank Manion, Richard Scheuermann, Rafick P. Sekaly, and Ryan Brinkman. FuGEFlow: data model and markup language for flow cytometry. *BMC Bioinformatics*, 10(1):184+, June 2009.

[59] Andrew Miller, Justin Marsh, Adam Reeve, Alan Garny, Randall Britten, Matt Halstead, Jonathan Cooper, David Nickerson, and Poul Nielsen. An overview of the CellML API and its implementation. *BMC Bioinformatics*, 11(1):178+, April 2010.

[60] Robert Arp and Barry Smith. Function, Role, and Disposition in Basic Formal Ontology. *Nature Precedings*, (713).

[61] Phillip Lord and Robert Stevens. Adding a Little Reality to Building Ontologies for Biology. *PLoS ONE*, 5(9):e12258+, September 2010.

[62] Phillip Lord. An evolutionary approach to Function. *Journal of Biomedical Semantics*, 1(Suppl 1):S4+, 2010.

[63] Robert Stevens. Unicorns in my Ontology, May 2011.

[64] Michel Dumontier and Robert Hoehndorf. Realism for scientific ontologies. In *Proceeding of the 2010 conference on Formal Ontology in Information Systems: Proceedings of the Sixth International Conference (FOIS 2010)*, pages 387–399, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[65] The OBI Consortium. OBI Ontology.

[66] V. Chelliah, L. Endler, N. Juty, C. Laibe, C. Li, N. Rodriguez, and N. Le Novere. Data Integration and Semantic Enrichment of Systems Biology Models and Simulations. In N. W.

Paton, P. Missier, and C. Hedeler, editors, *Data Integration in the Life Sciences, Proceedings; Lecture Notes in Computer Science; 6th International Workshop on Data Integration in the Life Sciences*, volume 5647, pages 5–15. [Chelliah, Vijayalakshmi; Endler, Lukas; Juty, Nick; Laibe, Camille; Li, Chen; Rodriguez, Nicolas; Le Novere, Nicolas] EMBL European Bioinformat Inst, Cambridge CB10 1SD, England.; Le Novere, N, EMBL European Bioinformat Inst, Wellcome Trust Genome Campus, Cambridge CB10 1SD, England., July 2009.

[67] Nicolas Le Novere. Principled annotation of quantitative models in systems biology. In *Genomes to Systems*, 2008.

[68] Alan Garny, David P. Nickerson, Jonathan Cooper, Rodrigo W. Santos, Andrew K. Miller, Steve Mckeever, Poul M. F. Nielsen, and Peter J. Hunter. CellML and associated tools and techniques. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1878):3017–3043, September 2008.

[69] Leslie M. Loew. The Virtual Cell project. *Novartis Foundation symposium*, 247, 2002.

[70] Dawn Field, Susanna-Assunta A. Sansone, Amanda Collis, Tim Booth, Peter Dukes, Susan K. Gregurick, Karen Kennedy, Patrik Kolar, Eugene Kolker, Mary Maxon, Siân Millard, Alexis-Michel M. Mugabushaka, Nicola Perrin, Jacques E. Remacle, Karin Remington, Philippe Rocca-Serra, Chris F. Taylor, Mark Thorley, Bela Tiwari, and John Wilbanks. Megascience. 'Omics data sharing. *Science (New York, N.Y.)*, 326(5950):234–236, October 2009.

[71] Chris F. Taylor, Dawn Field, Susanna-Assunta Sansone, Jan Aerts, Rolf Apweiler, Michael Ashburner, Catherine A. Ball, Pierre-Alain Binz, Molly Bogue, Tim Booth, Alvis Brazma, Ryan R. Brinkman, Adam Michael Clark, Eric W. Deutsch, Oliver Fiehn, Jennifer Fostel, Peter Ghazal, Frank Gibson, Tanya Gray, Graeme Grimes, John M. Hancock, Nigel W. Hardy, Henning Hermjakob, Randall K. Julian, Matthew Kane, Carsten Kettner, Christopher Kinsinger, Eugene Kolker, Martin Kuiper, Nicolas L. Novere, Jim Leebens-Mack, Suzanna E. Lewis, Phillip Lord, Ann-Marie Mallon, Nishanth Marthandan, Hiroshi Masuya, Ruth McNally, Alexander Mehrle, Norman Morrison, Sandra Orchard, John Quackenbush, James M. Reecy, Donald G. Robertson, Philippe Rocca-Serra, Henry Rodriguez, Heiko Rosenfelder, Javier Santoyo-Lopez, Richard H. Scheuermann, Daniel Schober, Barry Smith, Jason Snape, Christian J. Stoeckert, Keith Tipton, Peter Sterk, Andreas Untergasser, Jo Vandesompele, and Stefan Wiemann. Promoting coherent minimum reporting guidelines for biological and biomedical investigations: the MIBBI project. *Nature Biotechnology*, 26(8):889–896, August 2008.

[72] Abhishek Tiwari. BioPAX or SBML?, July 2009.

[73] Dagmar Waltemath, Richard Adams, Daniel A. Beard, Frank T. Bergmann, Upinder S. Bhalla, Randall Britten, Vijayalakshmi Chelliah, Michael T. Cooling, Jonathan Cooper, Edmund J. Crampin, Alan Garny, Stefan Hoops, Michael Hucka, Peter Hunter, Edda Klipp, Camille Laibe, Andrew K. Miller, Ion Moraru, David Nickerson, Poul Nielsen, Macha Nikolski, Sven Sahle, Herbert M. Sauro, Henning Schmidt, Jacky L. Snoep, Dominic Tolle, Olaf Wolkenhauer, and Nicolas Le Novère. Minimum Information About a Simulation Experiment (MIASE). *PLoS Comput Biol*, 7(4):e1001122+, April 2011.

