Model Checking Stochastic Hybrid Systems

Dissertation

Zur Erlangung des Grades des
Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

vorgelegt von
Ernst Moritz Hahn

Saarbrücken
Oktober 2012

Tag des Kolloquiums:	21.12.2012
Dekan:	Prof. Dr. Mark Groves
Prüfungsausschuss:	
Vorsitzende:	Prof. Dr. Verena Wolf
Gutachter:	Prof. Dr.-Ing. Holger Hermanns Prof. Dr. Martin Fränzle Prof. Dr. Marta Kwiatkowska
Akademischer Mitarbeiter:	Dr. Andrea Turrini

Abstract

The interplay of random phenomena with discrete-continuous dynamics deserves increased attention in many systems of growing importance. Their verification needs to consider both *stochastic* behaviour and *hybrid* dynamics. In the verification of classical hybrid systems, one is often interested in deciding whether unsafe system states can be reached. In the stochastic setting, we ask instead whether the probability of reaching particular states is bounded by a given threshold. In this thesis, we consider stochastic hybrid systems and develop a general abstraction framework for deciding such problems. This gives rise to the first mechanisable technique that can, in practice, formally verify safety properties of systems which feature all the relevant aspects of nondeterminism, general continuous-time dynamics, and probabilistic behaviour. Being based on tools for classical hybrid systems, future improvements in the effectiveness of such tools directly carry over to improvements in the effectiveness of our technique.

We extend the method in several directions. Firstly, we discuss how we can handle *continuous* probability distributions. We then consider systems which we are in partial *control* of. Next, we consider systems in which probabilities are *parametric*, to analyse entire system families at once. Afterwards, we consider systems equipped with *rewards*, modelling costs or bonuses. Finally, we consider all orthogonal combinations of the extensions to the core model.

Zusammenfassung

In vielen Systemen wachsender Bedeutung tritt zufallsabhängiges Verhalten gleichzeitig mit diskret-kontinuierlicher Dynamik auf. Um solche Systeme zu verifizieren, müssen sowohl ihr *stochastisches* Verhalten als auch ihre *hybride* Dynamik betrachtet werden. In der Analyse klassischer hybrider Systeme ist eine wichtige Frage, ob unsichere Zustände erreicht werden können. Im stochastischen Fall fragen wir stattdessen nach garantierten Wahrscheinlichkeitsschranken. In dieser Arbeit betrachten wir stochastische hybride Systeme und entwickeln eine allgemeine Abstraktionsmethode um Probleme dieser Art zu entscheiden. Dies ermöglicht die erste automatische und praktisch anwendbare Methode, die Sicherheitseigenschaften von Systeme beweisen kann, in denen Nichtdeterminismus, komplexe Dynamik und probabilistisches Verhalten gleichzeitig auftreten. Da die Methode auf Analysetechniken für nichtstochastische hybride Systeme beruht, profitieren wir sofort von zukünftigen Verbesserungen dieser Verfahren.

Wir erweitern diese Grundmethode in mehrere Richtungen: Zunächst ergänzen wir das Modell um *kontinuierliche* Wahrscheinlichkeitsverteilungen. Dann betrachten wir *partiell kontrollierbare* Systeme. Als nächstes untersuchen wir *parametrische* Systeme, um eine Klasse ähnlicher Modelle gleichzeitig behandeln. Anschließend betrachten wir Eigenschaften, die auf der Abwägung von *Kosten und Nutzen* beruhen. Schließlich zeigen wir, wie diese Erweiterungen orthogonal kombiniert werden können.

Acknowledgements

I would like to begin by thanking my advisor, Holger Hermanns. He shaped my understanding of research and his stewardship has guided me through this dissertation. Furthermore, I would like to thank my coauthors and my colleagues at the Dependable Systems Group for providing a friendly and supportive environment. Especially, I would like to thank Lijun Zhang for his support, in particular at the initial phase of my thesis. I would like to thank Luis María Ferrer Fioriti for the time he devoted to reading the thesis and his valuable feedback.

This thesis was supported by the Saarbrücken Graduate School of Computer Science, the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme", the exchange program DAAD-MINCyT: Programm zum Projektbezogenen Personenaustauschs PROALAR, the German Research Foundation (DFG) as part of the transregional research center SFB/TR 14 AVACS, the European Community's Seventh Framework Programme under grant agreement n^o 214755 QUASIMODO, the European Union Seventh Framework Programme under grant agreement number 295261 as part of the MEALS project, and the NWO-DFG bilateral project ROCKS.

Contents

1 Introduction

Hybrid systems constitute a general and widely applicable class of dynamical systems with both discrete and continuous components. Classical hybrid system formalisms [Alu+95; Pre+98; Cla+03; RS07] capture many characteristics of real systems. However, in many modern application areas, also *stochastic dynamics* occurs. This is especially true for wireless sensing and control applications, where message loss probabilities and other random effects (node placement, node failure, battery drain) turn the overall control problem into a problem that can only be managed with a certain, hopefully sufficiently high, probability. Due to the influence of these random effects, the success (for example keeping the hybrid system in a safe region) can only be guaranteed with a probability less than one. Since these phenomena are important aspects when aiming at faithful models for networked and embedded applications, the interest in answering such questions and determining the probabilities is growing [FHT08; TF09].

The need to integrate probabilities into hybrid system formalisms has led to a number of different notions of *stochastic hybrid automata*, each from a distinct perspective [AG97; Spr00; Spr01; Buj04; BLB05; Aba+08]. The most important distinction lies in the point where to introduce randomness. One option is to replace deterministic jumps, specified by *guarded commands*, by probability distributions over deterministic jumps. Here, one can distinguish between different kinds of probability distributions: distributions with *finite support* only feature a random choice between a finite number of possible successor states. *Continuous* distributions instead choose over an uncountably large set of possible successors. Being able to use this kind of distributions increases the expressiveness of models. For instance, the *normal distribution* is of interest, in particular because it allows to model measurement processes: here, the mean of the distribution is used to model the actual value of a quantity to be measured. The values which are indeed measured appear as samples from the distribution and are therefore more or less skewed, depending on the variance of the distribution. Another option is to generalise the differential equation components inside a mode by a stochastic differential equations component. More general models can be obtained by blending the above two choices, and by combining them with memoryless timed stochastic jumps [Dav84; BL06a]. Additionally, *nondeterminism*, can be integrated, to model phenomena to which one is unable to assign a probability to.

An important problem in hybrid systems theory is that of *reachability analysis*. In general terms, solving a reachability analysis problem means evaluating whether a given system may reach certain unsafe states, starting from some given initial states. This problem is associated with the safety verification problem where one intends to prove that the system can never reach any unsafe state. On the other hand, one might desire to prove that certain desirable states will always be reached. In the stochastic setting, the corresponding problems are to check whether the probability that the system trajectories reach a certain set of states from the initial state can be bounded from above or from below by a given threshold $\mathfrak{p}$.

It is often the case that some aspects of a hybrid system's nondeterminism are *controllable*, while other parts depend on its environment and cannot be assumed to follow known random behaviour [LGS95; Beh+07]. In addition, there might be aspects of a system of which stochastic aspects are known. Because of this, such a hybrid system can be seen as combining stochastic behaviour with two kinds of nondeterminism. It can thus be regarded as a game in which the controller tries to enforce certain guarantees on the system's behaviour, regardless of the behaviour of the environment.

In many cases, an entire class of models is of interest. Such a class can be represented by a model in which certain *parameters* are left open [And+09; DISS11; FK11]. These parameters might describe, for instance, the dynamics of the system or the stochastic behaviour. One is then interested in reasoning about parameter instantiations, representing specific models of the given class, which guarantee a certain behaviour, or finding optimal parameter values subject to a given optimality function.

Reachability is an important property of stochastic hybrid systems, but other aspects of a system might be of interest as well. For instance, in addition to proving safety or termination guarantees, one might want to analyse properties which are related to resource consumption (energy, memory, bandwidth, etc.) or lie more on the economical side (monetary gain, the expected time or cost until termination, etc.) which cannot be expressed as reachability problems. Those properties can be described using *rewards*. A reward structure attaches numeric values to discrete-time transitions of a system, or to the passage of time. Depending on the context, rewards can then be interpreted as costs or bonuses. The values of interest can be formalised, for instance, as the expected *total reward* or the expected *long-run average reward* [Put94; Alf97; EJ11].

1.1 Contribution

Stochastic hybrid systems have found various applications in diverse areas [Aba+08, Section 3][PH09] [MW12] and tool support for several subclasses of such systems exists [Kwi+02][Aba+08, Section 5][TF09, Section 4]. Thus far, it was however not possible to faithfully handle models which feature both probabilistic behaviour and general continuous-time dynamics as well as nondeterminism. In this thesis, we develop the first generic framework which does not require manual intervention to decide properties of system models which indeed contain all these features. As we build our solution methods

on top of the rapidly developing solutions methods for classical, nonstochastic hybrid automata, improvements in this field immediately carry over to our setting.

In the following we give an overview of our achievements.

Probabilistic Reachability

One of the most important problems in hybrid systems and safety critical applications in general is that of *reachability*, that is the problem of deciding whether an unsafe system state can be reached from the initial state. Another, similarly relevant, problem is to decide whether certain desirable states will *inevitably* be reached. In the setting of stochastic hybrid systems, the presence of probabilistic decisions in addition to the nondeterministic ones turns this problem from a qualitative into a quantitative one: in most systems involving stochastic behaviour, it will neither be the case that such states will always be reached, nor that they will never be reached. Instead of asking *whether* unsafe or desirable states might be reached, we can only ask for *probabilities* with which this might happen. Still, there is not a single probability answering these queries, because the stochastic behaviour depends on the different possible resolutions of the nondeterminism of the stochastic hybrid system. It is thus natural to be interested in proving that the probability to reach a set of unsafe states can never exceed a given probability threshold $\mathfrak{p}_{\text{upper}}$, and that the probability to reach desirable states is never below $\mathfrak{p}_{\text{lower}}$.

To solve this principle problem, we aim at computing probability bounds on reachability properties of a relevant class of stochastic hybrid systems. If the bound computed turns out to be below (or above respectively) the threshold $\mathfrak{p}_{\text{upper}}$ ($\mathfrak{p}_{\text{lower}}$), this proves the safety of the stochastic hybrid system. The class of models we consider, *probabilistic hybrid automata*, is an extension of classical hybrid automata, in which the guarded commands may choose their successors according to a probability distribution with finite support [Spr00; Spr01]. This model thus maintains the expressive power of the dynamics of general classical hybrid automata, while adding probabilistic choices in a well-defined way. To compute probability bounds, we compute finite abstractions of the original models and then solve reachability problems there. This gives rise to an effective mechanisable method to decide a large class of properties of this generic class of probabilistic hybrid automata.

We build our abstractions on solvers for the reachability problem in classical hybrid automata, for which strong tool support exists (e.g. [HHWT97b; HHWT97a; RS07; Fre08; Fre+11]). Basically, we convert a probabilistic hybrid automaton into a classical hybrid automaton, where distributions are replaced by labels. Then, we use one of the solvers to obtain an abstraction in form of a labelled transition system [Kel76], which does not yet involve stochastic behaviour. From this abstraction, we can reconstruct a probabilistic automaton [SL95], which is a safe overapproximation of the original probabilistic hybrid automaton. The approach we use has the advantage of orthogonality: future computational advances in the analysis of hybrid automata directly carry over to improvements of our framework.

Extensions

We develop several extensions of the basic analysis method:

- we extend our framework to *continuous distributions*. For this, we have to use a different semantical model and solve intricate well-definedness issues. We also have to extend our analysis framework, thereby faithfully abstracting continuous distributions to finitely-probabilistic ones.
- We consider systems with *partial control*. To do so, we discuss a different interpretation of the model semantics. We then show how we can extend our abstraction framework to apply existing algorithms for the synthesis of controllers, which can steer a system so as to guarantee certain properties.
- We consider *parametric* system models. We firstly adapt our hybrid automata model, its semantics and the abstraction method to the parametric case. We then develop algorithms which work on abstractions of parametric models to obtain probability bounds and answer the validity of properties.
- We integrate *rewards* into our probabilistic hybrid automata. Then, we discuss how we can handle them in our abstraction framework, in particular how we can compute abstract reward structures. We then discuss how to apply existing algorithms to decide reward-based properties in the abstraction.

We also discuss the orthogonal combination of all these extensions.

We implement our method in a tool called ProHVer. To demonstrate the practical applicability of our method, we apply it on a number of case studies, which are diverse in the nature of their behaviour and thus serve as trustworthy benchmarks of our technique.

1.2 Layout and Origin of the Thesis

We now describe the structure of the thesis and its origins in terms of previous publications by the author. A sketch of the structure is given in Figure 1.1. The arrows between the chapters describe dependencies, so that for example Chapter 3 depends on Chapter 2, and chapters 4, 5, 6 and 7 can be read independently of each other.

In Chapter 1, we have discussed the context and contribution of the thesis.

Chapter 2 describes the basic classical hybrid automata formalism which we build on in later chapters. We discuss how we can prove that certain states are never reached, or are guaranteed to be reached eventually.

In Chapter 3, we extend the model of classical hybrid automata to probabilistic hybrid automata [Spr00] with distributions of finite support. Building on results of classical hybrid automata, we show how we can prove bounds on reachability probabilities. This chapter mainly builds on two previous publications [Zha+10; Zha+11]. The main contribution of the author of this thesis to these previous publications are the implementation of the analysis technique and the selection and preparation of most of the case studies.

Chapter 4 considers stochastic hybrid automata which feature continuous probability

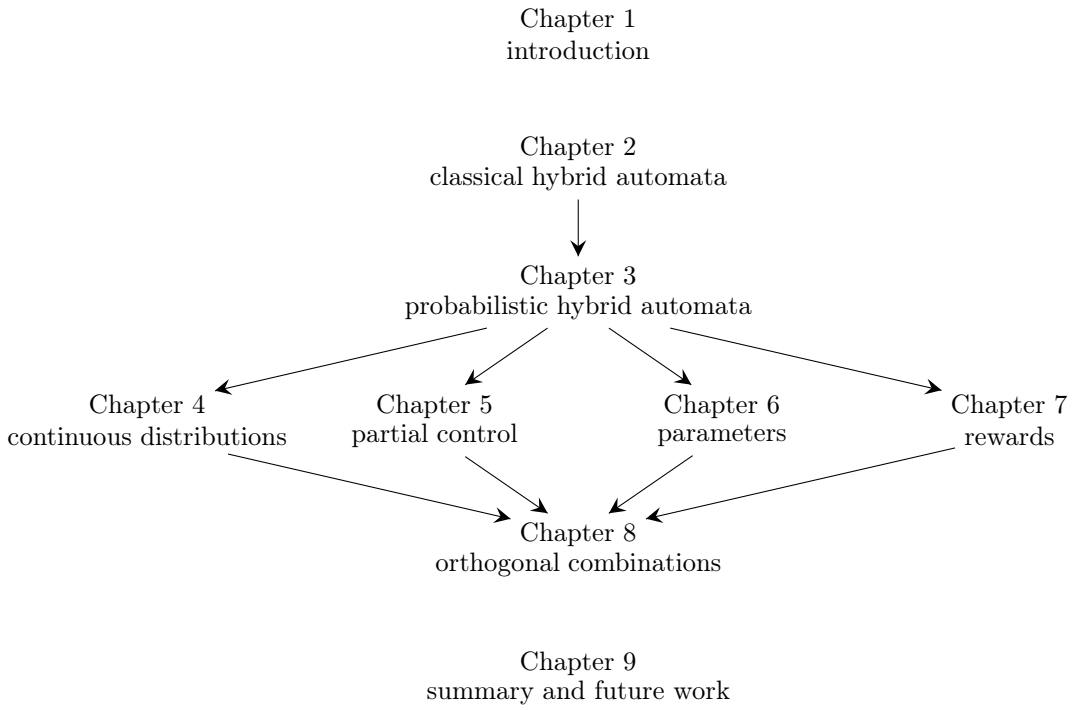


Figure 1.1: Structure of the thesis.

distributions. We show how to bound values for this extended case of stochastic models, and how to solve well-definedness problems. The chapter expands results of [Frä+11].

We extend our methods thus to steer stochastic hybrid systems of which we have partial control in Chapter 5. This chapter is rooted in [Hah+11].

In Chapter 6, we consider probabilistic hybrid automata with parametric probabilities. This allows us to quickly obtain an overview of the behaviour of a family of stochastic hybrid systems much faster, than if we had to do a separate analysis for each of them. It also allows to estimate optimal parameters of such systems. The chapter is based on methods originating from [HHZ09; Hah+10; HHZ11a; HHZ11b].

Chapter 7 describes how we can extend our analysis to reward-based properties, rather than just probabilistic reachability. We extend our abstraction framework thus to obtain upper and lower bounds of these property classes. The chapter also builds on [Hah+11], but has been extended considerably.

In Chapter 8, we describe orthogonal combinations of the previous chapters.

Finally, in Chapter 9, we summarise the thesis and discuss promising attack points for future work.

2

Classical Hybrid Automata

In this chapter, we formalise the basic notions of *classical hybrid automata* without stochastic behaviours. This chapter does not contain new results, but reformulates existing ones from the perspective of this thesis. It is important anyway, because it will build the foundation of the later extensions to stochastic models: analysis techniques for our stochastic models will build on classical hybrid automata, thus we need to provide a formalisation of this model. In addition, many results established for stochastic models are generalisations of the nonprobabilistic case, and we thus believe that they are easier to understand when reading this chapter first.

In Section 2.1 we describe the low-level models which will form the semantics of hybrid automata. We also define behaviour and reachability properties of them, that is, whether a certain set of states can be reached (safety) or will always be reached (inevitability). Then, in Section 2.2 we discuss the high-level hybrid automata model. This section also describes the semantics in terms of the models of Section 2.1, as well as properties of hybrid automata and their mapping to the low-level semantics. Next, in Section 2.3 we specify the abstractions of hybrid automata. This includes discussing the conditions under which an abstraction is correct for a given hybrid automaton. Section 2.4 will discuss how properties can be decided once the abstraction has been computed. In Section 2.5 we will apply a hybrid solver on a small case study which we will develop throughout the chapter. Afterwards, Section 2.6 discusses related work. Section 2.7 concludes the chapter.

2.1 Semantical Models

In this section, we will introduce the model which will later form the semantics of our hybrid automata. For clarity, we state a few conventions which we will use in the following parts.

2.1.1 Conventions

The set of natural number $\mathbb{N} = \{0, 1, 2, \ldots\}$ does include 0. By $\mathbb{N}^+ = \{1, 2, \ldots\}$ we denote the set of natural numbers without 0. The set of rational numbers will be denoted by $\mathbb{Q}$

and we will denote real numbers by $\mathbb{R}$. In case we restrict to nonnegative real numbers, we write $\mathbb{R}_{\geq 0}$. We might leave out the domain of a variable if it is clear from the context. For example, we might write "for $1 \leq n \leq 10$ we have [...]" rather than "for n with $n \in \mathbb{N}$ and $1 \leq n \leq 10$ we have [...]" if it is clear that n is a natural number.

For a set A, by $A^* \stackrel{\text{def}}{=} \{(a_0, \ldots, a_n) \mid n \in \mathbb{N} \wedge \forall i, 0 \leq i \leq n.\ a_i \in A\}$ we denote the set of finite sequences of elements of A. With $A^\omega \stackrel{\text{def}}{=} \{(a_0, a_1, a_2, \ldots) \mid \forall i \geq 0.\ a_i \in A\}$ we denote the set of infinite sequences of elements of A. For a finite sequence $a = (a_1, \ldots, a_n)$, we let $a^\omega \stackrel{\text{def}}{=} (a_1, \ldots, a_n, a_1, \ldots, a_n, a_1, \ldots, a_n, \ldots)$ denote the infinite repetition of a.

We write $\langle A_i \rangle_{i \in I}$ to denote a family of entities, in which we identify each object A_i by an element i of the index set I.

We use $\uplus$ as the union of sets in case we want to emphasise that these sets are assumed to be disjoint.

The complement of a set A is denoted by A^C.

2.1.2 Labelled Transition Systems

The semantic models we will be using, and later on extend to the probabilistic cases, are *labelled transition systems* [Kel76].

Definition 2.1. *A* labelled transitions system (LTS) *is a tuple*

$$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}),$$

where

- S *is a set of* states*,*
- $\overline{s} \in S$ *is the* initial state*,*
- Act *is a set of* actions*, and the*
- transition matrix $\mathcal{T}\colon (S \times Act) \to 2^S$ *assigns subsets of possible successor states to pairs of states and actions.*

We require that for each $s \in S$ we have $\{a \in Act \mid \mathcal{T}(s, a) \neq \emptyset\} \neq \emptyset$.

Thus, an LTS contains a (possibly uncountable) set of states, one of which is initial. In each state s, by the transition matrix, one can nondeterministically choose a successor action a and state s' with $s' \in \mathcal{T}(s, a)$.

Example 2.2. *In Figure 2.1, we depict a finite example LTS $\mathcal{M} \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T})$. Here, we have $S \stackrel{\text{def}}{=} \{s_0, s_1, s_2\}$, $\overline{s} \stackrel{\text{def}}{=} s_0$, $Act = \{a, b\}$, and*

$$\begin{aligned}
&\mathcal{T}(s_0, a) \stackrel{\text{def}}{=} \{s_1\},\ \mathcal{T}(s_0, b) \stackrel{\text{def}}{=} \{s_2\},\\
&\mathcal{T}(s_1, a) \stackrel{\text{def}}{=} \{s_0, s_1\},\ \mathcal{T}(s_1, b) \stackrel{\text{def}}{=} \emptyset,\\
&\mathcal{T}(s_2, a) \stackrel{\text{def}}{=} \mathcal{T}(s_2, b) \stackrel{\text{def}}{=} \{s_2\}.
\end{aligned}$$

△

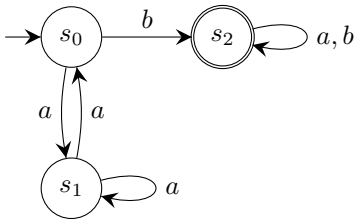


Figure 2.1: Labelled transition system.

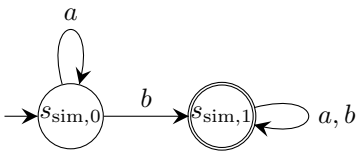


Figure 2.2: Labelled transition system simulating the one of Figure 2.1.

To specify properties of LTSs, we need to define the possible behaviours of such models.

Definition 2.3. *A* finite path *of an LTS* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *is a tuple*

$$\beta_{\text{fin}} = s_0\, a_0\, \ldots\, s_{n-1}\, a_{n-1}\, s_n \in (S \times Act)^* \times S,$$

where $s_0 = \overline{s}$ *and for all* i *with* $0 \leq i < n$ *we have* $s_{i+1} \in \mathcal{T}(s_i, a_i)$. *An* infinite path *is a tuple*

$$\beta_{\text{inf}} = s_0\, a_0\, \ldots \in (S \times Act)^\omega,$$

where $s_0 = \overline{s}$ *and* $s_{i+1} \in \mathcal{T}(s_i, a_i)$ *holds for all* $i \geq 0$. *By* $Path^{\text{fin}}_{\mathcal{M}}$ *we denote the set of all finite paths and by* $Path^{\text{inf}}_{\mathcal{M}}$ *we denote the set of all infinite paths of* $\mathcal{M}$.

The length *of a finite path* β_{fin} *is denoted by* $|\beta_{\text{fin}}| \stackrel{\text{def}}{=} n$. *We let* $\beta_{\text{fin}}[i] \stackrel{\text{def}}{=} \beta_{\text{inf}}[i] \stackrel{\text{def}}{=} s_i$ *denote the* $(i+1)$*-th state of a finite or infinite path (for the* i*-s defined). By* $\text{last}(\beta_{\text{fin}}) \stackrel{\text{def}}{=} s_n$ *we denote the last state of a finite path.*

We define the trace *of finite paths as* $\text{trace}(\beta_{\text{fin}}) = a_0\, a_1\, \ldots\, a_{n-1}$ *and accordingly for infinite ones. The sets of all finite and infinite traces are defined as* $Trace^*_{\mathcal{M}} \stackrel{\text{def}}{=} Act^*$ *and* $Trace^\omega_{\mathcal{M}} \stackrel{\text{def}}{=} Act^\omega$. *Given* $\gamma = a_0\, a_1\, \ldots \in Trace^*_{\mathcal{M}} \uplus Trace^\omega_{\mathcal{M}}$, *we define* $\gamma[i] \stackrel{\text{def}}{=} a_i$ *as the* $(i+1)$*-th action on the trace.*

Consider a subset $Act_{\text{fair}} \subseteq Act$ *of the actions of* $\mathcal{M}$. *We call a path* $\beta \in Path^{\text{inf}}_{\mathcal{M}}$ Act_{fair}-fair *if there are infinitely many* $i \geq 0$ *with* $\text{trace}(\beta)[i] \in Act_{\text{fair}}$. *By* $Path^{Act_{\text{fair}}}_{\mathcal{M}}$ *we denote the set of all* Act_{fair}*-fair paths of* $\mathcal{M}$.

Paths are thus sequences of transitions which are legal according to the transition matrix and start in the initial state. A path is fair if it contains infinitely many positions in which fair actions are chosen.

Example 2.4. *An example of a finite path in the LTS of Figure 2.1 is*

$$\beta_1 \stackrel{\text{def}}{=} s_0\, a\, s_1\, a\, s_1\, a\, s_1,$$

and an example of an infinite path is

$$\beta_2 \stackrel{\text{def}}{=} s_0\, a\, (s_1\, a)^\omega.$$

With $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$, *the path* β_2 *is not* Act_{fair}*-fair, but for instance*

$$\beta_3 \stackrel{\text{def}}{=} s_1\, a\, s_1\, a\, s_1\, a\, s_1\, a\, s_0\, b\, (s_2\, b)^\omega$$

is. We have

$$\begin{aligned}&\mathrm{trace}(\beta_1) = a\,a\,a, \mathrm{trace}(\beta_2) = a^\omega, \mathrm{trace}(\beta_3) = a\,a\,a\,a\,b^\omega,\\ &\mathrm{last}(\beta_1) = s_1,\\ &\beta_1[0] = s_0, \beta_1[1] = \beta_1[2] = s_1, \beta_2[15] = s_1,\\ &\mathrm{trace}(\beta_1)[0] = \mathrm{trace}(\beta_1)[1] = \mathrm{trace}(\beta_1)[2] = \mathrm{trace}(\beta_2)[15] = a.\end{aligned}$$ △

The basic properties we consider are safety properties, which we are now able to express.

Definition 2.5. *Let $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ be an LTS and let $Reach \subseteq S$ be a set of unsafe states. We define $\mathcal{M}$ as Reach-*safe *if there is no $\beta \in Path^{\inf}_{\mathcal{M}}$ for which there is $i \geq 0$ with $\beta[i] \in Reach$.*

Instead of a set of unsafe states, we might also be given a set of desirable states which we want the automaton to always reach eventually, a property called *inevitability* [PW06].

Definition 2.6. *We define an LTS $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ as being Act_{fair}-*restricted *Reach-*inevitable *for a set of desirable states $Reach \subseteq S$ and set of actions $Act_{\mathrm{fair}} \subseteq Act$ if for all $\beta \in Path^{Act_{\mathrm{fair}}}_{\mathcal{M}}$ there is $i \geq 0$ with $\beta[i] \in Reach$.*

In contrast to Definition 2.5, in Definition 2.6 we restrict to fair subsets of paths in which we require *Reach* to be eventually reached. The reason is that later on, in the definition of hybrid automata, we will have to exclude certain unrealistic behaviours (which is not necessary for safety properties).

Example 2.7. *Again, consider the LTS $\mathcal{M}$ of Figure 2.1. Let $Act_{\mathrm{fair}} \stackrel{\mathrm{def}}{=} \{b\}$ and $Reach \stackrel{\mathrm{def}}{=} \{s_2\}$. The LTS is not Reach-safe, because there are paths which lead from $\overline{s}$ to Reach, for instance β_3 of Example 2.4. Indeed, $\mathcal{M}$ is Act_{fair}-restricted Reach-inevitable, because all Act_{fair}-fair paths eventually reach Reach.* △

We now define a way to describe how two LTSs can be related (similar to the notions of [Mil71; Mil89; BK08]).

Definition 2.8. *Given two LTSs $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ and $\mathcal{M}_{\mathrm{sim}} = (S_{\mathrm{sim}}, \overline{s}_{\mathrm{sim}}, Act, \mathcal{T}_{\mathrm{sim}})$, we say that $\mathcal{M}_{\mathrm{sim}}$* simulates *$\mathcal{M}$, denoted by $\mathcal{M} \preceq \mathcal{M}_{\mathrm{sim}}$, if and only if there exists a relation $R \subseteq S \times S_{\mathrm{sim}}$, which we will call* simulation relation *from now on, where*

1. *we have $(\overline{s}, \overline{s}_{\mathrm{sim}}) \in R$,*
2. *for each $(s, s_{\mathrm{sim}}) \in R$, $a \in Act$, and $s' \in \mathcal{T}(s, a)$, there is $s'_{\mathrm{sim}} \in \mathcal{T}_{\mathrm{sim}}(s_{\mathrm{sim}}, a)$ with $(s', s'_{\mathrm{sim}}) \in R$.*

*For two sets of states $Reach \subseteq S$ and $Reach_{\mathrm{sim}} \subseteq S_{\mathrm{sim}}$, we call a simulation relation $Reach$-$Reach_{\mathrm{sim}}$-*compatible *if for all $(s, s_{\mathrm{sim}}) \in R$, we have $s \in Reach$ if and only if $s_{\mathrm{sim}} \in Reach_{\mathrm{sim}}$. If there exists such a relation, we write*

$$(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\mathrm{sim}}, Reach_{\mathrm{sim}}).$$

If an LTS $\mathcal{M}_{\text{sim}}$ simulates another LTS $\mathcal{M}$, it can mimic all possible behaviours of $\mathcal{M}$. Because of this, if $\mathcal{M}_{\text{sim}}$ is safe, also $\mathcal{M}$ is safe.

Lemma 2.9. *Consider two LTSs $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ and $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ and sets of states $Reach \subseteq S$ and $Reach_{\text{sim}} \subseteq S_{\text{sim}}$ with $(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}})$. Then, if $\mathcal{M}_{\text{sim}}$ is $Reach_{\text{sim}}$-safe this implies that $\mathcal{M}$ is Reach-safe.*

Proof. From $(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}})$ we know that there is a simulation relation R. Assume $\mathcal{M}_{\text{sim}}$ is safe, but $\mathcal{M}$ is not. If $\mathcal{M}$ is unsafe, there is a path leading to an unsafe state. For each $\beta = s_0\, a_0\, s_1\, a_1\, \ldots \in Path^{\text{inf}}_{\mathcal{M}}$ there is $\beta_{\text{sim}} = s_{\text{sim},0}\, a_0\, s_{\text{sim},1}\, a_1\, \ldots \in Path^{\text{inf}}_{\mathcal{M}_{\text{sim}}}$ with $(s_i, s_{\text{sim},i}) \in R$ for all $i \geq 0$. This holds by Definition 2.8 and induction on the length of the paths. Because of this and because R is $Reach$-$Reach_{\text{sim}}$-compatible, if β is a path of $\mathcal{M}$ which has states contained in $Reach$, there is also a path of $\mathcal{M}_{\text{sim}}$ with states contained in $Reach_{\text{sim}}$. This contradicts the assumption that $\mathcal{M}_{\text{sim}}$ is safe. □

We can also use simulation relations to show inevitability properties.

Lemma 2.10. *Consider two LTSs $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ and $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$, $Act_{\text{fair}} \subseteq Act$ and two sets of states $Reach \subseteq S$ and $Reach_{\text{sim}} \subseteq S_{\text{sim}}$ with $(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}})$. Then if $\mathcal{M}_{\text{sim}}$ is Act_{fair}-restricted $Reach_{\text{sim}}$-inevitable, this implies that $\mathcal{M}$ is also Act_{fair}-restricted Reach-inevitable.*

Proof. The proof is similar to the one of Lemma 2.9: assume that $\mathcal{M}_{\text{sim}}$ is Act_{fair}-restricted $Reach_{\text{sim}}$-inevitable but $\mathcal{M}$ is not Act_{fair}-restricted $Reach$-inevitable. Then there is a path $\beta = s_0\, a_0\, s_1\, a_1\, \ldots \in Path^{\text{inf}}_{\mathcal{M}}$ which is fair but does not reach $Reach$. Again, we can construct $\beta_{\text{sim}} = s_{\text{sim},0}\, a_0\, s_{\text{sim},1}\, a_1\, \ldots \in Path^{\text{inf}}_{\mathcal{M}_2}$, which is Act_{fair}-fair but does not end in $Reach_{\text{sim}}$, which contradicts the assumption. □

Example 2.11. *Consider the LTSs $\mathcal{M}$ and $\mathcal{M}_{\text{sim}}$ depicted in Figure 2.1 and Figure 2.2. Further, let $Reach \stackrel{\text{def}}{=} \{s_2\}$ and $Reach_{\text{sim}} \stackrel{\text{def}}{=} \{s_{\text{sim},1}\}$. Then we have that $(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}})$, as seen from the $Reach$-$Reach_{\text{sim}}$-compatible simulation relation*

$$R \stackrel{\text{def}}{=} \{(s_0, s_{\text{sim},0}), (s_1, s_{\text{sim},0}), (s_2, s_{\text{sim},1})\}.$$

It is easy to check that R fulfils all requirements of Definition 2.8. It fulfils Requirement 1 because we have $(s_0, s_{\text{sim},0}) \in R$. For Requirement 2, consider for instance $(s_0, s_{\text{sim},0}) \in R$ and $s_1 \in \mathcal{T}(s_0, a)$. We have $s_{\text{sim},0} \in \mathcal{T}(s_{\text{sim},0}, a)$ and $(s_1, s_{\text{sim},0})$ which means that the requirement holds for this transition.
With $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$, we have that $\mathcal{M}_{\text{sim}}$ is Act_{fair}-restricted $Reach_{\text{sim}}$-inevitable. From this, we can conclude that $\mathcal{M}$ is Act_{fair}-restricted Reach-inevitable, as we have already discussed in Example 2.7. △

2.2 Hybrid Automata

In the following, we describe the high-level specification model for hybrid systems which we are going to use. To do so, we need to discuss the component which corresponds

to the timed part of such models. We require this component to fulfil a number of requirements. Some of them are naturally fulfilled by usual mechanisms to describe the timed behaviour of hybrid systems. Others are more restrictive, but all hybrid automata in which these requirements are not fulfilled can be transformed into equivalent hybrid automata in which they are.

Definition 2.12. *A k-dimensional* post operator *with $k \in \mathbb{N}^+$ is a function*

$$Post\colon (\mathbb{R}^k \times \mathbb{R}_{\geq 0}) \to 2^{\mathbb{R}^k},$$

where for each $v \in \mathbb{R}^k$ we have

1. $Post(v, 0) = \{v\}$,
2. *for all* $\mathtt{t}, \mathtt{t}' \in \mathbb{R}_{\geq 0}$ *with* $0 \leq \mathtt{t}' \leq \mathtt{t}$ *we have* $Post(v, \mathtt{t}) = \bigcup_{v' \in Post(v,\mathtt{t}')} Post(v', \mathtt{t} - \mathtt{t}')$,
3. *there exists* $\mathtt{t} \geq 0$ *where for all* $\mathtt{t}' \geq \mathtt{t}$ *we have* $Post(v, \mathtt{t}') = Post(v, \mathtt{t})$ *and for all* $v' \in Post(v, \mathtt{t}')$ *and all* $\mathtt{t}'' \in \mathbb{R}_{\geq 0}$ *we have* $Post(v', \mathtt{t}'') = \{v'\}$.

If v is an evaluation of continuous variables, then $Post(v, \mathtt{t})$ describes the possible values of these variables after trying to wait for time $\mathtt{t}$. Some $v' \in Post(v, \mathtt{t})$ might correspond to letting less time than $\mathtt{t}$ pass, because of possible restrictions of the variable values specified in the model. For example, for a v in which we have a *time stop*, that is, no further time passage is allowed, we have $Post(v, \mathtt{t}) = \{v\}$ for all $\mathtt{t} \geq 0$.

Requirement 1 of Definition 2.12 means that a timed transition has no effect if we let time 0 pass. Requirement 2 is a form of time additivity; instead of letting $\mathtt{t}_1 + \mathtt{t}_2$ time units pass, we can wait firstly for time $\mathtt{t}_1$ and then for time $\mathtt{t}_2$. These requirements are naturally fulfilled by all common hybrid systems mechanisms. They will become especially important later on when reasoning about controller synthesis (cf. Chapter 5).

Requirement 3 states that there is a maximal time for which the automaton can wait in a given state. For all $v' \in Post(v, \mathtt{t}) = Post(v, \mathtt{t}')$ we will have a time stop, that is $Post(v', \mathtt{t}'') = \{v'\}$ for all $\mathtt{t}'' \geq 0$. This will become relevant later when we have to discuss fairness conditions in probabilistic hybrid automata. Models which do not fulfil the condition can easily be transformed to do so.

The post operator allows one to describe various forms of continuous-time dynamics: an important submodel of the general hybrid automata we are going to define are *timed automata* [AD90]. In this model class, all continuous variables are *clocks*, which means that the continuous variables are increased by 1 per time unit, and the post operator can be expressed as

$$Post((v_1, \ldots, v_k), \mathtt{t}) = \{(v_1 + \mathtt{t}', \ldots, v_k + \mathtt{t}') \wedge G\},$$

with a G used to restrict the time flow to a maximal $\mathtt{t}'$ so that a given invariant holds for all $\mathtt{t}''$ with $0 \leq \mathtt{t}'' \leq \mathtt{t}'$. Hybrid systems are often described by differential (in)equations (e.g. [Alu+95; ADI06]). When we are using differential equations, the post operator has the form

$$Post(v, \mathtt{t}) = \{v' \mid \exists (f\colon \mathbb{R}_{\geq 0} \to \mathbb{R}^k).\ f(0) = v \wedge f(\mathtt{t}') = v' \wedge \forall \mathtt{t}'' \in (0, \mathtt{t}').\ \dot{f}(\mathtt{t}'') = g(f(\mathtt{t}''), \mathtt{t}'') \wedge G\}.$$

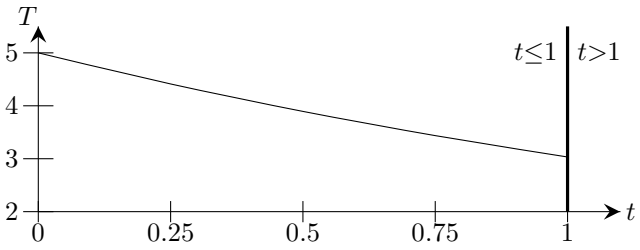


Figure 2.3: Post operator.

Here, $g(f(x), x)$ describes the derivative of the possible trajectories of the automaton, and $f(x)$ is a so-called *witness* function. In a similar way, post operators described by differential inclusions (inequations) can be expressed.

In the following, there is no need to restrict to a specific way of how post operators are defined, except the requirements given in Definition 2.12. As a minor exception, in Chapter 4 we have to put measurability restrictions on these operators.

Example 2.13. *Consider the post operator* $Post_{\text{Check}} \colon (\mathbb{R}^3 \times \mathbb{R}_{\geq 0}) \to 2^{\mathbb{R}^3}$ *with*

$$Post_{\text{Check}}((t, T, c), \mathsf{t}) \stackrel{\text{def}}{=} \{(t + \mathsf{t}', T', c + \mathsf{t}') \mid T' = T \exp(-\mathsf{t}'/2)\},$$

where

$$\mathsf{t}' \stackrel{\text{def}}{=} \max(\{\mathsf{t}' \mid \mathsf{t}' + t \leq 1 \wedge c + \mathsf{t}' \leq \mathfrak{T} \wedge \mathsf{t}' \leq \mathsf{t}\} \cup \{0\}),$$

and $\mathfrak{T} \in \mathbb{R}_{>0}$ *is a time bound discussed later. Here,* exp *denotes the exponential function. We remark that* $T' = T \exp(-\mathsf{t}'/2)$ *is the solution of the differential equation* $\dot{T}' = -T'/2$ *with initial value* T*.*

For the initial value $(0, 5, 0)$*, the behaviour is depicted in Figure 2.3. The graph denotes the set of points which can be reached by a timed transition. The axis labelled with* t *denotes both the values of the time which has passed as well as the continuous variable* t*. The axis* T *denotes the second dimension. We leave out the last dimension* c*. Thus, after time* 0.25*,* T *has a value of about* 4.41*.* △

Our hybrid automata definition then is as follows:

Definition 2.14. *A* hybrid automaton (HA) *is a tuple*

$$\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds),$$

where

- M *is a finite set of* modes*,*
- $k \in \mathbb{N}^+$ *is the* dimension *of the automaton,*
- $\overline{m} \in M$ *is the* initial mode*,*
- $Post_m$ *is a* k*-dimensional post operator for each mode* m*,*
- *Cmds is a finite set of* guarded commands *of the form* $g \to u$*, where*

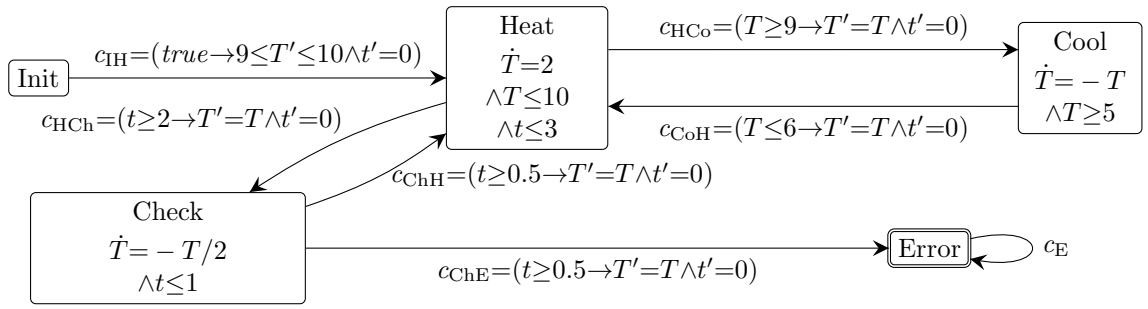


Figure 2.4: HA modelling a thermostat.

  - $g \subseteq M \times \mathbb{R}^k$ *is a* guard,
  - $u\colon (M \times \mathbb{R}^k) \to 2^{M\times\mathbb{R}^k}$ *is an* update function,
  - $u(s) \neq \emptyset$ *for all* $s \in g$, *and*
- *for* $s = (m, v)$ *with* $Post_m(v, \mathtt{t}) = \{v\}$ *for all* $\mathtt{t} \in \mathbb{R}_{\geq 0}$, *there is a command with guard* g *and* $s \in g$.

Thus, HAs consist of a finite set of modes M along with post operators $Post_m$ and guarded commands $Cmds$, as well as a number of k continuous variables. The value of the continuous variables changes over time while being in a given mode m. If the current assignment to the continuous variables is v, then $Post_m(v, \mathtt{t})$ describes the set of variable evaluations which can be reached by trying to wait for time $\mathtt{t}$. A command $c = (g \to u)$ can be executed in mode m with variable assignments v if $(m, v) \in g$. If it is executed, there is a nondeterministic choice of successor modes and variable values $(m', v') \in u(m, v)$ (in case $u(m, v)$ is not singleton). Time proceeds only in transitions resulting from $Post_m$, but the command-driven transitions take zero time. The last requirement that we have to be able to execute a command in case of a time stop will be needed when considering fairness.

We remark that it is not strictly necessary to include guards in the definition of commands; given a guarded command $c = (g \to u)$, we could transform it into a command of the form $c' \stackrel{\text{def}}{=} (u')$ so that $u'(s) \stackrel{\text{def}}{=} u(s)$ for all $s \in g$ and $u'(s) \stackrel{\text{def}}{=} \emptyset$ else. This way, the fact that a guard is valid in a state is expressed by a nonempty update in this state. However, to make the notations more handy and descriptive, we will use commands with guards in this thesis.

HAs allow several degrees of freedom, to which there is no associated probabilistic interpretation, and which is resolved nondeterministically: there can be nondeterminism in the continuous-time part, for instance in case of differential inclusions, so that more than one possible state can be reached by waiting for a certain time. Another source of nondeterminism results because a guarded command can usually be executed at different points of time, so there is a nondeterministic choice over the amount of time to wait. Further, the execution of a guarded command might lead to several possible outcomes.

Example 2.15. *Consider the example HA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m\rangle_{m\in M}, Cmds)$ *shown in Figure 2.4, which models a thermostat. There are five modes:* $M = \{\mathrm{Init}, \mathrm{Heat}, \mathrm{Cool},$

Check, Error$\}$. Init *is the initial mode. Modes* Heat *and* Cool *implement heating and cooling functions. In the mode* Check*, the thermostat can fail nondeterministically. In it does, the* Error *mode is entered, whereas otherwise the thermostat continues its normal operations.*

There are $k = 3$ *continuous variables,* t*,* T *and* c*.* T *represents the temperature and* t *represents the time since entering a mode. We thus have* $(\dot{t} = 1)$ *in each mode (which is left out in the figure). The variable* c *is a global timer which initially is* 0*, we have* $\dot{c} = 1$ *in all modes,* $(c' = c)$ *in each of the command updates, and have the additional constraint* $(c \leq \mathfrak{T})$ *in each mode, where* $\mathfrak{T}$ *is the time the system is supposed to run.*

Commands are represented as arrows. The set of all commands is $Cmds = \{c_{\mathrm{IH}}, c_{\mathrm{HCo}}, c_{\mathrm{HCh}}, c_{\mathrm{ChH}}, c_{\mathrm{ChE}}, c_{\mathrm{E}}\}$*. Primed variables denote the values which are assigned by the updates. For instance, there are two commands available in mode* Check*:* $c_{\mathrm{ChH}} = (g \to u_{\mathrm{ChH}})$ *and* $c_{\mathrm{ChE}} = (g \to u_{\mathrm{ChE}})$ *where*

- $g = \{\mathrm{Check}\} \times [0.5, \infty) \times \mathbb{R} \times \mathbb{R}$*,*
- $u_{\mathrm{ChH}}((m, t, T, c)) = \{(\mathrm{Heat}, 0, T, c)\}$*, and*
- $u_{\mathrm{ChE}}((m, t, T, c)) = \{(\mathrm{Error}, 0, T, c)\}$*.*

The post operator of each mode is described using differential equations. The post operator of Check *is described in detail in Example 2.13, and the others are formalised in the same way. As noted, in addition to the restrictions given in the figure, we assume that everywhere we have* $(\dot{t} = 1)$*,* $(\dot{c} = 1)$ *and* $(c \leq \mathfrak{T})$*. The command* c_{E} *is needed by the last requirements on HAs and post operators, although the exact definition of the timed behaviour in the error mode is not important. For the same reason, we assume that in the other modes* c_{E} *is activated in case* $(c = \mathfrak{T})$*.* △

We have given a definition of classical HAs which comprises the usual cases to be analysed by the solvers for such systems, and are now almost ready to define the semantics of our HAs. However, some of the usual hybrid systems solvers, which we want to build our analysis methods on, do not directly represent all timed transitions of the hybrid system semantics by corresponding timed transitions in the abstractions they compute. Timed transitions might be left out in case they can be substituted by a sequence of shorter timed transitions. As a preparation for the later definition of abstractions, we define a restriction of post operators which takes into account this issue.

Definition 2.16. *A* time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$ *for a family of* k*-dimensional post operators* $\langle Post_m \rangle_{m \in M}$ *is defined so that for each* $m \in M$ *we have*

- $\mathbf{t}_m \colon \mathbb{R}^k \to \mathbb{R}_{\geq 0}$*,*
- *for each* $v \in \mathbb{R}^k$*,* $\mathtt{t} \geq 0$ *and* $v' \in Post_m(v, \mathtt{t})$ *there is* $n \geq 1$ *for which there are* $v_1, \ldots, v_n \in \mathbb{R}^k$ *and* $\mathtt{t}_1, \ldots, \mathtt{t}_{n-1} \in \mathbb{R}_{\geq 0}$ *where*
  - $v = v_1$*,*
  - $v' = v_n$*,*
  - $\sum_{i=1}^{n-1} \mathtt{t}_i = \mathtt{t}$*, and*
  - *for* i *with* $1 \leq i < n$ *we have* $\mathtt{t}_i \leq \mathbf{t}_m(v_i)$ *and* $v_{i+1} \in Post_m(v_i, \mathtt{t}_i)$*.*

The definition specifies a maximal amount of time $\mathbf{t}_m(v)$ which can pass with a single transition in a given mode m given the values v of the continuous variables. We must make sure that the relevant properties of the hybrid automaton are not affected by this restriction. For this, the definition requires that, if it is possible to reach v' by letting time $\mathtt{t}$ pass in m with an assignment v to the continuous variables, we can still always reach v' by waiting instead a number of times for durations $\mathtt{t}_i$, so that each $\mathtt{t}_i$ is legal by the time restriction and their sum equals $\mathtt{t}$. In Chapter 5, the time restrictions will even turn out to be useful rather than just necessary because of the hybrid solvers, as they will then allow for more precise results.

We can now define the semantics of HAs in terms of LTSs. As a preparation for the later abstractions, we also define a variant in which time restrictions are taken into account.

Definition 2.17. *The* semantics *of an HA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *is the LTS*

$$[\![\mathcal{H}]\!] \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T}),$$

where

- $S \stackrel{\text{def}}{=} M \times \mathbb{R}^k$,
- $\overline{s} \stackrel{\text{def}}{=} (\overline{m}, 0, \ldots, 0)$,
- $Act \stackrel{\text{def}}{=} \mathbb{R}_{\geq 0} \uplus Cmds$,
- *for* $s = (m, v) \in S$ *we have*
  - *for* $c = (g \to u) \in Cmds$: $\mathcal{T}(s, c) \stackrel{\text{def}}{=} \begin{cases} u(s) & \text{if } s \in g, \\ \emptyset & \text{else}, \end{cases}$
  - *for* $\mathtt{t} \in \mathbb{R}_{\geq 0}$: $\mathcal{T}(s, \mathtt{t}) \stackrel{\text{def}}{=} \{(m, v') \mid v' \in Post_m(v, \mathtt{t})\}$.

Given a time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$, *the* time-restricted semantics *is the LTS* $[\![\mathcal{H}, \mathbf{T}]\!] = (S, \overline{s}, Act, \mathcal{T})$ *where* S, $\overline{s}$, Act *and* $\mathcal{T}(s, c)$ *for each* $s = (m, v) \in S$ *and* $c \in Cmds$ *are as in the semantics above. We restrict the timed transitions where for* $\mathtt{t} \in \mathbb{R}_{\geq 0}$ *and* $\mathtt{t}_{\mathbf{T}} \stackrel{\text{def}}{=} \min\{\mathtt{t}, \mathbf{t}_m(v)\}$ *we have* $\mathcal{T}(s, \mathtt{t}) \stackrel{\text{def}}{=} \{(m, v') \mid v' \in Post_m(v, \mathtt{t}_{\mathbf{T}})\}$.

Example 2.18. *Consider the HA* $\mathcal{H}$ *of Figure 2.4. Then we have*

$$[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T}),$$

where with the formalisations of Example 2.15 we have

- $S = M \times \mathbb{R}^3$,
- $\overline{s} = (\mathrm{Init}, 0, 0, 0)$,
- $Act = Cmds \uplus \mathbb{R}_{\geq 0}$,
- $\mathcal{T}\colon (S \times Act) \to 2^S$.

