Probabilistic Bounded Reachability for Stochastic Hybrid Systems

Fedor Shmarov

School of Computing

University of Newcastle

A thesis submitted for the degree of

Doctor of Philosophy

January 2018

Acknowledgements

I would like to say a massive thank you to my supervisor, Dr. Paolo Zuliani for the tremendous amount of support and guidance in undertaking this research, and for playing an active role in my professional development.

Also, I want to thank Dr. Curtis Madsen, Dr. Nicola Paoletti and Dr. Ezio Bartocci for the very productive collaborations which resulted into several scientific publications.

Moreover, I wish to express my gratitude to my examiners, Prof. Maciej Koutny and Prof. Martin Fränzle for their expertise and professional assessment of my work.

Finally, I thank my family and friends for their moral support in my day-to-day life.

This project has been supported by award N00014-13-1-0090 of the US Office of Naval Research.

Abstract

Stochastic parametric hybrid systems provide a means of formalising automata with continuous nonlinear dynamics, discrete interruptions, and parametric uncertainty (*e.g.* randomness and/or nondeterminism). They can be used for modelling a vast class of cyber-physical systems – machines comprising physical components orchestrated by a digital control (*e.g.* medical devices, self-driving cars, and aircraft autopilots). Assuring correct and safe behaviour of such systems is crucial as human lives are often involved.

One of the main problems in system verification is reachability analysis. It amounts to determining whether the studied model reaches an unsafe state during its evolution. Introduction of parametric randomness allows the formulation of a quantitative version of the problem – computing the probability of reaching the undesired state.

Reachability analysis is a highly challenging problem due to its general undecidability for hybrid systems and undecidability of nonlinear arithmetic (*e.g.* involving trigonometric functions) over the real numbers. A common approach in this case is to solve a simpler, yet useful, problem. In particular, there are techniques for solving reachability rigorously up to a given numerical precision.

The central problem of this research is probabilistic reachability analysis of hybrid systems with random and nondeterministic parameters. In this thesis I have developed two new distinct techniques: a formal approach, based on formal reasoning which provides absolute numerical guarantees; and a statistical one, utilising Monte Carlo sampling that gives statistical guarantees. Namely, the former computes an interval which is guaranteed to contain the exact reachability probability value, while the latter returns an interval containing the probability value with some statistical confidence.

By providing weaker guarantees, the statistical approach is capable of handling difficult cases more efficiently than the formal one, which in turn, can be used for parameter set synthesis in the absence of random uncertainty. The latter is one of the key problems in system modelling: identifying sets of parameter values for which a given model satisfies the desired behaviour.

I have implemented the described techniques in the publicly available tool `ProbReach`, which I have then applied to several realistic case studies such as the synthesis of safe and robust controllers for artificial pancreas and the design of UVB treatment for psoriasis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mathematical modelling and model verification have a greatly positive impact on the system design process. They can be used for rejecting faulty implementations which would be very costly and time-consuming, or even impossible to investigate experimentally. Also, *in silico* (computational) analysis can provide further guidance and predictions for the physical experiments.

Stochastic hybrid systems find application in modelling numerous real world systems from various domains. For instance, they can model biological systems such as gene regulatory networks and DNA replication [58], closed-loop (with feedback) systems such as insulin delivery for patients with type 1 diabetes [44] (also known as artificial pancreas), and cyber-physical systems such as powertrains [49], wind turbines [87] and autonomous underwater vehicles [19].

Verification of stochastic hybrid systems allows solving important problems such as probabilistic reachability and safety analysis [2], devising control and planning strategies [27], and parameter set synthesis [60].

This chapter aims to introduce the relevant background and the related work in the field of verification of stochastic parametric hybrid systems, *i.e.*, hybrid systems parametrised by random and nondeterministic initial conditions. Then it states the aim and the objectives of this work, and declares the contributions of this thesis according to the identified goals.

1.1 Background

1.1.1 Hybrid Systems

Hybrid systems [3] can be seen as a generalisation of finite-state machines depicting the continuous behaviour. Hybrid systems comprise continuous components - *flows* behaving according to the laws of physics and generally modelled using nonlinear ordinary differential equations (ODEs), and discrete control - *jumps* specifying the discrete state changes between the flows. The discrete transitions are represented by a set of Boolean predicates defining when the transition between the continuous flows may take place. Also, when a discrete transition takes place it typically causes a *reset* of the initial values of continuous variables in the successor flow. Besides, hybrid systems can feature invariants – conditions which should be satisfied for all time points in the flow.

A cannonball presents a simple example of a hybrid system. Consider a ball launched with some initial speed and some angle to horizon (see Figure 1.1). The ball's dynamics evolve continuously while it is in the air (*flow*), and the discrete state change occurs when it touches the ground (*jump*). During the contact with the ground its speed is reduced (*reset*).

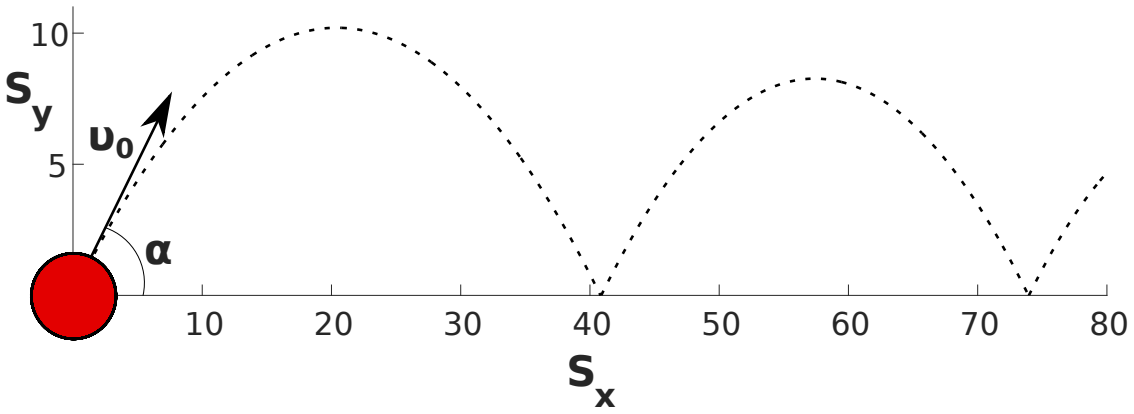


Figure 1.1: A trajectory of a cannonball launched from the point (0,0) with initial speed v_0 and angle to horizon α.

Systems featuring uncertainties can be modelled by means of stochastic parametric hybrid systems (SPHS). There are different variants of SPHSs depending on the level of abstraction, starting from systems with random and nondeterministic parameters

[58, 78, 89] to systems with stochastic dynamics (defined by stochastic differential equations).

1.1.2 Model Checking

Model checking is a set of techniques for the verification of finite state machines [5, 12]. Since its introduction in the 1980's [14, 67] it has been successfully applied to the verification of hardware and software systems of various complexity [41]. Model checking allows reasoning about numerous system properties in a sound and rigorous manner (unlike the approaches based on testing).

A model checker is an automated tool which takes a model of a system and a desired property formulated in the defined specification language, and correctly decides whether the corresponding property holds or not. If the latter is the case, a counter-example falsifying the property is provided.

Reachability is one the most important properties studied in verification. It asks whether a specified state (set of states) is reachable. Many system properties (*e.g.*, safety, unreachable states) can be expressed in terms of reachability.

One of the fundamental problems in model checking is *state space explosion* – the number of states of the model can be incredibly large. Approaches such as symbolic and bounded model checking allow dealing with this problem rather successfully.

Symbolic Model Checking The main idea behind symbolic model checking is to represent and treat a finite state system as a Boolean function [4, 16, 62]. This approach proved to be quite powerful for finite state systems, but extending their application from finite state machines to hybrid systems introduces some complications. Dealing with the latter implies that the system's continuous flows must be taken into account. As symbolic model checking cannot easily deal with the infinite state space introduced by the continuous dynamics, its applicability is limited to hybrid systems whose continuous behaviour can be abstracted to a finite state space (*e.g.*, affine hybrid system [8, 90]). In this regard, bounded model checking can be applied to a wider range of hybrid systems while still successfully dealing with the state space explosion problem.

Bounded Model Checking The aim of bounded model checking (BMC) [13, 15] is to create a formula of a finite length l and feed it to a Boolean satisfiability (SAT) solver. With respect to reachability analysis this means that reachability of the desired state will be explored for the given depth value l. If for none of the considered paths the goal state is reachable the bound l can be increased. Otherwise, a witness of width l is returned.

BMC demonstrated to be more efficient than symbolic model checking for finite state systems, and it can also be applied to verification of hybrid systems. This requires introducing the appropriate logic on top of SAT such as nonlinear arithmetic (including numerical solvers for the nonlinear ODEs, as in general they may not have explicit solutions [22, 37, 47]). However, the introduction of nonlinear arithmetic adds another problem to the state space explosion – undecidability.

Undecidability of Reachability Checking reachability in hybrid systems is generally undecidable (even for linear hybrid systems)[3]. Furthermore, bounded reachability in hybrid systems featuring nonlinear continuous dynamics is undecidable due to undecidability of nonlinear arithmetic over the reals. Although Tarski has proven that the first-order logic of real polynomials is decidable [84], the problem becomes undecidable when trigonometric functions (*e.g.* $\sin, \cos$) are introduced [54, 70, 88]. This proof is based on the well-known Hilbert's tenth problem – whether a Diophantine equation with any number of unknowns and with integral coefficients is solvable in the integers, and it was proven do be undecidable by Davis, Putnam, Robinson and Matiyasevich [21, 61]. However, the reachability becomes decidable for robust hybrid systems [28, 35] (such systems where the reachability property holds under small input perturbations).

There exist decision procedures which can return one sided guaranteed answers. For example, δ-satisfiability tackles the problem of undecidability over the reals by introducing a procedure which returns one-sided guaranteed answers: one of the two answers can be trusted while the other one is subject to some over-approximation [35, 64]. Such problem, called δ-decision, is decidable – there exists an algorithm which always terminates correctly returning one of the answers above.

Analogously, in [30] the authors introduce a procedure that integrates interval constraint propagation (ICP) and SAT solving techniques to provide one-sided decisions for Boolean combinations on nonlinear arithmetic constraints.

Similarly, in [69] the author introduces a semi-terminating algorithm for safety verification of nonlinear hybrid systems. This algorithm terminates for robust instances of the problem (*i.e.*, safety holds comfortably for some positively perturbed version of the system), and it may run eternally otherwise.

The δ-complete decision procedure is one of the fundamental elements of the novel theory introduced in this thesis.

1.1.3 Delta-Complete Decision Procedure

Delta-complete decision procedures are defined for bounded $\mathcal{L}_{\mathbb{R}}$-sentences, and they correctly decide whether a given sentence is false or its weaker version, called δ-weakening, is true.

Definition 1.1. (Bounded $\mathcal{L}_{\mathbb{R}}$-formula [35]) *A bounded $\mathcal{L}_{\mathbb{R}}$-formula is defined as follows:*

$$\begin{aligned} &t := c \mid x \mid f\big(t(x)\big), \\ &\phi := t(x) > 0 \mid t(x) \geq 0 \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists^{[u,v]} x : \phi \mid \forall^{[u,v]} x : \phi, \end{aligned} \tag{1.1}$$

where c is a constant, x is a variable, f is a computable real function, and $\exists^{[u,v]} x, \forall^{[u,v]} x$ are bounded quantifiers – shorthand for $\exists x \in [u, v]$ and $\forall x \in [u, v]$.

In this thesis, computable real functions are defined in terms of Type 2 computability (see Definition A.2 in Appendix A.2). Informally, a real function is computable if its value can be algorithmically approximated with arbitrary finite precision. An important property of computable functions is that they are continuous (while the opposite is not true) [50].

Definition 1.2. (δ-Weakening [35]) *Given an arbitrary $\delta > 0$ and a bounded $\mathcal{L}_{\mathbb{R}}$-sentence*

$$\phi := Q_1^{X_1} x_1, \ldots, Q_n^{X_n} x_n : \bigwedge_{i=1}^{m} \Big(\bigvee_{j=1}^{k_i} \big(f_{i,j}(x_1, \ldots, x_n) \circ 0\big) \Big),$$

where each $f_{i,j}$ is a real computable function, $Q_i = \{\exists, \forall\}$, and $\circ \in \{>, \geq\}$, its δ-weakening is

$$\phi^{\delta} := Q_1^{X_1} x_1, \ldots, Q_n^{X_n} x_n : \bigwedge_{i=1}^{m} \Big(\bigvee_{j=1}^{k_i} \big(f_{i,j}(x_1, \ldots, x_n) \circ -\delta\big) \Big).$$

Definition 1.3. (δ-complete decision procedure [35]) *Given $\delta > 0$, a δ-complete decision procedure correctly decides whether an arbitrary bounded $\mathcal{L}_{\mathbb{R}}$-sentence is false or its δ-weakening is true, returning* **unsat** *and* **δ-sat** *respectively. When both cases overlap, either answer can be returned.*

From the above definition it is clear that **unsat** means that the given $\mathcal{L}_{\mathbb{R}}$-sentence is false. However, **δ-sat** implies satisfiability of the δ-weakening of the given sentence while its original version can still be false. This is usually referred to as a *false* alarm, *i.e.*, a δ-complete decision procedure returns **δ-sat** for the unsatisfiable $\mathcal{L}_{\mathbb{R}}$-sentence due to the coarse over-approximation introduced by δ.

There are several SAT ODE solvers such as `iSAT-ODE` [23], and `dReal` [36] which can provide δ-decisions.

1.1.4 Related Work

Introducing random parameters to a hybrid system adds a quantitative measure to bounded reachability – the reachability probability. Adding nondeterministic parameters to the system above introduces a range of reachability probabilities (*i.e.* the reachability probability becomes a function of nondeterministic parameters).

Verification of such systems can be done formally – integrating the probability measure of random parameters over the parameter sets satisfying the bounded reachability property, or statistically – sampling the parameter space according to the parameters' distributions and evaluating bounded reachability for each drawn sample. The former provides stronger (absolute) guarantees but suffers from high computational complexity [91], while the latter grants weaker (statistical) guarantees relaxing the complexity [92].

Regarding the formal verification approach, in [85] the authors introduce Stochastic Satisfiability Modulo Theory (SSMT) by extending the classical nonlinear Satisfiability Modulo Theory with randomised quantifiers. However, this work is limited to finite domains (only discrete randomness is supported). In [38] the authors address this problem by extending SSMT to continuous domains (CSSMT) to support continuous randomness. However, the presented approach does not feature ODEs, and the guarantees on the accuracy of the produced results are not discussed. At the same time the techniques discussed in [38] solve a more general problem than bounded reachability

in hybrid systems and can be applied to computing the probability of satisfiability of an arbitrary bounded CSSMT formula.

In [25] the authors present a technique for computing p-boxes using validated ODE integration. However, the technique is restricted to ODE systems and finite-support random parameters. Moreover, it is not clear what guarantees are given for models containing only continuous and/or discrete random parameters: the size of the computed p-box might be quite large. In contrast, the technique presented in Section 3.6.2 of this thesis computes an arbitrarily small interval containing the exact reachability probability for systems featuring both continuous (possibly with unbounded support) and discrete random parameters.

In [1] the authors introduce a technique for computing bounds on reachability probabilities for stochastic parametric hybrid systems, using abstraction by discrete-time Markov chains. The technique is further extended to full Linear Temporal Logic (LTL) and nondeterminism [86]. In [68] the authors give model checking algorithms for Probabilistic Computation Tree Logic (PCTL) formulae over continuous-time stochastic parametric hybrid systems. However, in [1, 68, 86] the continuous state space is handled through finite discretisation, and approximate numerical solutions are provided for the experiments. The algorithms presented in this thesis instead consider continuous time and space, and give full mathematical/numeric guarantees.

Regarding the statistical verification approach, in [24] the authors introduce a statistical model checking technique for verifying hybrid systems with continuous randomness and nondeterminism. However, the presented approach combines SMT decision procedures with the fixed-sample size techniques, *i.e.*, based on Hoeffding's inequality. Also stochastic hybrid systems whose dynamics are defined by ODEs are not discussed in this work. In contrast, the algorithm developed in Section 4.2.2 employs more efficient sequential Bayesian approach, and it incorporates an SMT-based verification procedure that can handle Lipschitz-continuous ODEs.

1.1.5 Related Software

dReal Tool Family The tool `dReach` [51] performs bounded reachability analysis in hybrid system. Given a model of a hybrid system and a reachability depth value l it creates a reachability formula of length l which is then verified using `dReal` –

an SMT solver implementing a δ-complete decision procedure with nonlinear ODEs support. `dReach` was successfully applied for parameter set identification in biological systems such as cardiac disorders and hormone therapy for treating prostate cancer [60]. However, it does not support stochastic parametric hybrid systems, and it bases its verdict on δ-decisions, which implies that the correctness of the *sat* answer is subject to over-approximation introduced by δ. The technique developed in Section 2.3.1 attempts to strengthen the δ-sat verdict, and thus, to provide a more precise answer.

The tool `SReach` [89] combines δ-complete procedures with statistical estimation techniques in order to accommodate SPHSs. Namely, it uses Monte Carlo methods for sampling the domain of random parameters, and each sample is evaluated using `dReach`. As a result of the one-sided guarantees provided by `dReach`, `SReach` produces confidence intervals for the over-approximation of the bounded reachability probability. In contrast, the algorithms developed in Section 4.2.1 and Section 4.2.2 of this thesis provide confidence intervals containing the *exact* probability value by taking into account the under-approximation of the reachability probability (along with its over-approximation).

Also, `SReach` handles nondeterministic parameters at the SMT level (directly inside the δ-complete decision procedure) where complexity grows exponentially with the number of nondeterministic parameters, and it can only estimate the maximum reachability probability. At the same time the technique developed in Section 4.3.1 of this thesis handles nondeterministic parameters by sampling the parameter search space randomly, and allows computing both the maximum and the minimum reachability probabilities.

iSAT Tool Family Analogously to `dReach`, the tool `iSAT3` [74] performs bounded reachability analysis in hybrid systems. It also supports nonlinear ODEs and allows checking the *δ-sat* answer automatically, unlike `dReach`.

The tool `SiSAT` [31] solves probabilistic bounded reachability by returning answers guaranteed to be numerically accurate. However, it does not currently support continuous random parameters in the formal setting. The tool `CSiSAT` [39] solves this problem for continuous random parameters with bounded support. However, it does not provide ODEs support, which significantly limits the range of supported SPHSs.

Other Tools C2E2 [26] is a tool that verifies bounded reachability in nonlinear hybrid systems through computing and analysing over-approximations of systems dynamics. This tool, however, does not yet support systems with random parameters. UPPAAL [55] is an extremely powerful model checker for timed automata, and it has been recently extended to support (dynamic) networks of stochastic timed automata via UPPAAL SMC [20]. PRISM [53] is a state-of-the-art model checker for a variety of discrete-state stochastic systems, but with respect to real-time systems it is limited to probabilistic timed automata. The tool FAUST^2 [81] utilises abstraction techniques to verify nondeterministic continuous-state Markov models, although currently for discrete-time models only. ProHVer computes an upper bound for the maximal reachability probability [94], and handles continuous random parameters via discrete over-approximation only [29].

1.2 Aim and Objectives

The aim of this work is to

devise novel techniques for the verification of stochastic parametric hybrid systems in a numerically rigorous and sound manner.

The following objectives were identified:

- investigate formal techniques for probabilistic reachability analysis and develop an algorithm for computing bounded reachability probability in SPHSs with absolute numerical guarantees,
- explore statistical methods for improving the performance of the formal approach, while providing statistically and numerically accurate results (*i.e.*, the precision for samples evaluation should not affect the correctness of produced result),
- implement attained theoretical findings in a software tool, and apply it to several complex case studies.

1.3 Thesis Outline and Contributions

This section introduces the outline and discusses the contributions of this thesis with respect to the declared aim and objectives.

- **Chapter 2** defines bounded reachability in parametric hybrid systems (PHS) and introduces the evaluation procedure that is based on a δ-complete decision procedure, and which can be used for guaranteed reasoning about parameter subsets of a PHS. Some of the material presented in this chapter was published in [78].

- **Chapter 3** extends the notion of PHS to stochastic parametric hybrid systems (SPHS) by introducing random parameters, and presents a formal approach for computing enclosures (sometimes arbitrarily tight) containing the range of the bounded reachability probability with absolute numerical guarantees. Some of the material presented in this chapter was published in [79].

- **Chapter 4** presents a technique combining Monte Carlo sampling with numerically rigorous evaluation procedure from Chapter 2 for producing confidence intervals for the bounded reachability probability that are both statistically and numerically correct. Some of the material presented in this chapter was published in [78].

- **Chapter 5** describes the architecture and implementation details of the developed tool `ProbReach`, which incorporates the algorithms introduced in Chapters 2, 3 and 4. Some of the material presented in this chapter was published in [76, 77].

- **Chapter 6** demonstrates application of `ProbReach` to several case studies, such as UVB therapy for treating psoriasis and automated synthesis of PID controllers for an artificial pancreas model. Some of the material presented in this chapter was published in [75].

- **Chapter 7** contains final remarks and several directions for future work.

1.4 List of Publications

Portions of the work within this thesis have been documented in the following publications.

F. Shmarov and P. Zuliani, "ProbReach: Verified Probabilistic Delta-Reachability for stochastic parametric hybrid systems," in *HSCC*. ACM, 2015, pp. 134–139.

F. Shmarov and P. Zuliani, "ProbReach: a Tool for Guaranteed Reachability Analysis of stochastic parametric hybrid systems," in *Symbolic and Numerical Methods for Reachability Analysis, 1st International Workshop, SNR 2015*, ser. EPiC Series in Computing, S. Bogomolov and A. Tiwari, Eds., vol. 37, 2015, pp. 40–48.

F. Shmarov and P. Zuliani, "SMT-based Reasoning for Uncertain Hybrid Domains," in *AAAI-16 Workhop on Planning for Hybrid Systems, 30th AAAI Conference on Artificial Intelligence*, 2016, pp. 624–630.

F. Shmarov and P. Zuliani, "Probabilistic Hybrid Systems Verification via SMT and Monte Carlo Techniques" in *HVC*. LNCS, vol. 10028, 2016, pp. 152–168.

F. Shmarov, N. Paoletti, E. Bartocci, S. Lin, S. A. Smolka and P. Zuliani, "SMT-based Synthesis of Safe and Robust PID Controllers for Stochastic Hybrid Systems", in *HVC*. LNCS, vol. 10629, 2017, pp. 131–146.

Chapter 2

Bounded Reachability in Parametric Hybrid Systems

2.1 Introduction

Hybrid systems provide a framework for modelling and verification of systems with continuous components and digital control. One of the most important properties in hybrid systems verification is reachability. It aims at deciding whether a hybrid system reaches some predefined goal state (or set of states).

It was already discussed that reachability is undecidable even for linear hybrid systems. Moreover, its bounded version (when the number of discrete transitions in the reachability analysis is finite) is also undecidable.

By applying a δ-complete decision procedure [35] it is possible to decide whether a bounded reachability question is unsatisfiable – a goal state cannot be reached – or its relaxed version (characterised by some positive over-approximation) holds. However, due to the properties of δ-decision procedures, the latter does not imply that the goal state is reachable. Tools such as `dReach` [51] and `iSAT-ODE` [23] can solve bounded reachability in hybrid systems by incorporating SMT solvers that implement δ-decision procedures.

In this chapter I study parametric hybrid systems (PHS) – systems with parametrised continuous and discrete dynamics. The system's parameters are defined in the initial state and stay invariant during the system's evolution. Here I introduce a procedure

that *attempts* to decide whether bounded reachability is true for all parameter values in a given non-empty subset of the system's parameter space, or whether it is unsatisfiable on this subset. While the latter can be checked using one of the existing tools, the former is a non-trivial task that cannot be handled directly by existing SMT solvers. Also note that it is not guaranteed that this procedure always decides given such parameter subset due to undecidability. This evaluation procedure plays a central role in *probabilistic reachability analysis* and *parameter set synthesis*, which will be discussed in Chapter 3.

In this chapter I give a formal definition of parametric hybrid systems and formulate the bounded reachability property in terms of bounded $\mathcal{L}_{\mathbb{R}}$-sentences. Then I introduce a theoretical evaluation procedure utilising δ-complete decision procedures and discuss several challenges faced during the implementation stage due to the limitations of the existing SMT solvers. Finally, I provide an over-approximating implementation of the theoretical evaluation procedure and identify a subclass of PHSs for which "theory" and "implementation" are equivalent.

2.2 Parametric Hybrid Systems

A parametric hybrid system (PHS) is a hybrid system featuring continuous and discrete parameters [10, 51] whose values are set in the initial state and do not change during the system's evolution. The parameters can be defined over intervals (continuous parameters) or over finite sets of constants (discrete parameters). Formally, a PHS can be defined as the following.

Definition 2.1. (Parametric Hybrid System) *A parametric hybrid system (PHS) is a tuple:*

$$H =< Q, X, P, T, \boldsymbol{init}, \boldsymbol{param}, \boldsymbol{flow}, \boldsymbol{invt}, \boldsymbol{jump}, \boldsymbol{reset}, \boldsymbol{goal} >,$$

where

- $Q := \{q_0, \ldots, q_m\}$ – *set of the discrete states (modes) of the system,*
- $X := \bigtimes_{i=1}^{n} [u_i, v_i] \subset \mathbb{R}^n$ – *domain of the continuous state variables,*
- $P := \bigtimes_{i=1}^{k} [a_i, b_i] \times \bigtimes_{i=k+1}^{k+j} \{a_{(i,1)}, \ldots, a_{(i,d_i)}\} \subset \mathbb{R}^{k+j}$ – *parameter space (Cartesian product of the continuous and the discrete parameter domains),*

- $T > 0 \in \mathbb{R}$ – *time upper bound,*
- $\boldsymbol{param} := \{p_1, \ldots, p_{k+j}\}$ – *list of the system's parameters,*
- $\boldsymbol{init} := \{\boldsymbol{init}_q : P \to X, q \in Q\}$ – *set of initial states. Each* $\boldsymbol{init}_q(\mathbf{p})$ *is a computable function that assigns the value* $\mathbf{x}_0$ *to the continuous dynamics at time* $t = 0$ *in mode* q *(initial mode),*
- $\boldsymbol{flow} := \{\boldsymbol{flow}_q : X \times P \times [0,T] \to X, q \in Q\}$ – *set of continuous system dynamics. Each* $\boldsymbol{flow}_q(\mathbf{x}_0, \mathbf{p}, t)$ *defines an initial value problem (IVP) with Lipschitz-continuous ODEs with initial value* $\mathbf{x}_0$ *at* $t = 0$ *in mode* q*.*
- $\boldsymbol{jump} := \{\boldsymbol{jump}_{(q,q')} : X \times P \times [0,T] \to \mathbb{B}, q, q' \in Q\}$ – *set of discrete system transitions (*$\mathbb{B}$ *is the Boolean set). Each* $\boldsymbol{jump}_{(q,q')}(\mathbf{x}, \mathbf{p}, t)$ *defines a jump from mode* q *to* q' *which may (but does not have to) occur if* $\mathbf{x}$ *and* $\mathbf{p}$ *satisfy the jump condition at some time point* $t \in [0,T]$ *in mode* q*,*
- $\boldsymbol{reset} := \{\boldsymbol{reset}_{(q,q')} : X \times P \to X, q, q' \in Q\}$ – *set of reset functions. Each* $\boldsymbol{reset}_{(q,q')}(\mathbf{x}, \mathbf{p})$ *is a computable function that defines the initial value of the continuous dynamics at time* $t = 0$ *in mode* q' *after taking the transition from mode* q*,*
- $\boldsymbol{invt} := \{\boldsymbol{invt}_q : X \times P \times [0,T] \to \mathbb{B}, q \in Q\}$ – *set of mode invariants. Each* $\boldsymbol{invt}_q(\mathbf{x}, \mathbf{p}, t)$ *defines a condition which should be satisfied by the continuous state variables* $\mathbf{x}$ *and the parameter vector* $\mathbf{p}$ *for all time points in mode* q*,*
- $\boldsymbol{goal} := \{\boldsymbol{goal}_q : X \times P \times [0,T] \to \mathbb{B}, q \in Q\}$ – *set of goal states. Each* $\boldsymbol{goal}_q(\mathbf{x}, \mathbf{p}, t)$ *defines the set of continuous goal states in mode* q *(goal mode),*

and

- *each* $\boldsymbol{jump}$*,* $\boldsymbol{invt}$ *and* $\boldsymbol{goal}$ *is represented by a finite Boolean combination of atomic formulae featuring only computable functions.*

Remark 2.1. (Unique Jumps) *Definition 2.1 assumes that each* $\boldsymbol{jump}_{(q,q')}$ *and* $\boldsymbol{reset}_{(q,q')}$ *define a unique discrete transition between modes* q *and* q'*. However, if it is necessary to introduce multiple jumps between a pair of modes, one can create a new mode* q'' *with the same* $\boldsymbol{flow}$ *and* $\boldsymbol{invt}$ *as in* q' *and define the desired transition for* q *and* q''*.*

The following running example of a simple PHS will be used throughout this thesis for explaining the main concepts.

Example 2.1. (Cannonball Model (CB)) *Consider the following scenario visualised in Figure 1.1. A ball is launched from the point* $(S_x = 0, S_y = 0)$ *with initial speed* $\upsilon_0 = 25$ *and angle to horizon* α *which can take one of the three possible values:* 0.7584, 1.0472 *or* 0.5236. *The horizontal and the vertical distances travelled by the ball are governed by the differential equations* $\frac{dS_x}{dt} = \upsilon \cdot \cos(\alpha)$ *and* $\frac{dS_y}{dt} = \upsilon \cdot \sin(\alpha) - g \cdot t$ *(where* $g = 9.8$*), respectively. After the ball reaches the ground it bounces, and its speed is multiplied by the drag coefficient* K*, that can take any value within the interval* $[0.5, 0.9]$. *The described system consists of a single mode and a single jump. A PHS formalizing this model can be defined as follows:*

- $Q := \{q_0\}$,
- $X := [0, 1000] \times [0, 100] \times [0, 50]$,
- $P := \{0.7854, 1.0472, 0.5236\} \times [0.5, 0.9] \times \{25\}$,
- $T := 10$,
- $\boldsymbol{param} := \{\alpha, K, \upsilon_0\}$,
- $\boldsymbol{init} := \{\boldsymbol{init}_{q_0}(\alpha, K, \upsilon_0) := \{0, 0, \upsilon_0\}\}$,
- $\boldsymbol{flow} := \{\boldsymbol{flow}_{q_0}(S_x^{init}, S_y^{init}, \upsilon^{init}, \alpha, K, \upsilon_0, t) := \{\frac{dS_x}{dt} := \upsilon \cdot \cos(\alpha), \frac{dS_y}{dt} := \upsilon \cdot \sin(\alpha) - 9.8 \cdot t, \frac{dv}{dt} := 0, S_x(0) := S_x^{init}, S_y(0) := S_y^{init}, \upsilon(0) := \upsilon^{init}\}\}$,
- $\boldsymbol{invt} := \emptyset$,
- $\boldsymbol{jump} := \{\boldsymbol{jump}_{(q_0,q_0)}(S_x, S_y, \upsilon, \alpha, K, \upsilon_0, t) := (t > 0) \wedge (S_y = 0)\}$,
- $\boldsymbol{reset} := \{\boldsymbol{reset}_{(q_0,q_0)}(S_x, S_y, \upsilon, \alpha, K, \upsilon_0) := \{S_x, 0, K \cdot \upsilon\}\}$,
- $\boldsymbol{goal} := \{\boldsymbol{goal}_{q_0}(S_x, S_y, \upsilon, \alpha, K, \upsilon_0, t) := (t > 0) \wedge (S_y = 0) \wedge (S_x \geq 100)\}$.

2.3 Bounded Reachability in PHS

I now formally define bounded reachability in PHS. It is easy to check whether the goal mode is reachable in l steps by finding a path π (see Definition 2.2) such that the initial element ($\pi[0]$) of π belongs to the set of initial modes, the last element ($\pi[l]$) of π is in the set of goal modes, and for each pair of successive modes $(\pi[i], \pi[i+1])$ there exists a discrete transition defined by $\mathbf{jump}_{(\pi[i],\pi[i+1])}$ and $\mathbf{reset}_{(\pi[i],\pi[i+1])}$.

Definition 2.2. *Given a PHS H, a path π of depth l is a finite sequence of modes of H such that $\boldsymbol{init}_{\pi[0]} \in \boldsymbol{init}$, $\boldsymbol{jump}_{(\pi[i],\pi[i+1])} \in \boldsymbol{jump}$ for $0 < i < l$, and $\boldsymbol{goal}_{\pi[l]} \in \boldsymbol{goal}$. A trajectory defines a continuous evolution of the system along the given path for the given initial value of the continuous dynamics.*

Let $\mathbf{Paths}(H, l)$ be the set of all such paths for the given PHS H and the reachability depth l. It can be obtained using a breadth-first search (BFS) algorithm [80]. The original algorithm should be modified for this purpose in two ways: a search depth bound l should be introduced, and the algorithm should not terminate after finding the first path and exhaustively explore all paths of specified length l.

It is clear that finding such path π is not enough for concluding the reachability of the goal state as there might not be a trajectory satisfying the corresponding jump conditions, invariants and goal predicates. This requires checking the values of the continuous dynamics over $P \times [0, T]$. Thus, bounded reachability can be formulated as:

Definition 2.3. (Bounded Reachability) *The bounded reachability property for a PHS H, a reachability depth l, and a subset B of the parameter space of H is defined as the bounded $\mathcal{L}_{\mathbb{R}}$-sentence:*

$$\mathbf{Reach}(H, l, B) := \exists^{B}\mathbf{p}, \exists^{[0,T]}t_0, \forall^{[0,t_0]}t'_0, \cdots, \exists^{[0,T]}t_{|\pi|-1}, \forall^{[0,t_{|\pi|-1}]}t'_{|\pi|-1} :$$

$$\bigvee_{\pi \in \mathbf{Paths}(H,l)} \Bigg[\big(\mathbf{x}_0(t_0) := \boldsymbol{flow}_{\pi[0]}(\boldsymbol{init}_{\pi[0]}(\mathbf{p}), \mathbf{p}, t_0)\big) \wedge \boldsymbol{invt}_{\pi[0]}\big(\mathbf{x}_0(t'_0), \mathbf{p}, t'_0\big) \wedge$$

$$\bigwedge_{i=0}^{|\pi|-2} \Big[\big(\mathbf{x}_{i+1}(t_{i+1}) := \boldsymbol{flow}_{\pi[i+1]}(\boldsymbol{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}), \mathbf{p}, t_{i+1})\big) \wedge$$

$$\boldsymbol{jump}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}, t_i) \wedge \boldsymbol{invt}_{\pi[i+1]}\big(\mathbf{x}_{i+1}(t'_{i+1}), \mathbf{p}, t'_{i+1}\big) \Big] \wedge$$

$$\boldsymbol{goal}_{\pi[|\pi|-1]}\big(\mathbf{x}_{|\pi|-1}(t_{|\pi|-1}), \mathbf{p}, t_{|\pi|-1}\big) \Bigg].$$

The following proposition is necessary to demonstrate that formula **Reach** can be verified using δ-complete decision procedures.

Proposition 2.1. *Formula* **Reach** *comprises a bounded* $\mathcal{L}_{\mathbb{R}}$*-sentence.*

Proof. First of all, functions in **init** and **reset** are computable, and the formulae in **jump**, **invt**, **goal** consist of a finite combination of atomic formulae featuring only computable functions. Moreover, **flow** features Lipschitz-continuous ODEs whose solutions are unique [7, Chapter 6, Theorem 1] and computable [50, Theorem 7.2]. Finally, a path π of length l is a finite Boolean combination of predicates from **jump**, **invt**, **goal**, and the set of all such paths is finite.

As all variables in **Reach** are defined over bounded intervals, **Reach** defines a bounded $\mathcal{L}_{\mathbb{R}}$-sentence. □

Example 2.2. (Bounded Reachability for CB) *The bounded reachability property for the cannonball model (Example 2.1) with reachability depth* $l = 1$ *on the entire parameter space* ($B = P$) *is satisfiable as the corresponding formula* $\mathbf{Reach}(H, 1, P)$ *holds.*

$$\mathbf{Reach}(H,1,P) := \exists^{P}\{\alpha, K, \upsilon_0\}, \exists^{[0,10]} t_0, \exists^{[0,10]} t_1 :$$
$$(\frac{dS_x^{(0)}}{dt_0} = \upsilon^{(0)} \cdot \cos(\alpha)) \wedge (\frac{dS_y^{(0)}(t_0)}{dt_0} = \upsilon^{(0)}(t_0) \cdot \sin(\alpha) - 9.8 \cdot t_0) \wedge (\frac{d\upsilon^{(0)}}{dt_0} = 0) \wedge$$
$$(S_x^{(0)}(0) := 0) \wedge (S_y^{(0)}(0) := 0) \wedge (\upsilon^{(0)}(0) = \upsilon_0) \wedge (S_y^{(0)}(t_0) = 0) \wedge (t_0 > 0) \wedge$$
$$(\frac{dS_x^{(1)}}{dt_1} = \upsilon^{(1)}(t_1) \cdot \cos(\alpha)) \wedge (\frac{dS_y^{(1)}}{dt_1} = \upsilon^{(1)}(t_1) \cdot \sin(\alpha) - 9.8 \cdot t_1) \wedge (\frac{d\upsilon^{(1)}}{dt_1} = 0) \wedge$$
$$(S_x^{(1)}(0) := S_x^{(0)}(t_0)) \wedge (S_y^{(1)}(0) := 0) \wedge (\upsilon^{(1)}(0) := K \cdot \upsilon^{(0)}(t_0)) \wedge$$
$$(S_x^{(1)}(t_1) \geq 100) \wedge (S_y^{(1)}(t_1) = 0).$$

For any value of α *and* K *and given bounds* T *and* l *the landing distance of the ball after a single jump can be obtained analytically as* $S_x = \frac{2\upsilon_0^2 \cos(\alpha)\sin(\alpha)(K^2+1)}{9.8}$*. Thus,* $\mathbf{Reach}(H, 1, P)$ *is satisfiable by* $\mathbf{p} = \{0.7854, 0.8, 25\}$*,* $t_0 = \frac{2\upsilon_0 \sin(\alpha)}{9.8} \approx 3.6077$ *and* $t_1 = \frac{2K\upsilon_0 \sin(\alpha)}{9.8} \approx 2.8862$ *as the distance* $S_x^{(1)}(t_1)$ *travelled by the ball is approximately equal to* 104.5918*.*

In order to check reachability for all values in a given parameter subset it is necessary to introduce a universal quantifier in formula **Reach**, as shown below.

Definition 2.4. (Universal Bounded Reachability) *The universal bounded reachability property for a PHS H, a reachability depth l, and a subset B of the parameter space of H is defined as the bounded $\mathcal{L}_{\mathbb{R}}$-sentence* $\mathbf{Reach}^{\forall}(H, l, B) := \forall^{B}\mathbf{p} : \mathbf{Reach}(H, l, \{\mathbf{p}\})$.

Example 2.3. (Universal Bounded Reachability for CB) *It is easy to see that the universal reachability property* $\mathbf{Reach}^{\forall}(H, 1, P)$ *for the cannonball model (Example 2.1) does not hold, as there are parameter values in P for which the ball fails to reach the distance $S_x^{(1)} = 100$. For example, for* $\mathbf{p} = \{0.7854, 0.5, 25\}$ *the landing distance of the ball is around* 79.7194.

2.3.1 Verifying Bounded Reachability in PHS

Formulae **Reach** and $\mathbf{Reach}^{\forall}$ can be verified by a δ-complete decision procedure as they are defined by bounded $\mathcal{L}_{\mathbb{R}}$-sentences. Given a bounded $\mathcal{L}_{\mathbb{R}}$-sentence and a positive δ, a δ-complete decision procedure correctly decides whether the given sentence is false (returning *unsat*) or its relaxed version (δ-weakening) is true (outputting δ-*sat*). Thus, *unsat* is a stronger answer implying unsatisfiability of the given formula, while δ-*sat* might in fact be a false alarm due to a coarse over-approximation characterised by $\delta > 0$, and therefore, does not guarantee satisfiability of the given bounded $\mathcal{L}_{\mathbb{R}}$-sentence.