[74] Melanie Courtot, Nick Juty, Christian Knupfer, Dagmar Waltemath, Anna Zhukova, Andreas Drager, Michel Dumontier, Andrew Finney, Martin Golebiewski, Janna Hastings, Stefan Hoops, Sarah Keating, Douglas B. Kell, Samuel Kerrien, James Lawson, Allyson Lister, James Lu, Rainer Machne, Pedro Mendes, Matthew Pocock, Nicolas Rodriguez, Alice Villeger, Darren J. Wilkinson, Sarala Wimalaratne, Camille Laibe, Michael Hucka, and Nicolas

Le Novere. Controlled vocabularies and semantics in systems biology. *Molecular Systems Biology*, 7(1), October 2011.

[75] Nick Juty, Nick Juty, and Nick Juty. Systems Biology Ontology: Update. *Nature Precedings*, (713), October 2010.

[76] Michael Hucka, Michael Hucka, Frank Bergmann, Stefan Hoops, Sarah Keating, Sven Sahle, James Schaff, Lucian Smith, Darren Wilkinson, Michael Hucka, Frank T. Bergmann, Stefan Hoops, Sarah M. Keating, Sven Sahle, James C. Schaff, Lucian P. Smith, and Darren J. Wilkinson. The Systems Biology Markup Language (SBML): Language Specification for Level 3 Version 1 Core. *Nature Precedings*, (713), October 2010.

[77] Nicolas L. Novere, Michael Hucka, Huaiyu Mi, Stuart Moodie, Falk Schreiber, Anatoly Sorokin, Emek Demir, Katja Wegner, Mirit I. Aladjem, Sarala M. Wimalaratne, Frank T. Bergman, Ralph Gauges, Peter Ghazal, Hideya Kawaji, Lu Li, Yukiko Matsuoka, Alice Villeger, Sarah E. Boyd, Laurence Calzone, Melanie Courtot, Ugur Dogrusoz, Tom C. Freeman, Akira Funahashi, Samik Ghosh, Akiya Jouraku, Sohyoung Kim, Fedor Kolpakov, Augustin Luna, Sven Sahle, Esther Schmidt, Steven Watterson, Guanming Wu, Igor Goryanin, Douglas B. Kell, Chris Sander, Herbert Sauro, Jacky L. Snoep, Kurt Kohn, and Hiroaki Kitano. The Systems Biology Graphical Notation. *Nature Biotechnology*, 27(8):735–741, August 2009.

[78] Dagmar Köhn and Nicolas Le Novère. SED-ML – An XML Format for the Implementation of the MIASE Guidelines. In Monika Heiner and Adelinde Uhrmacher, editors, *Computational Methods in Systems Biology*, volume 5307 of *Lecture Notes in Computer Science*, chapter 15, pages 176–190. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2008.

[79] James B. Bassingthwaighte. Strategies for the Physiome Project. *Annals of Biomedical Engineering*, 28(8):1043–1058, August 2000.

[80] L. M. Loew and J. C. Schaff. The Virtual Cell: a software environment for computational cell biology. *Trends in biotechnology*, 19(10):401–406, October 2001.

[81] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J. Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J. Mungall, Neocles Leontis, Philippe Rocca-Serra, Alan Ruttenberg, Susanna-Assunta Sansone, Richard H. Scheuermann, Nigam Shah, Patricia L. Whetzel, and Suzanna Lewis. The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11):1251–1255, November 2007.

[82] Judith A. Blake and Carol J. Bult. Beyond the data deluge: data integration and bio-ontologies. *Journal of biomedical informatics*, 39(3):314–320, June 2006.

[83] J. Lomax and A. T. McCray. Mapping the gene ontology into the unified medical language system. *Comparative and functional genomics*, 5(4):354–361, 2004.

[84] Seth Carbon, Amelia Ireland, Christopher J. Mungall, ShengQiang Shu, Brad Marshall, Suzanna Lewis, AmiGO Hub, and Web Presence Working Group. AmiGO: online access to ontology and annotation data. *Bioinformatics (Oxford, England)*, 25(2):288–289, January 2009.

[85] Erick Antezana, Ward Blondé, Mikel Egaña, Alistair Rutherford, Robert Stevens, Bernard De Baets, Vladimir Mironov, and Martin Kuiper. BioGateway: a semantic systems biology tool for the life sciences. *BMC bioinformatics*, 10 Suppl 10(Suppl 10):S11+, 2009.

[86] Nicolas Le Novère. Model storage, exchange and integration. *BMC neuroscience*, 7 Suppl 1(Suppl 1):S11+, 2006.

[87] Jonathan Bard, Seung Y. Rhee, and Michael Ashburner. An ontology for cell types. *Genome biology*, 6(2):R21+, 2005.

[88] J. S. Luciano. PAX of mind for pathway researchers. *Drug discovery today*, 10(13):937–942, July 2005.

[89] Lena Stromback and Patrick Lambrix. Representations of molecular pathways: an evaluation of SBML, PSI MI and BioPAX. *Bioinformatics*, 21(24):4401–4407, December 2005.

[90] L. Strömbäck, V. Jakoniene, H. Tan, and P. Lambrix. Representing, storing and accessing molecular interaction data: a review of models and tools. *Briefings in bioinformatics*, 7(4):331–338, December 2006.