We exemplify the behaviour on the states of the mode Check. *For* $s = (\mathrm{Check}, t, T, c) \in \{\mathrm{Check}\} \times \mathbb{R}^3$ *we have*

$$\mathcal{T}(s, c_{\mathrm{IH}}) = \mathcal{T}(s, c_{\mathrm{HCo}}) = \mathcal{T}(s, c_{\mathrm{HCh}}) = \mathcal{T}(s, c_{\mathrm{CoH}}) = \mathcal{T}(s, c_{\mathrm{E}}) = \emptyset,$$

$$\mathcal{T}(s, c_{\mathrm{ChH}}) = \begin{cases} \emptyset & \text{if } t < 0.5, \\ \{(\mathrm{Heat}, 0, T, c)\} & \text{else}, \end{cases}$$
$$\mathcal{T}(s, c_{\mathrm{ChE}}) = \begin{cases} \emptyset & \text{if } t < 0.5, \\ \{(\mathrm{Error}, 0, T, c)\} & \text{else}, \end{cases},$$

and for $\mathsf{t} \in \mathbb{R}_{\geq 0}$ *we have*

$$\mathcal{T}(s, \mathsf{t}) = \{(\mathrm{Check}, t + \mathsf{t}', T', c + \mathsf{t}') \mid T' = T \exp(-\mathsf{t}'/2)\},$$

where

$$\mathsf{t}' \stackrel{\text{def}}{=} \max(\{\mathsf{t}' \mid \mathsf{t}' + t \leq 1 \wedge c + \mathsf{t}' \leq \mathfrak{T} \wedge \mathsf{t}' \leq \mathsf{t}\} \cup \{0\}).$$

Consider the time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$ *where for each* $m \in M$ *we have* $\mathbf{t}_m(v) = 0.25$. *Then for* $[\![\mathcal{H}, \mathbf{T}]\!]$ *we have*

$$\mathsf{t}' \stackrel{\text{def}}{=} \max(\{\mathsf{t}' \mid \mathsf{t}' + t \leq 1 \wedge c + \mathsf{t}' \leq \mathfrak{T} \wedge \mathsf{t}' \leq \mathsf{t} \wedge \mathsf{t} \leq 0.25\} \cup \{0\}).$$

Thus, $\mathcal{T}((\mathrm{Check}, 0, 5, 0), 0.25) = \mathcal{T}((\mathrm{Check}, 0, 5, 0), 0.3) = \mathcal{T}((\mathrm{Check}, 0, 5, 0), 100) \approx (\mathrm{Check}, 0.25, 4.41, 0.25)$. *The fact that we cannot let more than* 0.25 *units of time pass is not an actual restriction, as we have for instance* $\mathcal{T}((\mathrm{Check}, 0.25, 4.41, 0.25), 0.25) \approx (\mathrm{Check}, 0.5, 3.89, 0.5)$, *that is we can replace a single time transition which is longer than the maximal time allowed by the time restriction by a sequence of transitions, each of which is legal by the time restriction.* △

We now state when we consider an HA as safe for a given unsafe mode.

Definition 2.19. *Let* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *be an HA with a mode* m_{Reach}. *We say that* $\mathcal{H}$ *is* m_{Reach}*-safe for* $m_{Reach} \in M$ *if* $[\![\mathcal{H}]\!]$ *is* $(m_{Reach} \times \mathbb{R}^k)$*-safe.*

Inevitability properties of HAs are defined as follows:

Definition 2.20. *We define an HA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *as* m_{Reach}-inevitable *for* $m_{Reach} \in M$ *if* $[\![\mathcal{H}]\!]$ *is Cmds-restricted* $(m_{Reach} \times \mathbb{R}^k)$*-inevitable.*

Contrary to the safety property, we restrict to the paths in which infinitely many commands occur. The reason we apply the restriction is that otherwise there would be no nontrivial m_{Reach}-inevitable automata at all: we will always find a path which is *time convergent* [BK08, Chapter 9], that is, its trace will end in a sequence

$$\mathsf{t}_0\, \mathsf{t}_1\, \mathsf{t}_2\, \mathsf{t}_3\, \ldots,$$

with $\sum_{i=0}^{\infty} \mathsf{t}_i < \infty$. This way, time will effectively stop, and m_{Reach} will never be reached just because of this reason (except in the trivial case where the initial mode equals m_{Reach}). Under the assumption that in reality time progress will never stop, paths with such traces can be considered as being unrealistic. Now consider a path with a trace which ends in an infinite sequence of timed actions and which is time divergent,

that is $\sum_{i=0}^{\infty} \mathsf{t}_i = \infty$. By the definition of the post operators (cf. Definition 2.12, Requirement 3), at some point the timed actions do not have any further effect, which is also equivalent to a time stop. In such a case, by Definition 2.14 we must be able to execute a guarded command, and our fairness assumptions imply that we will. Thus, all realistic paths involve the execution of infinitely many actions.

If we only require that on a path infinitely many commands are executed, it can still happen that time stops, in case we are able to execute an infinite number of guarded commands in finite time. However, this possibility can be excluded by making the automaton *structurally (strongly) nonzeno* [AMP94][Asa+00, Definition 6], that is by defining the guards of the guarded commands in a way that they cannot be executed without a minimal delay. In addition, this might not even be necessary, in case all paths in which infinitely many actions are executed already lead to either m_{Reach} or to a mode from which it is clear that m_{Reach} can never be reached again.

In principle, we could instead simply have restricted to time-divergent paths. However, the mechanism we use here is more appropriate when using the abstractions of the HA semantics we will later on discuss. In addition, it will turn out to be useful when considering reward-based properties in Chapter 7.

The following lemma shows that, concerning the properties we define on HAs, for our properties of interest it does not matter whether or not we consider the time restriction.

Lemma 2.21. *Let $\mathcal{H}$ be an HA with commands Cmds, let $\mathbf{T}$ be a suitable time restriction and let Reach $\subseteq S$ where S is the set of states of $[\![\mathcal{H}]\!]$. Then $[\![\mathcal{H}]\!]$ is Reach-safe if and only if $[\![\mathcal{H}, \mathbf{T}]\!]$ is Reach-safe. Likewise, we have that $[\![\mathcal{H}]\!]$ Cmds-restricted Reach-inevitable if and only if $[\![\mathcal{H}, \mathbf{T}]\!]$ is Cmds-restricted Reach-inevitable.*

Proof. We consider the case that $[\![\mathcal{H}]\!]$ is safe. Assume $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$. Let $\beta_{\mathbf{T}}$ be an arbitrary path of $[\![\mathcal{H}, \mathbf{T}]\!]$. Without loss of generality, we can assume that for each state s of $\beta_{\mathbf{T}}$ only time durations up to the maximum allowed by the time restriction are chosen; choosing longer durations has no effect other than choosing the maximal time allowed by the time restriction. This way, for $\beta_{\mathbf{T}} = (m_0, v_0)\, a_0\, (m_1, v_1)\, a_1 \ldots$ for all $i \geq 0$ with $a_i \in \mathbb{R}_{\geq 0}$ we have $a_i \leq \mathbf{t}_{m_i}(v_i)$. By the definition of the time-restricted semantics, for a path $\beta_{\mathbf{T}} = (m_0, v_0)\, a_0\, (m_1, v_1)\, a_1 \ldots$ for which this does not hold, we can construct a path with the same sequence of states, namely $\beta'_{\mathbf{T}} = (m_0, v_0)\, a'_0\, (m_1, v_1)\, a'_1 \ldots$ where we have $a'_i = \min\{\mathbf{t}_{m_i}(v_i), a_0\}$ in case $a_i \in \mathbb{R}_{\geq 0}$ and $a'_i = a_i$ otherwise. All paths of the required form are also paths of $[\![\mathcal{H}]\!]$. Thus, also $[\![\mathcal{H}, \mathbf{T}]\!]$ is safe, because otherwise there would be a path of $[\![\mathcal{H}]\!]$ leading to an unsafe state, which would violate the assumption.

Now assume that $[\![\mathcal{H}, \mathbf{T}]\!]$ is safe. Consider an arbitrary path β of $[\![\mathcal{H}]\!]$. We construct a path $\beta_{\mathbf{T}}$ of $[\![\mathcal{H}, \mathbf{T}]\!]$ as follows: because of the requirements on post operators (cf. Definition 2.12) and time restrictions (cf. Definition 2.16), each part $(m, v)\, \mathsf{t}\, (m, v')$ of the path with $\mathsf{t} > \mathbf{t}_m(v)$ can be replaced by a sequence $(m, v_1)\, \mathsf{t}_1 \ldots \mathsf{t}_{n-1}\, (m, v_n)$ with $v_1 = v$, $v_n = v'$ and for all i with $1 \leq i < n$ we have $\mathsf{t}_i \leq \mathbf{t}_m(v_i)$ and $v_{i+1} \in \mathit{Post}_m(v_i, \mathsf{t}_i)$. The path $\beta_{\mathbf{T}}$ constructed this way is thus a path of $[\![\mathcal{H}]\!]$ as well as $[\![\mathcal{H}, \mathbf{T}]\!]$. Because $[\![\mathcal{H}, \mathbf{T}]\!]$ is safe, there cannot be any states of $\beta_{\mathbf{T}}$ which are contained in *Reach*. This also

holds for β, as the set of states occurring in $\beta_{\mathbf{T}}$ is a superset of those in β. Because β was chosen arbitrarily, we can conclude the safety of $[\![\mathcal{H}]\!]$.

The proof concerning *Cmds*-restricted *Reach*-inevitable properties follows by the same construction. It maintains Act_{fair}-fairness of paths, because it only modifies timed transitions, so that the fact whether or not infinitely many actions of $Act_{\text{fair}} = Cmds$ occur is not affected. □

Example 2.22. *Consider the HA $\mathcal{H}$ of Figure 2.4 with the semantics in Example 2.18, consider the unsafe mode* Error *and let Reach* $\stackrel{\text{def}}{=} \{\text{Error}\} \times \mathbb{R}^3$. *Whether $\mathcal{H}$ is* Error*-safe depends on the time bound $\mathfrak{T}$. If $\mathfrak{T}$ is small enough it is indeed safe, but for $\mathfrak{T} \geq 3.69$, the path*

$$(\text{Init}, 0, 0, 0)\ c_{\text{IH}}\ (\text{Heat}, 0, 9, 0)\ 0.5\ (\text{Heat}, 0.5, 10, 0.5)\ c_{\text{HCo}}\ (\text{Cool}, 0, 10, 0.5)\ 0.69$$
$$(\text{Cool}, 0.69, 5.016, 1.19)\ c_{\text{CoH}}\ (\text{Heat}, 0, 5.016, 1.19)\ 2\ (\text{Heat}, 2, 9.016, 3.19)\ c_{\text{HCh}}$$
$$(\text{Check}, 0, 9.016, 3.19)\ 0.5\ (\text{Check}, 0.5, 7.016, 3.69)\ c_{\text{ChE}}\ (\text{Error}, 0, 7.016, 3.69)$$

(where the numbers are only approximate) leads from the initial state to the unsafe mode. The automaton is not Error*-inevitable: the only possibility to move to* Error *is in* Check*, and there are indeed paths which are Cmds-fair but never enter this mode. Also, in* Check *it is never necessary to move to* Error*, because instead of c_{ChE} one can always choose c_{ChH}. However, if we started in the mode* Check *and removed c_{ChH}, we would always end up in* Error *with each Cmds-fair path, because then c_{ChE} would be the only element of Cmds left in* Check*. If we do not restrict to Cmds-fair paths, the automaton would still not be guaranteed to reach* Error*, because paths in which we repeatedly choose to let some time pass would be allowed.* △

2.3 Abstractions

In the following, we will be concerned with abstractions of HAs. Firstly, we have to describe abstract state spaces, which we will use to subsume uncountably many states of the infinite semantics of the HAs.

Definition 2.23. *An* abstract state space *of dimension k for a set of modes M is a finite set $\mathbf{A} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ where $\mathbf{z}_i = (m_i, \zeta_i) \in M \times 2^{\mathbb{R}^k}$ and we have $\bigcup_{(m,\zeta)\in\mathbf{A}} \zeta = \mathbb{R}^k$ for all $m \in M$. We identify (m, ζ) with the set $\{m\} \times \zeta$ which allows us to apply the usual set operations on abstract states, and we will for instance write $s \in (m, \zeta)$.*

We do not require $\mathbf{A}$ to be a *partitioning* of $M \times \mathbb{R}^k$, that is we do allow overlapping states. This way, one concrete state may be contained in several abstract states. We need to allow this, because in several hybrid system solvers from which we obtain these abstractions, these cases indeed happen. For instance, in the tool HSolver [RS07] we may have overlapping borders, whereas for PHAVer [Fre05; Fre08] we may also have common interiors of abstract states.

With these preparations, we can now define the abstraction of an HA.

Definition 2.24. *Consider an HA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$*, an abstract state space* $\mathbf{A} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ *of compatible dimension and modes as well as a time restriction* $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$*. We say that*

$$\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T})$$

is an abstraction *of* $\mathcal{H}$ *using* $\mathbf{A}$ *and* $\mathbf{T}$ *if*

- $(\overline{m}, 0, \ldots, 0) \in \overline{\mathbf{z}}$*,*
- *for all* $\mathbf{z} \in \mathbf{A}$*,* $s \in \mathbf{z}$*,* $c = (g \to u) \in Cmds$*, if* $s \in \mathbf{z} \cap g$ *then for all* $s' \in u(s)$ *there is* $\mathbf{z}' \in \mathbf{A}$ *with* $s' \in \mathbf{z}'$ *and* $\mathbf{z}' \in \mathcal{T}(\mathbf{z}, c)$*,*
- *for all* $\mathbf{z} \in \mathbf{A}$*,* $s = (m, v) \in \mathbf{z}$*,* $\mathtt{t} \in \mathbb{R}_{\geq 0}$ *with* $\mathtt{t} \leq \mathbf{t}_m(v)$ *and all* $s' = (m, v')$ *with* $v' \in Post_m(v, \mathtt{t})$*, we require that* $s' \in \mathbf{z}$ *or there is* $\mathbf{z}' \in \mathbf{A}$ *with* $s' \in \mathbf{z}'$ *and* $\mathbf{z}' \in \mathcal{T}(\mathbf{z}, \tau)$*.*

By $Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ *we denote the set of all such abstractions.*

The basic idea of the abstraction definition is that it must at least include all behaviours of the semantics of the HA it is representing. It is allowed to include more; if we add additional transitions to an abstraction, it will remain an abstraction, but might become less precise. Instead of having actions of $\mathbb{R}_{\geq 0}$, we now have a single action τ representing the passing of time. This means that we have no information about the durations of timed transitions in an abstraction. The definition does not construct timed self loops ($\mathbf{z} \in \mathcal{T}(\mathbf{z}, \tau)$) even though we always have timed self loops ($s \in \mathcal{T}(s, \mathtt{t})$) in the semantics by letting time 0 pass ($s \in \mathcal{T}(s, 0)$): for reachability properties, self loops are unnecessary anyway while, when considering the inevitability properties, we indirectly restrict to paths which are not time convergent.

The following theorem states how we can use abstractions to show the safety of HAs.

Theorem 2.25. *Consider an HA* $\mathcal{H}$*, an abstraction* $\mathcal{M} \in Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ *and an unsafe mode* m_{Reach}*. Let* $Reach \stackrel{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{Reach}\}$*. If* $\mathcal{M}$ *is Reach-safe, then* $\mathcal{H}$ *is* m_{Reach}*-safe.*

Proof. Assume $\mathcal{M}$ is *Reach*-safe. Because of Lemma 2.21, it suffices to show that the time-restricted semantics $[\![\mathcal{H}, \mathbf{T}]\!]$ is safe concerning $m_{Reach} \times \mathbb{R}^k$.
For $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T})$ we define

$$\mathcal{M}_{\text{loop}} \stackrel{\text{def}}{=} (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \mathbb{R}_{\geq 0}, \mathcal{T}_{\text{loop}}),$$

where for all $\mathbf{z} \in \mathbf{A}$ we have

- $\mathcal{T}_{\text{loop}}(\mathbf{z}, c) \stackrel{\text{def}}{=} \mathcal{T}(\mathbf{z}, c)$ for $c \in Cmds$, and
- for all $\mathtt{t} \in \mathbb{R}_{\geq 0}$ we have $\mathcal{T}_{\text{loop}}(\mathbf{z}, \mathtt{t}) \stackrel{\text{def}}{=} \mathcal{T}(\mathbf{z}, \tau) \cup \{\mathbf{z}\}$.

Then $\mathcal{M}$ is safe if and only if $\mathcal{M}_{\text{loop}}$ is safe, as the self loops do not influence whether a certain state is reached or not. Also, we can show that $\mathcal{M}_{\text{loop}}$ simulates $[\![\mathcal{H}, \mathbf{T}]\!]$ in the notion of Definition 2.8 by using the simulation relation

$$R \stackrel{\text{def}}{=} \{(s, \mathbf{z}) \mid \mathbf{z} \in \mathbf{A} \wedge s \in \mathbf{z}\}.$$

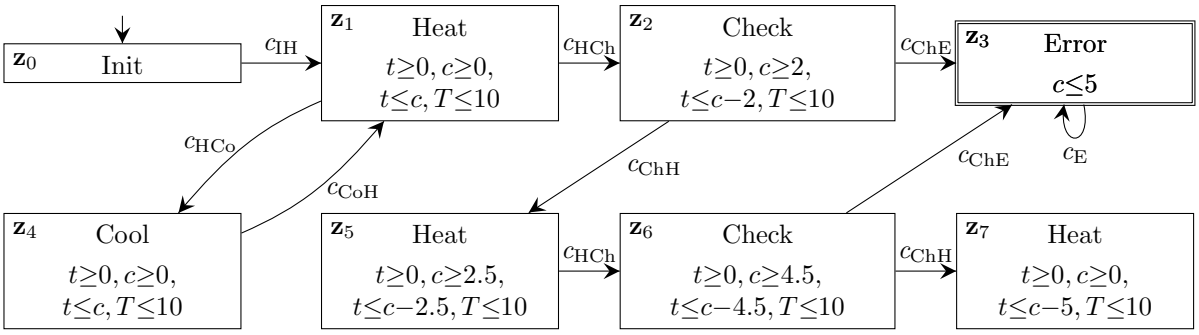


Figure 2.5: Abstraction of the thermostat HA.

It is easy to check that by definition all the requirements on a $(\{m_{Reach}\} \times \mathbb{R}^k)$-*Reach*-compatible simulation relation are fulfilled. By Lemma 2.9, the safety of $\mathcal{M}_{\text{loop}}$ thus implies the safety of $[\![\mathcal{H}, \mathbf{T}]\!]$.

In total, the safety of $\mathcal{H}$ follows:

$$(\mathcal{M} \text{ safe}) \Rightarrow (\mathcal{M}_{\text{loop}} \text{ safe}) \Rightarrow ([\![\mathcal{H}, \mathbf{T}]\!] \text{ safe}) \Rightarrow ([\![\mathcal{H}]\!] \text{ safe}). \qquad \square$$

We might have an HA which is indeed safe, but an abstraction which is not safe. In this case, we will have to use a different abstraction of the automaton, possibly by splitting abstract states.

We have a corresponding result for inevitability properties.

Theorem 2.26. *Consider an HA $\mathcal{H}$ with the set of guarded commands Cmds, an abstraction $\mathcal{M} \in Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ and a mode m_{Reach}. Let Reach $\stackrel{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{Reach}\}$. If $\mathcal{M}$ is Cmds-restricted Reach-inevitable, then $\mathcal{H}$ is m_{Reach}-inevitable.*

Proof. The proof follows along the lines of the one of Theorem 2.25. When adding self loops in $\mathcal{M}_{\text{loop}}$, we make use of the fact that $\mathtt{t} \in \mathbb{R}_{\geq 0}$ is not contained in *Cmds*, ensuring that fair paths of $\mathcal{M}_{\text{loop}}$ cannot end in infinite sequences of timed self loops. For each path β which only contains a finite number of timed self loops, there is a path β_{noloop} in which the timed self loops have been left out and which contains an unsafe state if and only if β contains an unsafe state. In turn, it suffices to consider the paths which do not contain such self loops at all, and the set of these paths agrees on $\mathcal{M}$ and $\mathcal{M}_{\text{loop}}$. $\square$

Example 2.27. *In Figure 2.5 we depict a part of an abstraction $\mathcal{M}$ of the HA $\mathcal{H}$ of Figure 2.4 for $\mathfrak{T} = 5$. For each of the abstract states, we give the corresponding mode and the restrictions on the contained variable evaluations. Let Reach $\stackrel{\text{def}}{=} \{\mathbf{z}_3\}$. Obviously, $\mathcal{M}$ is not Reach-safe but also not Cmds-restricted Reach-inevitable. This was to be expected, as in Example 2.22 we have already seen that $\mathcal{H}$ is neither* Error*-safe nor* Error*-inevitable. There is a path which has no direct correspondence in the concrete model, namely*

$$\mathbf{z}_0\ c_{\text{IH}}\ \mathbf{z}_1\ c_{\text{HCh}}\ \mathbf{z}_2\ c_{\text{ChE}}\ \mathbf{z}_3.$$

There is no corresponding path in the concrete semantics, as there it is not possible to move from the mode Heat *to* Check *and then to* Error *without visiting other modes in between.* △

2.4 Algorithmic Consideration

Once an abstraction has been computed, it remains to check whether the property of interest holds in the abstraction. In the case of nonstochastic HAs which we are considering here, properties can be decided by standard graph-based algorithms: for reachability, we have to find out whether an unsafe state can be reached from the initial state of the LTS, which is indeed a special kind of graph. For inevitability, we have to find out whether all paths starting in the initial state will eventually reach a given set of states. Many hybrid system solvers, for instance PHAVer [Fre05; Fre08], do not explore the complete state space at all, but only the subset of states necessary to decide these properties.

2.5 Case Study

We applied the hybrid systems solver PHAVer [Fre05; Fre08] to our thermostat model from Example 2.15, as a preparation for the later analyses of probabilistic variants of this model. The solver is used both to create an abstract state space as well as to compute the transitions between the abstract states. If PHAVer is applied on linear HAs, in which all the differential (in)equations are equal to or bounded by constants, it will compute the exact sets of reachable states in case it succeeds. Having done so, it is trivial to decide whether a set of unsafe states is reachable. Because the problem of reachability is already undecidable for this class of HAs [Hen+98, Section 4], PHAVer cannot always succeed in computing the set of reachable states, however. Abstract states constructed by PHAVer consist of a mode and a set of convex polyhedra of the dimension k of the automaton. In order to be able to handle dynamics which are not linear over time (e.g. $\dot{T} = -T$, as in our thermostat), it overapproximates reachable states by sets of states which can be described exactly by polyhedra. Starting from the initial state, PHAVer explores the state space by extending abstract states in such a way that they contain the states which can be reached by timed transitions. For linear HAs, this operation is exact. Afterwards, it computes the effects of the guarded commands, and repeats the procedure until a fixed point is reached. To handle nonlinear dynamics, it overapproximates them by linear dynamics. In order to improve the precision of the computation of reachable states, it can constrain the maximal width of polyhedra in a given dimension. This way, abstract states contain fewer concrete states, and the overapproximation by linear dynamics becomes usually more precise. This corresponds to the time restrictions we considered: if PHAVer, using such a constraint, constructs a sequence of abstract states $\mathbf{z}_1, \ldots, \mathbf{z}_n$ in which $\mathbf{z}_j$ corresponds to a later point in time than $\mathbf{z}_i$ for $i \leq j$, we might not have transitions from $\mathbf{z}_i$ to $\mathbf{z}_j$ directly, but only from $\mathbf{z}_i$ to $\mathbf{z}_{i+1}$. Decreasing the

$\mathfrak{T}$	constraint length 10		constraint length 2	
	build (s)	states	build (s)	states
2	0	8	0	11
4	0	12	0	43
5	0	13	1	58
20	1	95	13	916
40	18	609	36	2207
80	59	1717	85	4916
120	31	1502	91	4704
160	177	4260	183	10195
180	136	3768	219	10760
600	625	12617	1268	47609

Table 2.1: Thermostat performance figures.

splitting constraint is not an abstraction refinement in the classical sense [Cla+00], as it does not split abstract states from former runs, but builds an entirely new state space. Because of this, the splitting is not guaranteed to improve the abstraction, though it usually does. As an optional feature, PHAVer can also subsume polyhedra by building their convex hull, thus overapproximating the set of states they contain.

The experiments we performed were run on an Intel(R) Core(TM)2 Duo CPU with 2.67 GHz and 4 GB RAM. In Table 2.1, we give performance statistics (time to build the abstraction and number of constructed abstract states) for different time bounds $\mathfrak{T}$. For the left (right) part of the table, we set the interval length for the variable c to 10 (respectively 2). We did not use the convex-hull overapproximation here. The time needed for the analysis as well as the number of states of the abstract transition systems grows about linearly in the time bound, though with oscillations. Comparing the left and the right side, we see that for the larger interval we need less resources, as was to be expected. PHAVer is never able to show that the system is safe, because indeed it is not. In the corresponding section of the next chapter, we will however use the results of the analysis to prove probabilistic bounds.

2.6 Related Work

The development of HA models began in the years 1991-1993 [MMP91; NSY91; NSY93; Nic+92], as an extension of *timed automata* of which they form a superclass. Among the most notable authors in the area of HAs are Oded Maler, Joseph Sifakis, and Thomas Henzinger. HAs have found successful applications in many areas of interest, including biological models [GT04; Ye+06], verification of analog electronics [Fre05, Section 4.3],[GKR04] and industrial applications [KSB01; ZBL07].

The most important difference between the different formalisations of HAs is in the description of the timed behaviour. Most often, it is described by solutions of differential equations or inclusions [Alu+95; ADI06] like we do. Other mechanisms exist, for in-

stance using o-minimal theories to describe the timed behaviours [LPS98]. During the transitions from mode to mode, the continuous variables can be modified according to the transition.

The continuous variables of *timed automata* [AD90; AD94; Dil89; Hen+92; Hen+94; Lew90; ACD93] are restricted to *clocks* which must be constantly increased by 1 per time unit, that is, we have $\dot{x} = 1$ for all continuous variables x. In addition, all clocks start with value 0 and might only be left unchanged or be reset to 0 on guarded commands. There are some other requirements on the specification of these systems. Although these restrictions seem quite severe, in many cases timed automata are a sufficient mechanism, for instance for the description of real-time systems. They also have the advantage that many properties are decidable, as for instance reachability. Other relevant subclasses of HAs exist, for instance *(piecewise) linear HAs* (with constant bounds on derivatives of continuous variables), *(initialised) rectangular HAs*, *o-minimal HAs*, for which there are also a number of (non)decidability results [Pre+98; Hen+98; LPY99; Alu+95]. The safety of more general HAs can be proven if they are invariant under small disturbances [Rat10].

The problem of reachability is undecidable for general HAs classes and even seemingly slight extensions of the aforementioned classes [Hen+98, Section 4]. Nevertheless, there exist tools which can, often, solve the reachability problem even for such automata. Most tools concentrate on solving the problem for *affine* HAs (often also called "linear systems", although bounds on derivatives need not be constant, that is, automata in which the continuous behaviour is given by linear differential (in)equations. There are however also tools for the general case which even involves differential (in)equations given in an implicit form [RS07].

Many tools work by exploring an overapproximation of the states reachable from the initial states and then check whether the unsafe states are reachable [Alu+95; AHH96; AHH93; HHWT97a; HHWT97b; Fre05; Fre08; Fre+11; GG09]. Alternatively, they can start with the unsafe states and go back in time, thus to check whether initial states can be reached backwards this way. Many of these works use convex polyhedra in higher dimensions (polytopes), but mechanisms using high-dimensional boxes, ellipsoides, zonotopes, or support functions exist.

Another approach is to overapproximate a given general HA by one of a decidable class [Pre+98]. The new automaton can then be analysed, and its safety implies the one of the original automaton. Other solvers start with a coarse abstraction, such as assuming that the complete state space is reachable, which they then iteratively refine [Cla+03; RS07]. For this, they generate paths in the semantics, which are used as counterexamples to refine the abstraction at the right place, that is, split an abstract state into two or more separate abstract states.

The LTSs, which form the semantical model of formal descriptions of all these classes, have a longer history than HAs themselves. One of the first authors to use LTSs was Keller [Kel76], then for the formal verification of parallel programs. The model plays a role in various contexts, and there are a large number of papers and textbooks on LTSs and their applications [Pet81; BK85; Har87; Hal98; Kro99; BK08].

2.7 Conclusion

In this chapter, we have formalised the basic notions of classical HAs, as well as the common properties one usually wants to solve for such models. We also gave an overview of the literature, formalisms and some special cases of this model. The model class we have described consists of a finite number of modes, along with a number of continuous variables and descriptions of the transitions inside a mode and between modes. The HAs we have described are a continuous-time model, the semantics of which are given as LTSs. The system occupies one mode at a given point of time. While being in a mode, the timed behaviours, that is, the continuous changes in the model variables, are given by the definition of this mode, specified for instance by differential (in)equations. Under certain conditions, we can move from one mode to another mode, while possibly changing the model variables. Transitions of this type do not take time.

Nondeterminism occurs, because the specification of the timed behaviour might allow for different ways of changing the model variables, because there might be different points of time to change from one mode to another, and because there might be different possible modes and different changes to the variables to choose from when performing a mode change. Thus, an HA can allow a very large number of different possible trajectories through the system.

We are interested in checking whether certain unsafe states might be reached, or whether certain desirable states are always reached. For this, we have defined abstractions. These abstractions subsume model states into a finite number of abstract states, thus making HAs subject to automatic analysis. The correctness of such abstractions can be shown by simulation relations. We demonstrated the practical application of this idea by applying one of the leading tools on a case study we have developed throughout the chapter.

This chapter will build the foundation of the following ones, the models of which are extensions of the classical HAs model.

3

Probabilistic Hybrid Automata

This chapter targets computing conservative bounds on reachability properties of a generic class of stochastic hybrid systems. The *probabilistic hybrid automata* which we consider here are an extension of classical hybrid system formalisms, in which the guarded commands may choose their successors according to a probability distribution with finite support [Spr00; Spr01]. Our model thus allows for both probabilistic choice, resulting from uncertainties of the behaviour of the model for which at least a probabilistic estimate is known, as well as nondeterminism, that is, a degree of uncertainty to which we cannot even assign a probability distribution. If we are given a set of unsafe states, we are interested in proving that the maximal probability to reach them does not exceed a certain bound. On the other hand, if we are given a set of desirable states, we are interested in proving that the minimal reachability probability is above the bound. We achieve this target by building on abstractions for classical hybrid automata (cf. Chapter 2), thereby developing the first generic framework which does not require manual intervention to decide properties of system models which contain nondeterminism, complex timed dynamics as well as stochastic behaviour, all of which are relevant for many real-world systems.

The chapter requires some understanding of basic stochastics, but we will state the notations we are going to use. The structure of this chapter corresponds to the one of Chapter 2: in Section 3.1 we extend the labelled transition systems to *probabilistic automata*, which will form the semantics of the probabilistic hybrid automata we consider. In Section 3.2 we discuss the high-level model of probabilistic hybrid automata and their semantics in term of probabilistic automata. Because this model is a conservative extension of nonprobabilistic hybrid automata, we will reuse many formalisms from Chapter 2. In Section 3.3 we will describe an abstraction technology for the new model class. Section 3.4 will discuss how properties can be decided once the abstraction has been computed. In Section 3.5 we will apply our methods using a hybrid systems solver on a probabilistic version of our running example and several other case studies. Section 3.6 discusses related work. Section 3.7 concludes the chapter.

3.1 Stochastic Models

This section will describe probabilistic automata. They constitute the formal semantics of probabilistic hybrid automata, just as the LTSs have served as the formal semantics of classical HAs in Section 2.1. We will recapitulate a few basic notions from probability theory which are needed to interpret probabilistic automata. Then, we will discuss probabilistic automata and the properties we consider.

3.1.1 Stochastic Recapitulation

We consider an arbitrary set Ω, the *sample space*. A family Σ of subsets of Ω is a *σ-algebra*, provided $\Omega \in \Sigma$, and Σ is closed under complement and σ-union (countable union). This means that for each family $\langle A_i \rangle_{i \in I}$ with a countable index set I and $A_i \in \Sigma$ for all $i \in I$, we have $\bigcup_{i \in I} A_i \in \Sigma$. A set $B \in \Sigma$ is then called *measurable*. Given a family of sets $\mathcal{A}$, by $\sigma(\mathcal{A})$ we denote the σ-algebra *generated* by $\mathcal{A}$, that is the smallest σ-algebra containing all sets of $\mathcal{A}$. The *Borel σ-algebra* over Ω is generated by the open subsets of Ω, and it is denoted $\mathcal{B}(\Omega)$. The pair (Ω, Σ) is called a *measurable space*.

A function $\mu \colon \Sigma \to \mathbb{R}_{\geq 0}$ is called *σ-additive* if $\mu(\biguplus_{i \in I} B_i) = \sum_{i \in I} \mu(B_i)$ for countable index sets I and disjoint B_i. Under this condition, we speak of a *measure* if $\mu(\emptyset) = 0$, and if also $\mu(\Omega) = 1$ then μ is a *probability measure*.

Given two measurable spaces (Ω_1, Σ_1) and (Ω_2, Σ_2), a function $f \colon \Omega_1 \to \Omega_2$ is Σ_1-Σ_2-*measurable* if every preimage of a measurable set is measurable, i.e. $f^{-1}(B) \in \Sigma_1$ for all $B \in \Sigma_2$. Given a probability measure μ and a measurable function f, we have the *integral* $\int_\Omega f(\omega)\, \mu(\mathrm{d}\omega)$ of f by the probability measure μ, see e.g. [ADD00] for a thorough definition.

A *(discrete-time) stochastic process* is a function $X \colon (\Omega_1 \times \mathbb{N}) \to \Omega_2$ where for each $n \in \mathbb{N}$ we have that $X(\cdot, n)$ is Σ_1-Σ_2-measurable. We will write X_n to denote $X(\cdot, n)$.

3.1.2 Probabilistic Automata

To describe the behaviour of probabilistic automata, we firstly need to define probability distributions.

Definition 3.1. *A* finite probability distribution *over a set Ω is a function $\mu \colon \Omega \to [0,1]$ where there are only finitely many $a \in \Omega$ with $\mu(a) > 0$, and we have $\sum_{a \in \Omega} \mu(a) = 1$. In a* Dirac *probability distribution μ, there is only a single $a \in \Omega$ with $\mu(a) = 1$. With $Distr(\Omega)$ we denote the set of all finite probability distributions over Ω. Given n elements $a_i \in \Omega$ and probabilities $p_i \geq 0$, $1 \leq i \leq n$ with $\sum_{i=1}^{n} p_i = 1$, we let $[a_1 \mapsto p_1, \ldots, a_n \mapsto p_n]$ denote the probability distribution that chooses $a \in \Omega$ with probability $\sum_{\substack{a_i = a, \\ 1 \leq i \leq n}} p_i$.*

Finite probability distributions can be seen as special cases of probability measures, which explicitly assign a nonzero probability to a number of singleton sets $\{a_i\} \in \Sigma$. The representation $[a_1 \mapsto p_1, \ldots, a_n \mapsto p_n]$ of a probability distribution is not unique.

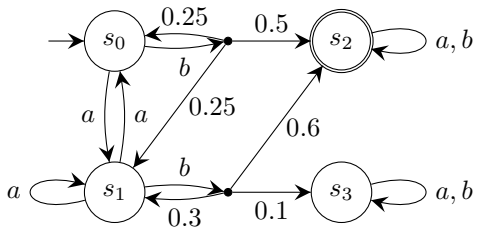


Figure 3.1: Probabilistic automaton.

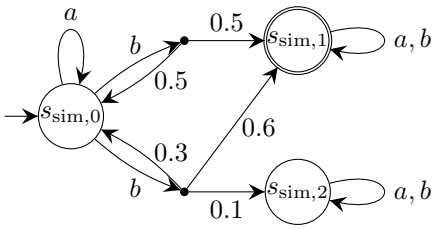


Figure 3.2: Probabilistic automaton simulating the one of Figure 3.1.

Example 3.2. *Consider* $\Omega \stackrel{\text{def}}{=} \mathbb{R}$. *The probability distribution* $\mu\colon \Omega \to [0,1]$ *with* $\mu(0.3) \stackrel{\text{def}}{=} 0.12$, $\mu(0.7) \stackrel{\text{def}}{=} 0.8$, $\mu(\pi) \stackrel{\text{def}}{=} 0.08$ *and* $\mu(\cdot) \stackrel{\text{def}}{=} 0$ *otherwise can be represented as* $[0.3{\mapsto}0.12, 0.7{\mapsto}0.8, \pi{\mapsto}0.08]$. *An alternative representation is for instance* $[0.7{\mapsto}0.3, \pi{\mapsto}0.08, 0.3{\mapsto}0.1, 0.7{\mapsto}0.5, 0.3{\mapsto}0.02]$. △

We can now define the model which will later on form the semantics of our probabilistic hybrid automata.

Definition 3.3. *A* probabilistic automaton (PA) *is a tuple*

$$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}),$$

where

- S *is a set of* states,
- $\overline{s} \in S$ *is the* initial state,
- Act *is a set of actions, and*
- *the* transition matrix $\mathcal{T}\colon (S \times Act) \to 2^{Distr(S)}$ *assigns sets of probability distributions to pairs of states and actions.*

We require that for each $s \in S$ *we have* $\{a \in Act \mid \mathcal{T}(s,a) \neq \emptyset\} \neq \emptyset$.

Like LTSs, PAs contain a (possibly uncountable) set of states, whereof one is initial. The choice of successors in each state s is now so that one selects an action a and a distribution over successor states μ with $\mu \in \mathcal{T}(s,a)$. LTSs can be seen as special cases of PAs, where we only use Dirac distributions: if we are given an LTS transition matrix $\mathcal{T}_{\text{lts}}$ with $s' \in \mathcal{T}_{\text{lts}}(s,a)$, this transition can be represented by $[s'{\mapsto}1] \in \mathcal{T}_{\text{pa}}(s,a)$, where $\mathcal{T}_{\text{pa}}$ is the transition matrix of the PA representation.

Example 3.4. *In Figure 3.1, we depict a finite example PA* $\mathcal{M} \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T})$. *Here, we have* $S \stackrel{\text{def}}{=} \{s_0, s_1, s_2, s_3\}$, $\overline{s} \stackrel{\text{def}}{=} s_0$, $Act = \{a, b\}$, *and*

$$\begin{aligned}
&\mathcal{T}(s_0,a) \stackrel{\text{def}}{=} \{[s_1{\mapsto}1]\},\ \mathcal{T}(s_0,b) \stackrel{\text{def}}{=} \{[s_0{\mapsto}0.25, s_1{\mapsto}0.25, s_2{\mapsto}0.5]\},\\
&\mathcal{T}(s_1,a) \stackrel{\text{def}}{=} \{[s_0{\mapsto}1], [s_1{\mapsto}1]\},\ \mathcal{T}(s_1,b) \stackrel{\text{def}}{=} \{[s_1{\mapsto}0.3, s_2{\mapsto}0.6, s_3{\mapsto}0.1]\},\\
&\mathcal{T}(s_2,a) \stackrel{\text{def}}{=} \mathcal{T}(s_2,b) \stackrel{\text{def}}{=} \{[s_2{\mapsto}1]\},\\
&\mathcal{T}(s_3,a) \stackrel{\text{def}}{=} \mathcal{T}(s_3,b) \stackrel{\text{def}}{=} \{[s_3{\mapsto}1]\}.
\end{aligned}$$ △

As for LTSs, we describe the possible behaviours of PAs by paths.

Definition 3.5. *A* finite path *of a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *is a tuple*

$$\beta_{\text{fin}} = s_0\, a_0\, \mu_0\, \ldots\, s_{n-1}\, a_{n-1}\, \mu_{n-1}\, s_n \in (S \times Act \times Distr(S))^* \times S,$$

where $s_0 = \overline{s}$ *and for all* i *with* $0 \leq i < n$ *we have* $\mu_i \in \mathcal{T}(s_i, a_i)$. *An* infinite path *is a tuple*

$$\beta_{\text{inf}} = s_0\, a_0\, \mu_0\, \ldots \in (S \times Act \times Distr(S))^\omega,$$

where $s_0 = \overline{s}$ *and* $\mu_i \in \mathcal{T}(s_i, a_i)$ *holds for all* $i \geq 0$. *By* $Path^{\text{fin}}_{\mathcal{M}}$ *we denote the set of all finite paths and by* $Path^{\text{inf}}_{\mathcal{M}}$ *we denote the set of all infinite paths of* $\mathcal{M}$.

The length *of a finite path* β_{fin} *is denoted by* $|\beta_{\text{fin}}| \stackrel{\text{def}}{=} n$. *We let* $\beta_{\text{fin}}[i] \stackrel{\text{def}}{=} \beta_{\text{inf}}[i] \stackrel{\text{def}}{=} s_i$ *denote the* $(i+1)$*-th state of a finite or infinite path (for the* i*-s defined). By* $\text{last}(\beta_{\text{fin}}) \stackrel{\text{def}}{=} s_n$ *we denote the last state of a finite path. For* $\beta, \beta' \in Path^{\text{fin}}_{\mathcal{M}} \uplus Path^{\text{inf}}_{\mathcal{M}}$ *we write* $\beta \leq \beta'$ *in case either* $\beta = \beta'$ *or if* β *is a finite prefix of* β'.

We define the trace *of finite paths as* $\text{trace}(\beta_{\text{fin}}) = a_0\, a_1\, \ldots\, a_{n-1}$ *and accordingly for infinite ones. The* distribution trace *is defined as* $\text{distrs}(\beta_{\text{fin}}) = \mu_0\, \mu_1\, \ldots\, \mu_{n-1}$ *and accordingly for infinite paths. The sets of all finite and infinite traces are defined as* $Trace^*_{\mathcal{M}} \stackrel{\text{def}}{=} Act^*$ *and* $Trace^\omega_{\mathcal{M}} \stackrel{\text{def}}{=} Act^\omega$. *Given* $\gamma = a_0\, a_1\, \ldots \in Trace^*_{\mathcal{M}} \uplus Trace^\omega_{\mathcal{M}}$, *we define* $\gamma[i] \stackrel{\text{def}}{=} a_i$ *as the* $(i+1)$*-th action on the trace, and accordingly for distribution traces.*

Consider a subset $Act_{\text{fair}} \subseteq Act$ *of the actions of* $\mathcal{M}$. *We consider a path* $\beta \in Path^{\text{inf}}_{\mathcal{M}}$ *as* Act_{fair}-fair *if there are infinitely many* $i \geq 0$ *with* $\text{trace}(\beta)[i] \in Act_{\text{fair}}$. *By* $Path^{Act_{\text{fair}}}_{\mathcal{M}}$ *we denote the set of all* Act_{fair}*-fair paths of* $\mathcal{M}$.

Example 3.6. *An example of a finite path in the PA of Figure 3.1 is*

$$\beta_1 \stackrel{\text{def}}{=} s_0\, a\, [s_1 \mapsto 1]\, s_1\, a\, [s_1 \mapsto 1]\, s_1\, a\, [s_1 \mapsto 1]\, s_1,$$

and an example for an infinite path is

$$\beta_2 \stackrel{\text{def}}{=} s_0\, a\, ([s_1 \mapsto 1]\, s_1\, a)^\omega.$$

With $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$, *the path* β_2 *is not* Act_{fair}*-fair, but for instance*

$$\beta_3 \stackrel{\text{def}}{=} s_0\, a\, [s_1 \mapsto 1]\, s_1\, a\, [s_1 \mapsto 1]\, s_1\, a\, [s_0 \mapsto 1]\, s_0\, b\, [s_0 \mapsto 0.25, s_1 \mapsto 0.25, s_2 \mapsto 0.5]\, (s_2\, b\, [s_2 \mapsto 1])^\omega$$

is. We have

$$\begin{aligned}
&\text{trace}(\beta_1) = a\, a\, a, \text{trace}(\beta_2) = a^\omega, \text{trace}(\beta_3) = a\, a\, a\, b^\omega,\\
&\text{distrs}(\beta_1) = [s_1 \mapsto 1]\, [s_1 \mapsto 1]\, [s_1 \mapsto 1], \text{distrs}(\beta_2) = [s_1 \mapsto 1]^\omega,\\
&\text{last}(\beta_1) = s_1,\\
&\beta_1[0] = s_0, \beta_1[1] = \beta_1[2] = s_1, \beta_2[15] = s_1.
\end{aligned}$$

△

Contrary to LTSs, the path sets are not sufficient to describe the properties of PAs which we consider completely. For LTSs, there was merely a nondeterministic choice along the paths of a model. For PAs, we have nondeterminism mixed with probabilistic behaviour. Because of this, we need to define a way to resolve the nondeterminism, thus to obtain a purely probabilistic behaviour, which is then subject to stochastic analyses.

Definition 3.7. *A* scheduler *for a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *is a function* $\sigma\colon Path^{\mathrm{fin}}_{\mathcal{M}} \to Distr(Act \times Distr(S))$. *For all* $\beta \in Path^{\mathrm{fin}}_{\mathcal{M}}$ *we require that if* $\sigma(\beta)((a,\mu)) > 0$ *then* $\mu \in \mathcal{T}(\mathrm{last}(\beta), a)$.

A scheduler σ *is called* simple *if it only maps to Dirac distributions and if for all* $\beta, \beta' \in Path^{\mathrm{fin}}_{\mathcal{M}}$ *with* $\mathrm{last}(\beta) = \mathrm{last}(\beta')$ *we have* $\sigma(\beta) = \sigma(\beta')$. *We can interpret it as being of the form* $\sigma\colon S \to (Act \times Distr(S))$.

With $Sched_{\mathcal{M}}$ *we denote the sets of all schedulers of* $\mathcal{M}$ *and with* $Sched^{\mathrm{simple}}_{\mathcal{M}}$ *we denote the subset of simple schedulers.*

In contrast to extensions presented in the sequel in the context of general continuous distributions (cf. Definition 4.5), this definition only allows to use finite-support schedulers. This is done in order to avoid introducing unnecessary measurability issues already at this point where they can be avoided without losing expressive power.

Using a scheduler, we can define a probability measure on the paths of a PA.

Definition 3.8. *We define* $Pr_{\mathcal{M},\sigma}\colon Path^{\mathrm{fin}}_{\mathcal{M}} \to [0,1]$ *for a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and a scheduler* $\sigma\colon Path^{\mathrm{fin}}_{\mathcal{M}} \to Distr(Act \times Distr(S))$*: for a given finite path* $\beta = s_0\, a_0\, \mu_0\, s_1\, a_1\, \mu_1 \ldots s_n \in Path^{\mathrm{fin}}_{\mathcal{M}}$, *we let*

$$Pr_{\mathcal{M},\sigma}(\beta) \stackrel{\mathrm{def}}{=} \sigma(s_0)((a_0,\mu_0))\mu_0(s_1)\sigma(s_1)((a_1,\mu_1))\mu_1(s_2)\cdots\mu_{n-1}(s_n).$$

We define the cylinder

$$Cyl(\beta) \stackrel{\mathrm{def}}{=} \{\beta' \in Path^{\mathrm{inf}}_{\mathcal{M}} \mid \beta \leq \beta'\}$$

of β *as the set of infinite paths which start with the finite path* β. *Then, we let*

$$\Sigma_{\mathcal{M}} \stackrel{\mathrm{def}}{=} \sigma(\{Cyl(\beta) \mid \beta \in Path^{\mathrm{fin}}_{\mathcal{M}}\}),$$

that is the σ*-algebra generated by all cylinders of* $\mathcal{M}$. *We thus obtain the measurable space* $(Path^{\mathrm{inf}}_{\mathcal{M}}, \Sigma_{\mathcal{M}})$.

There is a unique extension [KSK66] of $Pr_{\mathcal{M},\sigma}\colon Path^{\mathrm{fin}}_{\mathcal{M}} \to [0,1]$ *to* $Pr_{\mathcal{M},\sigma}\colon \Sigma_{\mathcal{M}} \to [0,1]$ *where for all* $\beta \in Path^{\mathrm{fin}}_{\mathcal{M}}$ *we have*

$$Pr_{\mathcal{M},\sigma}(Cyl(\beta)) \stackrel{\mathrm{def}}{=} Pr_{\mathcal{M},\sigma}(\beta).$$

Using the definition of a fair path and the probability of paths, we can define fair schedulers.

Definition 3.9. *A scheduler $\sigma \in Sched_{\mathcal{M}}$ of a PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ is called Act_{fair}-*fair *for $Act_{\text{fair}} \subseteq Act$ if*

$$Pr_{\mathcal{M},\sigma}(Path_{\mathcal{M}}^{Act_{\text{fair}}}) = 1.$$

By $Sched_{\mathcal{M}}^{Act_{\text{fair}}}$ we denote the set of all Act_{fair}-fair schedulers of $\mathcal{M}$.

We will now define a number of stochastic processes of a PA.

Definition 3.10. *Let $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ be a PA. We define the* state process*, the* action process *and the* distribution process *as*

$$\begin{aligned} X^{\mathcal{M}}&: (Path_{\mathcal{M}}^{\text{inf}} \times \mathbb{N}) \to S \text{ with } X^{\mathcal{M}}(\beta, n) \stackrel{\text{def}}{=} \beta[n], \\ Y^{\mathcal{M}}&: (Path_{\mathcal{M}}^{\text{inf}} \times \mathbb{N}) \to Act \text{ with } Y^{\mathcal{M}}(\beta, n) \stackrel{\text{def}}{=} \text{trace}(\beta)[n], \\ Z^{\mathcal{M}}&: (Path_{\mathcal{M}}^{\text{inf}} \times \mathbb{N}) \to Distr(S) \text{ with } Z^{\mathcal{M}}(\beta, n) \stackrel{\text{def}}{=} \text{distrs}(\beta)[n], \end{aligned}$$

for all $\beta \in Path_{\mathcal{M}}^{\text{inf}}$ and $n \in \mathbb{N}$.

As for LTSs, we want to consider safety properties, that is we want to prove that an automaton cannot reach certain sets of states. However, in a PA we often find a situation in which it is neither impossible for unsafe states to be reached, nor can they ever be reached with certainty. Instead, we have a *probability* that the set will be reached. Thus, we are interested in proving that the probability is always bounded by a certain threshold $\mathfrak{p}$. As we might obtain different probabilities from different schedulers, we consider the probability to reach the set in the worst case, that is the *maximal* probability. Thus, a PA can be considered safe if the maximal probability is not higher than $\mathfrak{p}$.

Definition 3.11. *Consider a PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$. Given a set $Reach \subseteq S$ of states and a scheduler $\sigma \in Sched_{\mathcal{M}}$, we let*

$$\text{val}_{\mathcal{M},Reach}^{\sigma} \stackrel{\text{def}}{=} Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)$$

denote the reachability probability value *for Reach under σ. Further, we let*

$$\text{val}_{\mathcal{M},Reach}^{+} \stackrel{\text{def}}{=} \sup_{\sigma \in Sched_{\mathcal{M}}} Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)$$

denote the maximal reachability probability *for Reach.*

It is known [BK08, Section 10.6] that $\text{val}_{\mathcal{M},Reach}^{\sigma}$ is well-defined, and thus so is $\text{val}_{\mathcal{M},Reach}^{+}$. We might also have a situation where certain states are desirable and we are given a probabilistic inevitability problem. In this case, we want to prove that the probability to reach these states is not below a given threshold $\mathfrak{p}$. We use fair schedulers here, for the same reason as in the nonprobabilistic case (cf. Definition 2.6), namely because of the possible time convergence in the semantics of (probabilistic) hybrid automata.