Namely, a δ-complete decision procedure returns *unsat* for $\mathbf{Reach}(H, l, B)$ if for all parameter values in $B \subseteq P$ the system H does not reach a goal state; and δ-*sat* if there exists $\mathbf{p} \in B$ such that $\big(\mathbf{Reach}(H, l, B)\big)^{\delta}$ (the weakening of $\mathbf{Reach}(H, l, B)$) is true. Likewise, if the δ-decision procedure returns δ-*sat* for $\mathbf{Reach}^{\forall}$ its satisfiability is not implied. However, if *unsat* is returned for $\neg\mathbf{Reach}^{\forall}(H, l, B)$ (negation of $\mathbf{Reach}^{\forall}$), it means that $\neg\mathbf{Reach}^{\forall}(H, l, B)$ is false, and thus, $\mathbf{Reach}^{\forall}(H, l, B)$ is true, which means that for all parameter values in B a goal state can be reached in l steps.

Algorithm 1 incorporates formulae **Reach**, $\mathbf{Reach}^{\forall}$ and the properties of δ-complete decision procedures. It defines procedure **evaluate** which, given a PHS H, a reachability depth l, a subset B of the system's parameter space P and a positive δ, returns:

- **sat** if for all parameter values in B the goal state is reachable in l steps;
- **unsat** if no values from B satisfy the bounded reachability;

- **undet** if neither of the above can be decided, or a *false* alarm occurred due to a large value of δ used by the δ-complete decision procedure (see Example 3.6). Choosing a smaller δ can sometimes help reducing the number of *false* alarms.

Algorithm 1: evaluate(H, l, B, δ)

Input : H: PHS,
$l \in \mathbb{N}$: reachability depth,
$B \subseteq P$: subset of the system's parameter space,
$\delta > 0$: precision.

Output: **sat** / **unsat** / **undet**.

```
if δ-decision(Reach(H, l, B)) == δ-sat then
    if δ-decision(¬Reach^∀(H, l, B)) == δ-sat then
        return undet;
    return sat;
return unsat;
```

Example 2.4. (Applying Algorithm 1 to CB) *Given the cannonball model (see Example 2.1), with reachability depth* $l = 1$, $\delta = 10^{-3}$ *and parameter subsets* $B_1 = \{0.7854\} \times [0.5, 0.6] \times \{25\}$, $B_2 = \{0.7854\} \times [0.8, 0.9] \times \{25\}$ *and* $B_3 = \{0.7854\} \times [0.7, 0.8] \times \{25\}$, *Algorithm 1 returns* **unsat**, **sat** *and* **undet**, *respectively.*

2.4 Evaluation Procedure Implementation

The evaluation procedure introduced above can be used for exploring the parameter space of a PHS. However, to the best of my knowledge, there are no publicly available SMT solvers that can handle bounded $\mathcal{L}_{\mathbb{R}}$-sentences with arbitrary alternation of quantifiers. Namely, solvers such as `dReal` and `iSAT-ODE` only allow existentially quantified bounded formulae with Lipschitz-continuous ODEs and mode invariants. The latter means that only one free variable per mode can be quantified universally. In other words, formulae of the type $\exists^{[0,T]}t_1, \exists^{[0,T]}t_2, \forall^{[0,T]}t_3 : \phi(t_1, t_2, t_3)$ (where $\phi(t_1, t_2, t_3)$ is some quantifier-free bounded $\mathcal{L}_{\mathbb{R}}$-formula) are supported by the existing solvers implementing δ-decision procedures. However, formulae with two or more universally quantified free variables (*e.g.*, $\exists^{[0,T]}t_1, \forall^{[0,T]}t_2, \forall^{[0,T]}t_3 : \phi(t_1, t_2, t_3)$) currently cannot be

handled. Thus, **Reach** can be verified using one of the available SMT solvers, but formula $\neg\mathbf{Reach}^{\forall}$ introduces $l+1$ universally quantified time variables for the reachability depth l, and therefore, cannot be checked if $l > 0$. As a result, procedure **evaluate** cannot be implemented because formula $\neg\mathbf{Reach}^{\forall}$ cannot be verified by the existing SMT solvers.

This section shows how formula $\mathbf{Reach}^{\forall}$ can be approximated by a set of simpler bounded $\mathcal{L}_{\mathbb{R}}$-sentences supported by the available implementations of δ-complete decision procedures.

2.4.1 Verifying Universal Bounded Reachability

As it is necessary to avoid arbitrary alternation of existential and universal quantifiers in the newly defined formulae, it is assumed that considered PHSs do not feature any invariants. Also, a corresponding *invariant*-free formula **reach** is introduced. It defines the bounded reachability property for a path $\pi \in \mathbf{Paths}(H, l)$ and a parameter value $\mathbf{p} \in P$. It can be seen that $\exists^{B}\mathbf{p} : \bigvee_{\pi\in\mathbf{Paths}(H,l)} \mathbf{reach}(\pi, \mathbf{p}) \Leftrightarrow \mathbf{Reach}(H, l, B)$. Formula **reach** will be used for introducing the main methodological concepts.

$$
\begin{aligned}
&\mathbf{reach}(\pi, \mathbf{p}) := \exists^{[0,T]} t_0, \cdots, \exists^{[0,T]} t_{|\pi|-1} : \\
&\big(\mathbf{x}_0(t_0) := \mathbf{flow}_{\pi[0]}(\mathbf{init}_{\pi[0]}(\mathbf{p}), \mathbf{p}, t_0)\big) \wedge \bigwedge_{i=0}^{|\pi|-2} \Big[\mathbf{jump}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}, t_i) \wedge \\
&\big(\mathbf{x}_{i+1}(t_{i+1}) := \mathbf{flow}_{\pi[i+1]}(\mathbf{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}), \mathbf{p}, t_{i+1})\big)\Big] \wedge \\
&\mathbf{goal}_{\pi[|\pi|-1]}\big(\mathbf{x}_{|\pi|-1}(t_{|\pi|-1}), \mathbf{p}, t_{|\pi|-1}\big).
\end{aligned}
\tag{2.1}
$$

Despite the absence of invariants, they can still be encoded and checked in terms of non-reachability of their complement in each mode along the considered path. However, this requires considering up to l additional reachability formulae similar to **reach** of length smaller than or equal to l. Thus, for simplicity I consider PHSs without invariants.

Now I employ the approach described in [78] to define formula $\mathbf{fail}_j$ for every discrete transition in **reach** such that: 1) each $\mathbf{fail}_j$ features only one universally quantified time variable, and thus, can be handled by current SMT solvers; and 2) their conjunction implies satisfiability of formula **reach**. Intuitively, each $\mathbf{fail}_j$ describes whether the j-th jump condition (if $j < |\pi| - 1$) or the goal predicate (if $j = |\pi| - 1$) is satisfiable

for all time points in $[0, T]$.

Given a path $\pi \in \mathbf{Paths}(H, l)$ and some parameter value $\mathbf{p} \in P$, formula $\mathbf{fail}_j$ is defined as:

$$\begin{aligned}
&\mathbf{fail}_j(\pi, \mathbf{p}) := \exists^{[0,T]} t_0, \cdots, \exists^{[0,T]} t_{j-1}, \forall^{[0,T]} t_j : \\
&\big(\mathbf{x}_0(t_0) := \mathbf{flow}_{\pi[0]}(\mathbf{init}_{\pi[0]}(\mathbf{p}), \mathbf{p}, t_0)\big) \\
&\bigwedge_{i=0}^{j-1} \Big[\mathbf{jump}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}, t_i) \wedge \\
&\big(\mathbf{x}_{i+1}(t_{i+1}) := \mathbf{flow}_{\pi[i+1]}(\mathbf{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}), \mathbf{p}, t_{i+1})\big)\Big] \wedge \\
&\neg \mathbf{jump}_{\pi[j],\pi[j+1]}\big(\mathbf{x}_j(t_j), \mathbf{p}, t_j\big).
\end{aligned} \tag{2.2}$$

if $j < |\pi| - 1$, and as

$$\begin{aligned}
&\mathbf{fail}_j(\pi, \mathbf{p}) := \exists^{[0,T]} t_0, \cdots, \exists^{[0,T]} t_{j-1}, \forall^{[0,T]} t_j : \\
&\big(\mathbf{x}_0(t_0) := \mathbf{flow}_{\pi[0]}(\mathbf{init}_{\pi[0]}(\mathbf{p}), \mathbf{p}, t_0)\big) \\
&\bigwedge_{i=0}^{j-1} \Big[\mathbf{jump}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}, t_i) \wedge \\
&\big(\mathbf{x}_{i+1}(t_{i+1}) := \mathbf{flow}_{\pi[i+1]}(\mathbf{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i), \mathbf{p}), \mathbf{p}, t_{i+1})\big)\Big] \wedge \\
&\neg \mathbf{goal}_{\pi[j]}\big(\mathbf{x}_j(t_j), \mathbf{p}, t_j\big).
\end{aligned} \tag{2.3}$$

if $j = |\pi| - 1$.

Formula (2.2) states that the system arrives at the j-th mode and fails to satisfy the j-th jump, while formula (2.3) asserts the same but for the goal predicate at the j-th mode.

At this point the first condition requiring formulae $\mathbf{fail}_j$ to feature only one universally quantified variable is met. The following proposition establishes implication between the conjunction of formulae $\mathbf{fail}_j$ and **reach**. Note that the opposite implication (from **reach** to $\mathbf{fail}_j$) does not hold in general, as shown in Example 2.5.

Proposition 2.2. *With the definitions in (2.1), (2.2) and (2.3) the following holds:*

$$\bigwedge_{j=0}^{|\pi|-1} \neg \mathbf{fail}_j(\pi, \mathbf{p}) \;\Rightarrow\; \mathbf{reach}(\pi, \mathbf{p}).$$

Proof. Let's introduce the following notations:

$$
\begin{aligned}
J_{n-1}(t_0,\ldots,t_{n-1}) :=& \big(\mathbf{x}_0(t_0) := \mathbf{flow}_{\pi[0]}(\mathbf{init}_{\pi[0]}(\mathbf{p}),\mathbf{p},t_0)\big)\wedge \\
& \bigwedge_{i=0}^{n-2} \big(\mathbf{x}_{i+1}(t_{i+1}) := \mathbf{flow}_{\pi[i+1]}(\mathbf{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i),\mathbf{p}),\mathbf{p},t_{i+1})\big)\wedge \\
& \mathbf{jump}_{(\pi(n-1),\pi(n))}(\mathbf{x}_{\pi(n-1)}(t_{n-1}),\mathbf{p},t_{n-1}), \\
J_n(t_0,\ldots,t_n) :=& \big(\mathbf{x}_0(t_0) := \mathbf{flow}_{\pi[0]}(\mathbf{init}_{\pi[0]}(\mathbf{p}),\mathbf{p},t_0)\big)\wedge \\
& \bigwedge_{i=0}^{n-1} \big(\mathbf{x}_{i+1}(t_{i+1}) := \mathbf{flow}_{\pi[i+1]}(\mathbf{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i),\mathbf{p}),\mathbf{p},t_{i+1})\big)\wedge \\
& \mathbf{goal}_{\pi(n)}(\mathbf{x}_{\pi(n)}(t_n),\mathbf{p},t_n).
\end{aligned}
$$

Note that $\mathbf{x}_{i+1}(t_{i+1}) := \mathbf{flow}_{\pi[i+1]}\big(\mathbf{reset}_{(\pi[i],\pi[i+1])}(\mathbf{x}_i(t_i),\mathbf{p}),\mathbf{p},t_{i+1}\big)$ are assignments. Hence, the notion of negation does not apply to them. Using the given notation and assuming $n = |\pi| - 1$:

$$
\begin{aligned}
&\bigwedge_{j=0}^{n} \neg\mathbf{fail}_j(\pi,\mathbf{p}) \;\Leftrightarrow \Big[\exists^{[0,T]} t_0 : J_0(t_0)\Big]\wedge \\
&\bigwedge_{j=1}^{n} \Big[\forall^{[0,T]} t_0,\ldots,\forall^{[0,T]} t_{j-1},\exists^{[0,T]} t_j : \Big(\bigwedge_{i=0}^{j-1} J_i(t_0,\ldots,t_i)\Big) \rightarrow J_j(t_0,\ldots,t_j)\Big].
\end{aligned}
$$

By Lemma A.1 (see Appendix A.1) it is easy to see that

$$
\begin{aligned}
&\Big[\exists^{[0,T]} t_0 : J_0(t_0)\Big] \wedge \Big[\forall^{[0,T]} t_0,\exists^{[0,T]} t_1 : J_0(t_0) \rightarrow J_1(t_0,t_1)\Big] \Rightarrow \\
&\Big[\exists^{[0,T]} t_0,\exists^{[0,T]} t_1 : J_0(t_0) \wedge J_1(t_0,t_1)\Big].
\end{aligned}
$$

Applying the reasoning above recursively (n times) to $\bigwedge_{j=0}^{n} \neg\mathbf{fail}_j(\pi,\mathbf{p})$ and reverting

the notations the following holds:

$$\Big[\exists^{[0,T]} t_0 : J_0(t_0)\Big] \wedge \Big[\forall^{[0,T]} t_0, \exists^{[0,T]} t_1 : J_0(t_0) \to J_1(t_0,t_1)\Big] \wedge$$
$$\bigwedge_{j=2}^{n} \Big[\forall^{[0,T]} t_0,\ldots,\forall^{[0,T]} t_{j-1}, \exists^{[0,T]} t_j : \Big(\bigwedge_{i=0}^{j-1} J_i(t_0,\ldots,t_i)\Big) \to J_j(t_0,\ldots,t_j)\Big] \Rightarrow$$
$$\Big[\exists^{[0,T]} t_0, \exists^{[0,T]} t_1 : J_0(t_0) \wedge J_1(t_0,t_1)\Big] \wedge$$
$$\bigwedge_{j=2}^{n} \Big[\forall^{[0,T]} t_0,\ldots,\forall^{[0,T]} t_{j-1}, \exists^{[0,T]} t_j : \Big(\bigwedge_{i=0}^{j-1} J_i(t_0,\ldots,t_i)\Big) \to J_j(t_0,\ldots,t_j)\Big] \Rightarrow$$
$$\Big[\exists^{[0,T]} t_0,\ldots,\exists^{[0,T]} t_{j-1}, \exists^{[0,T]} t_j : \Big(\bigwedge_{i=0}^{j} J_i(t_0,\ldots,t_i)\Big)\Big] \Leftrightarrow \mathbf{reach}(\pi,\mathbf{p}).$$

□

Therefore, Proposition 2.2 allows defining a procedure that can be used instead of Algorithm 1 for verifying bounded reachability in PHSs without invariants, and that can be implemented using the existing SMT solvers.

2.4.2 Implementation

Let $\mathbf{Reach}^*$ be the bounded $\mathcal{L}_{\mathbb{R}}$-sentence defined as the following:

$$\mathbf{Reach}^*(H,l,B) := \exists^B \mathbf{p} : \bigwedge_{\pi \in \mathbf{Paths}(H,l)} \Big[\bigvee_{j=0}^{|\pi|-1} \mathbf{fail}_j(\pi,\mathbf{p})\Big].$$

The following holds by Proposition 2.2:

$$\begin{aligned}
\neg\big(\mathbf{Reach}^*(H,l,B)\big) := \neg\Big(\exists^B \mathbf{p}, \bigwedge_{\pi \in \mathbf{Paths}(H,l)} \Big[\bigvee_{j=0}^{|\pi|-1} \mathbf{fail}_j(\pi,\mathbf{p})\Big]\Big) \Leftrightarrow \\
\neg\Big(\exists^B \mathbf{p}, \forall \pi \in \mathbf{Paths}(H,l) : \bigvee_{j=0}^{|\pi|-1} \mathbf{fail}_j(\pi,\mathbf{p})\Big) \Leftrightarrow \\
\forall^B \mathbf{p}, \exists \pi \in \mathbf{Paths}(H,l) : \bigwedge_{j=0}^{|\pi|-1} \neg\mathbf{fail}_j(\pi,\mathbf{p}) \Rightarrow \\
\forall^B \mathbf{p}, \exists \pi \in \mathbf{Paths}(H,l) : \mathbf{reach}(\pi,\mathbf{p}) \Leftrightarrow \mathbf{Reach}^{\forall}(H,l,B).
\end{aligned} \tag{2.4}$$

It can be seen that $\mathbf{Reach}^*$ contains at most one universally quantified variable per mode, and therefore, both $\mathbf{Reach}$ and $\mathbf{Reach}^*$ can be verified using an SMT solver.

The modified version of Algorithm 1 is presented in procedure **compute** (Algorithm 2). Note that Algorithm 2 utilises *solver* (*e.g.*, `dReal`, `iSAT-ODE`) – the implementation of the theoretical δ-decision procedure used by Algorithm 1.

Algorithm 2: compute(H, l, B, δ)

Input : H: PHS,
$l \in \mathbb{N}$: reachability depth,
$B \subseteq P$: subset of the system's parameter space,
$\delta > 0$: precision.

Output: **sat** / **unsat** / **undet**.

```
1 if solver(Reach(H, l, B), δ) == δ-sat then
2     if solver(Reach*(H, l, B), δ) == δ-sat then
3         return undet;
4     return sat;
5 return unsat;
```

The following proposition establishes connection between the procedures **evaluate** and **compute**.

Proposition 2.3. *Given an invariant-free parametric hybrid system H, a reachability depth l, a subset B of the parameter space of H, a positive δ and procedures* **evaluate** *and* **compute** *defined in Algorithm 1 and Algorithm 2 respectively, the following holds:*

$$\big(\mathbf{compute}(H, l, B, \delta) == \mathbf{unsat}\big) \Leftrightarrow \big(\mathbf{evaluate}(H, l, B, \delta) == \mathbf{unsat}\big),$$
$$\big(\mathbf{compute}(H, l, B, \delta) == \mathbf{sat}\big) \Rightarrow \big(\mathbf{evaluate}(H, l, B, \delta) == \mathbf{sat}\big).$$

Proof. The equivalence holds because the *unsat* outcome of $solver(\mathbf{Reach}(H, l, B), \delta)$ is equivalent to $\neg\mathbf{Reach}(H, l, B)$ by definition of the SMT *solver* implementing the δ-complete decision procedure.

Likewise, as $[solver(\mathbf{Reach}^*(H, l, B), \delta) == unsat] \Leftrightarrow \neg\big(\mathbf{Reach}^*(H, l, B)\big) \Rightarrow \mathbf{Reach}^{\forall}(H, l, B)$ and by (2.4) the implication holds too. □

It can be seen that Proposition 2.3 does not establish equivalence between **evaluate** and **compute** (which can be generally done only for $l = 0$ as it introduces only a single universally quantified variable to formula $\mathbf{Reach}^*$) due to the implication introduced by Proposition 2.2 (see also Example 2.5 below).

This does not affect the correctness of the answer (in a sense that **compute** identifies **sat** and **unsat** boxes correctly) but it might affect the accuracy of the result, *i.e.*, by reducing the number of **sat** boxes and increasing the size of the **undet** region instead.

Example 2.5. (Nondeterministic Cannonball) *The cannonball model from Example 2.1 is defined as a PHS with a deterministic jump. However, it can be easily modified to feature a nondeterministic jump by changing the jump condition from* $(t > 0) \wedge (S_y = 0)$ *to* $(S_y = 0)$*. This introduces a second time point* $t = 0$ *where the transition can be enabled. Algorithm 1 returns* **sat** *for the deterministic version of the cannonball model and the parameter box* $B_2 = \{0.7854\} \times [0.8, 0.9] \times \{25\}$ *from Example 2.4. However,* **compute** *returns* **undet** *for* B_2 *when the jump is nondeterministic.*

For the given reachability depth $l = 1$ *there is a single path* $\pi = \{q_0, q_0\}$ *between the initial and the goal modes, and formulae* $\mathbf{fail}_0$ *and* $\mathbf{fail}_1$ *are defined as:*

$$\mathbf{fail}_0(\pi, \alpha, K, \upsilon_0) := \forall^{[0,10]} t_0 : (S_x^{(0)}(0) := 0) \wedge (S_y^{(0)}(0) := 0) \wedge (\upsilon^{(0)}(0) = \upsilon_0) \wedge$$
$$(\frac{dS_x^{(0)}}{dt_0} = \upsilon^{(0)} \cdot \cos(\alpha)) \wedge (\frac{dS_y^{(0)}(t_0)}{dt_0} = \upsilon^{(0)}(t_0) \cdot \sin(\alpha) - 9.8 \cdot t_0) \wedge (\frac{d\upsilon^{(0)}}{dt_0} = 0) \wedge$$
$$(S_y^{(0)}(t_0) \neq 0),$$
$$\mathbf{fail}_1(\pi, \alpha, K, \upsilon_0) := \exists^{[0,10]} t_0, \forall^{[0,10]} t_1 :$$
$$(\frac{dS_x^{(0)}}{dt_0} = \upsilon^{(0)} \cdot \cos(\alpha)) \wedge (\frac{dS_y^{(0)}(t_0)}{dt_0} = \upsilon^{(0)}(t_0) \cdot \sin(\alpha) - 9.8 \cdot t_0) \wedge (\frac{d\upsilon^{(0)}}{dt_0} = 0) \wedge$$
$$(S_x^{(0)}(0) := 0) \wedge (S_y^{(0)}(0) := 0) \wedge (\upsilon^{(0)}(0) = \upsilon_0) \wedge (S_y^{(0)}(t_0) = 0) \wedge$$
$$(\frac{dS_x^{(1)}}{dt_1} = \upsilon^{(1)}(t_1) \cdot \cos(\alpha)) \wedge (\frac{dS_y^{(1)}}{dt_1} = \upsilon^{(1)}(t_1) \cdot \sin(\alpha) - 9.8 \cdot t_1) \wedge (\frac{d\upsilon^{(1)}}{dt_1} = 0) \wedge$$
$$(S_x^{(1)}(0) := S_x^{(0)}(t_0)) \wedge (S_y^{(1)}(0) := 0) \wedge (\upsilon^{(1)}(0) := K \cdot \upsilon^{(0)}(t_0)) \wedge$$
$$\big((S_x^{(1)}(t_1) < 100) \vee (S_y^{(1)}(t_1) \neq 0)\big).$$

Now $\mathbf{fail}_0$ *is unsatisfiable for any parameter value from* B_2*, but* $\mathbf{fail}_1$ *holds for* $\alpha = 0.7854$*,* $K = 0.8$*,* $\upsilon_0 = 25$ *and* $t_0 = 0$ *and any* $t_1 \in [0, 10]$*, as the predicate* $\big((S_x^{(1)}(t_1) < 100) \vee (S_y^{(1)}(t_1) \neq 0)\big)$ *is satisfied for all* $t_1 \in [0, 10]$*. This is because there are only two time points where* $S_y^{(1)}(t_1) = 0$*, but the distance travelled by the ball is* $S_x^{(1)} = 0$ *or* $S_x^{(1)} = \frac{2\upsilon_0^2 \cos(\alpha) \sin(\alpha) K^2}{9.8} \approx 40.816$ *(in both cases* $S_x^{(1)} < 100$*). Thus, formula* $\mathbf{Reach}^*(H, 1, B_2)$ *is satisfiable and, as a result,* **compute** *returns* **undet** *for the parameter box* B_2 *which should be* **sat** *(even in the nondeterministic version). At the same time boxes* B_1 *and*

B_3 from Example 2.4 remain **unsat** *and* **undet**, *respectively.*

2.4.3 PHS with Deterministic Jumps

Despite providing correct **sat** and **unsat** answers, procedure **compute** is not equivalent to the theoretical **evaluate**. However, for PHSs featuring only deterministic jumps the implication in Proposition 2.2 can be turned into equivalence. Intuitively, a deterministic jump can be described as a discrete transition that can be enabled only once (for a single time point of the time domain $[0, T]$) within the corresponding mode for any initial value of the continuous flow. Formally, this can be defined as the following.

Definition 2.5. (Deterministic Jump) *A jump between modes q and q' is deterministic iff for any $\mathbf{x}_0 \in X$ and $\mathbf{p} \in P$ the following holds:*

$$\exists^{[0,T]} t : \boldsymbol{jump}_{(q,q')}(\boldsymbol{flow}(\mathbf{x}_0, \mathbf{p}, t), \mathbf{p}, t) \rightarrow \exists^{[0,T]!} t : \boldsymbol{jump}_{(q,q')}(\boldsymbol{flow}(\mathbf{x}_0, \mathbf{p}, t), \mathbf{p}, t).$$

The following proposition can be proven now for the PHSs with deterministic jumps.

Proposition 2.4. *With the definitions in (2.1), (2.2) and (2.3), and the assumption of deterministic jumps (Definition 2.5) the following holds:*

$$\bigwedge_{j=0}^{|\pi|-1} \neg\mathbf{fail}_j(\pi, \mathbf{p}) \;\Leftrightarrow\; \mathbf{reach}(\pi, \mathbf{p}).$$

Proof. This proposition can be proven by applying Lemma A.2 (see Appendix A.1) instead of Lemma A.1 to the proof of Proposition 2.2. □

The following proposition establishes equivalence between the procedures **evaluate** and **compute** for PHSs with deterministic jumps.

Proposition 2.5. *Given an invariant-free parametric hybrid system H with deterministic jumps, a reachability depth l, a subset B of the parameter space of H, a positive δ and procedures* **evaluate** *and* **compute** *defined in Algorithm 1 and Algorithm 2 respectively, the following holds:*

$$\big(\mathbf{compute}(H, l, B, \delta) == \mathbf{unsat}\big) \Leftrightarrow \big(\mathbf{evaluate}(H, l, B, \delta) == \mathbf{unsat}\big),$$
$$\big(\mathbf{compute}(H, l, B, \delta) == \mathbf{sat}\big) \Leftrightarrow \big(\mathbf{evaluate}(H, l, B, \delta) == \mathbf{sat}\big).$$

Proof. The first equivalence holds by Proposition 2.3.

Likewise, as $[solver(\mathbf{Reach}^*(H, l, B), \delta) == unsat] \Leftrightarrow \neg\big(\mathbf{Reach}^*(H, l, B)\big)$. The latter is equivalent to $\mathbf{Reach}^\forall(H, l, B)$ by applying Proposition 2.4 to formula (2.4). □

Proposition 2.5 demonstrates that procedure **compute** can be used equivalently to **evaluate** for reasoning about nonempty subsets of the parameter space of a parametric hybrid system with deterministic jumps.

2.4.4 Complexity

Studying the computational complexity of procedures **evaluate** and **compute** is outside the scope of this thesis. However, some upper bounds can be deduced from the complexity of the δ-complete decision problems and SMT-based techniques in general. The δ-decision problem for a bounded $\mathcal{L}_\mathbb{R}$-sentence with Lipschitz-continuous ODEs is PSPACE-complete [35, Corollary 39]. Moreover, the algorithmic complexity of a δ-complete decision procedure is exponential in the number of the quantified variables (artefact of SMT solving [6]) and depends on the chosen precision δ (in the worst case with n variables, an ICP algorithm would need to examine $(\frac{1}{\delta})^n = 2^{n \log(\frac{1}{\delta})}$ boxes [34]). In case of bounded reachability the quantified variables are introduced by the system parameters and the time variables. Thus, the computational complexity of procedure **evaluate** is exponential as well as it uses a δ-decision procedure at most twice. Likewise, procedure **compute** also features exponential complexity as it can use a δ-decision procedure up to $l + 1$ times.

2.5 Discussion

In this chapter I presented a technique for bounded reachability analysis of parametric hybrid systems.

Namely, I utilized δ-complete decision procedures to introduce an algorithm (procedure **evaluate**) which, given a parametric hybrid system, a reachability depth, a subset of the system's parameter space and some positive δ, correctly decides a weakened bounded reachability property for the given parameter subset. In particular, my algorithm returns **sat** if for all parameter values in the given subset the goal state is

reachable, **unsat** if none of the parameter values lead the system to the goal state, or **undet** if none of the above can be concluded.

Then it was shown that **evaluate** cannot be implemented due to the current limitations of existing SMT solvers, which allow only one universally quantified variable per predicate. Accordingly, I restricted the usage of invariants in the considered PHSs and modified Algorithm 1. As a result, I developed procedure **compute** (Algorithm 2) which uses $\mathcal{L}_{\mathbb{R}}$-sentences that can be handled by existing implementations of δ-complete decision procedures. The technique introduced in Algorithm 2 can solve the same problem as Algorithm 1, but produces an answer of poorer quality: in general, **compute** returns more **undet** verdicts than **evaluate**.

Finally, I also demonstrated that the theoretical procedure **evaluate** and its implementation **compute** are equivalent if the verified PHS features only deterministic jumps – each jump condition can be satisfied only once within the corresponding mode.

Chapter 3

Bounded Reachability Probability in Stochastic Parametric Hybrid Systems

3.1 Introduction

Hybrid systems with probabilistic behaviour cannot be modelled as PHSs, which feature only nondeterministic parameters. By introducing random parameters into PHS it is possible to compute a probability of reaching the goal state in a finite number of steps. I call such parametric hybrid systems stochastic.

In this chapter I give a formal definition of stochastic parametric hybrid system (SPHS), introduce the notion of a goal set and define the bounded reachability probability function. Then I introduce an algorithm for computing the range of the reachability probability function, and demonstrate that it can also be used in the absence of random parameters. Finally, I present several sub-routines incorporated in the presented algorithm and discuss its computational complexity.

3.2 Stochastic Parametric Hybrid Systems

Stochastic parametric hybrid systems (SPHS) are a generalization of parametric hybrid systems featuring continuous and/or discrete random parameters whose probability measure is defined by a set of probability density functions and probability mass functions, respectively.

Definition 3.1. (Stochastic Parametric Hybrid System) *A stochastic parametric hybrid system is a pair $(H, \mathbb{P})$, where H is a parametric hybrid system (as per Definition 2.1) and $\mathbb{P}$ is a probability measure over a subset of the parameters of H which will be denoted as random.*

Example 3.1. (Stochastic Cannonball (SCB)) *A stochastic version of the cannonball model from Example 2.1 can defined as an SPHS $(H, \mathbb{P})$ with*

$$H =< Q, X, P, T, \boldsymbol{init}, \boldsymbol{param}, \boldsymbol{flow}, \boldsymbol{invt}, \boldsymbol{jump}, \boldsymbol{reset}, \boldsymbol{goal} >,$$

where

- $Q := \{q_0\}$,
- $P := \big\{\{0.7854, 1.0472, 0.5236\}, [0.5, 0.9], (-\infty, \infty)\big\}$,
- $T := 10$,
- $\boldsymbol{init} := \big\{\boldsymbol{init}_{q_0}(\alpha, K, \upsilon_0) := \{\alpha, 0, \upsilon_0, 0\}\big\}$,
- $\boldsymbol{param} := \{\alpha, K, \upsilon_0\}$,
- $\boldsymbol{flow} := \big\{\boldsymbol{flow}_{q_0}(S_x, S_y, \upsilon, t) := \{\frac{dS_x}{dt} = \upsilon{\cdot}\cos(\alpha), \frac{dS_y}{dt} = \upsilon{\cdot}\sin(\alpha){-}9.8{\cdot}t, \frac{d\upsilon}{dt} = 0\}\big\}$,
- $\boldsymbol{invt} := \emptyset$,
- $\boldsymbol{jump} := \big\{\boldsymbol{jump}_{(q_0,q_0)}(S_x, S_y, \upsilon, t) := (t > 0) \wedge (S_y = 0)\big\}$,
- $\boldsymbol{reset} := \big\{\boldsymbol{reset}_{(q_0,q_0)}(S_x, S_y, \upsilon, t) = \{S_x, 0, K \cdot \upsilon, 0\}\big\}$,
- $\boldsymbol{goal} := \big\{\boldsymbol{goal}_{q_0}(S_x, S_y, \upsilon, t) := (S_y = 0) \wedge (S_x \geq 100)\big\}$,

and $\mathbb{P} := \{f_{\upsilon_0}, f_\alpha\}$, where

- $f_{\upsilon_0}(x) := \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$ *- probability density function of a normal distribution with $\mu = 25$ and $\sigma = 3$,*
- $f_\alpha(x) := \begin{cases} 0.9, & \text{if } x = 0.7854, \\ 0.09, & \text{if } x = 1.0472, \\ 0.01, & \text{if } x = 0.5236, \\ 0, & \text{if } x \notin \{0.7854, 1.0472, 0.5236\}. \end{cases}$

The defined SPHS features a continuous nondeterministic parameter K, a continuous random parameter v_0 and a discrete random parameter α.

3.3 Bounded Reachability Probability in SPHS

The extension of a PHS with random parameters allows computing the probability of reaching the goal state in a finite number of steps. This entails integrating the probability measure of the random parameters over the set of the random parameter values satisfying the bounded reachability property. The set of parameter (both random and nondeterministic) values satisfying the bounded reachability property comprise the goal set of the system. In this work I assume that all random parameters are independent.

Definition 3.2. (Goal Set) *The goal set $G \subseteq P$ of a parametric hybrid system H for the reachability depth l is a subset of the parameter space of H such that:*

$$G = \{\mathbf{p} \in P : \mathbf{Reach}(H, l, \{\mathbf{p}\})\}.$$

*G^C is the complement of G in P (*i.e. *$G^C = P \setminus G$).*

Example 3.2. (Goal Set for CB) *Consider the cannonball model from Example 2.1. It features two parameters: discrete parameter α and continuous parameter K. The goal set for this model with reachability depth value $l = 1$ can be obtained analytically as:*

$$G = \bigcup_{i=1}^{3} \{\alpha_i\} \times \left(\left[\sqrt{\frac{9.8 \cdot S_x^{goal}}{2 \cdot v_0^2 \cos(\alpha_i) \sin(\alpha_i)}} - 1, \infty \right) \cap [0.5, 0.9] \right) \times \{v_0\}.$$

As $\alpha \in \{0.7854, 1.0472, 0.5236\}$, $K \in [0.5, 0.9]$, $v_0 \in \{25\}$ and the goal distance $S_x^{goal} \geq 100$,

$$G = \{0.7854\} \times \left(\left[\sqrt{\frac{980}{1250 \cdot \cos(0.7854) \sin(0.7854)}} - 1, \infty \right) \cap [0.5, 0.9] \right) \times \{25\} \cup$$
$$\{1.0472\} \times \left(\left[\sqrt{\frac{980}{1250 \cdot \cos(1.0472) \sin(1.0472)}} - 1, \infty \right) \cap [0.5, 0.9] \right) \times \{25\} \cup$$
$$\{0.5236\} \times \left(\left[\sqrt{\frac{980}{1250 \cdot \cos(0.5236) \sin(0.5236)}} - 1, \infty \right) \cap [0.5, 0.9] \right) \times \{25\} \approx$$
$$\{0.7854\} \times [0.7537, 0.9] \times \{25\} \cup \{1.0472\} \times \emptyset \times \{25\} \cup \{0.5236\} \times \emptyset \times \{25\}.$$

Thus, the goal set is approximately equal to $G = \{0.7854\} \times [0.7537, 0.9] \times \{25\}$.

3.3.1 Bounded Reachability Probability Function

In this section I formally define the notion of reachability probability for SPHS. Borel measurability is a technical condition for well-definedness of probabilities.

Lemma 3.1. (Borel Measurability) *Finite* $\mathcal{L}_{\mathbb{R}}$*-formulae define Borel sets.*

Proof. It is necessary to show that the set of points satisfying a (finite) $\mathcal{L}_{\mathbb{R}}$-formula is Borel. Any $\mathcal{L}_{\mathbb{R}}$-formula is a composition of terms, using computable functions, by comparisons, disjunctions, conjunctions, and quantifications. Computable real functions are continuous, hence, Borel measurable, and therefore, by definition of measurable function (see Definition A.4 in Appendix A.2), comparisons define Borel sets. Disjunction and conjunction correspond to set union and intersection, respectively, so result in Borel sets.

For existential quantification the proof proceeds as follows: let $\varphi(x, y)$ be a $\mathcal{L}_{\mathbb{R}}$-formula and suppose the set $\Phi = \{x, y : \varphi(x, y)\}$ is Borel. Without loss of generality, it needs to be shown that $Y = \{y : \exists x \ \varphi(x, y)\}$ is Borel. It is easy to see that $Y = \{y : \exists x(x, y) \in \Phi\}$, *i.e.*, it is the image of Φ under a continuous function (the projection), and thus, Y is analytic (see Definition A.5 in Appendix A.2). An analytic set is Borel if and only if its complement is analytic (see Corollary 8.3.3 [17]), so it is necessary to show that $Y^c = \{y : \forall x \ \neg\varphi(x, y)\}$ is analytic. (Negation of $\mathcal{L}_{\mathbb{R}}$-formulae is easily obtained by inverting comparisons and set complementation.) Note that the set $\{y : \exists x \ \neg\varphi(x, y)\}$ is analytic (because it can be written as $\{y : \exists x \ (x, y) \notin \Phi\}$). For $n \in \mathbb{N}^+$ we define the sets

$$B^n = \bigcup_{i=1}^{m} B_i^n,$$

where each B_i^n is defined as

$$B_i^n = \{y : \exists x \in I_i^n \ \neg\varphi(x, y)\},$$

and the sets I_i^n's form a disjoint finite covering of the compact domain X of variables x, y (we assume the same domain for simplicity), *i.e.*, $\cup_{i=1}^{m} I_i^n = X$, $\forall i \neq j \ I_i^n \cap I_j^n = \emptyset$, and each I_i^n has Lebesgue measure bounded above by $\frac{1}{n}$. Now, each B^n is analytic,

since it is a finite union of analytic sets. Finally, it is easy to see that

$$Y^c = \{y : \forall x \, \neg\varphi(x, y)\} = \bigcap_{n=1}^{\infty} \bigcup_{i=1}^{m} B_i^n = \bigcap_{n=1}^{\infty} B^n,$$

which is again analytic, since it is a countable intersection of analytic sets. So both Y and Y^c are analytic hence Borel, and this concludes the proof. □

The parameter space P of an SPHS can be divided into two compartments: the domain of *random parameters* P_R (those for which there is an associated probability measure in $\mathbb{P}$), and the domain of *nondeterministic parameters* P_N (those without probability measure). Hence, $P = P_R \times P_N$, and a vector $\mathbf{p} \in P$ can be written as $\mathbf{p} = \{\mathbf{p}_R, \mathbf{p}_N\}$, where $\mathbf{p}_R \in P_R$ and $\mathbf{p}_N \in P_N$. It is always assumed that P_R is bounded (P_N is bounded by definition of PHS), and later it will be shown how to choose the bounds for P_R if the system features continuous random parameters with unbounded support (*e.g.*, normal or exponential).

As the system's parameters may also be nondeterministic, it follows that in general the bounded reachability probability is a function of the nondeterministic parameters.

Definition 3.3. (Bounded Reachability Probability Function) *Given a stochastic parametric hybrid system* $\{H, \mathbb{P}\}$ *and a reachability depth* l*, the* bounded reachability probability *is the function* $\mathbf{Pr} : P_N \to [0, 1]$ *such that for each* $\mathbf{p}_N \in P_N$*:*

$$\mathbf{Pr}(\mathbf{p}_N) = \int_{G(\mathbf{p}_N)} d\mathbb{P},$$

where $G(\mathbf{p}_N)$ *is the projection of the goal set* G *(see Definition 3.2) onto the random parameter space* P_R *for the given* $\mathbf{p}_N$*.*

Proposition 3.1. (Well-definedness of Probability Function) *Function* $\mathbf{Pr} : P_N \to [0, 1]$ *defines a probability.*

Proof. It is sufficient to show that $G(\mathbf{p}_N)$ is Borel for any $\mathbf{p}_N \in P_N$. Note that $G(\mathbf{p}_N) = \Big\{\mathbf{p}_R \in P_R : \exists y = \mathbf{p}_N : \mathbf{Reach}\big(H, l, \{\mathbf{p}_R, y\}\big)\Big\}$. Thus, by Proposition 3.1, $G(\mathbf{p}_N)$ is Borel. □

Example 3.3. (Bounded Reachability Probability Function for SCB) *The bounded reachability probability function* $\mathbf{Pr}$ *for the SCB from Example 3.1 and reach-*

ability depth $l = 1$ can be obtained analytically as:

$$\mathbf{Pr}(K) = \sum_{i=1}^{3} \left[f_\alpha(\alpha_i) \cdot \int_{\sqrt{\frac{980}{\sin(2\alpha_i)(K^2+1)}}}^{\infty} f_{v_0}(x)dx \right],$$

The plot of the function $\mathbf{Pr}(K)$ generated in MATLAB is shown in Figure 3.1.