[91] B. Aranda, P. Achuthan, Y. Alam-Faruque, I. Armean, A. Bridge, C. Derow, M. Feuermann, A. T. Ghanbarian, S. Kerrien, J. Khadake, J. Kerssemakers, C. Leroy, M. Menden, M. Michaut, L. Montecchi-Palazzi, S. N. Neuhauser, S. Orchard, V. Perreau, B. Roechert, K. van Eijk, and H. Hermjakob. The IntAct molecular interaction database in 2010. *Nucleic Acids Research*, 38(Database issue):D525–D531, October 2009.

[92] Tamara Kulikova, Ruth Akhtar, Philippe Aldebert, Nicola Althorpe, Mikael Andersson, Alastair Baldwin, Kirsty Bates, Sumit Bhattacharyya, Lawrence Bower, Paul Browne, Matias Castro, Guy Cochrane, Karyn Duggan, Ruth Eberhardt, Nadeem Faruque, Gemma Hoad, Carola Kanz, Charles Lee, Rasko Leinonen, Quan Lin, Vincent Lombard, Rodrigo Lopez, Dariusz Lorenc, Hamish McWilliam, Gaurab Mukherjee, Francesco Nardone, Maria P. Pastor, Sheila Plaister, Siamak Sobhany, Peter Stoehr, Robert Vaughan, Dan Wu, Weimin Zhu, and Rolf Apweiler. EMBL Nucleotide Sequence Database in 2006. *Nucleic Acids Research*, 35(suppl 1):D16–D20, January 2007.

[93] Carola Eschenbach and Michael Grüninger, editors. *Ontology (Science)*, volume 183 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2008.

[94] Robert Hoehndorf, Michel Dumontier, Anika Oellrich, Dietrich Rebholz-Schuhmann, Paul N. Schofield, and Georgios V. Gkoutos. Interoperability between Biomedical Ontologies through Relation Expansion, Upper-Level Ontologies and Automatic Reasoning. *PLoS ONE*, 6(7):e22006+, July 2011.

[95] Natalya F. Noy and Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology.

[96] Thomas V. Wal. Folksonomy, February 2007.

[97] Thomas Gruber. Ontology of Folksonomy: A Mash-up of Apples and Oranges. *International Journal on Semantic Web & Information Systems*, 3(2):1–11, 2007.

[98] National Information Standards Organization. *ANSI/NISO Z39.19 - Guidelines for the Construction, Format, and Management of Monolingual Controlled Vocabularies*. National Information Standards Organization, Bethesda, Maryland, U.S.A.

[99] The W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview. http://www.w3.org/TR/owl2-overview/, October 2009.

[100] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[101] Tom Gruber. *Ontology*. Springer-Verlag, 2009.

[102] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, June 2007.

[103] Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, and Mike Smith. OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, June 2009.

[104] Robert Stevens, Alan Rector, and Duncan Hull. What is an ontology? *Ontogenesis*, January 2010.

[105] Nino B. Cocchiarella. *Formal Ontology and Conceptual Realism*, volume 339 of *Synthese Library*. Springer, 2007.

[106] Mary C. MacLeod and Eric M. Rubenstein. *Universals*. February 2010.

[107] Alexander Miller. *Realism*. Fall 2008 edition, 2008.

[108] Andrey Rzhetsky and James A. Evans. War of Ontology Worlds: Mathematics, Computer Code, or Esperanto? *PLoS Comput Biol*, 7(9):e1002191+, September 2011.

[109] Ed J. Barkmeyer and Leo Obrst. Re: [ontolog-forum] Just What Is an Ontology, Anyway? (archive of ontolog-forum mailing list), October 2009.

[110] Franz Baader, Diego Calvanese, Deborah Mcguinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook - Cambridge University Press*. Cambridge University Press, first edition, January 2003.

[111] Diego Calvanese, Jeremy Carroll, Giuseppe De Giacomo, Jim Hendler, Ivan Herman, Bijan Parsia, Peter F. Patel-Schneider, Alan Ruttenberg, Uli Sattler, and Michael Schneider. OWL2 Web Ontology Language Profiles, October 2009.

[112] Matthew Horridge, Sean Bechhofer, and Olaf Noppens. Igniting the OWL 1.1 Touch Paper: The OWL API. In *Proceedings of OWLEd 2007: Third International Workshop on OWL Experiences and Directions*, 2007.

[113] Xiaoshu Wang, Robert Gorlitsky, and Jonas S. Almeida. From XML to RDF: how semantic web technologies will change the design of 'omic' standards. *Nature Biotechnology*, 23(9):1099–1103, September 2005.

[114] D. Quan. Improving life sciences information retrieval using semantic web technology. *Brief Bioinform*, 8(3):172–182, May 2007.

[115] Kei-Hoi Cheung, Andrew Smith, Kevin Yip, Christopher Baker, and Mark Gerstein. Semantic Web Approach to Database Integration in the Life Sciences. pages 11–30. 2007.

[116] Matthew Horridge, Nick Drummond, John Goodwin, Alan L. Rector, Robert Stevens, and Hai Wang. The Manchester OWL Syntax. In Bernardo C. Grau, Pascal Hitzler, Conor Shankey, Evan Wallace, Bernardo C. Grau, Pascal Hitzler, Conor Shankey, and Evan Wallace, editors, *OWLED*, volume 216 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2006.

[117] Barry Smith, Werner Ceusters, Bert Klagges, Jacob Kohler, Anand Kumar, Jane Lomax, Chris Mungall, Fabian Neuhaus, Alan Rector, and Cornelius Rosse. Relations in biomedical ontologies. *Genome Biology*, 6(5):R46+, 2005.