Definition 3.12. *Consider a PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$, a set $Reach \subseteq S$ of states and a subset $Act_{\text{fair}} \subseteq Act$ of the actions of $\mathcal{M}$. We let*

$$\text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},Reach} \stackrel{\text{def}}{=} \inf_{\sigma \in Sched^{Act_{\text{fair}}}_{\mathcal{M}}} Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X^{\mathcal{M}}_i \in Reach)$$

denote the minimal Act_{fair}-restricted reachability probability *for $Reach$.*

Example 3.13. *Consider the PA $\mathcal{M}$ of Figure 3.1 and the paths of Example 3.6. We consider the simple scheduler $\sigma_1 \colon S \to (Act \times Distr(S))$ where*

$$\sigma_1(s_0) \stackrel{\text{def}}{=} (a, [s_1 \mapsto 1]),\ \sigma_1(s_1) \stackrel{\text{def}}{=} (a, [s_1 \mapsto 1]),\ \sigma_1(s_2) \stackrel{\text{def}}{=} (a, [s_2 \mapsto 1]),\ \sigma_1(s_3) \stackrel{\text{def}}{=} (a, [s_2 \mapsto 1]),$$

and let $Reach \stackrel{\text{def}}{=} \{s_2\}$ and $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$. We have (with the interpretation of the scheduler as a function $\sigma \colon Path^{\text{fin}}_{\mathcal{M}} \to Distr(Act \times Distr(S))$)

$$\begin{aligned}
&Pr_{\mathcal{M},\sigma}(\beta_1) \\
&= \sigma_1(s_0)((a, [s_1 \mapsto 1]))[s_1 \mapsto 1](s_1) \\
&\quad \sigma_1(s_0\, a\, [s_1 \mapsto 1]\, s_1)((a, [s_1 \mapsto 1]))[s_1 \mapsto 1](s_1) \\
&\quad \sigma_1(s_0\, a\, [s_1 \mapsto 1]\, s_1\, a\, [s_1 \mapsto 1]\, s_1)((a, [s_1 \mapsto 1]))[s_1 \mapsto 1](s_1) \\
&= 1,
\end{aligned}$$

and we have $\text{val}^{\sigma_1}_{\mathcal{M},Reach} = 0$. We remark that σ_1 is not Act_{fair}-fair. Indeed, we have $Pr_{\mathcal{M},\sigma_1}(Path^{Act_{\text{fair}}}_{\mathcal{M}}) = 0$. Another simple scheduler $\sigma_2 \colon S \to (Act \times Distr(S))$ with

$$\begin{aligned}
&\sigma_2(s_0) \stackrel{\text{def}}{=} (b, [s_0 \mapsto 0.25, s_1 \mapsto 0.25, s_2 \mapsto 0.5]),\ \sigma_2(s_1) \stackrel{\text{def}}{=} (a, [s_0 \mapsto 1]), \\
&\sigma_2(s_2) \stackrel{\text{def}}{=} (b, [s_2 \mapsto 1]),\ \sigma_3(s_3) \stackrel{\text{def}}{=} (b, [s_3 \mapsto 1])
\end{aligned}$$

is fair and we have $\text{val}^{\sigma_2}_{\mathcal{M},Reach} = 1$. The minimal reachability probability over all Act_{fair}-fair schedulers is $\text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},Reach} = \frac{6}{7} \approx 0.857$. It can be obtained for instance by the simple scheduler σ_3 with

$$\begin{aligned}
&\sigma_3(s_0) \stackrel{\text{def}}{=} (a, [s_1 \mapsto 1]),\ \sigma_3(s_1) \stackrel{\text{def}}{=} (b, [s_1 \mapsto 0.3, s_2 \mapsto 0.6, s_3 \mapsto 0.1]), \\
&\sigma_3(s_2) \stackrel{\text{def}}{=} (b, [s_2 \mapsto 1]),\ \sigma_3(s_3) \stackrel{\text{def}}{=} (b, [s_3 \mapsto 1]).
\end{aligned}$$

△

Now, as for LTSs, we recall the notion of simulation relations [SL95; Seg95], adapted to reachability properties. A simulation relation requires that every successor distribution of a state of a simulated PA $\mathcal{M}$ is related to a successor distribution of its corresponding state of a simulating PA $\mathcal{M}_{\text{sim}}$ using a *weight function* [JL91, Definition 4.3].

Definition 3.14. *Let $\mu \in Distr(S)$ and $\mu_{\text{sim}} \in Distr(S_{\text{sim}})$ be two distributions. For a relation $R \subseteq S \times S_{\text{sim}}$, a* weight function *for (μ, μ_{sim}) with respect to R is a function $w \colon (S \times S_{\text{sim}}) \to [0, 1]$ with*

1. *$w(s, s_{\text{sim}}) > 0$ implies $(s, s_{\text{sim}}) \in R$,*

2. $\mu(s) = \sum_{s_{\text{sim}} \in S_{\text{sim}}} w(s, s_{\text{sim}})$ *for* $s \in S$, *and*
3. $\mu_{\text{sim}}(s_{\text{sim}}) = \sum_{s \in S} w(s, s_{\text{sim}})$ *for* $s_{\text{sim}} \in S_{\text{sim}}$.

We write $\mu \sqsubseteq_R \mu_{\text{sim}}$ *if and only if there exists a weight function for* (μ, μ_{sim}) *with respect to* R.

Definition 3.15. *Given two PAs* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and* $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$, *we say that* $\mathcal{M}_{\text{sim}}$ simulates $\mathcal{M}$, *denoted by* $\mathcal{M} \preceq \mathcal{M}_{\text{sim}}$, *if and only if there exists a relation* $R \subseteq S \times S_{\text{sim}}$, *which we will call* simulation relation *from now on, where*

1. *we have* $(\overline{s}, \overline{s}_{\text{sim}}) \in R$,
2. *for each* $(s, s_{\text{sim}}) \in R$, $a \in Act$, *and* $\mu \in \mathcal{T}(s, a)$, *there is a distribution* $\mu_{\text{sim}} \in Distr(S_{\text{sim}})$ *with* $\mu_{\text{sim}} \in \mathcal{T}_{\text{sim}}(s_{\text{sim}}, a)$ *and* $\mu \sqsubseteq_R \mu_{\text{sim}}$.

For two sets of states $Reach \subseteq S$ *and* $Reach_{\text{sim}} \subseteq S_{\text{sim}}$, *we call a simulation relation* $Reach$-$Reach_{\text{sim}}$-compatible *if for all* $(s, s_{\text{sim}}) \in R$, *we have* $s \in Reach$ *if and only if* $s_{\text{sim}} \in Reach_{\text{sim}}$. *If there exists such a relation, we write*

$$(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}}).$$

Because one can see LTSs as a special case of PAs, one can also treat the definition of simulations between LTSs of Definition 2.8 as a special case of the above definition, in which only trivial weight functions are used.

Corollary 3.16. *Let* $\mathcal{M}_{\text{lts}} = (S, \overline{s}, Act, \mathcal{T}_{\text{lts}})$ *and* $\mathcal{M}_{\text{lts,sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{lts,sim}})$ *be two LTSs and let* $Reach \subseteq S$ *and* $Reach_{\text{sim}} \subseteq S_{\text{sim}}$ *be two sets of states with*

$$(\mathcal{M}_{\text{lts}}, Reach) \preceq (\mathcal{M}_{\text{lts,sim}}, Reach_{\text{sim}})$$

with the simulation relation R. *Then for the PAs* $\mathcal{M}_{\text{pa}} = (S, \overline{s}, Act, \mathcal{T}_{\text{pa}})$ *and* $\mathcal{M}_{\text{pa,sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{pa,sim}})$ *with*

$$\mathcal{T}_{\text{pa}}(s, a) \stackrel{\text{def}}{=} \{[s' \mapsto 1] \mid s' \in \mathcal{T}_{\text{lts}}(s, a)\} \text{ and } \mathcal{T}_{\text{pa,sim}}(s, a) \stackrel{\text{def}}{=} \{[s' \mapsto 1] \mid s' \in \mathcal{T}_{\text{lts,sim}}(s, a)\}$$

for the corresponding states s *and actions* a *we have*

$$(\mathcal{M}_{\text{pa}}, Reach) \preceq (\mathcal{M}_{\text{pa,sim}}, Reach_{\text{sim}}).$$

For this, we use the same simulation relation R *and weight function* w *with* $w(s, s_{\text{sim}}) \stackrel{\text{def}}{=} 1$ *if* $(s, s_{\text{sim}}) \in R$ *and* $w(\cdot, \cdot) \stackrel{\text{def}}{=} 0$ *else.*

The corollary follows immediately by checking that the requirements of Definition 3.14 and Definition 3.15 are fulfilled.

As for LTSs, we can use simulation relations to prove the safety of PAs indirectly using a simulating PA.

Lemma 3.17. *Consider two PAs* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and* $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ *and sets of states* $Reach \subseteq S$ *and* $Reach_{\text{sim}} \subseteq S_{\text{sim}}$ *with* $(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}})$. *Then for each* $\sigma \in Sched_{\mathcal{M}}$ *there is* $\sigma_{\text{sim}} \in Sched_{\mathcal{M}_{\text{sim}}}$ *with*

$$\text{val}^{\sigma}_{\mathcal{M}, Reach} = \text{val}^{\sigma_{\text{sim}}}_{\mathcal{M}_{\text{sim}}, Reach_{\text{sim}}} \text{ and thus } \text{val}^{+}_{\mathcal{M}, Reach} \leq \text{val}^{+}_{\mathcal{M}_{\text{sim}}, Reach_{\text{sim}}}.$$

Proof. Consider the PAs

$$\mathcal{M}' \stackrel{\text{def}}{=} (S, \overline{s}, Act \uplus \{a_{Reach}\}, \mathcal{T}') \text{ and } \mathcal{M}'_{\text{sim}} \stackrel{\text{def}}{=} (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act \uplus \{a_{Reach}\}, \mathcal{T}'_{\text{sim}}),$$

where for all $s \in S$ we have

- $\mathcal{T}'(s,a) \stackrel{\text{def}}{=} \mathcal{T}(s,a)$ for $a \in Act$,
- if $s \in Reach$ then $\mathcal{T}'(s, a_{Reach}) = \{[s \mapsto 1]\}$, and else $\mathcal{T}'(s, a_{Reach}) = \emptyset$,

and accordingly for $\mathcal{T}'_{\text{sim}}$.
Thus, these models extend $\mathcal{M}$ and $\mathcal{M}_{\text{sim}}$ by a self-loop action in the *Reach* states. Given a scheduler σ for $\mathcal{M}$, we can construct a corresponding scheduler σ' for $\mathcal{M}'$ with

$$\text{val}^{\sigma}_{\mathcal{M},Reach} = \text{val}^{\sigma'}_{\mathcal{M}',Reach}, \tag{3.1}$$

where for $\beta \in Path^{\text{fin}}_{\mathcal{M}'}$ we let $\sigma'(\beta) \stackrel{\text{def}}{=} [(a_{Reach}, [\text{last}(\beta) \mapsto 1]) \mapsto 1]$ if $\text{last}(\beta) \in Reach$ and $\sigma'(\beta) \stackrel{\text{def}}{=} \sigma(\beta)$ else, and accordingly for $\mathcal{M}_{\text{sim}}$ and $\mathcal{M}'_{\text{sim}}$. A similar construction for the reverse direction is also possible. We have

$$\text{val}^{\sigma'}_{\mathcal{M}',Reach} = Pr_{\mathcal{M}',\sigma'}(\exists i \geq 0.\ Y^{\mathcal{M}'} = a_{Reach}), \tag{3.2}$$

and a similar result holds for $\mathcal{M}_{\text{sim}}$ and $\mathcal{M}'_{\text{sim}}$. We have $\mathcal{M}' \preceq \mathcal{M}'_{\text{sim}}$ with the same relation R as between $\mathcal{M}$ and $\mathcal{M}_{\text{sim}}$. By [Seg95, Proposition 7.7.1], for all $\sigma' \in Sched_{\mathcal{M}'}$ there is $\sigma'_{\text{sim}} \in Sched_{\mathcal{M}'_{\text{sim}}}$ for which measures on traces agree. In turn, we have

$$Pr_{\mathcal{M}',\sigma'}(\exists i \geq 0.\ Y^{\mathcal{M}'} = a_{Reach}) = Pr_{\mathcal{M}'_{\text{sim}},\sigma'_{\text{sim}}}(\exists i \geq 0.\ Y^{\mathcal{M}'_{\text{sim}}} = a_{Reach}). \tag{3.3}$$

In total, from Equation 3.1, Equation 3.2 and Equation 3.3 we conclude

$$\text{val}^{\sigma}_{\mathcal{M},Reach} = \text{val}^{\sigma_{\text{sim}}}_{\mathcal{M}_{\text{sim}},Reach_{\text{sim}}}. \qquad \square$$

A corresponding result holds for probabilistic inevitability properties.

Lemma 3.18. *Consider two PAs $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ and $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ and sets of states $Reach \subseteq S$ and $Reach_{\text{sim}} \subseteq S_{\text{sim}}$ with $(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}})$ and $Act_{\text{fair}} \subseteq Act$. Then for each $\sigma \in Sched^{Act_{\text{fair}}}_{\mathcal{M}}$ there is $\sigma_{\text{sim}} \in Sched^{Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}}}$ with*

$$\text{val}^{\sigma}_{\mathcal{M},Reach} = \text{val}^{\sigma_{\text{sim}}}_{\mathcal{M}_{\text{sim}},Reach_{\text{sim}}} \text{ and thus } \text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},Reach} \geq \text{val}^{-,Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}},Reach_{\text{sim}}}.$$

Proof. The proof follows along the lines of the one of Lemma 3.17. The only notable difference is that we are reasoning about fair schedulers. This is no problem however: for fairness to hold in $\mathcal{M}'$, we must have $Pr_{\mathcal{M}',Reach}(Path^{Act'_{\text{fair}}}_{\mathcal{M}'}) = 1$ with $Act'_{\text{fair}} \stackrel{\text{def}}{=} Act_{\text{fair}} \uplus \{a_{Reach}\}$. We used a scheduler σ'_{sim} for which the measures on traces agree [Seg95, Proposition 7.7.1], which then implies $Pr_{\mathcal{M}'_{\text{sim}},Reach_{\text{sim}}}(Path^{Act'_{\text{fair}}}_{\mathcal{M}'_{\text{sim}}}) = 1$. $\square$

Example 3.19. *Consider the PAs $\mathcal{M}$ and $\mathcal{M}_{\text{sim}}$ depicted in Figure 3.1 and Figure 3.2. Let $Reach \stackrel{\text{def}}{=} \{s_2\}$ and $Reach_{\text{sim}} \stackrel{\text{def}}{=} \{s_{\text{sim},1}\}$. We have $(\mathcal{M}, Reach) \preceq (\mathcal{M}_{\text{sim}}, Reach_{\text{sim}})$, as seen from the Reach-$Reach_{\text{sim}}$-compatible simulation relation*

$$R \stackrel{\text{def}}{=} \{(s_0, s_{\text{sim},0}), (s_1, s_{\text{sim},0}), (s_2, s_{\text{sim},1}), (s_3, s_{\text{sim},2})\}.$$

With $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$ we have

$$\text{val}^{+}_{\mathcal{M}_{\text{sim}}, Reach_{\text{sim}}} = 1 \text{ and } \text{val}^{-, Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}}, Reach_{\text{sim}}} = \frac{6}{7} \approx 0.857.$$

From this, we can conclude that

$$\text{val}^{+}_{\mathcal{M}, Reach} \leq 1 \text{ and } \text{val}^{-, Act_{\text{fair}}}_{\mathcal{M}, Reach} \geq \frac{6}{7} \approx 0.857,$$

which is consistent with what we observed in Example 3.13. △

3.2 Probabilistic Hybrid Automata

We can now define probabilistic hybrid automata [Spr01].

Definition 3.20. *A* probabilistic hybrid automaton (PHA) *is a tuple*

$$\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds),$$

where

- *M is a finite set of* modes*,*
- *$k \in \mathbb{N}^+$ is the* dimension *of the automaton,*
- *$\overline{m} \in M$ is the* initial mode*,*
- *$Post_m$ is a k-dimensional post operator (cf. Definition 2.12) for each mode m,*
- *$Cmds$ is a finite set of* probabilistic guarded commands *of the form*

$$g \to p_1 : u_1 + \ldots + p_n : u_n,$$

  where
  - *$g \subseteq M \times \mathbb{R}^k$ is a* guard*,*
  - *we have $p_i \geq 0$ for $1 \leq i \leq n$,*
  - *we have $\sum_{i=1}^{n} p_i = 1$,*
  - *$u_i \colon (M \times \mathbb{R}^k) \to 2^{M \times \mathbb{R}^k}$ is an* update function *for $1 \leq i \leq n$,*
  - *if $s \in g$ then $u_i(s) \neq \emptyset$ for $1 \leq i \leq n$, and*
- *for $s = (m, v)$ with $Post_m(v, \mathtt{t}) = \{v\}$ for all $\mathtt{t} \in \mathbb{R}_{\geq 0}$, there is a command with guard g with $s \in g$.*

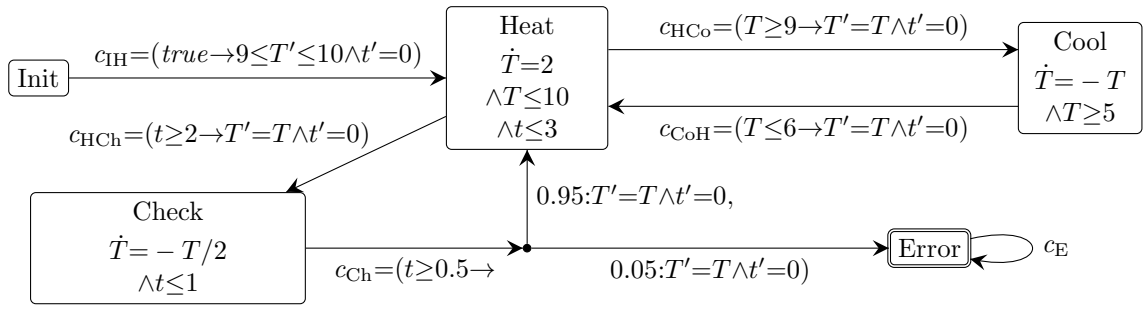


Figure 3.3: PHA modelling a thermostat.

The main difference to the HAs of Definition 2.14 is thus in the definition of the commands: instead of allowing a nondeterministic choice over possible successor states in case its guard is fulfilled, a probabilistic guarded command allows a nondeterministic choice over a number of finite probability distributions. Classical HAs can be seen as special cases of PHAs: we can consider each nonprobabilistic guarded command $g \to u$ as a probabilistic guarded command $g \to 1 : u$.

Example 3.21. *In Figure 3.3 we give a PHA which is a probabilistic variant of the HA of Figure 2.4. The main difference to the former automaton is, that in* Check *we no longer have a nondeterministic choice of whether to move to* Error *resulting from two commands* c_{ChH} *and* c_{ChE}*. Instead, when the command* c_{Ch} *is executed, we move to* Heat *with a probability of* 0.95 *where the execution is continued, or move to the* Error *mode with a probability of* 0.05*, that is we have*

$$c_{\mathrm{Ch}} = (g \to 0.95 : u_{\mathrm{ChH}} + 0.05 : u_{\mathrm{ChE}}),$$

where g*,* u_{ChH} *and* u_{ChE} *are as in Example 2.15. The other commands are as before, that is they only involve Dirac distributions.* △

The semantics of a PHA in terms of the PA model we defined beforehand is as follows.

Definition 3.22. *The* semantics *of a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *is a PA*

$$[\![\mathcal{H}]\!] \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T}),$$

where

- $S \stackrel{\text{def}}{=} M \times \mathbb{R}^k$,
- $\overline{s} \stackrel{\text{def}}{=} (\overline{m}, 0, \ldots, 0)$,
- $Act \stackrel{\text{def}}{=} \mathbb{R} \uplus Cmds$,
- *for* $s = (m, v) \in S$ *we have*
  - *for* $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ *we have* $\mathcal{T}(s, c) \stackrel{\text{def}}{=} \emptyset$ *if* $s \notin g$ *and else:*

$$\mathcal{T}(s, c) \stackrel{\text{def}}{=} \{[s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \mid s'_1 \in u_1(s), \ldots, s'_n \in u_n(s)\},$$

– *for* $\mathsf{t} \in \mathbb{R}_{\geq 0}$ *we have*

$$\mathcal{T}(s, \mathsf{t}) \stackrel{\text{def}}{=} \{[(m, v') \mapsto 1] \mid v' \in \mathit{Post}_m(v, \mathsf{t})\}.$$

Given a time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$*, the* time-restricted semantics *is the PA* $[\![\mathcal{H}, \mathbf{T}]\!] \stackrel{\text{def}}{=} (S, \overline{s}, \mathit{Act}, \mathcal{T})$ *where* S*,* $\overline{s}$*,* Act *and* $\mathcal{T}(s, c)$ *for each* $c \in \mathit{Cmds}$ *are as in the semantics above. We restrict for* $s = (m, v) \in S$ *the timed transitions, so that for* $\mathsf{t} \in \mathbb{R}_{\geq 0}$ *and* $\mathsf{t}_{\mathbf{T}} \stackrel{\text{def}}{=} \min\{\mathsf{t}, \mathbf{t}_m(v)\}$ *we have* $\mathcal{T}(s, \mathsf{t}) \stackrel{\text{def}}{=} \{[(m, v') \mapsto 1] \mid v' \in \mathit{Post}_m(v, \mathsf{t}_{\mathbf{T}})\}$.

The semantics is similar to the one of the nonprobabilistic case in Definition 2.17. The difference is in the probabilistic guarded commands, where we can have distributions other than Dirac. In the definition of $\mathcal{T}(s, c)$, we take advantage of the feature that $[s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n]$ will add up probabilities p_i and p_j if $s'_i = s'_j \in u_i(s) \cap u_j(s)$ for $i \neq j$.

Example 3.23. *Consider the PHA* $\mathcal{H}$ *of Figure 3.3. Then we have*

$$[\![\mathcal{H}]\!] = (S, \overline{s}, \mathit{Act}, \mathcal{T}),$$

where S*,* $\overline{s}$ *and* Act *are as in Example 2.18. For* $\mathcal{T} \colon (S \times \mathit{Act}) \to 2^{\mathit{Distr}(S)}$*, we exemplify the behaviour on the states of the mode* Check*. For* $s = (\mathrm{Check}, t, T, c) \in \{\mathrm{Check}\} \times \mathbb{R}^3$ *we have*

$$\mathcal{T}(s, c_{\mathrm{IH}}) = \mathcal{T}(s, c_{\mathrm{HCo}}) = \mathcal{T}(s, c_{\mathrm{HCh}}) = \mathcal{T}(s, c_{\mathrm{CoH}}) = \mathcal{T}(s, c_{\mathrm{E}}) = \emptyset,$$

$$\mathcal{T}(s, c_{\mathrm{Ch}}) = \begin{cases} \emptyset & \text{if } t < 0.5, \\ \{[(\mathrm{Heat}, 0, T, c) \mapsto 0.95, (\mathrm{Error}, 0, T, c) \mapsto 0.05]\} & \text{else}, \end{cases}$$

and with $\mathsf{t} \in \mathbb{R}_{\geq 0}$ *we have*

$$\mathcal{T}(s, \mathsf{t}) = \left\{ [(\mathrm{Check}, t + \mathsf{t}', T', c + \mathsf{t}') \mapsto 1] \mid T' = T \exp\left(\frac{-\mathsf{t}'}{2}\right) \right\},$$

where

$$\mathsf{t}' \stackrel{\text{def}}{=} \max(\{\mathsf{t}' \mid \mathsf{t}' + t \leq 1 \wedge x + \mathsf{t}' \leq \mathfrak{T} \wedge \mathsf{t}' \leq \mathsf{t}\} \cup \{0\}).$$

For the time-restricted semantics $[\![\mathcal{H}, \mathbf{T}]\!]$ *with* $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$*, the timed transitions are as before, except that we have*

$$\mathsf{t}' \stackrel{\text{def}}{=} \max(\{\mathsf{t}' \mid \mathsf{t}' + t \leq 1 \wedge c + \mathsf{t}' \leq \mathfrak{T} \wedge \mathsf{t}' \leq \mathsf{t} \wedge \mathsf{t}' \leq \mathbf{t}_{\mathrm{Check}}(t, T, c)\} \cup \{0\}). \qquad \triangle$$

We can now define minimal and maximal reachability probabilities of PHA.

Definition 3.24. *Let* $\mathcal{H} = (M, k, \overline{m}, \langle \mathit{Post}_m \rangle_{m \in M}, \mathit{Cmds})$ *be a PHA with a mode* m_{Reach}*. We define*

$$\mathrm{val}^{+}_{\mathcal{H}, m_{\mathit{Reach}}} \stackrel{\text{def}}{=} \mathrm{val}^{+}_{[\![\mathcal{H}]\!], \{m_{\mathit{Reach}}\} \times \mathbb{R}^k} \text{ and } \mathrm{val}^{-}_{\mathcal{H}, m_{\mathit{Reach}}} \stackrel{\text{def}}{=} \mathrm{val}^{-, \mathit{Cmds}}_{[\![\mathcal{H}]\!], \{m_{\mathit{Reach}}\} \times \mathbb{R}^k}.$$

As for nonprobabilistic HAs, for the properties of interest it does not matter whether or not we use the time restriction. To prove this, we firstly define time-restricted paths and schedulers.

Definition 3.25. *Consider a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *with a time restriction* $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$. *Let* $\beta \in Path^{\mathrm{fin}}_{[\![\mathcal{H}]\!]}$ *be a finite path of the semantics without time restriction. We define the tuple* $\mathbf{T}^{\mathrm{fin}}(\beta)$ *of* time-restricted paths *inductively: for a state* $s \in S$ *we let*

$$\mathbf{T}^{\mathrm{fin}}(s) \stackrel{\mathrm{def}}{=} \langle s \rangle.$$

For $c \in Cmds$ *and* $\mu \in Distr(S)$, *we have*

$$\mathbf{T}^{\mathrm{fin}}(\beta\, c\, \mu\, s) \stackrel{\mathrm{def}}{=} \langle \mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta))\, c\, \mu\, s \,\rangle,$$

where for a tuple C *by* $\mathrm{last}(C)$ *we denote the last element of* C. *For* $\mathtt{t} \in \mathbb{R}_{\geq 0}$, *we have*

$$\begin{aligned}\mathbf{T}^{\mathrm{fin}}(\beta\, \mathtt{t}\, s) \stackrel{\mathrm{def}}{=} \langle & \mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta))\, \mathtt{t}\, \mu_0\, s_1, \\ & \mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta))\, \mathtt{t}\, \mu_0\, s_1\, \mathtt{t}\, \mu_1\, s_2, \\ & \ldots \\ & \mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta))\, \mathtt{t}\, \mu_0\, s_1\, \mathtt{t}\, \mu_1\, s_2\, \ldots\, \mathtt{t}'\, \mu'\, s \rangle,\end{aligned}$$

where

- $s_0 = (m, v_0) = \mathrm{last}(\beta')$ *with* $\beta' = \mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta))$,
- $s_n = (m, v_n) = s$,
- $\mathtt{t}' = \mathtt{t} - \sum_{i=0}^{n-1} \mathbf{t}_m(v_i)$, *and*
- *for all* i *with* $0 \leq i < n$ *we have* $s_{i+1} \in Post_m(v_i, \mathbf{t}_m(v_i))$ *and* $\mu_i = [s_{i+1} \mapsto 1]$.

For an infinite path $\beta \in Path^{\mathrm{inf}}_{[\![\mathcal{H}]\!]}$, *we define* $\mathbf{T}^{\mathrm{inf}}(\beta)$ *so that for* $n \in \mathbb{N}$ *we have*

$$(\mathbf{T}^{\mathrm{inf}}(\beta))[n] \stackrel{\mathrm{def}}{=} (\mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta')))[n],$$

where $\beta' \in Path^{\mathrm{fin}}_{[\![\mathcal{H}]\!]}$ *is a finite path with* $\beta' \leq \beta$ *and* $|\mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta'))| \geq n$.
The time-restricted scheduler $\mathbf{T}(\sigma) \in Sched_{[\![\mathcal{H},\mathbf{T}]\!]}$ *of the scheduler* $\sigma \in Sched_{[\![\mathcal{H}]\!]}$ *is defined as follows. For* $\beta' = \mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta))$ *we have*

$$\mathbf{T}(\sigma)(\beta') \stackrel{\mathrm{def}}{=} \sigma(\beta),$$

whereas for $\beta'\, \mathtt{t}\, \mu\, s \in \mathbf{T}^{\mathrm{fin}}(\beta)$ *with* $\beta'\, \mathtt{t}\, \mu\, s \neq \mathrm{last}(\mathbf{T}^{\mathrm{fin}}(\beta))$ *we let*

$$\mathbf{T}(\sigma)(\beta')((\mathtt{t}, [s \mapsto 1])) \stackrel{\mathrm{def}}{=} 1.$$

In the above, we assume we always have

$$Pr_{[\![\mathcal{H}]\!],\sigma}(\beta') = 0,$$

that is the intermediate paths were originally chosen with probability zero. If this is not the case, we can change $\mathbf{T}^{\mathrm{fin}}$ *to choose different (shorter) paths, in order to fulfil this requirement.*

In the above definition, $\mathbf{T}^{\text{fin}}(\beta)$ divides a general finite path into a sequence of finite paths which are extensions of each other, implying that each of them only chooses time durations respecting the time restriction. If β is an infinite path, $\mathbf{T}^{\text{inf}}(\beta)$ is an infinite path which respects the time restrictions. The time-restricted scheduler $\mathbf{T}(\sigma)$ chooses the last path of $\mathbf{T}^{\text{fin}}(\beta)$ with the same probability as σ did for β.

Using this construction, we can show that the following holds.

Lemma 3.26. *Given a PHA $\mathcal{H}$ with a mode m_{Reach}, Reach $\stackrel{\text{def}}{=}$ $\{m_{Reach}\} \times \mathbb{R}^k$ and commands Cmds, and a time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$ we have*

$$\text{val}^{+}_{[\![\mathcal{H}]\!],Reach} = \text{val}^{+}_{[\![\mathcal{H},\mathbf{T}]\!],Reach} \text{ and } \text{val}^{-,Cmds}_{[\![\mathcal{H}]\!],Reach} = \text{val}^{-,Cmds}_{[\![\mathcal{H},\mathbf{T}]\!],Reach}.$$

Proof. We prove $\text{val}^{+}_{[\![\mathcal{H}]\!],Reach} = \text{val}^{+}_{[\![\mathcal{H},\mathbf{T}]\!],Reach}$. Firstly, we show $\text{val}^{+}_{[\![\mathcal{H}]\!],Reach} \geq \text{val}^{+}_{[\![\mathcal{H},\mathbf{T}]\!],Reach}$. Consider an arbitrary scheduler σ of $[\![\mathcal{H},\mathbf{T}]\!]$. As in the proof for Lemma 2.21, we can assume that for each state the scheduler chooses only time durations up to the time restriction; for other schedulers, we can construct an equivalent scheduler by choosing the maximal time duration for each state no larger than the time restriction. Each such scheduler is also one of $[\![\mathcal{H}]\!]$, and it induces the same stochastic behaviour in both models.

We now show $\text{val}^{+}_{[\![\mathcal{H}]\!],Reach} \leq \text{val}^{+}_{[\![\mathcal{H},\mathbf{T}]\!],Reach}$. Consider an arbitrary scheduler $\sigma \in Sched_{[\![\mathcal{H}]\!]}$ of $[\![\mathcal{H}]\!]$ and let $\sigma_{\mathbf{T}} \stackrel{\text{def}}{=} \mathbf{T}(\sigma)$. For all $A \in \Sigma_{[\![\mathcal{H}]\!]}$ we have

$$Pr_{[\![\mathcal{H}]\!],\sigma}(A) = Pr_{[\![\mathcal{H},\mathbf{T}]\!],\sigma_{\mathbf{T}}}(\mathbf{T}^{\text{inf}}(A)), \tag{3.4}$$

and for $\overline{Path^{\text{inf}}_{[\![\mathcal{H},\mathbf{T}]\!]}} \stackrel{\text{def}}{=} \mathbf{T}^{\text{inf}}(Path^{\text{inf}}_{[\![\mathcal{H}]\!]}) \subseteq Path^{\text{inf}}_{[\![\mathcal{H},\mathbf{T}]\!]}$ we have

$$Pr_{[\![\mathcal{H},\mathbf{T}]\!],\sigma_{\mathbf{T}}}(\overline{Path^{\text{inf}}_{[\![\mathcal{H},\mathbf{T}]\!]}}) = 1. \tag{3.5}$$

Because of this, we have

$$\begin{aligned}
&\text{val}^{\sigma}_{[\![\mathcal{H}]\!],Reach} \\
&= Pr_{[\![\mathcal{H}]\!],\sigma}(\exists i \geq 0.\ X^{[\![\mathcal{H}]\!]} \in Reach) \\
&\stackrel{\text{Eqn. 3.4}}{=} Pr_{[\![\mathcal{H},\mathbf{T}]\!],\sigma_{\mathbf{T}}}(\mathbf{T}^{\text{inf}}(\exists i \geq 0.\ X^{[\![\mathcal{H}]\!]} \in Reach)) \\
&\stackrel{\text{Eqn. 3.5}}{=} Pr_{[\![\mathcal{H},\mathbf{T}]\!],\sigma_{\mathbf{T}}}(\exists i \geq 0.\ X^{[\![\mathcal{H},\mathbf{T}]\!]} \in Reach) \\
&= \text{val}^{\sigma_{\mathbf{T}}}_{[\![\mathcal{H},\mathbf{T}]\!],Reach}.
\end{aligned}$$

The proof for $\text{val}^{-,Cmds}_{[\![\mathcal{H}]\!],Reach} = \text{val}^{-,Cmds}_{[\![\mathcal{H},\mathbf{T}]\!],Reach}$ is similar, as the transformation used maintains that the set of fair paths has a probability measure of 1. □

Example 3.27. *Consider the PHA $\mathcal{H}$ of Figure 3.3 with the semantics in Example 3.23, consider the unsafe mode* Error *and let Reach $\stackrel{\text{def}}{=}$ $\{\text{Error}\} \times \mathbb{R}^3$. The probability with which $\mathcal{H}$ can reach* Error *depends on the time bound $\mathfrak{T}$. If $\mathfrak{T}$ is small enough it is completely safe. Consider however $\mathfrak{T} \stackrel{\text{def}}{=} 5$. Then, the path*

$$(\text{Init}, 0, 0, 0)\ c_{\text{IH}}\ [(\text{Heat}, 0, 9, 0) \mapsto 1]\ (\text{Heat}, 0, 9, 0)\ 0.5\ [(\text{Heat}, 0.5, 10, 0.5) \mapsto 1]$$

$(\text{Heat}, 0.5, 10, 0.5)\ c_{\text{HCo}}\ [(\text{Cool}, 0, 10, 0.5){\mapsto}1]\ (\text{Cool}, 0, 10, 0.5)\ 0.69$
$[(\text{Cool}, 0.69, 5.016, 1.19){\mapsto}1]\ (\text{Cool}, 0.69, 5.016, 1.19)\ c_{\text{CoH}}\ [(\text{Heat}, 0, 5.016, 1.19){\mapsto}1]$
$(\text{Heat}, 0, 5.016, 1.19)\ 2\ [(\text{Heat}, 2, 9.016, 3.19){\mapsto}1]\ (\text{Heat}, 2, 9.016, 3.19)\ c_{\text{HCh}}$
$[(\text{Check}, 0, 9.016, 3.19){\mapsto}1]\ (\text{Check}, 0, 9.016, 3.19)\ 0.5\ [(\text{Check}, 0.5, 7.016, 3.69){\mapsto}1]$
$(\text{Check}, 0.5, 7.016, 3.69)\ c_{\text{Ch}}\ [(\text{Heat}, 0, 7.016, 3.69){\mapsto}0.95, (\text{Error}, 0, 7.016, 3.69){\mapsto}0.05]$
$(\text{Error}, 0, 7.016, 3.69)$

(where the numbers are only approximate) leads from the initial state to the unsafe mode. This shows that the maximal reachability probability must be at least 0.05*. Then, in the case that in the last part of the path the system moves to* $(\text{Heat}, 0, 7.016, 3.69)$*, which happens with a probability of* 0.95*, there is not enough time left for a second try to reach* Error*. Indeed, the maximal probability to reach* Error *is* 0.05*.*

Similarly to the nonprobabilistic case, the minimal probability over Cmds-restricted schedulers to reach Error *is* 0*, as we cannot force a move to* Check*. However, if we started in the mode* Check*, we would always end up in* Error *with nonzero probability with each Cmds-fair path, because* c_{Ch} *is the only element of Cmds left in* Check*. But if we had not restricted to Cmds-fair paths, the automaton would still not be guaranteed to reach* Error*, because paths in which we repeatedly choose to let some time pass would be allowed.* △

3.3 Abstractions

In this section, we extend the definition of abstractions to the probabilistic hybrid case. We will also show how we can compute a valid abstraction for a given PHA by using methods from the nonprobabilistic setting.

We will need to transfer probability distributions over the states of the PHA semantics to the states of abstractions.

Definition 3.28. *Consider an abstract state space* $\mathbf{A} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ *of state space* $S = M \times \mathbb{R}^k$ *and a distribution* $[s_1{\mapsto}p_1, \ldots, s_n{\mapsto}p_n] \in \mathit{Distr}(S)$*. The set of* lifted distributions *is given as*

$$\mathit{Lift}_{\mathbf{A}}(\mu) \stackrel{\text{def}}{=} \{[\mathbf{z}_1{\mapsto}p_1, \ldots, \mathbf{z}_n{\mapsto}p_n] \mid \mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbf{A} \wedge s_1 \in \mathbf{z}_1, \ldots, s_n \in \mathbf{z}_n\}.$$

If $\mathbf{A}$ is a partitioning of the state space, the set $\mathit{Lift}_{\mathbf{A}}$ is a singleton, but if $\mathbf{A}$ contains overlapping states, $\mathit{Lift}_{\mathbf{A}}(\mu)$ can indeed contain multiple elements.

Example 3.29. *Consider a probability distribution* $\mu\colon (\{m\} \times \mathbb{R}) \to [0, 1]$ *for which* $\mu((m, 0.3)) \stackrel{\text{def}}{=} 0.12$*,* $\mu((m, 0.7)) \stackrel{\text{def}}{=} 0.8$ *and* $\mu((m, \pi)) \stackrel{\text{def}}{=} 0.08$ *and* $\mu(\cdot) \stackrel{\text{def}}{=} 0$ *else. Consider the partitioning* $\mathbf{A} \stackrel{\text{def}}{=} \{A, B\}$ *with* $A = (m, (-\infty, 0.8])$ *and* $B = (m, (0.8, \infty))$*. Then we have*

$$\mathit{Lift}_{\mathbf{A}}(\mu) = \{[A{\mapsto}0.12, A{\mapsto}0.8, B{\mapsto}0.08]\} = \{[A{\mapsto}0.92, B{\mapsto}0.08]\}.$$

However, with the abstract state space $\mathbf{A}' \stackrel{\text{def}}{=} \{A', B', C'\}$ *with* $A' \stackrel{\text{def}}{=} (m, (-\infty, 0.8))$, $B' \stackrel{\text{def}}{=} (m, (0.5, 2])$ *and* $C' \stackrel{\text{def}}{=} (m, (2, \infty))$ *(which is not a partitioning because* A' *and* B' *overlap) we have*

$$\mathit{Lift}_{\mathbf{A}'}(\mu) = \{[A' \mapsto 0.92, C' \mapsto 0.08], [A' \mapsto 0.8, B' \mapsto 0.12, C' \mapsto 0.08]\}. \qquad \triangle$$

An abstraction of a PHA is defined as follows.

Definition 3.30. *Consider a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle \mathit{Post}_m \rangle_{m \in M}, \mathit{Cmds})$*, an abstract state space* $\mathbf{A} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ *of corresponding dimension and modes as well as a time restriction* $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$*. We say that*

$$\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, \mathit{Cmds} \uplus \{\tau\}, \mathcal{T})$$

is an abstraction *of* $\mathcal{H}$ *using* $\mathbf{A}$ *and* $\mathbf{T}$ *if*

- $(\overline{m}, 0, \ldots, 0) \in \overline{\mathbf{z}}$*,*
- *for all* $\mathbf{z} \in \mathbf{A}$*,* $s \in \mathbf{z}$*,* $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in \mathit{Cmds}$*, if* $s \in \mathbf{z} \cap g$ *then for all*
$$\mu \in \{[s_1' \mapsto p_1, \ldots, s_n' \mapsto p_n] \mid s_1' \in u_1(s), \ldots, s_n' \in u_n(s)\}$$
there is $\mu_{\mathbf{A}} \in \mathit{Lift}_{\mathbf{A}}(\mu)$ *with* $\mu_{\mathbf{A}} \in \mathcal{T}(\mathbf{z}, c)$*,*
- *for all* $\mathbf{z} \in \mathbf{A}$*,* $s = (m, v) \in \mathbf{z}$*,* $\mathtt{t} \in \mathbb{R}_{\geq 0}$ *with* $\mathtt{t} \leq \mathbf{t}_m(v)$ *and all* $s' = (m, v') \in \mathit{Post}_m(s, \mathtt{t})$*, if* $s' \notin \mathbf{z}$ *then we require that there is* $\mathbf{z}' \in \mathbf{A}$ *with* $s' \in \mathbf{z}'$ *and* $[\mathbf{z}' \mapsto 1] \in \mathcal{T}(\mathbf{z}, \tau)$*.*

By $\mathit{Abs}(\mathcal{H}, \mathbf{A}, \mathbf{T})$ *we denote the set of all such abstractions.*

We can then show that abstractions preserve safety properties in a similar way as in the nonprobabilistic case.

Theorem 3.31. *Consider a PHA* $\mathcal{H}$*, an abstraction* $\mathcal{M} \in \mathit{Abs}(\mathcal{H}, \mathbf{A}, \mathbf{T})$ *and an unsafe mode* m_{Reach}*. Let* $\mathit{Reach} \stackrel{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{\mathit{Reach}}\}$*. Then*

$$\mathrm{val}^+_{\mathcal{H}, m_{\mathit{Reach}}} \leq \mathrm{val}^+_{\mathcal{M}, \mathit{Reach}}.$$

Proof. From Lemma 3.26 we know that we only need to prove that the maximal reachability probability in the time-restricted semantics $[\![\mathcal{H}, \mathbf{T}]\!]$ is lower than $\mathrm{val}^+_{\mathcal{M}, \mathit{Reach}}$.
For $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, \mathit{Cmds} \uplus \{\tau\}, \mathcal{T})$ we define

$$\mathcal{M}_{\text{loop}} \stackrel{\text{def}}{=} (\mathbf{A}, \overline{\mathbf{z}}, \mathit{Cmds} \uplus \mathbb{R}_{\geq 0}, \mathcal{T}_{\text{loop}}),$$

where for all $\mathbf{z} \in \mathbf{A}$ we have

- $\mathcal{T}_{\text{loop}}(\mathbf{z}, c) \stackrel{\text{def}}{=} \mathcal{T}(\mathbf{z}, c)$ for $c \in \mathit{Cmds}$,
- for all $\mathtt{t} \in \mathbb{R}_{\geq 0}$ we have $\mathcal{T}_{\text{loop}}(\mathbf{z}, \mathtt{t}) \stackrel{\text{def}}{=} \mathcal{T}(\mathbf{z}, \tau) \cup \{[\mathbf{z} \mapsto 1]\}$.

For a maximising scheduler σ_{loop} of $\mathcal{M}_{\text{loop}}$, we can assume that for all $\mathbf{z} \in \mathbf{A}$ and all paths β with $\text{last}(\beta) = \mathbf{z}$ we have $\sigma_{\text{loop}}(\beta)((\mathsf{t}, [\mathbf{z} \mapsto 1])) = 0$ for all $\mathsf{t} \in \mathbb{R}_{\geq 0}$. This is the case, because the choice of every other transition can at least obtain an equal probability to reach *Reach*, as choosing a self loop cannot contribute to eventually reaching this set. Also, we can assume that for all paths $\beta_{\text{loop},1}$ and $\beta_{\text{loop},2}$ which differ only in actions $\mathsf{t} \in \mathbb{R}_{\geq 0}$ along their traces, we have $\sigma_{\text{loop}}(\beta_{\text{loop},1}) = \sigma_{\text{loop}}(\beta_{\text{loop},2})$. This holds, because by the definition of $\mathcal{M}_{\text{loop}}$ the exact choices of t do not influence the reachability probability. Let β be obtained by replacing all $\mathsf{t} \in \mathbb{R}_{\geq 0}$ in $\beta_{\text{loop},1}$ by τ. We can define a scheduler $\sigma \in \mathit{Sched}_{\mathcal{M}}$ with $\sigma(\beta)((c, \mu)) \stackrel{\text{def}}{=} \sigma_{\text{loop}}(\beta_{\text{loop},1})((c, \mu))$ for $c \in \mathit{Cmds}$ and $\sigma(\beta)((\tau, \mu)) \stackrel{\text{def}}{=} \sum_{\mathsf{t} \in \mathbb{R}_{\geq 0}} \sigma_{\text{loop}}(\beta_{\beta,1})((\mathsf{t}, \mu))$ which induces the same reachability probability in $\mathcal{M}$.
Starting with a scheduler σ of $\mathcal{M}$, we can construct a scheduler σ_{loop} which induces the same reachability probability in a similar way: consider $\beta \in \mathit{Path}^{\text{fin}}_{\mathcal{M}}$ and let β_{loop} be derived from β by replacing actions τ by 0 (or any other $\mathsf{t} \in \mathbb{R}_{\geq 0}$). We can then assign $\sigma_{\text{loop}}(\beta_{\text{loop}}) \stackrel{\text{def}}{=} \sigma(\beta)$. The choice of $\sigma_{\text{loop}}(\beta'_{\text{loop}})$ for other β'_{loop} not constructed from paths β of $\mathcal{M}$ is arbitrary, because in any case we will have $Pr_{\mathcal{M}_{\text{loop}}, \sigma_{\text{loop}}}(\beta'_{\text{loop}}) = 0$.
This shows that

$$\text{val}^{+}_{\mathcal{M}, \mathit{Reach}} = \text{val}^{+}_{\mathcal{M}_{\text{loop}}, \mathit{Reach}}. \tag{3.6}$$

Also, we can show that $([\![\mathcal{H}, \mathbf{T}]\!], \{m_{\mathit{Reach}}\} \times \mathbb{R}^k) \preceq (\mathcal{M}_{\text{loop}}, \mathit{Reach})$ in the notion of Definition 3.15. By Lemma 3.17, this then shows that

$$\text{val}^{+}_{[\![\mathcal{H}, \mathbf{T}]\!], \mathit{Reach}} \leq \text{val}^{+}_{\mathcal{M}_{\text{loop}}, \mathit{Reach}}. \tag{3.7}$$

To do so, we define the simulation relation

$$R \stackrel{\text{def}}{=} \{(s, \mathbf{z}) \mid \mathbf{z} \in \mathbf{A} \wedge s \in \mathbf{z}\}.$$

We have to prove the requirements on a simulation according to Definition 3.15 and that we also have $(\{m_{\mathit{Reach}}\} \times \mathbb{R}^k)$-*Reach*-compatible. Requirement 1 is fulfilled trivially, and the compatibility is easy to see. For Requirement 2, we consider an arbitrary $(s, \mathbf{z}) \in R$ and $a \in \mathit{Act}$. For each $\mu \in \mathcal{T}_{[\![\mathcal{H}, \mathbf{T}]\!]}(s, a)$, where $\mathcal{T}_{[\![\mathcal{H}, \mathbf{T}]\!]}$ is the transition matrix of $[\![\mathcal{H}, \mathbf{T}]\!]$, we have to find $\mu_{\mathbf{A}} \in \mathcal{T}_{\text{loop}}(\mathbf{z}, a)$ with $\mu \sqsubseteq_R \mu_{\mathbf{A}}$.
In case $a = \mathsf{t}$, we have $\mu = [s' \mapsto 1]$ and $\mu_{\mathbf{A}} = [\mathbf{z}' \mapsto 1]$ for some $s' \in S$ and $\mathbf{z}' \in \mathbf{A}$ with $s' \in \mathbf{z}'$. Thus, $(s', \mathbf{z}') \in R$ and we can use a trivial weight function between μ and $\mu_{\mathbf{A}}$.
Consider the case $a = c \in \mathit{Cmds}$. For each $\mu = [s_1 \mapsto p_1, \ldots, s_n \mapsto p_n] \in \mathcal{T}(s, a)$ there is $\mu_{\mathbf{A}} = [\mathbf{z}_1 \mapsto p_1, \ldots \mathbf{z}_n \mapsto p_n]$ with $s_i \in \mathbf{z}_i$ for all i with $1 \leq i \leq n$. We define the weight function $w \colon (S \times \mathbf{A}) \to [0, 1]$ where we have $w(s_i, \mathbf{z}_i) \stackrel{\text{def}}{=} \mu(s_i)$ for all $i \in \{1, \ldots, n\}$ and else $w(\cdot, \cdot) \stackrel{\text{def}}{=} 0$. We have to show that w is indeed a weight function by proving the three requirements of Definition 3.14. Requirement 1 is fulfilled trivially by the definition of R. For Requirement 2, we have

$$\sum_{\mathbf{z} \in \mathbf{A}} w(s_i, \mathbf{z}) = w(s_i, \mathbf{z}_i) = \mu(s_i).$$

For Requirement 3, we have

$$\sum_{S \in S} w(s, \mathbf{z}_i) = \sum_{\substack{\mathbf{z}_j = \mathbf{z}_i, \\ s_j \in \mathbf{z}_j}} w(s_j, \mathbf{z}_i) = \sum_{\substack{\mathbf{z}_j = \mathbf{z}_i, \\ s_j \in \mathbf{z}_j}} \mu(s_j) = \mu_{\mathbf{A}}(\mathbf{z}_i).$$

Thus, all requirements of a compatible simulation relation are fulfilled.
In total, we have

$$\mathrm{val}^{+}_{\mathcal{M},Reach} \overset{\text{Eqn. 3.6}}{=} \mathrm{val}^{+}_{\mathcal{M}_{\text{loop}},Reach} \overset{\text{Eqn. 3.7}}{\geq} \mathrm{val}^{+}_{[\![\mathcal{H},\mathbf{T}]\!],\{m_{Reach}\}\times\mathbb{R}^k} = \mathrm{val}^{+}_{\mathcal{H},m_{Reach}}. \qquad \square$$

Thus, we can prove that the reachability probability in a given PHA does not exceed $\mathfrak{p}$ by proving that the reachability probability does not exceed this bound in the abstraction. We have a similar result also for probabilistic inevitability.