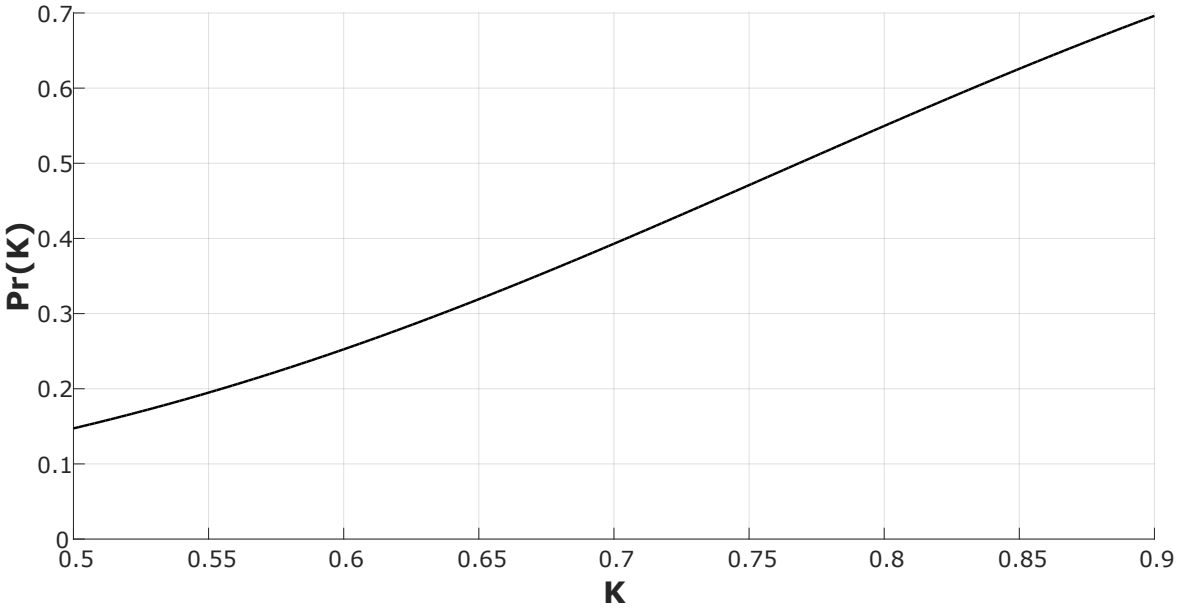


Figure 3.1: Graph of the bounded reachability probability function $\mathbf{Pr}(K)$ obtained analytically for the SCB (Example 3.1) with the reachability depth value $l = 1$.

Note that, if $\mathbf{Pr}$ does not depend on $\mathbf{p}_N$, then it is constant. Also, $\mathbf{Pr}$ needs not to be a continuous function like in Example 3.3. In fact, $\mathbf{Pr}$ can have discontinuities (*i.e.*, points for which the right and left limits differ) because of hybrid dynamics and discrete randomness as shown in the example below.

Example 3.4. (Discontinuous Probability Function) *Consider a version of SCB model with $v_0 = 25$. The goal set $G = \{0.7854\} \times [\sqrt{\frac{980}{1250 \cdot \cos(0.7854)\sin(0.7854)}} - 1, 0.9] \times \{25\}$ for the reachability depth $l = 1$ was obtained in Example 3.2. As the goal state is only reachable for $\alpha_1 = 0.7854$ and $f_\alpha(0.7854) = 0.9$, the reachability probability function can be found as:*

$$\mathbf{Pr}(K) = \begin{cases} 0.9, & \text{if } K \in [\sqrt{\frac{980}{1250 \cdot \cos(0.7854)\sin(0.7854)} - 1}, 0.9], \\ 0, & \text{otherwise.} \end{cases}$$

Thus, the obtained probability function is discontinuous at

$$K = \sqrt{\frac{980}{1250 \cdot \cos(0.7854)\sin(0.7854)}} - 1 \approx 0.7537.$$

This chapter presents an algorithm for computing enclosures for the range of function **Pr**.

Definition 3.4. (Probability Enclosure) *Given a subset of the nondeterministic parameter space $B \subseteq P_N$, a* probability enclosure *for the range of function* **Pr** *on B is an interval $[a, b]$ such that $\forall \mathbf{p}_N \in B : \mathbf{Pr}(\mathbf{p}_N) \in [a, b]$.*

Note that the enclosures depend on the given nondeterministic parameter set, and it is easy to build examples for which probability enclosures are necessarily the full $[0, 1]$ interval, and **Pr** takes any value in $[0, 1]$. Also, it is fair to say that $[0, 1]$ is a valid probability enclosure for any $B \subseteq P_N$. However, by reducing the nondeterministic parameter set it is sometimes possible to reduce the size of the enclosure, and in some cases (*e.g.*, if **Pr** is continuous) the enclosure can be made arbitrarily tight. In other words, it is possible to obtain such $B \subseteq P_N$ for which the size of the interval $[a, b]$ is not greater that some given $\epsilon > 0$.

The rest of the chapter works with parameter subsets represented by boxes (hyper-boxes), which can be defined as the following.

Definition 3.5. (Parameter Box) *A parameter box (hyper-box) is a subset of the parameter space represented by the Cartesian product of closed intervals (possibly of length 0) from the domains of the corresponding parameters. The parameter boxes from P_R and P_N are called random and nondeterministic parameter boxes, respectively.*

3.4 Algorithm for Computing Probability Enclosures

The technique presented in Algorithm 3 computes probability enclosures for the range of the bounded reachability probability function **Pr**. The algorithm takes

- an SPHS $(H, \mathbb{P})$,
- a reachability depth $l \in \mathbb{N}$,

- a precision $\epsilon > 0$ for the size of probability enclosures,
- a constant $\kappa \in (0, \epsilon)$ for bounding the domain of continuous random parameters with the unbounded support,
- a precision vector ρ for nondeterministic parameter boxes, and
- a parameter η controlling the precision of procedure **evaluate**

as input and returns a list of probability enclosures as per Definition 3.4. Such list is indexed by a finite set of disjoint nondeterministic parameter boxes that will cover P_N (*i.e.*, to each nondeterministic parameter box there will be associated a probability enclosure). The main idea behind the algorithm is first partitioning the domain of nondeterministic parameters with boxes and then obtaining a probability enclosure for each such box. The latter is computed by refining the under-approximation and the over-approximation of the definite integral over the random parameter space for the corresponding nondeterministic box. The proof of correctness of Algorithm 3 is given in Proposition 3.2.

If an SPHS does not feature nondeterministic parameters, only one enclosure is returned, and its size is bounded above by the ϵ input if the given system generates robust bounded $\mathcal{L}_{\mathbb{R}}$-sentences (see Theorem 3.1). In general, if nondeterministic parameters are present, the size of the enclosure(s) cannot be controlled by ϵ. This is because the reachability probability function **Pr** might be discontinuous (see Example 3.4) meaning that the enclosure's size cannot be reduced by refining the nondeterministic parameters box. Thus, the precision vector ρ limits the size of the smallest nondeterministic parameter box that will be analysed, allowing termination of the algorithm in the most general case.

Remark 3.1. (Algorithm Notations) *The following operators are applied to lists (queues) and parameter boxes in Algorithm 3:*

- $Q \rightarrow B$ *- assigns to* B *the value of the element at the front of the queue* Q*, and removes that element from* Q*.*
- $Q \leftarrow B$ *- pushes the value of* B *at the back of the queue* Q*. If* B *is a list of boxes, then the elements of* B *are appended at the back of* Q *one by one.*

- Q.clear() - *removes all elements from the list Q.*
- $|Q|$ - *returns the number of elements in the queue Q. When applied to a hyper-box, the result of operation is a vector where each element is the width of the corresponding edge.*

It is also assumed that relation $\mathbf{p}_1 \circ \mathbf{p}_2$ (where $\mathbf{p}_1$ and $\mathbf{p}_2$ are real-valued vectors and $\circ \in \{\geq, >, =, <, \leq\}$) evaluates to true when $\circ$ holds for every pair of corresponding elements of the given vectors.

Algorithm 3 starts by initialising the queue Q (lines 1-5) consisting of triplets $(B_N, [a, b], \Pi_R)$ where B_N is a nondeterministic parameter box, $[a, b]$ is a probability enclosure and Π_R is a list of random parameter boxes.

It is assumed that all types of parameters (random and nondeterministic) are present in the system (line 2). However, the algorithm can still be applied to systems lacking any of the parameter types by introducing "dummy" parameters (lines 3, 4 and 5). These parameters do not affect the system's dynamics, and they are only used for the correct initialisation of the queue Q.

In the outer loop (line 6) the algorithm iterates trough a finite set of triplets $(B_N, [a, b], \Pi_R)$ of the queue Q (line 7) and computes the probability enclosures in the inner loop (line 8). Here the algorithm goes through all random parameter boxes B_R from Π_R (line 9) and calculates the probability value associated with each B_R using procedure **measure** (line 10) defined in Algorithm 5 in Section 3.5.2. The **measure** procedure computes the probability values for continuous and discrete random parameter boxes, and it returns 1 in case no probability measure is defined, *i.e.*, $\mathbb{P} = \emptyset$. Note that function $\mu^+(B_R)$ returns the product of the lengths of all positively-sized edges of box B_R. It is also assumed that if B_R is a singleton (this happens when the system features only discrete random parameters), then $\mu^+(B_R) = 1$. Each pair of boxes (B_N, B_R) is evaluated using procedure **evaluate** (line 12). The precision value δ passed to the decision procedure is calculated as a product of the user-defined parameter η and the width of the shortest positively-sized edge of B_R (line 11). It can be seen that δ always decreases with the size of the analysed random parameter box B_R. (Note that $\eta = \delta$ for SPHS with only discrete random parameters). This is crucial as it allows applying Lemma 3.3 where a special case (when the probability enclosures can be reduced to an arbitrary $\epsilon > 0$) is discussed.

Algorithm 3: Algorithm for Computing Probability Enclosures

Input : $(H, \mathbb{P})$: stochastic parametric hybrid system,
$l \in \mathbb{N}$: reachability depth,
$\epsilon \in \mathbb{Q}^+$: enclosure precision,
$\kappa \in (0, \epsilon)$: constant for distributions with unbounded support,
$\rho \in \mathbb{Q}^+$: precision for nondeterministic parameter box,
$\eta \in \mathbb{Q}^+$: multiplier for controlling precision of δ-decision procedure.

Output: L: list of pairs (nondet. parameter box, probability enclosure).

switch (P_N, P_R) **do**
  case $\big((P_N \neq \emptyset) \wedge (P_R \neq \emptyset)\big)$: **do** $Q \leftarrow (P_N, [0,1], \{P_R\})$;
  case $\big((P_N = \emptyset) \wedge (P_R \neq \emptyset)\big)$: **do** $Q \leftarrow (\{0\}, [0,1], \{P_R\})$;
  case $\big((P_N \neq \emptyset) \wedge (P_R = \emptyset)\big)$: **do** $Q \leftarrow (P_N, [0,1], \{\{0\}\})$;
  case $\big((P_N = \emptyset) \wedge (P_R = \emptyset)\big)$: **do** $Q \leftarrow (\{0\}, [0,1], \{\{0\}\})$;
repeat
  $Q \rightarrow (B_N, [a,b], \Pi_R)$;
  repeat
    $\Pi_R \rightarrow B_R$;
    $[c,d] := \textbf{measure}(B_R, \mathbb{P}, (\epsilon - \kappa)\frac{\mu^+(B_R)}{\mu^+(P_R)})$;
    $\delta := \eta \cdot \min\big(|B_R|^+\big)$;
    switch (**evaluate**$(H, l, B_R \times B_N, \delta)$) **do**
      case unsat: **do** $b := b - c$;
      case sat: **do** $a := a + c$;
      case undet: **do**
        if $(P_R \neq \emptyset)$ **then** $Q_R \leftarrow$ **bisect**(B_R) **else** $Q_R \leftarrow \{B_R\}$;
  until $(|\Pi_R| = 0)$
  if $(|[a,b]| \leq \epsilon) \vee (|B_N| \leq \rho)$ **then**
    $L \leftarrow (B_N, [a,b])$;
  else
    if $(P_N \neq \emptyset)$ **then** $Q_N \leftarrow$ **bisect**(B_N) **else** $Q_N \leftarrow \{B_N\}$;
    for $(B \in Q_N)$ **do** $Q \leftarrow (B, [a,b], Q_R)$;
  Q_N.clear(); Q_R.clear();
until $(|Q| = 0)$
return L;

Now, if **evaluate** returns **unsat** then there is no value in $B_N \times B_R$ for which the goal state is reachable, and the upper bound of the probability enclosure $[a, b]$ can

be reduced (line 13). If **evaluate** returns **sat** then for every value in $B_N \times B_R$ it is possible to reach the goal and the lower bound of the probability enclosure $[a, b]$ can be increased (line 14). Note that the answers returned by the **evaluate** procedure are formally correct because they rely on the *unsat* answer from the δ-complete decision procedure, which can be trusted to be actually correct. Also, note that only the lower bound of the probability interval $[c, d]$ is used for modifying the probability enclosure because it is guaranteed that the true probability value for the given B_R is greater or equal to c.

If **evaluate** returns **undet** it means that $B_N \times B_R$ is a *mixed* box (*i.e.*, it contains some parameter values for which bounded reachability is true, and values for which it is false), or the value of δ is too large. If a "dummy" random parameter was not introduced, B_R is partitioned (line 16) using procedure **bisect**, and each obtained sub-box is pushed to the queue Q_R to be analysed in the next iteration of the outer loop.

Upon exiting the inner loop, the algorithm checks whether the size of the enclosure $[a, b]$ is smaller or equal to ϵ, or each edge of B_N is smaller or equal to the corresponding element of precision vector ρ (line 18). If the condition above is satisfied, then the resulting probability list L is appended with a pair $(B_N, [a, b])$ (line 19). Otherwise, if a dummy nondeterministic parameter was not introduced, box B_N is partitioned (line 21), and for each obtained sub-box B a triplet containing B, probability enclosure $[a, b]$ and a list of partitioned *undetermined* boxes Q_R is pushed to the back of the queue Q (line 22).

The algorithm exits the outer loop and terminates when the queue Q is empty. As it was discussed above, this happens if each triplet satisfies the condition in line 18. The proposition below formally proves correctness of Algorithm 3 when both types of parameters (random and nondeterministic) are present.

Proposition 3.2. *Given a stochastic parametric hybrid system $(H, \mathbb{P})$ featuring random and nondeterministic parameters, a reachability depth $l \in \mathbb{N}$, a probability enclosure precision $\epsilon \in (0, 1)$, a constant $\kappa \in (0, \epsilon)$ for bounding the domain of continuous random parameters with unbounded support, a precision for nondeterministic parameter boxes represented by a vector ρ with positive elements and a parameter $\eta > 0$ controlling the precision of procedure* **evaluate***, Algorithm 3 returns a list of probability enclosures*

as per Definition 3.4.

Proof. It is obvious that for any non-empty subset of the domain of nondeterministic parameters P_N the interval $[0, 1]$ is a valid probability enclosure.

A probability enclosure is only modified if the procedure **evaluate** returns **unsat** or **sat** (lines 13 and 14 respectively) for the given nondeterministic and random parameter boxes. Thus, every probability enclosure $[a, b]$ in the intermediate queue Q or in the resulting list L complies with Definition 3.4.

The condition in line 18 is always satisfied as the nondeterministic parameter boxes are always (remember, no dummy nondeterministic parameters are present here) partitioned (line 21) every time the algorithm completes the inner loop (line 8) which iterates over a finite set of boxes. This is sufficient to show that Algorithm 3 always exits the outer loop (line 6) and, thus, terminates. □

Thus, for each computed probability enclosure it is guaranteed that either its size is smaller than or equal to ϵ, or the size of the corresponding nondeterministic parameter box is smaller than or equal to ρ.

Example 3.5. (Probability Enclosures for SCB) *In this example Algorithm 3 was applied to the stochastic cannonball model from Example 3.1 with* $l = 1$, $\epsilon = 10^{-3}$, $\kappa = 10^{-4}$ *and* $\eta = 10^{-3}$. *The algorithm was executed twice with different values of* ρ *(*$\{5 \cdot 10^{-2}\}$ *and* $\{10^{-2}\}$*, respectively), and it produced 14 probability enclosures in the first case and 64 in the second one. (The algorithm produced 14 enclosures instead of 8 in the first execution due to the floating-point numbers representation). Both outputs were visualised in Figure 3.2.*

Every rectangle in Figure 3.2 is formed by a nondeterministic parameter box (horizontal edge) and a corresponding probability enclosure (vertical edge). It can be seen that the graph of reachability probability function $\mathbf{Pr}(K)$ *from Example 3.3 is fully contained inside the computed probability enclosures, thus, certifying correctness of the result. Also, reducing the value of precision vector* ρ *from* $\{5 \cdot 10^{-2}\}$ *to* $\{10^{-2}\}$ *resulted into obtaining tighter probability enclosures.*

In the example above the size of each enclosure is greater than $\epsilon = 10^{-3}$ implying that Algorithm 3 terminated because every nondeterministic parameter box reached

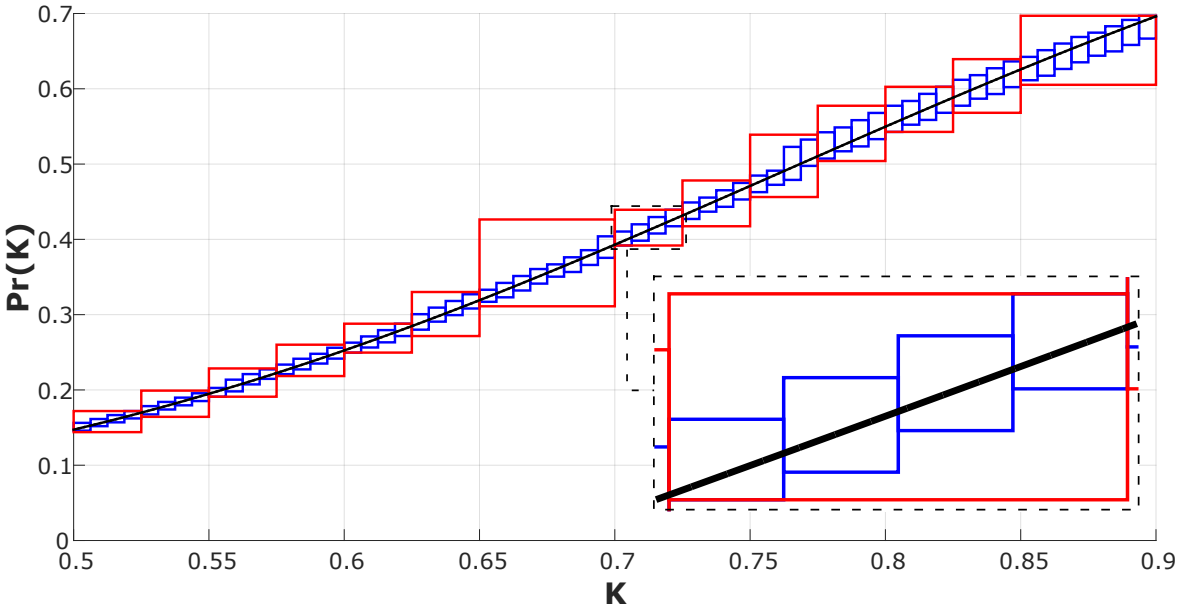


Figure 3.2: Probability enclosures returned by Algorithm 3 for the stochastic cannonball model (Example 3.1), where **red** boxes – probability enclosures computed for $\rho = 5 \cdot 10^{-2}$, **blue** boxes – probability enclosures computed for $\rho = 10^{-2}$ and **black** line – graph of the reachability probability function obtained analytically (Example 3.3).

the terminal size ρ. Section 3.6.2 features an example where the sizes of the probability enclosures can be reduced to the given ϵ.

The quality of the output of Algorithm 3 depends on three input arguments: probability enclosure precision ϵ, precision value ρ for nondeterministic parameter boxes and η – argument for controlling precision of the procedure **evaluate**. The first one defines the main objective – decreasing the sizes of probability enclosures to the arbitrarily small value. Taking into account that this cannot be done in general (due to possible discontinuities in function **Pr**), reducing the values of the second and the third arguments can sometimes (but not always) improve the quality of the output (*i.e.*, reducing the enclosures size). In Chapter 6, Algorithm 3 is applied to several case studies where it is shown how changing ϵ, ρ and η affects the results.

3.5 Auxiliary Procedures

This section introduces the auxiliary procedures utilized by Algorithm 3 for computing the probability values of the parameter boxes and partitioning the parameter space.

3.5.1 Partitioning Parameter Boxes

Algorithm 4 defines procedure **bisect** for partitioning parameter boxes. It is applied in line 16 of Algorithm 3 when a random parameter box is deemed undetermined (procedure **evaluate** returned **undet**) and in line 21 of Algorithm 3 when the size of a nondeterministic parameter box is greater than the specified ρ.

Algorithm 4: bisect(B)

Input : $B := [p_1, q_1] \times \cdots \times [p_n, q_n]$: parameter box.
Output: Q_B: list of parameter boxes.

```
1 for (1 ≤ i ≤ n) do
2     if (p_i ≠ q_i) then
3         Q_i ← [p_i, p_i + (q_i − p_i)/2];
4         Q_i ← [p_i + (q_i − p_i)/2, q_i];
5     else
6         Q_i ← [p_i, q_i];
7 Q_B := Q_1 × ··· × Q_n;
8 return Q_B;
```

Algorithm 4 takes a parameter box B with n edges as input and returns a list Q_B containing at most 2^n equally-sized (in terms of the length of the corresponding edges) boxes comprising B. The algorithm iterates through the edges of box B (line 1) and bisects them (splits in halves) if their sizes are greater than zero (lines 3 and 4) and ignores them otherwise (line 6). All newly obtained intervals are pushed to the corresponding lists Q_i. Thus, after exiting the loop each such list will contain one or two intervals.

The final step (line 7) of the algorithm is computing the resulting list Q_B containing all possible unique combinations of n intervals from the lists Q_i. Each such combination is a parameter box obtained as the Cartesian product of n intervals, where only one interval at a time is picked from the corresponding Q_i.

3.5.2 Computing Probability Values of Parameter Boxes

Algorithm 5 introduces procedure **measure** for computing the probability value for a given random parameter box with the specified precision. Given a random parameter box B_R with k continuous and j discrete random parameters, a probability measure of random parameters $\mathbb{P}$ and a precision value $\zeta > 0$, the algorithm returns an interval $[c, d]$ of size smaller than or equal to ζ containing the value of the integral $\int_{B_R} d\mathbb{P}$.

Algorithm 5: measure(B_R,$\mathbb{P}$,ζ)

Input : $B_R := [p_1, q_1] \times \cdots \times [p_k, q_k] \times \{p_{k+1}\} \times \cdots \{p_{k+j}\}$: parameter box,
$\mathbb{P} := \{f_1, \ldots, f_{k+j}\}$: probability measure,
$\zeta > 0$: precision.
Output: $[c, d]$: probability value interval.

1 $[c, d] := [1, 1]$;
2 **if** $(\mathbb{P} \neq \emptyset)$ **then**
3 $\quad \xi := such\_that\big(((1 + \xi)^k - 1) \leq \zeta\big)$;
4 $\quad$ **for** $(1 \leq i \leq k)$ **do** $[a, b] :=$ **integral**$(f_i, [p_i, q_i], \xi)$; $[c, d] := [c \cdot a, d \cdot b]$;
5 $\quad$ **for** $(1 \leq i \leq j)$ **do** $[c, d] := [f_{k+i}(p_{k+i}) \cdot c, f_{k+i}(p_{k+i}) \cdot d]$;
6 **return** $[c, d]$;

Algorithm 5 returns interval $[1, 1]$ if there is no probability measure associated with the given box (line 2). This happens when the given SPHS does not feature any random parameters and, as a result, Algorithm 3 introduces a dummy random parameter box. The probability value for the continuous parameters is obtained as a product (as all system's parameters are independent) of the integrals (line 4) of their probability density functions over the corresponding intervals. The value of each such integral is calculated using a verified integration procedure introduced in Algorithm 6 (see Section 3.5.3). The precision value ξ (line 3) for computing one dimensional integrals (line 4) is chosen in such a way that $((1 + \xi)^k - 1) \leq \zeta$ (see Section 3.5.4 for explanation). Finally, the resulting interval is multiplied by the value of the probability mass function (line 5) for each discrete parameter value in B_R.

3.5.3 Verified Integration Procedure

The main aim of the verified integration procedure **integral** is computing an interval (or enclosure) which size is not larger than some precision $\xi > 0$ and that contains

the value of the definite integral $F([c,d]) = \int_c^d f(x)dx$ of some function $f(x)$ over an interval $[c,d]$ where f is integrable. The integrands used in Algorithm 3 are probability density functions, which are integrable. The verified integration procedure is a standard adaptive quadrature procedure employing the (1/3) Simpson rule [52]:

$$\int_c^d f(x)\,dx = \frac{|d-c|}{6}(f(c) + 4f(\frac{c+d}{2}) + f(d)) - \frac{|d-c|^5}{2880}f^{(4)}(\gamma) \tag{3.1}$$

where $\gamma \in [c,d]$ and $f^{(4)}$ is the fourth derivative of f. Note the equality sign in (3.1), *i.e.*, the Simpson rule gives a precise way to compute $F([c,d])$, *assuming* knowledge of $\gamma \in [c,d]$.

An interval extension of function $f : X \to Y$ is an operator $[f]$ such that for any interval $[c,d] \subset X$:

$$\forall x \in [c,d] : f(x) \in [f]([c,d]) \subseteq Y$$

An interval version of Simpson's rule can be obtained simply by replacing in (3.1) the occurrences of f and $f^{(4)}$ with their interval extension $[f]$, and by replacing γ with the entire interval $[c,d]$:

$$[F]([c,d]) = \frac{|d-c|}{6}([f](c) + 4[f](\frac{c+d}{2}) + [f](d)) - \frac{|d-c|^5}{2880}[f]^{(4)}([c,d])$$

and thus $F([c,d]) \in [F]([c,d])$. Furthermore, by the definition of integral:

$$F([c,d]) \in \Sigma_{i=1}^{n}[F](\Pi_i) \tag{3.2}$$

where the collection of Π_i's is a partition of $[c,d]$. (The intersections of the intervals in partition Π_i may have (Lebesgue) measure 0, since these have no effect on integration.) In order to guarantee integration with precision ξ it is sufficient to divide $[c,d]$ into n intervals Π_i such that for each Π_i we have $|[F](\Pi_i)| < \xi\frac{|\Pi_i|}{d-c}$. Then, the exact value of the integral will belong to an interval (3.2) of width smaller than ξ. Algorithm 6 features the pseudo-code for a standard adaptive quadrature procedure that computes integral (3.1) up to an arbitrary $\xi > 0$ [40].

3.5.4 Multiple Continuous Random Parameters

Assuming that the system's parameter are independent, the verified integration procedure from Algorithm 6 can be used for calculating $\int_{B_R} d\mathbb{P}$ with arbitrary positive precision (where $B_R \subseteq P_R$ is a multidimensional parameter box). The proposition below states that the precision ξ for integrating each parameter must be stricter than

Algorithm 6: integral$(f, [c,d], \xi)$

Input : $f \in C^5[c,d]$: integrand function,
$[c,d]$: integration domain,
$\xi \in \mathbb{Q}^+$: precision.
Output: interval I.

```
I = [0,0]; Π = ∅;
Q ← [c,d];
while |Q| > 0 do
    Q → [x,y];
    if |[F](x)| ≥ ξ · (y−x)/(d−c) then
        Q ← [x, (x+y)/2]; Q ← [(x+y)/2, y];
    else
        I = I + [F]([x,y]);          // add partial sum to the integral value
        Π ← [x,y];
return I;
```

the given ζ for calculating the multidimensional integral value.

Proposition 3.3. *To compute with precision $\zeta > 0$ the value of the integral $\int_{B_R} d\mathbb{P}$, where $B_R = [c_1, d_1] \times \ldots \times [c_n, d_n]$ it is sufficient to apply Algorithm 6 to each $[c_i, d_i]$ with a precision ξ such that $\zeta \geq (1+\xi)^n - 1$.*

Proof. As all the parameters are independent then $\int_{B_R} d\mathbb{P} = \int_{B_R} \prod_{i=1}^n f_i(x)\, dx$ (where f_i is the probability function of the i-th continuous random parameter in B_R), which by Fubini's theorem can be calculated as $\prod_{i=1}^n \int_{c_i}^{d_i} f_i(x)\, dx$.

Now, using Algorithm 6 it is possible to compute an interval of length ξ_i containing the *exact* value of each integral $\int_{c_i}^{d_i} f_i(x)\, dx$, and let $[e_i, e_i + \xi_i]$ denote such interval. It is thus sufficient to demonstrate how the values ξ_i's should be chosen in order for the integral to be contained in an interval of length ζ.

According to the rules of interval arithmetics, the product of the intervals is contained in the interval:

$$[e_1, e_1+\xi_1] \cdot [e_2, e_2+\xi_2] \cdot \ldots \cdot [e_n, e_n+\xi_n] \;\subseteq\; [\prod_{i=1}^n e_i, \prod_{i=1}^n (e_i + \xi_i)] \tag{3.3}$$

Therefore, the ξ_i's should be chosen such that the interval at the RHS of inclusion

(3.3) has length smaller than ζ, *i.e.*, the following should hold:

$$\prod_{i=1}^{n}(e_i + \xi_i) - \prod_{i=1}^{n} e_i \leq \zeta \tag{3.4}$$

Choosing ξ_i in such a way that (3.4) holds will guarantee that the *exact* value of the product of n integrals is contained in an interval of size ζ. Requiring all the ξ_i's to be equal to the same value ξ, formula (3.4) can be satisfied by assuming the worst case $e_i = 1$ for all i, which gives: $\zeta \geq \prod_{i=1}^{n}(1+\xi) - 1 = (1+\xi)^n - 1$. □

3.5.5 Unbounded Random Parameters

A random parameter with unbounded domain (*e.g.*, Gaussian) cannot be directly used, since it would define an unbounded $\mathcal{L}_{\mathbb{R}}$-formula. Also, Algorithm 6 can be applied only to *bounded* continuous random parameters. However, for any κ such that $0 < \kappa < 1$ it is possible to find a *bounded* region of every random parameter domain over which the integral with respect to the probability measure is larger than $1 - \kappa$. Parameter κ characterizes the error due to bounding the domain of random parameters. Starting with some initial c and d one can apply Algorithm 6 to obtain the value of the integral $\int_c^d f_i(x)\,dx$ by enlarging the bounds (decreasing c and increasing d) until the lower bound of integral $\int_c^d f_i(x)\,dx$ computed by procedure **integral** is greater or equal to $1 - \kappa$ (*i.e.*, $\underline{\int_c^d f_i(x)\,dx} \geq 1 - \kappa$). This can always be done as probability density functions are always positive.

It easy to see that if the system features n independent continuous random parameters and it is required to bound the parameter space in such a way that $\underline{\int_{P_R} d\mathbb{P}} \geq 1 - \kappa$, each parameter with unbounded support should be bounded by $[c_i, d_i]$ such that $\underline{\int_{c_i}^{d_i} f_i(x)\,dx} \geq (1-\kappa)^{\frac{1}{n}}$.

3.6 Algorithm Guarantees

It was proven in Proposition 3.2 that Algorithm 3 always terminates returning a list of valid probability enclosures, which are guaranteed to contain the range of the reachability probability function. Depending on the absence or presence of certain parameter types in the considered SPHS and assumptions about the reachability probability function, Algorithm 3 can also provide guarantees on the size of the computed probability

enclosures. In particular, with the *ϵ-guarantee* all computed enclosures are tighter than some user defined $\epsilon > 0$.

3.6.1 Goal Set Synthesis in PHSs

Algorithm 3 can be applied to parametric hybrid systems *i.e.*, with no random parameters. In this case it does not return probability enclosures, but a list of intervals from the set $\{[0,0],[1,1],[0,1]\}$. The proposition below defines the meaning of the computed intervals.

Proposition 3.4. *Given a parametric hybrid system H, a reachability depth $l \in \mathbb{N}$, a probability enclosure precision $\epsilon \in (0,1)$, a precision for nondeterministic parameter boxes represented by a vector ρ with positive elements and a parameter $\eta > 0$ controlling the precision of the decision procedure* **evaluate***, Algorithm 3 returns a list of intervals L such that for each nondeterministic parameter box B_N indexing L:*

$$\big(L[B_N] = [1,1]\big) \Leftrightarrow \big(\textbf{evaluate}(H, l, B_N, \eta) = \textbf{sat}\big),$$
$$\big(L[B_N] = [0,0]\big) \Leftrightarrow \big(\textbf{evaluate}(H, l, B_N, \eta) = \textbf{unsat}\big),$$
$$\big(L[B_N] = [0,1]\big) \Leftrightarrow \big(\textbf{evaluate}(H, l, B_N, \eta) = \textbf{undet}\big).$$

Proof. In the inner loop (line 8) the algorithm iterates through a list containing random parameter boxes (line 9). Such list always contains a single box, as a dummy random parameter is introduced (line 4), and it is never partitioned (line 16). Thus, every line of the inner loop is executed only once.

The procedure **measure** returns the interval $[c,d] = [1,1]$ for B_R, as $\mathbb{P} = \emptyset$ (line 10). The precision value δ for the procedure **evaluate** is equal to η as B_R is a singleton for which $\min\big(\mu^+(B_R)\big) = 1$ (line 11). As the parameter box B_R is dummy, $\textbf{evaluate}(H, l, B_R \times B_N, \eta)$ (line 12) is equivalent to $\textbf{evaluate}(H, l, B_N, \eta)$. Now, if $\textbf{evaluate}(H, l, B_N, \eta) = \textbf{unsat}$ then the upper bound of the interval $[a,b]$ (which is initially equal to $[0,1]$) is reduced by $c = 1$ (line 13) resulting into $[a,b] = [0,0]$. If $\textbf{evaluate}(H, l, B_N, \eta) = \textbf{sat}$ then the lower bound of $[a,b]$ is increased by $c = 1$ producing $[a,b] = [1,1]$. When $\textbf{evaluate}(H, l, B_N, \eta) = \textbf{undet}$ the enclosure $[a,b]$ is not refined, and thus, $[a,b] = [0,1]$. Also, the condition in line 18 of Algorithm 3 guarantees that the size of each such box (for which interval $[0,1]$ is returned) is smaller than or equal to ρ. □

Remark 3.2. *The returned intervals* $[0,0]$ *and* $[1,1]$ *in Proposition 3.4 are not meant to be probability enclosures* $[0,0]$ *and* $[1,1]$*, since this proposition applies to PHS. Also, for an event to have zero probability does not necessarily mean that such event cannot* **absolutely** *occur. This is because (Lebesgue) integration cannot "see" sets with countable numbers of points — they all integrate to zero. That is, if a system reaches the goal state for no value of the random parameters but one, then it cannot be said that the system is absolutely safe. On the other hand, when no continuous random parameters are present, the returned enclosures imply that a given event never (*$[0,0]$*) or always (*$[1,1]$*) happens.*

From Proposition 3.4 it is clear that for each interval in L the following holds: if $L[B_N] = [1,1]$ then $B_N \subseteq G$, if $L[B_N] = [0,0]$ then $B_N \subseteq G^C$ and $L[B_N] = [0,1]$ implies that B_N contains values from both sets (G and its complement G^C) or it is due to a false alarm (see Example 3.6). Thus, Algorithm 3 can be applied to *parameter set synthesis*: finding a subset of the system's parameter space for which the system reaches the goal state. In this setting Algorithm 3 will partition the parameter space of the system and obtain under-approximations for sets G and G^C.

Example 3.6. (Goal Set Synthesis for CB) *The goal of this experiment is to perform goal set synthesis for the cannonball model from Example 2.1 with* $\alpha = 0.7854$*. Algorithm 3 was applied to the described model with* $\rho = \{10^{-5}\}$ *and different values of* η *(*10^{-3} *and* 10^{-6}*, respectively). The obtained results are presented in Figures 3.3 and 3.4.*

It can be seen that the computed under-approximations of the goal set $G = [K_{border}, 0.9]$ *(where* $K_{border} \approx 0.753657747263689$*) and its complement* $G^C = [0.5, K_{border})$ *are correct.*

Also, in the first case ($\rho = 10^{-3}$*) Algorithm 3 returned two undetermined boxes, and in the second case (*$\rho = 10^{-6}$*) their number reduced to one. This is a false alarm: the parameter box* $B_N = [0.753649902, 0.753656006]$ *should be in fact unsatisfiable (*$L[B_N]$ *must be* $[0,0]$*).*

3.6.2 ϵ-guarantee

The size of the returned probability enclosures can be reduced to an arbitrarily small positive ϵ for SPHS with at least one continuous random parameter and without non-

B_N	$L[B_N]$
$[0.500000000, 0.700000000]$	$[0, 0]$
$[0.700000000, 0.750000000]$	$[0, 0]$
$[0.750000000, 0.753125000]$	$[0, 0]$
$[0.753125000, 0.753515625]$	$[0, 0]$
$[0.753515625, 0.753613281]$	$[0, 0]$
$[0.753613281, 0.753637695]$	$[0, 0]$
$[0.753637695, 0.753649902]$	$[0, 0]$
$[\mathbf{0.753649902}, \mathbf{0.753656006}]$	$[\mathbf{0}, \mathbf{1}]$
$[\mathbf{0.753656006}, \mathbf{0.753662109}]$	$[\mathbf{0}, \mathbf{1}]$
$[0.753662109, 0.753710938]$	$[1, 1]$
$[0.753710938, 0.753906250]$	$[1, 1]$
$[0.753906250, 0.754687500]$	$[1, 1]$
$[0.754687500, 0.756250000]$	$[1, 1]$
$[0.756250000, 0.762500000]$	$[1, 1]$
$[0.762500000, 0.775000000]$	$[1, 1]$
$[0.775000000, 0.800000000]$	$[1, 1]$
$[0.800000000, 0.900000000]$	$[1, 1]$

Figure 3.3: Algorithm 3 output for Example 3.6 with $\rho = \{10^{-5}\}$ and $\eta = 10^{-3}$.

B_N	$L[B_N]$
$[0.500000000, 0.700000000]$	$[0, 0]$
$[0.700000000, 0.750000000]$	$[0, 0]$
$[0.750000000, 0.753125000]$	$[0, 0]$
$[0.753125000, 0.753515625]$	$[0, 0]$
$[0.753515625, 0.753613281]$	$[0, 0]$
$[0.753613281, 0.753637695]$	$[0, 0]$
$[0.753637695, 0.753649902]$	$[0, 0]$
$[0.753649902, 0.753656006]$	$[0, 0]$
$[\mathbf{0.753656006}, \mathbf{0.753662109}]$	$[\mathbf{0}, \mathbf{1}]$
$[0.753662109, 0.753710938]$	$[1, 1]$
$[0.753710938, 0.753906250]$	$[1, 1]$
$[0.753906250, 0.754687500]$	$[1, 1]$
$[0.754687500, 0.756250000]$	$[1, 1]$
$[0.756250000, 0.762500000]$	$[1, 1]$
$[0.762500000, 0.775000000]$	$[1, 1]$
$[0.775000000, 0.800000000]$	$[1, 1]$
$[0.800000000, 0.900000000]$	$[1, 1]$

Figure 3.4: Algorithm 3 output for Example 3.6 with $\rho = \{10^{-5}\}$ and $\eta = 10^{-6}$.

deterministic parameters, and in which the formulae verified in procedure **evaluate** are robust. It is required demonstrating the following: the goal set and its complement can be approximated arbitrarily well by parameter boxes; all such boxes can be correctly detected by procedure **evaluate**, and the corresponding definite integrals can be computed with the adequate precision.

The following proposition proves a supporting claim that projection of a closed set from a product space onto one of its components is also closed in it.