[118] Christine Golbreich, Matthew Horridge, Ian Horrocks, Boris Motik, and Rob Shearer. OBO and OWL: Leveraging Semantic Web Technologies for the Life Sciences. *ISWC 2007*, 4825:169–182, 2007.

[119] Martin Boeker, Ilinca Tudose, Janna Hastings, Daniel Schober, and Stefan Schulz. Unintended consequences of existential quantifications in biomedical ontologies. *BMC Bioinformatics*, 12(1):456+, 2011.

[120] Syed Tirmizi, Stuart Aitken, Dilvan Moreira, Chris Mungall, Juan Sequeda, Nigam Shah, and Daniel Miranker. Mapping between the OBO and OWL ontology languages. *Journal of Biomedical Semantics*, 2(Suppl 1):S3+, 2011.

[121] Daniel Schober, Barry Smith, Suzanna Lewis, Waclaw Kusnierczyk, Jane Lomax, Chris Mungall, Chris Taylor, Philippe R. Serra, and Susanna A. Sansone. Survey-based naming conventions for use in OBO Foundry ontology development. *BMC Bioinformatics*, 10(1):125+, 2009.

[122] Phillip Lord. Components of an Ontology. *Ontogenesis*, January 2010.

[123] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/, May 2004.

[124] Adila Krisnadhi, Frederick Maier, and Pascal Hitzler. OWL and Rules. In *Reasoning Web 2011*. Springer, to appear.

[125] The W3C Consortium. OWL Web Ontology Language Overview, February 2004.

[126] Larisa N. Soldatova and Ross D. King. An ontology of scientific experiments. *Journal of The Royal Society Interface*, 3(11):795–803, December 2006.

[127] Gergely Héja, Péter Varga, Péter Pallinger, and György Surján. Restructuring the foundational model of anatomy. *Studies in health technology and informatics*, 124:755–760, 2006.

[128] Gergely Héja, György Surján, Gergely Lukácsy, Péter Pallinger, and Miklós Gergely. GALEN based formal representation of ICD10. *International Journal of Medical Informatics*, 76(2-3):118–123, February 2007.

[129] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, chapter 26, pages 292–297. Springer Berlin / Heidelberg, Berlin, Heidelberg, 2006.

[130] Boris Motik, Rob Shearer, and Ian Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

[131] Matthew Horridge, Bijan Parsia, and Ulrike Sattler. Explaining Inconsistencies in OWL Ontologies. In *Proceedings of the 3rd International Conference on Scalable Uncertainty Management*, SUM '09, pages 124–137, Berlin, Heidelberg, 2009. Springer-Verlag.

[132] Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and James Hendler. Debugging unsatisfiable classes in OWL ontologies. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(4):268–293, 2005.

[133] Alan L. Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In *K-CAP '03: Proceedings of the 2nd international conference on Knowledge capture*, pages 121–128, New York, NY, USA, 2003. ACM.

[134] Alan Rector, Matthew Horridge, and Nick Drummond. Building Modular Ontologies and Specifying Ontology Joining, Binding, Localizing and Programming Interfaces in Ontologies Implemented in OWL. In Derek Sleeman and Mark Musen, editors, *AAAI Spring Symposium on Symbiotic Relationships between Semantic Web and Knowledge Engineering*, volume Technical Report SS-08-07, pages 69+, Menlo Park, California, 2008. AAAI, The AAAI Press.

[135] Mikel Aranguren, Sean Bechhofer, Phillip Lord, Ulrike Sattler, and Robert Stevens. Understanding and using the meaning of statements in a bio-ontology: recasting the Gene Ontology in OWL. *BMC Bioinformatics*, 8(1):57+, February 2007.

[136] Matthew Horridge and Peter F. Patel-Schneider. OWL 2 Web Ontology Language Manchester Syntax. http://www.w3.org/TR/owl2-manchester-syntax/, October 2009.

[137] Lee Harland, Christopher Larminie, Susanna-Assunta A. Sansone, Sorana Popa, M. Scott Marshall, Michael Braxenthaler, Michael Cantor, Wendy Filsell, Mark J. Forster, Enoch Huang, Andreas Matern, Mark Musen, Jasmin Saric, Ted Slater, Jabe Wilson, Nick Lynch, John Wise, and Ian Dix. Empowering industrial research with shared biomedical vocabularies. *Drug discovery today*, 16(21-22):940–947, September 2011.

[138] Andrew R. Joyce and Bernhard O. Palsson. The model organism as a system: integrating 'omics' data sets. *Nature Reviews Molecular Cell Biology*, 7(3):198–210, March 2006.

[139] Lincoln D. Stein. Integrating biological databases. *Nature Reviews Genetics*, 4(5):337–345, May 2003.

[140] W. Sujansky. Heterogeneous database integration in biomedicine. *J Biomed Inform*, 34(4):285–298, August 2001.

[141] Jonathan D. Wren. URL decay in MEDLINE—a 4-year follow-up study. *Bioinformatics*, 24(11):1381–1385, June 2008.

[142] Jonathan D. Wren and Alex Bateman. Databases, data tombs and dust in the wind. *Bioinformatics*, 24(19):2127–2128, October 2008.

[143] Stella Veretnik, J. Lynn Fink, and Philip E. Bourne. Computational Biology Resources Lack Persistence and Usability. *PLoS Comput Biol*, 4(7):e1000136+, July 2008.

[144] Alvis Brazma, Maria Krestyaninova, and Ugis Sarkans. Standards for systems biology. *Nature Reviews Genetics*, 7(8):593–605, August 2006.