Theorem 3.32. *Let $\mathcal{H}$ be a PHA, let m_{Reach} be a desirable mode, let $\mathcal{M} \in Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ be an abstraction of $\mathcal{H}$ and let $Reach \overset{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{Reach}\}$. Then*

$$\mathrm{val}^{-}_{\mathcal{H},m_{Reach}} \geq \mathrm{val}^{-,Cmds}_{\mathcal{M},Reach}.$$

Proof. The proof is similar to the one of Theorem 3.31. The major difference is that we can no longer argue that the scheduler σ_{loop} will never choose the timed self loops with which we extended $\mathcal{M}_{\text{loop}}$. Indeed, as σ_{loop} is now a minimising scheduler, taking these loops is advantageous to obtain a low reachability probability. The reason that minimal reachability probabilities in $\mathcal{M}_{\text{loop}}$ and $\mathcal{M}$ are still the same is the fairness assumption. For fairness to hold, we must have $Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}(Path^{Act_{\text{fair}}}_{\mathcal{M}_{\text{loop}}}) = 1$. This also implies that for each subset $A \subseteq Path^{\text{inf}}_{\mathcal{M}_{\text{loop}}} \setminus Path^{Act_{\text{fair}}}_{\mathcal{M}_{\text{loop}}}$ we have $Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}(A) = 0$. In particular, this holds for

$$A \overset{\text{def}}{=} \{\beta_{\text{loop}} \in Path^{\text{inf}}_{\mathcal{M}_{\text{loop}}} | \exists \mathbf{z} \in \mathbf{A}.\ \exists i \geq 0.\ \forall j \geq i.\ \beta_{\text{loop}}[j] = \mathbf{z} \wedge \mathrm{trace}(\beta_{\text{loop}})[j] \in \mathbb{R}_{\geq 0}\}.$$

This implies that for all infinite sequences $\langle \beta_{\text{loop},i} \rangle_{i \in \mathbb{N}}$ of finite paths of $\mathcal{M}_{\text{loop}}$ with

- $|\beta_{\text{loop},i}| = i$,
- $\beta_{\text{loop},i} \leq \beta_{\text{loop},i+1}$, and
- $Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}(\beta_{\text{loop},i}) > 0$

we have infinitely many $j \in \mathbb{N}$ with

$$\sum_{\mathsf{t} \in \mathbb{R}_{\geq 0}} \sigma_{\text{loop}}(\beta)(\mathsf{t}, [\mathrm{last}(\beta_{\text{loop},j}) \mapsto 1]) < 1. \tag{3.8}$$

Otherwise, we would have a single nonfair path with a positive probability, which would then imply that the probability of the above set A is positive.
For $\beta = \mathbf{z}_0\, a_0\, \mu_0\, \ldots\, \mathbf{z}_{n-1}\, a_{n-1}\, \mu_{n-1}\, \mathbf{z}_n \in Path^{\text{fin}}_{\mathcal{M}}$ we define

$$\mathrm{loop}(\beta) \overset{\text{def}}{=} \mathbf{z}_0\, (\mathbb{R}_{\geq 0}\, [\mathbf{z}_0 \mapsto 1]\, \mathbf{z}_0)^*\, a_0\, \mu_0\, \ldots\, \mathbf{z}_{n-1}\, (\mathbb{R}_{\geq 0}\, [\mathbf{z}_{n-1} \mapsto 1]\, \mathbf{z}_{n-1})^*\, a_{n-1}\, \mu_{n-1}\, \mathbf{z}_n,$$

which means that $\text{loop}(\beta) \subseteq Path^{\text{fin}}_{\mathcal{M}_{\text{loop}}}$ is the set of all variants of β with timed self loops inserted.

We define $\sigma \in Sched_{\mathcal{M}}$ where for $(a,\mu) \in Act \times Distr(\mathbf{A})$, if $(a,\mu) \in \mathbb{R}_{\geq 0} \times \{[\text{last}(\beta) \mapsto 1]\}$ then $\sigma(\beta)(a,\mu) \stackrel{\text{def}}{=} 0$, and otherwise

$$\sigma(\beta)(a,\mu) \stackrel{\text{def}}{=} \frac{1}{Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}(\bigcup_{\beta_{\text{loop}} \in \text{loops}(\beta)} Cyl(\beta_{\text{loop}}))} \sum_{\beta_{\text{loop}} \in \text{loops}(\beta)} Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}(\beta_{\text{loop}})(a,\mu).$$

This way, σ takes decisions for β according to those of $\text{loop}(\beta)$ weighted with their probability. Because of Equation 3.8, we have

$$\sum_{\substack{a \in Act,\\ \mu \in Distr(\mathbf{A})}} \sigma(\beta)(a,\mu) = 1,$$

which implies that σ is well-defined. Without the validity of Equation 3.8, the above sum could be smaller than 1.

For $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ we then have

$$Pr_{\mathcal{M},\sigma}(Cyl(\beta)) = Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}\left(\bigcup_{\beta_{\text{loop}} \in \text{loop}(\beta)} Cyl(\beta_{\text{loop}})\right), \tag{3.9}$$

we have for $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ that

$$Pr_{\mathcal{M},\sigma}\left(\bigcup_{\beta \in Path^{\text{fin}}_{\mathcal{M}}} \bigcup_{\beta_{\text{loop}} \in \text{loop}(\beta)} Cyl(\beta_{\text{loop}})\right) = 1, \tag{3.10}$$

and for $\beta_1, \beta_2 \in Path^{\text{fin}}_{\mathcal{M}}$ with $\beta_1 \neq \beta_2$ we have

$$\text{loop}(\beta_1) \cap \text{loop}(\beta_2) = \emptyset. \tag{3.11}$$

From this, we can conclude that

$$\begin{aligned}
&\text{val}^{\sigma}_{\mathcal{M},Reach} \\
&= Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach) \\
&\stackrel{\text{Eqn. 3.9}}{=} Pr_{\mathcal{M},\sigma}\left(\left\{\beta' \in \bigcup_{\beta \in Path^{\text{fin}}_{\mathcal{M}}} Cyl(\beta) \mid \exists i \geq 0.\ X_i^{\mathcal{M}}(\beta') \in Reach\right\}\right) \\
&\stackrel{\text{Eqn. 3.10}}{=} Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}\left(\left\{\beta' \in \bigcup_{\beta \in Path^{\text{fin}}_{\mathcal{M}}} \bigcup_{\beta_{\text{loop}} \in \text{loop}(\beta)} Cyl(\beta_{\text{loop}}) \mid \exists i \geq 0.\ X_i^{\mathcal{M}}(\beta') \in Reach\right\}\right) \\
&\stackrel{\text{Eqn. 3.11}}{=} Pr_{\mathcal{M}_{\text{loop}},\sigma_{\text{loop}}}(\exists i \geq 0.\ X_i^{\mathcal{M}_{\text{loop}}} \in Reach) \\
&= \text{val}^{\sigma_{\text{loop}}}_{\mathcal{M}_{\text{loop}},Reach_{\text{loop}}},
\end{aligned}$$

which means that the reachability probabilities under these two schedulers agree. □

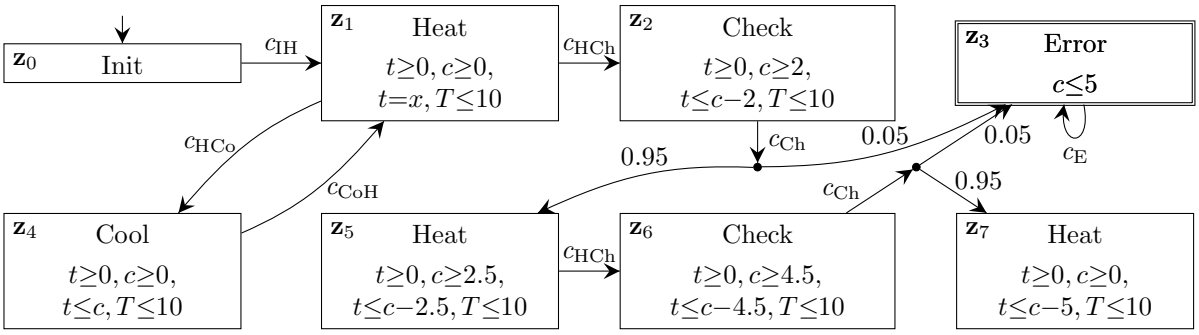


Figure 3.4: Abstraction of the thermostat HA.

Example 3.33. *In Figure 3.4 we give the relevant parts of an abstraction $\mathcal{M}$ of the PHA $\mathcal{H}$ with $\mathfrak{T} = 5$ of Figure 3.3. As before, we let Reach $\stackrel{\text{def}}{=} \{\mathbf{z}_3\}$ where $\mathbf{z}_3 = \text{Error} \times \mathbb{R}^k$. A short calculation shows that*

$$\text{val}^{+}_{\mathcal{M},Reach} = 0.05 + 0.95 \cdot 0.05 = 0.0975.$$

Thus, if we are given a threshold of $\mathfrak{p} \stackrel{\text{def}}{=} 0.0975$ or higher, this shows that $\mathcal{H}$ is sufficiently safe. Indeed, as we have seen in Example 3.27, the actual reachability probability of $\mathcal{H}$ is only 0.05, so the probability in the abstraction is strictly higher. △

3.3.1 Computing Abstractions

We have already defined abstractions of a given PHA, and have already shown that the safety of such an abstraction induces the safety of the semantics of the PHA. However, it is thus far not clear how we can obtain a valid abstraction. For the abstractions of nonprobabilistic HAs, we did not have this problem, because our definition matches the abstractions returned by usual solvers, like HSolver or PHAVer.

To compute conservative bounds on reachability probabilities in the PHAs we considered, we aim at reusing such tools to obtain abstractions of PHAs. For this, we firstly define the *induced HA* of a given PHA, which is a purely nondeterministic overapproximation of the PHA.

Definition 3.34. *The* induced HA *of a PHA $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ is defined as*

$$\text{ind}(\mathcal{H}) \stackrel{\text{def}}{=} (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{ind}}).$$

Each $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ induces a set of nonprobabilistic guarded commands $\text{ind}(c) \stackrel{\text{def}}{=} \{(g \to u_1), \ldots, (g \to u_n)\}$. Then, $Cmds_{\text{ind}} \stackrel{\text{def}}{=} \bigcup_{c \in Cmds} \text{ind}(c)$.

The induced HA can be obtained from the description of a PHA by a simple syntactical transformation. It is an HA as defined in Chapter 2, which can be subjected to analysis by the usual solvers as demonstrated there. From the abstractions we obtain this way, we will construct an abstraction of the original PHA:

Definition 3.35. *Let* $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, \mathit{Cmds}_{\mathrm{ind}} \uplus \{\tau\}, \mathcal{T}) \in \mathit{Abs}(\mathrm{ind}(\mathcal{H}), \mathbf{A}, \mathbf{T})$ *be an abstraction of the induced HA* $\mathrm{ind}(\mathcal{H})$ *of a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle \mathit{Post}_m \rangle_{m \in M}, \mathit{Cmds})$ *(cf. Definition 2.24). We define the* induced abstraction *of* $\mathcal{H}$ *as*

$$\mathrm{abs}(\mathcal{H}, \mathcal{M}) = (\mathbf{A}, \overline{\mathbf{z}}, \mathit{Cmds} \uplus \{\tau\}, \mathcal{T}),$$

where

- *for all*
  - $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in \mathit{Cmds}$ *with* $\mathrm{ind}(c) = \{g \to u_1, \ldots, g \to u_n\}$,
  - *and* $\mathbf{z} \in \mathbf{A}$, $\mathbf{z}_1, \ldots, \mathbf{z}_n \in \mathbf{A}$ *with* $\mathbf{z}_1 \in \mathcal{T}(\mathbf{z}, (g \to u_1)), \ldots, \mathbf{z}_n \in \mathcal{T}(\mathbf{z}, (g \to u_n))$

  we have $[\mathbf{z}_1 \mapsto p_1, \ldots, \mathbf{z}_n \mapsto p_n] \in \mathcal{T}(\mathbf{z}, c)$,
- *for all* $\mathbf{z}, \mathbf{z}' \in \mathbf{A}$ *with* $\mathbf{z}' \in \mathcal{T}(\mathbf{z}, \tau)$ *we have* $[\mathbf{z}' \mapsto 1] \in \mathcal{T}(\mathbf{z}, \tau)$.

The induced abstraction uses the LTS abstraction $\mathcal{M}$ of the induced HA $\mathrm{ind}(\mathcal{H})$ to obtain a PA abstraction $\mathrm{ind}(\mathcal{H}, \mathcal{M})$ for the original PHA $\mathcal{H}$. It does so by using the labels of the LTS $\mathcal{M}$ to build a superset of the transitions that must be present in $\mathrm{ind}(\mathcal{H}, \mathcal{M})$ to be a valid abstraction of $\mathcal{H}$ for the given abstract state space $\mathbf{A}$.

Example 3.36. *The HA of Figure 2.4 is the induced nonprobabilistic HA* $\mathrm{ind}(\mathcal{H})$ *of the PHA* $\mathcal{H}$ *of Figure 3.3 and we have*

$$\mathrm{ind}(c_{\mathrm{Ch}}) = \{c_{\mathrm{ChH}}, c_{\mathrm{ChE}}\}.$$

For all other commands, we can identify the induced command with the command itself, if we interpret the sets of successor states of updates $u(s)$ *of the nonprobabilistic guarded commands as sets* $\{[s' \mapsto 1] \mid s' \in u(s)\}$ *of Dirac distributions.* △

The construction of $\mathrm{abs}(\mathcal{H}, \mathcal{M})$ is applicable in practice, because the LTS which is the abstraction of the induced HA is finite.

Although we called the model obtained using Definition 3.35 an abstraction of a PHA, we yet have to show that it is indeed an abstraction in the notion of Definition 3.30.

Lemma 3.37. *For a PHA* $\mathcal{H}$*, abstract state space* $\mathbf{A}$*, time restriction* $\mathbf{T}$ *and abstraction* $\mathcal{M} \in \mathit{Abs}(\mathrm{ind}(\mathcal{H}), \mathbf{A}, \mathbf{T})$*, we have* $\mathrm{abs}(\mathcal{H}, \mathcal{M}) \in \mathit{Abs}(\mathcal{H}, \mathbf{A}, \mathbf{T})$.

Proof. Assume $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, \mathit{Cmds}_{\mathrm{ind}} \uplus \{\tau\}, \mathcal{T}_{\mathrm{ind}})$ and $\mathrm{abs}(\mathcal{H}, \mathcal{M}) = (\mathbf{A}, \overline{\mathbf{z}}, \mathit{Cmds} \uplus \{\tau\}, \mathcal{T})$. It is immediately clear that the requirements on the initial state are fulfilled: this requirement is the same as in a nonprobabilistic abstraction which we use to obtain our probabilistic abstraction. Also, the requirements on the timed transitions are easy to check. In the PHAs abstractions, we basically have the same timed transitions as in HAs, just that in the probabilistic case they are wrapped in Dirac distributions.

Thus, we only consider the distributions resulting from probabilistic guarded commands. Let $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in \mathit{Cmds}$ be an arbitrary probabilistic guarded

command of $\mathcal{H}$, let $\mathbf{z} \in \mathbf{A}$ be an abstract state, and consider $s \in \mathbf{z} \cap g$. We thus have to show that for each

$$\mu \in \{[s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \mid s'_1 \in u_1(s), \ldots, s'_n \in u_n(s)\}$$

there is $\mu_{\mathbf{A}} \in \mathit{Lift}_{\mathbf{A}}(\mu)$ with $\mu_{\mathbf{A}} \in \mathcal{T}(\mathbf{z}, c)$. By definition of the induced HA, for $1 \leq i \leq n$ we have $\mathbf{z}_i \in \mathcal{T}_{\text{ind}}(\mathbf{z}, (g \to u_i))$ with $s_i \in \mathbf{z}_i$ (where the $(g \to u_i)$ are the induced guarded commands of c). Thus, by construction of $\mathit{Abs}(\mathcal{H}, \mathbf{A}, \mathbf{T})$ we have $\mu_{\mathbf{A}} \stackrel{\text{def}}{=} [\mathbf{z}_1 \mapsto p_1, \ldots, \mathbf{z}_n \mapsto p_n] \in \mathcal{T}(\mathbf{z}, c)$. By the definition of lifted distributions (cf. Definition 3.28), we also have $\mu_{\mathbf{A}} \in \mathit{Lift}_{\mathbf{A}}(\mu)$. □

Thus, we have a practical means to obtain a model which can bound probabilities of properties in the semantics of our model.

Corollary 3.38. *Consider a PHA $\mathcal{H}$, an abstraction $\mathcal{M}_{\text{ind}}$ of the induced HA $\text{ind}(\mathcal{H})$, $\mathcal{M} \stackrel{\text{def}}{=} \text{abs}(\mathcal{H}, \mathcal{M}_{\text{ind}})$ and a mode m_{Reach} of $\mathcal{H}$. Let Reach be defined as in Theorem 3.31 and let Cmds be the set of probabilistic guarded commands of $\mathcal{H}$. Then*

$$\text{val}^{+}_{\mathcal{H}, m_{\mathit{Reach}}} \leq \text{val}^{+}_{\mathcal{M}, \mathit{Reach}} \text{ and } \text{val}^{-}_{\mathcal{H}, m_{\mathit{Reach}}} \geq \text{val}^{-, \mathit{Cmds}}_{\mathcal{M}, \mathit{Reach}}.$$

This corollary follows directly from Lemma 3.37, Theorem 3.31 and Theorem 3.32.

Example 3.39. *Let $\mathcal{H}$ denote the PHA of Figure 3.3. If we let $\mathcal{M}$ denote the LTS of Figure 2.5, we have $\mathcal{M} \in \mathit{Abs}(\mathcal{H}, \mathbf{A}, \mathbf{T})$ with corresponding $\mathbf{A}$ and $\mathbf{T}$. If we denote by $\mathcal{M}'$ the PA of Figure 3.4, then we have $\mathcal{M}' = \text{abs}(\mathcal{H}, \mathcal{M})$.* △

3.4 Algorithmic Consideration

We have discussed how we can obtain an abstraction which safely overapproximates the behaviour of the semantics of our PHAs. While in the nonprobabilistic case properties at this point can be decided on the abstract models by simple graph-based methods, the probabilistic case requires a bit more of discussion.

There are several approaches to compute minimal and maximal reachability values in finite PAs, for instance linear programming, value iteration, and policy iteration [Put94; BK08]. In the following, we will describe a policy iteration algorithm in detail, because it fits well with the extensions of the basic method we discuss in this chapter. For instance, in Chapter 6 we are going to extend this method to parametric models.

The method discussed here does not take fairness into account. Yet, the values obtained without this restriction are still lower and upper bounds for the values in the model semantics. This is the case because fairness restricts the set of all possible schedulers among which to find the minimising or maximising from. It might happen that the values obtained this way are too small to be of use. The problem does not occur for safety properties, as fairness is not needed here, while for inevitability, from the way we defined the abstraction, we have avoided having timed self loops in the abstract states,

which would otherwise have led to a minimal probability bound of 0. The problem of too coarse bounds will thus disappear in many cases, because we have removed the genuine source of bad bounds. In case that these bounds are still an issue, it is possible to adapt an existing algorithm which does take fairness into account [BK98] to the specific definition of fairness we use here.

To describe the policy iteration algorithm, we firstly have to define simpler models which, other than PAs, do not contain nondeterminism but only probabilistic behaviour. Because of this, no scheduler will be needed to specify reachability probabilities.

Definition 3.40. *A* Markov chain (MC) *is a tuple*

$$\mathcal{M} = (S, \overline{s}, \mathcal{T}),$$

where

- S *is a set of* states*,*
- $\overline{s} \in S$ *is the* initial state*, and the*
- transition matrix $\mathcal{T} \colon S \to \mathit{Distr}(S)$ *assigns to each state a probability distribution over successor states.*

The adjacency graph *of* $\mathcal{M}$ *is a graph* $(S, \overline{s}, \mathcal{T}_{\mathrm{adj}})$ *where we have* $(s, s') \in \mathcal{T}_{\mathrm{adj}}$ *if and only if* $\mathcal{T}(s)(s') \neq 0$*.*

Reachability probabilities in MCs could be specified using measures on sets of paths, as for PAs. For simplicity, we define them directly as solutions of equation systems.

Definition 3.41. *Given an MC* $\mathcal{M} = (S, \overline{s}, \mathcal{T})$ *and a set of states* $\mathit{Reach} \subseteq S$*, we define*

$$\mathrm{vals}_{\mathcal{M},\mathit{Reach}} \colon S \to [0,1]$$

as the unique solution of the equation system where for all $s \in S$ *we have*

$$v(s) = \begin{cases} 1 & \text{if } s \in \mathit{Reach}, \\ 0 & \text{if } s \in \overline{\mathit{Reach}}, \\ \sum_{s' \in S} \mathcal{T}(s)(s') v(s') & \text{else}, \end{cases}$$

where by $\overline{\mathit{Reach}}$ *we denote the set of states which cannot reach Reach in the adjacency graph. We also let* $\mathrm{val}_{\mathcal{M},\mathit{Reach}} \stackrel{\mathrm{def}}{=} \mathrm{vals}_{\mathcal{M},\mathit{Reach}}(\overline{s})$*.*

In the definition above, $\mathrm{vals}_{\mathcal{M},\mathit{Reach}}(s)$ is the value which would be obtained if the initial state were s instead of $\overline{s}$. There exists a number of possible methods to solve this equation system. *Direct methods*, like Gaussian elimination, obtain a solution by a transformation of the matrix, and are thus in principle able to find an exact solution. *Iterative methods* try to obtain an approximate solution by repeated vector-matrix multiplications. In cases where the transition matrix is represented by a sparse matrix, iterative methods often perform better, because direct methods usually destroy the sparsity of the matrix.

input : finite MC $\mathcal{M} = (S, \overline{s}, \mathcal{T})$, state set $Reach \subseteq S$, precision ε.
output: $\text{vals}_{\mathcal{M},Reach}$.

1 **forall the** $s \in S$ **do** $v'(s) := \begin{cases} 1 & \text{if } s \in Reach, \\ 0 & \text{else} \end{cases}$
2 **repeat**
3 $\quad v := v'$
4 $\quad$ **forall the** $s \in S \setminus Reach$ **do**
5 $\quad\quad v'(s) := \sum_{s' \in S} \mathcal{T}(s)(s') v(s')$
6 **until** $\max_{s \in S} |v(s) - v'(s)| < \varepsilon$
7 **return** v

Algorithm 3.1: REACHITER($\mathcal{M}, Reach, \varepsilon$).

One variant of an iterative solution method, the Jacobi method, is given in Algorithm 3.1. Firstly, in Line 1 it initialises a vector v to be 1 for states of *Reach* and 0 else. Then, in Line 5 it uses the definition of the equation system to assign weighted probability values to a primed version of v, swaps v and v', and repeats. This way, v approaches the exact solution of the equation system from below. The iteration is stopped once in Line 6 the two vectors are similar enough, so that no larger changes are expected.

We can combine a PA and a simple scheduler to obtain an MC.

Definition 3.42. *Consider a PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ and a simple scheduler $\sigma \colon S \to (Act \times Distr(S))$. The MC* induced *by a simple scheduler σ of $\mathcal{M}$ is defined as $\mathcal{M}_\sigma \stackrel{\text{def}}{=} (S, \overline{s}, \mathcal{T}_\sigma)$ where for $s \in S$ we have $\mathcal{T}_\sigma(s) \stackrel{\text{def}}{=} \mu$ if $\sigma(s) = (a, \mu)$.*

This MC then suffices to compute the reachability probabilities we obtain from the PA and scheduler.

Lemma 3.43. *Given a PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$, a simple scheduler σ, and $Reach \subseteq S$, we have*

$$\text{vals}^{\sigma}_{\mathcal{M},Reach} = \text{vals}_{\mathcal{M}_\sigma,Reach}.$$

Here, for $s \in S$ we have $\text{vals}^{\sigma}_{\mathcal{M},Reach}(s)$ the reachability probability if s were the initial state of $\mathcal{M}$ rather than $\overline{s}$.

It is known [BK08] that for a finite PA there is a simple scheduler which achieves the minimal or maximal reachability value. In principle, we could now just try out all simple schedulers, as there are only finitely many. However, as there are exponentially many, this would be too costly. Instead, we apply a dynamic programming algorithm, where we try to locally optimise scheduler decisions, based on the following lemma.

Lemma 3.44. *Let $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ be a PA, $Reach \subseteq S$, consider a simple scheduler σ of $\mathcal{M}$ and let*

$$\overline{Reach} \stackrel{\text{def}}{=} \{s \in S \mid \text{vals}^{-}_{\mathcal{M},Reach}(s) = 0\}.$$

Here, for $s \in S$ we denote by $\mathrm{vals}^{-}_{\mathcal{M},Reach}(s)$ the minimal reachability probability if s were the initial state of $\mathcal{M}$ rather than $\overline{s}$. If we have for all $s \in \overline{Reach}$ that $\mathrm{vals}^{\sigma}_{\mathcal{M},Reach}(s) = 0$ and for all $s \in S \setminus (Reach \cup \overline{Reach})$ we have

$$\mathrm{vals}^{\sigma}_{\mathcal{M},Reach}(s) = \min_{\substack{a \in Act,\\ \mu \in \mathcal{T}(s,a)}} \sum_{s' \in S} \mu(s') \mathrm{vals}^{\sigma}_{\mathcal{M},Reach}(s'),$$

then we have $\mathrm{vals}^{-}_{\mathcal{M},Reach} = \mathrm{vals}^{\sigma}_{\mathcal{M},Reach}$.

This means that the optimality of a simple scheduler can be proven by a local check for each state of the model. A corresponding lemma for the maximising case exists, in which case we do not need to take into account the set $\overline{Reach}$.

The variant of policy iteration we use for the minimal reachability probability is depicted in Algorithm 3.3. The algorithm for the maximum is similar. In Line 1, we choose an initial scheduler using the procedure minReachInit of Algorithm 3.2. The algorithm computes a scheduler σ with $\mathrm{vals}_{\mathcal{M}_\sigma,Reach}(s) = 0$ for all $s \in \overline{Reach}$, which is necessary for the correctness of Algorithm 3.3. For maximising the value, this step is not necessary and we can start with an arbitrary scheduler. Algorithm 3.2 firstly computes the set $\overline{Reach}$ in lines 1 to 10. It does so by computing the complement T of this set, that is the set of states which reach $Reach$ with a probability larger than zero for all possible schedulers. Once $\overline{Reach}$ is computed, in lines Line 11 to Line 13 we can compute the scheduler by fixing decisions for states of $\overline{Reach}$ where each successor state chosen with positive probability is also in $\overline{Reach}$.

Then in Algorithm 3.3, lines 2 to 9 try to improve the scheduler systematically. In Line 4 we compute the reachability values when using the current scheduler. In lines 5 to 8 we improve the local decision for each state which is not contained in $Reach$, if this is possible. Afterwards, the new reachability values from the new scheduler are computed and so forth, until the scheduler cannot be improved any further. Finally, Line 10 returns the maximal value together with a maximising scheduler.

Example 3.45. *Again, consider the PA $\mathcal{M}$ of Figure 3.4 and let $Reach \stackrel{\text{def}}{=} \{\mathbf{z}_3\}$. We want to use the maximising variant of Algorithm 3.3 to compute $\mathrm{val}^{+}_{\mathcal{M},Reach}$ (see Example 3.33). From Line 1 we get an arbitrary simple scheduler. We assume that we obtain $\sigma' \colon \mathbf{A} \to (Act, Distr(\mathbf{A}))$ with*

$$\sigma'(\mathbf{z}_0) \stackrel{\text{def}}{=} (c_{\mathrm{IH}}, [\mathbf{z}_1 \mapsto 1]), \sigma'(\mathbf{z}_1) \stackrel{\text{def}}{=} (c_{\mathrm{HCo}}, [\mathbf{z}_4 \mapsto 1]), \sigma'(\mathbf{z}_2) \stackrel{\text{def}}{=} (c_{\mathrm{Ch}}, [\mathbf{z}_5 \mapsto 0.95, \mathbf{z}_3 \mapsto 0.05]),$$
$$\sigma'(\mathbf{z}_4) \stackrel{\text{def}}{=} (c_{\mathrm{CoH}}, [\mathbf{z}_1 \mapsto 1]), \sigma'(\mathbf{z}_5) \stackrel{\text{def}}{=} (c_{\mathrm{HCh}}, [\mathbf{z}_6 \mapsto 1]), \sigma'(\mathbf{z}_6) \stackrel{\text{def}}{=} (c_{\mathrm{Ch}}, [\mathbf{z}_7 \mapsto 0.95, \mathbf{z}_3 \mapsto 0.05]).$$

We now enter the main loop at Line 2. After setting $\sigma := \sigma'$, in Line 4 we compute the induced MC $\mathcal{M}_\sigma$.

Then, we start the computation of reachability probabilities in $\mathcal{M}_\sigma$ using the value iteration in Algorithm 3.1. The sequence of v we compute is given in Table 3.1. Thus, we reach the exact solution in the MC after a few iterations. Then, in Line 5 of Algorithm 3.3, we try to improve the decisions of the scheduler. The only state for which

input : finite PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$, state set $Reach \subseteq S$.
output: $\sigma \in Sched_{\mathcal{M}}^{\text{simple}}$ with $\text{vals}^{\sigma}_{\mathcal{M},Reach}(s) = 0$ for $s \in \overline{Reach}$.

```
1  T := R := Reach
2  while R ≠ ∅ do
3  |  choose t ∈ R
4  |  R := R \ {t}
5  |  forall the (s, a, μ) with μ ∈ T(s, a), μ(t) > 0 and s ∉ T do
6  |  |  T(s, a) := T(s, a) \ {μ}
7  |  |  if for all a ∈ Act we have T(s, a) = ∅ then
8  |  |  |  R := R ∪ {s}
9  |  |  |  T := T ∪ {s}
10 Reach := S \ T
11 forall the s ∈ Reach do
12 |  choose σ(s) from {(a, μ) | μ ∈ T(s, μ) ∧ ∀s' ∈ S, μ(s') > 0. s' ∈ Reach}
13 return σ
```

Algorithm 3.2: minReachInit($\mathcal{M}$, $Reach$).

there are multiple possible decisions is $\mathbf{z}_1$*. Indeed, in Line 6 (where we use* argmax *instead of* argmin*), we obtain* $A(\mathbf{z}_1) = \{(c_{\text{HCh}}, \mu)\}$ *rather than the old decision, and thus modify the decision* $\sigma(\mathbf{z}_0)$ *accordingly in Line 8. The updated scheduler now also yields a value of* 0.0975 *for* $\mathbf{z}_0$*, and a subsequent iteration of the main loop does not lead to further improvements. Thus, we return* (v, σ)*.*

Algorithm 3.1 does not *guarantee that an exact result is ever reached. For instance, in Table 3.2 we give the values for the first few iterations when using scheduler* σ_2 *of Example 3.13 on the PA of Figure 3.1. The values for* s_0 *converge to* $\frac{6}{7}$*, but never reach this value, which means that the computation has to be stopped at some point when the result is precise enough, that is if* $\max_{s \in S} |v(s) - v'(s)| < \varepsilon$*.* △

3.5 Case Studies

In this section, we use the method we have developed throughout this chapter to obtain upper bounds for probabilistic reachability properties in several case studies. We use the hybrid solver PHAVer, which we already employed in Section 2.5. We implemented the method to obtain reachability probabilities in a tool called ProHVer, which consists of a modified version of PHAVer plus a component to compute values in the abstraction PAs. All experiments were run on an Intel(R) Core(TM)2 Duo CPU with 2.67 GHz and 4 GB RAM as in the previous chapter.

input : finite PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$, state set $Reach \subseteq S$.
output: $(\text{vals}^{-}_{\mathcal{M},Reach}, \sigma)$ with $\sigma \in Sched^{\text{simple}}_{\mathcal{M}}$ and $\text{vals}^{-}_{\mathcal{M},Reach} = \text{vals}^{\sigma}_{\mathcal{M},Reach}$.

$\sigma' :=$ minReachInit($\mathcal{M}$) /* cf. Algorithm 3.2 */
repeat
  $\sigma := \sigma'$
  $v := \text{vals}_{\mathcal{M}_\sigma, Reach}$
  forall the $s \in S \setminus Reach$ **do**
    $A(s) := \text{argmin}_{\substack{(a,\mu),\\ \mu \in \mathcal{T}(s,a)}} \sum_{s' \in S} \mu(s')v(s')$
    if $\sigma(s) \in A(s)$ **then** $\sigma'(s) := \sigma(s)$
    else choose $\sigma'(s)$ from $A(s)$
until $\sigma' = \sigma$
return (v, σ)

Algorithm 3.3: minReach($\mathcal{M}$, *Reach*).

iter	$\mathbf{z}_0$	$\mathbf{z}_1$	$\mathbf{z}_2$	$\mathbf{z}_3$	$\mathbf{z}_4$	$\mathbf{z}_5$	$\mathbf{z}_6$	$\mathbf{z}_7$
1	0	0	1	0	0	0	0	0
2	0	0.05	1	0	0	0	0.05	0
3	0	0.05	1	0	0	0.05	0.05	0
4	0	0.0975	1	0	0	0.05	0.05	0
5	0	0.0975	1	0	0	0.05	0.05	0

Table 3.1: Iteration values using the model of Figure 3.4.

iter	s_0	s_1	s_2	s_3
1	0	0	1	0
2	0	0.6	1	0
3	0.6	0.78	1	0
4	0.78	0.834	1	0
5	0.834	0.8502	1	0

Table 3.2: Iteration values using the model of Figure 3.1.

3.5.1 Thermostat

We applied ProHVer on the probabilistic thermostat developed throughout the chapter. The induced nonprobabilistic HA of this model was already analysed in Section 2.5. We thus use the abstractions obtained there to build the abstractions of the PHA model. ProHVer can then build an abstraction of the probabilistic thermostat using the abstraction of the induced HA. In Table 3.3, we give performance figures and probabilities (rounded to a sufficient number of decimal places) for the analyses for different time bounds $\mathfrak{T}$. Because the time to compute values from the abstract PA was negligible, the performance values are the same as in Section 2.5. We use the PHAVer-specific refinement technique described in Section 2.5. As expected, the probability bounds usually decrease with an decreasing constraint length and time bound. This is however not always the case. As discussed, PHAVer does not apply a refinement in the usual sense, and we thus cannot guarantee monotonicity of the abstraction.

3.5.2 Bouncing Ball

We consider a bouncing ball assembled from different materials. Assume this ball consisting of three parts: 50 per cent of its surface consists of hard material, 25 per cent

$\mathfrak{T}$	constraint length 10			constraint length 2		
	build (s)	states	prob.	build (s)	states	prob.
2	0	8	0.000	0	11	0.000
4	0	12	1.000	0	43	0.050
5	0	13	1.000	1	58	0.098
20	1	95	1.000	13	916	0.370
40	18	609	0.512	36	2207	0.642
80	59	1717	1.000	85	4916	0.884
120	31	1502	0.878	91	4704	0.940
160	177	4260	0.954	183	10195	0.986
180	136	3768	0.961	219	10760	0.986
600	625	12617	1.000	1268	47609	1.000

Table 3.3: Probabilistic thermostat performance figures.

$\mathfrak{T}$	constraint length 0.15			constraint length 0.15, hull			constraint length 0.05, hull		
	build (s)	states	prob.	build (s)	states	prob.	build (s)	states	prob.
1.0	0	38	0	0	17	0	2	56	0
2.0	1	408	0.25	2	59	0.25	5	185	0.25
3.0	9	2907	0.5	2	124	0.5	12	347	0.3125
3.5	43	11216	0.5	3	145	1	16	425	0.5
3.6	63	14297	0.5	3	150	1	19	436	0.5
3.7	94	18338	0.5	3	154	1	-	-	-

Table 3.4: Bouncing ball performance figures.

consists of a material of medium hardness and the rest consists of very soft material. We drop the ball from a height $h = 2$. Further, we assume the gravity constant to be $g = 1$. If the ball hits the floor and falls on the hard side, it will jump up again with one half of the speed it had before, and if it falls down on the medium side it will jump up with a quarter of the speed. However, if it hits the ground with its soft side, it will not jump up again. We assume that the side with which the ball hits the ground is determined randomly, according to the amount of surface covered by each of the materials.

We study the probability that the ball falls on the soft side before a given time bound $\mathfrak{T}$. Results are given in Table 3.4. We conducted three main analysis settings. In the left and medium part of the table, we used partitioning with constraint length of 0.15 on the position and speed variables. For the medium part, we used the convex hull overapproximation (cf. Section 2.5). For the right part of the table, we used a constraint length of 0.05, and the convex hull overapproximation. Entries for which the analysis did not terminate within one hour are marked by "-".

We ascertain here that, without the convex hull overapproximation, with a constraint of length 0.15, we obtain nontrivial upper bounds. However, the analysis time as well as the number of states grows very fast with increasing time $\mathfrak{T}$. The reason for this to happen is that, each time the ball hits the ground, there are three possibilities with which side it will hit the ground. Thus, the number of possibilities for the amount of energy the

ball still has after a number of times n the ground is hit is exponential in n. Also, there is some $\mathfrak{T}$ up to which the ball has done an infinite number of jumps in all cases, which means that this model is not structurally nonzeno (cf. Section 2.2). This indicates that for a $\mathfrak{T}$ near this value we have to use very small partitioning constraints to obtain tight probability bounds. From this, we have a large time and memory consumption in the analysis.

If we use the convex hull overapproximation and a constraint length of 0.15, far less resources have to be used. But we only obtain nontrivial results for $\mathfrak{T}$ up to 3. Using a constraint length limit of 0.05, we obtain a tighter probability bound, while using still less resources than the first configuration. However, for $\mathfrak{T} = 3.7$ the third configuration does not terminate within one hour.

For instance, for $\mathfrak{T} = 3$, we can also compute the probability manually: the ball can hit the ground with its soft side when it hits the floor in first place with a probability of 0.25. It hits the ground initially with its hard side with a probability of 0.5. In this case, it will not hit the ground again before $T = 3$. If it first hits the ground with its medium hard side, it will hit the ground a second time before $T = 3$. After this, there is no chance of touching the ground again before time $\mathfrak{T}$. Because of this, the overall probability of the property under consideration is $0.25 + 0.25 \cdot 0.25 = 0.3125$. This means that the result obtained by using a constraint limit of 0.05 and the convex hull overapproximation is exact.

3.5.3 Water Level Control

We consider a model of a water level control system (extended from the one of Alur et al. [Alu+95]) which uses wireless sensors. Values submitted are thus subject to probabilistic delays, due to the unreliable transport medium.

A sketch of the model is given in Figure 3.5. The water level W of a tank is controlled by a monitor. Its change is specified by a linear function. Initially, the water level is $W = 1$. When no pump is turned on (Fill), the tank is filled by a constant stream of water ($\dot{W}$). We let the system run up to $\mathfrak{T}$, by using a variable to measure the total time until the system is started, which is only given implicitly here. When a water level of $W = 10$ is seen by a sensor of the tank, the pump should be turned on. However, the pump features a certain delay, which results from submitting control data via a wireless network. With a probability of 0.95 this delay takes 2 time units (FillD2), but with a probability of 0.05 it takes 3 time units (FillD3). The delay is modelled by the timer t. After the delay has passed, the water is pumped out with a higher speed than it is filled into the tank ($\dot{W} = -2$ in Drain). There is another sensor to check whether the water level is below 5. If this is the case, the pump must turn off again. Again, we have a distribution over delays here (DrainD2 and DrainD3). For the system to work correctly, the water level must stay between 1 and 12.

We are interested in the probability that the pump system violates the property given above because either the water level falls below 1 or increases above 12, before the

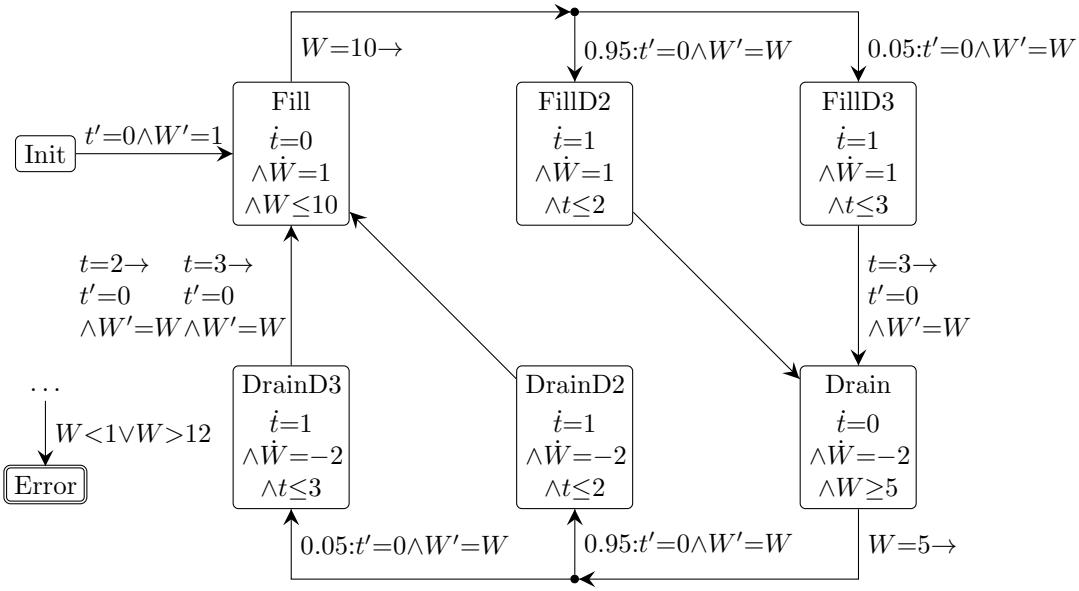


Figure 3.5: Water level control automaton.

$\mathfrak{T}$	no partitioning			constraint length 2		
	build (s)	states	prob.	build (s)	states	prob.
40	0	69	0.185	1	150	0.185
82	0	283	0.370	1	623	0.370
83	0	288	1.000	1	640	0.401
120	1	537	1.000	2	1220	0.512
500	4	3068	1.000	16	7158	0.954
1000	19	6403	1.000	60	14977	0.998

Table 3.5: Water level control performance figures.

larger water tank from which the water tank is filled becomes empty. We model the system in ProHVer and reason about this property. Performance statistics are given in Table 3.5. Without using partitioning, we were only able to obtain useful values for $\mathfrak{T}$ up to 82. We did not use the convex hull overapproximation, like in the bouncing ball case study, nor another overapproximation. For $\mathfrak{T}$ larger than this value, we always obtained a probability limit of 1. To get tighter results, we partitioned t by a constraint of length 2. For $\mathfrak{T}$ below 83 we obtain the nontrivial values in both cases, whereas for 83 we obtain a useful upper bound only when using partitioning. A plot of probabilities for different $\mathfrak{T}$ is given in Figure 3.6. The graph has a staircase form where wide steps alternate with narrow ones. This form arises because, each time the longer time bound is randomly chosen, the tank will overflow or underflow respectively, provided there is enough time left. The wide steps corresponds to the chance of overflow in the tank, the narrow ones to the chance of underflow.

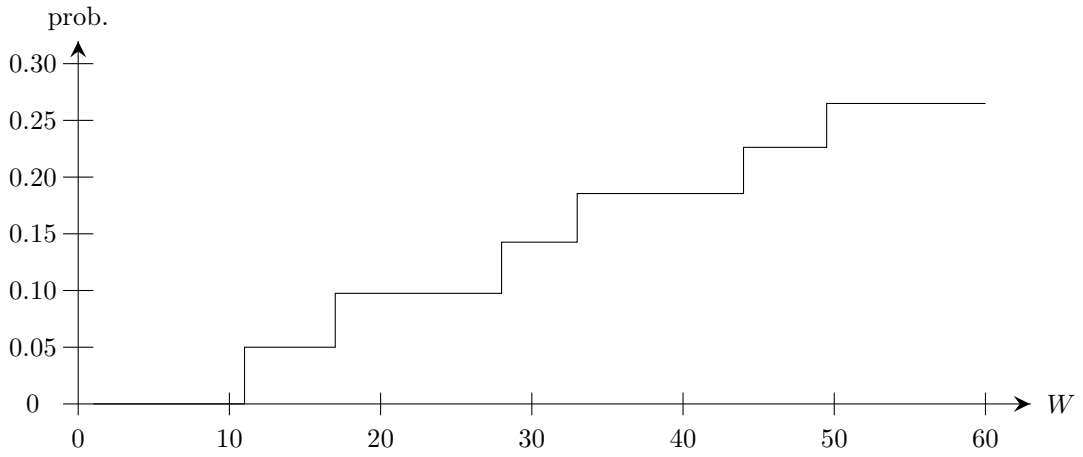


Figure 3.6: Graph of probabilities in water level control.

3.5.4 Autonomous Lawn-mower

We consider an autonomous lawn-mower that uses a probability bias to avoid patterns on lawns. This mower is started on a rectangular grassed area. When it reaches the border of the area, it changes its direction. To prevent the mower from repeating a simple pattern, this direction is randomly biased, such as to ensure that eventually the entire area has been cut.

A sketch of the automaton is given in Figure 3.7. There, l is the length and h the width of the area. The position of the mower on the area is given by (x, y). With (v_x, v_y) we denote the speed in (x, y) direction, which the mower takes with a probability of 0.95 when reaching a border, whereas with (v'_x, v'_y) denotes a variation of the direction that is taken with probability 0.05. Further, (x_g, y_g) describes the mower's initial position. For clarity, we use a statechart-like extension of the PHA description. There is a transition from Init to NE1. Except for this transition, once one of the four macro modes is entered, there is a probabilistic choice between moving to one of the inner modes. For instance, if there is a transition to the macro mode containing NE1 and NE2, this transition represents a transition to NE1 with probability 0.95 and to NE2 with 0.05.

At the region with $x \geq 90 \wedge x \leq 100 \wedge y \geq 170 \wedge y \leq 200$ the owner of the lawn has left a tarpaulin. We are interested in the probability that within a time bound of $t = 120$ the mower hits the tarpaulin, thereby inevitably ripping it up.

The creation of the LTS for this automaton took 98 seconds whereas the computation time of the failure probability was negligible. The upper bound we obtained was 0.000281861. We did not use any constraint specifications.

As in the other case studies, we also varied the time bound. Results are given in Table 3.6. For larger time bounds the analysis time as well as the number of states grows quickly. This is due to a similar effect as in the bouncing ball case study. Each time the mower reaches a border, it may head into two different directions, which leads to a combinatorial explosion, evident by the above statistics.

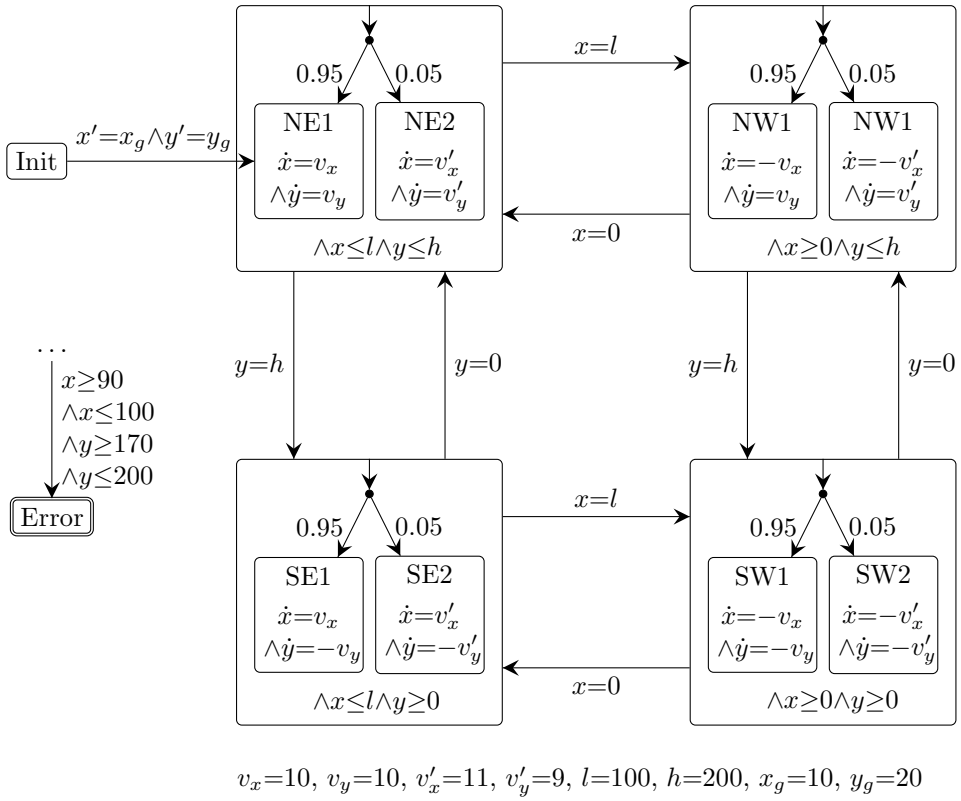


Figure 3.7: Probabilistic autonomous lawn-mower.

We also experimented with convex hull overapproximation without constraint refinement. However, the probability bounds obtained were always 1. Using constraints also did not improve on this, as it only made the analyses take more time. We think that for this case study there is not much hope of obtaining better results in resource usage. The complexity does not really live in the hybrid behaviour, but results from the excessive number of ways the mower can pass around the area. Because of this, we do not think it is possible to find an abstraction to handle this case study for larger time bounds.

3.6 Related Work

There are a number of HAs-based formalisms which extend the classical model by stochastic behaviour. Kwiatkowska et al. [Kwi+02; KNP09] analyse reachability problems and related properties on probabilistic timed automata, in which the guarded commands have been extended by finite-support probability distributions. Thus, this model contains both stochastic as well as nondeterministic behaviour similar to the one we have, but its timed behaviour is more restricted. As here, the interest is in the minimal or maximal probability of certain events. Like the reachability problem in nonstochastic timed automata, the problem of computing the maximal probability of reaching a given set of states is decidable, and the authors also discuss efficient solution methods.

$\mathfrak{T}$	build (s)	states	prob.
10	0	4	0
70	1	632	1.11984E-05
100	5	3022	1.11984E-05
110	30	9076	0.000281861
120	57	12660	0.000281861
130	200	25962	0.000281861

Table 3.6: Lawn-mower performance figures.

Sproston [Spr00; Spr01] pioneered extensions of HAs to probabilistic HAs in a similar way. He proposed a semantics given as a PA which is essentially the same as the one one used in this thesis. For the case of rectangular and o-minimal probabilistic HAs, he also discusses an exact solution method to obtain the maximal reachability probability. Assouramou and Desharnais extend the overapproximation technique for classes of decidable HAs to the probabilistic case [AD11].

All of the approaches discussed above consider solution methods for certain subclasses of HAs, for which these methods are then exact. As more general probabilistic HAs can be overapproximated by these classes, they can still be used to handle general models in many cases. The approach of this thesis is different, as we start with general probabilistic HAs and then rely on tools for general classical, nonprobabilistic HAs to build our abstractions on. We therefore inherit the ignorance wrt. decidability from the undecidability of these model classes.

Teige et al. [FHT08; TF09; FTE10a; Tei12] specialise in probabilistic HAs which constitute discrete-time, continuous-state models with finite-support probability distributions. They also discuss how the model they finally analyse can be obtained as an approximation of continuous-time stochastic hybrid systems. Their method is based on *satisfiability modulo theories* [Gan+04]. Basically, they build a formula which reasons about the minimal or maximal probability to reach a certain set of states within N steps. Their analysis is thus in principle limited to depth-bounded reachability properties, i.e., the probability of reaching a location within at most N discrete jumps, but they discuss how to circumvent this restriction. The method has the advantage to scale to a very large number of different modes. The approach is orthogonal to the one we use, as the solution method is completely different and does not rely on the construction of an abstract state space. Another difference is that they prove lower bounds for the worst-case probability to reach a set of unsafe states, while the method of this thesis targets at computing upper bounds.

A model similar to the PAs which form the semantics of our PHAs is known as Markov decision processes (MDPs). A number of papers and textbooks on both models exist [Bel57; How60; Var85; Put94; BK98; BK08].