Proposition 3.5. (Corollary to the Tube Lemma) *Let Y be compact and $\pi_X : X \times Y \to X$ be the projection of $X \times Y$ on X. Then for any closed $F \subset X \times Y$ the set $\pi_X(F)$ is closed in X.*

Proof. Let $x \notin \pi_X(F)$ (note that in case $\pi_X(F) = X$, $\pi_X(F)$ it is closed in X, and thus, the proposition holds). This implies that $\{x\} \times Y$ is contained in the open subset

$(X \times Y) \setminus F$ (open in $X \times Y$). Then by the *Tube Lemma* [63, Lemma 26.8] there exists an open subset $V_x \subset X$ such that $x \in V_x$ and $V_x \times Y \subset (X \times Y) \setminus F$. The latter means that the open set $V_x \subset X$ lies in the complement of $\pi_X(F)$. As the choice of x was arbitrary then the complement of $\pi_X(F)$ can be obtained as a union of such open sets V_x, which is open (this is because the union of an arbitrary number of open sets is open). Thus, $\pi_X(F)$ is closed in X (as its complement is open in X). □

The following lemma states that in a PHS featuring only continuous parameters any parameter box $B \subseteq P$ with positively-sized edges is either fully inside the goal set G or contains a smaller box $B' \subseteq B$ with positively-sized edges from the goal set complement G^C.

Lemma 3.2. (Goal Set Shape) *Given an invariant-free PHS H featuring only continuous parameters and a reachability depth $l \in \mathbb{N}$ the following holds:*

$$\forall B \subseteq P : (B \subseteq G) \oplus (\exists B' \subseteq B : B' \subseteq G^C),$$

where both B and B' are parameter boxes with positively sized edges, and G is the goal set for depth l.

Proof. In order to prove this lemma it is necessary to show that for any hyper-box $B = [p_1, q_1] \times \cdots \times [p_n, q_n] \subseteq P$ (where $\forall i \in \{1, \ldots, n\} : |q_i - p_i| > 0$) only one the following outcomes is possible:

1) $B \subseteq G$,

2) $\exists B' \subseteq B : B' \subseteq G^C$,

where $B' = [c'_1, d'_1] \times \cdots \times [c'_n, d'_n]$ such that $\forall i \in \{1, \ldots, n\} : |d'_i - c'_i| > 0$.

Clearly, given an arbitrary box $B \subseteq P$ with positively-sized edges one of the following is true:

a) $B \subseteq G$,

b) $B \subseteq G^C$,

c) $(B \cap G \neq \emptyset) \wedge (B \cap G^C \neq \emptyset)$.

Obviously, a) is equivalent to 1), and b) implies 2) for any $B' \subseteq B$. Consider now case c). Let $\hat{G^C} = B \cap G^C \neq \emptyset$ and $\hat{G} = B \cap G \neq \emptyset$. We want to show that $\hat{G^C}$ contains an open ball $Ball(\mathbf{p}, r)$ of radius $r > 0$ centred at some point $\mathbf{p} \in \hat{G^C}$. Suppose that it is not true. Then,

$$\forall \mathbf{p} \in \hat{G^C}, \forall r > 0 : Ball(\mathbf{p}, r) \not\subseteq \hat{G^C}. \tag{3.5}$$

This implies that the interior of $\hat{G^C}$ is empty, making $\hat{G^C}$ a boundary set, which must be closed in B.

Note that **Reach** is a bounded $\mathcal{L}_{\mathbb{R}}$-formula which can be presented in the following form by [34, Lemma 2.1]:

$$\mathbf{Reach}(H, l, B) = \exists^{B}\mathbf{p}, \exists^{[0,T]^{n\cdot m}}\mathbf{t}_{i,j} : \bigvee_{i=0}^{m} \big(\bigwedge_{j=0}^{k_i} \big[f_{i,j}(\mathbf{p}, \mathbf{t}_{i,j}) = 0 \big] \big), \tag{3.6}$$

where each $f_{i,j} : P_R \times [0,T]^{n\cdot m} \to \mathbb{R}$ is a computable (hence continuous) function.

Therefore, set $\hat{G}$ can be written in the form:

$$\hat{G} = \Big\{ \mathbf{p} \in B \mid \exists^{[0,T]^{n\cdot m}}\mathbf{t}_{i,j} : \bigvee_{i=0}^{m} \Big(\bigwedge_{j=0}^{k_i} \big[f_{i,j}(\mathbf{p}, \mathbf{t}_{i,j}) = 0 \big] \Big) \Big\}. \tag{3.7}$$

Let $D_{i,j} = \{\mathbf{p} \in B, \mathbf{t}_{i,j} \in [0,T]^{n\cdot m} \mid f_{i,j}(\mathbf{p}, \mathbf{t}_{i,j}) = 0\}$. As $\{0\}$ is closed in the range of continuous function $f_{i,j}$ then by [73, Corollary to Theorem 4.8] $f_{i,j}^{-1}(0) = D_{i,j}$ is closed in $B \times [0,T]^{n\cdot m}$. As each $D_{i,j}$ is closed in $B \times [0,T]^{n\cdot m}$ then by [73, Theorem 2.24] the sets $\bigcap_{j=0}^{k_i} D_{i,j}$ and $D = \bigcup_{i=0}^{m} \big(\bigcap_{j=0}^{k_i} D_{i,j} \big)$ are closed in $B \times [0,T]^{n\cdot m}$.

Set $\hat{G}$ can be obtained as a projection of set D onto B. Let $\pi_B : B \times [0,T]^{n\cdot m} \to B$ be the projection function. Thus, $\hat{G} = \pi_B(D)$, and as D is closed in $B \times [0,T]^{n\cdot m}$, $[0,T]^{n\cdot m}$ is compact then by Proposition 3.5, $\pi_B(D)$ is closed in B. Hence, $\hat{G^C}$ cannot be closed in B as a complement of a closed set $\hat{G}$ in B. This contradicts the conclusion made above about $\hat{G^C}$ being closed. Therefore, (3.5) is not true, which means that $\hat{G^C}$ must contain an open ball $Ball(\mathbf{p}, r)$ of some positive radius r.

Now it is possible to derive a hypercube $B' = [p_1 - \frac{r}{\sqrt{n}}, p_1 + \frac{r}{\sqrt{n}}] \times \cdots \times [p_n - \frac{r}{\sqrt{n}}, p_n + \frac{r}{\sqrt{n}}] \subset Ball(\mathbf{p}, r)$. Thus, there exists a box $B' \subset G^C$ (as $\hat{G^C} \subseteq G^C$) with positively-sized edges. Therefore, c) implies 2). □

Remark 3.3. *Lemma 3.2 describes the shape of the goal set and its complement, and allows obtaining their under-approximation with an arbitrary positive precision. It entails that neither the goal set (nor its complement) can be dense in the parameter*

space.

However, it is *not guaranteed* that given a box from the system's goal set or its complement there exists a positive $\delta > 0$ for which procedure **evaluate** returns **sat** or **unsat**, respectively. In other words, a false alarm may occur for the given box regardless of how small the value of δ is (**evaluate** returns **undet**). However, if the formulae used in procedure **evaluate** are robust for some positive δ, then it is possible to decide whether the given box belongs to the goal set or its complement.

Definition 3.6. (δ-Robustness [35]) *A bounded $\mathcal{L}_{\mathbb{R}}$-sentence ϕ is called δ-robust if $\phi^{\delta} \Rightarrow \phi$ for a given $\delta > 0$. Also, ϕ is called robust if it is δ-robust for some $\delta > 0$.*

In other words, if a bounded $\mathcal{L}_{\mathbb{R}}$-sentence is robust then it is true or comfortably false. Corollary 24 from [35] states that robustness implies decidability, in the sense that a δ-decision procedure can correctly decide whether a robust bounded $\mathcal{L}_{\mathbb{R}}$-sentence is true or false for some $\delta > 0$.

The following lemma establishes the connection between sets G and G^C, and procedure **evaluate** with the robustness assumption about formulae **Reach** and $\neg$**Reach**$^{\forall}$.

Lemma 3.3. (Adaptive δ) *Given a PHS H, a reachability depth l, a nonempty subset B of the parameter space of H, and assuming that sentences* **Reach**(H, l, B) *and* $\neg$**Reach**$^{\forall}(H, l, B)$ *are robust, the following hold:*

1. $B \subseteq G \Leftrightarrow \exists \delta > 0 : \textbf{evaluate}(H, l, B, \delta) = \textbf{sat}$,
2. $B \subseteq G^C \Leftrightarrow \exists \delta > 0 : \textbf{evaluate}(H, l, B, \delta) = \textbf{unsat}$,

where G is the goal set for depth l.

Proof. Consider 1). If $B \subseteq G$ then by Definition 3.2 formula **Reach**$^{\forall}(H, l, B)$ is true, and equivalently, $\neg$**Reach**$^{\forall}(H, l, B)$ is false. As $\neg$**Reach**$^{\forall}(H, l, B)$ is robust for some $\delta > 0$ then by [35, Corollary 24] a δ-complete decision procedure with precision δ will return *unsat* for $\neg$**Reach**$^{\forall}(H, l, B)$. The latter is equivalent to the **sat** outcome of procedure **evaluate**. The implication $B \subseteq G \Leftarrow \exists \delta : \textbf{evaluate}(H, l, B, \delta) = \textbf{sat}$ holds by the definition of **evaluate** procedure.

Similarly, consider 2). If $B \subseteq G^C$ then by Definition 3.2 formula **Reach**(H, l, B) is false. Again, as **Reach**(H, l, B) is robust for some $\delta > 0$ then by [35, Corollary 24] a

δ-complete decision procedure returns *unsat* for $\mathbf{Reach}(H, l, B)$, which is equivalent to **unsat** returned by **evaluate**. The implication $B \subseteq G^C \Leftarrow \exists\delta : \mathbf{evaluate}(H, l, B, \delta) = \mathbf{unsat}$ holds by the definition of the **evaluate** procedure. □

Lemma 3.3 states that a nonempty parameter box B belongs to the goal set G or its complement G^C if and only if there exists a positive δ for which **evaluate** with robust sentences **Reach** and $\neg\mathbf{Reach}^\forall$ returns **sat** or **unsat** respectively. At the same time if B intersects the border of the goal set G then **evaluate** returns **undet** for all $\delta > 0$ even if both **Reach** and $\neg\mathbf{Reach}^\forall$ are robust for all parameter values in B. Note that Lemma 3.3 also holds for PHSs with invariants.

The following theorem proves that Algorithm 3 provides ϵ-guarantee for SPHSs featuring only continuous random parameters.

Theorem 3.1. (Continuous Random Parameters) *Given an invariant-free stochastic parametric hybrid system $(H, \mathbb{P})$ featuring only continuous random parameters, a reachability depth $l \in \mathbb{N}$, a probability enclosure precision $\epsilon \in (0, 1)$, a constant $\kappa \in (0, \epsilon)$ for bounding the domain of continuous random parameters with unbounded support, and $\eta > 0$ controlling precision of procedure* **evaluate** *with sentences $\mathbf{Reach}(H, l, \{\mathbf{p}\})$ and $\neg\mathbf{Reach}^\forall(H, l, \{\mathbf{p}\})$ robust for all $\mathbf{p} \in P$, Algorithm 3 returns a probability enclosure of size smaller than ϵ.*

Proof. According to Definition 3.3, calculating the bounded reachability probability amounts to calculating the integral $\int_G d\mathbb{P}$, where $G \subseteq P_R$ (remember no nondeterministic parameters are featured in this case) is the goal set as per Definition 3.2 and $\mathbb{P}$ is the probability measure of the random parameters. The termination of the algorithm (with the returned probability enclosure satisfying Definition 3.3) was proven in Proposition 3.2.

As no nondeterministic parameters are present, the bounded reachability probability function is constant. Also, Algorithm 3 introduces a dummy nondeterministic parameter box which is never bisected, and therefore, condition $|B_N| \leq \rho$ in line 18 is never satisfied. Thus, it is necessary to show that the size of the computed probability enclosure can be reduced up to the given ϵ in order to prove termination of Algorithm 3.

Let Π_0 be the partition of P_R containing only one parameter box comprising the entire random parameter space P_R. Now, for any $i \in \mathbb{N}^+$ let Π_i be the partition

obtained by bisecting every hyper-box in Π_{i-1}. Thus, Π_i consists of $2^{n \cdot i}$ boxes (where n is the number of random continuous parameters), and each box in Π_i has volume $\frac{\mu^+(P_R)}{2^{n \cdot i}}$ (where $\mu^+(P_R)$ is the volume of the hyper-box P_R computed as the product of the lengths of its edges).

Let P_R be bounded in such a way that $\int_{P_R} d\mathbb{P} \geq 1 - \kappa$ as shown in Section 3.5.5, and let $\Pi_i = \{\Pi_i^G, \Pi_i^{G^C}, \Pi_i^X\}$ be a partition of P_R such that $\forall B \in \Pi_i^G : B \subseteq G$, $\forall B \in \Pi_i^{G^C} : B \subseteq G^C$ and $\forall B \in \Pi_i^X : (B \cap G \neq \emptyset) \wedge (B \cap G^C \neq \emptyset)$. For a given $i \in \mathbb{N}$ the probability enclosure for the reachability probability is computed as

$$[a, b] = \Big[\sum_{B \in \Pi_i^G} \int_B d\mathbb{P}, 1 - \sum_{B \in \Pi_i^{G^C}} \int_B d\mathbb{P} \Big].$$

Now, in order to provide ϵ-guarantee it is necessary to find a partition Π_i such that:

$$\sum_{B \in (\Pi_i^G \cup \Pi_i^{G^C})} \int_B d\mathbb{P} \geq 1 - \epsilon \tag{3.8}$$

(this way it can be guaranteed that $|[a, b]| \leq \epsilon$).

Lemma 3.2 defines the shape of the goal set and its complement, and it suggests that G and G^C can be under-approximated by boxes from Π_i^G and $\Pi_i^{G^C}$ arbitrarily well. Thus, there exists a finite partition such that (3.8) holds by Remark 3.3. However, in order to show that $[a, b]$ can be computed by Algorithm 3 it is necessary to show that

1. boxes from Π_i^G and $\Pi_i^{G^C}$ can always be correctly identified, and

2. the values of the corresponding integrals are computed with sufficient precision (as choosing a very coarse precision can lead to "discarding too much information" from the interval returned by procedure **measure**).

Point 1) is addressed by Lemma 3.3 which requires both sentences **Reach** and $\neg$**Reach**$^\forall$ to be robust on all boxes in Π_i^G and $\Pi_i^{G^C}$. In other words, given a box B from the goal set or its complement the following must be true:

$$\begin{aligned} \exists \delta > 0 : \big(\mathbf{Reach}^\delta(H, l, B) \rightarrow \mathbf{Reach}(H, l, B)\big) \wedge \\ \big((\neg\mathbf{Reach}^\forall)^\delta(H, l, B) \rightarrow \neg\mathbf{Reach}^\forall(H, l, B)\big) \end{aligned}$$

Suppose that the above does not hold. Hence,

$$\begin{aligned} \forall \delta > 0 : \big(\mathbf{Reach}^\delta(H, l, B) \wedge \neg\mathbf{Reach}(H, l, B)\big) \vee \\ \big((\neg\mathbf{Reach}^\forall)^\delta(H, l, B) \wedge \mathbf{Reach}^\forall(H, l, B)\big) \end{aligned} \tag{3.9}$$

Let $B \in \Pi_i^G$, which means that $\mathbf{Reach}(H, l, B)$ is true. Hence, the left-hand side of the disjunction above evaluates to false, and (3.9) becomes equivalent to $\forall \delta > 0 : \big((\neg\mathbf{Reach}^{\forall})^{\delta}(H, l, B) \wedge \mathbf{Reach}^{\forall}(H, l, B)\big)$, which can be rewritten as $\forall \delta > 0, \forall \mathbf{p} \in B : \big((\neg\mathbf{Reach}^{\forall})^{\delta}(H, l, \{\mathbf{p}\}) \wedge \mathbf{Reach}^{\forall}(H, l, \{\mathbf{p}\})\big)$, or $\forall \mathbf{p} \in B, \forall \delta > 0 : \big((\neg\mathbf{Reach}^{\forall})^{\delta}(H, l, \{\mathbf{p}\}) \wedge \mathbf{Reach}^{\forall}(H, l, \{\mathbf{p}\})\big)$. The latter states that sentence $\neg\mathbf{Reach}^{\forall}$ is not robust for all parameter values in B. However, this contradicts the assumption of the theorem that both sentences must be robust for all parameter values. Therefore, both **Reach** and $\neg\mathbf{Reach}^{\forall}$ are robust on B, and consequently, on every box in Π_i^G. The proof for the boxes from $\Pi_i^{G^C}$ follows the same steps.

Since the precision value $\delta = \eta\frac{\mu^+(P_R)}{2^{n \cdot i}}$ is always positive and decreases as i increases (line 11 in Algorithm 3), then Algorithm 3 can always find such δ for which **Reach** and $\neg\mathbf{Reach}^{\forall}$ are robust on all boxes of partition Π_i satisfying $\big|[a, b]\big| \leq \epsilon$.

Finally, 2) is guaranteed by the precision value passed to procedure **measure** for computing the value of integrals in line 10 of Algorithm 3. From the way the bounds for P_R are chosen it follows that the value of the sum in (3.8) is always bounded by $1 - \kappa$. Therefore, the error of computing the sum of integrals cannot be greater than $\epsilon - \kappa$. Computing each integral with precision $(\epsilon - \kappa)\frac{\mu^+(B)}{\mu^+(P_R)}$ guarantees that the total error when computing the sum of integrals over partition Π_i is equal to $\sum_{B \in (\Pi_i^G \cup \Pi_i^{G^C})} (\epsilon - \kappa)\frac{\mu^+(B)}{\mu^+(P_R)}$ which is smaller than or equal to $\epsilon - \kappa$. □

Remark 3.4. *The robustness assumption must be analogously applied to the formulae in procedure* **compute** *in order to provide ϵ-guarantee in the implementation of Algorithm 3.*

Example 3.7. (Robustness is Important) *This example shows that if* **Reach** *and* $\neg\mathbf{Reach}^{\forall}$ *are not robust for some parameter values then the size of the probability enclosure cannot be refined to an arbitrarily small $\epsilon > 0$.*

Suppose that $\mathbf{Reach}(H, 0, B) := \exists^B p, \exists^{[0,1]} t : -pt > 0$, $\neg\mathbf{Reach}^{\forall}(H, 0, B) := \exists^B p, \forall^{[0,1]} t : pt \geq 0$, *and $p \sim \mathcal{U}(-1, 1)$ is a uniform random parameter. It is easy to see that δ-weakenings of the sentences above ($\exists^B p, \exists^{[0,1]} t : -pt > -\delta$ and $\exists^B p, \forall^{[0,1]} t : pt \geq -\delta$, respectively) are always satisfiable for any $\delta > 0$ if $p \geq 0$. (In other words,* **Reach** *and* $\neg\mathbf{Reach}^{\forall}$ *are not robust for $p \in [0, 1]$.) That means that* **evaluate** *returns* **undet** *for any parameter box in $[0, 1]$ and any $\delta > 0$. At the same time it is possible to find $\delta > 0$ for which* **evaluate** *returns* **sat** *for any box in $[-1, 0)$. This implies that*

Algorithm 3 will always return a probability enclosure $[c, 1]$ *(where* $c < 0.5$*) for any given* $\epsilon > 0$*. Therefore, the size of the computed probability enclosure will never be smaller than 0.5, although all remaining requirements of Theorem 3.1 are met.*

So far it has been proven that Algorithm 3 provides ϵ-guarantee for SPHSs featuring only continuous random parameters. The proposition below demonstrates that introducing discrete random parameters does not affect the size of the computed probability enclosures.

Proposition 3.6. (Introducing Discrete Random Parameters) *With the same assumptions of Theorem 3.1 and given an SPHS* $(H, \mathbb{P})$ *with both continuous and discrete random parameters, Algorithm 3 returns a probability enclosure of size smaller than* ϵ*.*

Proof. Let D be the domain of the discrete random parameters, their probability measure $\mathbb{P}_D = \{f_1, \ldots, f_n\}$ be the product of their probability mass functions, and C – the domain of continuous random parameters and $\mathbb{P}_C$ their probability measure. Thus, $P_R = D \times C$ and $\mathbb{P} = \{\mathbb{P}_D, \mathbb{P}_C\}$.

For every $\mathbf{p} \in D$ a stochastic parametric hybrid system $(H_\mathbf{p}, \mathbb{P}_C)$ can be defined by substituting all discrete random parameters in H with $\mathbf{p}$. Now, each $(H_\mathbf{p}, \mathbb{P}_C)$ is featuring only continuous random parameters.

As all random parameters are independent, the bounded reachability probability for $(H, \mathbb{P})$ can be obtained as $\mathbf{Pr} = \sum_{\mathbf{p}\in D}(\prod_{i=1}^{n} f_i(p_i) \cdot \mathbf{Pr}_\mathbf{p})$ where $\mathbf{Pr}_\mathbf{p}$ is the probability of reaching the goal state by $(H_\mathbf{p}, \mathbb{P}_C)$.

The probability enclosure $[a, b]$ for $(H, \mathbb{P})$ (such that $\mathbf{Pr} \in [a, b]$) can be computed as $[a, b] = \sum_{\mathbf{p}\in D}(\prod_{i=1}^{n} f_i(p_i) \cdot [a, b]_\mathbf{p})$ where each $[a, b]_\mathbf{p}$ is a probability enclosure for $(H_\mathbf{p}, \mathbb{P}_C)$.

By Theorem 3.1, the size of each such $[a, b]_\mathbf{p}$ can be reduced to some arbitrarily small $\epsilon > 0$. Thus, the size of $[a, b]$ can be obtained as:

$$|[a, b]| = \sum_{\mathbf{p}\in D}(\prod_{i=1}^{n} f_i(p_i) \cdot |[a, b]_\mathbf{p}|) \leq \sum_{\mathbf{p}\in D}(\prod_{i=1}^{n} f_i(p_i) \cdot \epsilon) \leq \epsilon \cdot \sum_{\mathbf{p}\in D}\prod_{i=1}^{n} f_i(p_i) \leq \epsilon.$$

(as each f_i is a probability mass function, then $\sum_{\mathbf{p}\in D}\prod_{i=1}^{n} f_i(p_i) = 1$). □

In Theorem 3.1 and Proposition 3.6 it was proven that the bounded reachability probability function for SPHSs featuring at least one continuous random parameter

and procedure **evaluate** with robust formulae is computable. In other words, there is a procedure (Algorithm 3) which given a nondeterministic parameter value and some positive ϵ, computes a value which is at most ϵ-far from the **exact** value of the bounded reachability probability (as any value from the interval returned by Algorithm 3 satisfies this property).

Example 3.8. (ϵ-guarantee for Constant Reachability Probability) *In this example Algorithm 3 was applied to the SCB with $K = 0.7$ and reachability depth $l = 1$ for computing probability enclosure of the desired length ϵ. The algorithm returns enclosures $[0.392681179, 0.393270279]$ for $\epsilon = 10^{-3}$ and $[0.392956194, 0.392963577]$ for $\epsilon = 10^{-5}$, which both contain the reachability probability value $\mathbf{Pr}(0.7) = \sum_{i=1}^{3} \left[f_\alpha(\alpha_i) \cdot \int_{\sqrt{\frac{980}{1.49 \cdot \sin(2\alpha_i)}}}^{\infty} f_{v_0}(x)dx \right] \approx 0.392964374383292$ obtained analytically in Example 3.3.*

Remark 3.5. *Using the reasoning from the proofs of Theorem 3.1 and Proposition 3.6 it is possible to show that Algorithm 3 can compute probability enclosures of size ϵ for a given nondeterministic parameter box B_N if the range of* **Pr** *on B_N is smaller than ϵ and given that* **evaluate** *features robust formulae.*

The above suggests that condition $|B_N| \leq \rho$ in line 18 of Algorithm 3 can be ignored if **Pr** is continuous on P_N. In this case the algorithm will continue partitioning the nondeterministic parameter space and refining the probability enclosures until the size of each enclosure is smaller than or equal to ϵ.

Example 3.9. (ϵ-guarantee for Continuous Reachability Probability) *In Example 3.5 it was concluded that in both cases Algorithm 3 terminated upon reaching the condition $|B_N| \leq \rho$ on each nondeterministic parameter box. However, from Example 3.3 it can be seen that the reachability probability function is continuous. Thus, the condition $|B_N| \leq \rho$ can be ignored. Algorithm 3 was applied with $\epsilon = 10^{-2}$, and the computed probability enclosures are visualised in Figure 3.5. It can be seen that the probability function obtained analytically is fully contained inside the computed probability enclosures (thus, the result is correct), and the size of each such enclosure is smaller than ϵ (thus, ϵ-guarantee is provided).*

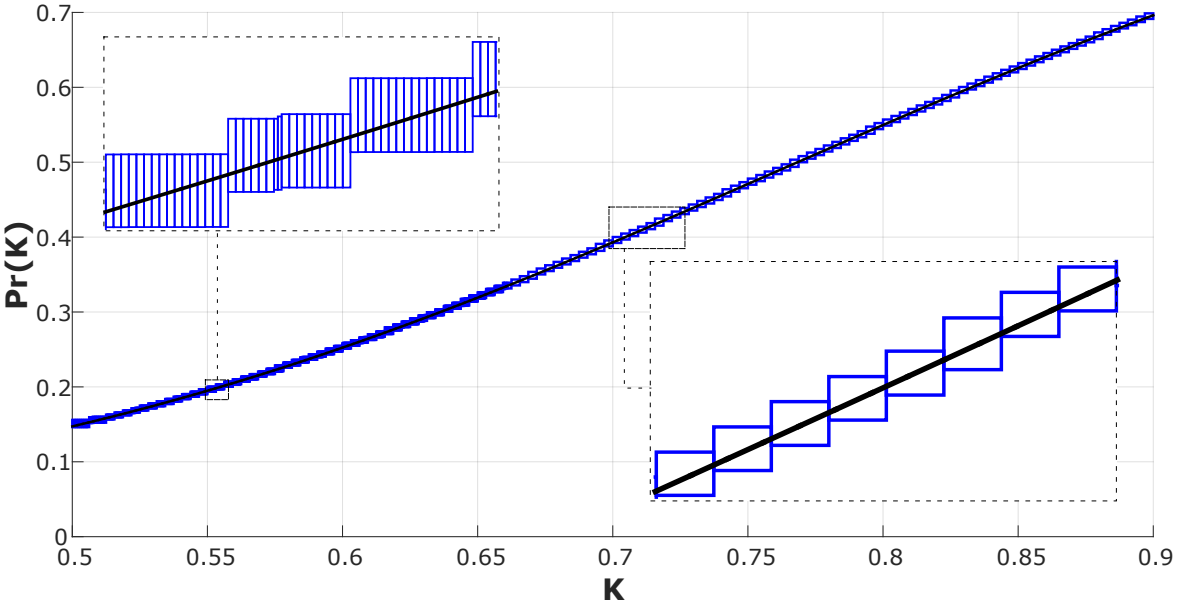


Figure 3.5: Probability enclosures returned by Algorithm 3 for the SCB model, where **blue** boxes – probability enclosures computed for $\epsilon = 10^{-2}$ and **black** line – graph of the reachability probability function obtained analytically (Example 3.3).

3.7 Discussion

In this Chapter I presented an SMT-based technique (Algorithm 3) for computing bounded reachability probability in stochastic parametric hybrid systems. The introduced algorithm computes probability enclosures for the range of the bounded reachability probability function.

In some cases such enclosures can be arbitrarily tight. In particular, for systems featuring at least one continuous random parameter and no nondeterministic parameters (see Proposition 3.6) or when the reachability probability function is continuous and under the assumption that the formulae in procedure **evaluate** are robust.

I also demonstrated that the presented technique can be applied to systems without random parameters. In this case Algorithm 3 performs goal set synthesis (see Proposition 3.4).

All of the above was formally proven and demonstrated on a running example of the stochastic cannonball featuring random (initial speed and angle to the horizon) and nondeterministic (drag coefficient) parameters.

3.7.1 Computational Complexity

It is clear that the search space (the number of evaluated parameter boxes) of Algorithm 3 grows exponentially with the number of system parameters. This is due to the fact that the partitioning procedure **bisect** produces 2^n boxes for each parameter box with n positive edges if the required precision ϵ is not reached. Now each such box is checked by procedure **evaluate**, whose worst case complexity depends exponentially on the number of quantified variables - positively sized edges of the verified box and l time variables. Thus, the overall computational complexity of Algorithm 3 grows exponentially with the number of system parameters and the reachability depth.

Chapter 4 presents an approach for computing bounded reachability probability using statistical model checking techniques. The approach is based on random sampling, which is not negatively influenced by the number of system parameters. Moreover, it takes advantage of the fact that **evaluate** depends only on the reachability depth l as the evaluated parameter boxes are singletons. Unfortunately, the guarantees provided by this approach are statistical, while Algorithm 3 provides absolute numerical guarantees.

3.7.2 Future Work

Algorithm 3 and all auxiliary routines it incorporates could be improved. Firstly, when a parameter box does not fully lie in the goal set or its complement, it is bisected – each positively sized edge of a parameter box is split in halves (Algorithm 4). Thus, deriving an optimal partitioning technique based on the system's behaviour can significantly speed up convergence of the size of probability enclosures to the desired one. Namely, Algorithm 4 can benefit from *sensitivity analysis* – finding how much each parameter contributes to changes in the system's dynamics.

Also, alterations between partitioning the continuous random and continuous nondeterministic parameter boxes are important, as unnecessary partitioning affects the performance of Algorithm 3. This happens when a nondeterministic parameter box is bisected but the probability enclosure is not improved. For example, some unnecessary partitioning can be found in Example 3.9, where the nondeterministic precision is ignored (see Figure 3.5). At the same time the algorithm can suffer from insufficient partitioning of the domain of random parameters which can result in returning

excessively large probability enclosures. Both problems can be partially tackled by pre-partitioning the parameter space prior to the algorithm execution. In other words, Algorithm 4 can be recursively applied to the parameters space until the size of every box in the obtained partition is smaller than or equal to ρ (in case of nondeterministic parameter boxes) or until the corresponding probability value (in case of random parameter boxes) is smaller than or equal to some fraction of ϵ.

Furthermore, the performance of the solver implementing procedure **evaluate** often depends on the size of the evaluated box. Thus, decreasing the size of the boxes can potentially speed up Algorithm 3 (this is another advantage of statistical model checking which operates on singletons) but this will increase the number of parameter boxes. Thus, finding a compromise between the number of parameter boxes and their size is crucial. A further improvement could be an introduction of an adaptive procedure that would not partition a nondeterministic box until no considerable progress can be made on the probability enclosure.

Finally, analysing the system's dynamics and choosing an appropriate precision δ as a result, can be useful for avoiding false alarms.

Chapter 4

Bounded Reachability Probability via Monte Carlo

4.1 Introduction

This chapter presents techniques combining statistical (Monte Carlo) methods and numerically sound procedures to estimate the bounded reachability probability in SPHSs [78].

A statistical approach to probabilistic reachability is important because it trades correctness guarantees with efficiency, and so can scale much better with system size than other approaches. For example, statistical model checking [92] can be faster than probabilistic model checking, which is based on exhaustive state space search [91].

The algorithms presented in this chapter compute statistically guaranteed probability enclosures, while Algorithm 3 (Chapter 3) provides absolute numerical guarantees. However, the number of samples evaluated by the statistical methods does not grow with the number of system parameters, unlike Algorithm 3 featuring exponential growth of the search space.

Monte Carlo techniques for probability estimation assume that one can sample the random variable representing the true system behaviour. However, this is impossible since bounded reachability is undecidable. A novel aspect of the techniques presented in this chapter is that they explicitly take into account undecidability and numerical precision by employing procedure **evaluate** (see Chapter 2) that allows bounding the

value of the desired random variable.

In this chapter I define the bounded reachability probability in terms of Bernoulli random variables and introduce two algorithms for computing confidence intervals for the bounded reachability probability in SPHSs featuring only random parameters. Then I present an algorithm for computing an approximation of the maximum/minimum bounded reachability probability in systems featuring all types of parameters.

4.2 Computing Confidence Intervals

Bounded reachability in SPHS $(H, \mathbb{P})$ can be defined as a Bernoulli random variable

$$X(\mathbf{p}_N, \mathbf{p}_R) = \begin{cases} 1 & \text{if system } H \text{ reaches the goal in } l \text{ steps for } \mathbf{p}_N \in P_N, \mathbf{p}_R \in P_R, \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

Thus, the bounded reachability probability is equal to the expected value of X, *i.e.*, $\mathbf{Pr}(\mathbf{p}_N) = \mathbb{E}[X(\mathbf{p}_N)]$. However, samples of variable X cannot be directly evaluated due to the undecidability of bounded reachability in hybrid systems.

The solution proposed in this section introduces two Bernoulli random variables X_{sat} and X_{usat} that can be used for bounding the range of X, and whose values can be computed. For any given $\delta > 0$, $\mathbf{p}_N \in P_N$ and $\mathbf{p}_R \in P_R$, X_{sat} and X_{usat} are defined as the following:

$$X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta) = \begin{cases} 1 & \text{if } \mathbf{evaluate}(H, l, \{\mathbf{p}_N, \mathbf{p}_R\}, \delta) = \mathbf{sat}, \\ 0 & \text{otherwise,} \end{cases}$$

$$X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta) = \begin{cases} 0 & \text{if } \mathbf{evaluate}(H, l, \{\mathbf{p}_N, \mathbf{p}_R\}, \delta) = \mathbf{unsat}, \\ 1 & \text{otherwise.} \end{cases}$$

Therefore, $X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ is 1 if it is possible to conclude that H reaches the goal state for the given $\mathbf{p}_N$ and $\mathbf{p}_R$, while $X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ is 0 if it can be decided that H does *not* reach the goal state for these $\mathbf{p}_N$ and $\mathbf{p}_R$. If no decision can be made (because of the precision δ being used or of the nature of the reachability question), $X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ and $X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta)$ take 0 and 1, respectively. Thus, by definition of

procedure **evaluate** the following holds for any $\delta > 0$:

$$X_{sat}(\mathbf{p}_N, \mathbf{p}_R, \delta) \leq X(\mathbf{p}_N, \mathbf{p}_R) \leq X_{usat}(\mathbf{p}_N, \mathbf{p}_R, \delta) \ . \tag{4.2}$$

For n random variables iid (independent and identically distributed) as X_{sat} and X_{usat}, the following two random variables are defined:

$$\hat{S}_n = \frac{\Sigma_{i=1}^{n} X_{sat}(\mathbf{p}_N, \mathbf{p}_i, \delta)}{n}, \qquad \hat{U}_n = \frac{\Sigma_{i=1}^{n} X_{usat}(\mathbf{p}_N, \mathbf{p}_i, \delta)}{n}, \tag{4.3}$$

where $\mathbf{p}_i \in P_R$ is the i-th random sample.

The estimates (4.3) can be used for producing confidence intervals for the bounded reachability probability. The remainder of this section proposes two such techniques: the Chernoff-Hoeffding bound algorithm and the Bayesian estimation algorithm. Also, in this section it is assumed that considered SPHSs feature only random parameters, *i.e.*, random variable X is a function of the random parameters only.

4.2.1 Chernoff-Hoeffding Bound Algorithm

The Chernoff-Hoeffding bound algorithm (see Algorithm 7) produces an interval that contains the probability **Pr** with a desired confidence (coverage probability). Given a stochastic parametric hybrid system $(H, \mathbb{P})$, a reachability depth l, a positive δ, an accuracy value $\xi > 0$ and a confidence $c \in (0, 1)$, the algorithm utilises the Chernoff-Hoeffding inequality [43] to compute an interval of length *not smaller* than 2ξ containing the value of the reachability probability **Pr** with confidence c. An upper bound on the interval length cannot be achieved in general, as for example, the formulae in **evaluate** might not be robust for the given δ.

The following proposition can be used for obtaining the sample size guaranteeing the confidence c for the returned interval. It also shows that the sample size grows quadratically as the the value of ξ decreases, and logarithmically with respect to the increasing confidence value c.

Proposition 4.1. *Let* $\mathbf{Pr} = \mathbb{E}[X]$*, where* X *is defined by (4.1). For* $\xi > 0$ *and* n *samples of* X_{sat} *and* X_{usat}*, the following holds:*

$$P(\mathbf{Pr} \in [\hat{S}_n - \xi, \hat{U}_n + \xi]) \geq 1 - 2e^{-2n\xi^2},$$

where $\hat{S}_n$ *and* $\hat{U}_n$ *are defined by (4.3).*

Proof. Applying the Chernoff-Hoeffding inequality to the X_{sat} and X_{usat} samples and

denoting $\mathbf{Pr}_l = \mathbb{E}[X_{sat}]$ and $\mathbf{Pr}_u = \mathbb{E}[X_{usat}]$:

$$P(\mathbf{Pr}_l - \hat{S}_n \geq \xi) = P(\mathbf{Pr}_l \leq \hat{S}_n - \xi) \leq e^{-2n\xi^2},$$
$$P(\mathbf{Pr}_u - \hat{U}_n \geq \xi) = P(\mathbf{Pr}_u \geq \hat{U}_n + \xi) \leq e^{-2n\xi^2}.$$

By (4.2) it can be concluded that $\mathbf{Pr}_l \leq \mathbf{Pr} \leq \mathbf{Pr}_u$, and therefore,

$$P(\mathbf{Pr} \leq \hat{S}_n - \xi) \leq P(\mathbf{Pr}_l \leq \hat{S}_n - \xi) \leq e^{-2n\xi^2},$$
$$P(\mathbf{Pr} \geq \hat{U}_n + \xi) \leq P(\mathbf{Pr}_u \geq \hat{U}_n + \xi) \leq e^{-2n\xi^2}.$$

Finally,

$$P(\mathbf{Pr} \notin [\hat{S}_n - \xi, \hat{U}_n + \xi]) = P(\mathbf{Pr} \leq \hat{S}_n - \xi) + P(\mathbf{Pr} \geq \hat{U}_n + \xi) \leq 2e^{-2n\xi^2},$$

and therefore,

$$P(\mathbf{Pr} \in [\hat{S}_n - \xi, \hat{U}_n + \xi]) = 1 - P(\mathbf{Pr} \notin [\hat{S}_n - \xi, \hat{U}_n + \xi]) \geq 1 - 2e^{-2n\xi^2}.$$

□

The Chernoff-Hoeffding bound algorithm starts by calculating the sample size (Algorithm 7, line 1) guaranteeing the confidence c by inverting the inequality of Proposition 4.1, and initialising the values of the counters s and u (line 2). Then the algorithm enters the loop (line 3) where it draws random samples according to the parameter distributions $\mathbb{P}$ (line 4). Note that differently from [42], it is impossible to sample X, and hence, X_{sat} and X_{usat} are sampled via δ-complete simulation instead. The drawn sample is evaluated using the procedure **evaluate** (line 5), and the counter s is incremented if **evaluate** returns **sat** (line 6) or the counter u is decremented if **unsat** is returned (line 7). Algorithm 7 returns an interval with the left bound obtained as $\frac{s}{n} - \xi$ if it is positive and 0 otherwise, and the right bound calculated as the minimum between $\frac{u}{n} + \xi$ and 1 (line 8). Note that it can happen that the length of the returned interval is greater than 2ξ. This means that for some samples the reachability question could not be correctly decided, because of non-robustness issues or insufficient precision.

Example 4.1. (Chernoff-Hoeffding Bound for SCB) *Consider Example 3.8 from the previous chapter, where the presented stochastic parametric hybrid system does not feature nondeterministic parameters (as $K = 0.7$). In this experiment Algorithm 7 was applied with different settings, and the obtained results are presented in Table 4.1.*

It can be seen that the returned confidence intervals are correct as they contain the exact probability value $\mathbf{Pr}(0.7) \approx 0.392964374383292$ *(see Example 3.3). The size of*

Algorithm 7: chernoff(H, l, δ, ξ, c)

Input : $(H, \mathbb{P})$: SPHS,
$l \in \mathbb{N}$: reachability depth,
$\delta > 0$: solver precision,
$c \in (0, 1)$: confidence (coverage probability),
$\xi \in (0, 1)$: accuracy.