[145] R. Alonso-Calvo, V. Maojo, H. Billhardt, F. Martin-Sanchez, M. García-Remesal, and D. Pérez-Rey. An agent- and ontology-based system for integrating public gene, protein, and disease databases. *J Biomed Inform*, 40(1):17–29, February 2007.

[146] Alan Ruttenberg, Tim Clark, William Bug, Matthias Samwald, Olivier Bodenreider, Helen Chen, Donald Doherty, Kerstin Forsberg, Yong Gao, Vipul Kashyap, June Kinoshita, Joanne Luciano, M. Scott Marshall, Chimezie Ogbuji, Jonathan Rees, Susie Stephens, Gwendolyn Wong, Elizabeth Wu, Davide Zaccagnini, Tonya Hongsermeier, Eric Neumann, Ivan Herman, and Kei H. Cheung. Advancing translational research with the Semantic Web. *BMC bioinformatics*, 8 Suppl 3(Suppl 3):S2+, 2007.

[147] Michel Dumontier. Review of Semantic Integration in the Life Sciences. *Ontogenesis*, January 2010.

[148] Allyson L. Lister. Semantic Integration in the Life Sciences. *Ontogenesis*, January 2010.

[149] C. Goble and R. Stevens. State of the nation in data integration for bioinformatics. *Journal of Biomedical Informatics*, 41(5):687–693, October 2008.

[150] Thomas Hernandez and Subbarao Kambhampati. Integration of biological sources: current systems and challenges ahead. *SIGMOD Rec.*, 33(3):51–60, September 2004.

[151] Aaron Birkland and Golan Yona. BIOZON: a system for unification, management and analysis of heterogeneous biological data. *BMC bioinformatics*, 7(1), February 2006.

[152] Paul Shannon, David Reiss, Richard Bonneau, and Nitin Baliga. The Gaggle: An open-source software system for integrating bioinformatics software and data sources. *BMC Bioinformatics*, 7(1):176+, 2006.

[153] Bjorn Junker, Christian Klukas, and Falk Schreiber. VANTED: A system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics*, 7(1):109+, 2006.

[154] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(01):1–31, January 2003.

[155] Satya S. Sahoo, Olivier Bodenreider, Joni L. Rutter, Karen J. Skinner, and Amit P. Sheth. An ontology-driven semantic mashup of gene and biological pathway information: application to the domain of nicotine dependence. *Journal of biomedical informatics*, 41(5):752–765, October 2008.

[156] Stefan Schulz, Elena Beisswanger, Laszlo van den Hoek, Olivier Bodenreider, and Erik M. van Mulligen. Alignment of the UMLS semantic network with BioTop: methodology and assessment. *Bioinformatics (Oxford, England)*, 25(12):i69–76, June 2009.

[157] Holger Wache, T. Vögele, Ubbo Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information — a survey of existing approaches. In H. Stuckenschmidt, editor, *Proceedings of the IJCAI'01 Workshop on Ontologies and Information Sharing, Seattle, Washington, USA, Aug 4-5*, pages 108–117, 2001.

[158] Marie C. Rousset and Chantal Reynaud. Knowledge representation for information integration. *Inf. Syst.*, 29(1):3–22, 2004.

[159] Jinguang Gu, Baowen Xu, and Xinmeng Chen. An XML query rewriting mechanism with multiple ontologies integration based on complex semantic mapping. *Information Fusion*, 9(4):512–522, October 2008.

[160] Allyson L. Lister, Phillip Lord, Matthew Pocock, and Anil Wipat. Annotation of SBML models through rule-based semantic integration. *Journal of biomedical semantics*, 1 Suppl 1(Suppl 1):S3+, 2010.

[161] Allyson L. Lister, Phillip Lord, Matthew Pocock, and Anil Wipat. Annotation of SBML Models Through Rule-Based Semantic Integration. In Phillip Lord, Susanna-Assunta Sansone, Nigam Shah, Susie Stephens, and Larisa Soldatova, editors, *The 12th Annual Bio-Ontologies Meeting, ISMB 2009*, pages 49+, June 2009.

[162] Tim Berners-Lee. Linked Data - Design Issues. `http://www.w3.org/DesignIssues/LinkedData.html`, July 2006.

[163] Evgeni M. Zdobnov, Rodrigo Lopez, Rolf Apweiler, and Thure Etzold. The EBI SRS server—recent developments. *Bioinformatics*, 18(2):368–373, February 2002.

[164] Sohrab P. Shah, Yong Huang, Tao Xu, Macaire M. Yuen, John Ling, and B. Francis Ouellette. Atlas - a data warehouse for integrative bioinformatics. *BMC bioinformatics*, 6(1):34+, 2005.

[165] Peter Ansell. Model and prototype for querying multiple linked scientific datasets. *Future Generation Computer Systems*, 27(3):329–333, March 2011.

[166] F. Belleau, M. Nolin, N. Tourigny, P. Rigault, and J. Morissette. Bio2RDF: Towards a mashup to build bioinformatics knowledge systems. *Journal of Biomedical Informatics*, 41(5):706–716, October 2008.

[167] Alan Ruttenberg, Jonathan A. Rees, Matthias Samwald, and M. Scott Marshall. Life sciences on the Semantic Web: the Neurocommons and beyond. *Briefings in bioinformatics*, 10(2):193–204, March 2009.

[168] Alistair Miles, Jun Zhao, Graham Klyne, Helen White-Cooper, and David Shotton. OpenFlyData: An exemplar data web integrating gene expression data on the fruit fly Drosophila melanogaster. *Journal of Biomedical Informatics*, 43(5):752–761, October 2010.