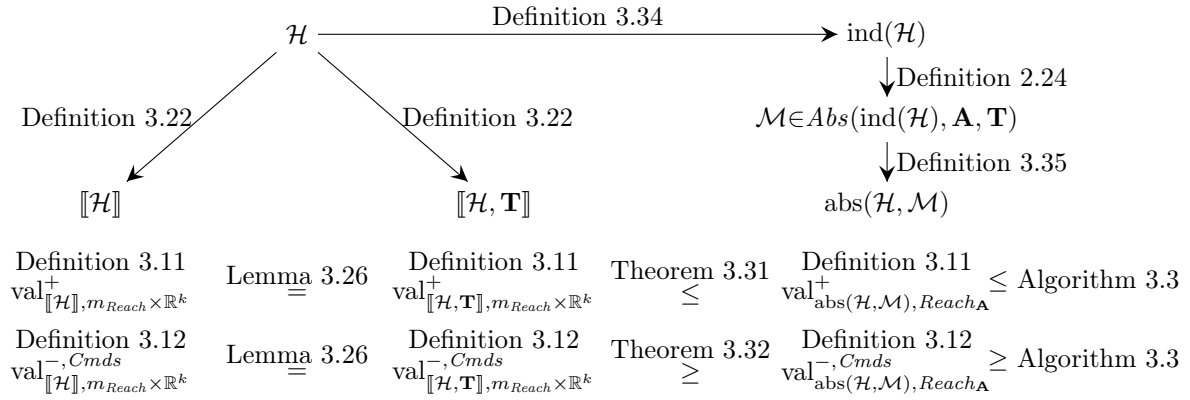


Figure 3.8: Abstraction scheme for PHAs.

3.7 Conclusion

We have described an extension of classical HAs to probabilistic HAs, in which successor states can be chosen according to finite-support distributions. This model thus involves nondeterministic behaviours intertwined with probabilistic ones. We have formalised the high-level model as well as its semantics in the form of PAs. We also compared our approach to similar existing notions of probabilistic hybrid automata from the literature. Like for the nonprobabilistic setting, we want to prove properties over the reachability of unsafe or desirable states. However, as we are now operating in a probabilistic setting, reachability is no longer a yes-or-no question. Instead, we reach sets of states with a given probability, which depends on the resolution of the nondeterminism. Because of this, we have targeted computing bounds of such reachability probabilities over all possible resolutions. For this, we have defined an appropriate notion of abstraction, and have also shown the correctness of the approach using probabilistic simulation relations. We build our abstractions on solvers for the reachability problem in classical hybrid automata of Chapter 2, for which a strong tool support exists (e.g. [HHWT97b; HHWT97a; RS07; Fre08; Fre+11]), and thus provide the first automatic framework for the analysis of general probabilistic hybrid automata. The approach we use has the advantage of orthogonality: future computational advances in hybrid automata analysis directly carry over to improvements of our framework. The practical applicability of our method was demonstrated by analysing a number of case studies. We also sketched how the probability bounds can be obtained in the finite abstractions we compute.

In Figure 3.8, we give an overview of the abstraction developed in this chapter.

4
Continuous Distributions

This chapter extends our abstraction framework to handle models in which jump targets can be chosen according to continuous probability distributions. The number of potential successor states is then uncountably large. This extension allows us to handle more expressive classes of models, in particular models of systems which takes measurements with a given precision.

This chapter is organised analogously to the previous ones. In Section 4.1 we describe nondeterministic labelled Markov processes, which will form the semantics of our extended stochastic hybrid system formalism. The formulation of the new model class is somewhat more complicated, as we also have to take measurability issues into account Then, in Section 4.2, we extend our probabilistic hybrid automata to allow the specification of continuous probability measures. We will define the semantics, and discuss how to ensure measurability. Afterwards, in Section 4.3 we will show how to overapproximate the extended model by the probabilistic hybrid automata mechanism discussed before, so that results from that model carry over to the new definition. In Section 4.4 we apply the extended analysis method to a number of case studies. Section 4.5 discusses related work and Section 4.6 concludes the chapter.

4.1 Semantic Model

In this section, we will define the semantic model for our extended class of stochastic hybrid automata. To do so, we need to extend the stochastic recapitulation of Subsection 3.1.1 by a few additional constructs.

4.1.1 Stochastic Recapitulation

A family Σ of subsets of a set Ω is a *π-system*, provided it is closed under binary intersection. This means that for each $A_1, A_2 \in \Sigma$ we have $A_1 \cap A_2 \in \Sigma$.

Given a σ-algebra Σ on Ω and $E \subseteq \Omega$, we can define the *trace σ-algebra* $\Sigma|E \stackrel{\text{def}}{=} \{A \cap E \mid A \in \Sigma\}$ as

$$\Sigma|E \stackrel{\text{def}}{=} \{A \cap E \mid A \in \Sigma\}.$$

It is known that $\Sigma|E$ is indeed a σ-algebra.

A Borel space is called *standard* if it is the Borel space on a Polish space. This includes the cases $(\mathbb{R}^k, \mathcal{B}(\mathbb{R}^k))$ of the real numbers, as well as the cases $(M \times \mathbb{R}^k, 2^M \otimes \mathcal{B}(\mathbb{R}^k))$, which we will later on use to equip the state space of stochastic hybrid automata with a measurable space.

Consider a measurable space (Ω, Σ). Remember that finite probability distributions are a special case of probability measures: a finite probability distribution $\mu\colon \Omega \to [0,1]$ extends to a probability measure $\mu\colon \Sigma \to [0,1]$ by defining $\mu(A) \stackrel{\text{def}}{=} \sum_{a \in A} \mu(a)$ for all $A \in \Sigma$. This way, also the construction $[a_1 \mapsto p_1, \ldots, a_n \mapsto p_n]$ carries over to probability measures.

We denote the set of all probability measures in the measurable space by $\Delta(\Omega)$. It can be endowed with the σ-algebra $\Delta(\Sigma)$ [Gir82] generated by the measures that, when applied to $Q \in \Sigma$, give a value in $B \in \mathcal{B}([0,1])$:

$$\Delta(\Sigma) \stackrel{\text{def}}{=} \sigma(\{\Delta^B(Q) \mid Q \in \Sigma \wedge B \in \mathcal{B}([0,1])\}) \text{ where } \Delta^B(Q) \stackrel{\text{def}}{=} \{\mu \in \Delta(\Omega) \mid \mu(Q) \in B\}.$$

Note that $\Delta(\Sigma)$ is a set of sets of probability measures. Together with $\Delta(\Omega)$, it forms the measurable space $(\Delta(\Omega), \Delta(\Sigma))$. To generate $\Delta(\Sigma)$, the sets $\Delta^{\geq q}(Q)$ with $q \in \mathbb{Q} \cap [0,1]$ suffice [Wol12, Section 3.5].

Given a σ-algebra Σ, we can define the *hit σ-algebra* [Kec95; Nai03][Wol12, Definition 4.4] over Σ by

$$\mathcal{H}(\Sigma) \stackrel{\text{def}}{=} \sigma(\{H_Q \mid Q \in \Sigma\}) \text{ where } H_Q \stackrel{\text{def}}{=} \{C \in \Sigma \mid C \cap Q \neq \emptyset\}.$$

Thus, H_Q consists of all measurable sets C which have a nonempty intersection with Q. $\mathcal{H}(\Sigma)$ is then generated by all sets of sets $\{C_1, \ldots, C_n\}$ for which there is a set Q which "hits" all C_i.

Given a finite index set I and a family $(\Sigma_i)_{i \in I}$ of σ-algebras, the *product σ-algebra* $\bigotimes_{i \in I} \Sigma_i$ is defined as

$$\bigotimes_{i \in I} \Sigma_i \stackrel{\text{def}}{=} \sigma\left(\left\{\bigtimes_{i \in I} Q_i \mid \forall i \in I.\ Q_i \in \Sigma_i\right\}\right).$$

We use $\otimes$ as an infix operator on two σ-algebras. In case of a countable index set I, w.l.o.g. $I = \mathbb{N}$, the *product σ-algebra* is defined as

$$\bigotimes_{i \in I} \Sigma_i \stackrel{\text{def}}{=} \sigma\left(\bigcup_{n=1}^{\infty} \left\{\bigtimes_{i=0}^{n} Q_i \mid \forall i, 0 \leq i \leq n.\ Q_i \in \Sigma_i\right\}\right).$$

We define

$$\Sigma^n \stackrel{\text{def}}{=} \bigotimes_{i \in \{1,\ldots,n\}} \Sigma \text{ and } \Sigma^\omega \stackrel{\text{def}}{=} \bigotimes_{i \in \mathbb{N}} \Sigma.$$

Given two measurable spaces (Ω_i, Σ_i) for $i \in \{1,2\}$, a set $A \in \Sigma_1 \otimes \Sigma_2$ and an element $a_1 \in \Omega_1$, we define the *projection* to a_1 as

$$A_{|a_1} \stackrel{\text{def}}{=} \{a_2 \in \Omega_2 \mid (a_1, a_2) \in A\}.$$

We will need the following lemma.

Lemma 4.1. *Let (Ω, Σ) be measurable space on which μ is a measure and let $f\colon \Omega \to \mathbb{R}$ be a measurable function (cf. Subsection 3.1.1). Then there exist $\omega_l, \omega_u \in \Omega$ with*

$$\int_\Omega f(\omega_l)\,\mu(\mathrm{d}\omega) \leq \int_\Omega f(\omega)\,\mu(\mathrm{d}\omega) \leq \int_\Omega f(\omega_u)\,\mu(\mathrm{d}\omega).$$

Proof. We only discuss the case of ω_u, as the other one is similar. Consider the case that f does not have a supremum. In this case, we will always find an ω_u which is large enough to fulfil $f(\omega_u) \geq \frac{\int_\Omega f(\omega)\,\mu(\mathrm{d}\omega)}{\int_\Omega 1 \mu(\mathrm{d}\omega)}$, and which thus fulfils the original equation.
In the case that f has a supremum, we have

$$\int_\Omega f(\omega)\,\mu(\mathrm{d}\omega) \leq \int_\Omega \sup f\,\mu(\mathrm{d}\omega), \tag{4.1}$$

because $f(\omega) \leq \sup f$ for all $\omega \in \Omega$. If the maximum of f exists, it is equal to the supremum and we are done.
Thus, assume that f has a supremum but no maximum. In this case, Equation 4.1 is strict, we have

$$\int_\Omega f(\omega)\,\mu(\mathrm{d}\omega) < \int_\Omega \sup f\,\mu(\mathrm{d}\omega), \tag{4.2}$$

and there exists a sequence $(\omega_i)_{i\geq 0}$ with $\lim_{i\to\infty} f(\omega_i) = \sup f$. Because of this, we have

$$\lim_{i\to\infty} \int_\Omega f(\omega_i)\,\mu(\mathrm{d}\omega) = \int_\Omega \sup f\,\mu(\mathrm{d}\omega). \tag{4.3}$$

From Equation 4.2 and Equation 4.3 we know that there is some i for which $\omega_u = \omega_i$ fulfils the original equation. □

4.1.2 Nondeterministic Labelled Markov Processes

With the extensions defined above, we can now define nondeterministic labelled Markov processes [Wol12; D'A+09], which will later on form the semantics of stochastic hybrid automata with continuous distributions.

Definition 4.2 (NLMP)**.** *A* nondeterministic labelled Markov process (NLMP) *is a tuple of the form*

$$\mathcal{M} = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T}),$$

where

- *S is a set of* states*,*
- *Σ_S is a σ-algebra over S,*
- *$\overline{s} \in S$ is the* initial state*,*
- *Act is a set of* actions*,*
- *Σ_{Act} is a σ-algebra over Act, where for all $a \in Act$ we have $\{a\} \in \Sigma_{Act}$,*

- $\mathcal{T}\colon S \to (\Sigma_{Act} \otimes \Delta(\Sigma_S))$ *is the* Σ_S-$\mathcal{H}(\Sigma_{Act} \otimes \Delta(\Sigma_S))$-*measurable* transition matrix.

We define $\mathcal{T}(s,a) \stackrel{\text{def}}{=} \mathcal{T}(s)_{|a}$. *An NLMP is called* finitely probabilistic *in case for each* $s \in S$, $a \in Act$ *and* $\mu \in \mathcal{T}(s,a)$ *we have* $\mu \in Distr(S)$.

We need to equip the labels with a σ-algebra to ensure measurability of $\mathcal{T}$. Even if for all labels a we have that $\mathcal{T}(\cdot,a)\colon S \to \Delta(\Sigma_S)$ is measurable, this does not imply that their combination $\mathcal{T}$ is measurable, if the set of labels is uncountably large. For an example of an NLMP in which each individual $\mathcal{T}(\cdot,a)$ is indeed measurable, but $\mathcal{T}$ itself is not, see [Wol12, Example 4.10]. However, from the measurability of the transition matrix $\mathcal{T}$ it follows that each $\mathcal{T}(\cdot,a)$ is Σ_S-$\mathcal{H}(\Delta(\Sigma_S))$-measurable [Wol12, Proposition 4.3]. Written without measurability requirements, $\mathcal{T}(\cdot,a)$ is a function with signature $S \to 2^{\Delta(S)}$. Thus, for each state $s \in S$ and action $a \in Act$ we obtain a set of distributions $\mathcal{T}(s,a)$ to choose from. A finitely probabilistic NLMP has similar expressive power to a PA. The difference is that the measurability requirements still need to hold, and, as seen later, a larger class of schedulers is supported.

Example 4.3. *Let*

$$\mathcal{M} \stackrel{\text{def}}{=} (S, \Sigma_S, \overline{s}, \{\tau\}, \Sigma_{Act}, \mathcal{T})$$

be an NLMP with

- $S \stackrel{\text{def}}{=} \{\overline{s}\} \uplus [0,1] \uplus \{s_b\}$,
- $\Sigma_S \stackrel{\text{def}}{=} \sigma(\mathcal{B}([0,1]) \uplus \{\overline{s}\} \uplus \{s_b\})$,
- $\Sigma_{Act} \stackrel{\text{def}}{=} \{\emptyset, \{\tau\}\}$,
- $\mathcal{T}(\overline{s},\tau) \stackrel{\text{def}}{=} \{\mu\}$ *where* μ *is the uniform distribution over* $[0,1]$, *with* $\mu([a,b]) = b-a$ *for all* a,b *with* $0 \le a \le b \le 1$,
- *for* $s \in \mathfrak{V}$, *we let* $\mathcal{T}(s,\tau) \stackrel{\text{def}}{=} \{[s_b \mapsto 1]\}$, *and*
- *for* $s \notin \mathfrak{V}$ *we define* $\mathcal{T}(s,\tau) \stackrel{\text{def}}{=} \{[\overline{s} \mapsto 1]\}$.

It is known that a nonmeasurable set, a so-called Vitali set $\mathfrak{V} \subset [0,1]$, *exists. All* $\mathcal{T}(\cdot,\tau)$ *are singleton sets, so there is no nondeterminism in the example. Thus, the probability to reach* s_b *from* $\overline{s}$ *should be uniquely defined. However, this probability does not exist, because it depends on the measure of the nonmeasurable Vitali set. Discrete measures (which are the only ones allowed in PAs) eliminate this problem, since they discard all but a finite set of points out of it. For instance, if instead of using the uniform distribution we set* $\mu(s_1) \stackrel{\text{def}}{=} \mu(s_2) \stackrel{\text{def}}{=} \mu(s_3) \stackrel{\text{def}}{=} \frac{1}{3}$ *for some choice of* $s_1, s_2 \in \mathfrak{V}$ *and* $s_3 \notin \mathfrak{V}$, $\mu(\cdot) \stackrel{\text{def}}{=} 0$ *otherwise, the reachability probability is* $\frac{2}{3}$, *although the model still does not fulfil the measurability requirements.* △

As for PAs, one can define the set of paths of an NLMP and the σ-algebra of infinite paths (cf. [Wol12, Chapter 7]). However, the construction is somewhat different. Because we now allow continuous measures, the probabilities of all individual finite paths might be zero, which means that only certain sets of finite paths can be assigned a probability other than zero. To take this into account, we use the σ-algebras of states and actions to construct the σ-algebra of finite paths, instead of starting with cylinders of individual finite paths.

Definition 4.4. *Consider an NLMP* $\mathcal{M} = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T})$. *The measurable space of paths of length* $n \in \mathbb{N}$ *is*

$$(Path^0_{\mathcal{M}}, \Sigma_{Path^0_{\mathcal{M}}}) \stackrel{\text{def}}{=} (\{\overline{s}\}, \{\{\overline{s}\}, \emptyset\}),$$
$$(Path^n_{\mathcal{M}}, \Sigma_{Path^n_{\mathcal{M}}}) \stackrel{\text{def}}{=} (\{\overline{s}\} \times (Act \times \Delta(S) \times S)^n, \{\{\overline{s}\}, \emptyset\} \otimes (\Sigma_{Act} \otimes \Delta(\Sigma_S) \otimes \Sigma_S)^n).$$

Then, we define

$$(Path^{\text{fin}}_{\mathcal{M}}, \Sigma_{Path^{\text{fin}}_{\mathcal{M}}}) \stackrel{\text{def}}{=} \left(\biguplus_{n \in \mathbb{N}} Path^n_{\mathcal{M}}, \biguplus_{n \in \mathbb{N}} \Sigma_{Path^n_{\mathcal{M}}} \right).$$

We define the measurable space of infinite paths as

$$(Path^{\text{inf}}_{\mathcal{M}}, \Sigma_{Path^{\text{inf}}_{\mathcal{M}}}) \stackrel{\text{def}}{=} (\{\overline{s}\} \times (Act \times \Delta(S) \times S)^\omega, \{\{\overline{s}\}, \emptyset\} \otimes (\Sigma_{Act} \otimes \Delta(\Sigma_S) \otimes \Sigma_S)^\omega).$$

For $\beta \in Path^{\text{fin}}_{\mathcal{M}} \uplus Path^{\text{inf}}_{\mathcal{M}}$ *we write* $\text{last}(\beta)$*, etc. as for PAs.*

By setting $\Sigma_S \stackrel{\text{def}}{=} 2^S$, we could in principle see the definition of the measurable space of paths of PAs (cf. Definition 3.8) as a special case of the definition above. We remark that we can ignore the restriction of Definition 3.5 that for a finite or infinite path $\beta = s_0\, a_0\, \mu_0 \ldots$ we have for all i with $0 \leq i < n$ that $\mu_i \in \mathcal{T}(s_i, a_i)$. As for NLMPs, paths which do not fulfil this restriction can be defined as being valid; they will then have a probability measure of 0.

Later on however, this most general σ-algebra 2^S would be too general, because not every $A \in 2^S$ is measurable by a continuous probability measure, which can be used in this model class. This concerns for instance Vitali sets which we considered in Example 4.3. Thus, in the later definition of the later definition of the semantics of stochastic hybrid automata as NLMPs, we will use Σ_S smaller than 2^S.

We state the definition of schedulers for NLMPs.

Definition 4.5. *A* scheduler *for an NLMP* $\mathcal{M} = (S, \Sigma_S, s_0, Act, \Sigma_{Act}, \mathcal{T})$ *is a function* $\sigma \colon Path^{\text{fin}}_{\mathcal{M}} \to \Delta(Act \times \Delta(S))$. *We require it to be* $\Sigma_{Path^{\text{fin}}_{\mathcal{M}}}$*-*$\mathcal{H}(\Delta(Act \times \Delta(S)))$*-measurable, and demand that for all* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ *we have* $\sigma(\beta)((Act \times \Delta(S)) \setminus \mathcal{T}(\text{last}(\beta))) = 0$. *With* $Sched_{\mathcal{M}}$ *we denote the sets of all schedulers of* $\mathcal{M}$.

A scheduler is finitely probabilistic, *if for all* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ *there are* $(a_1, \mu_1), \ldots, (a_n, \mu_n) \in Act \times \Delta(S)$ *with*

$$\sigma(\beta)(\{(a_1, \mu_1), \ldots, (a_n, \mu_n)\}) = 1.$$

We can thus interpret it as being of the form $\sigma \colon Path^{\text{fin}}_{\mathcal{M}} \to Distr(Act \times \Delta(S))$.

A scheduler is Act_{semi}-semi finitely probabilistic *for a given set* Act_{semi}, *if for all* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ *there are* $(a_1, \mu_1), \ldots, (a_n, \mu_n) \in Act \times \Delta(S)$ *and* $A \subseteq Act_{\text{semi}} \times \Delta(S)$ *with*

$$\sigma(\beta)(\{(a_1, \mu_1), \ldots, (a_n, \mu_n)\} \cup A) = 1.$$

In contrast to schedulers for PAs (cf. Definition 3.7), schedulers for NLMPs can thus apply a continuous probability distribution to choose over successor transitions. This will later turn out to be useful, as it will allow abstractions of stochastic hybrid automata with continuous distributions to follow exactly the behaviour of the original model. To ensure well-defined probability measures on the paths, schedulers are required to be measurable functions. Finitely probabilistic schedulers only choose from a finite number of successor distributions, like the schedulers of PAs. For semi finitely probabilistic schedulers, this restriction is only required to hold for transitions which are not Act_{semi}-labelled. We note that semi finitely probabilistic schedulers are not necessarily fair.

As for PAs, a scheduler induces a measure on sets of paths.

Definition 4.6. *Consider an NLMP $\mathcal{M} = (S, \Sigma_S, s_0, Act, \Sigma_{Act}, \mathcal{T})$, a scheduler $\sigma \in Sched_{\mathcal{M}}$ and a finite path $\overline{\beta} \in Path^{\text{fin}}_{\mathcal{M}}$. The* combined transition *$Tr_{\mathcal{M},\sigma,\overline{\beta}}(\cdot)\colon (\Sigma_{Act} \otimes \Delta(\Sigma_S) \otimes \Sigma_S) \to [0,1]$ is defined as*

$$Tr_{\mathcal{M},\sigma,\overline{\beta}}(A, \xi, Q) \stackrel{\text{def}}{=} \int_{A \times \xi} \mu(Q)\, \sigma(\overline{\beta})(\mathrm{d}a, \mathrm{d}\mu),$$

where $A \in \Sigma_{Act}$, $\xi \in \Delta(\Sigma_S)$ and $Q \in \Sigma_S$.

We define the conditional finite path measure *$Pr_{\mathcal{M},\sigma,\overline{\beta}}\colon \Sigma_{Path^{\text{fin}}_{\mathcal{M}}} \to [0,1]$ on* finite cylinders*. Given a family $\langle M_i \rangle_{i=0}^{n}$ of sets with $M_0 = \{\overline{s}\}$ and $M_i \in (\Sigma_{Act} \otimes \Delta(\Sigma_S) \otimes \Sigma_S)$ for all $i > 0$ and $M \stackrel{\text{def}}{=} \bigtimes_{i=0}^{n} M_i$, we let*

$$Pr_{\mathcal{M},\sigma,\overline{\beta}}(M) \stackrel{\text{def}}{=} Pr^{n}_{\mathcal{M},\sigma,\overline{\beta}}(M),$$

where in case $n \leq |\overline{\beta}|$, given the only $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ with $\beta \leq \overline{\beta}$ and $|\beta| = n$, we have

$$Pr^{n}_{\mathcal{M},\sigma,\overline{\beta}}(M) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } \beta \in M, \\ 0 & \text{else,} \end{cases}$$

and if $n > |\overline{\beta}|$ we define inductively

$$Pr^{n}_{\mathcal{M},\sigma,\overline{\beta}}\left(\left(\bigtimes_{i=0}^{n-1} M_i\right) \times M_n\right) \stackrel{\text{def}}{=} \int_{(\bigtimes_{i=0}^{n-1} M_i)} Tr_{\mathcal{M},\sigma,\beta}(M_n)\, Pr^{n-1}_{\mathcal{M},\sigma,\overline{\beta}}(\mathrm{d}\beta).$$

The measure extends uniquely to a measure $Pr_{\mathcal{M},\sigma,\overline{\beta}}\colon \Sigma_{Path^{\text{inf}}_{\mathcal{M}}} \to [0,1]$ on the σ-algebra of infinite paths if we require that for all such finite cylinders M and $M_{\text{inf}} \stackrel{\text{def}}{=} M \times (\Sigma_{Act} \times \Delta(\Sigma_S) \times \Sigma_S)^{\omega}$ we have

$$Pr_{\mathcal{M},\sigma,\overline{\beta}}(M_{\text{inf}}) = Pr_{\mathcal{M},\sigma,\overline{\beta}}(M).$$

The (nonconditional) path measure *is then defined as*

$$Pr_{\mathcal{M},\sigma} \stackrel{\text{def}}{=} Pr_{\mathcal{M},\sigma,\overline{s}}.$$

The state, action and distribution processes $X^{\mathcal{M}}\colon (Path^{\text{inf}}_{\mathcal{M}} \times \mathbb{N}) \to S$, $Y^{\mathcal{M}}\colon (Path^{\text{inf}}_{\mathcal{M}} \times \mathbb{N}) \to Act$ and $Z^{\mathcal{M}}\colon (Path^{\text{inf}}_{\mathcal{M}} \times \mathbb{N}) \to \Delta(S)$ are defined as for PAs. We define fair paths in the same way.

The definition above follows mainly the one of Wolovick [Wol12, Chapter 7], who also shows well-definedness for the constructions above. A minor extension is that we define path measures conditional on initial finite paths.

Like for PAs, we can now also define when a scheduler is fair:

Definition 4.7. *A scheduler $\sigma \in Sched_{\mathcal{M}}$ of an NLMP $\mathcal{M}$ is called* Act_{fair}-fair *if it is* Act_{fair}*-semi finitely probabilistic and we have*

$$Pr_{\mathcal{M},\sigma}(Path_{\mathcal{M}}^{Act_{\text{fair}}}) = 1.$$

By $Sched_{\mathcal{M}}^{Act_{\text{fair}}}$, we denote the set of all Act_{fair}-fair schedulers of $\mathcal{M}$.

We remark that we require fair schedulers to be semi finitely probabilistic, to later ensure the comparability to schedulers of PAs.

We need to slightly adapt the definition of the properties we consider, so as to take measurability issues into account.

Definition 4.8. *Consider an NLMP $\mathcal{M} = (S, \Sigma_S, s_0, Act, \Sigma_{Act}, \mathcal{T})$. Given a measurable set $Reach \in \Sigma_S$ of states and a scheduler $\sigma \in Sched_{\mathcal{M}}$, we let*

$$\text{val}_{\mathcal{M},Reach}^{\sigma} \stackrel{\text{def}}{=} Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)$$

denote the reachability probability value *for Reach under σ. Further, let*

$$\text{val}_{\mathcal{M},Reach}^{+} \stackrel{\text{def}}{=} \sup_{\sigma \in Sched_{\mathcal{M}}} Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)$$

denote the maximal reachability probability *for Reach. For $Act_{\text{fair}} \subseteq Act$ we define the* minimal Act_{fair}-restricted reachability probability *for Reach as*

$$\text{val}_{\mathcal{M},Reach}^{-,Act_{\text{fair}}} \stackrel{\text{def}}{=} \inf_{\sigma \in Sched_{\mathcal{M}}^{Act_{\text{fair}}}} Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach).$$

The following lemma shows that the set of paths which finally reach a set *Reach* is measurable. This ensures that the probability values of Definition 4.8 do exist.

Lemma 4.9. *Given an NLMP $\mathcal{M} = (S, \Sigma_S, s_0, Act, \Sigma_{Act}, \mathcal{T})$, a measurable set $Reach \in \Sigma_S$ of states and a scheduler $\sigma \in Sched_{\mathcal{M}}$, the set $\{\beta \in Path_{\mathcal{M}}^{\text{inf}} \mid \exists i \geq 0.\ X_i^{\mathcal{M}}(\beta) \in Reach\}$ is measurable.*

Proof. We have

$$\begin{aligned}
&\{\beta \in Path_{\mathcal{M}}^{\text{inf}} \mid \exists i \geq 0.\ X_i^{\mathcal{M}}(\beta) \in Reach\} \\
&= \bigcup_{i=0}^{\infty} \{\beta \in Path_{\mathcal{M}}^{\text{inf}} \mid X_i^{\mathcal{M}}(\beta) \in Reach\} \\
&= \bigcup_{i=0}^{\infty} \left(\left(\bigtimes_{j=0}^{i-1} S \times Act \times \Delta(S) \right) \times Reach \times \left(\bigtimes_{j=i+1}^{\infty} Act \times \Delta(S) \times S \right) \right).
\end{aligned}$$

Thus, the set is measurable as it can be expressed as a countable union of cylinders. □

We will later use the following lemma when abstracting our extended stochastic hybrid automata to the PHAs of Chapter 3, because in PHAs we are always restricted to finitely probabilistic schedulers.

Lemma 4.10. *Given a finitely probabilistic NLMP $\mathcal{M} = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T})$ with a set of states $Reach \in \Sigma_S$ and a finite subset of actions $Act_{\text{fair}} \subseteq Act$, for each scheduler $\sigma \in Sched_{\mathcal{M}}$ there is a finitely probabilistic scheduler σ_u with*

$$\text{val}^{\sigma}_{\mathcal{M},Reach} \leq \text{val}^{\sigma_u}_{\mathcal{M},Reach}.$$

Also, if σ is Act_{fair}-fair, we always find an Act_{fair}-fair finitely probabilistic scheduler σ_l with

$$\text{val}^{\sigma}_{\mathcal{M},Reach} \geq \text{val}^{\sigma_l}_{\mathcal{M},Reach}.$$

Proof. We consider the case $\text{val}^{\sigma}_{\mathcal{M},Reach} \geq \text{val}^{\sigma_l}_{\mathcal{M},Reach}$. Using Fubini's theorem, for $n \in \mathbb{N}$ we have

$$\begin{aligned}
&\text{val}^{\sigma}_{\mathcal{M},Reach} \\
&= Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach) \\
&= \int_{Path^{n+1}_{\mathcal{M}}} Pr_{\mathcal{M},\sigma,\beta}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)\, Pr_{\mathcal{M},\sigma}(\mathrm{d}\beta) \qquad (4.4) \\
&= \int_{Path^{n}_{\mathcal{M}}} \int_{Act\times\Delta(S)\times S} Pr_{\mathcal{M},\sigma,\beta\, a\, \mu\, s}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)\, \mu(\mathrm{d}s)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu)\, Pr_{\mathcal{M},\sigma}(\mathrm{d}\beta).
\end{aligned}$$

For $n \in \mathbb{N}$, we define a sequence of fair schedulers $\sigma_n \colon Path^{\text{fin}}_{\mathcal{M}} \to \Delta(Act \times \Delta(S))$ inductively as follows: for $\beta \in Path^{\text{fin}}_{\mathcal{M}}$, we let

- $\sigma_n(\beta) \stackrel{\text{def}}{=} \sigma_{n-1}(\beta)$ if $|\beta| < n$,
- $\sigma_n(\beta) \stackrel{\text{def}}{=} \sigma(\beta)$ for $|\beta| > n$ and
- in case $|\beta| = n$,
  - for a measurable set $A \subseteq (Act \setminus Act_{\text{fair}}) \times \Delta(S)$ we let $\sigma_n(\beta)(A) \stackrel{\text{def}}{=} \sigma(\beta)(A)$,
  - if $\sigma_{n-1}(\beta)(Act_{\text{fair}} \times \Delta(S)) > 0$ then for $a_{\text{fair}} \in Act_{\text{fair}}$ and $\mu_{\text{fair}} \in \Delta(S)$ defined below, we define $\sigma_n(\beta)(a, \mu_{\beta,a}) \stackrel{\text{def}}{=} \sigma(\beta)(Act_{\text{fair}} \times \Delta(S))$,

  which uniquely defines the probability of other sets $A \in \Sigma_{Act} \otimes \Delta(\Sigma_S)$.

We set $\sigma_{-1} \stackrel{\text{def}}{=} \sigma$ and using Lemma 4.1 then choose a_{fair} and μ_{fair} with

$$\begin{aligned}
&\int_{Act_{\text{fair}}\times\Delta(S)\times S} Pr_{\mathcal{M},\sigma_{n-1},\beta\, a\, \mu\, s}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)\, \mu(\mathrm{d}s)\, \sigma_{n-1}(\beta)(\mathrm{d}a, \mathrm{d}\mu) \\
&\geq \int_{Act_{\text{fair}}\times\Delta(S)\times S} Pr_{\mathcal{M},\sigma_{n-1},\beta\, a_{\text{fair}}\, \mu_{\text{fair}}\, s}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)\, \mu(\mathrm{d}s)\, \sigma_{n-1}(\beta)(\mathrm{d}a, \mathrm{d}\mu) \\
&= \int_S Pr_{\mathcal{M},\sigma_{n-1},\beta\, a_{\text{fair}}\, \mu_{\text{fair}}\, s}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)\, \mu_{\text{fair}}(\mathrm{d}s).
\end{aligned}$$

For the choice of a_{fair} and μ_{fair}, we also require that if

$$Pr_{\mathcal{M},\sigma_{n-1}}(\{\beta\}) > 0,$$

then for all $s \in S$ with $\mu_{\text{fair}}(s) > 0$ we have

$$Pr_{\mathcal{M},\sigma_n,\beta\, a_{\text{fair}}\, \mu_{\text{fair}}\, s}(Path_{\mathcal{M}}^{Act_{\text{fair}}}) = 1.$$

Fulfilling this additional requirement is always possible. Let

$$\Omega \stackrel{\text{def}}{=} \left\{(a,\mu) \in Act_{\text{fair}} \times \Delta(S) \mid \int_S Pr_{\sigma_{n-1},\beta\, a\, \mu\, s}(Path_{\mathcal{M}}^{Act_{\text{fair}}})\, \mu_{\text{fair}}(\mathrm{d}s) = 1\right\}.$$

We have that $\sigma_{n-1}(\beta)(\Omega) = 1$, otherwise the probability of fair paths under σ_{n-1} would be lower than 1. Because of this, it suffices to integrate over this set, and we have

$$\begin{aligned}&\int_{Act_{\text{fair}}\times\Delta(S)\times S} Pr_{\mathcal{M},\sigma_{n-1},\beta\, a\, \mu\, s}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)\, \mu(\mathrm{d}s)\, \sigma_{n-1}(\beta)(\mathrm{d}a,\mathrm{d}\mu)\\ =&\int_{\Omega\times S} Pr_{\mathcal{M},\sigma_{n-1},\beta\, a\, \mu\, s}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach)\, \mu(\mathrm{d}s)\, \sigma_{n-1}(\beta)(\mathrm{d}a,\mathrm{d}\mu).\end{aligned}$$

Thus, in Lemma 4.1 we can choose a_{fair} and μ_{fair} from this Ω rather than from $Act_{\text{fair}} \times \Delta(S)$. The measurabiltiy of Ω is not an issue. We have $\sigma_{n-1}(\beta)((Act \times \Delta(S)) \setminus \Omega) = 0$. Thus, either $(Act \times \Delta(S)) \setminus \Omega$ is measurable, or we can construct a *completion* of this measure, in such a way that $(Act \times \Delta(S)) \setminus \Omega$ is measurable there. In turn, Ω is also measurable there.

We can now define a scheduler σ_l which is completely finitely probabilistic by setting

$$\sigma_l(\beta) \stackrel{\text{def}}{=} \sigma_{|\beta|}(\beta)$$

for all $\beta \in Path_{\mathcal{M}}^{\text{fin}}$. By induction and Equation 4.4, one can show that

$$Pr_{\mathcal{M},\sigma_l}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach) \leq Pr_{\mathcal{M},\sigma}(\exists i \geq 0.\ X_i^{\mathcal{M}} \in Reach).$$

The scheduler constructed this way is also Act_{fair}-fair: because we assumed the NLMP to be finitely probabilistic and the scheduler we construct is also finitely probabilistic, the probability is concentrated in cylinders of finite paths β with positive probability. For $\beta \in Path_{\mathcal{M}}^{\text{fin}}$ with $Pr_{\mathcal{M},\sigma_l}(\{\beta\}) > 0$ we have

$$Pr_{\mathcal{M},\sigma_l,\beta}\left(\left\{\beta' \in Path_{\mathcal{M}}^{\text{inf}} \mid \beta \leq \beta' \wedge \exists i > |\beta|.\ \beta'[i] \in Act_{\text{fair}}\right\}\right) = 1, \tag{4.5}$$

that is, almost surely we finally execute a fair action when having started with β. This is the case, because from $Pr_{\mathcal{M},\sigma_l}(\{\beta\}) > 0$ we know that scheduler $\sigma_{|\beta|}$ must have chosen a fair continuation of β, which implies that Equation 4.5 is fulfilled when replacing σ_l by $\sigma_{|\beta|}$ there. The following schedulers σ_n with $n > |\beta|$ do not change the probabilities of choices of nonfair actions, and thus leave the probability of reaching fair actions unchanged. In turn, the property also holds for σ_l. Now assume that

$Pr_{\mathcal{M},\sigma_l}(Path_{\mathcal{M}}^{Act_{\text{fair}}}) < 1$. Then there must be some $\beta \in Path_{\mathcal{M}}^{\text{fin}}$ with $Pr_{\mathcal{M},\sigma_l}(\{\beta\}) > 0$ but for which Equation 4.5 is invalid. However, we have already seen that Equation 4.5 holds, which means that σ_l is indeed fair.

The proof of the case $\text{val}^{\sigma}_{\mathcal{M},Reach} \leq \text{val}^{\sigma_u}_{\mathcal{M},Reach}$ is similar but simpler, because we do not have to take fairness into account. One notable difference is that in this case we also have to modify the transitions labelled with nonfair actions, because for this case we do not assume the scheduler to be semi finitely probabilistic. □

4.2 Stochastic Hybrid Automata

In this section, we will define the extended hybrid automata, to allow handling continuous distributions. We will call them *stochastic* hybrid automata, to distinguish them from the former *probabilistic* hybrid automata. To guarantee well-defined behaviour, that is to ensure that the semantics fulfils the requirements of NLMPs, we will have to put measurability restrictions on the automata components.

We extend the probabilistic guarded commands of Definition 3.20 in two different directions. The first one resembles the definition of probabilistic guarded commands in PHAs (cf. Definition 3.20) in that it only allows one to use finite-support probability distributions. However, in contrast to the previous setting, we have to add some measurability requirements to ensure the well-definedness of the semantics of our extended model class. The second extension of probabilistic guarded commands does not allow nondeterminism, but does allow continuous probability distributions. In addition, we have to put measurability restrictions on the post operators.

Definition 4.11. *For a dimension $k \in \mathbb{N}^+$ and a finite set of modes M, let $S \stackrel{\text{def}}{=} M \times \mathbb{R}^k$ and $\Sigma_S \stackrel{\text{def}}{=} 2^M \otimes \mathcal{B}(\mathbb{R}^k)$. A* measurable finite guarded command *is of the form*

$$g \to p_1 : u_1 + \ldots + p_n : u_n,$$

where

- *$g \in \Sigma_S$ is a* guard*,*
- *we have $p_i \geq 0$ for $1 \leq i \leq n$,*
- *we have $\sum_{i=1}^{n} p_i = 1$,*
- *the* updates *$u_i \colon S \to \Sigma_S$ are Σ_S-$\mathcal{H}(\Sigma_S)$-measurable, and*
- *$u_i(s) \neq \emptyset$ for $1 \leq i \leq n$ if $s \in g$.*

A measurable continuous guarded command *is of the form*

$$g \to M,$$

where

- *g is as for finite commands, and*
- *$M \colon S \to \Delta(S)$ is a Σ_S-$\Delta(\Sigma_S)$-measurable function.*

A post operator $Post\colon (\mathbb{R}^k \times \mathbb{R}_{\geq 0}) \to 2^{\mathbb{R}^k}$ *is called* measurable *if for the function* $T_{Post}\colon \mathbb{R}^k \to (\mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \mathcal{B}(\mathbb{R}^k))$ *with*

$$T_{Post}(v) \stackrel{\text{def}}{=} \{(\mathtt{t}, v') \mid \mathtt{t} \in \mathbb{R}_{\geq 0} \wedge v' \in Post(v, \mathtt{t})\}$$

we have

- $T_{Post}(v) \in \mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \mathcal{B}(\mathbb{R}^k)$*, for* $v \in \mathbb{R}^k$*, and*
- *the function is* $\mathcal{B}(\mathbb{R}^k)$-$\mathcal{H}(\mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \mathcal{B}(\mathbb{R}^k))$*-measurable.*

It might seem like a severe restriction that these guarded commands allow either a probabilistic choice with finite support but uncountable nondeterminism, or have a single distribution with continuous support. This is however not the case, as more complex distributions can be simulated by a guarded command of the first type followed by one of the second. The division into the two forms will ease the specification of models in our new class, and will simplify the derivation of theoretical results.

Measurability of most of the above model constituents can be guaranteed by considering o-minimal definable sets, following an idea of Martin Fränzle. General results connecting o-minimal definability with measurability [BP98; BO04] show that a sufficient criterion for the above T_{Post} being measurable is that it is definable in some o-minimal theory over the reals. In practice, this holds for the T_{Post} manipulated by hybrid model checkers: all current hybrid model checkers tackle overapproximations of the set of states reachable by timed transitions via sets definable in o-minimal theories over the reals. This includes finite unions of rectangular boxes, zonotopes, polyhedra, ellipsoids, and differential invariants. Because of this, in the models already overapproximated by hybrid solvers measurability concerns are not given. In a nutshell, the general results connecting o-minimality with measurability consider the standard parts [BP98] of o-minimal theories and shows them to be Borel measurable. This, together with the fact that the standard part $\mathrm{st}(A)$ satisfies $\mathrm{st}(A) = A$ provided $A \subseteq \mathbb{R}^k$ [BO04] implies that relations definable by o-minimal theories over the reals are Borel measurable [BP98]. Hence, T_{Post} is measurable if described in some o-minimal theory. Notably, even if each $T_{Post}(v)$ were a measurable set, this does not imply measurability of T_{Post} itself, which is why we need to require it.

With there preparations, we are now ready to define the extended model class.

Definition 4.12. *A* stochastic hybrid automaton (SHA) *is defined to be a tuple*

$$\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}}, Cmds_{\text{cts}}),$$

where

- M *is a finite set of* modes*,*
- $k \in \mathbb{N}^+$ *is the* dimension *of the automaton,*
- $\overline{m} \in M$ *is the* initial mode*,*
- *for each* $m \in M$*, we have that* $Post_m$ *is a measurable post operator,*
- $Cmds_{\text{fin}}$ *is a set of measurable finite guarded commands,*

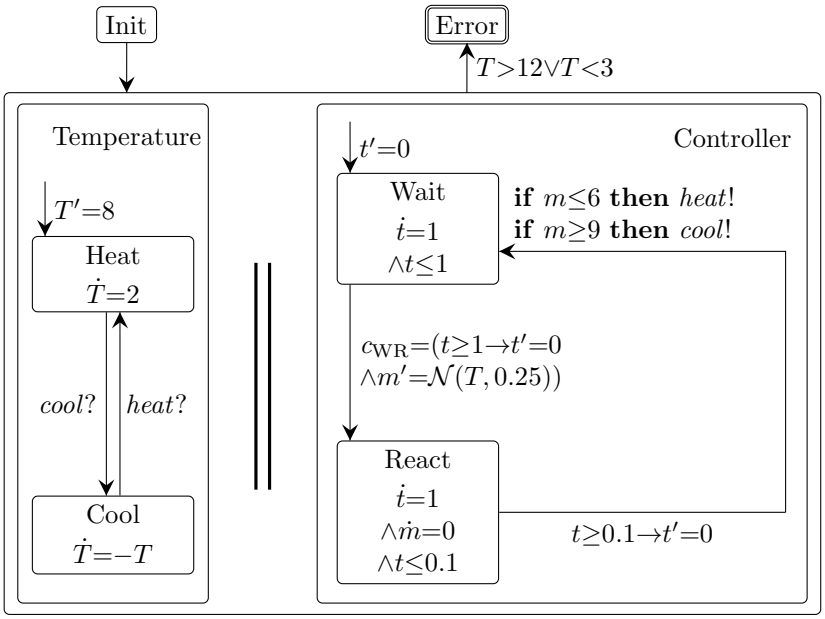


Figure 4.1: SHA modelling a thermostat.

- *$Cmds_{\mathrm{cts}}$ is a set of measurable continuous guarded commands, and*
- *for $s=(m,v)$ with $Post_m(v,\mathtt{t})=\{v\}$ for all $\mathtt{t}\in\mathbb{R}_{\geq 0}$, there is a (finite or continuous) command with guard g with $s\in g$.*

The notable difference to our basic model of Definition 3.20 is the extension by the continuous guarded commands and the measurability restrictions.

Example 4.13. *In Figure 4.1 we describe a variant of the thermostat model from Figure 3.3. It consists of the parallel composition of two components. The modes of the automaton are thus pairs of modes of these two components, e.g.* (Heat, Wait)*, with the exception of* Init *and* Error*. The component* Temperature *models the temperature of a room. Similarly to the model of Figure 3.3, there are two modes, one in which the temperature increases and one in which it decreases.* Controller *is supposed to ensure that the temperature stays in a given range. If it does not succeed with this, the model can move to the* Error *mode. The controller firstly waits for time* 1*, then measures the temperature with a (continuous) normal distribution with standard deviation* 0.25*. After a further delay of* 0.1*, it changes the state of* Temperature *accordingly, and at the same time moves back to its waiting state.*

In contrast to the previous example, there is no longer an explicit probability with which we move to Error*. Instead, because of the perturbation of measurements, there is a chance that the controller does not react appropriately to the actual temperature, which in turn might become too low or too high. If the measurement of the temperature were completely accurate, one would indeed never reach* Error*.* △

The semantics of SHAs can then be defined in terms of NLMPs.

Definition 4.14. *The* semantics *of an SHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}}, Cmds_{\text{cts}})$ *is an NLMP*

$$[\![\mathcal{H}]\!] = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T}),$$

where

- $S \stackrel{\text{def}}{=} M \times \mathbb{R}^k$,
- $\Sigma_S \stackrel{\text{def}}{=} 2^M \otimes \mathcal{B}(\mathbb{R}^k)$,
- $Act \stackrel{\text{def}}{=} \mathbb{R}_{\geq 0} \uplus Cmds_{\text{fin}} \uplus Cmds_{\text{cts}}$,
- $\Sigma_{Act} \stackrel{\text{def}}{=} \sigma(\mathcal{B}(\mathbb{R}_{\geq 0}) \uplus 2^{Cmds_{\text{fin}} \uplus Cmds_{\text{cts}}})$,
- *for* $s = (m, v) \in S$ *we have*
  - *for* $c = (g \to (p_1 : u_1 + \ldots + p_n : u_n)) \in Cmds_{\text{fin}}$ *we have* $\mathcal{T}(s, c) \stackrel{\text{def}}{=} \emptyset$ *if* $s \notin g$ *and else:*

$$\mathcal{T}(s, c) \stackrel{\text{def}}{=} \{[s_1' \mapsto p_1, \ldots, s_n' \mapsto p_n] \mid s_1' \in u_1(s), \ldots, s_n' \in u_n(s)\},$$

  - *for* $c = (g \to M) \in Cmds_{\text{cts}}$ *we have* $\mathcal{T}(s, c) \stackrel{\text{def}}{=} \emptyset$ *if* $s \notin g$ *and else:*

$$\mathcal{T}(s, c) \stackrel{\text{def}}{=} \{M(s)\},$$

  - *for* $\mathtt{t} \in \mathbb{R}_{\geq 0}$ *we have:*

$$\mathcal{T}(s, \mathtt{t}) \stackrel{\text{def}}{=} \{[(m, v') \mapsto 1] \mid v' \in Post_m(v, \mathtt{t})\}.$$

We have to show that the semantics of an SHA is indeed an NLMP, that is that all the requirements of Definition 4.2 are fulfilled. [Wol12, Lemma 5.8] contains an according statement for the slightly different semantics of SHAs considered there.

Lemma 4.15. *If* $\mathcal{H}$ *is an SHA, then* $[\![\mathcal{H}]\!]$ *is an NLMP.*

Proof. Assume that $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}}, Cmds_{\text{cts}})$ and further that $[\![\mathcal{H}]\!] = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T})$. The requirements on states, actions and the σ-algebras on them are obviously fulfilled. We have to show that for the transition matrix $\mathcal{T} : S \to (\Sigma_{Act} \otimes \Delta(\Sigma_S))$ for $s \in S$ we have $\mathcal{T}(s) \in \Sigma_{Act} \otimes \Delta(\Sigma_S)$ and that $\mathcal{T}$ is Σ_S-$\mathcal{H}(\Sigma_{Act} \otimes \Delta(\Sigma_S))$-measurable.
For $s = (m, v) \in S$ we have

$$\mathcal{T}(s) = \left(\biguplus_{c \in Cmds} \{c\} \times \mathcal{T}(s, c) \right) \uplus \mathcal{T}_{\mathbb{R}_{\geq 0}}(s),$$

where

$$\mathcal{T}_{\mathbb{R}_{\geq 0}}(s) \stackrel{\text{def}}{=} \{(\mathtt{t}, [(m, v') \mapsto 1]) \mid \mathtt{t} \in \mathbb{R}_{\geq 0} \wedge v' \in Post_m(v, \mathtt{t})\}.$$

We will proceed with the proof as follows: firstly, we handle the timed transitions $\mathcal{T}_{\mathbb{R}_{\geq 0}}$. Then, we consider the transitions $\{c\} \times \mathcal{T}(\cdot, c)$ from guarded commands, at first with

guard *true*, and afterwards extend the result to arbitrary guards. Finally, we argue that $\mathcal{T}$, the union of these parts, is measurable.

We define $T\colon S \to (\mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \Sigma_S)$ where for $s = (m, v) \in S$ we have

$$T(s) = \{(\mathsf{t}, (m, v')) \mid (\mathsf{t}, v') \in T_{Post_m}(v)\},$$

with T_{Post} as in Definition 4.11. By the definition of T_{Post}, we have $T(s) \in (\mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \Sigma_S)$ and T is Σ_S-$\mathcal{H}(\mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \Sigma_S)$-measurable.

The function $\delta\colon S \to \Delta(S)$, defined as $\delta(s) \stackrel{\text{def}}{=} [s \mapsto 1]$, is measurable since $\delta^{-1}(\Delta^{\geq q}(Q)) = \{s \mid \delta(s)(Q) \geq q\}$ and this set is identical to Q if $q > 0$, and else it equals S. Because of this, $D\colon (\mathbb{R}_{\geq 0} \times S) \to (\mathbb{R}_{\geq 0} \times \Delta(S))$ with $D(\mathsf{t}, s) \stackrel{\text{def}}{=} (\mathsf{t}, \delta(s))$ is also measurable. We have $\mathcal{T}_{\mathbb{R}_{\geq 0}}(s) = \{D(\mathsf{t}, s) \mid (\mathsf{t}, s) \in T(s)\}$, that is $\mathcal{T}_{\mathbb{R}_{\geq 0}}(s) = D(T(s))$.

We have that $\Delta(S)$ standard Borel, because S is standard Borel [DTW12, Proposition 5.1], and thus so are $\mathbb{R}_{\geq 0} \times S$ and $\mathbb{R}_{\geq 0} \times \Delta(S)$. Since

- $\mathbb{R}_{\geq 0} \times S$ and $\mathbb{R}_{\geq 0} \times \Delta(S)$ are standard Borel,
- $D\colon (\mathbb{R}_{\geq 0} \times S) \to (\mathbb{R}_{\geq 0} \times \Delta(S))$ is measurable and injective, and
- $T(s) \in \mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \Sigma_S$,

we have that $D(T(s)) \in \mathcal{B}(\mathbb{R}_{\geq 0}) \otimes \Delta(\Sigma_S)$ [Kec95, Corollary 15.2].