Output: confidence interval with coverage not smaller than c

```
n := ⌈(1/(2ξ²)) log(2/(1−c))⌉;
s := 0; u := n;
for i = 1 : n do
    p := sample(P_R, ℙ);
    switch (evaluate(H, l, {p}, δ)) do
        case sat do  s := s + 1 ;
        case unsat do  u := u − 1 ;
return [max(0, s/n − ξ), min(1, u/n + ξ)];
```

δ	2ξ	c	CI	$\lvert CI\rvert$	n
10^{-3}	10^{-2}	0.99	$[0.389241, 0.399306]$	1.0065×10^{-2}	119,780
10^{-6}	10^{-2}	0.99	$[0.387235, 0.397235]$	10^{-2}	119,780
10^{-6}	5×10^{-3}	0.99	$[0.389857, 0.394857]$	5×10^{-3}	479,117
10^{-6}	5×10^{-3}	0.999	$[0.390762, 0.395762]$	5×10^{-3}	663,504

Table 4.1: Application of Algorithm 7 to SCB model with reachability depth $l = 1$, where δ - precision for the δ-complete decision procedure, 2ξ - desired size of the confidence interval, c - confidence value, CI - confidence interval, $|CI|$ - size of the obtained confidence interval, n - number of verified samples.

the confidence interval for $\delta = 10^{-3}$ *is greater then the desired* 2ξ*. This suggests that formulae* **Reach** *and* $\neg$**Reach**$^{\forall}$ *in the procedure* **evaluate** *were not robust for several samples with the given* δ*. Reducing the value of* δ *to* 10^{-6} *solves this problem. This, however, does not mean that* **Reach** *and* $\neg$**Reach**$^{\forall}$ *are robust for all parameter values in the system's parameter space with this* δ*.*

The presented results also demonstrate quadratic growth of the sample size with respect to the decreasing accuracy: halving the value of ξ *results into a four-fold increase of the number of drawn samples. Increasing the target confidence value c also increases the sample size.*

The technique presented in Algorithm 7 allows bounding the sample size to guarantee the desired confidence. However, there are methods requiring fewer samples, in general, to provide results with similar accuracy and confidence. One such approach based on Bayesian estimation is discussed in the following section.

4.2.2 Bayesian Sequential Estimation

The Bayesian approach assumes that the (unknown) reachability probability $\mathbf{Pr}$ is itself a random quantity [95]. Bayes' theorem enables computing the *posterior* distribution of the unknown quantity given its *prior* distribution and the likelihood of the data (*i.e.*, samples of X). The posterior distribution of $\mathbf{Pr}$ can be directly used to build Bayesian confidence (credibility) intervals. As the random variable X cannot be sampled directly, the technique presented below bounds the posterior of $\mathbf{Pr}$ by the posteriors built from X_{sat} and X_{usat}.

The introduced approach uses Beta distribution priors since they are conjugate to the Bernoulli likelihood; the cumulative distribution function (CDF) of a Beta with parameters $\alpha, \beta > 0$ is denoted $F_{(\alpha,\beta)}(\cdot)$. The following lemma establishes some technical properties of the Beta CDF.

Lemma 4.1. *For any $n > 0$, $s \leq x \leq u \leq n$, $\alpha, \beta > 0$ ($n, s, x, u \in \mathbb{N}$), $t \in [0, 1]$ the following holds:*

$$F_{(u+\alpha,n-u+\beta)}(t) \leq F_{(x+\alpha,n-x+\beta)}(t) \leq F_{(s+\alpha,n-s+\beta)}(t). \tag{4.4}$$

Proof. Consider the LHS inequality of (4.4). When $s = x$ the inequality holds trivially. Consider the case $s < x$. By definition of the Beta distribution function:

$$F_{(s+\alpha,n-s+\beta)}(t) = \int_0^t \frac{v^{s+\alpha-1}(1-v)^{n-s+\beta-1}}{B(s+\alpha, n-s+\beta)} dv. \tag{4.5}$$

The following are the formulae 8.17.1, 8.17.2 and 8.17.18 from [65], respectively:

$$B_y(a, b) = \int_0^y t^{a-1}(1-t)^{b-1}\, dt, \tag{4.6}$$

$$I_y(a, b) = \frac{B_y(a, b)}{B(a, b)}, \tag{4.7}$$

$$I_y(a+1, b-1) = I_y(a, b) - \frac{y^a(1-y)^{b-1}}{aB(a, b)}. \tag{4.8}$$

By (4.6) and (4.7) the Beta distribution function (4.5) can be presented as an *incomplete* Beta function $I_t(s+\alpha, n-s+\beta)$ (the Beta distribution functions for the variables x and u can be written in the same form). Now it will be shown by induction that the following holds:

$$I_t(s+\alpha, n-s+\beta) \geq I_t(x+\alpha, n-x+\beta). \tag{4.9}$$

As $s < x$, $s, x \in \mathbb{N}$ and $s, x > 0$ the base case is $s = 0$ and $x = 1$. Thus, it should be proven that $I_t(\alpha, n+\beta) \geq I_t(\alpha+1, (n+\beta)-1)$. By (4.8):

$$I_t((\alpha)+1, (n+\beta)-1) = I_t(\alpha, n+\beta) - \frac{t^\alpha (1-t)^{n+\beta-1}}{\alpha B(\alpha, n+\beta)}.$$

It is easy to see that $\frac{t^\alpha (1-t)^{n+\beta-1}}{\alpha B(\alpha, n+\beta)} \geq 0$, and therefore, the base case holds.

Suppose now that $x = s+1$. By the same formula (4.8):

$$I_t((s+\alpha)+1, (n-s+\beta)-1) = I_t(s+\alpha, n-s+\beta) - \frac{t^{s+\alpha}(1-t)^{n-s+\beta-1}}{(s+\alpha)B(s+\alpha, n-s+\beta)}.$$

As $\frac{t^{s+\alpha}(1-t)^{n-s+\beta-1}}{(s+\alpha)B(s+\alpha, n-s+\beta)} \geq 0$ the induction step holds as well. Hence, for any $s \leq x$ and $s, x > 0$ the LHS inequality of (4.4) holds, and the proof is complete. The proof of the RHS follows similar steps. □

Proposition 4.2 below states how to bound the posterior distribution of the unknown probability $\mathbf{Pr}$, by using the posteriors built from X_{sat} and X_{usat}. Given n samples of X_{sat}, X_{usat} and a Beta prior with parameters $\alpha, \beta > 0$ it is easy to show [71] that the posterior means are:

$$p_{sat} = \frac{s+\alpha}{n+\alpha+\beta}, \qquad p_{usat} = \frac{u+\alpha}{n+\alpha+\beta}, \tag{4.10}$$

where $s = \sum_{i=1}^{n} X_{sat}(\mathbf{p}_i, \delta)$ and $u = \sum_{i=1}^{n} X_{usat}(\mathbf{p}_i, \delta)$, $\mathbf{p}_i \in P_R$ is the i-th random sample and $\delta > 0$ is the precision of the **evaluate** procedure. Note that this section considers SPHSs without nondeterministic parameters, and thus, variables X_{sat} and X_{usat} depend only on the random parameters and precision δ. For clarity the dependence on δ is omitted.

Proposition 4.2. *Given $\xi > 0$, the posterior probability with respect to n samples of*

X of the interval $[p_{sat} - \xi, p_{usat} + \xi]$ *is bounded below as follows*

$$P(\mathbf{Pr} \in [p_{sat} - \xi, p_{usat} + \xi] | X_1, \ldots, X_n) \geq$$
$$F_{(u+\alpha, n-u+\beta)}(p_{usat} + \xi) - F_{(s+\alpha, n-s+\beta)}(p_{sat} - \xi),$$

where $X_1, \ldots, X_n$ *are iid as* X*, and* p_{sat} *and* p_{usat} *are the posterior means (4.10).*

Proof. By definition of posterior CDF and Lemma 4.1:

$$P(\mathbf{Pr} \leq p_{sat} - \xi | X_1, \ldots, X_n) \leq F_{(s+\alpha, n-s+\beta)}(p_{sat} - \xi),$$
$$P(\mathbf{Pr} \geq p_{usat} + \xi | X_1, \ldots, X_n) \leq 1 - F_{(u+\alpha, n-u+\beta)}(p_{usat} + \xi),$$

and therefore,

$$P(\mathbf{Pr} \in [p_{sat} - \xi, p_{usat} + \xi] | X_1, \ldots, X_n) =$$
$$1 - P(\mathbf{Pr} \leq p_{sat} - \xi | X_1, \ldots, X_n) - P(\mathbf{Pr} \geq p_{usat} + \xi | X_1, \ldots, X_n) \geq$$
$$1 - F_{(s+\alpha, n-s+\beta)}(p_{sat} - \xi) - 1 + F_{(u+\alpha, n-u+\beta)}(p_{usat} + \xi) =$$
$$F_{(u+\alpha, n-u+\beta)}(p_{usat} + \xi) - F_{(s+\alpha, n-s+\beta)}(p_{sat} - \xi).$$

□

The described technique is shown in Algorithm 8. It solves the same problem as Algorithm 7 but the sample size is not known in advance. Another feature of this algorithm is that it utilises different number of samples depending on the (unknown) probability value **Pr**. Namely, it requires significantly fewer samples for computing confidence intervals in cases when the probability value is close to 0 or 1. Thus, the difference between the algorithms is particularly vivid in estimating rare event or large probabilities.

Along with the same input parameters as Algorithm 7, Algorithm 8 also requires parameters for the Beta distribution α and β. The algorithm starts by initializing the counters s, u and v and the sample size n (line 1). The counter v is introduced because the sample size n changes dynamically (line 4).

The samples are drawn inside the loop (line 3) and evaluated using the procedure **evaluate** (line 5). Counters s and v are incremented when **sat** (line 6) and **unsat** (line 7) are returned, respectively, and the **undet** samples are ignored. The values of s and newly calculated u (line 8) are utilised for updating the posterior means (line 9), and current confidence value $\hat{c}$ (line 11) is obtained using the adjusted values of the posterior means (line 10). Algorithm 8 exits the loop and returns the computed

confidence interval when the confidence value $\hat{c}$ reaches (or exceeds) the desired value c (line 12). The following proposition demonstrates the probabilistic termination of the presented algorithm.

Proposition 4.3. *Algorithm 8 terminates almost surely.*

Proof. Recall that Algorithm 8 generates two sequences of random variables $\{X_{sat,n}\}_{n\in\mathbb{N}}$ and $\{X_{usat,n}\}_{n\in\mathbb{N}}$. From [95, Theorem 1] it follows that $X_{sat,n}$ ($X_{usat,n}$) converges a.s., for $n \to \infty$, to the *constant* random variable $\mathbb{E}[X_{sat}]$ ($\mathbb{E}[X_{usat}]$). In particular, the posterior probability of any open interval containing the posterior mean (4.10) must converge to 1. Therefore, the posterior probability of any interval not including the posterior mean must converge to 0.

Now, the interval $(0, p_{usat} + \xi)$ contains the posterior mean p_{usat} of $X_{usat,n}$ and therefore the posterior probability $F_{(u+\alpha,n-u+\beta)}(p_{usat} + \xi)$ converges to 1. Also, the interval $(0, p_{sat} - \xi)$ does not contain the mean (p_{sat}) of $X_{sat,n}$, so $F_{(s+\alpha,n-s+\beta)}(p_{sat} - \xi)$ tends to 0, and this concludes the proof. □

Example 4.2. (Bayesian Estimation for SCB) *In this example Algorithm 8 was applied to SCB model from Example 3.8. The obtained results with the different input settings and the values of K are featured in Table 4.2.*

It can be seen that the computed confidence intervals are correct as they contain the corresponding probability values $\mathbf{Pr}(0.7) \approx 0.3929643743$, $\mathbf{Pr}(0.5) \approx 0.14728404068$ *and* $\mathbf{Pr}(0.9) \approx 0.6961960101$ *(these three values were obtained with MATLAB using the bounded reachability probability function from Example 3.3).*

Similarly to Example 4.1, the precision $\delta = 10^{-3}$ was not enough for obtaining the confidence intervals of the desired length. Differently from Algorithm 7, Algorithm 8 requires almost half as many samples for computing confidence intervals of the same length and with the same confidence.

The solver precision δ had to be reduced from 10^{-6} to 10^{-12} for $K = 0.5$ in order to obtain the confidence interval of the desired length 2ξ. It also can be seen that the sample size varies depending on the reachability probability value. Namely, the probability values closer to 0.5 required more samples than the ones closer to 0 or 1 (412,977 vs. 215,742 and 366,947, respectively).

Finally, the presented results demonstrate a similar sample size growth with respect to the given accuracy and confidence values as Algorithm 7.

Algorithm 8: bayes$(H, l, \delta, \xi, c, \alpha, \beta)$

Input : $(H, \mathbb{P})$: SPHS,
$l \in \mathbb{N}$: reachability depth,
$\delta > 0$: solver precision,
$c \in (0, 1)$: confidence (coverage probability),
$\xi \in (0, 1)$: accuracy,
α, β: Beta distribution parameters.

Output: confidence interval with posterior probability not smaller than c.

$n = 0$; $s = 0$; $u = 0$; $v = 0$;
repeat
  $\mathbf{p} =$ **sample**$(P_R, \mathbb{P})$;
  $n = n + 1$;
  switch evaluate$(H, l, \{\mathbf{p}\}, \delta)$ **do**
    case sat do $s = s + 1$;
    case unsat do $v = v + 1$;
  $u = n - v$;
  $p_{sat} = \frac{s+\alpha}{n+\alpha+\beta}$; $p_{usat} = \frac{u+\alpha}{n+\alpha+\beta}$;
  $p_{sat} = max(\xi, p_{sat})$; $p_{usat} = min(1 - \xi, p_{usat})$;
  $\hat{c} = F_{(u+\alpha, n-u+\beta)}(p_{usat} + \xi) - F_{(s+\alpha, n-s+\beta)}(p_{sat} - \xi)$;
until $\hat{c} \geq c$;
return $[p_{sat} - \xi, p_{usat} + \xi]$;

K	δ	2ξ	c	CI	$\lvert CI\rvert$	n
0.7	10^{-3}	10^{-2}	0.99	$[0.384350, 0.394413]$	1.0063×10^{-2}	63,098
0.7	10^{-6}	10^{-2}	0.99	$[0.388787, 0.398787]$	10^{-2}	63,352
0.7	10^{-6}	5×10^{-3}	0.99	$[0.391046, 0.396046]$	5×10^{-3}	253,363
0.7	10^{-6}	5×10^{-3}	0.999	$[0.389728, 0.394728]$	5×10^{-3}	412,977
0.9	10^{-6}	5×10^{-3}	0.999	$[0.692939, 0.697939]$	5×10^{-3}	366,947
0.5	10^{-6}	5×10^{-3}	0.999	$[0.145017, 0.150021]$	5.004×10^{-3}	217,898
0.5	10^{-12}	5×10^{-3}	0.999	$[0.143237, 0.148237]$	5×10^{-3}	215,742

Table 4.2: Application of Algorithm 8 to SCB model with reachability depth $l = 1$, where K - drag coefficient value in SCB model, δ - precision for the δ-complete decision procedure, 2ξ - desired size of the confidence interval, c - confidence value, CI - confidence interval, $|CI|$ - size of the obtained confidence interval, n - number of verified samples.

Remark 4.1. *Note that if procedure* **evaluate** *features formulae which are robust for the given δ and all parameter values, the size of the confidence interval returned by Algorithms 7 and 8 can be guaranteed to be of size not larger than 2ξ. This is because both algorithms use procedure* **evaluate** *to verify the singletons drawn from the parameter space. Each such singleton belongs either to the goal set or its complement. Therefore, if both formulae in procedure* **evaluate** *are robust for the given δ, then* **evaluate** *returns either* **sat** *or* **unsat**.

4.3 Handling Nondeterminism

The introduction of nondeterministic parameters defines a bounded reachability function (see Chapter 3). Analogously to Algorithm 3, the nondeterministic parameter space can be partitioned into boxes, and for each such box a confidence interval containing the entire range of the bounded reachability probability function on the corresponding parameter box can be computed using one the algorithms from Section 4.2. However, this does not solve the problem of exponential complexity growth as the nondeterministic parameter boxes will introduce quantified variables into the formulae in procedure **evaluate**. In order to tackle this issue there are methods for solving optimization problem (that is finding the minimum/maximum reachability probability) based on sampling such as Monte Carlo and Quasi-Monte Carlo techniques [11, 56].

This chapter presents an adaptation of the Cross-Entropy (CE) algorithm [72] – a Monte Carlo approach for handling SPHSs featuring nondeterministic parameters, and Quasi-Monte Carlo methods will be considered for future work.

The CE is a powerful stochastic technique that solves *approximately* the problem of finding a value $\mathbf{p}^* \in P_N$ for the nondeterministic parameters that minimises (maximises) the reachability probability function $\mathbf{Pr}$ on P_N. In other words, it returns an estimate $\hat{\mathbf{p}}$ for $\mathbf{p}^*$ and a confidence interval $[a, b]$ containing $\mathbf{Pr}(\hat{\mathbf{p}})$ with specified confidence $c \in (0, 1)$. However, it cannot be guaranteed that $\hat{\mathbf{p}}$ is a global optimum (*i.e.*, $\hat{\mathbf{p}} \neq \mathbf{p}^*$ in general).

4.3.1 Cross-Entropy Algorithm

The probabilistic reachability analysis for SPHS featuring both random and nondeterministic parameters is performed by solving an optimisation problem aimed at finding

the nondeterministic parameter values for which the system achieves the maximum (minimum) bounded reachability probability. Algorithm 9 presents a technique based on the Cross-Entropy method.

The main idea behind the CE method is obtaining the optimal parameter distribution by minimizing the distance between two probability density functions. The cross-entropy (or Kullback-Leibler divergence) between two probability density functions g and f is equal to:

$$\Theta(g, f) = \int g(\mathbf{p}) \ln \frac{g(\mathbf{p})}{f(\mathbf{p})} d\mathbf{p} \ .$$

The CE is non-negative and $\Theta(g, f) = 0$ iff $g = f$, but it is not symmetric (*i.e.*, $\Theta(g, f) \neq \Theta(f, g)$), so it is not a distance in the formal sense.

The optimisation problem solved by the CE method can be formulated as the following: given a family of densities $\{f(\cdot; \mathbf{v})\}_{\mathbf{v} \in V}$ find the value $\mathbf{v} \in V$ that minimizes $\Theta(g^*, f(\cdot; \mathbf{v}))$ (where g^* is the optimal density). Essentially, the CE method performs a randomised search in the nondeterministic parameter space P_N, "guided" by the Kullback-Leibler divergence. The CE comprises two general steps:

1. generating random samples from some initial distribution and computing the confidence intervals for each sample using one of the algorithms from Section 4.2,

2. updating the distribution based on a portion of the best samples (also called *elite*) in order to draw better samples in the next iteration.

Figure 4.1 provides visual aid for explaining the intuition behind the algorithm. It can be seen that the distribution's mean (μ_1, μ_2, μ_3) is moving towards the optimal value μ^* while the distribution's variance is decreasing.

Note that for solving optimisation problems it is necessary that the family $\{f(\cdot; \mathbf{v})\}_{\mathbf{v} \in V}$ contains distributions that can approximate arbitrarily well single-point distributions. Also, the formulae for updating the distribution parameters $\mathbf{v}$ based on the elite samples must satisfy the following stochastic program:

$$\frac{1}{k} \sum_{i=1}^{k} P(\mathbf{p}_i) \frac{d}{d\mathbf{v}} \ln f(\mathbf{p}_i; \mathbf{v}) = 0, \tag{4.11}$$

where k is the number of elite samples, $\mathbf{p}_i$ is the i-th elite sample drawn from P_N, $P(\mathbf{p}_i)$ is the sample performance which is equal to 1 if $\mathbf{p}_i$ is elite, $\mathbf{v}$ is the parameter of

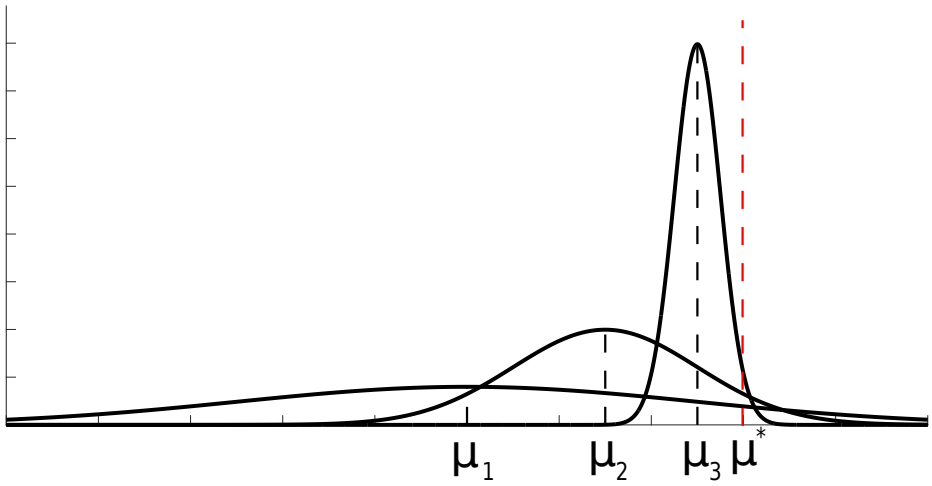


Figure 4.1: Explanation of the principles of the Cross-Entropy algorithm using an example of a family of normal distributions, where μ_1, μ_2 and μ_3 – the means of the PDFs $f(\cdot; \mathbf{v})$ used for at the first, the second and the third iterations of the algorithm and μ^* – the mean of the optimal density g^*.

the distribution for the next iteration of the algorithm, and f is the probability density function of the chosen distribution.

The described technique is presented in Algorithm 9. The algorithm starts by initializing the distribution parameter $\mathbf{v}$ (line 1) with the user-defined value $\mathbf{v}_0$ and calculating the elite sample size $k = \lceil \lambda s \rceil$ (line 2), where $s > 0$ is the chosen sample size and $\lambda \in (0, 1)$ is the fraction of the sample size defining the number of elite samples.

In the outer loop (line 4) the current distribution variance is computed, and the main queue is initialised (line 5). Then in the inner loop (line 6) it draws s random samples from the distribution $f(\cdot; \mathbf{v})$ (line 7) and evaluates performance of each sample $\mathbf{p}_i$ using Algorithm 8 (line 10), where $H(\mathbf{p}_i)$ is a SPHS obtained by substituting all nondeterministic parameters in the given system H with the drawn value $\mathbf{p}_i$. The Chernoff-Hoeffding bound algorithm (Algorithm 7) can also be used for the same purpose. However, as it was discussed in the previous section, it requires more samples for estimating the bounded reachability probability with the same precision.

The evaluated samples and the corresponding confidence intervals are pushed to the main queue (line 11). Note that the distribution $f(\cdot, \mathbf{v})$ can have unbounded support. Therefore, it is necessary to check whether the drawn samples are inside the domain of nondeterministic parameters P_N (line 8) as the system might not be well-defined

Algorithm 9: $\mathbf{CE}(H, l, \delta, c, \xi, \alpha, \beta, s, \lambda, \hat{\sigma}^2, \mathbf{v}_0, f(\cdot;\cdot))$

Input : $(H, \mathbb{P})$: SPHS,
$l \in \mathbb{N}$: reachability depth,
$\delta > 0$: solver precision,
$c \in (0,1)$: confidence (coverage probability),
$\xi \in (0,1)$: accuracy,
α, β: Beta distribution parameters,
s: sample size,
λ: elite samples ratio,
$\hat{\sigma}^2$: maximum variance,
$\mathbf{v}_0$: initial parameters of the distribution,
$f(\cdot;\cdot)$: parametric distribution family,
N: maximum number of iterations.

Output: parameter value, maximum probability

1 $\mathbf{v} = \mathbf{v}_0$;
2 $k = \lceil \lambda s \rceil$;
3 **counter** $:= 0$;
4 **repeat**
5 $\quad \sigma^2 = Var(f(\cdot; \mathbf{v}))$; $Q = \emptyset$;
6 $\quad$ **for** $i = 1 : s$ **do**
7 $\quad\quad \mathbf{p}_i = \mathbf{sample}(f(\cdot; \mathbf{v}))$;
8 $\quad\quad$ **if** $\mathbf{p}_i \notin P_N$ **then** $[a, b] = [-\infty, -\infty]$; ;
9 $\quad\quad$ **else**
10 $\quad\quad\quad [a, b] = \mathbf{bayes}(H(\mathbf{p}_i), l, \delta, c, \xi, \alpha, \beta))$;
11 $\quad\quad Q \leftarrow \{\mathbf{p}_i, \mathrm{mid}([a, b])\}$;
12 $\quad$ **sort**(Q);
13 $\quad E = \{Q[1], \cdots, Q[k]\}$;
14 $\quad res = E[1]$;
15 $\quad \mathbf{v} = \mathbf{update}(E)$;
16 $\quad$ **counter** $:=$ **counter** $+ 1$;
17 **until** $\Big((\max_{1 \le j \le n} \sigma_j^2) \le \hat{\sigma}^2\Big) \vee \Big(\mathbf{counter} > N\Big)$;
18 **return** res;

outside P_N. Also, when solving a probability minimization problem the value assigned to $[a, b]$ in line 8 should be changed to $[\infty, \infty]$.

After all samples are evaluated they are sorted (line 12) by the midpoint of their confidence intervals in descending order (ascending in the case of probability minimiza-

tion), and first k of them are identified as *elite* (line 13). The set of *elite* samples E is then used for updating the distribution parameters $\mathbf{v}$ (line 15).

The formulae for updating the distribution parameters (*i.e.*, procedure **update**) depend on the type of the distribution. The following sections discuss the application of normal and Beta distributions to **update** in Algorithm 9. If the termination condition (line 17) is not met then the algorithm repeats the previous step with the updated distribution parameters.

The algorithm terminates when the largest element of variance vector σ^2 reaches a user-defined precision $\hat{\sigma}^2$, or when the maximum number of iterations is reached, and it outputs the estimated maximum (minimum) confidence interval $[a, b]$ for the bounded reachability probability and the (nondeterministic) parameter value $\mathbf{p}_i$ for which $\mathbf{Pr}(\mathbf{p}_i) \in [a, b]$.

There are several factors affecting the quality of Algorithm 9. First of all, using more samples per iteration and choosing appropriate initial parameters of the utilised distribution increases the coverage of the nondeterministic parameter space and prevents from falling into local extrema. An example of potential converging to a local maximum can be found in Section 6.2.5. Secondly, reducing the terminal variance can provide better "fine-tuning" – obtaining a more accurate estimate. Finally, the accuracy of evaluating the sample performance and the solver precision used for computing confidence intervals for the drawn nondeterministic samples is an equally important factor.

4.3.2 Normal Distribution for CE

This section considers a parametrized family of normal distributions $f(\cdot; \mathbf{v})$ with $\mathbf{v} = \{\mu, \sigma\}$, where the first element of $\mathbf{v}$ is the mean and the second element is the standard deviation. Note that initially the standard deviation should be relatively large in order to cover a larger space on the first iteration of the algorithm. Thus, μ_0 is chosen to be in the centre of P_N, and each element of σ_0 is a half-size of the corresponding parameter domain:

$$\mu_0 = \{\frac{c_1 + d_1}{2}, \cdots, \frac{c_n + d_n}{2}\}, \quad \sigma_0 = \{\frac{d_1 - c_1}{2}, \cdots, \frac{d_n - c_n}{2}\},$$

where $[c_i, d_i]$ is the domain of the i-th nondeterministic system parameter.

The distribution parameters μ_i and σ_i on the i-th iteration of the outer loop of Algorithm 9 are updated using the formulae from [72, Chapter 8.7]:

$$\mu_i = \frac{\sum_{j=1}^{k} E[j]}{k}, \quad \sigma_i = \sqrt{\frac{\sum_{j=1}^{k} (E[j] - \mu_i)^2}{k}},$$

where E is a set of elite samples.

Also, as the normal distribution has unbounded support then, given the desired number of nondeterministic samples s^*, it is easy to see that as the number of nondeterministic parameters increases, the more difficult it becomes to draw samples lying inside of P_N. In fact, given n nondeterministic parameters the probability that a sample $\mathbf{p}_i$ belongs to P_N is equal to:

$$P(\mathbf{p}_i \in P_N) = \prod_{j=1}^{n} \int_{c_j}^{d_j} f(x_j | \mu_j, \sigma_j)\, dx_j \tag{4.12}$$

Hence, in order to increase the likelihood that s^* samples lie in P_N it is sufficient to generate $s = \lceil \frac{s^*}{\eta} \rceil$ samples, where $\eta = P(\mathbf{p}_i \in P_N)$ is obtained using (4.12). Using distributions with bounded support (*e.g.* Beta distribution) allows avoiding this issue.

Example 4.3. (Maximum Reachability Probability in SCB)

Algorithm 9 was applied to computing the maximum reachability probability for SCB model using a parametrised family of normal distributions with reachability depth $l = 1$*, precision* $\delta = 10^{-12}$*, accuracy* $2\xi = 10^{-2}$*, confidence* $c = 0.99$*, and terminal variance* $\hat{\sigma}^2 = 10^{-2}$*.*

λ	s	K^*	CI	$\mathbf{Pr}(K^*)$	s_N	i	s_{out}
10^{-1}	10	0.89301	$[0.68238, 0.69238]$	0.68677	26	2	4
10^{-1}	20	0.88407	$[0.67208, 0.68208]$	0.6745	51	2	10
5×10^{-1}	10	0.8819	$[0.66715, 0.67715]$	0.67148	26	2	5

Table 4.3: Results of applying Algorithm 9 in Example 4.3, where λ - elite ratio, s - number of nondeterministic samples per iteration of Algorithm 9, K^* - nondeterministic parameter estimate resulting into the maximum probability reachability, CI - corresponding confidence interval, $\mathbf{Pr}(K^*)$ - approximate value of the probability function obtained analytically for the given K^*, s_N - total number of nondeterministic samples, i - number of iterations of Algorithm 9, s_{out} - number of nondeterministic samples drawn from outside P_N.

The computed confidence intervals (see Table 4.3) contain the reachability probability values calculated analytically (see Example 3.3) for the corresponding nondeterministic estimates, and the point of maximum is at $K_{max} = 0.9$, where $\mathbf{Pr}(K_{max}) \approx 0.69617$.

At the same time, the obtained results are rather counter-intuitive as increasing the nondeterministic sample size (s) and the elite samples ratio (λ) did not result into better maximum reachability probability estimates.

The sample size correction prevented Algorithm 9 from under-sampling the nondeterministic parameter space, as inequality $s \leq \frac{s_N - s_{out}}{i}$ holds for all three cases. In other words, Algorithm 9 drew more nondeterministic samples per iteration than the desired value s.

4.3.3 Beta Distribution for CE

This section presents a parametrized family of Beta distributions $f(\cdot; \mathbf{v})$ with $\mathbf{v} = \{\alpha, \beta\}$, where $\alpha \geq 1$ and $\beta \geq 1$ are the parameters of a Beta distribution. As there are no analytic formulas for updating α and β in the literature, I have directly derived the updating formulae through solving the following stochastic program (4.11).

$$\frac{1}{k} \sum_{i=1}^{k} P(\mathbf{p}_i) \frac{d}{d\mathbf{v}} \ln f(\mathbf{p}_i; \mathbf{v}) = 0.$$

The probability density function of Beta distribution is a function:

$$f(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)},$$

where $B(\alpha, \beta)$ – Beta function.

The derivative of the logarithm of f with respect to α and β are:

$$\frac{d}{d\alpha} \ln f(\mathbf{p}_i, \alpha, \beta) = \ln \mathbf{p}_i - \frac{d}{d\alpha} B(\alpha, \beta) \frac{1}{B(\alpha, \beta)} = \ln(\mathbf{p}_i) - \psi(\alpha) + \psi(\alpha + \beta),$$

$$\frac{d}{d\beta} \ln f(\mathbf{p}_i, \alpha, \beta) = \ln(1 - \mathbf{p}_i) - \frac{d}{d\beta} B(\alpha, \beta) \frac{1}{B(\alpha, \beta)} = \ln(1 - \mathbf{p}_i) - \psi(\beta) + \psi(\alpha + \beta),$$

where $\psi(\alpha) = \frac{\Gamma'(\alpha)}{\Gamma(\alpha)}$ – digamma function, and $\Gamma(\alpha) = \int_0^\infty z^{\alpha-1} e^{-z} dz$ – gamma function.

Hence, the values α and β satisfying the stochastic program (4.11) are the solution

for the system of equations:

$$\frac{1}{k}\sum_{i=1}^{k} P(\mathbf{p}_i)(\ln(\mathbf{p}_i) - \psi(\alpha) + \psi(\alpha+\beta)) = 0,$$
$$\frac{1}{k}\sum_{i=1}^{k} P(\mathbf{p}_i)(\ln(1-\mathbf{p}_i) - \psi(\beta) + \psi(\alpha+\beta)) = 0.$$

Let now E be a set of *elite* samples. Thus, for each $\mathbf{p}_i$ its performance $P(\mathbf{p}_i) = 1$ (by definition of the elite samples and the sample performance). Hence,

$$\sum_{i=1}^{k}(\ln(E[i]) - \psi(\alpha) + \psi(\alpha+\beta)) = 0,$$
$$\sum_{i=1}^{k}(\ln(1-E[i]) - \psi(\beta) + \psi(\alpha+\beta)) = 0,$$

or equivalently,

$$\psi(\alpha) - \psi(\alpha+\beta) - c_1(E) = 0,$$
$$\psi(\beta) - \psi(\alpha+\beta) - c_2(E) = 0,$$

where $c_1(E) = \frac{\sum_{i=1}^{k}\ln(E[i])}{|E|}$ and $c_2(E) = \frac{\sum_{i=1}^{k}\ln(1-E[i])}{|E|}$.

Function ψ can be approximated as $\psi = \psi^* + O(\frac{1}{\alpha^{16}})$ [48], where $\psi^* = \ln(\alpha) - \frac{1}{2\alpha} - \frac{1}{12\alpha^2} + \frac{1}{120\alpha^4} - \frac{1}{252\alpha^6} + \frac{1}{240\alpha^8} - \frac{5}{660\alpha^{10}} + \frac{691}{32760\alpha^{12}} - \frac{1}{12\alpha^{14}}$. Therefore, the values of α and β for the next iteration of the Cross-Entropy algorithm can be obtained as the solution of the system below.

$$\psi^*(\alpha) - \psi^*(\alpha+\beta) - c_1(E) = 0, \quad \psi^*(\beta) - \psi^*(\alpha+\beta) - c_2(E) = 0.$$

Example 4.4. (Minimum Reachability Probability in SCB)

Algorithm 9 was applied to computing the minimum reachability probability for SCB model using normal and Beta distributions with reachability depth $l = 1$, precision $\delta = 10^{-12}$, accuracy $2\xi = 10^{-2}$, confidence $c = 0.99$, elite sample ratio $\lambda = 10^{-1}$, and nondeterministic sample size $s = 10$.

The computed confidence intervals (see Table 4.4) contain the reachability probability values calculated analytically (see Example 3.3) for the corresponding nondeterministic estimates, and the point of minimum is at $K_{min} = 0.5$, where $\mathbf{Pr}(K_{min}) \approx 0.147284$.

It can also be seen that the Cross-Entropy algorithm with normal distribution drew more samples from the domain of nondeterministic parameters than with Beta distribution (25 against 20 samples) for the same terminal variance value $\hat{\sigma}^2 = 10^{-2}$. However,

$\hat{\sigma}^2$	K^*	CI	$\mathbf{Pr}(K^*)$	s_N	i	s_{out}	$f(\cdot;\cdot)$
10^{-2}	0.52182	$[0.16223, 0.17223]$	0.1668	20	2	0	$\mathcal{B}$
10^{-2}	0.50425	$[0.14464, 0.15464]$	0.15093	26	2	1	$\mathcal{N}$
10^{-6}	0.52766	$[0.17117, 0.18117]$	0.17235	40	4	0	$\mathcal{B}$

Table 4.4: Results of applying Algorithm 9 in Example 4.4, where $\hat{\sigma}^2$ - terminal variance, K^* - nondeterministic parameter estimate resulting into the minimum probability reachability, CI - corresponding confidence interval, $\mathbf{Pr}(K^*)$ - approximate value of the probability function obtained analytically for the given K^*, s_N - total number of nondeterministic samples, i - number of iterations of Algorithm 9, s_{out} - number of nondeterministic samples drawn from outside P_N, $f(\cdot;\cdot)$ - distribution used by Algorithm 9, $\mathcal{N}$ - normal distribution, $\mathcal{B}$ - Beta distribution.

the former variation of Algorithm 9 produced a better estimate of the minimum reachability probability.

Also decreasing terminal variance from 10^{-2} *to* 10^{-6} *increased the number of iterations of Algorithm 9 from 2 to 4, but it did not result into producing better probability estimates.*

Example 4.4 demonstrates that using distributions with bounded support can improve performance of Algorithm 9, as there is no oversampling that can happen due to the sample size correction. At the same time, the normal distribution provided a better probability estimate than the Beta distribution. A possible explanation to this phenomenon could be that distributions with bounded support perform worse when the minimum/maximum reachability probability resides at the border of the nondeterministic parameter space. This way the probability of drawing a sample closer to the border is smaller than for distributions with unbounded support.

4.4 Discussion

In this chapter I introduced novel statistical techniques for computing the bounded reachability probability in SPHSs. The presented algorithms provide numerically and statistically rigorous confidence intervals by combining the numerically guaranteed procedure **evaluate** and Monte Carlo techniques such as the Chernoff-Hoeffding bound, Bayesian estimation, and the Cross-Entropy algorithm. The presented algorithms help

reducing the computational cost with respect to the number of system parameters in comparison to Algorithm 3. Now the number of evaluated parameter boxes remains constant with respect to the number of random parameters. However, the CE algorithm (Algorithm 9) might still require more samples when the number of nondeterministic parameters increases.

The Chernoff-Hoeffding bound and the Bayesian estimation algorithm are used for computing confidence intervals containing the bounded reachability probability value for the systems featuring only random parameters. The latter was demonstrated to be more efficient than the former for obtaining results with similar accuracy and confidence values.

The Cross-Entropy algorithm can handle SPHSs with all types of parameters. It navigates through the nondeterministic parameter space for obtaining an estimate for the minimum (maximum) probability value. Such estimate is not guaranteed to be the value where the *exact* minimum (maximum) probability resides. However, the confidence interval obtained for this value is statistically and numerically guaranteed.

Algorithm 9 can incorporate different distributions from the exponential family (*e.g.* normal and Beta distributions). Distributions with unbounded support require increasing the number of samples in order to guarantee sufficient coverage of the nondeterministic domain, which, however, can result in oversampling. Nevertheless, they provide better sampling around the borders of the nondeterministic parameter space than distributions with bounded support.

4.4.1 Future Work

The future direction of this work is studying statistical methods with better convergence rate (*e.g.*, randomised Quasi-Monte Carlo techniques) for computing confidence intervals, analysing the system's dynamics for obtaining the optimal precision for the δ-complete decision procedure, and incorporating different distributions for every iteration of the Cross-Entropy algorithm depending on the current probability estimate.

Chapter 5

ProbReach: A Software Tool for Computing Bounded Reachability Probability in SPHS

5.1 Introduction

In this chapter I present `ProbReach`, a tool that I developed for computing bounded reachability probability in stochastic parametric hybrid systems [77]. It provides a C++ implementation (about 10,000 lines of code) of the algorithms introduced in the previous chapters. `ProbReach` uses publicly available libraries, and it is distributed under the GNU General Public License[1] (GPL). This chapter discusses the architecture of `ProbReach`, its implementation details, and presents several usage scenarios.

5.2 Input format

`ProbReach` uses the **P**robabilistic **D**elta-**R**eac**H**ability (PDRH) format for PHSs and SPHSs encoding. PDRH extends **D**elta-**R**eac**H**ability (DRH) format utilised by `dReach` [51] with random parameters. Figure 5.1 shows the PDRH encoding of SCB model from Example 3.1. The full description of the PDRH format can be found in the `ProbReach`

[1] `http://www.gnu.org/licenses/gpl.html`

documentation[1].

```
#define g 9.8
[0, 10000] Sx; // horizontal distance (m)
[0, 1000] Sy; // vertical distance (m)
[0, 10] tau; // local time
[0, 10] time; // local time (required)
[0, 500] v; // speed of the ball (m/s)
[0.5, 0.9] K; // drag coefficient
dist_normal(25,3) v0; // initial speed of the ball (m)
dist_discrete(0.7854:0.9,
              1.0472:0.09,
              0.5236:0.01) alpha; // angle to horizon (rad)
{
mode 1;
flow:
    d/dt[Sx]    = v * cos(alpha);
    d/dt[Sy]    = v * sin(alpha) - g * tau;
    d/dt[tau]   = 1.0;
    d/dt[v] = 0.0;
jump:
    (and (tau > 1e-3) (Sy <= 0) (Sy >= 0)) ==> @1(and  (Sx' = Sx)
                                                (Sy' = 0)
                                                (tau' = 0)
                                                (v' = K * v)
                                                (v0' = v0)
                                                (alpha' = alpha)
                                                (K' = K));
}
init:
@1(and (Sx = 0) (Sy = 0) (tau = 0) (v = v0));
goal:
@1(and (tau = 0) (Sx >= 100));
```

Figure 5.1: SCB model encoded in PDRH format.