[169] Kei-Hoi H. Cheung, Kevin Y. Yip, Andrew Smith, Remko Deknikker, Andy Masiar, and Mark Gerstein. YeastHub: a semantic web use case for integrating data in the life sciences domain. *Bioinformatics (Oxford, England)*, 21 Suppl 1, June 2005.

[170] Andrew Smith, Kei H. Cheung, Kevin Yip, Martin Schultz, and Mark Gerstein. LinkHub: a Semantic Web system that facilitates cross-database queries and information retrieval in proteomics. *BMC Bioinformatics*, 8(Suppl 3):S5+, 2007.

[171] Helena F. Deus, Romesh Stanislaus, Diogo F. Veiga, Carmen Behrens, Ignacio I. Wistuba, John D. Minna, Harold R. Garner, Stephen G. Swisher, Jack A. Roth, Arlene M. Correa, Bradley Broom, Kevin Coombes, Allen Chang, Lynn H. Vogel, and Jonas S. Almeida. A Semantic Web Management Model for Integrative Biomedical Informatics. *PLoS ONE*, 3(8):e2946+, August 2008.

[172] Kazuharu Arakawa, Yohei Yamada, Kosaku Shinoda, Yoichi Nakayama, and Masaru Tomita. GEM System: automatic prototyping of cell-wide metabolic pathway models from genomes. *BMC Bioinformatics*, 7(1):168+, 2006.

[173] Minoru Kanehisa, Michihiro Araki, Susumu Goto, Masahiro Hattori, Mika Hirakawa, Masumi Itoh, Toshiaki Katayama, Shuichi Kawashima, Shujiro Okuda, Toshiaki Tokimatsu, and Yoshihiro Yamanishi. KEGG for linking genomes to life and the environment. *Nucleic acids research*, 36(Database issue):D480–484, January 2008.

[174] R. Overbeek, N. Larsen, G. D. Pusch, M. D'Souza, E. Selkov, N. Kyrpides, M. Fonstein, N. Maltsev, and E. Selkov. WIT: integrated system for high-throughput genome sequence analysis and metabolic reconstruction. *Nucleic acids research*, 28(1):123–125, January 2000.

[175] Maurice Scheer, Andreas Grote, Antje Chang, Ida Schomburg, Cornelia Munaretto, Michael Rother, Carola Söhngen, Michael Stelzer, Juliane Thiele, and Dietmar Schomburg. BRENDA, the enzyme information system in 2011. *Nucleic acids research*, 39(Database issue), January 2011.

[176] Thomas J. Lee, Yannick Pouliot, Valerie Wagner, Priyanka Gupta, David W. Stringer-Calvert, Jessica D. Tenenbaum, and Peter D. Karp. BioWarehouse: a bioinformatics database warehouse toolkit. *BMC bioinformatics*, 7(1):170+, 2006.

[177] Michael Baitaluk, Xufei Qian, Shubhada Godbole, Alpan Raval, Animesh Ray, and Amarnath Gupta. PathSys: integrating molecular interaction graphs for systems biology. *BMC bioinformatics*, 7(1), February 2006.

[178] Sergey Kozhenkov, Yulia Dubinina, Mayya Sedova, Amarnath Gupta, Julia Ponomarenko, and Michael Baitaluk. BiologicalNetworks 2.0 - an integrative view of genome biology data. *BMC Bioinformatics*, 11(1):610+, 2010.

[179] Michael Baitaluk and Julia Ponomarenko. Semantic integration of data on transcriptional regulation. *Bioinformatics (Oxford, England)*, 26(13):1651–1661, July 2010.

[180] Hee-Joon Chung, Chan H. Park, Mi R. Han, Seokho Lee, Jung H. Ohn, Jihoon Kim, Jihun Kim, and Ju H. Kim. ArrayXPath II: mapping and visualizing microarray gene-expression data with biomedical ontologies and integrated biological pathway resources using Scalable Vector Graphics. *Nucleic Acids Research*, 33(suppl 2):W621–W626, July 2005.

[181] L. M. Haas, P. M. Schwarz, P. Kodali, E. Kotlar, J. E. Rice, and W. C. Swope. DiscoveryLink: A system for integrated access to life sciences data sources. *IBM Systems Journal*, 40(2):489–511, 2001.

[182] R. Stevens, P. Baker, S. Bechhofer, G. Ng, A. Jacoby, N. W. Paton, C. A. Goble, and A. Brass. TAMBIS: transparent access to multiple bioinformatics information sources. *Bioinformatics (Oxford, England)*, 16(2):184–185, February 2000.

[183] Robert Stevens, Phillip Lord, and Andrew Gibson. Something Nasty in the Woodshed: The Public Knowledge Model. In *OWLED Workshop on OWL: Experiences and Directions*, November 2006.

[184] Norman W. Paton. Managing and sharing experimental data: standards, tools and pitfalls. *Biochemical Society Transactions*, 36(1):33–36, February 2008.

[185] A. L. Lister, A. R. Jones, M. Pocock, O. Shaw, and A. Wipat. CS-TR Number 1016: Implementing the FuGE Object Model: a Systems Biology Data Portal and Integrator. Technical report, Newcastle University, April 2007.

[186] Clifford Lynch. Big data: How do your data grow? *Nature*, 455(7209):28–29, September 2008.

[187] Frank Gibson, Paul G. Overton, Tom V. Smulders, Simon R. Schultz, Stephen J. Eglen, Colin D. Ingram, Stefano Panzeri, Phil Bream, Evelyne Sernagor, Mark Cunningham, Christopher Adams, Christoph Echtermeyer, Jennifer Simonotto, Marcus Kaiser, Daniel C. Swan, Marty Fletcher, and Phillip Lord. Minimum Information about a Neuroscience Investigation (MINI) Electrophysiology : Nature Precedings. *Nature Precedings*, March 2008.