Measurability follows using the measurability of T and D:

$$\begin{aligned}
&\mathcal{T}^{-1}_{\mathbb{R}_{\geq 0}}(H_\Theta) \\
&= \{s \mid \mathcal{T}_{\mathbb{R}_{\geq 0}}(s) \cap \Theta \neq \emptyset\} \\
&= \{s \mid D(T(s)) \cap \Theta \neq \emptyset\} \\
&= T^{-1}(H_{D^{-1}(\Theta)}).
\end{aligned}$$

For the transitions resulting from guarded commands, there are two components: measurable finite guarded command and measurable continuous guarded command. Let $c = (\mathit{true} \to p_1 : u_1 + \ldots + p_n : u_n)$ be a measurable finite guarded command. We have to show that $\mathcal{T}(s, c) = \{[s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \mid s'_1 \in u_1(s), \ldots, s'_n \in u_n(s)\}$ gives a measurable set for each s and that the function $\mathcal{T}(\cdot, c)\colon S \to \Delta(S)$ is Σ_S-$\mathcal{H}(\Delta(\Sigma))$-measurable. It can be shown that, for *disjoint* u_i, the semantics is the set of discrete probability measures $\Phi_{\leq n}$ having at most n points of mass, so that any of those discrete measures when applied to event $u_i(s)$ is equal to p_i:

$$\mathcal{T}(s, c) = \Phi_{\leq n} \cap \bigcap_{i=1}^{n} \Delta^{=p_i}(u_i(s)).$$

For $n \geq 1$, the set $\Phi_{\leq n}$ for a one-dimensional state space $\mathbb{R}$ is defined as:

$$\Phi_{\leq n} \stackrel{\text{def}}{=} \bigcap_{\substack{q_i < q'_i, q_i, q'_i \in \mathbb{Q}, \\ i \neq j \Rightarrow [q_i, q'_i] \cap [q_j, q'_j] = \emptyset}} \left(\Delta^{=0}\left(\left(\bigcup_{i=1}^{n} [q_i, q'_i] \right)^C \right) \cup \bigcup_{i=1}^{n} \Delta^{=0}([q_i, q'_i]) \right).$$

Therefore, the complete set $\mathcal{T}(s, c)$ is measurable, as it is a denumerable intersection of finite unions of generators for $\Delta(\Sigma)$. The generalisation to $\mathbb{R}^n$ is obtained using rectangles with rational endpoints in the n-dimensional space. Adding the mode component of the state space S does not raise any measurability issues, since it is a finite set having a finite number of subsets as measurable events.

If $u_i(s)$ sets are not disjoint, the expression is also measurable, but we have to take into account the intersections where two different update functions can choose the same point of mass, therefore the probabilities have to be added. Using $\Phi_{\leq n}$, we can define $\Phi_{=n} \stackrel{\text{def}}{=} \left(\bigcap_{i=1}^{n-1} \Phi^C_{\leq i}\right) \cap \Phi_{\leq n}$ the measurable set of discrete measures having exactly n points of mass. There are as many intersection combinations as partitions of the set $\{1 \dots n\}$, and we denote this family as $\mathit{SetPart}(n)$:

$$\mathcal{T}(s, c) = \bigcup_{P \in \mathit{SetPart}(n)} \Phi_{=|P|} \cap \bigcap_{i=1}^{|P|} \Delta^{=\sum_{j \in P_i} p_j} \left(\bigcap_{j \in P_i} u_j(s) \right).$$

Let $f \colon S^n \to \Delta(S)$ with $f(s_1, \dots, s_n) \stackrel{\text{def}}{=} [s_1 \mapsto p_1, \dots, s_n \mapsto p_n]$ be a function generating discrete measures, and let $u \colon S \to S^n$ with $u(s) \stackrel{\text{def}}{=} (u_1(s), \dots, u_n(s))$ be a function building the cross product of the nondeterministic update functions. Taking the set-wise extension of f we have $\mathcal{T}(s, c) = f(u(s))$, for $c = (\mathit{true} \to c = p_1 : u_1 + \ldots + p_n : u_n)$.

We now develop the backwards image of a $\Delta(\Sigma)$ generator:

$$\begin{aligned}
&\mathcal{T}(\cdot, c)^{-1}(H_\Theta) \\
&= \{s \mid f(u(s)) \cap \Theta \neq \emptyset\} \\
&= \{s \mid \exists (s_1, \dots, s_n) \in u(s).\ f(s_1, \dots, s_n) \in \Theta\} \\
&= u^{-1}(H_{f^{-1}(\Theta)}).
\end{aligned}$$

We have that $f \colon S \to \Delta(S)$ is measurable if and only if its *uncurried* version $f' \colon (S \times \Sigma_S) \to [0, 1]$ with $f'(s, A) \stackrel{\text{def}}{=} f(s)(A)$ for $A \in \Sigma_S$ is measurable in its first coordinate [D'A+09, Lemma 1]. Being the uncurried version of f, a linear combination of measurable functions, by standard results $f'(\cdot, Q)$ is measurable, therefore $f \colon S \to \Delta(S)$ is also measurable. Function u inherits measurability from its components [ADD00, Theorem 1.5.8], concluding that $\mathcal{T}(\cdot, c)^{-1}(H_\Theta)$ is a measurable set.

For a measurable continuous guarded command $c = (\mathit{true} \to M)$, the Markov kernel $M \colon S \to \Delta(S)$ is embedded as $\mathcal{T}(s, c) = \{M(s)\}$. Using a result of D'Argenio et al. [D'A+09, Proposition 5]—the embedding of labelled Markov processes (LMPs) into NLMPs—the result follows. We have that singletons $\{\mu\} \in \Delta(\Sigma)$ since the state space is generated by a denumerable π-system, namely the intervals with rational endpoints [D'A+09, Lemma 2].

Given a command c with a guard g other than *true*, we have

$$\mathcal{T}(s, c) = \begin{cases} \mathcal{T}(s, c_{\mathit{true}}) & \text{if } s \in g, \\ \emptyset & \text{else,} \end{cases}$$

where c_{true} is a variant of c where we replaced the guard by *true*, and this construction preserves both properties required for $\mathcal{T}(\cdot, c)$. From the measurability of $\mathcal{T}(\cdot, c)$, we can conclude the one of $\{c\} \times \mathcal{T}(\cdot, c)$.

Now we have to show that the complete $\mathcal{T}$ is measurable. As seen above, the transition relation $\mathcal{T}(s)$ is the finite union $\mathcal{T}(s) = \bigcup_{i=1}^{n} \mathcal{T}_i(s)$ of a number of components $\mathcal{T}_i \colon S \to \Sigma$ for $1 \leq i \leq n$ representing timed or command transitions, and each T_i is $\mathcal{B}(S)$-$\mathcal{H}(\Sigma)$-measurable, with $\Sigma \stackrel{\text{def}}{=} \Sigma_{Act} \otimes \Delta(\Sigma_S)$. Thus, we have

$$\begin{aligned}
&\mathcal{T}^{-1}(H_\Theta) \\
&= \{s \mid \mathcal{T}(s) \cap \Theta \neq \emptyset\} \\
&= \left\{ s \mid \left(\bigcup_{i=1}^{n} \mathcal{T}_i(s) \right) \cap \Theta \neq \emptyset \right\} \\
&= \bigcup_{i=1}^{n} \{s \mid \mathcal{T}_i(s) \cap \Theta \neq \emptyset\},
\end{aligned}$$

and because the $\mathcal{T}_i$ are measurable, each $\{s \mid \mathcal{T}_i(s) \cap \Theta \neq \emptyset\} \in \Sigma_S$ and thus their finite union is an element of Σ_S, which shows that $\mathcal{T}$ is measurable. □

Example 4.16. *Consider the SHA of Figure 4.1. Most parts of its behaviour could be described also by a PA using Definition 3.22. The exception here is the command* c_{WR} *which features a normal distribution. We are given* $k = 3$*, and the three dimensions* t*,* T *and* m*. Assume we are in state* $s \stackrel{\text{def}}{=} ((\text{Heat}, \text{Wait}), 1, 10, 0)$*, that is we are just about to move to mode* react *and have a temperature of* 10*. We have* $\mathcal{T}(s, c_{\text{WR}}) = \{\mu\}$ *where for* $a, b \in \mathbb{R}$ *with* $a \leq b$ *we have*

$$\mu(\{(\text{Heat}, \text{React}), 0, 10\} \times [a, b]) \stackrel{\text{def}}{=} \frac{1}{0.25\sqrt{2\pi}} \int_a^b \exp\left(-\frac{1}{2} \left(\frac{x - 10}{0.25} \right)^2 \right) \mathrm{d}x,$$

which uniquely determines the probabilities of other measurable sets. △

We are now able to determine minimal and maximal reachability probabilities of SHAs.

Definition 4.17. *Let* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}}, Cmds_{\text{cts}})$ *be an SHA with a mode* m_{Reach}*. We define*

$$\text{val}^{+}_{\mathcal{H}, m_{Reach}} \stackrel{\text{def}}{=} \text{val}^{+}_{[\![\mathcal{H}]\!], \{m_{Reach}\} \times \mathbb{R}^k} \text{ and } \text{val}^{-}_{\mathcal{H}, m_{Reach}} \stackrel{\text{def}}{=} \text{val}^{-, Cmds}_{[\![\mathcal{H}]\!], \{m_{Reach}\} \times \mathbb{R}^k}.$$

Because of the complex stochastic behaviour of the extended running example, we cannot give a manual estimation of the reachability probability here. We will however apply an automatic analysis in Subsection 4.4.1.

4.3 Abstractions

In this section, we will show how we can overapproximate SHAs by PHAs, which are then guaranteed to safely overapproximate the behaviour of the original model.

The following definition states how a single measurable continuous guarded command can be overapproximated by a measurable finite guarded command.

Definition 4.18. *Consider a measurable continuous guarded command* $c = (g \to M)$. *Fix* $p_i \geq 0$ *with* $1 \leq i \leq n$ *and* $\sum_{i=1}^{n} p_i = 1$. *Let* $\hat{u}_1, \ldots, \hat{u}_n : S \to \Sigma$ *and* $u_1, \ldots, u_n : S \to \Sigma$ *be functions where for all* $s \in S$ *we have that*

- $\hat{u}_i$ *and* u_i *are* Σ_S-$\mathcal{H}(\Sigma_S)$*-measurable for all* i *with* $1 \leq i \leq n$,
- $M(s)(\hat{u}_i(s)) = p_i$,
- $M(s)\left(\bigcup_{i=1}^{n} \hat{u}_i(s)\right) = 1$,
- *the sets* $\hat{u}_1(s), \ldots, \hat{u}_n(s)$ *are pairwise disjoint, and*
- $\hat{u}_i(s) \subseteq u_i(s)$ *for all* $1 \leq i \leq n$.

We call $\mathbf{f} = (\hat{u}_1, \ldots, \hat{u}_n, u_1, \ldots, u_n, p_1, \ldots, p_n)$ *a* command abstraction *and define the measurable finite guarded command*

$$\mathrm{abs}(c, \mathbf{f}) \stackrel{\text{def}}{=} (g \to p_1 : u_1 + \ldots + p_n : u_n).$$

By abstracting a command this way, we introduce (in most cases) uncountable additional nondeterminism, but reduce the complexity of the stochastic behaviour, because the abstractions of guarded commands no longer use continuous probability distributions. Overlapping sets $u_i(s), u_j(s)$, $i \neq j$ are allowed. This feature can be used, for instance, if the exact $\hat{u}_i(s)$, $\hat{u}_j(s)$ corresponding to probabilities p_i, p_j cannot be computed, thus to obtain a tolerance to guard against the imprecision of the p_i. The abstraction of a single command will be done symbolically in the high-level description of the probabilistic hybrid automaton (instead of the low-level model, as for instance in grid-based methods [PH06; HLS00]).

Example 4.19. *Consider the stochastic guarded command* c_{WR} *of the SHA of Figure 4.1. Let* $p_1 \stackrel{\text{def}}{=} \ldots \stackrel{\text{def}}{=} p_5 \stackrel{\text{def}}{=} 0.2$ *and consider* $a_1 \in T + [-0.22, -0.21]$, $a_2 \in T + [-0.07, -0.06]$, $a_3 \in T + [0.06, 0.07]$, $a_4 \in T + [0.21, 0.22]$. *We define for* $s = ((\mathrm{Heat}, \mathrm{Wait}), t, T, m)$

$$\begin{aligned}
\hat{u}_1(s) &\stackrel{\text{def}}{=} \{((\mathrm{Heat}, \mathrm{Wait}), t, T)\} \times (-\infty, a_1], \\
\hat{u}_2(s) &\stackrel{\text{def}}{=} \{((\mathrm{Heat}, \mathrm{Wait}), t, T)\} \times (a_1, a_2], \\
\hat{u}_3(s) &\stackrel{\text{def}}{=} \{((\mathrm{Heat}, \mathrm{Wait}), t, T)\} \times (a_2, a_3], \\
\hat{u}_4(s) &\stackrel{\text{def}}{=} \{((\mathrm{Heat}, \mathrm{Wait}), t, T)\} \times (a_3, a_4], \\
\hat{u}_5(s) &\stackrel{\text{def}}{=} \{((\mathrm{Heat}, \mathrm{Wait}), t, T)\} \times (a_4, \infty),
\end{aligned}$$

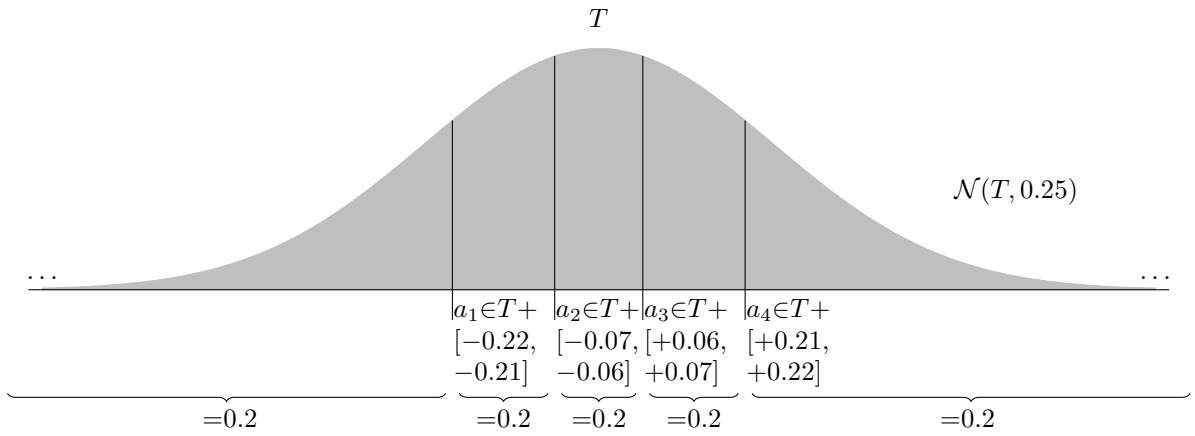


Figure 4.2: Abstraction of a normal distribution.

and accordingly for $s = ((\text{Cool}, \text{Wait}), t, T, m)$*. Assume that by a precomputation (which will later on be described in Subsection 4.3.1) we known that*

$$M(s)(\hat{u}_1(s)) = M(s)(\hat{u}_2(s)) = M(s)(\hat{u}_3(s)) = \ldots = p_i = 0.2,$$

as illustrated in Figure 4.2. Then the $\hat{u}_i$ *fulfil the requirements of Definition 4.18. Further define*

$$\begin{aligned}
u_1(s) &\stackrel{\text{def}}{=} \{((\text{Heat}, \text{Wait}), t, T)\} \times (-\infty, -0.21], \\
u_2(s) &\stackrel{\text{def}}{=} \{((\text{Heat}, \text{Wait}), t, T)\} \times [-0.22, -0.06], \\
u_3(s) &\stackrel{\text{def}}{=} \{((\text{Heat}, \text{Wait}), t, T)\} \times [-0.07, 0.07], \\
u_4(s) &\stackrel{\text{def}}{=} \{((\text{Heat}, \text{Wait}), t, T)\} \times [0.06, 0.22], \\
u_5(s) &\stackrel{\text{def}}{=} \{((\text{Heat}, \text{Wait}), t, T)\} \times [0.21, \infty).
\end{aligned}$$

Then we have $\hat{u}_i(s) \subseteq u_i(s)$ *for all* $i \in \{1, \ldots, 5\}$*. Because of this, the probabilistic guarded command*

$$g \to 0.2 : u_1 + 0.2 : u_2 + 0.2 : u_3 + 0.2 : u_4 + 0.2 : u_5$$

with $g = \{(m, t, T, m) \mid m = (\text{Heat}, \text{Wait}) \vee m = (\text{Cool}, \text{Wait})\}$ *is a command abstraction of* c_{WR}*.* △

To overapproximate an entire SHA, we just abstract all measurable continuous guarded commands.

Definition 4.20. *Let* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}}, Cmds_{\text{cts}})$ *be an SHA and consider a family of command abstractions* $\mathbf{F} = \langle \mathbf{f}_c \rangle_{c \in Cmds_{\text{cts}}}$*. Then we define the* PHA abstraction *of* $\mathcal{H}$ *as*

$$\text{abs}(\mathcal{H}, \mathbf{F}) \stackrel{\text{def}}{=} (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}} \uplus \{\text{abs}(c, \mathbf{f}_c) \mid c \in Cmds_{\text{cts}}\}).$$

Example 4.21. *If we overapproximate the only stochastic guarded command* c_{WR} *of the SHA in Figure 4.1 as in Example 4.19, we can easily check that the resulting model indeed fulfils all requirements of a PHA and of Definition 4.20.* △

The following theorem shows that we can use the abstractions of Definition 4.20 to prove probability bounds $\mathfrak{p}$ of SHAs by transforming them to PHAs and proving the bound there.

Theorem 4.22. *Given an SHA* $\mathcal{H}$ *with a mode* m_{Reach} *and a family of command abstractions* $\mathbf{F}$ *for its commands, we have*

$$\mathrm{val}^{+}_{\mathcal{H},m_{Reach}} \leq \mathrm{val}^{+}_{\mathrm{abs}(\mathcal{H},\mathbf{F}),m_{Reach}} \text{ and } \mathrm{val}^{-}_{\mathcal{H},m_{Reach}} \geq \mathrm{val}^{-}_{\mathrm{abs}(\mathcal{H},\mathbf{F}),m_{Reach}}.$$

Proof. Assume that

- $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m\in M}, Cmds_{\mathrm{fin}}, Cmds_{\mathrm{cts}})$,
- $\mathbf{F} = \langle \mathbf{f}_c \rangle_{c\in Cmds_{\mathrm{cts}}}$,
- $\mathrm{abs}(\mathcal{H},\mathbf{F}) = (M, k, \overline{m}, \langle Post_m \rangle_{m\in M}, Cmds)$,
- $[\![\mathcal{H}]\!] = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T})$,
- $[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!] = (S, \overline{s}, Act_{\mathrm{abs}}, \mathcal{T}_{\mathrm{abs}})$.

Let

$$f\colon (Path^{\mathrm{fin}}_{[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!]} \uplus Path^{\mathrm{inf}}_{[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!]}) \to (Path^{\mathrm{fin}}_{[\![\mathcal{H}]\!]} \uplus Path^{\mathrm{inf}}_{[\![\mathcal{H}]\!]})$$

be defined so that for $\beta_{\mathrm{abs}} \in Path^{\mathrm{inf}}_{[\![\mathcal{H},\mathbf{F}]\!]} \uplus Path^{\mathrm{inf}}_{[\![\mathcal{H},\mathbf{F}]\!]}$ we have

$$f(\beta_{\mathrm{abs}}) \stackrel{\mathrm{def}}{=} \beta,$$

with $|\beta_{\mathrm{abs}}| = |\beta|$ (we let $|\beta_{\mathrm{abs}}| \stackrel{\mathrm{def}}{=} |\beta| \stackrel{\mathrm{def}}{=} \infty$ for infinite paths), where for valid $i \geq 0$ we have

- $\beta[i] \stackrel{\mathrm{def}}{=} \beta_{\mathrm{abs}}[i]$,
- $\mathrm{trace}(\beta)[i] \stackrel{\mathrm{def}}{=} \mathrm{trace}(\beta_{\mathrm{abs}})[i]$ if $\mathrm{trace}(\beta_{\mathrm{abs}})[i] \in \mathbb{R}_{\geq 0} \uplus Cmds_{\mathrm{fin}}$,
- $\mathrm{trace}(\beta)[i] \stackrel{\mathrm{def}}{=} c$ if $\mathrm{trace}(\beta_{\mathrm{abs}})[i] = \mathrm{abs}(c, \mathbf{f}_c)$,
- $\mathrm{distrs}(\beta)[i] \stackrel{\mathrm{def}}{=} \mathrm{distrs}(\beta_{\mathrm{abs}})[i]$ if $\mathrm{trace}(\beta_{\mathrm{abs}})[i] \in \mathbb{R}_{\geq 0} \uplus Cmds_{\mathrm{fin}}$,
- $\mathrm{distrs}(\beta)[i] \stackrel{\mathrm{def}}{=} M(\beta[i])$ if $\mathrm{trace}(\beta_{\mathrm{abs}})[i] = \mathrm{abs}(c, \mathbf{f}_c)$ with $c = (g \to M) \in Cmds_{\mathrm{cts}}$.

This way, f maps paths of $\mathrm{abs}(\mathcal{H},\mathbf{F})$ to the corresponding paths of $\mathcal{H}$.

To show the theorem, we interpret $[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!]$ as a finitely probabilistic NLMP

$$[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!] = (S, \Sigma_S, \overline{s}, Act_{\mathrm{abs}}, \Sigma_{Act}, \mathcal{T}_{\mathrm{abs}}),$$

in which schedulers are allowed to take a randomised choice. We then proove that for an arbitrary scheduler $\sigma \in Sched_{[\![\mathcal{H}]\!]}$ we can construct a scheduler $\sigma_{\mathrm{abs}} \in Sched_{[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!]}$ where for all $A \in \Sigma_{Path^{\mathrm{inf}}_{[\![\mathcal{H}]\!]}}$ and $A_{\mathrm{abs}} \in \Sigma_{Path^{\mathrm{inf}}_{[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!]}}$ with $f(A_{\mathrm{abs}}) = A$ we have

$$Pr_{[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!]}(A_{\mathrm{abs}}) = Pr_{[\![\mathcal{H}]\!]}(A). \tag{4.6}$$

This means that, apart from the renaming of guarded commands and the distributions appearing on the paths, the stochastic behaviour under these two schedulers agrees. In turn, reachability probabilities agree, and fairness is also maintained.

Let $\beta \in Path^{\mathrm{fin}}_{[\![\mathrm{abs}(\mathcal{H},\mathbf{F})]\!]}$. For measurable sets $A \subseteq (\mathbb{R}_{\geq 0} \uplus Cmds_{\mathrm{fin}}) \times \Delta(S)$ we define

$$\sigma_{\mathrm{abs}}(\beta)(A) \stackrel{\mathrm{def}}{=} \sigma(f(\beta))(f(A)). \tag{4.7}$$

We consider

- $c = (g \to M) \in Cmds_{\mathrm{cts}}$ with
- $c_{\mathrm{abs}} = \mathrm{abs}(c, \mathbf{f}_c) = (g \to p_1 : u_1, \ldots, p_n : u_n)$ where
- $\mathbf{f}_c = (\hat{u}_1, \ldots, \hat{u}_n, u_1, \ldots, u_n, p_1, \ldots, p_n)$.

We define a family of measure spaces

$$\langle (S_i, \Sigma_i, \mu_i) \rangle_{i=1}^{n},$$

with

- $S_i \stackrel{\mathrm{def}}{=} \hat{u}_i(\mathrm{last}(\beta))$,
- $\Sigma_i \stackrel{\mathrm{def}}{=} \Sigma_S | \hat{u}_i(\mathrm{last}(\beta))$, and
- $\mu_i(A_i) \stackrel{\mathrm{def}}{=} \frac{M(\mathrm{last}(\beta))(A)}{p_i}$ for $A_i \in \Sigma_i$.

Then consider the product measure space

$$(S_{\mathrm{prod}}, \Sigma_{\mathrm{prod}}, \mu_{\mathrm{prod}}),$$

with

- $S_{\mathrm{prod}} \stackrel{\mathrm{def}}{=} \bigtimes_{i=0}^{n} S_i$,
- $\Sigma_{\mathrm{prod}} \stackrel{\mathrm{def}}{=} \bigotimes_{i=0}^{n} \Sigma_i$,
- $\mu_{\mathrm{prod}}(\bigtimes_{i=1}^{n} A_i) = \mu_i(A_i)$ where $A_i \in \Sigma_i$ for $i \in \{1, \ldots, n\}$,

where μ_{prod} is uniquely extended to other sets $A \in \Sigma_{\mathrm{prod}}$. Define

$$U_c \stackrel{\mathrm{def}}{=} \{[s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \mid s'_1 \in \hat{u}_1(\mathrm{last}(\beta)), \ldots, s'_n \in \hat{u}_n(\mathrm{last}(\beta))\},$$

and let $g \colon U_c \to S_{\mathrm{prod}}$ be the bijection defined as

$$g([s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n]) \stackrel{\mathrm{def}}{=} (s'_1, \ldots, s'_n).$$

Using g, we can define a measure μ_c on U_c where for measurable $B \subseteq U_c$ we have

$$\mu_c(B) \stackrel{\mathrm{def}}{=} \mu_{\mathrm{prod}}(g(B)). \tag{4.8}$$

Then for $A = \{c_{\mathrm{abs}}\} \times A_{\mathrm{distrs}} \subseteq \{c_{\mathrm{abs}}\} \times \Delta(S)$ we define

$$\sigma_{\mathrm{abs}}(\beta)(A) \stackrel{\mathrm{def}}{=} \mu_c(A_{\mathrm{distrs}} \cap U_c)\sigma(\beta)(\{c\} \times \Delta(S)). \tag{4.9}$$

For the combined transition of Definition 4.6, with

- $c = (g_c \to M_c)$ for all $c \in Cmds_{\text{cts}}$,
- $\text{abs}(c, \mathbf{f}_c) = (g_c \to p_{c,1} : u_{c,1} + \ldots + p_{c,n_c} : u_{c,n_c})$ for all $c \in Cmds_{\text{cts}}$ with $\mathbf{f}_c = (\hat{u}_{c,1}, \ldots, \hat{u}_{c,n}, u_{c,1}, \ldots, u_{c,n}, p_{c,1}, \ldots, p_{c,n})$,
- $\beta \in Path^{\text{fin}}_{[\![\mathcal{H}]\!]}$,
- $\beta_{\text{abs}} \in Path^{\text{fin}}_{[\![\text{abs}(\mathcal{H},\mathbf{F})]\!]}$ with $\beta_{\text{abs}} = f(\beta)$,
- $A \in \Sigma_{Act}$,
- $A_{\text{abs}} \in \Sigma_{Act_{\text{abs}}}$ with $A_{\text{abs}} = (A \cap (\mathbb{R}_{\geq 0} \uplus Cmds_{\text{fin}})) \uplus \{\text{abs}(c, \mathbf{f}_c) \mid c \in A\}$,
- $A_{\text{rest}} = A_{\text{abs}} \cap (\mathbb{R}_{\geq 0} \uplus Cmds_{\text{fin}})$ and $A_c = A_{\text{abs}} \cap \text{abs}(c, \mathbf{f}_c)$,
- $Q \in \Sigma_S$,
- $Q_{c,i} = Q \cap \hat{u}_{c,i}(\text{last}(\beta))$,
- $\xi \in \Delta(\Sigma_S)$,
- $\xi_{\text{abs}} \in \Delta(\Sigma_S)$ where $\xi_{\text{abs}} = \xi \cup \bigcup_{c \in Cmds_{\text{cts}}} \begin{cases} U_c & \text{if } M_c(\beta) \in \xi, \\ \emptyset & \text{else,} \end{cases}$,
- $\xi_{c,i} \stackrel{\text{def}}{=} \{\mu \in \xi_{\text{abs}} \mid \mu(Q_{c,i}) = p_{c,i}\}$

we have

$$
\begin{aligned}
&Tr_{[\![\text{abs}(\mathcal{H},\mathbf{F})]\!],\sigma_{\text{abs}},\beta_{\text{abs}}}(A_{\text{abs}}, \xi_{\text{abs}}, Q) \\
&= \int_{A_{\text{abs}} \times \xi_{\text{abs}}} \mu(Q)\, \sigma_{\text{abs}}(\beta_{\text{abs}})(\mathrm{d}a, \mathrm{d}\mu) \\
&= \int_{A_{\text{rest}} \times \xi_{\text{abs}}} \mu(Q)\, \sigma_{\text{abs}}(\beta_{\text{abs}})(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \int_{A_c \times \xi_{\text{abs}}} \mu(Q)\, \sigma_{\text{abs}}(\beta_{\text{abs}})(\mathrm{d}a, \mathrm{d}\mu) \\
&\stackrel{\text{Eqn. 4.7}}{=} \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \int_{A_c \times \xi_{\text{abs}}} \mu(Q)\, \sigma_{\text{abs}}(\beta_{\text{abs}})(\mathrm{d}a, \mathrm{d}\mu) \\
&\stackrel{\text{Eqn. 4.9}}{=} \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \int_{A_c \times \xi_{\text{abs}}} \mu(Q)\, \mu_c(\mathrm{d}\mu)\sigma(\beta)(\{c\} \times \Delta(S)) \\
&= \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \sum_{i=1}^{n_c} \int_{A_c \times \xi_{\text{abs}}} \mu(Q_{c,i})\, \mu_c(\mathrm{d}\mu)\sigma(\beta)(\{c\} \times \Delta(S)) \\
&\stackrel{\text{Eqn. 4.8}}{=} \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \sum_{i=1}^{n_c} \int_{A_c \times \xi_{c,i}} \mu(Q_{c,i})\, \mu_c(\mathrm{d}\mu)\sigma(\beta)(\{c\} \times \Delta(S)) \\
&= \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \sum_{i=1}^{n_c} p_{c,i} \frac{M_c(Q_{c,i})}{p_{c,i}} \sigma(\beta)(\{c\} \times \Delta(S)) \\
&= \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \sum_{i=1}^{n_c} M_c(Q_{c,i}) \sigma(\beta)(\{c\} \times \Delta(S)) \\
&= \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} M_c(Q) \sigma(\beta)(\{c\} \times \Delta(S))
\end{aligned}
$$

$$= \int_{A_{\text{rest}} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu) + \sum_{c \in Cmds_{\text{cts}}} \int_{\{c\} \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu)$$
$$= \int_{A \times \xi} \mu(Q)\, \sigma(\beta)(\mathrm{d}a, \mathrm{d}\mu)$$
$$= Tr_{[\![\mathcal{H}]\!],\sigma,\beta}(A, \xi, Q).$$

From this, by the definition of path measures in Definition 4.6 the validity of Equation 4.6 follows, and thus reachability probabilities and fairness agree.

By Lemma 4.10, we can transform the scheduler constructed this way to a finitely probabilistic scheduler σ_l (σ_u) which is guaranteed to obtain a lower (higher) probability than σ_{abs} and maintains fairness. This scheduler is also a scheduler in the interpretation of $[\![\text{abs}(\mathcal{H}, \mathbf{F})]\!]$ as a PA. Interpreting this model as a PA does not change the reachability probabilities, from which we can conclude the lemma. □

4.3.1 Computing Abstractions

For the abstraction to be applicable in practice, it is crucial to compute the family of abstraction functions effectively. As there exist quite diverse forms of random variables, we cannot give an algorithm to handle all cases. Instead, we sketch how to obtain over-approximation functions for certain classes of random variables.

Firstly, consider a one-dimensional probability measure $\mu\colon \mathcal{B}(\mathbb{R}) \to [0,1]$ given by a density function $f(x)$, for instance the normal distribution. Using numerical methods, we then compute bounds for a_i with $\mu((-\infty, a_1]) = p_1, \mu((a_1, a_2]) = p_2, \ldots, \mu((a_{n-1}, \infty)) = p_n$, for some $p_1, \ldots, p_n$. In Example 4.19, we assumed that we are already given a valid abstraction of the normal distribution $\mathcal{N}(T, 0.25)$. We first consider the case of a constant $T \stackrel{\text{def}}{=} 0$, that is $\mathcal{N}(0, 0.25)$. An abstraction of this distribution could be obtained as follows. Fixing $n = 5$ and $p_i = 0.2$, we use numerical methods to find out that the a_i are in the intervals $a_1 \in [-0.22, -0.21], a_2 \in [-0.06, -0.06], \ldots$. We can now transform the random variable to handle the general case $\mathcal{N}(T, 0.25)$, in which the distribution depends on the state variable T. We use the fact that $\mathcal{N}(x, y) = \frac{\mathcal{N}(0,1)-x}{y}$. Thus, we can transform corresponding interval endpoints b_i to $b_i(x, y) = b_i(0, 1) \cdot y + x$. When setting $x = T, y = 0.25$, we obtain the same intervals as given in Example 4.19.

If the cumulative distribution function $F(x)$ of a random variable is known and we can compute a closed-form of F^{-1}, we can use a method similar to the inverse transform method. Consider the exponential distribution with state-dependent λ. We have that if $F_\lambda(a_i) = p_i$ then $a_i = -\ln(1 - p_i)\frac{1}{\lambda}$. We can then obtain adjoint intervals which have a certain probability by precomputing $[b_i, b_i'] \ni -\ln(1 - p_i)$ and thus we specify command branches $p_i : \frac{b_{i-1}}{\lambda} \leq x \leq \frac{b_i'}{\lambda}$.

For probability measures in two variables, we consider $f(\cdot, (-\infty, \infty))$ at first, and then split each $f([a_i, a_{i+1}], \cdot)$ again. This technique extends to any finite number k of variables. If we split each of them into a number of n parts, the support of the abstracting distribution has a size of n^k. Thus, the worst-case complexity of this method is rather

bad, because the number of parts increases quickly with an increasing number of dimensions. As an alternative, one could divide the space by spheres of similar distance. This way, one would only need n parts, but would loose more information. However, the case that only one or few variables change appears to be the practically relevant case for us. It occurs in settings where the environment can be observed only with limited accuracy, as the ones discussed in Section 4.4.

After we have transformed each stochastic guarded command into a probabilistic guarded command, we can use the methods from the previous chapters to obtain an overapproximation of the new PHA.

Corollary 4.23. *Consider a PHA $\mathcal{H}$, let $\mathcal{H}_{\text{abs}} \stackrel{\text{def}}{=} \text{abs}(\mathcal{H}, \mathbf{F})$ for an according family of abstraction functions $\mathbf{F}$. Further, consider an abstraction $\mathcal{M}_{\text{ind}}$ of $\text{ind}(\mathcal{H}_{\text{abs}})$, $\mathcal{M} \stackrel{\text{def}}{=} \text{abs}(\mathcal{H}_{\text{abs}}, \mathcal{M}_{\text{ind}})$ and a mode m_{Reach} of $\mathcal{H}$. Let Reach be defined as in Theorem 4.22 and let Cmds be the set of probabilistic guarded commands of $\mathcal{H}_{\text{abs}}$. Then*

$$\text{val}^{+}_{\mathcal{H},m_{Reach}} \leq \text{val}^{+}_{\mathcal{M},Reach} \text{ and } \text{val}^{-}_{\mathcal{H},m_{Reach}} \geq \text{val}^{-,Cmds}_{\mathcal{M},Reach}.$$

This follows directly from Theorem 4.22 and Corollary 3.38.

4.4 Case Studies

We use the extended analysis method described above to obtain upper bounds for reachability properties in several case studies. In each case, we firstly abstract stochastic guarded commands to probabilistic guarded commands, which ProHVer can handle. Thus, we obtain a PHA which overapproximates our SHA, as described. The transformation to PHAs is so far done manually. The methods described in Subsection 4.3.1 could however be automated further in such a way that one for instance only had to specify the intervals into which a one-dimensional random variable is to be divided instead of applying the transformation by hand. Again, we use PHAVer and our tool ProHVer to obtain overapproximations of the reachability probabilities in the PHA. Experiments were run on an Intel(R) Core(TM)2 Duo CPU with 2.67 GHz and 4 GB RAM as before.

4.4.1 Thermostat

We reconsider the extended thermostat of Figure 4.1 and ask for the maximal probability to reach Error within a given time bound $\mathfrak{T}$. Like the thermostat from Chapter 3, this model features dynamics which can be described by affine differential equations, but which is not linear. We choose a fixed splitting of the normal distribution, but vary the refine constraints used by PHAVer (cf. Section 2.5) to analyse the model.

In Table 4.1, we give probability bounds and performance statistics. We used a refine constraint on the variable T which models the temperature. To abstract the normal

$\mathfrak{T}$	constraint length 2			constraint length 1			constraint length 0.5		
	build (s)	states	prob.	build (s)	states	prob.	build (s)	states	prob.
2	1	16	0.000	0	21	0.000	1	31	0.000
4	1	269	1.000	2	316	0.285	3	546	0.285
6	6	1518	1.000	10	2233	0.360	17	3797	0.360
8	21	4655	1.000	38	8261	1.000	87	16051	0.488
10	58	10442	1.000	131	20578	1.000	529	44233	0.591

Table 4.1: Thermostat results.

distribution $\mathcal{N}(T, 0.25)$, we used

$$T + \{[-0.25, 0.25], (-\infty, -0.25], [0.25, \infty)\}.$$

For all instances, there is a constraint length small enough to obtain a probability bound that is the best possible using the given abstraction of the normal distribution. Smaller constraints were of no use in this case. The drastic discontinuities in probability bounds obtained are a consequence of how the abstraction by PHAVER works: it splits abstract states into smaller abstract states when exploring the model parts reachable by timed transitions. This is necessary to obtain reasonable results for nonlinear hybrid automata. Using smaller constraints usually results in a better abstraction, but this is not guaranteed. For a more thorough discussion of the abstraction used by PHAVER, see Section 2.5.

4.4.2 Water Level Control

We consider a model of a water level control system, extended from Subsection 3.5.3. In particular, we use this case study to demonstrate the influence which different abstractions of the same continuous stochastic command have. A water tank is filled by a constant stream of water, and is connected to a pump which is used to avoid overflow of the tank. A control system operates the pump in order to keep the water level within predefined bounds. The controller is connected to a sensor measuring the level of water in the tank. A sketch of the model is given in Figure 4.3. The mode Tank models the tank and the pump, and W is the water level. Initially, the tank contains a given amount of water. Whenever the pump is turned off in mode Off, the tank fills with a constant rate due to the inflow. Conversely, more water is pumped out than flows in when the pump is on.

The controller is modelled by the part Controller. In mode Wait, the controller waits for a certain amount of time. Upon the transition to React, it measures the water level. To model the uncertainties in measurement, we set the variable m to a normal distribution with expected value W (the actual water level) and standard deviation 1. According to the measurement obtained, the controller switches the pump off or on. We remark that this measurement is not necessarily realistic, because no filtering is involved here to guard against measurements which are obviously far off from the real value.

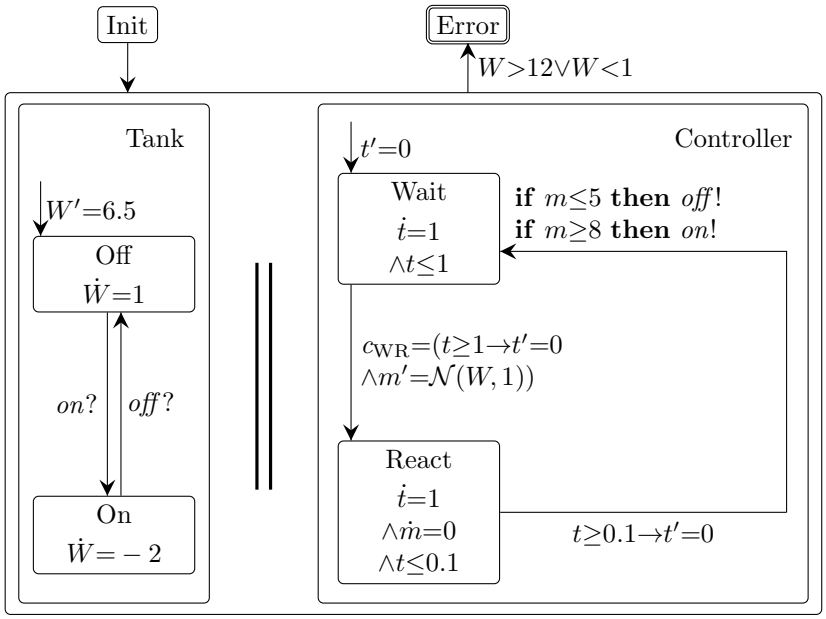


Figure 4.3: Sketch of water level control case study.

$\mathfrak{T}$	Abstraction A			Abstraction B			Abstraction C			Abstraction D		
	b.(s)	states	prob.	b.(s)	states	prob.	b.(s)	states	prob.	b.(s)	states	prob.
20	3	999	0.199	3	1306	0.098	3	1306	0.136	5	1920	0.047
30	7	2232	0.283	8	2935	0.143	8	2935	0.187	16	4341	0.069
40	16	3951	0.358	20	5212	0.186	19	5212	0.255	48	7734	0.092
50	36	6156	0.425	44	8137	0.226	45	8137	0.302	111	12099	0.113
60	71	8847	0.485	89	11710	0.265	89	11710	0.358	224	17436	0.135

Table 4.2: Water level control results.

We are interested in the probability that within a given time bound $\mathfrak{T}$, the water level leaves the legal interval. In Table 4.2, we give upper bounds for this probability for different time bounds as well as the number of states in the abstraction computed by PHAVER and the time needed for the analysis. We round probabilities to four decimal places. For the stochastic guarded command simulating the measurement, we consider different abstractions by probabilistic guarded commands of different precision:

$$
\begin{aligned}
A &= w + \{[-2,2], (-\infty, 1.9] \cup [1.9, \infty)\},\\
B &= w + \{[-2,2], (-\infty, 1.9], [1.9, \infty)\},\\
C &= w + \{[-2.7,2.7], (-\infty, 1.2), [1.2, \infty)\},\\
D &= w + \{[-1.5,1.5], [-1.5,-2], [1.5,2], (-\infty, 1.9), [1.9, \infty)\}.
\end{aligned}
$$

For better comparability of the different abstractions, we visualise them in Figure 4.4. When we refine the abstraction A to the more precise abstraction B, the probability bound decreases. We remark that A consists of two different sets, whereas B consists of three, as we have split $(-\infty, 1.9] \cup [1.9, \infty)$ into $(-\infty, 1.9]$ and $[1.9, \infty)$. If we introduce

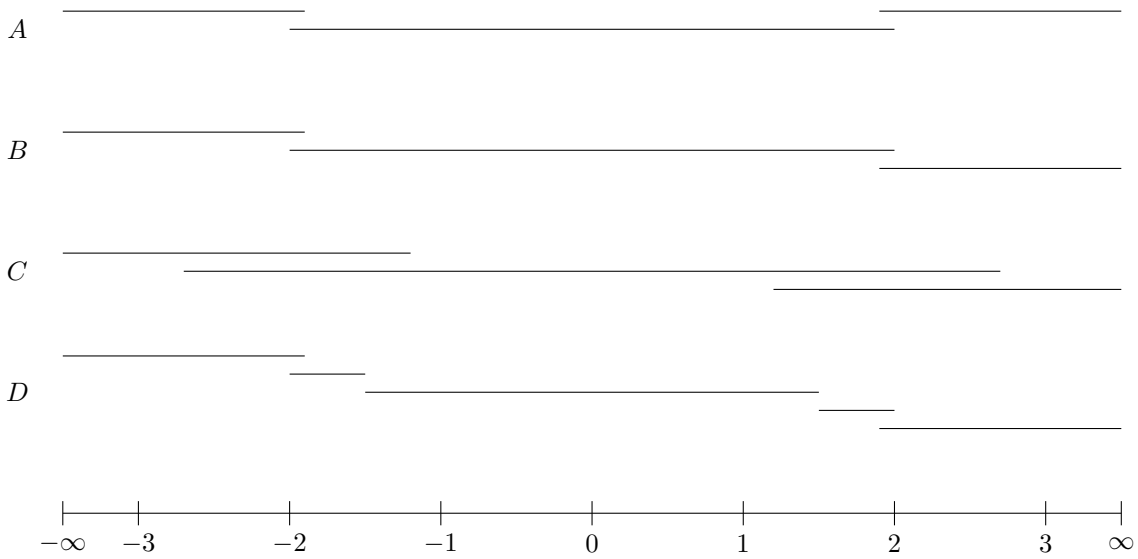


Figure 4.4: Visualisation of abstractions for water level control.

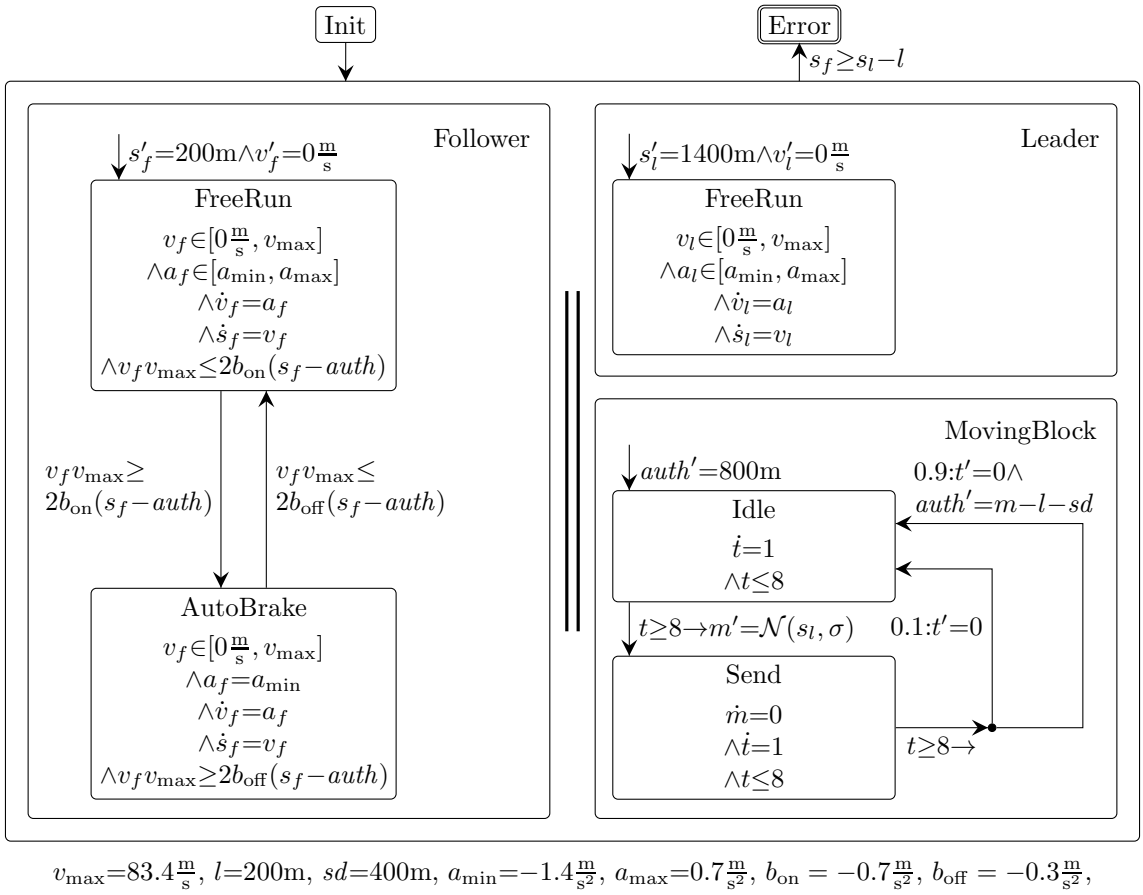
additional nondeterminism as in abstraction C, probabilities increase again. If we refine B again into D, we obtain even lower probability bounds. The price to be paid for increasing precision, however, is in the number of abstract states computed by PHAVER as well as a corresponding increase in the time needed to compute the abstraction.

Manual analysis shows that, in this case study, the over-approximation of the probabilities only results from the abstraction of the stochastic guarded command into a probabilistic guarded command and is not increased further by the state-space abstraction of PHAVER.

4.4.3 Moving-block Train Control

As a more complex example of a hybrid system implementing a safety-critical control policy, we present a model of headway control in the railway domain (Figure 4.5). A more extensive description of the setting plus a closely related case study containing a sampling-related bug not present in the current model appeared in a previous publication [Her+08]. In contrast to fully automated transport, which is in general simpler to analyse (as the system is completely under control of the embedded systems) our sample system implements safe-guarding technology that leaves trains under full human control provided safety is not at risk. It is thus an open system, giving rise to the aforementioned analysis problems.

Our model implements safe interlocking of railway track segments by means of a "moving block" principle of operation. While conventional interlocking schemes in the railway domain lock a number of static track segments in full, the moving block principle enhances traffic density by reserving a "moving block" ahead of the train which moves smoothly with the train. This block is large enough to guarantee safety even in cases



$v_{\max}$=83.4$\frac{\text{m}}{\text{s}}$, l=200m, sd=400m, $a_{\min}$=$-1.4\frac{\text{m}}{\text{s}^2}$, $a_{\max}$=$0.7\frac{\text{m}}{\text{s}^2}$, $b_{\text{on}} = -0.7\frac{\text{m}}{\text{s}^2}$, $b_{\text{off}} = -0.3\frac{\text{m}}{\text{s}^2}$,

Figure 4.5: Moving-block train distance control with perturbed measurement of leader train position (using normal distribution $\mathcal{N}(s_l, \sigma)$ centred around the actual value, with standard deviation σ) and unreliable communication of resultant movement authorities (failure probability 0.1). Error represents collision of the two trains.

requiring emergency stops, i.e. has a dynamically changing block-length depending on current speed and braking capabilities. There are two variants of this principle, namely train separation in relative braking distance, where the spacing of two successive trains depends on the current speeds of both trains, and train separation in absolute braking distance, where the distance between two trains equals the braking distance of the second train plus an additional safety distance (here given as $sd = 400\text{m}$). We study the second variant, as employed in the European Train Control System (ETCS) Level 3.

Our simplified model consists of a leader train, a follower train, and a moving-block control regularly measuring the leader train position and communicating a *movement authority* to the follower. The leader train is freely controlled by its operator within the physical limits of the train, while the follower train may be forced to controlled braking if coming close to the leader. The control principle is as follows:

1. 8 seconds after communicating the last movement authority, the moving-block con-

$\mathfrak{T}$	Abstraction A					Abstraction B				
	b.(s)	states	probability ($\sigma = 10, 15, 20$)			b.(s)	states	probability ($\sigma = 10, 15, 20$)		
60	59	571	7.110E-19	6.216E-09	2.141E-05	63	571	1.806E-06	2.700E-03	3.847E-02
80	199	1440	1.016E-18	8.879E-09	3.058E-05	199	1440	2.580E-06	3.855E-03	5.450E-02
100	524	2398	1.219E-18	1.066E-08	3.669E-05	493	2392	3.096E-06	4.624E-03	6.504E-02
120	1455	4536	1.524E-18	1.332E-08	4.587E-05	1347	4524	3.870E-06	5.777E-03	8.063E-02
140	3024	6568	1.727E-18	1.509E-08	5.198E-05	2885	6550	4.386E-06	6.544E-03	9.088E-02
160	6783	10701	2.031E-18	1.776E-08	6.116E-05	6678	10665	5.160E-06	7.695E-03	1.060E-01

Table 4.3: Train control results.

trol takes a fresh measurement m of the leader train position s_l. This measurement may be noisy.