[1] https://github.com/dreal/probreach/blob/master/doc/usage.md

5.3 ProbReach Architecture

`ProbReach` consists of several components (Figure 5.2): *PDRH Parser*, *Evaluation Procedure*, *Utility Package* and *Algorithms*.

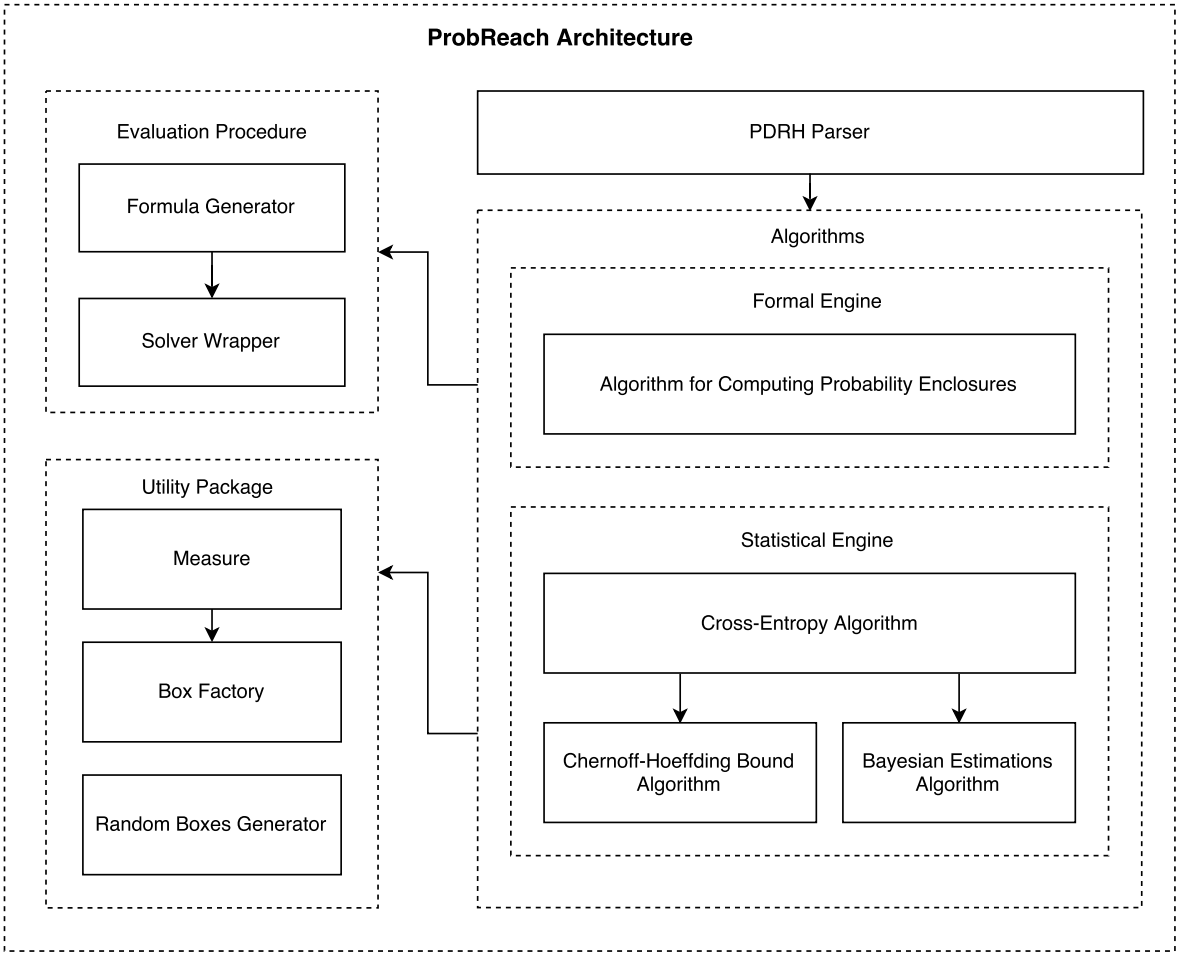


Figure 5.2: The `ProbReach` Architecture.

5.3.1 PDRH Parser

PDRH Parser incorporates two components: a C preprocessor and a PDRH model parser implemented using Flex and Bison [57]. The former is used for resolving constants and functions defined with C `#define` macro and removing comments. Note that the C preprocessor is not a required component as comments are not necessary

for the correct model specification, and constants can be defined by a means of PDRH syntax. The PDRH model parser translates the input model specified in PDRH format into its program representation.

5.3.2 Utility Package

This package implements the algorithms from Section 3.5, the methods for sampling the system's parameter space used in Algorithms 7, 8 and 9, and some auxiliary methods for parameter boxes.

- *Box Factory* implements the procedure **bisect** (Algorithm 4) for partitioning parameter boxes, the methods for computing their volumes, sorting and checking their intersection. All calculations are performed using the interval library CAPD[1]. *Box Factory* also uses native C++11 methods for sorting the lists of boxes.

- *Measure*. Firstly, it implements the procedure **measure** (Algorithm 5) which is used by *Formal Engine* for computing the probability values for the random parameter boxes. CAPD is utilised here for computing the fourth derivative of the integrands in Algorithm 6. Secondly, it implements the procedure for bounding the support of continuous random parameters with unbounded support, and it uses the IBEX library[2] for computing the probability precision value ϵ for multiple continuous random parameters. Finally, it implements the methods used by the Cross-Entropy Algorithm for sorting the parameter boxes by the mid-value of the corresponding probability enclosures.

- *Random Boxes Generator* generates random samples from all supported distributions using the GNU Scientific Library (GSL) [32]. It is used by the *Statistical Engine* for drawing random samples.

5.3.3 Evaluation Procedure

Evaluation Procedure implements the procedure **compute** (Algorithm 2). It is used by both engines of the *Algorithms* package for evaluating the parameter boxes, and it

[1] `http://capd.ii.uj.edu.pl/`
[2] `http://www.ibex-lib.org/`

incorporates: *Formula Generator*, which produces the formulae in the format required by the solver and *Solver Wrapper*, implementing the methods for executing the utilised SMT solver and parsing the solver output. Currently, `ProbReach` supports two solvers: `dReal` [36] and `iSAT-ODE` [22].

- *Formula Generator*. `dReal` uses and extended version of the SMT2 format for specifying the input files. The extension is made to support ODEs. *Formula Generator* produces up to $l+1$ formulae for the given reachability depth l (formula **reach** and l formulae $\mathbf{fail}_j$). The formulae are not generated in advance. They are created as they are evaluated by `dReal`. Also, it does not generate a formula if the set **Paths** is empty.

  `iSAT-ODE` is a standalone tool for verification of hybrid systems and it accepts models specified in its own HYS format. This language allows defining a PHS without random parameters which can be further evaluated by `iSAT-ODE`. The reachability depth can be passed as a command line argument to the solver. Thus, the task of unrolling the reachability formula is delegated entirely to `iSAT-ODE`.

- *Solver Wrapper*. `ProbReach` treats solvers as standalone applications. The solver execution is performed via a system call. During the system call the standard output is redirected to a file which is then parsed upon the solver's termination. As multiple solvers are supported by `ProbReach`, *Solver Wrapper* also detects the solver type automatically.

Evaluation Procedure is the most computationally consuming component of the tool, as it solves a PSPACE-complete problem. The reason for employing multiple solvers is that they perform differently depending on the evaluated formula. In particular, it was observed that sometimes `iSAT-ODE` takes less time for verifying satisfiable formulae, while `dReal` is faster for unsatisfiable formulae.

5.3.4 Algorithms

The *Algorithms* package combines *Formal Engine* and *Statistical Engine* implementing the algorithms for computing the bounded reachability probability. Also, all implemented algorithms are parallelised using OpenMP [66].

- *Formal Engine* implements Algorithm 3. It uses *Evaluation Procedure* for evaluating the parameter boxes, *Box Factory* for the parameter space partitioning, and *Measure* for calculating the probability values for the random parameter boxes.
- *Statistical Engine* implements Algorithms 7, 8 and 9. It uses *Random Boxes Generator* for sampling the parameter boxes (both random and nondeterministic), and *Evaluation Procedure* for evaluating the obtained samples. *Cross-Entropy Algorithm* also uses *Box Factory* to determine whether the generated sample belongs to the nondeterministic parameter space of the system.

5.4 Usage

`ProbReach` can be executed by running the following command:

```
ProbReach <options> <file.pdrh/file.drh> <solver-options>
```

The examples below demonstrate application of both verification approaches to SCB model. The full list of command line arguments available in `ProbReach` can be found in the tool documentation[1].

Example 5.1. (Applying Formal Engine to SCB)

```
ProbReach -k 2 --solver dReal -e 1e-3 --partition-prob
              --precision-nondet K 5e-2 cannon-ball-nondet.pdrh
```

where

`-k 2` - *specifies the reachability depth* $l = 2$.

`--solver dReal` - *specifies the full path to the solver executable. In this example it is assumed that the directory containing* `dReal` *is added to the path or defined as a symbolic link.*

`-e 1e-3` - *specifies the desired probability enclosure length* $\epsilon = 10^{-3}$.

`--partition-prob` - *instructs ProbReach to partition the domain of continuous random parameters before executing Algorithm 3.*

[1] `https://github.com/dreal/probreach/blob/master/doc/usage.md`

`--precision-nondet K 5e-2` - *defines the precision vector for nondeterministic parameters* $\rho = \{5 \cdot 10^{-2}\}$.

`cannon-ball-nondet.pdrh` - *specifies the full path to the file containing the PDRH model from Figure 5.1. In this case it is assumed that the model file and the* ***ProbReach*** *executable are located in the same directory.*

ProbReach *produces the following output:*

```
K:[5.00000000e-01,5.25000000e-01]; | [1.43947855e-01,1.71953293e-01]
K:[5.25000000e-01,5.50000000e-01]; | [1.64279428e-01,1.99103247e-01]
K:[5.50000000e-01,5.75000000e-01]; | [1.91010615e-01,2.28559788e-01]
K:[5.75000000e-01,6.00000000e-01]; | [2.18445265e-01,2.60084294e-01]
K:[6.00000000e-01,6.25000000e-01]; | [2.49595461e-01,2.87977794e-01]
K:[6.25000000e-01,6.50000000e-01]; | [2.71572265e-01,3.30073351e-01]
K:[6.50000000e-01,7.00000000e-01]; | [3.11115758e-01,4.26417024e-01]
K:[7.00000000e-01,7.25000000e-01]; | [3.91849026e-01,4.39309007e-01]
K:[7.25000000e-01,7.50000000e-01]; | [4.17414938e-01,4.78222884e-01]
K:[7.50000000e-01,7.75000000e-01]; | [4.56227071e-01,5.39012345e-01]
K:[7.75000000e-01,8.00000000e-01]; | [5.04066525e-01,5.77470467e-01]
K:[8.00000000e-01,8.25000000e-01]; | [5.42726540e-01,6.02637703e-01]
K:[8.25000000e-01,8.50000000e-01]; | [5.68108950e-01,6.39373616e-01]
K:[8.50000000e-01,9.00000000e-01]; | [6.05280209e-01,6.96983937e-01]
```

These probability enclosures were visualised in Figure 3.2 (the **red** *boxes).*

Example 5.2. (Applying Statistical Engine to SCB)

```
./ProbReach -k 2 --solver isat-ode --bayesian-acc 5e-3
                --bayesian-conf 0.99 --cross-entropy
                --cross-entropy-term-arg 1e-2 cannon-ball-nondet.pdrh
```

where

`--bayesian-acc 5e-3` - *specifies the half-size of the confidence interval to be computed by Algorithm 8.*

`--bayesian-conf 0.99` - *specifies the confidence value for Algorithm 8.*

`--cross-entropy` - *instructs ProbReach to use Algorithm 9.*

`--cross-entropy-term-arg 1e-2` - *specifies the terminal variance value for Algorithm 9.*

`ProbReach` produces the following output:

```
K:[8.87191917e-01,8.87191917e-01]; | [6.72533921e-01,6.82551161e-01]
```

All results for the running example in this thesis were obtained using `ProbReach`.

5.5 Discussion

In this chapter I presented `ProbReach`, a tool that I developed for computing bounded reachability probability in SPHSs. `ProbReach` provides C++ implementation of the algorithms from Chapters 2, 3 and 4, and it does not require any proprietary software. Also, `ProbReach` supports multiple SMT solvers (*i.e.*, `iSAT-ODE` and `dReal`), and can utilise any SMT solver supporting nonlinear ODEs and providing δ-decisions once the corresponding *Formula Generator* and *Solver Wrapper* are implemented. The implemented algorithms were parallelised using OpenMP for increasing the tool performance.

5.5.1 Future Work

There are several directions for future work. First of all, a more efficient parallelisation strategy should be implemented. This will require developing a sophisticated parallelisation manager monitoring the CPUs availability and dynamically distributing the load equally between the threads, in order to reduce CPU idle. This improvement can significantly increase the `ProbReach` performance.

Secondly, although formal model checking is a crucial problem, model simulation is also very important for the design process. Generally, simulation requires fewer computational resources than verification, and it can provide visual aid for better understanding the model's behaviour and potentially reduce the search space for the verification tools. Thus, another improvement could be an implementation of a translator from `ProbReach` input format into Stateflow/Simulink [18] – a state-of-art tool for model specification and simulation.

Moreover, investigating and benchmarking available SMT solvers and their further incorporation within `ProbReach` could also be beneficial.

Finally, improvement of sorting techniques, the procedure for verified integral computation, and general code enhancement can greatly contribute to increasing the performance of `ProbReach`.

Chapter 6

Case Studies

6.1 Introduction

This chapter demonstrates the application of `ProbReach` to several case studies. Section 6.2 features simple models, while Sections 6.3 and 6.4 present more complex case studies such as the automated synthesis of safe and robust PID controllers for the artificial pancreas, and UVB irradiation therapy for treating psoriasis.

All experiments were conducted on a 32-core (2.9 GHz) Ubuntu 16.04 machine.

6.2 Exploring `ProbReach` Settings

This section features several relatively simple nonlinear hybrid (and non-hybrid) models. The `ProbReach` settings and computation details are presented in Table 6.1. The main aim of the experiments conducted in this section is to study how different tool settings and model complexity (*i.e.*, the number of system's parameters) affect the computation result, and the `ProbReach` performance.

6.2.1 Good and Bad

This section presents a simple introductory example considering a single mode non-hybrid system with constant flow dynamics ($\frac{dx}{dt} = 0$). The initial state of the system is defined by the predicate $(x(0) = r) \wedge (n \in [0, 1])$, where r is uniformly distributed over [0,1], and n is a continuous nondeterministic parameter on $[0, 1]$. `ProbReach` was used

Model	ϵ	ρ	η	$\lvert L\rvert$	**Time**
Good	10^{-3}	$\{10^{-2}\}$	10^{-3}	128	9.0
Bad	10^{-3}	$\{10^{-2}\}$	10^{-3}	128	9.4
Deceleration	10^{-3}	$\{10^{-1}\}$	10^{-3}	32	30,018
Collision (**Basic**)	10^{-3}	$\{10^{-1}\}$	10^{-1}	7	240
Collision (**Basic**)	10^{-3}	$\{10^{-2}\}$	10^{-1}	64	1,080
Collision (**Basic**)	10^{-2}	$\{-\}$	10^{-1}	1,349	21,420
Collision (**Extended**)	10^{-3}	$\{5 \cdot 10^{-2}\}$	10^{3}	32	1,080
Collision (**Extended**)	10^{-3}	$\{10^{-2}\}$	10^{3}	128	3,420
Collision (**Extended**)	10^{-3}	$\{5 \cdot 10^{-2}\}$	10^{2}	32	128,220
Collision (**Advanced**)	10^{-3}	$\{10^{-1}, 10^{-1}\}$	10^{3}	256	119,074
Collision (**Advanced**)	10^{-4}	$\{10^{-1}, 10^{-1}\}$	10^{3}	256	225,863
Anaesthesia	5×10^{-2}	N/A	10^{-3}	1	80,823

Table 6.1: `ProbReach` settings and computation details for the case studies from Section 6.2, where **Model** - name of the model, ϵ - precision on the size of the probability enclosures, ρ - nondeterministic parameters precision vector, η - multiplier setting the precision for the solver, $|L|$ - number of computed probability enclosures, **Time** - CPU time in seconds, $\{-\}$ denotes that the precision on the nondeterministic parameters is ignored, and `ProbReach` terminates when all probability enclosures are of size smaller than or equal to ϵ, and N/A indicates that the model does not feature nondeterministic parameters.

for computing 0-step bounded reachability probability for two different goals: a *good* goal defined by the predicate $(x \leq 0.9n + 0.1) \wedge (x \geq 0.9n)$, and a *bad* one represented by $(x \leq 2(n - 0.5)^2 + 0.5) \wedge (x \geq -2(n - 0.5)^2 + 0.5)$.

The projections of the goal set G on the domain of continuous random parameters $P_R = [0, 1]$ for each n are the intervals $[0.9n, 0.9n + 0.1]$ and $[-2(n - 0.5)^2 + 0.5, 2(n - 0.5)^2 + 0.5]$ for the *good* and the *bad* cases, respectively. As the random parameter r is distributed uniformly and x is a constant, the probability of reaching the goal can be obtained as the difference of the right-hand side and the left-hand side of these intervals. Thus, in the *good* case the probability function is constant $\mathbf{Pr}(n) = 0.1$, and in the *bad* case it is equal to $\mathbf{Pr}(n) = 4n^2 - 4n + 1$, which reaches its minimum value of 0 at $n = 0.5$ and the maximum value of 1 at $n = 0$ and $n = 1$. Figure 6.1 demonstrates that the graphs of the reachability probability functions obtained analytically are fully

contained within the computed probability enclosures.

6.2.2 Car Deceleration Scenario

This case study considers a car deceleration scenario represented by a 2-step bounded reachability problem in a three-mode SPHS (Figure 6.2).

In the initial mode the car accelerates from 0 to 27.78 m/s (0 to 100 km/h). During this stage its velocity changes as $\frac{dv(t)}{dt} = \alpha \exp(-\alpha t + \beta) - c_d v^2(t)$, where $\alpha = 0.05776$ and $\beta \sim \mathcal{N}(4, 0.1)$ are coefficients modelling the acceleration properties of the car, and $c_d = 3.028 \cdot 10^{-4}\ m^{-1}$ is the drag coefficient. When the target velocity 27.78 m/s is reached, the driver takes $t_{react} = 1.2$ seconds to react and to start decelerating. In the "reaction" mode the car is not accelerating, and its velocity is governed by the equation $\frac{dv(t)}{dt} = -c_d v^2(t)$. In the final (braking) mode the car is decelerating according to the equation $\frac{dv(t)}{dt} = \mu a_d - c_d v^2(t)$ where $a_d \in [4.0, 6.0]$ is the car's deceleration (a nondeterministic parameter), and $\mu = 1$ is the coefficient modelling the road properties (*e.g.*, slope, friction). Throughout the modes the distance $s(t)$ travelled by the car is governed by $\frac{ds(t)}{dt} = v(t)$. `ProbReach` was applied to the described model for computing the probability of stopping within 400 metres. The obtained probability enclosures are shown in Figure 6.3, and the `ProbReach` settings are given in Table 6.1.

6.2.3 Cars Collision Scenario

This case study considers a cars collision scenario represented by a 2-step bounded reachability problem in a three mode SPHS (Figure 6.4).

Two cars (Car1 and Car2) are moving on the same lane, starting at $s_1(0) = 0$ and $s_2(0) = v_1 \cdot t_{safe}$ respectively (where t_{safe} is a time interval for maintaining a safe distance between the cars). The initial speed of both cars is 11.12 m/sec. In the initial mode Car1 changes lanes and starts accelerating at $a_{a1} = 3\ m/sec^2$, while Car2 is moving with the constant speed v_2. Car1 keeps speeding up until it gets ahead of Car1 by the distance $v_2 \cdot t_{safe}$. Then the system switches to the next mode where Car1 returns to the initial lane and starts decelerating at a_{d1}. The driver of Car2 takes t_{react} seconds to react before braking. Then the system switches to the final mode where Car2 decelerates as well with acceleration a_{d2} until it stops. `ProbReach` was used for computing a set of enclosures for the probability of the cars colliding for three different versions

of this model: **Basic** - featuring one random and one nondeterministic parameter, **Extended** - with two random and one nondeterministic parameter, and **Advanced** - containing two random and two nondeterministic parameters. The parameter values and distributions used in these models are given in Table 6.2. This case study also shows how the parameters ϵ, ρ and η affect the quality of the returned answer and the performance of Algorithm 3. The `ProbReach` settings are presented in Table 6.1.

Model	a_{d1}	a_{d2}	t_{safe}	t_{react}
Basic	$\mathcal{N}(-2.0, 0.2)$	[-0.7, -0.3]	1.0	1.5
Extended	$\mathcal{N}(-2.0, 0.2)$	$\mathcal{N}(-0.5, 0.1)$	[1.0, 2.0]	1.5
Advanced	$\mathcal{N}(-2.0, 0.2)$	$\mathcal{N}(-0.5, 0.1)$	[1.0, 2.0]	[0.5, 1.5]

Table 6.2: Parameter values and distributions for the cars collision model, where **Model** - name of the model, a_{d1} - deceleration of Car1, a_{d2} - deceleration of Car2, t_{safe} - time for maintaining safe distance, t_{react} - reaction time of the driver in Car2, $\mathcal{N}(\mu, \sigma)$ - represents the normal distribution with mean μ and standard deviation σ.

Basic Model Figure 6.5 demonstrates that ρ can greatly affect the size of the resulting probability enclosures. As a result of changing ρ from $\{10^{-1}\}$ to $\{10^{-2}\}$, the number of probability enclosures and the computation time increased (see Table 6.1) but the returned probability enclosures decreased in size.

Finally, when ρ was ignored, `ProbReach` produced 1,349 probability enclosures, and the size of each of them was smaller or equal to ϵ. This demonstrates that if the probability function is continuous, then Algorithm 3 can provide ϵ-guarantees. This, of course, cannot be guaranteed in general, due to the use of **compute** instead of **evaluate**.

Extended Model The results obtained for the **Extended** model (Figure 6.6) demonstrate that ρ is not the only factor affecting the quality of the results. It can be seen that reducing ρ from $\{5 \cdot 10^{-2}\}$ to $\{10^{-2}\}$ and using the same η did not result into tighter probability enclosures and required more computation time (1,080 and 3,420 seconds, respectively).

At the same time changing η from 10^3 to 10^2 decreased the size of each probability enclosure almost thrice at a cost of an increase in computation time by a factor of 120

(see Table 6.1).

Advanced Model Figure 6.7 demonstrates the results obtained for the **Advanced** model where the surfaces on the left-hand side represent the lower bounds of the probability enclosures returned by `ProbReach`, and the ones on the right-hand side are the upper bounds on the probability enclosures. Therefore, the bounded reachability probability function lies between the two surfaces. Reducing ϵ from 10^{-3} to 10^{-4} resulted into obtaining tighter probability enclosures. This happened due to a finer pre-partitioning of the domain of continuous random parameters (*i.e.*, the random parameter space was partitioned according to the specified ϵ prior to the algorithm execution).

6.2.4 Pharmocokinetics Model for Anaesthesia Delivery

The aim of this experiment is to apply the formal engine of `ProbReach` to a model featuring more than two continuous random parameters. This case study considers a pharmacokinetics model for anaesthesia delivery which tracks how the drug concentration changes as it is being metabolised by the body [33]. The model features three species: c_p - concentration of the drug in the plasma, c_1 - concentration of the drug in the fast peripheral compartment, and c_2 - concentration of the drug in the slow peripheral compartment. The dynamics of the system are governed by a set of differential equations (6.1), and the parameter values are given in Table 6.3.

$$\begin{aligned} &\frac{dc_p(t)}{dt} = -(k_{10}+k_{12}+k_{13})c_p(t) + k_{12}c_1(t) + k_{13}c_2(t) + \frac{u(t)}{V_1}, \\ &\frac{dc_1(t)}{dt} = k_{21}c_p(t) - k_{21}c_1(t), \ \frac{dc_2(t)}{dt} = k_{31}c_p(t) - k_{31}c_2(t), \\ &\frac{du(t)}{dt} = p\cos(\frac{2t\pi}{T_j}). \end{aligned} \tag{6.1}$$

Param.	**Value**	**Param.**	**Value**	**Param.**	**Value**	**Param.**	**Value**
k_{10}	$0.1527 \cdot M^{-0.3}$	k_{12}	0.114	k_{13}	0.0419	k_{21}	0.055
k_{31}	0.0033	V_1	$458.4 \cdot M$	M	35	p	100
$c_p(0)$	3.0	$c_1(0)$	3.0	$c_2(0)$	3.0	$u(0)$	7000
Δu_i	$\mathcal{N}(0, 210)$						

Table 6.3: Parameter values and initial conditions for the anaesthesia delivery model.

The scenario investigated here assumes that every 15 minutes, starting at time 0, the drug infusion rate $u(t)$ can change by $\Delta u \sim \mathcal{N}(0, 210)$ (see Figure 6.8). `ProbReach` computes the probability of reaching the unsafe state $(c_p(t) \geq 6) \vee (c_p(t) \leq 1) \vee (c_1(t) \geq 10) \vee (c_1(t) \leq 0) \vee (c_2(t) \geq 10) \vee (c_2(t) \leq 0)$ in 3 jumps within 60 minutes. Hence, the model features 4 continuous random parameters Δu_i (one in the initial state and one per each jump). As there are no nondeterministic parameters, `ProbReach` returns only one probability enclosure of the required length ϵ.

Initially, the experiment was conducted with $\epsilon = 10^{-2}$. However, `ProbReach` did not reach the required precision with 360 hours requiring almost 10 Gigabytes of RAM for storing the parameter boxes partitioning the parameter space. This demonstrates that the computation time grows dramatically with the number of parameters. Increasing the value of ϵ to 5×10^{-2} resulted in obtaining probability enclosure $[0.009769, 0.042274]$ of length 0.032505 within 80,823 seconds.

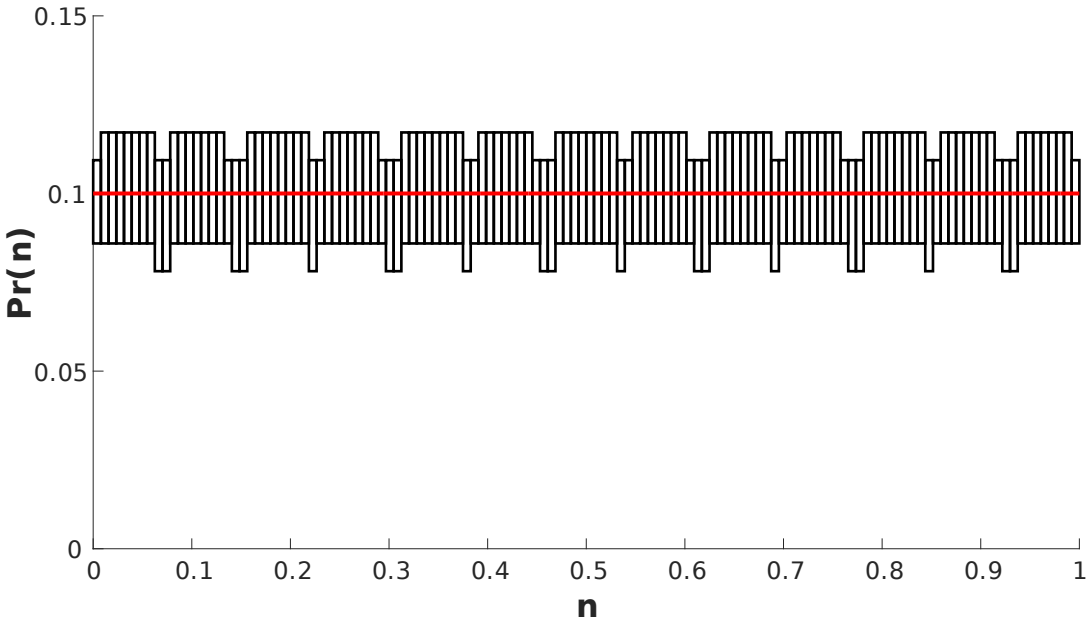
The statistical method of `ProbReach` was also used for sensitivity analysis of the parameters from Table 6.3. The aim of this experiment is to conclude how small changes in the parameter values affect the reachability probability. This was done by introducing intervals around the parameter values, thus, making the corresponding parameters nondeterministic (see Table 6.4). Now, obtaining confidence intervals for the minimum and the maximum reachability probabilities allows assessing how much the changes in the parameter values affect the value of the reachability probability. `ProbReach` computed confidence intervals $[0, 0.00568004]$ and $[0.207575, 0.217575]$ for the minimum and the maximum reachability probabilities, respectively. Therefore, the range of the reachability probability function on the considered parameter domain is guaranteed to be *at least* $[0, 0.217575]$ with confidence $c \geq 0.99$. This, however, does not guarantee that the range of the reachability probability function cannot be greater than the obtained interval, as the Cross-Entropy algorithm used by the statistical engine provides only an approximate value of the maximum/minimum. In order to provide stronger guarantees the formal method should be used, but its application is limited due to its inefficiency for problems with large number of parameters.

Param.	Interval	Param.	Interval	Param.	Interval	Param.	Interval
k_{31}	$[0.003, 0.004]$	k_{12}	$[0.1, 0.2]$	k_{13}	$[0.04, 0.05]$	k_{21}	$[0.05, 0.06]$
M	$[30, 40]$	p	$[90, 110]$	$c_p(0)$	$[2.9, 3.1]$	$c_1(0)$	$[2.9, 3.1]$
$c_2(0)$	$[2.9, 3.1]$	$u(0)$	$[6900, 7100]$				

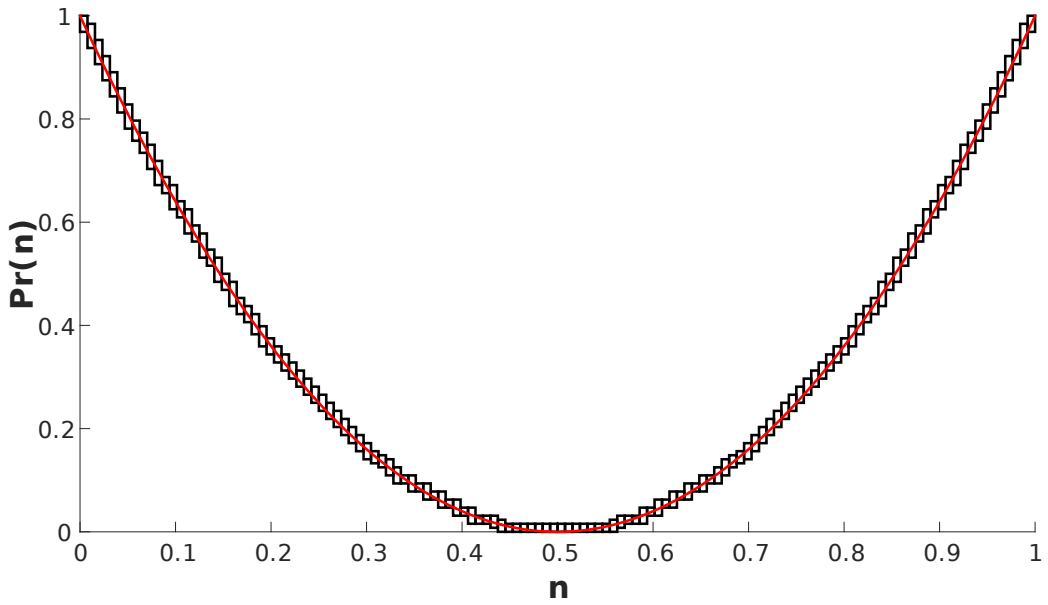
Table 6.4: Parameter intervals for sensitivity analysis in the anaesthesia delivery model.

6.2.5 Applying the Statistical Engine

The statistical engine of `ProbReach` was applied to computing the minimum and the maximum reachability probabilities in the case studies presented above. All experiments were conducted using the Cross-Entropy algorithm (Algorithm 9) with the default values: sample size $s = 10$, terminal variance $\hat{\sigma}^2 = 10^{-2}$ and elite samples ratio $\lambda = 10^{-1}$, and initial parameter of Beta distribution $\alpha = \beta = 1$ for the Bayesian estimations algorithm (Algorithm 8). The obtained results are shown in Table 6.5.



a) *Good* case.



b) *Bad* case.

Figure 6.1: Probability enclosures with respect to nondeterministic parameter n for the *good* and the *bad* cases of the introductory model, where **black** boxes - probability enclosures computed by `ProbReach` and **red** line - graph of the reachability probability function **Pr** obtained analytically.

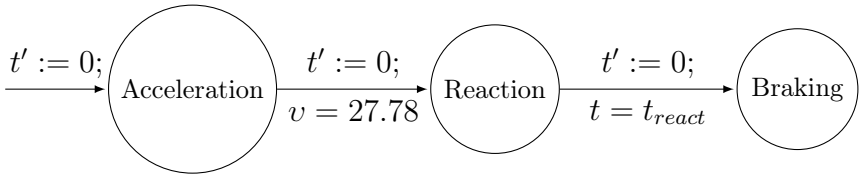


Figure 6.2: SPHS modelling car deceleration scenario.

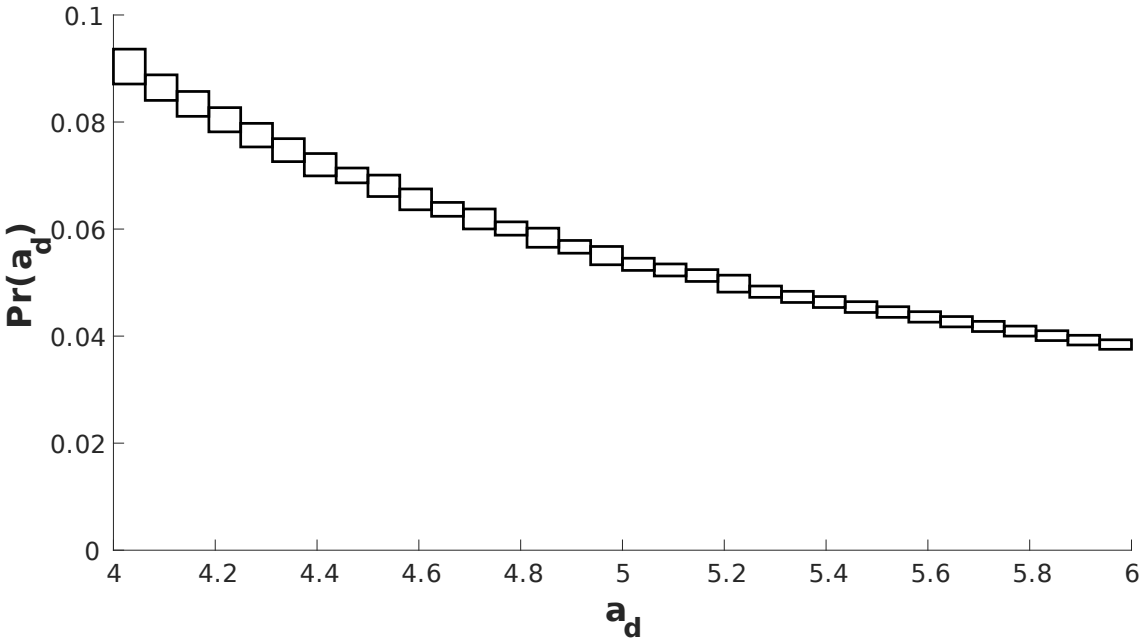


Figure 6.3: Probability enclosures with respect to nondeterministic parameter a_d for the car deceleration scenario.

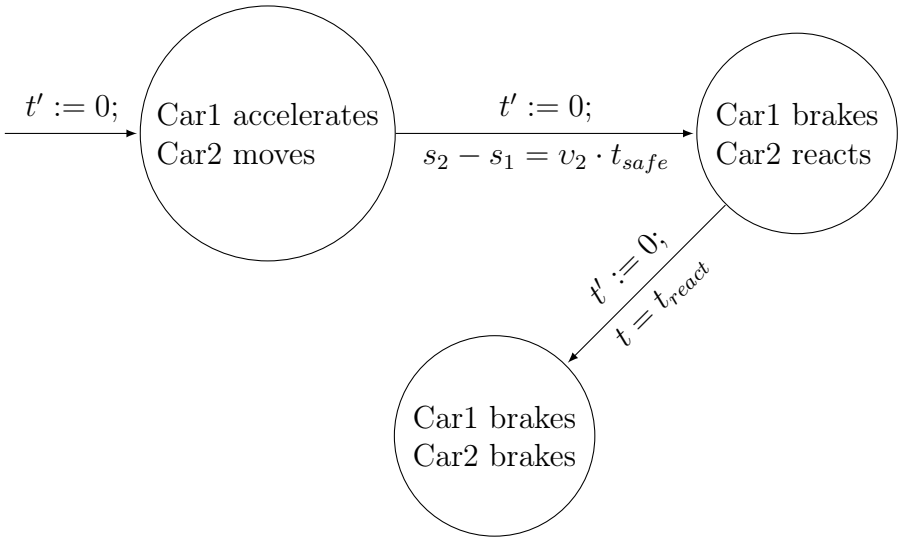


Figure 6.4: SPHS modelling the cars collision scenario.

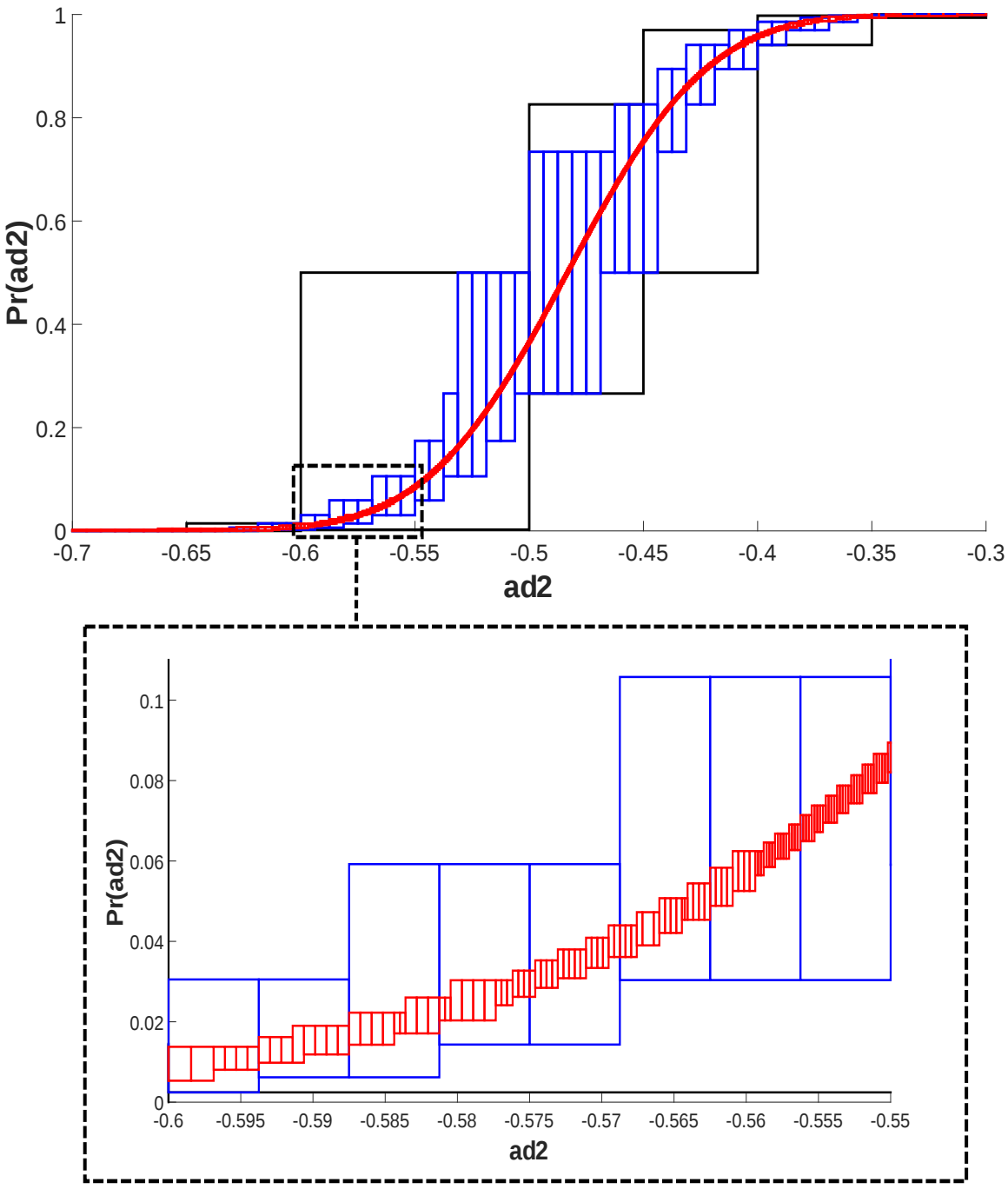


Figure 6.5: Probability enclosures with respect to nondeterministic parameter a_{d2} for the **Basic** model of the cars collision scenario (Section 6.2.3) with $\eta = 10^{-1}$ and different values of ρ and ϵ, where the **black** boxes are obtained with $\rho = \{10^{-1}\}$ and $\epsilon = 10^{-3}$, the **blue** boxes – with $\rho = \{10^{-2}\}$ and $\epsilon = 10^{-3}$, and the **red** boxes represent the setting where $\epsilon = 10^{-2}$ and ρ is ignored and `ProbReach` terminates when the size of each probability enclosure reaches ϵ. The bottom of the figure features a magnified image of the region where $a_{d2} \in [-0.6, -0.55]$.