[188] Juan Antonio A. Vizcaíno, Richard Côté, Florian Reisinger, Joseph M. Foster, Michael Mueller, Jonathan Rameseder, Henning Hermjakob, and Lennart Martens. A guide to the Proteomics Identifications Database proteomics data repository. *Proteomics*, 9(18):4276–4283, September 2009.

[189] Helen Parkinson, Ugis Sarkans, Nikolay Kolesnikov, Niran Abeygunawardena, Tony Burdett, Miroslaw Dylag, Ibrahim Emam, Anna Farne, Emma Hastings, Ele Holloway, Natalja Kurbatova, Margus Lukk, James Malone, Roby Mani, Ekaterina Pilicheva, Gabriella Rustici, Anjan Sharma, Eleanor Williams, Tomasz Adamusiak, Marco Brandizi, Nataliya Sklyar, and Alvis Brazma. ArrayExpress update—an archive of microarray and high-throughput sequencing-based functional genomics experiments. *Nucleic Acids Research*, 39(suppl 1):D1002–D1004, January 2011.

[190] Philippe Rocca-Serra, Marco Brandizi, Eamonn Maguire, Nataliya Sklyar, Chris Taylor, Kimberly Begley, Dawn Field, Stephen Harris, Winston Hide, Oliver Hofmann, Steffen Neumann, Peter Sterk, Weida Tong, and Susanna-Assunta Sansone. ISA software suite: supporting standards-compliant experimental annotation and enabling curation at the community level. *Bioinformatics*, 26(18):2354–2356, September 2010.

[191] Philippe Rocca-Serra, Susanna Sansone, Andy Jones, Allyson Lister, Frank Gibson, Ryan Brinkman, Josef Spindlen, and Michael Miller. XSL transformations for FuGE and FuGE extension documents for HTML and tab-delimited rendering. http://isatab.sourceforge.net/docs/FUGE-and-XSL-transformations-R1.doc, June 2008.

[192] Fran Berman, Geoffrey Fox, and Anthony J. G. Hey. *Grid Computing: Making the Global Infrastructure a Reality*. John Wiley &amp; Sons, Inc., New York, NY, USA, 2003.

[193] Benjamin J. Bornstein, Sarah M. Keating, Akiya Jouraku, and Michael Hucka. LibSBML: an API Library for SBML. *Bioinformatics*, 24(6):880–881, March 2008.

[194] Damian Smedley, Syed Haider, Benoit Ballester, Richard Holland, Darin London, Gudmundur Thorisson, and Arek Kasprzyk. BioMart–biological queries made easy. *BMC genomics*, 10(1):22+, 2009.

[195] Lars J. Jensen, Michael Kuhn, Manuel Stark, Samuel Chaffron, Chris Creevey, Jean Muller, Tobias Doerks, Philippe Julien, Alexander Roth, Milan Simonovic, Peer Bork, and Christian von Mering. STRING 8–a global view on proteins and their functional interactions in 630 organisms. *Nucleic acids research*, 37(Database issue):D412–416, January 2009.

[196] C. J. Proctor, D. A. Lydall, R. J. Boys, C. S. Gillespie, D. P. Shanley, D. J. Wilkinson, and T. B. L. Kirkwood. Modelling the checkpoint response to telomere uncapping in budding yeast. *Journal of The Royal Society Interface*, 4(12):73–90, February 2007.

[197] Carole Proctor and Douglas Gray. Explaining oscillations and variability in the p53-Mdm2 system. *BMC Systems Biology*, 2(1):75+, 2008.

[198] Bente Kofahl and Edda Klipp. Modelling the dynamics of the yeast pheromone pathway. *Yeast*, 21(10):831–850, July 2004.

[199] A. L. Lister, M. Pocock, and A. Wipat. Integration of constraints documented in SBML, SBO, and the SBML Manual facilitates validation of biological models. *Journal of Integrative Bioinformatics*, 4(3):80+, 2007.

[200] Sarah M. Keating, Benjamin J. Bornstein, Andrew Finney, and Michael Hucka. SBMLToolbox: an SBML toolbox for MATLAB users. *Bioinformatics (Oxford, England)*, 22(10):1275–1277, May 2006.

[201] Michael Sintek. XML Tab. http://protegewiki.stanford.edu/wiki/XML_Tab, June 2008.

[202] Nicolas Le Novère, Michael Hucka, Stefan Hoops, Sarah Keating, Sven Sahle, Darren Wilkinson, Michael Hucka, Stefan Hoops, Sarah M. Keating, Nicolas Le Novère, Sven Sahle, and Darren Wilkinson. Systems Biology Markup Language (SBML) Level 2: Structures and Facilities for Model Definitions. *Nature Precedings*, (713), December 2008.

[203] Tommi Aho, Henrikki Almusa, Jukka Matilainen, Antti Larjo, Pekka Ruusuvuori, Kaisa-Leena Aho, Thomas Wilhelm, Harri Lähdesmäki, Andreas Beyer, Manu Harju, Sharif Chowdhury, Kalle Leinonen, Christophe Roos, and Olli Yli-Harja. Reconstruction and Validation of RefRec: A Global Model for the Yeast Molecular Interaction Network. *PLoS ONE*, 5(5):e10662+, May 2010.

[204] S. M. Wimalaratne, M. D. B. Halstead, C. M. Lloyd, E. J. Crampin, and P. F. Nielsen. Biophysical annotation and representation of CellML models. *Bioinformatics*, 25(17):2263–2270, September 2009.