2. Afterwards, a fresh movement authority derived from this measurement is sent to the follower. The movement authority is the measured position m minus the length l of the leader train and further reduced by the safety distance sd. Due to an unreliable communication medium, this value may reach the follower (in which case its movement authority $auth$ is updated to $m - l - sd$) or not. In the latter case, which occurs with probability 0.1, the follower's movement authority stays as is.
3. Based on the movement authority, the follower continuously checks the deceleration required to stop exactly at the movement authority. Due to PHAVER being confined to linear arithmetic, this deceleration is conservatively approximated as $a_{\text{req}} = \frac{v \cdot v_{\max}}{2(s-auth)}$, where v is the actual speed, $v_{\max}$ the (constant) top speed, and s the current position of the follower train, rather than the physically more adequate, yet nonlinear, $a_{\text{req}} = \frac{v^2}{2(s-auth)}$ of the original model [Her+08].
4. The follower applies automatic braking whenever the value of a_{req} falls below a certain threshold b_{on}. In this case, the follower's brake controller applies maximal deceleration $a_{\min}$, leading to a stop before the movement authority as $a_{\min} < b_{\text{on}}$. Automatic braking ends as soon as the necessary deceleration a_{req} exceeds a switch-off threshold $b_{\text{off}} > b_{\text{on}}$. The thresholds b_{on} and b_{off} are separate to prevent the automatic braking system from repeatedly engaging and disengaging in intervals of approximately 8 seconds when the leading train is moving.

We consider the probability to reach the mode Error in which the follower train has collided with the leader train. In Table 4.3, we give probability bounds and performance results. For abstraction A we use a division of the normal distribution into $s_l + \{(-\infty, 91], [89, \infty)\}$. For B, we split the distribution into $s_l + \{(-\infty, 51], [49, \infty)\}$. We give probabilities for different values σ of the standard deviation of the measurement. We modelled the measurement error using a normal distribution with expected value s_l, i.e. the current position of the leader train. In the table, we considered different standard deviations of the measurement. The abstraction used for each of them can be obtained using structurally equal Markov decision processes, only with different probabilities. Thus, we only needed to compute the abstraction once for all deviations,

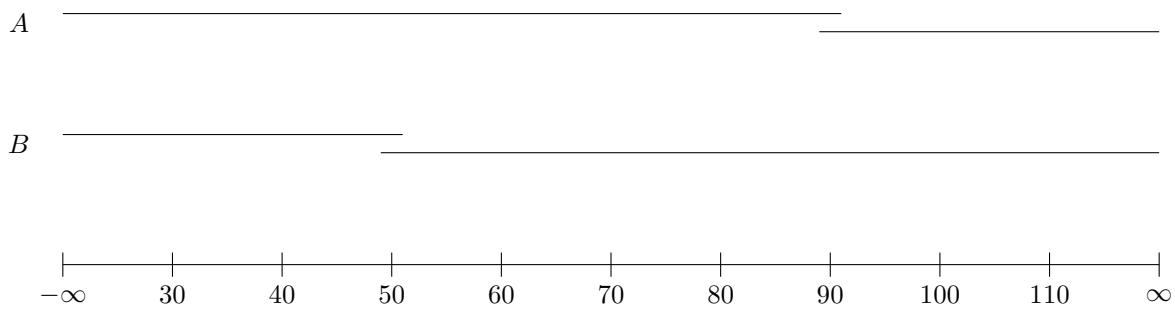


Figure 4.6: Visualisation of abstractions for moving-block train control.

and just had to change the transition probabilities before obtaining probability bounds from the abstraction. It was sufficient to split the normal distribution into two parts. Depending on where we set the split-point, we obtained probability bounds of different quality. Although this hybrid automaton is not linear, which implies that PHAVer needs to over-approximate the set of reachable states, we are still able to obtain useful probability bounds when using an adequate abstraction, without refine constraints.

4.5 Related Work

There are a number of existing notions of HAs which also feature continuous distributions. Most of them do not contain nondeterminism. All of them are built on specialised solvers and do not (in the end) rely on existing solvers for nonprobabilistic HAs.

Davis [Dav84] introduced piecewise-deterministic Markov processes (PDMPs). The continuous-time transitions of this model are described by ordinary differential equations. State changes are triggered, on the one hand, spontaneously according to a generalisation of exponential distributions, similarly to continuous-time Markov chains. On the other hand, jumps can be forced in case a differential equation is about to violate a certain invariant. Other notions of stochastic HAs comprise stochastic differential equations [Arn74; BL06a; Buj04; BLB05; BL06b; BLL08]. They can incorporate random perturbations, such as Brownian motion, into the continuous dynamics. As PDMPs, they usually do not contain nondeterminism and their solution methods are rather different from the ones of nonstochastic HAs. There are some versions of these models with nondeterministic decisions [AG97], but the solution methods for these models are rather different than those of the stochastic HAs models discussed before.

Solution methods for stochastic HAs without nondeterminism but with stochastic differential equations often rely on *grid-based methods*. These methods subsume concrete states of a stochastic HA and in turn approximate the original model by finite discrete-time Markov chains [HLS00; PH06]. It is proven that the results of analyses of these Markov chains converge to the result in the original hybrid models, but there are no exact error bounds. Related solution methods assume that the model has already been discretised in time, but still has a continuous state space [Aba+08; Aba+10; AKM11;

TA12]. Again, for the step from the continuous-time model to the discrete-time model no error bounds exist, but for the step from a discrete-time, continuous-state to a discrete-time, discrete-state model error bounds do exist.

The grid-based approach is similar to our approach insofar as it subsumes concrete states to abstract ones. Differently from ours, this abstract model is, in most cases, again a purely stochastic model without nondeterminism, from which an error has to be added or subtracted to obtain over- or underapproximations of the actual value. As we abstract to nondeterministic models, the maximal reachability probabilities in the abstract model are guaranteed to be no lower than the ones of the concrete model, and similarly for the minimal ones. Thus, in contrast to the methods discussed here, we do not explicitly compute the error bounds.

There are also some works which do not rely on a grid. Althoff et al. consider a method [ASB08] in which they use zonotopes to compute overapproximations of the states in which a system described by a stochastic differential equation will remain with a sufficient probability. Other methods use Lyapunov functions [JP09] or barrier certificates [PJP07] to decide safety properties.

The NLMPs we use as our semantical model have been developed by Wolovick et al. [D'A+09; DTW12; Wol12].

4.6 Conclusion

We have extended our basic PHA model of Chapter 3 by continuous distributions, thus allowing to faithfully model a class of relevant reals-world systems. In particular, this approach is a natural model of perturbed measurement processes, as was demonstrated by a number of case studies. Because PAs do not allow for the specification of continuous distributions, we had to change our underlying semantical model to NLMPs. The continuous distributions raised questions of measurability, which did not occur in PAs. Because of this, the formulation of the NLMP model and its properties was more complicated than previously. Also, when describing the semantics of SHAs in terms of NLMPs, it was no longer obvious that this semantics indeed fulfils all the requirement of an NLMP, as it was for PAs. To guarantee that it is, we put measurability restrictions on the system components, and showed that their measurability implies the measurability of the complete system. We have described how we can overapproximate SHAs by PHAs, and have proven the correctness of this abstraction. As we already have developed an abstraction framework for this restricted class of models, we concluded that we are now able to decide properties of general SHAs, and demonstrated our practical ability to do so on a number of case studies, one of which was a complex model of a real-world railroad coordination system.

The overall abstraction scheme is given in Figure 4.7. It thus extends Figure 3.8. For clarity we have left out the steps concerning the time-restricted semantics (cf. Definition 2.16 and Definition 3.22) of PHAs, which is used as an intermediate step of the abstraction of PHAs.

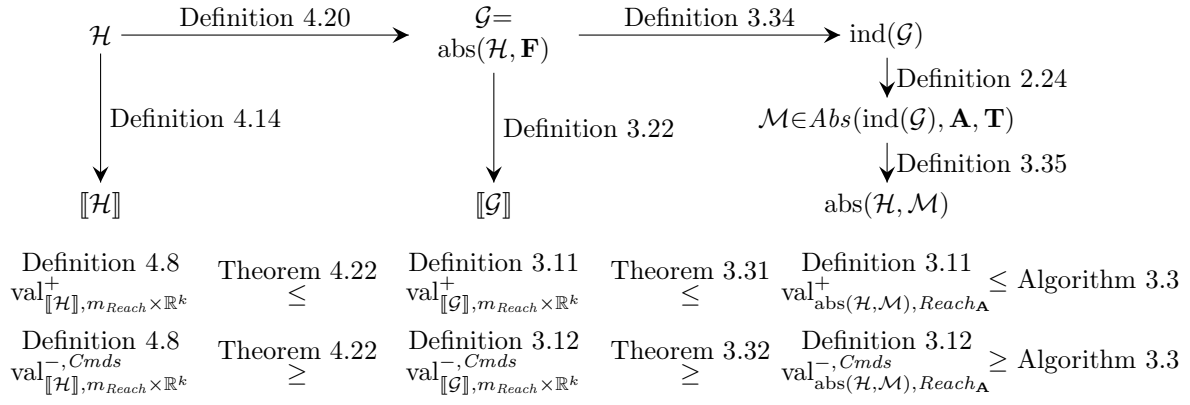


Figure 4.7: Abstraction scheme for SHAs.

5
Partial Control

In this chapter, we focus on probabilistic hybrid automata which are under our partial control. We divide the nondeterminism into two parts, one where we are in control and another which is assumed to be controlled by a possibly malevolent environment. In addition, we still have complex dynamics as well as probabilistic choices. In the setting we analyse, we assume that we are in control of the guarded commands which we can execute to influence the environment, and can decide to wait for a specific amount of time. The environment can then choose the exact effect of the guarded command, or of the timed transition. We chose this kind of division between the two types of nondeterminism, because it is a very natural one: mode-switching controllers of hybrid systems are often realised as digital circuits with a finite number of possible outputs, which corresponds to the finite number of commands between which it can choose. On the other hand, a controller can silently wait for a given time, during which continuous values of the environment (temperature, level of water in a tank, etc.) are changed. We then show how we can use results in the abstraction of such systems to synthesise controllers which can steer a system thus to guarantee bounds on reachability probabilities.

In Section 5.1 we show how probabilistic automata can be interpreted as probabilistic games. We also define the optimal values of reachability properties which a controller can enforce against a most malevolent environment. In Section 5.2 we discuss an interpretation of probabilistic hybrid automata as partially controllable systems and give their semantics in terms of a probabilistic game. Then, in Section 5.3 we show how we can obtain suitable abstractions, which are finite games. Section 5.4 will discuss how properties can be decided once the abstraction has been computed and Section 5.5 will discuss how we can then use these results to synthesise controllers for the concrete system. In Section 5.6 we apply the method on a case study. Afterwards, Section 5.7 discusses related work. Section 5.8 concludes the chapter.

5.1 Game Interpretation of Probabilistic Automata

As in Chapter 3, our semantical models are PAs. However, we interpret the transition matrix in a different way. Instead of assuming that the entire nondeterminism is resolved

by a single scheduler, we assume that one instance chooses between the different possible actions, and another instance is then responsible for choosing the exact effect of an action. Both choices combined in turn yield a scheduler in the previous sense.

Definition 5.1. *Consider a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$*. A* player-con strategy *is a function* $\sigma_{\text{con}} \colon Path^{\text{fin}}_{\mathcal{M}} \to Distr(Act)$*. We require, for each* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ *and* $a \in Act$*, that if* $\sigma_{\text{con}}(\beta)(a) > 0$ *then we have* $\mathcal{T}(\text{last}(\beta), a) \neq \emptyset$*. A* player-env strategy *is a function* $\sigma_{\text{env}} \colon (Path^{\text{fin}}_{\mathcal{M}} \times Act) \to Distr(Distr(S))$*. We require that if* $\mathcal{T}(\text{last}(\beta), a) \neq \emptyset$ *and* $\sigma_{\text{env}}(\beta, a)(\mu) > 0$ *then* $\mu \in \mathcal{T}(\text{last}(\beta), a)$*. By* $Strat^{\text{who}}_{\mathcal{M}}$ *for* $\text{who} \in \{\text{con}, \text{env}\}$*, we denote the set of all player-*who *strategies.*

*A player-*con *strategy* σ_{con} *is called* simple *if it only maps to Dirac distributions and if for all* $\beta, \beta' \in Path^{\text{fin}}_{\mathcal{M}}$ *with* $\text{last}(\beta) = \text{last}(\beta')$ *we have* $\sigma_{\text{con}}(\beta) = \sigma_{\text{con}}(\beta')$*. We can interpret it as being of the form* $\sigma_{\text{con}} \colon S \to Act$*. A player-*env *strategy* σ_{env} *is called* simple *if it only maps to Dirac distributions and if for all* $\beta, \beta' \in Path^{\text{fin}}_{\mathcal{M}}$ *with* $\text{last}(\beta) = \text{last}(\beta')$ *and* $a \in Act$ *we have* $\sigma_{\text{con}}(\beta, a) = \sigma_{\text{con}}(\beta', a)$*. We can interpret it as being of the form* $\sigma_{\text{con}} \colon (S, Act) \to Distr(S)$*. By* $Strat^{\text{who,simple}}_{\mathcal{M}}$ *for* $\text{who} \in \{\text{con}, \text{env}\}$*, we denote the set of simple player-*who *strategies.*

The joint scheduler $\text{join}(\sigma_{\text{con}}, \sigma_{\text{env}}) \in Sched_{\mathcal{M}}$ *of a player-*con *strategy* σ_{con} *and a player-*env *strategy* σ_{env} *is defined so that for* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$*,* $a \in Act$ *and* $\mu \in Distr(S)$ *we have*

$$\text{join}(\sigma_{\text{con}}, \sigma_{\text{env}})(\beta)((a, \mu)) \stackrel{\text{def}}{=} \sigma_{\text{con}}(\beta)(a)\sigma_{\text{env}}(\beta, a)(\mu).$$

A player-con strategy together with a player-env strategy is thus sufficient to determine the stochastic behaviour of a PA.

Because player con is responsible for choosing the actions, for fairness to hold, we require it to ensure that the behaviour of the model is fair with probability 1, regardless of what player env does.

Definition 5.2. *A player-*con *strategy* $\sigma_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}}$ *of a PA* $\mathcal{M}$ *is called* Act_{fair}-fair *if for all player-*env *strategies* $\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}$ *we have*

$$Pr_{\mathcal{M}, \text{join}(\sigma_{\text{con}}, \sigma_{\text{env}})}(Path^{Act_{\text{fair}}}_{\mathcal{M}}) = 1.$$

By $Strat^{Act_{\text{fair}}}_{\mathcal{M}}$ *we denote the set of all* Act_{fair}*-fair player-*con *strategies of* $\mathcal{M}$*.*

With these preparations, we can define the extremal values over all possible resolutions of strategies.

Definition 5.3. *For a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$*, a set* $Reach \subseteq S$ *of states, and a subset* $Act_{\text{fair}} \subseteq Act$ *of the actions of* $\mathcal{M}$ *and for* $\sigma_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}}$ *and* $\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}$ *we define*

$$\text{val}^{\sigma_{\text{con}}, \sigma_{\text{env}}}_{\mathcal{M}, Reach} \stackrel{\text{def}}{=} \text{val}^{\text{join}(\sigma_{\text{con}}, \sigma_{\text{env}})}_{\mathcal{M}, Reach}.$$

Then, let

$$\text{val}^{+,-}_{\mathcal{M}, Reach} \stackrel{\text{def}}{=} \sup_{\sigma_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}}} \inf_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{val}^{\sigma_{\text{con}}, \sigma_{\text{env}}}_{\mathcal{M}, Reach} \text{ and}$$

$$\mathrm{val}^{-,+,Act_{\mathrm{fair}}}_{\mathcal{M},Reach} \stackrel{\mathrm{def}}{=} \inf_{\sigma_{\mathrm{con}} \in Strat^{Act_{\mathrm{fair}}}_{\mathcal{M}}} \sup_{\sigma_{\mathrm{env}} \in Strat^{\mathrm{env}}_{\mathcal{M}}} \mathrm{val}^{\sigma_{\mathrm{con}},\sigma_{\mathrm{env}}}_{\mathcal{M},Reach}.$$

We use the fairness restriction only if player con minimises. The reason is that we will later on use the game interpretation of PAs to build a game-based semantics of PHAs. There, if player con maximises, it has no interest in choosing nonfair behaviours (see the discussion in Section 2.2 on time-convergent behaviours). In Definition 5.3, $\mathrm{val}^{+,+}_{\mathcal{M},Reach}$ and $\mathrm{val}^{-,-,Act_{\mathrm{fair}}}_{\mathcal{M},Reach}$, that is values in which the two players collaborate, are left out, because these values would be the same as the ones of Definition 3.11 and Definition 3.12.

Example 5.4. *Reconsider the PA $\mathcal{M}$ of Figure 3.1 and let Reach $\stackrel{\mathrm{def}}{=} \{s_2\}$. We want to compute* $\mathrm{val}^{+,-}_{\mathcal{M},Reach}$*. Now, player* con*, who has control over the choice of actions, tries to maximise the probability to reach Reach, whereas player* env*, who controls the remaining choices, tries to minimise it.*

It turns out that we have $\mathrm{val}^{+,-}_{\mathcal{M},Reach} = \frac{20}{21} \approx 0.952$*: in s_0, the best choice for player* con *is the action a. Here, there is only one choice for player* env *left. By a repeated choice of this transition, s_2 is reached with probability $\frac{2}{3}$, whereas s_1 is reached with probability $\frac{1}{3}$. In contrast to the maximising scheduler of Example 3.13, for player* con *it does not make sense to choose action a in s_1, because it has no control about the concrete successor distribution. If it had choosen this action, player* env *would choose the distribution leading back to s_1 with certainty instead of the one to s_0. Instead, in s_1 player* con *should repeatedly choose b. Under the condition that we are in state s_1, this leads to a probability of $\frac{6}{7}$ to reach s_2. In total, the reachability probability is thus $\frac{2}{3} + \frac{1}{3} \cdot \frac{6}{7} = \frac{20}{21} \approx 0.952$.* △

5.2 Controlled Probabilistic Hybrid Automata

Our high-level model is exactly the same as the one of Definition 3.20. However, we interpret it in a different way. We assume that we are in control of whether we let a certain amount of time pass, or whether to execute a certain action. Then, a possibly malevolent environment resolves the internal nondeterminism of the timed transition or the command chosen. This is then followed by a probabilistic choice.

Example 5.5. *In Figure 5.1 we sketch a variant of the PHA of Figure 3.3. In contrast to the former example, the automaton does not have an* Error *mode. Instead, we assume that it is broken from the beginning, and needs to be steered into the failsafe mode* Safe *as quickly as possible.*

The automaton now contains more timed behaviours. For instance, in mode Check*, instead of $\dot{T} = -\frac{T}{2}$, the behaviour of the temperature is now given as $-0.7T \leq \dot{T} \leq -0.3T$. In Figure 5.2 we depict the possible behaviours if we are in the state with mode* Check *and have a temperature of $T = 5$ and a timer value of $t = 0$. From the graph we see that if player* con *chooses to wait for time* 0.25*, player* env *can choose a temperature between about* 4.20 *and* 4.64*.* △

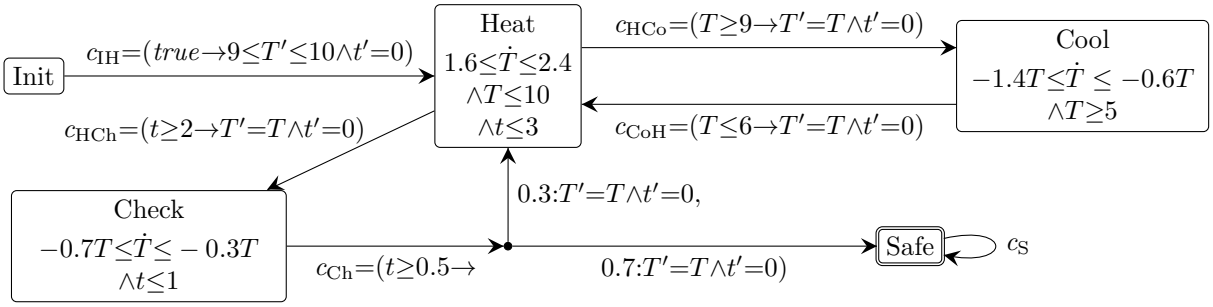


Figure 5.1: Controlled PHA modelling a thermostat.

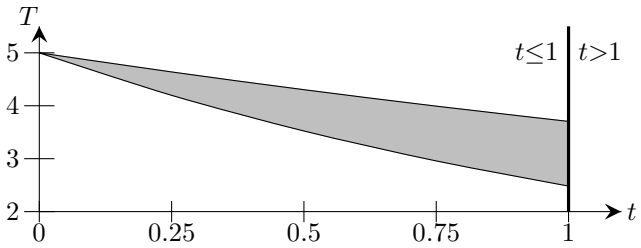


Figure 5.2: Post operator of PHA with partial control.

We consider the division of the two choices in which player con chooses commands or time durations whereas player env chooses the exact outcome as natural: the controller of a hybrid system is often given as a digital circuit, which can thus choose to between a finite number of different outputs, corresponding to the guarded commands. Using a stopwatch, it can also choose to wait for a specific time duration. The environment on the other hand is usually continuous, which is taken into account by allowing it to choose from the remaining continuous nondeterminism.

The definition of values in this interpretation is then as follows.

Definition 5.6. *Let $\mathcal{H} = (M, k, \overline{m}, \langle Post_m\rangle_{m\in M}, Cmds)$ be a PHA with a mode m_{Reach}. We define*

$$\mathrm{val}^{+,-}_{\mathcal{H},m_{Reach}} \stackrel{\mathrm{def}}{=} \mathrm{val}^{+,-}_{[\![\mathcal{H}]\!],\{m_{Reach}\}\times\mathbb{R}^k} \text{ and } \mathrm{val}^{-,+}_{\mathcal{H},m_{Reach}} \stackrel{\mathrm{def}}{=} \mathrm{val}^{-,+,Cmds}_{[\![\mathcal{H}]\!],\{m_{Reach}\}\times\mathbb{R}^k}.$$

As in Chapter 3, we also consider the time-restricted semantics and prove that it is equivalent to the semantics without time restrictions. In Chapter 3 we did so, because the abstractions we obtain from hybrid solvers are abstractions for the time restricted semantics, rather than for the original one. In this chapter, the time restrictions will also be used to obtain more precise results in the later abstraction. Thus, in contrast to the basic setting, it is even advantageous to consider the time restriction, rather than just necessary because of the way the hybrid solvers work.

Lemma 5.7. *Given a PHA $\mathcal{H}$ with commands $Cmds$, a set of states $Reach_{[\![\mathcal{H}]\!]} = Reach_{[\![\mathcal{H},\mathbf{T}]\!]} \subseteq M\times\mathbb{R}^k$, and a time restriction $\mathbf{T}$ we have*

$$\mathrm{val}^{+,-}_{[\![\mathcal{H}]\!],Reach_{[\![\mathcal{H}]\!]}} = \mathrm{val}^{+,-}_{[\![\mathcal{H},\mathbf{T}]\!],Reach_{[\![\mathcal{H},\mathbf{T}]\!]}} \text{ and } \mathrm{val}^{-,+,Cmds}_{[\![\mathcal{H}]\!],Reach_{[\![\mathcal{H}]\!]}} = \mathrm{val}^{-,+,Cmds}_{[\![\mathcal{H},\mathbf{T}]\!],Reach_{[\![\mathcal{H},\mathbf{T}]\!]}}.$$

Proof. The result follows similarly to Lemma 3.26: assume $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$ and that the dimension of $\mathcal{H}$ is k. Consider a situation where we are in mode m with variable valuation v_{start} of $\mathcal{H}$ and player con chooses a time duration of $\mathtt{t}$ with $\mathbf{t}_m(v_{\text{start}}) < \mathtt{t}$ in the semantics without time restriction. In the time-restricted semantics, player con can use a similar construction as the time-restricted scheduler $\mathbf{T}(\sigma)$ in Definition 3.25. By definition of the post operator and the time restriction (cf. Definition 2.12 and Definition 2.16), there is a function $f \colon \mathbb{R}^k \to \mathbb{R}_{\geq 0}$ for which

- $f(v) \in \mathbf{t}_m(v)$ for all $v \in \mathbb{R}^k$,
- $V_0 \stackrel{\text{def}}{=} \{v_{\text{start}}\}$,
- $V_{i+1} \stackrel{\text{def}}{=} \bigcup_{v \in V_i} Post_m(v, f(v))$ for $i \geq 0$, and
- the fixed point of the sequence V_i for $i \to \infty$ is V,

and we have

$$Post_m(v_{\text{start}}, \mathtt{t}) = V.$$

This means that player con can divide the duration $\mathtt{t}$ into a series of smaller time steps, which lead to the same set of states being potentially reached. From this, player env has the same choice of the actual successor state. As this division of $\mathtt{t}$ into several shorter durations is also possible in the original model, it also does not increase the power of the controller player. Because of this, the reachability probabilities in both models agree. □

5.3 Abstractions

As in Chapter 3, we want to estimate bounds on the behaviour of the controlled PHA. In addition, we want to synthesise a strategy in the abstraction which is guaranteed to maintain a certain bound also in the concrete model. To do so, we have to extend the previous definition of an abstraction slightly, to take into account that we have two competing players.

Definition 5.8. *Consider a PHA $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$, an abstract state space $\mathbf{A} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ of corresponding dimension and modes as well as a time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$. A* game abstraction

$$\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \mathbf{z}_0, Cmds \uplus \{\tau\}, \mathcal{T})$$

has to fulfil the requirements of the abstractions in Definition 2.24. In addition, it contains a new state $\bot$ and we add the requirements

- *for $\mathbf{z} \in \mathbf{A}$ for which there is $s = (m, v) \in \mathbf{z}$ so that for all $t \geq 0$ we have $Post_m(v, t) = \{v\}$, there is $[\mathbf{z}' \mapsto 1] \in \mathcal{T}(\mathbf{z}, \tau)$ where $s \in \mathbf{z}' \in \mathbf{A}$,*
- *if for some $\mathbf{z} \in \mathbf{A}$ and $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ we have $\mathbf{z} \setminus g \neq \emptyset$, then we have $[\bot \mapsto 1] \in \mathcal{T}(\mathbf{z}, c)$, and*
- *for all $a \in Cmds \uplus \{\tau\}$ we have $\mathcal{T}(\bot, a) \stackrel{\text{def}}{=} \{[\bot \mapsto 1]\}$.*

By $GAbs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ we denote the set of all game abstractions.

The state $\bot$ and the transitions to this state are used to prevent that player con chooses to execute c in an abstract state $\mathbf{z}$ which contains a state s in which c cannot actually be executed. In principle, we could also have set $\mathcal{T}(\mathbf{z}, c) \stackrel{\text{def}}{=} \emptyset$ in such a case. Basically, having a transition to $\bot$ has the same effect: we will define values so that transitions to $\bot$ are always most disadvantageous for player con, implying that player env will always choose such a transition in case player con executes c in the situation above. Thus, the transition has the same effect as disabling the command completely. However, we still want to use the LTSs computed by usual hybrid system solvers, and these solvers compute overapproximations and not underapproximations of the actual transitions. The timed self loops we introduce in Definition 5.8 are used for the same effect.

Because we are now considering two competing players, we can no longer use simulation relations to prove properties of abstractions. Instead, we will give a number of intermediate models, for which we show that they do not improve the chances of player con to win the game. Finally, this will allow us to build a chain of inequations allowing us to show that the abstraction yields safe bounds.

The first intermediate model is obtained by copying the states of the semantics for each abstract state they are contained in. The transitions between the new states are basically the same.

Definition 5.9. *Consider a PHA $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$, a time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$, and a game abstraction $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in GAbs(\mathcal{H}, \mathbf{A}, \mathbf{T})$. We say that*

$$\mathcal{M} \times [\![\mathcal{H}, \mathbf{T}]\!] \stackrel{\text{def}}{=} (S_{\text{copy}}, \overline{s}_{\text{copy}}, Act, \mathcal{T}_{\text{copy}})$$

is the $\mathcal{M}$-copied semantics *of $\mathcal{H}$ with time restriction $\mathbf{T}$ if*

- $S_{\text{copy}} \stackrel{\text{def}}{=} \{(\mathbf{z}, s) \mid \mathbf{z} \in \mathbf{A} \wedge s \in \mathbf{z}\}$,
- $\overline{s}_{\text{copy}} \stackrel{\text{def}}{=} (\overline{\mathbf{z}}, (\overline{m}, 0, \ldots, 0))$,
- *for all $(\mathbf{z}, s) \in S_{\text{copy}}$ and all $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ with $s \in g$ and*
  - $\mu = [s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \in \{[s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \mid s'_1 \in u_1(s), \ldots, s'_n \in u_n(s)\}$,
  - $[\mathbf{z}'_1 \mapsto p_1, \ldots, \mathbf{z}'_n \mapsto p_n] \in Lift_{\mathbf{A}}(\mu) \cap \mathcal{T}(\mathbf{z}, c)$ *with $s'_i \in \mathbf{z}'_i$ for $i \in \{1, \ldots, n\}$*

  we have
  $$[(\mathbf{z}'_1, s'_1) \mapsto p_1, \ldots, (\mathbf{z}'_n, s'_n) \mapsto p_n] \in \mathcal{T}_{\text{copy}}((\mathbf{z}, s), c).$$
- *For all $(\mathbf{z}, s) \in S_{\text{copy}}$ with $s = (m, v)$ and all $\mathtt{t} \in \mathbb{R}_{\geq 0}$*
  - *let $\mathtt{t}_{\mathbf{T}} \stackrel{\text{def}}{=} \min\{\mathtt{t}, \mathbf{t}_m(v)\}$,*
  - *consider $s' = (m, v')$ with $v' \in Post_m(v, \mathtt{t}_{\mathbf{T}})$,*
  - *then if $s' \in \mathbf{z}$, we require $[(\mathbf{z}, s') \mapsto 1] \in \mathcal{T}_{\text{copy}}((\mathbf{z}, s), \mathtt{t})$,*
  - *else, for all $\mathbf{z}' \in \mathcal{T}(\mathbf{z}, \tau)$ with $s' \in \mathbf{z}'$ we require that $[(\mathbf{z}', s') \mapsto 1] \in \mathcal{T}_{\text{copy}}((\mathbf{z}, s), \mathtt{t})$.*

There are no other transitions in $\mathcal{T}_{\text{copy}}$ *than the ones described above.*

As we have only copied the states of the semantics a few times, the bounds on reachability probabilities stay the same. However, we will only need the direction that they do not improve the reachability values for player con.

Lemma 5.10. *Consider a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *with a mode* m_{Reach}. *Let* $Reach_{[\![\mathcal{H},\mathbf{T}]\!]} \stackrel{\text{def}}{=} \{m_{Reach}\} \times \mathbb{R}^k$ *and* $Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]} \stackrel{\text{def}}{=} \{(\mathbf{z}, s) \in \mathbf{A} \times (M \times \mathbb{R}^k) \mid s \in \mathbf{z} \cap Reach_{[\![\mathcal{H},\mathbf{T}]\!]}\}$. *Then we have*

$$\begin{aligned} \text{val}^{+,-}_{[\![\mathcal{H},\mathbf{T}]\!],Reach_{[\![\mathcal{H},\mathbf{T}]\!]}} &\geq \text{val}^{+,-}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}} \text{ and} \\ \text{val}^{-,+,Cmds}_{[\![\mathcal{H},\mathbf{T}]\!],Reach_{[\![\mathcal{H},\mathbf{T}]\!]}} &\leq \text{val}^{-,+,Cmds}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}}. \end{aligned}$$

Proof. We consider the case

$$\text{val}^{-,+,Cmds}_{[\![\mathcal{H},\mathbf{T}]\!],Reach_{[\![\mathcal{H},\mathbf{T}]\!]}} \leq \text{val}^{-,+,Cmds}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}}. \tag{5.1}$$

The other one is similar but simpler, because we do not have to take fairness into account. To show the validity of Equation 5.1, it suffices to show that for each $\sigma_{\text{con,copy}} \in Strat^{Cmds}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}$ there is $\sigma_{\text{con}} \in Strat^{Cmds}_{[\![\mathcal{H},\mathbf{T}]\!]}$ with

$$\sup_{\sigma_{\text{env}} \in Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}} \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{[\![\mathcal{H},\mathbf{T}]\!],Reach_{[\![\mathcal{H},\mathbf{T}]\!]}} \leq \sup_{\sigma_{\text{env,copy}} \in Strat^{\text{env}}_{[\![\mathcal{H},\mathbf{T}]\!]}} \text{val}^{\sigma_{\text{con,copy}},\sigma_{\text{env,copy}}}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}}. \tag{5.2}$$

Consider $\sigma_{\text{con,copy}} \in Strat^{Cmds}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}$. We can construct $\sigma_{\text{con,impr}} \in Strat^{Cmds}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}$ so that for all $\beta, \beta' \in Path^{\text{fin}}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}$ with

$$\begin{aligned} \beta &= (\mathbf{z}_0, s_0)\, a_0\, \mu_0\, (\mathbf{z}_1, s_1)\, a_1\, \mu_1\, \ldots\, a_{n-1}\, \mu_{n-1}\, (\mathbf{z}_n, s_n), \\ \beta' &= (\mathbf{z}'_0, s_0)\, a_0\, \mu_0\, (\mathbf{z}'_1, s_1)\, a_1\, \mu_1\, \ldots\, a_{n-1}\, \mu_{n-1}\, (\mathbf{z}'_n, s_n), \end{aligned} \tag{5.3}$$

that is for finite paths which only differ in the components of abstract states, we have

$$\sigma_{\text{con,impr}}(\beta) = \sigma_{\text{con,impr}}(\beta'),$$

and

$$\sup_{\sigma_{\text{env,copy}} \in Strat^{\text{env}}_{[\![\mathcal{H},\mathbf{T}]\!]}} \text{val}^{\sigma_{\text{con,impr}},\sigma_{\text{env,copy}}}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}} \leq \sup_{\sigma_{\text{env,copy}} \in Strat^{\text{env}}_{[\![\mathcal{H},\mathbf{T}]\!]}} \text{val}^{\sigma_{\text{con,copy}},\sigma_{\text{env,copy}}}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}}.$$

This is the case because the abstract state component does not influence the reachability probability, so that $\sigma_{\text{con,impr}}$ can choose the better alternative of the two paths β and β'. As there are only finitely many abstract states, this argument carries over to more than two of such paths. Fairness is maintained by this construction, because otherwise the original strategy $\sigma_{\text{env,copy}}$ could not have been fair before: there would be a strategy of player env so that the probability is concentrated in the paths from which the decisions

are taken over in $\sigma_{\text{con,impr}}$. In turn, if $\sigma_{\text{env,copy}}$ was not fair, then the probability of fair paths would be below 1 when using the player env strategy and $\sigma_{\text{env,copy}}$.

Because of this, it suffices to show Equation 5.2 for these kinds of strategies. We can then define $\sigma_{\text{con}} \in \mathit{Strat}^{\mathit{Cmds}}_{[\![\mathcal{H},\mathbf{T}]\!]}$ for which with β in the form of Equation 5.3 and

$$\beta_{[\![\mathcal{H},\mathbf{T}]\!]} \stackrel{\text{def}}{=} s_0\, a_0\, \mu_0\, s_1\, a_1\, \mu_1\, \ldots\, a_{n-1}\, \mu_{n-1}\, s_n$$

we have

$$\sigma_{\text{con}}(\beta_{[\![\mathcal{H},\mathbf{T}]\!]}) \stackrel{\text{def}}{=} \sigma_{\text{con,impr}}(\beta). \tag{5.4}$$

We can consider an infinite-state model $[\![\mathcal{H},\mathbf{T}]\!]_{\sigma_{\text{con}}}$ in which we have integrated the decisions of σ_{con} into the model, which means that for all $\sigma_{\text{env}} \in \mathit{Strat}^{\text{env}}_{[\![\mathcal{H},\mathbf{T}]\!]}$ we have

$$\text{val}^{\sigma_{\text{env}}}_{[\![\mathcal{H},\mathbf{T}]\!]_{\sigma_{\text{con}}}} = \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{[\![\mathcal{H},\mathbf{T}]\!]}, \tag{5.5}$$

and a corresponding construction is possible to obtain $(\mathcal{M} \times [\![\mathcal{H},\mathbf{T}]\!])_{\sigma_{\text{con,impr}}}$ from $\mathcal{M} \times [\![\mathcal{H},\mathbf{T}]\!]$. Also, $[\![\mathcal{H},\mathbf{T}]\!]_{\sigma_{\text{con}}}$ and $(\mathcal{M} \times [\![\mathcal{H},\mathbf{T}]\!])_{\sigma_{\text{con,impr}}}$ simulate each other in the notion of Definition 3.15, which by Lemma 3.17 means that

$$\sup_{\sigma_{\text{env}} \in \mathit{Strat}^{\text{env}}_{\mathcal{M}_{\sigma_{\text{con}}}}} \text{val}^{\sigma_{\text{env}}}_{[\![\mathcal{H},\mathbf{T}]\!]_{\sigma_{\text{con}}}} = \sup_{\sigma_{\text{env,copy}} \in \mathit{Strat}^{\text{env}}_{(\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!])_{\text{con,impr}}}} \text{val}^{\sigma_{\text{env,copy}}}_{(\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!])_{\sigma_{\text{con,impr}}}}. \tag{5.6}$$

From Equation 5.5 and Equation 5.6 we can conclude the validity of Equation 5.2. □

In the next intermediate step, player con can no longer choose to wait for an exact time duration. Instead, it can only choose whether to execute a command or to let time pass. If it decides to let time pass, a sequence of timed transitions is executed, each with the maximal time allowed by the time restriction. These transitions are repeated until the current abstract state is left, if this is possible. Consider the sequence of state sets F_i which are reached by the i-th timed transition. The valuations of continuous model variables which are reached when a timed transition is chosen can be specified using the fixed point F of this sequence.

This intermediate step limits the power of player con, as it can no longer choose the exact time to wait. On the other hand, in case an abstract state is always eventually left by the sequence of timed transitions described above, we do not have to add a timed self loop to this abstract state. If we were not able to leave out most of the self loops, we would obtain very bad bounds (that is, 0 in most cases) on the reachability probabilities in case of a minimising player env. This player would then always choose to loop back to the abstract state in case player con chooses to let time pass.

Definition 5.11. *Consider a PHA $\mathcal{H} = (M, k, \overline{m}, \langle \mathit{Post}_m \rangle_{m\in M}, \mathit{Cmds})$, a time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m\in M}$, and a game abstraction $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \overline{\mathbf{z}}, \mathit{Cmds} \uplus \{\tau\}, \mathcal{T}) \in \mathit{GAbs}(\mathcal{H}, \mathbf{A}, \mathbf{T})$. We say that*

$$(\mathcal{M}, \tau) \times [\![\mathcal{H},\mathbf{T}]\!] \stackrel{\text{def}}{=} (S_{\text{copy}}, \overline{s}_{\text{copy}}, \mathit{Cmds} \uplus \{\tau\}, \mathcal{T}_{\text{copy}})$$

is the $\mathcal{M}$-copied τ-semantics *of* $\mathcal{H}$. *Here,* S_{copy}, $\overline{s}_{\text{copy}}$ *and the command transitions are as in the* $\mathcal{M}$*-copied semantics.*
Concerning the timed transitions, for $s = (m, v) \in \mathbf{z}$ *we define* F *as the fixed point of the sequence*

$$F^0 \stackrel{\text{def}}{=} \{v\}, \quad F^{n+1} \stackrel{\text{def}}{=} (F^n \setminus \mathbf{z}) \cup \left(\bigcup_{v' \in F^n \cap \mathbf{z}} Post_m(v', \mathbf{t}_m(v')) \right).$$

Consider an arbitrary $s' \in \{m\} \times F$. *For all* $[\mathbf{z}' \mapsto 1] \in \mathcal{T}(\mathbf{z}, \tau)$ *with* $s' \in \mathbf{z}'$ *we require that* $[(\mathbf{z}', s') \mapsto 1] \in \mathcal{T}_{\text{copy}}((\mathbf{z}, s), \tau)$. *These are all transitions of* $\mathcal{T}_{\text{copy}}((\mathbf{z}, s), \tau)$.

As we limit the choices of player con, the bounds get worse.

Lemma 5.12. *Consider a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *with a mode* m_{Reach}. *Let* $Reach_{\mathcal{M} \times [\![\mathcal{H}, \mathbf{T}]\!]} \stackrel{\text{def}}{=} Reach_{(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!]} \stackrel{\text{def}}{=} \{(\mathbf{z}, s) \in \mathbf{A} \times (M \times \mathbb{R}^k) \mid s \in \mathbf{z} \cap (\{m_{Reach}\} \times \mathbb{R}^k)\}$. *Then we have*

$$\begin{aligned} &\text{val}^{+,-}_{\mathcal{M} \times [\![\mathcal{H}, \mathbf{T}]\!], Reach_{\mathcal{M} \times [\![\mathcal{H}, \mathbf{T}]\!]}} \geq \text{val}^{+,-}_{(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!], Reach_{(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!]}} \text{ and} \\ &\text{val}^{-,+,Cmds}_{\mathcal{M} \times [\![\mathcal{H}, \mathbf{T}]\!], Reach_{\mathcal{M} \times [\![\mathcal{H}, \mathbf{T}]\!]}} \leq \text{val}^{-,+,Cmds}_{(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!], Reach_{(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!]}}. \end{aligned}$$

Proof. We remark that $F(s, \tau)$ is well defined, that is the fixed point of Definition 5.11 always exists. This holds because of requirements 2 and 3 of Definition 2.12.
The transitions resulting from commands are the same in both models. Concerning the timed transitions, the choice of player con is restricted in $(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!]$: instead of choosing an exact time duration, the player can only choose τ to execute a timed transition. The resulting transition defined by the fixed point F corresponds to forcing player con to let a maximal amount of time pass repeatedly, until either the current abstract state is left or letting time pass has no further effect. Player env can choose the exact target of these transitions. As this limits the power of player con but increases the one of player env, the inequations follow. □

The next model we consider consists of several copies of a game abstraction.

Definition 5.13. *Consider a game abstraction* $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \overline{\mathbf{z}}, Act, \mathcal{T})$ *of a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$. *The* S-copied abstraction *is defined as*

$$\mathcal{M} \times S \stackrel{\text{def}}{=} (S_{\text{copy}} \uplus \{\bot\}, \overline{s}_{\text{copy}}, Act, \mathcal{T}_{\text{copy}}),$$

where for $S \stackrel{\text{def}}{=} M \times \mathbb{R}^k$ *we have*

- $S_{\text{copy}} \stackrel{\text{def}}{=} \{(\mathbf{z}, s) \mid \mathbf{z} \in \mathbf{A} \wedge s \in \mathbf{z}\}$,
- $\overline{s}_{\text{copy}} \stackrel{\text{def}}{=} (\overline{\mathbf{z}}, (\overline{m}, 0, \ldots, 0))$,
- *for all* $(\mathbf{z}, s) \in S_{\text{copy}}$, $a \in Act$, $[\mathbf{z}_1 \mapsto p_1, \ldots, \mathbf{z}_n \mapsto p_n] \in \mathcal{T}(\mathbf{z}, a)$ *and* $s'_i \in \mathbf{z}'_i$ *for* $1 \leq i \leq n$, *we have* $\{[(\mathbf{z}'_1, s'_1) \mapsto p_1, \ldots, (\mathbf{z}'_n, s'_n) \mapsto p_n]\} \in \mathcal{T}_{\text{copy}}((\mathbf{z}, s), a)$,

- *for all* $\mathbf{z} \in \mathbf{A}$ *and* $a \in Act$*, if* $[\bot \mapsto 1] \in \mathcal{T}(\mathbf{z}, a)$ *then* $[\bot \mapsto 1] \in \mathcal{T}_{\text{copy}}(\mathbf{z}, a)$*,*
- *for all* $a \in Act$ *we have* $\mathcal{T}_{\text{copy}}(\bot, a) \stackrel{\text{def}}{=} \{[\bot \mapsto 1]\}$*, and*
- *there are no other transitions than the ones required above.*

Basically, for each state $s \in \mathbf{z}$ in $\mathcal{M}$, the state $(s, \mathbf{z})$ of $\mathcal{M}_{\text{copy}}$ contains the same possible behaviours as $\mathbf{z}$. If there is a transition from $\mathbf{z}$ to $\mathbf{z}'$ in $\mathcal{M}$, in $\mathcal{M}_{\text{copy}}$ we have a transition with the same action and probability from $(s, \mathbf{z})$ to $(s', \mathbf{z}')$, for all $s' \in \mathbf{z}'$. In the $\mathcal{M}$-copied semantics in Definition 5.9, we have already used a similar construction in which we have constructed a model that contains multiple copies of the states of the semantics.

Indeed, $\mathcal{M}_{\text{copy}}$ is just a blown-up version of $\mathcal{M}$, which is however easier to compare to the $\mathcal{M}$-copied τ-semantics. The probability bounds which player con can enforce in $\mathcal{M}_{\text{copy}}$ are worse than in the $\mathcal{M}$-copied τ-semantics, as the model limits the power of this player further.

Lemma 5.14. *For a game abstraction* $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T})$ *of a PHA* $\mathcal{H}$ *with a mode* m_{Reach} *and* $Reach_{(\mathcal{M},\tau)\times[\![\mathcal{H}]\!]} \stackrel{\text{def}}{=} Reach_{\mathcal{M}\times S} \stackrel{\text{def}}{=} \{(\mathbf{z}, s) \in \mathbf{A} \times (M \times \mathbb{R}^k) \mid s \in \mathbf{z} \cap \{m_{Reach}\} \times \mathbb{R}^k\}$ *we have*

$$\begin{aligned} \text{val}^{+,-}_{(\mathcal{M},\tau)\times[\![\mathcal{H}]\!], Reach_{(\mathcal{M},\tau)\times[\![\mathcal{H}]\!]}} &\geq \text{val}^{+,-}_{\mathcal{M}\times S, Reach_{\mathcal{M}\times S}} \text{ and} \\ \text{val}^{-,+,Cmds}_{(\mathcal{M},\tau)\times[\![\mathcal{H}]\!], Reach_{(\mathcal{M},\tau)\times[\![\mathcal{H}]\!]}} &\leq \text{val}^{-,+,Cmds}_{\mathcal{M}\times S, Reach_{\mathcal{M}\times S}\uplus\{\bot\}}. \end{aligned}$$

Proof. We compare the two models

$$(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!] = (S, \overline{s}, Act, \mathcal{T}) \text{ and } \mathcal{M} \times S = (S \uplus \{\bot\}, \overline{s}, Act, \mathcal{T}').$$

By their definition, we have for all $s \in S$ and $a \in Act$ that $\mathcal{T}(s, a) \subseteq \mathcal{T}'(s, a)$, and if $\mathcal{T}(s, a) = \emptyset$ then $[\bot \mapsto 1] \in \mathcal{T}'(s, a)$. From this and by the definition of the set of states to reach, in each state of S, in $\mathcal{M} \times S$ we leave more choices for player env and never add transitions which are an advantage to player con. Because of this, we can conclude the inequations. □

As the S-copied abstraction consists of copies of a game abstraction, the reachability values in the two models are the same.

Lemma 5.15. *Consider a game abstraction* $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T})$ *and its* S*-copied abstraction* $\mathcal{M} \times S = (S' \uplus \{\bot\}, \overline{s}', Cmds \uplus \{\tau\}, \mathcal{T}')$ *of a PHA* $\mathcal{H}$ *with a mode* m_{Reach}*. Then we have that for* $Reach_{\mathcal{M}} \stackrel{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{Reach}\}$ *and* $Reach_{\mathcal{M}\times S} \stackrel{\text{def}}{=} Reach_{\mathcal{M}} \times S$ *we have*

$$\begin{aligned} \text{val}^{+,-}_{\mathcal{M}, Reach_{\mathcal{M}}} &\leq \text{val}^{+,-}_{\mathcal{M}\times S, Reach_{\mathcal{M}\times S}} \text{ and} \\ \text{val}^{-,+,Cmds}_{\mathcal{M}, Reach_{\mathcal{M}}\uplus\{\bot\}} &\geq \text{val}^{-,+,Cmds}_{\mathcal{M}\times S, Reach_{\mathcal{M}\times S}\uplus\{\bot\}}. \end{aligned}$$

Proof. The proof is similar to the one of Lemma 5.10. We consider the case

$$\text{val}^{-,+,Cmds}_{\mathcal{M}, Reach_{\mathcal{M}}\uplus\{\bot\}} \geq \text{val}^{-,+,Cmds}_{\mathcal{M}\times S, Reach_{\mathcal{M}\times S}\uplus\{\bot\}}. \tag{5.7}$$

The other one is similar but simpler, because we do not have to take fairness into account. To show the validity of Equation 5.7, it suffices to show that for each $\sigma_{\text{con}} \in \mathit{Strat}^{\mathit{Cmds}}_{\mathcal{M}}$ there is $\sigma_{\text{con,copy}} \in \mathit{Strat}^{\mathit{Cmds}}_{\mathcal{M}\times S}$ with

$$\sup_{\sigma_{\text{env}} \in \mathit{Strat}^{\text{env}}_{\mathcal{M}}} \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M},\mathit{Reach}_{\mathcal{M}} \uplus \{\bot\}} \geq \sup_{\sigma_{\text{env,copy}} \in \mathit{Strat}^{\text{env}}_{\mathcal{M}\times S}} \text{val}^{\sigma_{\text{con,copy}},\sigma_{\text{env,copy}}}_{\mathcal{M}\times S,\mathit{Reach}_{\mathcal{M}\times S} \uplus \{\bot\}}. \tag{5.8}$$

Given σ_{con}, we define $\sigma_{\text{con,copy}}$ with

$$\sigma_{\text{con,copy}}(\beta_{\text{copy}})(a) \stackrel{\text{def}}{=} \sigma_{\text{con}}(\beta)(a)$$

for $\beta \in \mathit{Path}^{\text{fin}}_{\mathcal{M}}$ and $\beta_{\text{copy}} \in \mathit{Path}^{\text{fin}}_{\mathcal{M}\times S}$ where

$$\begin{aligned}\beta &= \mathbf{z}_0\, a_0\, \mu_0\, \ldots\, \mathbf{z}_{n-1}\, a_{n-1}\, \mu_{n-1}\, \mathbf{z}_n \text{ and}\\ \beta_{\text{copy}} &= (\mathbf{z}_0, s_0)\, a_0\, \mu_{0,\text{copy}}\, \ldots\, (\mathbf{z}_{n-1}, s_{n-1})\, a_{n-1}\, \mu_{n-1,\text{copy}}\, (\mathbf{z}_n, s_n),\end{aligned}$$

and where for all $i \geq 0$ we have $s_i \in \mathbf{z}_i$ and $\mu_{i,\text{copy}} = [(\mathbf{z}'_1, s'_1) \mapsto p_1, \ldots, (\mathbf{z}'_1, s'_m) \mapsto p_m]$ for $\mu_i = [\mathbf{z}'_1 \mapsto p_1, \ldots, \mathbf{z}'_1 \mapsto p_m]$ where $s'_j \in \mathbf{z}'_j$ for j with $1 \leq j \leq m$.