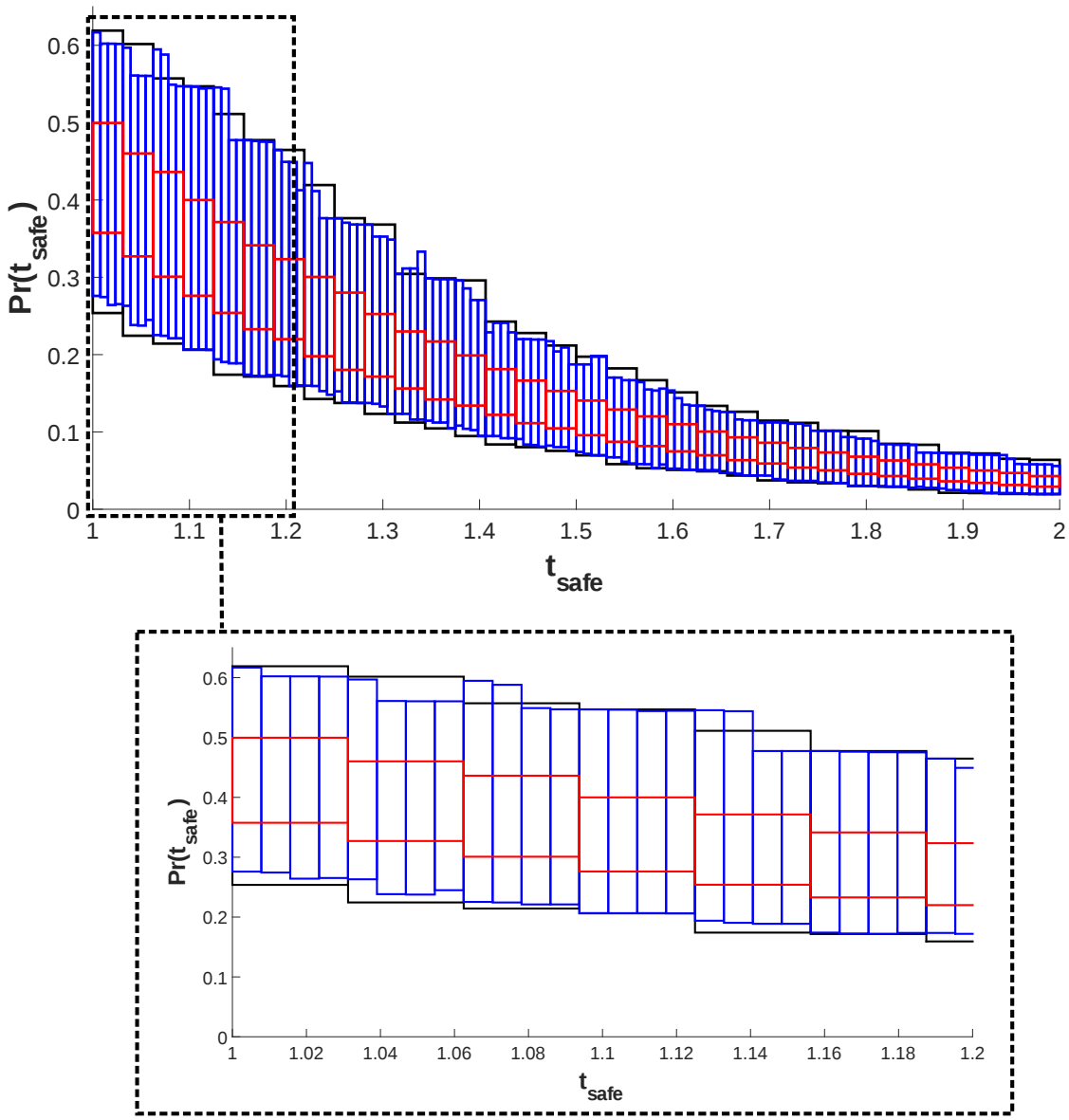
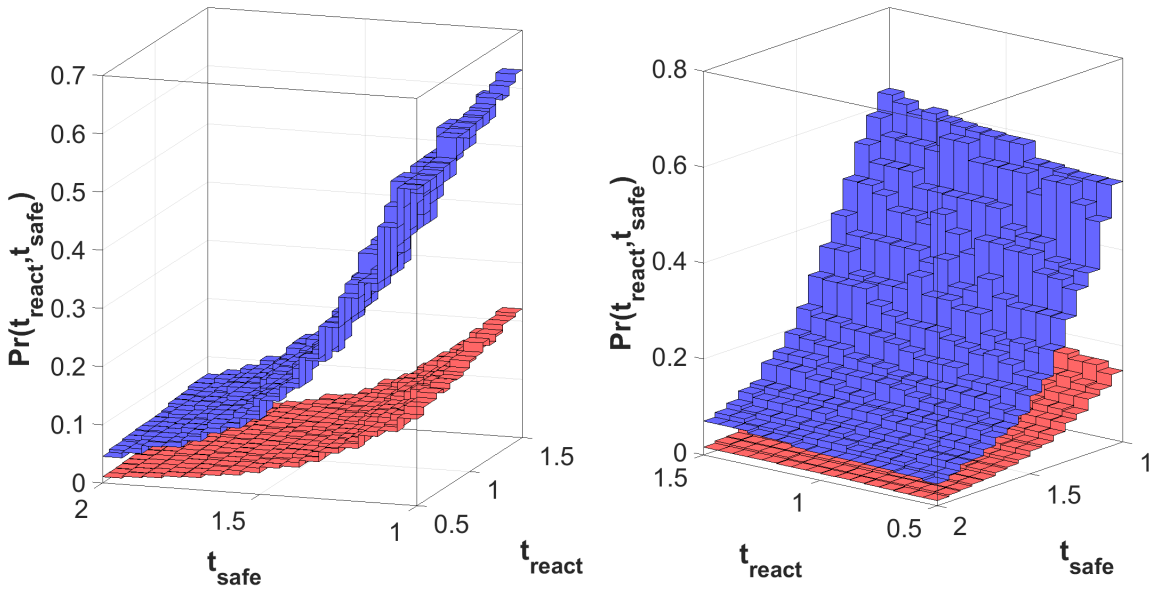
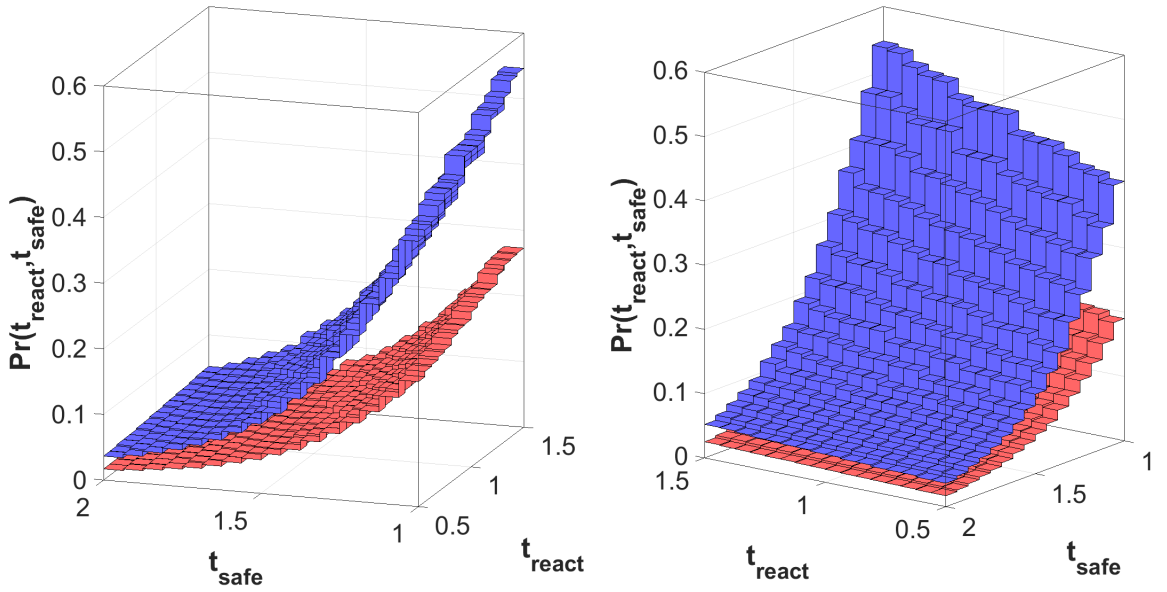


Figure 6.6: Probability enclosures with respect to nondeterministic parameter t_{safe} for the **Extended** model of the cars collision scenario (Section 6.2.3) with $\epsilon = 10^{-3}$ and different values of ρ and η, where the **black** boxes are obtained with $\rho = \{5 \cdot 10^{-2}\}$ and $\eta = 10^3$, the **blue** boxes - with $\rho = \{10^{-2}\}$ and $\eta = 10^3$, and the **red** boxes represent the setting where $\rho = \{5 \cdot 10^{-2}\}$ and $\eta = 10^2$. The bottom of the figure features a magnified image of the region where $t_{safe} \in [1, 1.2]$.



a) Results obtained with $\epsilon = 10^{-3}$, $\rho = \{10^{-1}, 10^{-1}\}$ and $\eta = 10^3$ and visualised from two different angles.



b) Results obtained with $\epsilon = 10^{-4}$, $\rho = \{10^{-1}, 10^{-1}\}$ and $\eta = 10^3$ and visualised from two different angles.

Figure 6.7: Probability enclosures with respect to nondeterministic parameters t_{safe} and t_{react} for the **Advanced** model of the cars collision scenario (Section 6.2.3), where the **red** surface - lower bound of the probability enclosures and the **blue** surface - upper bound of the probability enclosures.

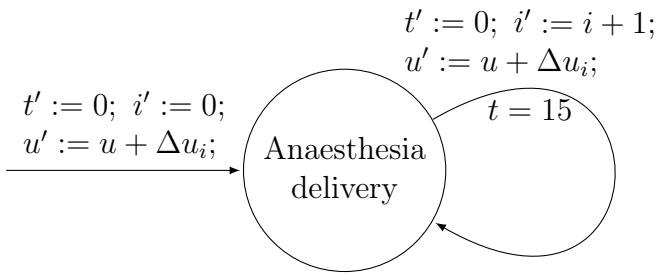


Figure 6.8: SPHS modelling anaesthesia delivery.

Model	**Type**	l	ξ	c	δ	$\mathbf{p}_N$	CI	**Time**
Good	**max**	0	5×10^{-3}	0.99	10^{-3}	$\{0.48306\}$	$[0.09031, 0.10031]$	711
	min	0	5×10^{-3}	0.99	10^{-3}	$\{0.09141\}$	$[0.09180, 0.10180]$	1,354
Bad	**max**	0	5×10^{-3}	0.99	10^{-3}	$\{0.98737\}$	$[0.94413, 0.95413]$	15,432
	max	0	5×10^{-3}	0.99	10^{-3}	$\{0.02897\}$	$[0.88408, 0.89408]$	10,295
	min	0	5×10^{-3}	0.99	10^{-3}	$\{0.50032\}$	$[0, 0.00535]$	1,968
Deceleration	**max**	2	5×10^{-3}	0.99	10^{-3}	$\{4.11713\}$	$[0.08411, 0.09411]$	2,369
	min	2	5×10^{-3}	0.99	10^{-3}	$\{5.69707\}$	$[0.03544, 0.04544]$	2,237
Collision	**max**	2	5×10^{-3}	0.99	10^{-3}	$\{-0.39351\}$	$[0.96413, 0.97413]$	37,968
(**Basic**)	**min**	2	5×10^{-3}	0.99	10^{-3}	$\{-0.67266\}$	$[0, 0.00534]$	15,444
Collision	**max**	2	5×10^{-3}	0.99	10^{-3}	$\{1.00841\}$	$[0.42465, 0.43479]$	77,828
(**Extended**)	**min**	2	5×10^{-3}	0.99	10^{-3}	$\{1.81601\}$	$[0.04523, 0.05523]$	15,729
Collision	**max**	2	5×10^{-3}	0.99	10^{-3}	$\{1.10781, 1.22802\}$	$[0.20794, 0.21796]$	27,094
(**Advanced**)	**min**	2	5×10^{-3}	0.99	10^{-3}	$\{1.08776, 1.92578\}$	$[0.02739, 0.03739]$	10,315
Anaesthesia	N/A	3	5×10^{-3}	0.99	10^{-3}	N/A	$[0.01378, 0.02378]$	407

Table 6.5: Results of applying the statistical engine of `ProbReach` to the case studies from Section 6.2, where **Model** - name of the model, **Type** - type of the extremum (minimum or maximum), l - reachability depth, ξ - half size of the confidence interval, c - desired confidence, δ - solver precision, $\mathbf{p}_N$ - nondeterministic parameter vector for which the minimum/maximum probability is obtained, CI - confidence interval containing the minimum/maximum probability with the confidence c, **Time** - CPU time in seconds, and N/A indicates that the corresponding option is not applicable.

The obtained results are consistent with those computed by the formal engine, as all the confidence intervals intersect with the corresponding probability enclosures.

It can be seen that `ProbReach` obtained two different estimates for the maximum probability in the *bad* model. This suggests that the Cross-Entropy algorithm can fall into local extrema if the input parameters (*e.g.*, the number of samples per iteration, the terminal variance, the elite sample ratio) are not chosen carefully.

The minimum probability estimate for the car deceleration scenario is not very accurate ($a_d = 5.69707$ is quite far from the point of minimum at $a_d = 6$) due to the chosen accuracy value being to large for the small minimum probability value.

As for the cars collision case study, in the **Advanced** version of the model t_{safe} influences the probability value more than t_{react}. Therefore, the estimate for the former is worse than the latter in both the minimum and the maximum reachability probability estimates. Also, Table 6.5 indicates that introducing more parameters to the system did not change the computation time significantly (the model with more parameters took the least time), while the time taken by the formal engine was increasing dramatically with every added parameter.

Finally, the formal approach can take less computation time than the statistical one for a small number of parameters (one or two). However, the anaesthesia delivery model demonstrated that the statistical technique completely outperforms the formal one when the number of parameters grows.

6.3 Artificial Pancreas

This case study presents an approach for the automated synthesis of safe and robust PID controllers for the closed-loop control of insulin treatment for Type 1 diabetes (T1D), also known as the *artificial pancreas* (AP) [44].

The main requirement for the AP is to keep the blood glucose (BG) level within the healthy range, typically between 70-180 mg/dL, in order to avoid *hyperglycemia* (BG above the healthy range) and *hypoglycemia* (BG below the healthy range). While some temporary hyperglycemia is allowed, hypoglycemia leads to severe health consequences, and thus, it should be avoided.

The AP consists of a continuous glucose monitor that provides measurements to a control algorithm running inside an insulin pump injecting insulin into the body. The

pump administers both *basal insulin*, a low and continuous dose that covers insulin needs outside the meals, and *bolus insulin*, a single, high dose for covering the meals.

Currently, state-of-art commercial systems can only regulate basal insulin and still require manual computation of bolus insulin. The latter is dealt with by applying PID control (one of the main control techniques for the AP [46, 82, 83]) whose purpose is minimising the difference between the measured system output and the desired value (set-point) in the presence of external disturbances. A PID controller is a sum of three components: *proportional*, *integral* and *derivative*, and its output is represented by:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de(t)}{dt}, \tag{6.2}$$

where $e(t) = sp - y(t)$ is the difference between the set-point sp and the measured system's output $y(t)$.

The goal of this experiment is finding the values of constants K_p, K_i and K_d (gains) for which the corresponding PID controller satisfy safety and robustness criteria. Safety means that some unsafe state – blood glucose level outside 70-180 mg/dL bounds – should never be reached or reached with a small probability, while robustness requires the measured system output to be close to the desired set-point, and convergence to a steady state. The robustness criteria are captured by performance indices [59] such as the *fundamental index* (FI) and the *weighted fundamental index* (FI_w) defined in (6.3).

$$FI(t) = \int_0^t (e(\tau))^2\, d\tau, \qquad FI_w(t) = \int_0^t \tau^2 \cdot (e(\tau))^2\, d\tau. \tag{6.3}$$

FI defines the total accumulated error, and thus, indicates how far the system output is from the set-point, while FI_w prioritises the error towards the end of the continuous flow, and therefore, demonstrates how well the system converges to its steady state.

6.3.1 Plant Model

The continuous system dynamics (*e.g.*, glucose and insulin concentrations) are defined by the well-established nonlinear ODE model (6.4) of Hovorka et al. [45]. The model

parameters are given in Table 6.6.

$$\begin{aligned}
&\frac{dQ_1(t)}{dt} = -F_{01} - x_1 Q_1 + k_{12} Q_2 - F_R + EGP_0(1 - x_3) + 0.18 U_G, \\
&\frac{dQ_2(t)}{dt} = x_1 Q_1 - (k_{12} + x_2) Q_2, \quad U_G(t) = \frac{D_G A_G}{0.18 t_{maxG}^2} t e^{\frac{-t}{t_{maxG}}}, \\
&G(t) = \frac{Q_1(t)}{V_G}, \ \frac{dS_1(t)}{dt} = u(t) + u_b - \frac{S_1}{t_{maxI}}, \ \frac{dS_2(t)}{dt} = \frac{S_1 - S_2}{t_{maxI}}, \\
&\frac{dI(t)}{dt} = \frac{S_2}{t_{maxI} V_I} - k_e I, \quad \frac{dx_i(t)}{dt} = -k_{a_i} x_i + k_{b_i} I, \quad (i = 1, 2, 3).
\end{aligned} \tag{6.4}$$

The model consists of three subsystems:

- *Glucose Subsystem*: it tracks the mass of glucose (in mmol) in the accessible ($Q_1(t)$) and non-accessible ($Q_2(t)$) compartments, $G(t)$ (mmol/L) represents the glucose concentration in plasma, EGP_0 (mmol/min) is the endogenous glucose production rate and $U_G(t)$ (mmol/min) defines the glucose absorption rate after consuming D_G grams of carbohydrates. D_G represents the main external disturbance of the system.

- *Insulin Subsystem*: it represents absorption of subcutaneously administered insulin. It is defined by a two-compartment chain, $S_1(t)$ and $S_2(t)$ measured in U (units of insulin), where $u(t)$ (U/min) is the administration of insulin computed by the PID controller, u_b (U/min) is the basal insulin infusion rate and $I(t)$ (U/L) indicates the insulin concentration in plasma.

- *Insulin Action Subsystem*: it models the action of insulin on glucose distribution/transport, $x_1(t)$, glucose disposal, $x_2(t)$, and endogenous glucose production, $x_3(t)$ (unitless).

The error function is defined as $e(t) = sp - Q_1(t)$ with the constant set-point $sp = \frac{110 V_G}{18}$, which corresponds to the plasma glucose concentration of $G(t) = 110$ mg/dL.

The format of the model specification in `ProbReach` does not allow defining integrals explicitly. Thus, the integrals in (6.2) and (6.3) are encoded by introducing three ODEs (6.5) whose solutions are the corresponding integrals:

$$\frac{de_{int}(t)}{dt} = sp - Q_1(t), \qquad \frac{dFI(t)}{dt} = e^2(t), \qquad \frac{dFI_w(t)}{dt} = t^2 e^2(t), \tag{6.5}$$

Param.	Value	Param.	Value	Param.	Value
w	100	k_e	0.138	k_{12}	0.066
k_{a_1}	0.006	k_{a_2}	0.06	k_{a_3}	0.03
k_{b_1}	0.0034	k_{b_2}	0.056	k_{b_3}	0.024
t_{maxI}	55	V_I	$0.12 \cdot w$	V_G	$0.16 \cdot w$
F_{01}	$0.0097 \cdot w$	t_{maxG}	40	F_R	0
EGP_0	$0.0161 \cdot w$	A_G	0.8		

Table 6.6: Parameter values for the glucose-insulin regulatory model (w (kg) is the body weight).

and the PID control (6.2) can now be encoded as

$$u(t) = K_p e(t) + K_i e_{int}(t) + K_d \frac{de(t)}{dt}.$$

In this case study the formal and the statistical methods of `ProbReach` were applied to synthesize the basal insulin infusion rate u_b, the controller parameters K_p, K_d and K_i and the maximum disturbance D_G.

6.3.2 Basal Insulin Rate Synthesis

The basal insulin rate u_b is a constant rate at which insulin is administered in the absence of external disturbances, and for which the system (6.4) reaches the steady state determined by the equation $Q_1(t) = sp$.

The aim of this experiment is two-fold: 1) synthesis of the basal rate value u_b for which the system of ODEs (6.4) reaches the approximate steady state and 2) computing the steady state.

Starting with $Q_1(0) = sp$ and 0 for the remaining differential equations, the formal approach of `ProbReach` is applied to synthesize a value for u_b such that the output glucose Q_1 reaches the interval $[sp - 0.5, sp + 0.5]$ in 2,000 minutes and remains there for the following 1,000 minutes. Since any reasonable basal rate cannot exceed 1 unit of insulin per minute, the parameter set synthesis is performed on the interval $[0, 1]$. As a result, `ProbReach` returned the interval $[0.0553359375, 0.055640625]$. Due to the absolute guarantees provided by Algorithm 3 for the synthesised parameter sets, the property above is satisfied for any value in the obtained interval, and $u_b = 0.0555$ is used for all further experiments.

Given the basal infusion rate $u_b = 0.0555$, the system's approximate steady state was obtained as the value of the system dynamics at $t = 3,000$ minutes (see Table 6.7). The system steady state is used as the initial state in all further experiments.

Var.	**Value**	**Var.**	**Value**	**Var.**	**Value**
Q_1	sp	Q_2	19.08024	S_1	3.0525
S_2	3.0525	I	0.03351	x_1	0.01899
x_2	0.03128	x_3	0.02681		

Table 6.7: Approximate value of the steady state of the ODE system (6.4) for the given set-point sp.

6.3.3 PID Controller Synthesis

Typical healthy glucose levels vary between 4 and 10 mmol/L. Since avoiding hypoglycemia ($G(t) < 4$ mmol/L) is the main safety requirement of the artificial pancreas, while temporary hyperglycemia is allowed and it is inevitable after meals, the interval $[4, 16]$ is considered safe. In this way, protection against both hypoglycemia and very severe levels of hyperglycemia is ensured.

Although insulin infusion at the basal rate $u_b = 0.0555$ helps preventing hypoglycemia ($G(t) < 4$ mmol/L), the patient is likely to experience a severe hyperglycemia ($G(t) > 16$ mmol/L) when a large meal is consumed ($D_G > 80$) or when the glucose level is not sufficiently low before the following meal is consumed (see controller C_0 in Figure 6.10 of Section 6.3.5).

The aim of this experiment is to synthesize PID controllers considering a one-day scenario consisting of three meals (breakfast, lunch and dinner) occurring at random times and with random sizes. This scenario can be expressed in terms of probabilistic bounded reachability in the SPHS defined in Figure 6.9.

The model features five random, normally-distributed parameters: the amount of carbohydrates of each meal, D_{G_1}, D_{G_2} and D_{G_3}, and the waiting times between meals, T_1 and T_2, where $T_1 \sim \mathcal{N}(300, 10)$ and $T_2 \sim \mathcal{N}(300, 10)$, and the distributions for D_{G_1}, D_{G_2} and D_{G_3} differ depending of the experiment (see Table 6.8).

The initial state of the continuous dynamics in mode $Meal$ 1 is the system steady state from Table 6.7. A meal containing D_{G_1} grams of carbohydrates is consumed

at time 0. When the time in the first mode reaches T_1 minutes the system makes a transition to the next mode $Meal$ 2 where the value of the variable D_G is set to D_{G_2} and the time is reset to 0. Analogously, the system makes a transition between modes $Meal$ 2 and $Meal$ 3, resetting variables D_G and t to D_{G_3} and 0, respectively. It is assumed that all remaining variables do not reset their values when a discrete transition takes place.

The statistical engine of `ProbReach` was applied to synthesise safe controllers C_1, C_2 by obtaining the values for K_p, K_i and K_d minimising the probability of reaching the goal state $G(t) \notin [4, 16]$ at some time point within $[0, T_1]$, $[0, T_2]$ and $[0, 1440 - T_1 - T_2]$ in mode $Meal$ 1, $Meal$ 2 and $Meal$ 3, respectively (reachability depth $l = 0$, 1 or 2).

The obtained results are presented in Table 6.8. Controller C_1 was obtained when the sizes of all three meals were assumed to be distributed normally with $\mathcal{N}(60, 20)$, while C_2 was obtained considering a more realistic daily meal profile with meal sizes being distributed as $\mathcal{N}(40, 10)$, $\mathcal{N}(90, 10)$ and $\mathcal{N}(60, 10)$, respectively. Controller C_3 was synthesized using the same realistic daily meal profile and a modified goal predicate that takes into account the performance criteria through the fundamental indices FI and FI_w, $(FI > 3.5) \vee (FI_w > 70) \vee (G(t) \notin [4, 16])$. Controller C_0 represents the case when K_p, K_i and K_d are equal to zero, meaning that insulin is administered at a constant basal rate u_b.

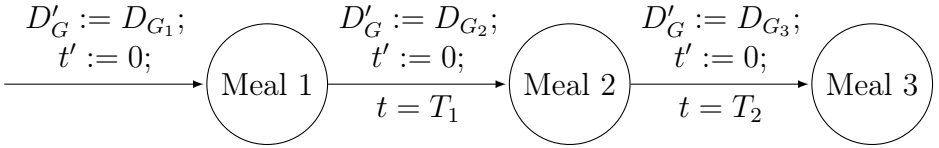


Figure 6.9: SPHS modelling the scenario of 3 meals consumed over 24 hours for the artificial pancreas model.

#	D_{G_1}	D_{G_2}	D_{G_3}	$K_d \times 10^{-2}$	$K_i \times 10^{-7}$	$K_p \times 10^{-4}$	CI	CPU_s	CPU_{CI}	ΔD_{G_1}
C_0	$\mathcal{N}(60, 20)$	$\mathcal{N}(60, 20)$	$\mathcal{N}(60, 20)$	0	0	0	$[0.86956, 0.88956]$	-	1,700	$[0, 75]$
C_0	$\mathcal{N}(40, 10)$	$\mathcal{N}(90, 10)$	$\mathcal{N}(60, 10)$	0	0	0	$[0.98861, 1]$	-	449	$[0, 75]$
C_1	$\mathcal{N}(60, 20)$	$\mathcal{N}(60, 20)$	$\mathcal{N}(60, 20)$	-6.06855	-5.61901	-5.979	$[0.09946, 0.10946]$	124,351	90,931	$[0, 88.1]$
C_2	$\mathcal{N}(40, 10)$	$\mathcal{N}(90, 10)$	$\mathcal{N}(60, 10)$	-6.02376	-3.53308	-6.166	$[0.20711, 0.21711]$	92,999	180,484	$[0, 88.07]$
C_3	$\mathcal{N}(40, 10)$	$\mathcal{N}(90, 10)$	$\mathcal{N}(60, 10)$	-5.7284	-3.00283	-6.39023	$[0.3324, 0.3524]$	152,135	200,094	$[0, 87.68]$

Table 6.8: Results of controller synthesis where: # – name of the synthesized controller, D_{G_i} – meal size distributions, K_d, K_i and K_p – synthesized values of the gain constants characterizing the corresponding controller, CI – confidence interval containing the probability of reaching the unsafe state with the confidence 0.99, CPU_s – time in seconds spent on finding the parameter values with the accuracy $2\xi = 10^{-1}$, CPU_{CI} – time in seconds spent on finding the confidence intervals with higher accuracy (2×10^{-2} and 10^{-2}) for the obtained parameter values, and ΔD_{G_1} – range of meal sizes for which the system never reaches the unsafe state.

The controller synthesis was performed applying the statistical engine of `ProbReach`, with the following domains for the controller parameters: $K_d \in [-10^{-1}, 0]$, $K_i \in [-10^{-5}, 0]$ and $K_p \in [-10^{-3}, 0]$. The controller parameters were found using the accuracy value 10^{-1}, and then the tighter confidence intervals were computed with higher accuracy (2×10^{-2} and 10^{-2}) for the obtained controllers. Note that performing the parameter search with higher precision is more beneficial as it may result into obtaining a better nondeterministic estimate. However, this will significantly increase computation time.

The results in Table 6.8 suggest that introducing PID controllers C_1 and C_2 significantly decreases the risk of reaching the unsafe state from $[0.86956, 0.88956]$ and $[0.98861, 1]$ for the basal controller C_0, to $[0.09946, 0.10946]$ and $[0.20711, 0.21711]$ for C_1 and C_2, respectively. Inspecting the confidence intervals returned for C_1,C_2 and C_3 indicates that it is slightly easier to find a controller satisfying only the safety property than both the safety and the robustness criteria ($[0.09946, 0.10946]$ and $[0.20711, 0.21711]$, as opposed to $[0.3324, 0.3524]$).

Note that controllers C_1, C_2 and C_3 can fail to maintain the safe state with the corresponding probability defined by the confidence intervals CI. Thus, the synthesized controllers can be sometimes unsafe as it will be show in Section 6.3.5.

6.3.4 Maximum Disturbance Synthesis

The aim of this experiment is to calculate the maximum initial disturbance for which the obtained controllers do not violate the safety requirements within 12 hours. This was done by performing the parameter set synthesis on the interval $[0, 120]$ with precision $\rho = \{10^{-3}\}$.

The obtained results (Table 6.8) indicate that applying a PID controller increases the size of the allowed meal from 75 for the basal controller C_0 to about 88 grams, and at the same time, the difference between C_1, C_2 and C_3 is negligibly small.

Although introducing a controller does not increase the maximum disturbance dramatically in comparison to the basal case, a PID control decreases the glucose level sufficiently enough so that a subsequent meal of similar size can be consumed without the risk of experiencing severe hyperglycemia. In contrast, C_0 does not bring the glucose level low enough before the following meal.

Also, note that the maximum meal size was computed based on the fact that the meal is consumed when the system (6.4) is in its steady state. Thus, the same controllers are capable of covering larger "second" meals, as will be shown next.

6.3.5 Performance and Safety Evaluation

The aim of this experiment is to verify safety and compute value of FI and FI_w for each of the synthesised controllers using a deterministic version of the SPHS from Figure 6.9, with $D_{G_1} = 50$, $D_{G_2} = 100$, $D_{G_3} = 70$, $T_1 = 300$ and $T_2 = 300$. The evaluation results are presented in Table 6.9, and the graphs of the glucose level and the insulin infusion rate ($u_b + u(t)$) are shown in Figure 6.10.

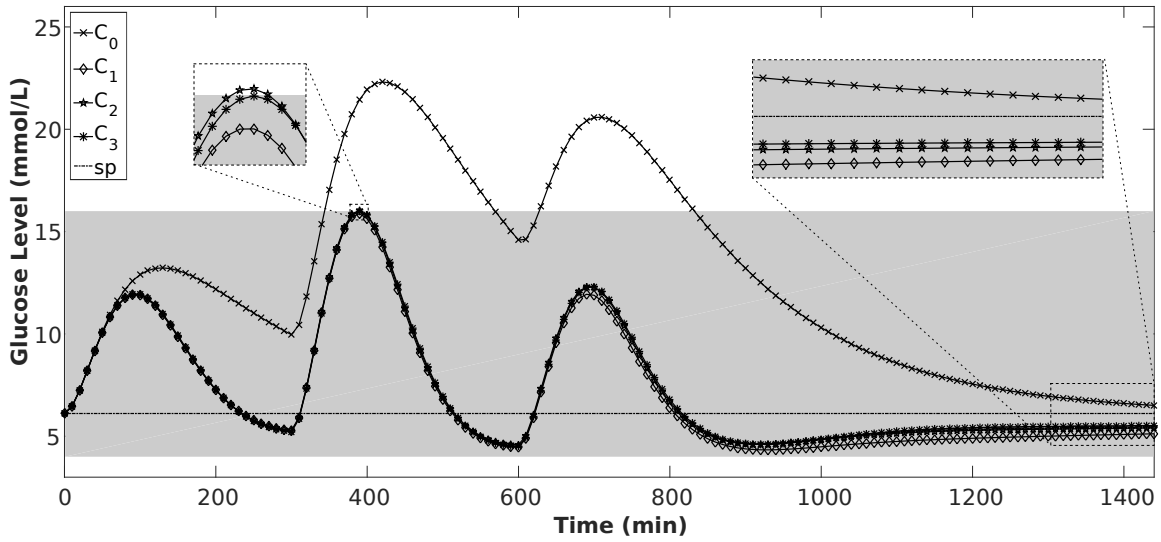
It can be seen that all three synthesised controllers perform dramatically better than C_0. The absence of a PID controller causes a long-term severe hyperglycemia, as shown in Figure 6.10. Also, the values of FI and FI_w for C_0 are significantly larger than those for the synthesised controllers C_1, C_2 and C_3.

Table 6.9 indicates that C_2 fails the safety requirement for the given meal profile and reaches a short-term severe hyperglycemia (see Figure 6.10), while the remaining two controllers keep the glucose level within the safe range.

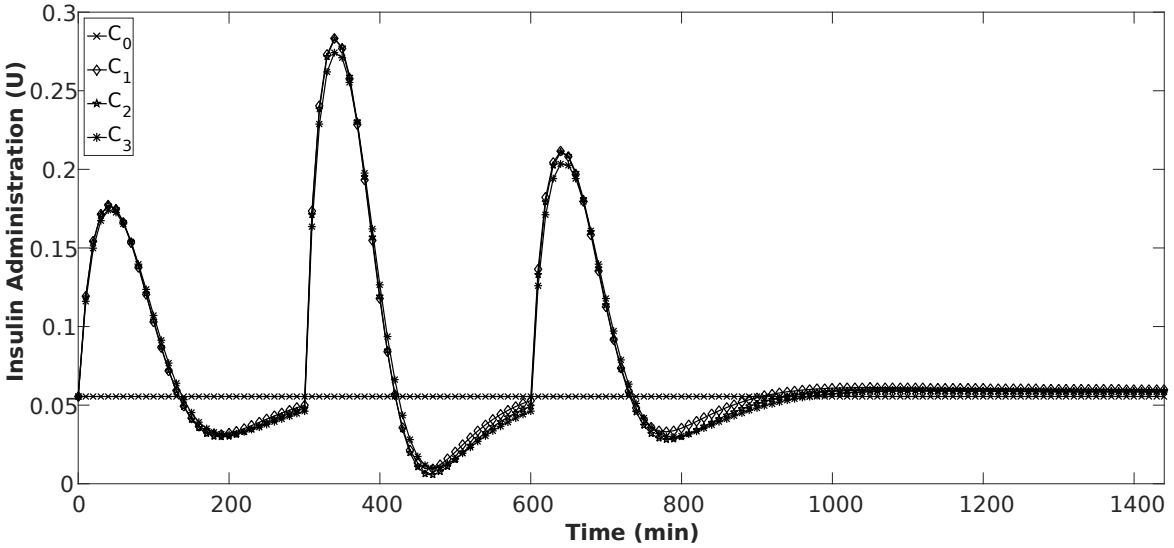
Despite featuring similar values of FI, meaning that all three controller maintain the glucose level equally far from the set-point on average, the values of the weighted fundamental index (FI_w) vary. For example, controller C_1 is worse than C_2 at keeping the glucose level close to the set-point towards the end of the considered time interval. Controller C_3 demonstrates the best performance in this aspect, implying that it has the best steady-state accuracy (thanks to the constraints on FI_w in the corresponding synthesis property). This is also confirmed by the plot in Figure 6.10.

#	Safety	$FI \times 10^{-6}$	$FI_w \times 10^{-9}$
C_0	Unsafe	26.2335	847.5063
C_1	Safe	3.89437	114.49821
C_2	Unsafe	3.95773	81.61823
C_3	Safe	3.96117	74.90655

Table 6.9: Evaluation of the synthesized PID controllers on three meals of 50, 100 and 70 grams consumed in 300-minute intervals over 24 hours.



a) Blood glucose level ($G(t)$).



b) Insulin administration (bolus insulin $u(t)$ + basal rate u_b).

Figure 6.10: Simulated blood glucose level (a) and insulin administration (b) computed by the synthesized controllers for three meals of 50, 100 and 70 grams consumed in 300-minute intervals over 24 hours.

6.4 UVB Irradiation Therapy for Treating Psoriasis

This case study considers a simplified version of a UVB irradiation therapy model [93] used for treating psoriasis, an immune system-mediated chronic skin condition which

is characterised by overproduction of keratinocytes.

The model comprises of three (six in the original model) categories of normal and three (five in the original model) categories of psoriatic keratinocytes whose dynamics are presented by nonlinear ODEs (6.6). The parameter values and the initial conditions used in this model are given in Table 6.10.

The therapy consists of several episodes of UVB irradiation, which is simulated in the model by increasing the apoptosis rate constants (β_1 and β_2) for stem cells (SC) and transit amplifying (TA) cells by In_A times. Every such episode lasts for 48 hours and is followed by 8 hours of rest ($In_A = 1$) before starting the next irradiation. The SPHS modelling the described therapy is given in Figure 6.11.

$$
\begin{aligned}
\frac{dSC}{dt} &= \gamma_1 \frac{\omega(1-\frac{SC+\lambda SC_d}{SC_{max}})SC}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n} - \beta_1 In_A SC - \frac{k_{1s}\omega}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n SC + k_1 TA},\\
\frac{dTA}{dt} &= \frac{k_{1a,s}\omega SC}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n} + \frac{2k_{1s\omega}}{1+(\omega-1)(\frac{TA+TA_d}{P_{ta,h}})^n + \gamma_2 GA - \beta_2 In_A TA - k_{2s}TA - k_1 TA},\\
\frac{dGA}{dt} &= (k_{2a,s}+2k_{2s})TA - k_2 GA - k_3 GA - \beta_3 GA\\
\frac{SC_d}{dt} &= \gamma_{1d}(1-\frac{SC+SC_d}{SC_{max,t}})SC_d - \beta_{1d} In_A SC_d - k_{1sd}SC_d - \frac{k_p SC_d^2}{k_a^2+SC_d^2} + k_{1d}TA_d),\\
\frac{dTA_d}{dt} &= k_{1a,sd}SC_d + 2k_{1sd}SC_d + \gamma_{2d}TA_d + k_{2d}GA_d - \beta_{2d}In_A TA_d - k_{2sd}TA_d - k_{1d}TA_d,\\
\frac{dGA_d}{dt} &= (k_{2a,sd}+2k_{2sd})TA_d - k_{2d}GA_d - k_{3d}GA_d - \beta_{3d}GA_d.
\end{aligned}
\tag{6.6}
$$

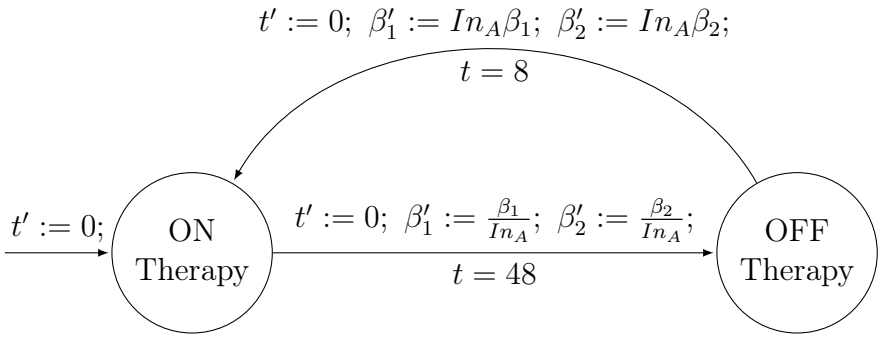


Figure 6.11: SPHS modelling the UVB irradiation therapy.