[205] Robert Hoehndorf, Michel Dumontier, John Gennari, Sarala Wimalaratne, Bernard de Bono, Daniel Cook, and Georgios Gkoutos. Integrating systems biology models and biomedical ontologies. *BMC Systems Biology*, 5(1):124+, August 2011.

[206] Natalya F. Noy and Mark A. Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *Int. J. Hum.-Comput. Stud.*, 59(6):983–1024, December 2003.

[207] Li Xu and David W. Embley. Combining the Best of Global-as-View and Local-as-View for Data Integration. In Anatoly E. Doroshenko, Terry A. Halpin, Stephen W. Liddle, Heinrich C. Mayr, Anatoly E. Doroshenko, Terry A. Halpin, Stephen W. Liddle, and Heinrich C. Mayr, editors, *ISTA*, volume 48 of *LNI*, pages 123–136. GI, 2004.

[208] Michael Bada, Kevin Livingston, and Lawrence Hunter. An Ontological Representation of Biomedical Data Sources and Records. *Ontogenesis*, June 2011.

[209] Marvin Schulz, Falko Krause, Nicolas Le Novere, Edda Klipp, and Wolfram Liebermeister. Retrieval, alignment, and clustering of computational models based on semantic annotations. *Molecular Systems Biology*, 7(1), July 2011.

[210] Bijan Parsia. Understanding SWRL (Part 1), August 2007.

[211] Pascal Hitzler. OWL and Rules. In *OWLED 2011*, June 2011.

[212] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. Description Logic Rules. In *Proceeding of the 2008 conference on ECAI 2008: 18th European Conference on Artificial Intelligence*, pages 80–84, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.

[213] Martin O'Connor, Holger Knublauch, Samson Tu, Benjamin Grosof, Mike Dean, William Grosso, and Mark Musen. Supporting Rule System Interoperability on the Semantic Web with SWRL. pages 974–986. 2005.

[214] M. J. O'Connor and A. K. Das. SQWRL: a Query Language for OWL. In *OWL: Experiences and Directions (OWLED), Fifth International Workshop*, 2009.

[215] Aitken Stuart, Korf Roman, Webber Bonnie, and Bard Jonathan. COBrA: a bio-ontology editor. *Bioinformatics*, 21(6):825–826, March 2005.

[216] S. M. Falconer, N. F. Noy, and M. A. Storey. Ontology Mapping - A User Survey. In *The Second International Workshop on Ontology Matching at ISWC 07 + ASWC 07*, 2007.

[217] Chris Stark, Bobby-Joe Breitkreutz, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, and Mike Tyers. BioGRID: a general repository for interaction datasets. *Nucleic Acids Research*, 34(suppl 1):D535–D539, January 2006.

[218] Ethan G. Cerami, Benjamin E. Gross, Emek Demir, Igor Rodchenkov, Özgün Babur, Nadia Anwar, Nikolaus Schultz, Gary D. Bader, and Chris Sander. Pathway Commons, a web resource for biological pathway data. *Nucleic Acids Research*, 39(suppl 1):D685–D690, January 2011.

[219] M. J. O'Connor, R. D. Shankar, S. W. Tu, C. I. Ny, and A. K. Dasulas. Developing a Web-Based Application using OWL and SWRL. In *AAAI Spring Symposium*, 2008.

[220] Christoph Lange. Krextor — An Extensible XML->RDF Extraction Framework. In Chris Bizer, Sören Auer, and Gunnar A. Grimnes, editors, *5th Workshop on Scripting and Development for the Semantic Web, Colocated with ESWC 2009*, May 2009.

[221] Michael Hucka. SBML Groups Proposal (2009-09). http://sbml.org/Community/Wiki/SBML_Level_3_Proposals/Groups_Proposal_%282009-09%29, September 2009.

[222] Martin O'Connor. SQWRLQueryTab. http://protege.cim3.net/cgi-bin/wiki.pl?SQWRLQueryTab, February 2010.

[223] C. J. Proctor. Personal Communication, 2009.

[224] María del Mar d. e. l. . M. Roldán-García, Ismael Navas-Delgado, Amine Kerzazi, Othmane Chniber, Joaquín Molina-Castro, and José F. Aldana-Montes. KA-SB: from data integration to large scale reasoning. *BMC bioinformatics*, 10 Suppl 10(Suppl 10):S5+, 2009.

[225] Christine Golbreich and Evan K. Wallace. hasKey, in OWL 2 Web Ontology Language: New Features and Rationale. http://www.w3.org/TR/owl2-new-features/#F9:a_Keys, October 2009.

[226] Web Ontology Working Group. OWL Web Ontology Language Use Cases and Requirements, February 2004.

[227] Stefan Schulz, Kent Spackman, Andrew James, Cristian Cocos, and Martin Boeker. Scalable representations of diseases in biomedical ontologies. *Journal of Biomedical Semantics*, 2(Suppl 2):S6+, 2011.

[228] Allyson L. Lister, Ruchira S. Datta, Oliver Hofmann, Roland Krause, Michael Kuhn, Bettina Roth, and Reinhard Schneider. Live Coverage of Intelligent Systems for Molecular Biology/European Conference on computational biology (ISMB/ECCB) 2009. *PLoS computational biology*, 6(1):e1000640+, January 2010.

[229] Allyson L. Lister, Ruchira S. Datta, Oliver Hofmann, Roland Krause, Michael Kuhn, Bettina Roth, and Reinhard Schneider. Live Coverage of Scientific Conferences Using Web Technologies. *PLoS Comput Biol*, 6(1):e1000563+, January 2010.