The efficiency of the therapy depends on the number of alternations between the irradiation and rest stages. An insufficient number of treatment episodes can result

Param.	Value	Param.	Value	Param.	Value
$SC(0)$	25.5285	$TA(0)$	6.1505	$GA(0)$	4.9181
$SC_d(0)$	313.934	$TA_d(0)$	1559.979	$GA_d(0)$	998.035
γ_1	0.0033	ω	100	β_{3d}	0.0003785
SC_{max}	225	$P_{ta,h}$	560.931	n	3
β_1	1.97×10^{-6}	k_{3d}	1.0805	k_{1s}	0.00164
k_1	10^{-6}	$k_{1a,s}$	0.0131	γ_2	0.014
β_2	2.08×10^{-5}	k_{2s}	0.01729	$k_{2a,s}$	0.1383
k_2	10^{-6}	k_3	0.2161	β_3	0.00026
γ_{1d}	0.0132	$SC_{max,t}$	787.5	β_{1d}	2.296×10^{-6}
k_{1sd}	0.00656	k_p	0.3	k_a	19
k_{1d}	10^{-6}	$k_{1a,sd}$	0.0524	γ_{2d}	0.056
k_{2d}	10^{-6}	β_{2d}	2.42×10^{-5}	k_{2sd}	0.06916
$k_{2a,sd}$	0.5532				

Table 6.10: UVB irradiation model parameters and initial conditions.

into early psoriasis relapse: the deterministic version of this model predicts psoriasis relapse for the number of therapy episodes less than seven [93].

`ProbReach` was applied to different configurations of the described model depending on the values of parameters In_A and λ. Namely, the following two sections feature the computation of bounded reachability probability (using both formal and statistical approaches) and parameter set synthesis.

6.4.1 Bounded Reachability Probability

For this experiment the parameter characterising the strength of UVB irradiation $In_A \sim \mathcal{N}(6 \times 10^4, 10^4)$ is set to be random, and $\lambda \in [0.2, 0.5]$ characterising the strength of psoriatic stem cells is considered nondeterministic. `ProbReach` was used for computing the probability of psoriasis relapse within 2,000 days after the last therapy episode for nine alternations ($l = 9$) between the ON and OFF therapy modes (five therapy cycles) with respect to the nondeterministic parameter λ.

The formal engine of `ProbReach` was applied with $\epsilon = 10^{-3}$, $\rho = \{10^{-2}\}$ and $\eta = 1$, and it computed 32 probability enclosures (Figure 6.12) in 251,380 seconds. It can

be seen that the range of reachability probability values spans over the interval $[0, 1]$, starting with the value close to 0 for $\lambda = 0.2$ and finishing with the value close to 1 for $\lambda > 0.45$.

The statistical engine of `ProbReach` was applied with accuracy $\xi = 5 \times 10^{-2}$, confidence $c = 0.99$ and solver precision $\delta = 1$. The computation results (see Table 6.11) demonstrate that the obtained estimates for the minimum and the maximum reachability probabilities are quite accurate ($\lambda = 0.21586$ and $\lambda = 0.42272$ respectively) despite the large accuracy value $\xi = 5 \times 10^{-2}$.

Regarding the computation times, the statistical engine required less time (about four times less) for computing both probability estimates than the formal engine. However, the formal method obtained the maximum and the minimum probabilities with better precision than the statistical approach. Providing a result with the same accuracy using the statistical approach would take considerably more time.

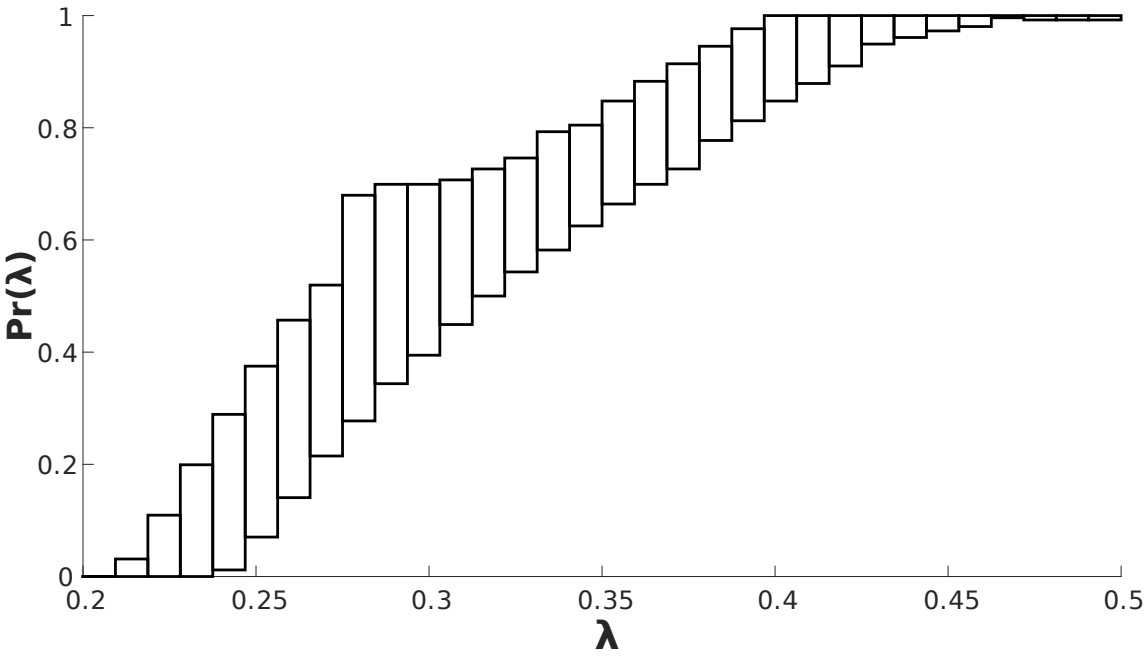


Figure 6.12: Probability enclosures with respect to the nondeterministic parameter λ for the UVB irradiation therapy model.

Type	λ	CI	**Time**
max	$\{0.42272\}$	$[0.94628, 1]$	19,356
min	$\{0.21586\}$	$[0, 0.05371]$	41,835

Table 6.11: Results of applying the Cross-Entropy algorithm to the UVB irradiation therapy model, where **Type** - type of the extremum (minimum or maximum), λ - nondeterministic parameter value for which the minimum/maximum probability is obtained, CI - corresponding confidence interval, **Time** - CPU time in seconds.

6.4.2 Parameter Set Synthesis

Parameter set synthesis was performed for the nondeterministic parameters In_A and λ over the intervals $[55000, 65000]$ and $[0.2, 0.4]$, respectively, with nondeterministic precision $\rho = \{1000, 0.1\}$ using synthetic data containing 65 time points. These were obtained by simulating the model in MATLAB with $In_A = 6 \times 10^4$ and $\lambda = 0.28571$. Each i-th time point is, thus, defined by a vector

$$\{t_i, SC_i, TA_i, GA_i, SC_{d,i}, TA_{d,i}, GA_{d,i}\},$$

where t_i is the time value, and the rest of the vector are the measured values of the corresponding system variables.

The main aim of parameter set synthesis is finding parameter subsets for which the system dynamics satisfy the time series data. In other words, for each parameter value in the obtained subsets the value of the system dynamics must be equal to the given data values at the corresponding time points. However, this problem might not be feasible in general due to the strictness of the formulated requirements. Therefore, this experiment features a relaxed version of parameter set synthesis, where it is sufficient for the value of a system variable to be in a small interval around the measured value at the given time point. As a result, for each i-th time point the goal state is defined as (6.7).

$$\begin{aligned}
&\mathbf{goal}_i(SC, TA, GA, SC_d, TA_d, GA_d, t) := \Big((t = t_i) \wedge \\
&(SC \in [SC_i - 10, SC_i + 10]) \wedge (TA \in [TA_i - 10, TA_i + 10]) \wedge \\
&(GA \in [GA_i - 10, GA_i + 10]) \wedge (SC_d \in [SC_{d,i} - 10^2, SC_{d,i} + 10^2]) \wedge \\
&(TA_d \in [TA_{d,i} - 10^2, TA_{d,i} + 10^2]) \wedge (GA_d \in [GA_{d,i} - 10^2, GA_{d,i} + 10^2])\Big).
\end{aligned} \tag{6.7}$$

The computation took 195 minutes of CPU time, and `ProbReach` returned 276 parameter boxes (see Figure 6.13) satisfying the goal predicate for all 65 time points. From the obtained result it can be seen that the computed **sat** boxes contain the parameter values $In_A = 6 \times 10^4$ and $\lambda = 0.28571$ used for generating the time series data.

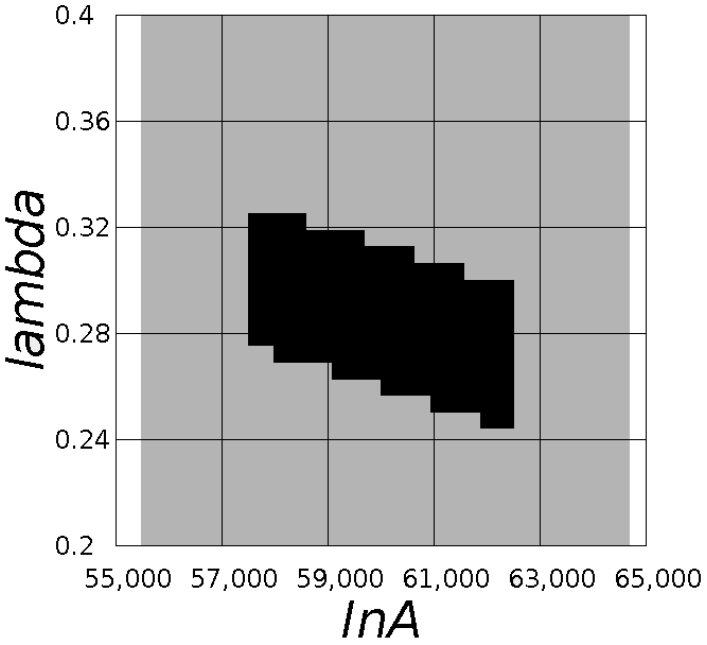


Figure 6.13: Parameter set synthesis result for the UVB irradiation model. The parameter values satisfying the time series data are highlighted with **black** colour, boxes for which the synthesis problem was undecidable for the last time point are highlighted with **grey**, and the **white** area represents the parameter values which do not satisfy the first time point.

6.5 Discussion

This chapter presented several case studies which were evaluated using `ProbReach`. It was demonstrated that all three types of precision parameters (*i.e.*, ϵ, ρ and η) can greatly affect the size of the computed probability enclosures.

The artificial pancreas and psoriasis treatment case studies suggest `ProbReach` can be applied to complex systems featuring nonlinear ODEs and nontrivial control.

Also, the implemented approaches (statistical and formal) showed consistent outputs. Their comparison confirmed that the statistical method is more efficient when the number of system parameters is large (*e.g.*, PID controller synthesis). However, sometimes it can fall into a local extremum. Besides, the formal approach demonstrated its applicability to parameter set synthesis, which cannot be performed using the statistical technique.

Chapter 7

Conclusions and Future work

7.1 Conclusions

In this thesis I presented my work on verification of stochastic parametric hybrid systems (SPHS). It contains both theoretical and implementation contributions, and demonstrates applicability of the devised methods and techniques to real-world case studies.

Chapter 2 features procedure **evaluate** for deciding bounded reachability on subsets of the parameter space of parametric hybrid systems. It is based on a δ-complete decision procedure and utilises the procedure output which is guaranteed to be correct (*i.e.*, the *unsat* answer). Given an arbitrary subset of the system parameter space **evaluate** *may* decide whether bounded reachability holds for all parameter values in the specified subset, for none of them or for some of them, returning **sat**, **unsat** and **undet**, respectively. However, while the answers are sound, the **evaluate** procedure cannot be complete. This can happen due to the insufficient precision δ or because the formulae generated by the **evaluate** procedure are non-robust for any positive δ (and deciding whether an arbitrary bounded $\mathcal{L}_{\mathbb{R}}$-sentence is robust is, of course, undecidable [28, 35]).

Also, Chapter 2 discusses the issues related to the implementation of δ-decision procedures. Namely, none of the existing SMT solvers (*e.g.*, `dReal`, `iSAT-ODE`) fully implement a δ-decision procedure, as currently they do not allow arbitrary combinations of existential and universal quantifiers (*i.e.*, only a single innermost universal

quantifier is allowed). As a result, I introduced procedure **compute** that can be implemented using existing solvers, and which **sat** and **unsat** answers are correct in the sense that they are consistent with **sat** and **unsat** outcomes of the **evaluate** procedure.

Finally, Chapter 2 shows that the two procedures are equivalent for PHSs with deterministic jumps – discrete transitions that can be enabled only once within the corresponding mode. The computational complexity of the presented algorithms grows exponentially with the number of quantified variables. Therefore, the main factors affecting the complexity are the number of system parameters and the reachability depth.

Chapter 3 presents an algorithm for computing ranges of the bounded reachability probability function in stochastic parametric hybrid systems (SPHS). Its main advantage is that it provides absolute numerical guarantees on the obtained results. It can also be applied to parameter set synthesis in SPHSs without random parameters (*i.e.*, general PHSs discussed in Chapter 2), which amounts to finding parameter subsets where the system reaches the goal state or a set of goal sets represented by time series data. Also, the size of the probability enclosures returned by this algorithm can sometimes be made arbitrarily small in some special cases. Namely, this can be done when the considered SPHS features at least one continuous random parameter, the reachability probability function is continuous (or constant), and the formulae generated by procedure **evaluate** are robust. The main disadvantage of the presented technique is that its computational complexity grows exponentially with the number of system parameters.

Chapter 4 features techniques solving the same problem and reducing the computational complexity at a cost of providing weaker (*i.e.*, statistical) guarantees. There are two algorithms (*i.e.*, Chernoff-Hoeffding Bound and Bayesian Estimation) for computing confidence intervals for the reachability probability function in systems featuring only random parameters. Systems featuring all types of parameters are handled using the Cross-Entropy algorithm (CE). It computes an approximation of the maximum (minimum) of the reachability probability function on the nondeterministic parameter space. Thus, the nondeterministic parameter value returned by the CE is not guaranteed to be the global optimum, however, the confidence interval returned for this value is guaranteed to contain the corresponding reachability probability value with the required confidence.

Chapter 5 introduces `ProbReach`, a tool for computing bounded reachability probability in SPHSs. It provides a C++ implementation for all of the algorithms presented in this thesis, and some of them (*i.e.*, Algorithms 3, 7, 8 and 9) were parallelised using OpenMP. Also, `ProbReach` currently supports two SMT solvers – `dReal` and `iSAT-ODE` – and it can be extended to supporting any SMT solver implementing a δ-complete decision procedure. `ProbReach` is publicly available and does not require any commercial software.

Finally, `ProbReach` was successfully applied to several complex case studies (as shown in Chapter 6) such as automated synthesis of safe PID controllers and devising UVB irradiation therapy for treating psoriasis.

7.2 Future Work

Future work can be undertaken in the following directions.

- Determining a more efficient way (perhaps one based on sensitivity analysis) of partitioning parameter boxes in Algorithm 3 from Chapter 3,
- Employing Quasi Monte Carlo techniques in the statistical algorithms presented in Chapter 4.
- Providing support for hybrid systems whose dynamics are defined by stochastic differential equations.
- Developing a more efficient parallelisation technique in order to decrease CPU idle.
- Providing support for more SMT solvers and performing static analysis of the input model in order to determine the most suitable solver for the task.
- Implementing a simulation engine in `ProbReach` for analysing the system's behaviour.
- Undertaking extensive benchmarking of `ProbReach` and applying it to several more complex case studies.

Appendix A

A.1 Supporting Claims

Lemma A.1. *The following implication holds:*

$$\Big[\exists x : A(x) \wedge \forall x, \exists y : A(x) \to B(x,y)\Big] \Rightarrow \Big[\exists x, \exists y : A(x) \wedge B(x,y)\Big] \tag{A.1}$$

Proof. By the assumption of implication (A.1) there is a point x such that $A(x)$ is true, and whenever $A(x)$ is true then there exists y such that $B(x,y)$ is also true. Suppose, $\forall x, \forall y : \neg A(x) \vee \neg B(x,y)$ which means that one of the following holds:

(a) $\forall x, \forall y : \neg A(x) \wedge \neg B(x,y)$,

(b) $\forall x, \forall y : \neg A(x) \wedge B(x,y)$,

(c) $\forall x, \forall y : A(x) \wedge \neg B(x,y)$.

It is easy to see that cases (a) and (b) contradict the assumption that $\exists x : A(x)$, and case (c) contradicts the assumption that $\forall x, \exists y : A(x) \to B(x,y)$. Therefore, (A.1) holds. □

Lemma A.2. *The following equivalence holds:*

$$\Big[\exists! x : A(x) \wedge \forall x, \exists y : A(x) \to B(x,y)\Big] \Leftrightarrow$$
$$\Big[\big(\exists x, \exists y : A(x) \wedge B(x,y)\big) \wedge \big(\exists x : A(x) \to \exists x! t : A(x)\big)\Big]$$

Proof. The following implication holds by Lemma A.1:

$$\Big[\exists! x : A(x) \wedge \forall x, \exists y : A(x) \to B(x,y)\Big] \Rightarrow$$
$$\Big[\big(\exists x, \exists y : A(x) \wedge B(x,y)\big) \wedge \big(\exists x : A(x) \to \exists! x : A(x)\big)\Big]$$

Now suppose that $\big(\exists x, \exists y : A(x) \wedge B(x,y)\big) \wedge \big(\exists x : A(x) \rightarrow \exists! x : A(x)\big)$ (which is equivalent to $\exists x, \exists y : A(x) \wedge B(x,y) \wedge \exists! x : A(x)$ as $\exists x, \exists y : A(x) \wedge B(x,y)$ means that $\exists x : A(x)$) is true but $\exists! x : A(x) \wedge \forall x, \exists y : A(x) \rightarrow B(x,y)$ does not hold. Thus, one of the three cases below must hold:

(a) $\neg\big(\exists! x : A(x)\big) \wedge \forall x, \exists y : A(x) \rightarrow B(x,y)$,

(b) $\neg\big(\exists! x : A(x)\big) \wedge \neg\big(\forall x, \exists y : A(x) \rightarrow B(x,y)\big)$,

(c) $\exists! x : A(x) \wedge \neg\big(\forall x, \exists y : A(x) \rightarrow B(x,y)\big)$.

Cases (a) and (b) contradict the assumption that $\exists! x : A(x)$. Consider case (c) now:

$$\exists! x : A(x) \wedge \neg\big(\forall x, \exists y : A(x) \rightarrow B(x,y)\big) \Leftrightarrow \exists! x : A(x) \wedge \big(\exists x, \forall y : A(x) \wedge \neg B(x,y)\big).$$

Let z be the only value for which $A(z)$ evaluates to true. Then the above formula can be rewritten as $A(z) \wedge \forall y : \neg B(z,y)$. However, this contradicts the assumption that $\exists x, \exists y : A(x) \wedge B(x,y) \wedge \exists! x : A(x)$ (which is equivalent to $A(z) \wedge \exists y : B(z,y)$). Therefore,

$$\Big[\exists x, \exists y : A(x) \wedge B(x,y) \wedge \big(\exists x : A(x) \rightarrow \exists! x : A(x)\big)\Big] \Rightarrow$$
$$\Big[\exists! x : A(x) \wedge \forall x, \exists y : A(x) \rightarrow B(x,y)\Big]$$

□

A.2 Definitions

Definition A.1. (Computable Real Number [9, Definition 3.1]) *A real number x is computable if there exists a computable sequence of rational numbers $(q_n)_{n\in\mathbb{N}}$ that converges to x (*i.e., *$\forall i : |x - q_i| < 2^{-i}$).*

Definition A.2. (Computable Real Function [9, Definition 4.1]) *A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is computable if there is an oracle Turing machine M that, given any precision $k \in \mathbb{N}$ and input $x \in dom(f)$, quires another procedure for an arbitrarily good rational approximation q_i of x satisfying $|x - q_i| < 2^{-i}$, and produces a rational number $M(q_i)$ such that $|f(x) - M(q_i)| < 2^{-k}$.*

Definition A.3. (σ-algebra [17]) *A σ-algebra on some arbitrary set X is a collection $\mathcal{A}$ of subsets of X such that: $\mathcal{A}$ contains X, a complement of any set from $\mathcal{A}$ belongs to $\mathcal{A}$, and sets $\bigcup_{i=1}^{\infty} A_i$ and $\bigcap_{i=1}^{\infty} A_i$ (where each $A_i \in \mathcal{A}$) are also in $\mathcal{A}$.*

Definition A.4. (Measurable Function [17]) *Let $(X, \mathcal{A})$ be a measurable space (*i.e., *$\mathcal{A}$ is a σ-algebra on set X) and A be a subset of X that belongs to $\mathcal{A}$. A function $f : A \to \mathbb{R}$ is measurable (with respect to $\mathcal{A}$) if for any $c \in \mathbb{R}$ the set $\{x \in A : f(x) \leq c\}$ belongs to $\mathcal{A}$.*

Definition A.5. (Analytic Set [17]) *A set $X \subseteq \mathbb{R}^n$ is analytic if there is a set $Y \subseteq \mathbb{R}^n$ and a continuous function $f : Y \to X$ such that $f(Y) = X$.*

References

[1] Alessandro Abate, Joost-Pieter Katoen, John Lygeros, and Maria Prandini. Approximate model checking of stochastic hybrid systems. *European Journal of Control*, **16**[6]:624 – 641, 2010. 7

[2] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, **44**[11]:2724–2734, November 2008. 1

[3] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, **736** of *LNCS*, pages 209–229, 1992. 2, 4

[4] Rajeev Alur, Thomas A. Henzinger, and Pei-Hsin Ho. Automatic symbolic verification of embedded systems. *IEEE Trans. Software Eng.*, **22**[3]:181–201, 1996. 3

[5] Christel Baier and Joost P. Katoen. *Principles of Model Checking*. The MIT Press, 2008. 3

[6] Clark W Barrett, Roberto Sebastiani, Sanjit A Seshia, and Cesare Tinelli. Satisfiability modulo theories. *Handbook of satisfiability*, **185**:825–885, 2009. 28

[7] Garrett Birkhoff and Gian-Carlo Rota. *Ordinary Differential Equations, 4th Edition*. Wiley, January 1989. 18

[8] Sergiy Bogomolov, Daniele Magazzeni, Stefano Minopoli, and Martin Wehrle. PDDL+ planning with hybrid automata: Foundations of translating must behavior. In *ICAPS*, pages 42–46, 2015. 3

[9] Vasco Brattka, Peter Hertling, and Klaus Weihrauch. A tutorial on computable analysis. In S. Barry Cooper, Benedikt Löwe, and Andrea Sorbi, editors, *New Computational Paradigms*, pages 425–491. Springer New York, 2008. 130

[10] Daniel Bryce, Sicun Gao, David J. Musliner, and Robert P. Goldman. SMT-based nonlinear PDDL+ planning. In *AAAI*, pages 3247–3253, 2015. 14

[11] Russel E. Caflisch. Monte carlo and quasi-monte carlo methods. *Acta Numerica*, **7**:1–49, 1998. 73

[12] Edmund Clarke, Orna Grumberg, and Doron A. Peled. *Model Checking*. The MIT Press, 1999. 3

[13] Edmund M. Clarke, Armin Biere, Richard Raimi, and Yunshan Zhu. Bounded model checking using satisfiability solving. *Formal Methods in System Design*, **19**[1]:7–34, 2001. 4

[14] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In Dexter Kozen, editor, *Logic of Programs*, **131** of *LNCS*, pages 52–71, 1981. 3

[15] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT Press, 2001. 4

[16] Edmund M. Clarke, Kenneth L. McMillan, Sérgio Vale Aguiar Campos, and Vassili Hartonas-Garmhausen. Symbolic model checking. In Rajeev Alur and Thomas A. Henzinger, editors, *CAV*, **1102** of *Lecture Notes in Computer Science*, pages 419–427. Springer, 1996. 3

[17] Donald L. Cohn. *Measure Theory*. Birkhäuser, 1980. 34, 131

[18] Richard Colgren. *Basic Matlab, Simulink And Stateflow*. AIAA (American Institute of Aeronautics & Ast, 2006. 90

[19] J. Estrela da Silva, Bruno Terra, Ricardo Martins, and João Borges de Sousa. Modeling and simulation of the lauv autonomous underwater vehicle. In *13th IEEE IFAC International Conference on Methods and Models in Automation and Robotics*, 2007. 1

[20] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikučionis, and Danny Bøgsted Poulsen. Uppaal SMC tutorial. *International Journal on Software Tools for Technology Transfer (STTT)*, **17**[4]:397–415, 2015. 9

[21] Martin Davis, Hilary Putnam, and Julia Robinson. The decision problem for exponential diophantine equations. *Annals of Mathematics*, **74**[3]:pp. 425–436, 1961. 4

[22] Andreas Eggers, Martin Fränzle, and Christian Herde. SAT modulo ODE: A direct SAT approach to hybrid systems. In *ATVA*, pages 171–185, 2008. 4, 87

[23] Andreas Eggers, Nacim Ramdani, Nedialko S. Nedialkov, and Martin Fränzle. Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. *Software & Systems Modeling*, **14**[1]:121–148, 2015. 6, 13

[24] Christian Ellen, Sebastian Gerwinn, and Martin Fränzle. Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains. *International Journal on Software Tools for Technology Transfer (STTT)*, **17**[4]:485–504, 2015. 7

[25] Joshua A. Enszer and Mark A. Stadtherr. Verified solution and propagation of uncertainty in physiological models. *Reliable Computing*, **15**:168–178, 2010. 7

[26] Chuchu Fan, Bolun Qi, Sayan Mitra, Mahesh Viswanathan, and Parasara Sridhar Duggirala. *Automatic Reachability Analysis for Nonlinear Hybrid Models with C2E2*, pages 531–538. Springer International Publishing, Cham, 2016. 9

[27] Maria Fox, Derek Long, and Daniele Magazzeni. Plan-based policies for efficient multiple battery load management. *J. Artif. Intell. Res. (JAIR)*, **44**:335–382, 2012. 1

[28] Martin Fränzle. *Analysis of Hybrid Systems: An Ounce of Realism Can Save an Infinity of States*, pages 126–139. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. 4, 125

[29] Martin Fränzle, Ernst Moritz Hahn, Holger Hermanns, Nicolás Wolovick, and Lijun Zhang. Measurability and safety verification for stochastic hybrid systems. In *HSCC*, pages 43–52, 2011. 9

[30] Martin Fränzle, Christian Herde, Tino Teige, Stefan Ratschan, and Tobias Schubert. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. *JSAT*, **1**[3-4]:209–236, 2007. 4

[31] Martin Fränzle, Tino Teige, and Andreas Eggers. Engineering constraint solvers for automatic analysis of probabilistic hybrid automata. *J. Log. Algebr. Program.*, **79**[7]:436–466, 2010. 8

[32] Mark Galassi, Jim Davies, James Theiler, Brian Gough, and Gerard Jungman. *GNU Scientific Library - Reference Manual, Third Edition, for GSL Version 1.12 (3. ed.)*. Network Theory Ltd, 2009. 86

[33] Victor Gan, Guy Albert Dumont, and Ian M. Mitchell. Benchmark problem: A pk/pd model and safety constraints for anesthesia delivery. In *ARCH@CPSWeek*, 2014. 97

[34] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. Delta-complete decision procedures for satisfiability over the reals. In *IJCAR*, pages 286–300, 2012. 28, 53

[35] Sicun Gao, Jeremy Avigad, and Edmund M. Clarke. Delta-decidability over the reals. In *LICS*, pages 305–314, 2012. 4, 5, 6, 13, 28, 54, 125

[36] Sicun Gao, Soonho Kong, and Edmund M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *CADE-24*, **7898** of *LNCS*, pages 208–214, 2013. 6, 87

[37] Sicun Gao, Soonho Kong, and Edmund M. Clarke. Satisfiability modulo ODEs. In *FMCAD*, pages 105–112, 2013. 4

[38] Yang Gao and Martin Fränzle. *A Solving Procedure for Stochastic Satisfiability Modulo Theories with Continuous Domain*, pages 295–311. Springer International Publishing, Cham, 2015. 6

[39] Yang Gao, Martin Fränzle, undefined, undefined, undefined, and undefined. CSiSAT: A satisfiability solver for SMT formulas with continuous probability distributions. **00**, pages 1–6. IEEE Computer Society, 2016. 8

[40] Pedro Gonnet. A review of error estimation in adaptive quadrature. *ACM Comput. Surv.*, **44**[4]:22:1–22:36, 2012. 46

[41] Orna Grumberg and Helmut Veith, editors. *25 Years of Model Checking - History, Achievements, Perspectives*, **5000** of *Lecture Notes in Computer Science*. Springer, 2008. 3

[42] Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. Approximate probabilistic model checking. In *VMCAI*, **2937** of *LNCS*, pages 73–84, 2004. 66

[43] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.*, **58**[301]:13–30, 1963. 65

[44] Roman Hovorka. Closed-loop insulin delivery: from bench to clinical practice. *Nature Reviews Endocrinology*, **7**[7]:385–395, 2011. 1, 108

[45] Roman Hovorka et al. Nonlinear model predictive control of glucose concentration in subjects with type 1 diabetes. *Physiological Measurement*, **25**[4]:905, 2004. 109

[46] Lauren M Huyett, Eyal Dassau, Howard C Zisser, and Francis J Doyle III. Design and evaluation of a robust pid controller for a fully implantable artificial pancreas. *Industrial & engineering chemistry research*, **54**[42]:10311–10321, 2015. 109

[47] Daisuke Ishii, Kazunori Ueda, and Hiroshi Hosobe. An interval-based sat modulo ode solver for model checking nonlinear hybrid systems. *STTT*, **13**[5]:449–461, 2011. 4

[48] Matthew James. Beal. Variational algorithms for approximate bayesian inference /. 01 2003. 80

[49] Xiaoqing Jin, Jyotirmoy V. Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. Powertrain control verification benchmark. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, pages 253–262, 2014. 1

[50] Ker-I. Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991. 5, 18

[51] Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. dReach: Delta-reachability analysis for hybrid systems. In *TACAS*, 2015. *to appear*. 7, 13, 14, 83

[52] Guy F. Kuncir. Algorithm 103: Simpson's rule integrator. *Commun. ACM*, **5**[6]:347–, 1962. 46

[53] Marta Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, **6806** of *LNCS*, pages 585–591, 2011. 9

[54] Miklós Laczkovich. The removal of π from some undecidable problems involving elementary functions. *Proceedings of the American Mathematical Society*, **131**[7]:pp. 2235–2240, 2003. 4

[55] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, **1**:134–152, 1997. 9

[56] Pierre L'Ecuyer. Quasi-Monte Carlo methods with applications in finance. *Finance and Stochastics*, **13**[3]:307–349, 2009. 73

[57] John Levine. *Flex & Bison: Text Processing Tools*. O'Reilly Media, 2009. 85

[58] Xiangfang Li, Oluwaseyi Omotere, Lijun Qian, and Edward R. Dougherty. Review of stochastic hybrid systems with applications in biological systems modeling and analysis. *EURASIP Journal on Bioinformatics and Systems Biology*, **2017**[1]:8, Jun 2017. 1, 3

[59] Yun Li, Kiam Heong Ang, Gregory CY Chong, Wenyuan Feng, Kay Chen Tan, and Hiroshi Kashiwagi. Cautocsd–evolutionary search and optimisation enabled computer automated control system design. *International Journal of Automation and Computing*, **1**[1]:76–88, 2004. 109

[60] Bing Liu, Soonho Kong, Sicun Gao, and Edmund Clarke. Parameter identification using delta-decisions for biological hybrid systems. CMU SCS Technical Report, CMU-CS-13-136, 2014. 1, 8

[61] Yuri V. Matijasevič. Diophantine representation of enumerable predicates. *Mathematics of the USSR-Izvestiya*, **5**[1]:1, 1971. 4

[62] Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, Norwell, MA, USA, 1993. 3

[63] James R. Munkres. *Topology*. Featured Titles for Topology Series. Prentice Hall, 2000. 52

[64] Erich Novak and Henryk Woźniakowski. Relaxed verification for continuous problems. *Journal of Complexity*, **8**[2]:124 – 152, 1992. 4

[65] Frank W. Olver, Daniel W. Lozier, Ronald F. Boisvert, and Charles W. Clark. *NIST Handbook of Mathematical Functions*. Cambridge University Press, 1st edition, 2010. 68

[66] OpenMP Architecture Review Board. OpenMP application program interface version 3.0, may 2008. 87

[67] Jean-Pierre Queille and Joseph Sifakis. Specification and verification of concurrent systems in cesar. In *Proceedings of the 5th Colloquium on International Symposium on Programming*, pages 337–351, London, UK, UK, 1982. Springer-Verlag. 3

[68] Federico Ramponi, Debasish Chatterjee, Sean Summers, and John Lygeros. On the connections between PCTL and dynamic programming. In *HSCC*, pages 253–262. ACM, 2010. 7

[69] Stefan Ratschan. Safety verification of non-linear hybrid systems is quasi-decidable. *Formal Methods in System Design*, **44**[1]:71–90, 2014. 5

[70] Daniel Richardson. Some undecidable problems involving elementary functions of a real variable. *The Journal of Symbolic Logic*, **33**[4]:pp. 514–520, 1968. 4

[71] Christian P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer Texts in Statistics. Springer, 2001. 69

[72] Reuven Y. Rubinstein and Dirk Kroese. *Simulation and the Monte Carlo Method*. Wiley, 2008. 73, 78

[73] Walter Rudin. *Principles of Mathematical Analysis*. McGraw-Hill, 1976. 53

[74] Karsten Scheibler, Felix Neubauer, Ahmed Mahdi, Martin Fränzle, Tino Teige, Tom Bienmüller, Detlef Fehrer, and Bernd Becker. Accurate ICP-based floating-point reasoning. In *2016 Formal Methods in Computer-Aided Design (FMCAD)*, pages 177–184, Oct 2016. 8

[75] Fedor Shmarov, Nicola Paoletti, Ezio Bartocci, Shan Lin, Scott A. Smolka, and Paolo Zuliani. Smt-based synthesis of safe and robust PID controllers for stochastic hybrid systems. In *Hardware and Software: Verification and Testing - 13th International Haifa Verification Conference, HVC 2017, Haifa, Israel, November 13-15, 2017, Proceedings*, pages 131–146, 2017. 10

[76] Fedor Shmarov and Paolo Zuliani. ProbReach: A tool for guaranteed reachability analysis of stochastic hybrid systems. In Sergiy Bogomolov and Ashish Tiwari, editors, *Symbolic and Numerical Methods for Reachability Analysis, 1st International Workshop, SNR 2015*, **37** of *EPiC Series in Computing*, pages 40–48, 2015. 10

[77] Fedor Shmarov and Paolo Zuliani. ProbReach: Verified probabilistic δ-reachability for stochastic hybrid systems. In *HSCC*, pages 134–139. ACM, 2015. 10, 83

[78] Fedor Shmarov and Paolo Zuliani. Probabilistic hybrid systems verification via smt and monte carlo techniques. In *Hardware and Software: Verification and Testing: 12th International Haifa Verification Conference, HVC 2016, Haifa, Israel, November 14-17, 2016, Proceedings*, pages 152–168, Cham, 2016. Springer International Publishing. 3, 10, 21, 63

[79] Fedor Shmarov and Paolo Zuliani. SMT-based reasoning for uncertain hybrid domains. In *AAAI-16 Workhop on Planning for Hybrid Systems, 30th AAAI Conference on Artificial Intelligence*, pages 624–630, 2016. 10

[80] Steven S. Skiena. *Sorting and Searching*, pages 103–144. Springer London, London, 2008. 17

[81] Sadegh Esmaeil Zadeh Soudjani, C. Gevaerts, and Alessandro Abate. FAUST^2: Formal abstractions of uncountable-state stochastic processes. In *TACAS*, **9035** of *LNCS*, pages 272–286, 2015. 9

[82] Garry M Steil, Cesar C Palerm, Natalie Kurtz, Gayane Voskanyan, Anirban Roy, Sachiko Paz, and Fouad R Kandeel. The effect of insulin feedback on closed loop glucose control. *The Journal of Clinical Endocrinology & Metabolism*, **96**[5]:1402–1408, 2011. 109

[83] Garry M. Steil, Antonios E. Panteleon, and Kerstin Rebrin. Closed-loop insulin delivery – the path to physiological glucose control. *Advanced drug delivery reviews*, **56**[2]:125–144, 2004. 109

[84] Alfred Tarski. A decision method for elementary algebra and geometry. 1948. 4

[85] Tino Teige and Martin Fränzle. *Stochastic Satisfiability Modulo Theories for Non-linear Arithmetic*, pages 248–262. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. 6

[86] Ilya Tkachev and Alessandro Abate. Formula-free finite abstractions for linear temporal verification of stochastic hybrid systems. In *HSCC*, pages 283–292. ACM, 2013. 7

[87] P. Wang, D.A. Barajas-Solano, E.M. Constantinescu, S. Abhyankar, D. Ghosh, B.F. Smith, Z. Huang, and A.M. Tartakovsky. Probabilistic density function method for stochastic odes of power systems with uncertain power input. *SIAM/ASA Journal on Uncertainty Quantification*, **3**:24, 2015. 1

[88] Paul S. Wang. The undecidability of the existence of zeros of real elementary functions. *J. ACM*, **21**[4]:586–589, October 1974. 4

[89] Qinsi Wang, Paolo Zuliani, Soonho Kong, Sicun Gao, and Edmund M. Clarke. SReach: A bounded model checker for stochastic hybrid systems. In *CMSB*, **9308** of *LNCS*, pages 15–27, 2015. 3, 8

[90] Boyan Yordanov and Calin Belta. Parameter synthesis for piecewise affine systems from temporal logic specifications. In *HSCC 2008*, **4981** of *LNCS*, pages 542–555, 2008. 3

[91] Håkan L. S. Younes, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Numerical vs. statistical probabilistic model checking. *STTT*, **8**[3]:216–228, 2006. 6, 63

[92] Håkan L. S. Younes and Reid G. Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Inf. Comput.*, **204**[9]:1368–1409, 2006. 6, 63

[93] Hong Zhang, Wenhong Hou, Laurence Henrot, Sylvianne Schnebert, Marc Dumas, Catherine Heusèle, and Jin Yang. Modelling epidermis homoeostasis and psoriasis pathogenesis. *Journal of The Royal Society Interface*, **12**[103], 2015. 117, 119

[94] Lijun Zhang, Zhikun She, Stefan Ratschan, Holger Hermanns, and Ernst Moritz Hahn. Safety verification for probabilistic hybrid systems. In *CAV*, **6174** of *LNCS*, pages 196–211, 2010. 9

[95] Paolo Zuliani, André Platzer, and Edmund M. Clarke. Bayesian statistical model checking with application to Stateflow/Simulink verification. *Formal Methods in System Design*, **43**[2]:338–367, 2013. 68, 71