By construction, if $\sigma_{\text{con}} \in \mathit{Strat}^{\mathit{Cmds}}_{\mathcal{M}}$ then we have $\sigma_{\text{con,copy}} \in \mathit{Strat}^{\mathit{Cmds}}_{\mathcal{M}\times S}$. We can consider an infinite-state model $\mathcal{M}_{\sigma_{\text{con}}}$ in which we have integrated the decisions of σ_{con} into the model, which means that for all $\sigma_{\text{env}} \in \mathit{Strat}^{\text{env}}_{\mathcal{M}}$ we have

$$\text{val}^{\sigma_{\text{env}}}_{\mathcal{M}_{\sigma_{\text{con}}}} = \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M}}, \tag{5.9}$$

and a corresponding construction is possible to obtain $(\mathcal{M}\times S)_{\sigma_{\text{con,copy}}}$ from $\mathcal{M}\times S$. Also, $\mathcal{M}_{\sigma_{\text{con}}}$ and $(\mathcal{M}\times S)_{\sigma_{\text{con,copy}}}$ simulate each other in the notion of Definition 3.15, which by Lemma 3.17 means that

$$\sup_{\sigma_{\text{env}} \in \mathit{Strat}^{\text{env}}_{\mathcal{M}_{\sigma_{\text{con}}}}} \text{val}^{\sigma_{\text{env}}}_{\mathcal{M}_{\sigma_{\text{con}}}} = \sup_{\sigma_{\text{env,copy}} \in \mathit{Strat}^{\text{env}}_{\mathcal{M}_{\sigma_{\text{con}}}}} \text{val}^{\sigma_{\text{env,copy}}}_{(\mathcal{M}\times S)_{\sigma_{\text{con,copy}}}}. \tag{5.10}$$

From Equation 5.9 and Equation 5.10 we can conclude the validity of Equation 5.8. □

With the definitions and lemmas above, we can now state the main result of this chapter. It allows us to reason about the bounds on reachability values we can enforce given a partially malevolent and partially probabilistic environment.

Theorem 5.16. *Consider a PHA $\mathcal{H} = (M, k, \overline{m}, \langle \mathit{Post}_m \rangle_{m\in M}, \mathit{Cmds})$, a game abstraction $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \overline{\mathbf{z}}, \mathit{Cmds} \uplus \{\tau\}, \mathcal{T})$ and a mode m_{Reach}. Let $\mathit{Reach} \stackrel{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{\mathit{Reach}}\}$. Then*

$$\text{val}^{+,-}_{\mathcal{H},m_{\mathit{Reach}}} \geq \text{val}^{+,-}_{\mathcal{M},\mathit{Reach}} \text{ and } \text{val}^{-,+}_{\mathcal{H},m_{\mathit{Reach}}} \leq \text{val}^{-,+,\mathit{Cmds}}_{\mathcal{M},\mathit{Reach}\uplus\{\bot\}}.$$

Proof. We only consider the first inequation, as the second one is analogous. We have to show that for $\mathit{Reach}_{[\![\mathcal{H}]\!]} \stackrel{\text{def}}{=} \{m_{\mathit{Reach}}\} \times \mathbb{R}^k$ and $\mathit{Reach}_{\mathcal{M}} \stackrel{\text{def}}{=} \mathit{Reach}$ we have

$$\text{val}^{+,-}_{[\![\mathcal{H}]\!],\mathit{Reach}_{[\![\mathcal{H}]\!]}} \geq \text{val}^{+,-}_{\mathcal{M},\mathit{Reach}_{\mathcal{M}}}.$$

This follows by the following chain of inequations, derived from the previous lemmas:

$$\begin{aligned}
&\mathrm{val}^{+,-}_{[\![\mathcal{H}]\!],Reach_{[\![\mathcal{H}]\!]}} \\
&\overset{\text{Lem. 5.7}}{=} \mathrm{val}^{+,-}_{[\![\mathcal{H},\mathbf{T}]\!],Reach_{[\![\mathcal{H}]\!]}} \\
&\overset{\text{Lem. 5.10}}{\geq} \mathrm{val}^{+,-}_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{\mathcal{M}\times[\![\mathcal{H},\mathbf{T}]\!]}} \\
&\overset{\text{Lem. 5.12}}{\geq} \mathrm{val}^{+,-}_{(\mathcal{M},\tau)\times[\![\mathcal{H},\mathbf{T}]\!],Reach_{(\mathcal{M},\tau)\times[\![\mathcal{H},\mathbf{T}]\!]}} \\
&\overset{\text{Lem. 5.14}}{\geq} \mathrm{val}^{+,-}_{\mathcal{M}\times S,Reach_{\mathcal{M}\times S}} \\
&\overset{\text{Lem. 5.15}}{\geq} \mathrm{val}^{+,-}_{\mathcal{M},Reach_{\mathcal{M}}}.
\end{aligned}$$

□

Example 5.17. *In Figure 5.3, we sketch the chain of abstraction used in the proof of Theorem 5.16. For this, we abstract the PHA of Figure 5.1. As the semantics is uncountably large, we are only able show a small part of the semantics for the model.*

In the part $[\![\mathcal{H}]\!]$*, we show a piece of the semantics of mode* Check*. All the states are thus of the form* (Check, t, T) *which (for compactness of the drawing) we show only as* (t, T)*. We also round the temperature values to just one decimal place, which suffices for this example. Thus, for instance in the state* $(0, 5)$ *we are in mode* Check*, have a local timer value of* $t = 0$ *and a temperature of* $T = 5$*. In this state, we can choose to wait, for instance, for* 0.25 *time units. The environment can then choose the exact outcome of this timed transition, which means that we can move to* $(0.25, 4.2)$*,* $(0.25, 4.6)$*, or to one of the states with a temperature between* 4.2 *and* 4.6*. We could also choose to wait for time* 0.5*, then possibly ending up in* $(0.5, 4.3)$*, or any other time value.*

In $[\![\mathcal{H}, \mathbf{T}]\!]$ *we have applied a time restriction of* 0.25 *in some of the states, thereby removing some of the timed transitions.*

In $\mathcal{M} \times [\![\mathcal{H}, \mathbf{T}]\!]$*, we have built the* $\mathcal{M}$*-copied semantics. Most states shown are only contained in one abstract state, but* $(0.5, 4.0)$ *is contained in both* $\mathbf{z}_2$ *and* $\mathbf{z}_3$*. If in the game abstraction* $\mathcal{M}$ *we have transitions from* $\mathbf{z}_1$ *to both of them, we also have two corresponding transitions here.*

Then, in $(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!]$ *we give the* $\mathcal{M}$*-copied* τ*-semantics. As seen, we no longer have control about the exact time, but instead are only given the action* τ*. Previously, we had a transition from* $(\mathbf{z}_1, 0, 5)$ *to* $(\mathbf{z}_1, 0.25, 46)$ *from which we could then reach* $(\mathbf{z}_2, 0.5, 4.0)$ *and* $(\mathbf{z}_3, 0.5, 4.0)$*. Because of the fixed point construction, we now rather have direct transitions from* $(\mathbf{z}_1, 0, 5)$ *to* $(\mathbf{z}_2, 0.5, 4.0)$ *and* $(\mathbf{z}_3, 0.5, 4.0)$*.*

In the S*-copied abstraction* $\mathcal{M} \times S$*, we have a few more timed transitions. In the example here, the only additional transition given is from* $(\mathbf{z}_1, 0.25, 4.6)$ *to* $(\mathbf{z}_0, 0.25, 4.2)$*, which by the definition of the* S*-copied abstraction is indeed necessary. The definition of the game abstraction would allow to have more additional transitions, for instance it would be perfectly legal to have one from* $(\mathbf{z}_4, 1, 3.7)$ *to* $(\mathbf{z}_0, 0.25, 4.2)$*.*

Finally, $\mathcal{M}$ *is the game abstraction of* $\mathcal{H}$*. The difference to the previous model is that we now have transitions directly between abstract states rather than between pairs of concrete and abstract states.*

The example shows why the time restriction allows to obtain tighter bounds. If we had applied the abstraction without the time restrictions, we would have more t*-labelled transitions in the final abstraction. For instance, because we can move from* $(0, 5)$ *to* (1.37) *by letting time* 1 *pass, we would have a transition from* $\mathbf{z}_1$ *to* $\mathbf{z}_4$*, in addition to the existing ones. Now, because we can only choose whether to execute* τ *or a command, player* env *is free to choose* $\mathbf{z}_4$ *as the successor state, even though we intended to wait for a shorter time than* 1*.*

For the same reason, the fixed point construction is in some way an advantage for player con*, although it restricts the choices of player* con *and thus initially decreases the chance of player* con *to win the game. Without this construction, we would still have a* τ*-labelled transition from* $(\mathbf{z}_1, 0, 5)$ *to* $(\mathbf{z}_1, 0.25, 4.6)$ *in* $(\mathcal{M}, \tau) \times [\![\mathcal{H}, \mathbf{T}]\!]$*. This transitions would have to be represented by a* τ*-labelled self loop in the abstraction, leading to bad probability bounds in case player* env *is the minimising player.* △

5.3.1 Computing Abstractions

We have defined the game abstraction in Definition 5.8, and previously in Subsection 3.3.1 we already described how we can obtain abstractions for PHAs without control. To transfer the mechanism to the game setting, we have to add the missing behaviours involving $\perp$ described in Definition 5.8. For this, we have to extend the induced non-probabilistic hybrid automaton accordingly, so that we can reconstruct the behaviours of the abstraction of this automaton.

Definition 5.18. *The* induced game HA *of a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *is defined as*

$$\mathrm{gind}(\mathcal{H}) \stackrel{\text{def}}{=} (M \uplus \{m_\perp\}, k, \overline{m}, \langle Post_{\mathrm{gind},m} \rangle_{m \in M}, Cmds_{\mathrm{gind}}).$$

Let $S \stackrel{\text{def}}{=} (M \uplus \{m_\perp\}) \times \mathbb{R}^k$*. Each* $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ *induces a set of nonprobabilistic guarded commands* $\mathrm{gind}(c) \stackrel{\text{def}}{=} \{(g_{\mathrm{gind}} \to u_{\mathrm{gind},1}), \ldots, (g_{\mathrm{gind}} \to u_{\mathrm{gind},n}), (\overline{g} \to \overline{u})\}$ *where we have*

- $g_{\mathrm{gind}} \stackrel{\text{def}}{=} g \uplus (\{m_\perp\} \times \mathbb{R}^k)$*,*
- $u_{\mathrm{gind},i}(s) \stackrel{\text{def}}{=} u_i(s)$ *for* $s \in M \times \mathbb{R}^k$ *and* i *with* $1 \leq i \leq n$*,*
- $u_{\mathrm{gind},i}(s) \stackrel{\text{def}}{=} \{[s \mapsto 1]\}$ *for* $s \in \{m_\perp\} \times \mathbb{R}^k$ *and* i *with* $1 \leq i \leq n$*,*
- $\overline{g} \stackrel{\text{def}}{=} S \setminus g$*, and*
- *for* $s \in S$ *we have* $\overline{u}(s) \stackrel{\text{def}}{=} \{m_\perp\} \times \mathbb{R}^k$*.*

Further, consider $c_\tau \stackrel{\text{def}}{=} (g_\tau \to u_\tau)$ *where we have*

- $g_\tau \stackrel{\text{def}}{=} \{(m, v) \mid \forall t \geq 0.\ Post_m(v, t) = \{v\}\}$*, and*
- $u_\tau(s) \stackrel{\text{def}}{=} \{s\}$*.*

Then, $Cmds_{\mathrm{gind}} \stackrel{\text{def}}{=} \bigcup_{c \in Cmds} \mathrm{gind}(c) \cup \{c_\tau\}$*.*

For the post operators we have

- $Post_{\mathrm{gind},m}(v) \stackrel{\mathrm{def}}{=} Post_m$ *for* $m \in M$, *and*
- $Post_{\mathrm{gind},m_\perp}(v) \stackrel{\mathrm{def}}{=} \{v\}$.

We use the command c_τ to introduce timed self loops at the right place of the automaton. The mode $m_\perp$ is used to introduce the sink state $\perp$. The commands $(\overline{g} \to \overline{u})$ are then used to add the transitions to it. The induced game HA can be constructed from the PHA on a syntactical level, following the recipe discussed in Subsection 3.3.1. For most parts, this construction is straightforward. In the part g_τ, we have to check for states in which no further time progress is possible. Whether and how this can be done depends on the type of the timed dynamics under consideration. For instance, assume it is specified by a combination of differential (in)equations together with a set of of constraints on the flow, e.g. $-0.7T \leq \dot{T} \leq -0.3T \wedge t \leq 1$ as in the mode Check of the automaton in Figure 5.1. In this case, we can replace the guard by the border of the constraint, e.g. $t = 1$ in the example.

Using the induced game HA, we can construct a game abstraction of the original PHA.

Definition 5.19. *Let* $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds_{\mathrm{gind}} \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathrm{gind}(\mathcal{H}), \mathbf{A}, \mathbf{T})$ *be an abstraction of the induced game HA* $\mathrm{gind}(\mathcal{H})$ *of a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$. *We define the* induced game abstraction *as*

$$\mathrm{gabs}(\mathcal{H}, \mathcal{M}) = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}_{\mathrm{gabs}}),$$

where

- *for* $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ *with* $\mathrm{gind}(c) = \{(g_{\mathrm{gind}} \to u_{\mathrm{gind},1}), \ldots, (g_{\mathrm{gind}} \to u_{\mathrm{gind},n}), (\overline{g} \to \overline{u})\}$,
  - *for all* $\mathbf{z} \in \mathbf{A}$, $\mathbf{z}'_1, \ldots, \mathbf{z}'_n \in \mathbf{A}$ *with* $\mathbf{z}'_1 \in \mathcal{T}(\mathbf{z}, (g_{\mathrm{gind}} \to u_{\mathrm{gind},1})), \ldots, \mathbf{z}'_n \in \mathcal{T}(\mathbf{z}, (g_{\mathrm{gind}} \to u_{\mathrm{gind},n}))$ *we have* $[\mathbf{z}'_1 \mapsto p_1, \ldots, \mathbf{z}'_n \mapsto p_n] \in \mathcal{T}_{\mathrm{gabs}}(\mathbf{z}, c)$,
  - *for all* $\mathbf{z} \in \mathbf{A}$, $\mathbf{z}' \in \mathbf{A}$ *with* $\mathbf{z}' \in \mathcal{T}(\mathbf{z}, (\overline{g} \to \overline{u}))$ *we have* $[\mathbf{z}' \mapsto 1] \in \mathcal{T}_{\mathrm{gabs}}(\mathbf{z}, c)$,
- *for* $\mathbf{z}, \mathbf{z}' \in \mathbf{A}$ *with* $\mathbf{z}' \in \mathcal{T}(\mathbf{z}, \tau)$ *we have* $[\mathbf{z}' \mapsto 1] \in \mathcal{T}_{\mathrm{gabs}}(\mathbf{z}, \tau)$,
- *for* $\mathbf{z}, \mathbf{z}' \in \mathbf{A}$ *with* $\mathbf{z}' \in \mathcal{T}(\mathbf{z}, c_\tau)$ *we have* $[\mathbf{z}' \mapsto 1] \in \mathcal{T}_{\mathrm{gabs}}(\mathbf{z}, \tau)$.

As in Lemma 3.37 the abstraction of Definition 5.19 is indeed a valid game abstraction.

Lemma 5.20. *For a PHA* $\mathcal{H}$, *abstract state space* $\mathbf{A}$, *time restriction* $\mathbf{T}$ *and abstraction* $\mathcal{M} \in Abs(\mathrm{gind}(\mathcal{H}), \mathbf{A}, \mathbf{T})$, *we have* $\mathrm{gabs}(\mathcal{H}, \mathcal{M}) \in GAbs(\mathcal{H}, \mathbf{A}, \mathbf{T})$.

The result follows along the lines of Lemma 3.37. The presence of two players does not alter the proof structure, because the resulting additional nondeterminism only adds to the power of player env.

Corollary 5.21. *Consider a PHA* $\mathcal{H}$, *an abstraction* $\mathcal{M}'$ *of the induced game HA* $\mathrm{gind}(\mathcal{H})$, $\mathcal{M} \stackrel{\mathrm{def}}{=} \mathrm{gabs}(\mathcal{H}, \mathcal{M}')$ *and a mode* m_{Reach} *of* $\mathcal{H}$. *Let Reach be defined as in Theorem 3.31 and let Cmds be the set of probabilistic guarded commands of* $\mathcal{H}$. *Then*

$$\mathrm{val}^{+,-}_{\mathcal{H},m_{Reach}} \geq \mathrm{val}^{+,-}_{\mathcal{M},Reach} \text{ and } \mathrm{val}^{-,+}_{\mathcal{H},m_{Reach}} \leq \mathrm{val}^{-,+,Cmds}_{\mathcal{M},Reach \uplus \{\perp\}}.$$

This corollary follows from Theorem 5.16 and Lemma 5.20.

5.4 Algorithmic Consideration

As in Section 3.4, we are interested in deciding properties of the actual hybrid system using the abstraction we have computed. In the context of this chapter, this means that we want to use the game abstraction to decide whether it is possible to ensure bounds on the reachability probability in the actual system, and we are also interested in computing a controller which indeed steers the system to enforce these bounds.

In the following, we will restate some results about the computation of optimal values and strategies [HK66; Con93; FV96] in the abstract game. It is known that in finite two-player probabilistic reachability games both players can obtain their respective optimal value by simple strategies. We will sketch the Hoffman-Karp algorithm, which provides such strategies. We picked this algorithm, as it is a direct extension of Algorithm 3.3 which we used previously, and can also be combined with other extensions of our PHA setting (see Section 8.5 and Section 8.4).

We need the following definition to obtain a PA modified by a simple player-con strategy.

Definition 5.22. *Consider a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and a simple player-*con *strategy* $\sigma_{\text{con}} \colon S \to Act$*. The PA* induced *by* σ_{con} *in* $\mathcal{M}$ *is defined as*

$$\mathcal{M}_{\sigma_{\text{con}}} \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T}_{\sigma_{\text{con}}}),$$

where for $s \in S$ *and we have* $\mathcal{T}_{\sigma_{\text{con}}}(s, \sigma_{\text{con}}(s)) \stackrel{\text{def}}{=} \mathcal{T}(s, \sigma_{\text{con}}(s))$ *and* $\mathcal{T}_{\sigma_{\text{con}}}(s, a) \stackrel{\text{def}}{=} \emptyset$ *for* $a \in Act \setminus \{\sigma_{\text{con}}(s)\}$.

In Algorithm 5.1 we consider the case where player con maximises and player env minimises. The reverse case is similar. In the algorithm, maxMinReachInit($\mathcal{M}$) selects two arbitrary strategies for the two players. As in Section 3.4, $\text{vals}^{+,-}_{\mathcal{M},Reach}(s)$ is the mutually optimal value in case s were the initial state.

However, when minimising, we have to take some care with respect to the result of the algorithm, because the algorithm does not guarantee to return fair strategies. As discussed in Section 3.4, it is unlikely that this happens. To check for this problem, we can do a graph-based search on the MC induced by the joint schedulers. In case we come across nonfair behaviours of player con, we can remove the nonfair behaviour of the affected abstract states by removing all timed transitions. This way, we force player con to execute a command rather than letting time pass. In case this does not provide a fair strategy of player con, we repeat this step until the resulting strategy is fair. This method is guaranteed to terminate, because the abstraction is finite.

5.5 Synthesising Controllers

Next, we consider the problem of synthesising a controller for a PHA that makes optimal decisions so as to provably maintaining a certain threshold of the reachability probability. That is, if the objective is to maximise (minimise) the reachability probability, then the

input : finite PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$, state set $Reach \subseteq S$.
output: $(\mathrm{vals}^{+,-}_{\mathcal{M},Reach}, \sigma_{\mathrm{con}}, \sigma_{\mathrm{env}})$ with $\sigma_{\mathrm{con}} \in Strat^{\mathrm{con,simple}}_{\mathcal{M}}$ and $\sigma_{\mathrm{env}} \in Strat^{\mathrm{env,simple}}_{\mathcal{M}}$ and $\mathrm{vals}^{+,-}_{\mathcal{M},Reach} = \mathrm{vals}^{\mathrm{join}(\sigma_{\mathrm{con}},\sigma_{\mathrm{env}})}_{\mathcal{M},Reach} = \inf_{\sigma'_{\mathrm{env}} \in Strat^{\mathrm{env}}_{\mathcal{M}}} \mathrm{vals}^{\sigma_{\mathrm{con}},\sigma'_{\mathrm{env}}}_{\mathcal{M},Reach} = \sup_{\sigma'_{\mathrm{con}} \in Strat^{\mathrm{con}}_{\mathcal{M}}} \mathrm{vals}^{\sigma'_{\mathrm{con}},\sigma_{\mathrm{env}}}_{\mathcal{M},Reach}$.

```
(σ_con, σ_env) := maxMinReachInit(M)
v := vals_{M_{join(σ_con,σ_env)},Reach}
repeat
    v' := v
    forall the s ∈ S \ Reach do
        A(s) := argmax_{a∈Act, T(s,a)≠∅} Σ_{s'∈S} σ_env(s,a)(s')v(s')
        if σ_con(s) ∉ A(s) then choose σ_con(s) from A(s)
    (v, σ_env) := minReach(M_{σ_con}, Reach)        /* cf. Algorithm 3.3 */
until v = v'
return (v, σ_con, σ_env)
```

Algorithm 5.1: maxMinReach($\mathcal{M}$, *Reach*).

controller takes decisions so as to maximise (minimise) this probability under all possible reactions of the environment. The controller can base its decisions on the current state of the PHA, including the values of the continuous variables. In practice, such a controller would be implemented with the aid of sensors and timers.

We will construct controllers from the strategies of player con in the game abstraction of a PHA. For clarity, we will restrict to abstractions which are partitionings of the concrete state space. We will also assume that each $s \in \mathbf{z}$ can execute the command c chosen by a scheduler in $\mathbf{z}$, because if this is not the case, the transition to the sink state will cause the abstraction to be too bad to be useful. The quality of the controller of course depends on the precision of the abstraction computed. Conversely, the complexity of the implemented controller will increase when extracted from a finer abstraction. We begin by defining a relation between strategies in a game abstraction and in the concrete semantics.

Definition 5.23. *Let $\mathcal{H}$ be a PHA with semantics $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T})$, let $\mathbf{A}$ be an abstract state space, let $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$ be a time restriction and consider a game abstraction $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \mathbf{z}_0, Cmds \uplus \{\tau\}, \mathcal{T}) \in GAbs(\mathcal{H}, \mathbf{A}, \mathbf{T})$. Further, consider a simple player-*con *strategy $\sigma_{\mathrm{con}} \colon \mathbf{A} \to (Cmds \uplus \{\tau\})$ of the abstraction. We assume that*

- *$\mathbf{A}$ is a partitioning, and*
- *for all $\mathbf{z} \in \mathbf{A}$ and $s \in \mathbf{z}$ we have $\mathcal{T}(s, \sigma_{\mathrm{con}}(\mathbf{z})) \neq \emptyset$.*

We define the concretisation strategy *$\sigma'_{\mathrm{con}} : S \to (Cmds \uplus \mathbb{R}_{\geq 0})$ of σ_{con} as follows: for each $s = (m, v) \in S$, let $\mathbf{z} \in \mathbf{A}$ be the abstract state with $s \in \mathbf{z}$. If $\sigma_{\mathrm{con}}(\mathbf{z}) = a$ then*

$$\sigma'_{\mathrm{con}}(s) \stackrel{\mathrm{def}}{=} \begin{cases} a & \text{if } a \in Cmds, \\ \mathbf{t}_m(v) & \text{if } a = \tau. \end{cases}$$

We can use Definition 5.23 to guarantee bounds on the reachability probabilities in the model semantics.

Theorem 5.24. *Let $\mathcal{H}$ be a PHA, let $\mathbf{A}$ be an abstract state space, let $\mathbf{T}$ be a time restriction and consider a game abstraction $\mathcal{M} \in GAbs(\mathcal{H}, \mathbf{A}, \mathbf{T})$. Further, let $\sigma_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}}$ be a simple player-*con *strategy in $\mathcal{M}$ and let $\sigma'_{\text{con}} \in Strat^{\text{con}}_{[\![\mathcal{H},\mathbf{T}]\!]}$ be the concretisation strategy of σ_{con}. Then we have*

$$\inf_{\sigma'_{\text{env}} \in Strat^{\text{env}}_{[\![\mathcal{H},\mathbf{T}]\!]}} \text{val}^{\sigma'_{\text{con}},\sigma'_{\text{env}}}_{[\![\mathcal{H},\mathbf{T}]\!]} \geq \inf_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M}} \text{ and}$$

$$\sup_{\sigma'_{\text{env}} \in Strat^{\text{env}}_{[\![\mathcal{H},\mathbf{T}]\!]}} \text{val}^{\sigma'_{\text{con}},\sigma'_{\text{env}}}_{[\![\mathcal{H},\mathbf{T}]\!]} \leq \sup_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M}}.$$

Proof. We restrict to the case with the infimum, as the other one is similar. Assume $\mathcal{M} = (\mathbf{A} \uplus \{\bot\}, \mathbf{z}_0, Cmds \uplus \{\tau\}, \mathcal{T})$. Then let

$$\mathcal{M}_{\sigma_{\text{con}}} \stackrel{\text{def}}{=} (\mathbf{A} \uplus \{\bot\}, \mathbf{z}_0, Cmds \uplus \{\tau\}, \mathcal{T}_{\sigma_{\text{con}}}),$$

where for $a \in Cmds \uplus \{\tau\}$ and $\mathbf{z} \in \mathbf{A} \uplus \{\bot\}$ we have $\mathcal{T}_{\sigma_{\text{con}}}(s,a) \stackrel{\text{def}}{=} \mathcal{T}(s,a)$ if $\sigma_{\text{con}}(\mathbf{z}) = a$, and $\mathcal{T}_{\sigma_{\text{con}}}(s,a) \stackrel{\text{def}}{=} \emptyset$. Further, let $[\![\mathcal{M},\mathbf{T}]\!]_{\sigma'_{\text{con}}}$ be defined accordingly. We have

$$\begin{aligned}
&\inf_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M}} \\
&= \inf_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}_{\sigma_{\text{con}}}}} \text{val}^{\sigma_{\text{env}}}_{\mathcal{M}_{\sigma_{\text{con}}}} \\
&= \sup_{\sigma''_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}_{\sigma_{\text{con}}}}} \inf_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}_{\sigma_{\text{con}}}}} \text{val}^{\sigma''_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M}_{\sigma_{\text{con}}}} \\
&= \text{val}^{+,-}_{\mathcal{M}_{\sigma_{\text{con}}}},
\end{aligned} \tag{5.11}$$

and accordingly for $[\![\mathcal{H},\mathbf{T}]\!]$. In a similar way as the chain of lemmas used in Theorem 5.16, we can show that

$$\text{val}^{+,-}_{[\![\mathcal{H},\mathbf{T}]\!]_{\sigma_{\text{con}}}} \geq \text{val}^{+,-}_{\mathcal{M}_{\sigma_{\text{con}}}}. \tag{5.12}$$

From Equation 5.11 and Equation 5.12, we have

$$\inf_{\sigma'_{\text{env}} \in Strat^{\text{env}}_{[\![\mathcal{H},\mathbf{T}]\!]}} \text{val}^{\sigma'_{\text{con}},\sigma'_{\text{env}}}_{[\![\mathcal{H},\mathbf{T}]\!]} \geq \inf_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M}}.$$ □

This way, we can also implement a controller CONTROL sketched in Algorithm 5.2 which guarantees these bounds under the assumption that it can react fast enough. Firstly, in Line 1 the controller finds out about the current system configuration by executing the method CURRENTSTATE, which reads values from sensors. If the unsafe or safe mode m_{Reach} has not been reached yet (cf. Line 2), in Line 3 it checks in which abstract state the current configuration is contained. If there the player-con strategy in the abstraction chooses a certain guarded command to execute, it executes the guarded command using EXEC, which influences the environment using actuators. If the abstract controller chooses τ, it waits as long as the time restriction allows in Line 5 using WAIT, which can be implemented using simple timers. By executing this sequence of commands in a loop between lines 2 and 6, by Theorem 5.24 we can control the system in a way that it fulfils reachability probability bounds that we can compute as discussed in Section 5.4.

input : Partially-controllable PHA $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$, set of abstract states $\mathbf{A}$,time restriction $\mathbf{T} = \langle \mathbf{t}_m \rangle_{m \in M}$, $\sigma_{\text{con}} \in Strat_{\mathcal{M}}^{\text{con,simple}}$, $m_{Reach} \in M$.

```
1 (m, v) := currentState()
2 while m ≠ m_Reach do
3 |   find z ∈ A with (m, v) ∈ z
4 |   if σ_con(z) ∈ Cmds then exec(σ_con(z))
5 |   else wait(t_m(v))
6 |   (m, v) := currentState()
```

Algorithm 5.2: CONTROL($\mathcal{H}, \mathbf{A}, \mathbf{T}, \sigma_{\text{con}}, m_{Reach}$).

$\mathfrak{T}$	interval length 0.5			interval length 0.2		
	prob.	build (s)	states	prob.	build (s)	states
1	0.000	0	20	0.000	0	79
4	0.000	10	917	0.000	44	3590
5	0.000	14	1051	0.700	54	4066
10	0.910	81	4330	0.910	413	16773
15	0.973	50	3216	0.992	2578	53289
20	0.998	214	10676	0.999	1435	41313
25	0.999	160	8671	1.000	928	32864

Table 5.1: Thermostat results.

5.6 Case Study

We have implemented the abstraction developed in this chapter in an extension of our tool PROHVER. It is able to reconstruct the game abstraction and compute values in a similar way as Algorithm 5.1. We return to the thermostat example and consider the probability that a failure is detected (i.e. that Safe is entered) within time bound $\mathfrak{T}$. As in the original thermostat example, we do so by using a global timer c. We assume that the player con tries to maximise the probability to reach Safe. In turn, player env minimises in order to simulate the worst-case behaviour of the environment. This way, we obtain lower bounds for the maximal reachability probabilities the controller can enforce against a most malevolent environment.

Table 5.1 gives probability bounds and performance results for different values of $\mathfrak{T}$. Probability bounds obtained are rounded to 3 decimal places. We see that increasing analysis precision gives tighter probability bounds but requires more resources in general. Because of the way PHAVER computes abstractions however, increase in memory and time usage is not monotonic, as we have already observed and discussed in more detail in Chapter 2.

5.7 Related Work

For classical HAs which are partially controllable, there exist works for both specialised classes such as timed automata [Beh+07; AMP94], rectangular hybrid automata [HK99] as well as general HAs notions [LGS95; Asa+00].

Sproston [Spr11] discusses solution methods of partially controllable rectangular probabilistic HAs, which can also be used to handle ω-regular properties which are more complex than reachability.

Our abstraction methods for probabilistic HAs were heavily inspired by abstraction methods for the area of probabilistic systems with purely discrete state space [KNP06; WZ10; Wac11], in particular the *menu-based*, or *parallel abstraction*, by Wachter et al.

Abate et al. [Aba+08] have provided a solution method in the area of hybrid systems with a continuous state-space but discrete-time behaviour. In contrast to the setting of this chapter, they assume that the entire nondeterminism of the hybrid system is controllable, which means that player con only has to play against the random behaviour of the system. Their setting could however be extended to the case of two opposing players.

To the best of our knowledge, our approach of using solvers for classical HAs to synthesise controllers of PHAs has not been considered before.

5.8 Conclusion

We have discussed how we can extend our existing analysis framework to PHAs which are under our partial control. A number of related works exist. In our own setting, we have assumed that the controller is able to choose which actions to execute and how long to wait. We consider this model to be natural, given that usually a discrete controller, which might use stopwatches, steers a system operating in a continuous environment. As for the noncontrolled case of Chapter 3, we developed an abstraction scheme building on classical HAs, which we also have shown to be work in practice by applying it on a case study. Using these abstractions, we not only compute worst-case bounds which we can achieve by an according controller, but we can also use results obtained from the abstraction to synthesise controllers which control the actual system, thus to enforce probability bounds there. When proving the correctness, we could no longer use probabilistic simulations, because we are given two different types of nondeterminism. Instead, we used a chain of intermediate models, each weakening power of the controller player a bit further, to finally prove that the abstraction indeed overapproximates the concrete semantics. We had to take care to avoid having to introduce timed self loops, because in certain cases they might lead to very bad value bounds.

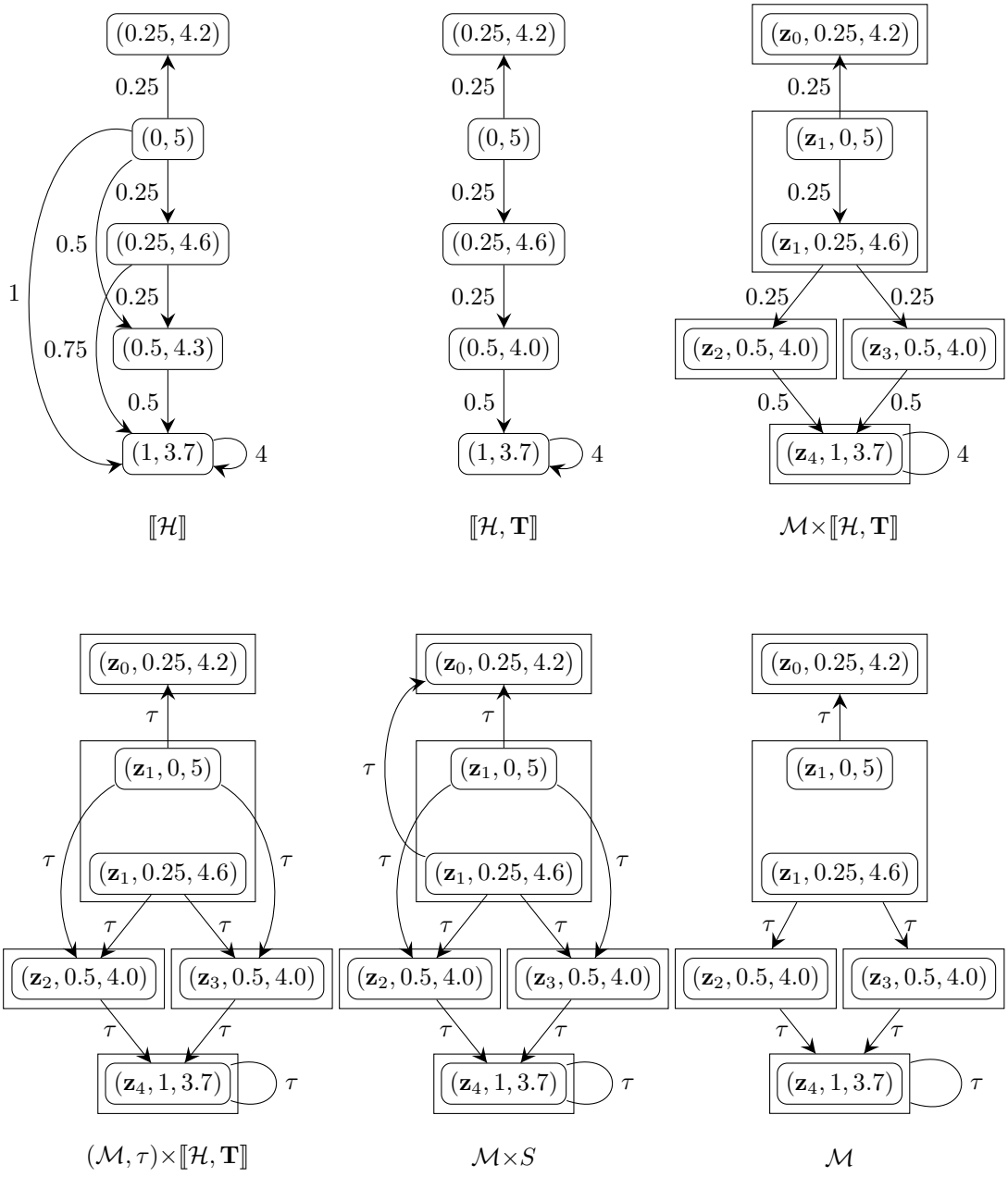


Figure 5.3: Sketch of game-based abstraction scheme.

6 Parameters

In this chapter, we consider probabilistic hybrid automata with parametric probabilities. Each instantiation of the parameters represents one concrete probabilistic hybrid automaton. We extend our analysis framework, so as to quickly obtain an overview of the behaviour of an entire family of models much faster, than if we had to do a separate analysis for each of them. It allows to establish the parameter instantiations for which a given property is guaranteed to hold, or to reason about optimal parameters of such system families.

This chapter is organised as follows: in Section 6.1, we extend PAs to parametric PAs which will be the semantics of parametric PHAs. Then, in Section 6.2, we describe parametric PHAs and define their semantics. The section Section 6.3 describes how we can obtain and work with abstractions of parametric PHAs. Section 6.4 will discuss how properties can be decided once the abstraction has been computed. In Section 6.5, we apply the methods developed here on parametric variants of the models of Section 3.5. Afterwards, in Section 6.6 we discuss related work. Section 6.7 concludes the chapter.

In contrast to the other main chapters, Section 6.4 about the algorithmic considerations will be the most extensive section of this chapter. The reason is that the extension of the models and abstractions of Chapter 3 is quite direct, but the subsequent analysis of abstract models is more intricate.

6.1 Parametric Probabilistic Models

In this section, we extend the PAs which formed the semantics of PHAs in Chapter 3 to parametric variants which can then form the semantics of parametric PHA. Firstly, we provide some basic definitions for the parameters and functions specifying probabilities.

Definition 6.1. *Consider a finite ordered set of* variables $V = \langle x_1, \ldots, x_n \rangle$. *With each variable x, we associate a closed interval* $\mathrm{range}(x) \stackrel{\text{def}}{=} [L_x, U_x]$ *specifying which values of x are valid. An* evaluation *v is a function $v \colon V \to \mathbb{R}$ respecting the variable ranges, that is for all $x \in V$ we have $v(x) \in [L_x, U_x]$. With $\mathit{Evals}(V)$ we denote the set of all evaluations on V. A* polynomial *$g \colon \mathbb{R}^n \to \mathbb{R}$ over V is a sum of monomials*

$$g(x_1, \ldots, x_n) = \sum_{i_1, \ldots, i_n} a_{i_1, \ldots, i_n} x_1^{i_1} \cdots x_n^{i_n},$$

where each $i_j \in \mathbb{N}$ *and each* $a_{i_1,\ldots,i_n} \in \mathbb{R}$. *A* rational function $f\colon Dom(f) \to \mathbb{R}$ *with* $Dom(f) \subseteq \mathbb{R}^n$ *is a fraction* $f(x_1,\ldots,x_n) = \frac{g_1(x_1,\ldots,x_n)}{g_2(x_1,\ldots,x_n)}$ *of two polynomials* g_1, g_2 *over* V. *We assume that numerator and denominator are in irreducible ratio form, so that* $Dom(f)$ *is maximal. Let* $Rat(V)$ *denote the set of all rational functions over* V. *Given* $f \in Rat(V)$ *and an evaluation* v *for which* $(v(x_1),\ldots,v(x_n)) \in Dom(f)$, *we let* $f\langle v\rangle \stackrel{\text{def}}{=} f(v(x_1),\ldots,v(x_n))$ *denote the rational number obtained by substituting each occurrence of* x_i *with* $v(x_i)$. *We identify* 0 *with the polynomial which is* 0 *for all evaluations.*

Each parameter thus has a certain range of values of interest. For instance, if x is a failure probability, the largest range would be from 0 to 1. Polynomials are always defined on the complete parameter space. We require rational functions to be free of common factors in numerator and denominator. For instance, instead of $f(x,y) = \frac{xy}{x}$, we would consider $f(x,y) = y$ where the x has been cancelled, which means that f is also defined for $x = 0$. Pointwise addition, subtraction, multiplication and division of two rational functions again yields a rational function, because of which we can use these operations on them in the following. We also identify constants with constant functions and write 0 instead of f if $f(\cdot) = 0$.

Assumption 6.2. *We remark that there are* $v \in Evals(V)$ *with* $(v(x_1),\ldots,v(x_n)) \notin Dom(f)$ *and in turn* $f\langle v\rangle$ *is not defined. In the following, when using* $f\langle v\rangle$ *we implicitly restrict to* v *for which this number is well-defined.*

With these preparations, we can now define parametric probability distributions.

Definition 6.3. *A* parametric probability distribution *over a finite set of parameters* A *is a function* $\mu\colon A \to Rat(V)$ *with* $\sum_{a\in A} \mu(a) = 1$ *and for all* $a \in A$ *and all evaluations* $v \in Evals(V)$ *we have* $\mu(a)\langle v\rangle \in [0,1]$. *With* $PDistr(A,V)$ *we denote the set of all finite parametric probability distributions with parameters* V *over* A. *The probability distribution* $\mu_v \in Distr(A)$ induced *by an evaluation* v *is defined so that* $\mu_v(a) \stackrel{\text{def}}{=} \mu(a)\langle v\rangle$ *for all* $a \in A$. *Given* $a_i \in A$ *and* $p_i \in Rat(V)$, $1 \le i \le n$ *for some* $n \ge 1$ *with* $p_i\langle v\rangle \in [0,1]$ *for* $v \in Evals(V)$ *and* $\sum_{i=1}^n p_i = 1$, *we let* $[a_1 \mapsto p_1,\ldots,a_n \mapsto p_n]$ *denote the parametric probability distribution that chooses* $a \in A$ *with probability* $\sum_{a_i=a} p_i$.
We say that a certain evaluation $v \in Evals(V)$ *is* valid *for such a parametric probability distribution if for each* $a \in A$, *either* $\mu_v(a) \neq 0$, *or for all* $v' \in Evals(V)$ *we have* $\mu_{v'}(a) = 0$. *By* $Evals(\mu)$ *we denote the set of all valid evaluations for* μ.

Parametric probability distributions thus allow to represent an uncountable number of induced nonparametric probability distributions at once. We usually consider valid evaluations of probability distributions. The analyses we will specify later on will guarantee correct results for only this kind of distributions. If necessary, we can however perform separate analyses to also obtain results for distributions which are not valid.
The following then defines the parametric PAs.

Definition 6.4. *A* parametric probabilistic automaton (PPA) *is a tuple*

$$\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V),$$

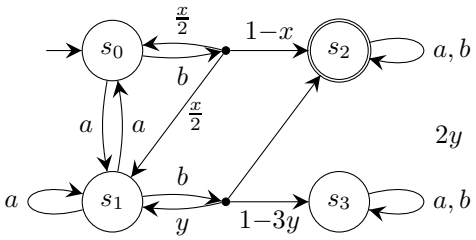


Figure 6.1: Parametric PA.

where S, $\overline{s}$ and Act are as for PAs (cf. Definition 3.3), V is a set of variables, and the transition matrix $\mathcal{T}\colon (S \times Act) \to 2^{PDistr(V,S)}$ *assigns sets of parametric probability distributions to pairs of states and actions. We require that for each $s \in S$ we have $\{a \in Act \mid \mathcal{T}(s) \neq \emptyset\} \neq \emptyset$. An evaluation $v \in Evals(V)$ is* valid *for $\mathcal{M}$ if if is valid for all parametric probability distributions occurring in $\mathcal{T}$. By $Evals(\mathcal{M})$ we define the set of all valid evaluations of $\mathcal{M}$.*

Example 6.5. *In Figure 6.1, we depict a finite example PPA $\mathcal{M} \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T}, V)$. We have $V \stackrel{\text{def}}{=} \{x, y\}$ and*

$$\mathcal{T}(s_0, b) \stackrel{\text{def}}{=} \{[s_0 \mapsto \frac{x}{2}, s_1 \mapsto \frac{x}{2}, s_2 \mapsto x]\} \text{ and } \mathcal{T}(s_1, b) \stackrel{\text{def}}{=} \{[s_1 \mapsto y, s_2 \mapsto 2y, s_3 \mapsto 1-3y]\}.$$

Thus, it makes sense to define the ranges of variables as

$$\text{range}(x) \stackrel{\text{def}}{=} [0, 1] \text{ and } \text{range}(y) \stackrel{\text{def}}{=} \left[0, \frac{1}{3}\right]. \qquad \triangle$$

PPAs represent an uncountable number of concrete PAs, which can be obtained using evaluations.

Definition 6.6. *Let $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$ be a PPA. The PA $\mathcal{M}_v$* induced *by an evaluation $v \in Evals(V)$ is defined as $\mathcal{M}_v \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T}_v)$. Here, the transition matrix $\mathcal{T}_v\colon (S \times Act) \to 2^{Distr(S)}$ is defined so that $\mathcal{T}_v(s, a) \stackrel{\text{def}}{=} \{\mu_v \mid \mu \in \mathcal{T}(s, a)\}$.*

Example 6.7. *Consider again the PPA $\mathcal{M}$ in Figure 6.1 and the evaluation v with $v(x) \stackrel{\text{def}}{=} 0.5$ and $v(y) \stackrel{\text{def}}{=} 0.3$. We have that $v \in Evals(\mathcal{M})$. The induced model $\mathcal{M}_v$ is the nonparametric PA already given in Figure 3.1. Further, consider the evaluation v' with $v'(x) \stackrel{\text{def}}{=} 1$ and $v'(y) \stackrel{\text{def}}{=} 0.3$. Then we have $v' \notin Evals(\mathcal{M})$: in $[s_0 \mapsto \frac{x}{2}, s_1 \mapsto \frac{x}{2}, s_2 \mapsto 1-x]$ we have $(1-x)\langle v' \rangle = 0$, but for instance $(1-x)\langle v \rangle \neq 0$.* $\triangle$

6.2 Parametric Hybrid Automata

In this section, we extend PHAs to involve parametric probability distributions. This way, we are able to specify an uncountably large family of PHAs with a single description. This is useful, for instance, if we are given a certain threshold $\mathfrak{p}$ as the largest

probability which is acceptable for a system failure, and we want to reason about the maximal distributions of a component failure which is still guaranteed to lead to a failure probability no larger than $\mathfrak{p}$.

A parametric PHA is defined as follows.

Definition 6.8. *A* parametric probabilistic hybrid automaton (PPHA) *is a tuple*

$$\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds, V),$$

where V is a set of variables and M, $k \in \mathbb{N}^+$, $\overline{m}$, $Post_m$ are as in the nonparametric case (cf. Definition 3.20). Each element of the finite set of parametric probabilistic guarded commands *$Cmds$ is of the form $g \to p_1 : u_1 + \ldots + p_n : u_n$, where*

- *$g \subseteq M \times \mathbb{R}^k$ is a* guard*,*
- *we have that $u_i \colon (M \times \mathbb{R}^k) \to 2^{M \times \mathbb{R}^k}$ is an* update *with $u_i(s) \neq \emptyset$ for $1 \leq i \leq n$ if $s \in g$, and*
- *for $1 \leq i \leq n$ we have $p_i \in Rat(V)$ and $\sum_{i=1}^{n} p_i = 1$.*

An evaluation $v \in Evals(V)$ is valid *for $(g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ if for all i with $1 \leq i \leq n$ we have $p_i\langle v \rangle \neq 0$ or if for all $v' \in Evals(V)$ $p_i\langle v' \rangle = 0$. An evaluation $v \in Evals(V)$ is* valid *for $\mathcal{H}$ if it is valid for all $c \in Cmds$. By $Evals(\mathcal{H})$ we denote the set of all evaluations which are valid for $\mathcal{H}$.*

The definition is thus as in Definition 3.20, with the exception that the probabilities in the guarded commands are parametric.

As stated, a PPHA represents an uncountably large family of nonparametric PHAs which are obtained by using a valid evaluation.

Definition 6.9. *The PHA* induced *by an evaluation $v \in Evals(V)$ in a PPHA $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds, V)$ is defined as $\mathcal{H}_v \stackrel{\text{def}}{=} (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_v)$ where*

$$Cmds_v \stackrel{\text{def}}{=} \{g \to p_1\langle v \rangle : u_1 + \ldots + p_n\langle v \rangle : u_n \mid (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds\}.$$

As for the PPAs before, we usually restrict to a set of valid evaluations for which we consider the induced PHA, because the later analysis method is only valid for this subset of evaluations. Evaluations within the range of variables which are not valid are usually pathological cases, which are usually not of interest. A common example is a failure probability of 0 or 1. If required however, one can perform analyses for these cases separately from the other ones.

Example 6.10. *In Figure 6.2 we depict a parametric version of the thermostat example of Figure 3.3. The continuous behaviour is the same as in the nonparametric case, as are most of the guarded commands. The main difference is the command c_{Ch} for which we now have*

$$c_{\text{Ch}} \stackrel{\text{def}}{=} (g \to (1 - x) : u_{\text{ChH}} + x : u_{\text{ChE}}),$$

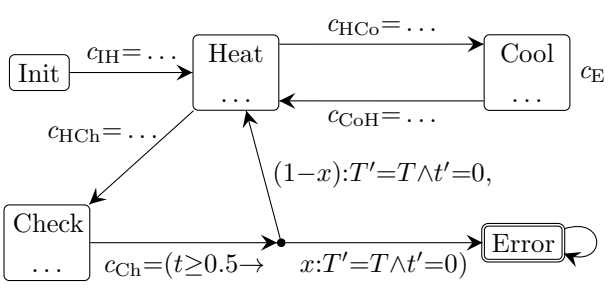


Figure 6.2: Parametric probabilistic hybrid automaton.

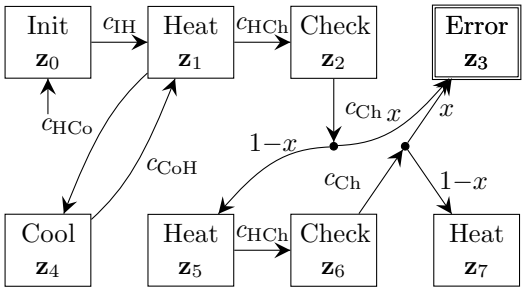


Figure 6.3: Abstraction for parametric probabilistic hybrid automaton.

with u_{ChH} *and* u_{ChE} *as in Example 2.15. We can define* $\mathrm{range}(x) \stackrel{\mathrm{def}}{=} [0,1]$. *The set of valid variable evaluations* $Evals(\mathcal{H})$ *however excludes* v' *and* v'' *with* $v'(x) \stackrel{\mathrm{def}}{=} 0$ *and* $v''(x) \stackrel{\mathrm{def}}{=} 1$*: both of them set either the probability to reach* Heat *or* Error *to* 0*, while there are other evaluations for which these values are not* 0.

The nonparametric PHA $\mathcal{H}_v$ *induced by* v *with* $v(x) \stackrel{\mathrm{def}}{=} 0.05$ *is then the previous nonparametric thermostat case study depicted in Figure 3.3.* △

The semantics of a PPHA is a PPA.

Definition 6.11. *The semantics* $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T}, V)$ *and time-restricted semantics* $[\![\mathcal{H}, \mathbf{T}]\!] = (S, \overline{s}, Act, \mathcal{T}', V)$ *of a PPHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds, V)$ *are defined as in the nonparametric case (cf. Definition 3.22), except that parametric probability distributions are used on the transitions.*

Example 6.12. *For the PHA* $\mathcal{H}$ *of Figure 6.2 we have* $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T})$ *and or* $s = (\mathrm{Check}, t, T, c) \in \{\mathrm{Check}\} \times \mathbb{R}^3$ *we have*

$$\mathcal{T}(s, c_{\mathrm{Ch}}) = \begin{cases} \emptyset & \text{if } t < 0.5, \\ \{[(\mathrm{Heat}, 0, T, c) \mapsto (1-x), (\mathrm{Error}, 0, T, c) \mapsto x]\} & \text{else.} \end{cases}$$

Other transitions stay as in Example 3.23, except that probabilities are now constant functions rather than constant numbers. △

6.3 Abstractions

As we target at practical means of bounding properties of our models, we also describe abstractions in the parametric case.

Definition 6.13. *The* abstractions $Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ *of a PPHA* $\mathcal{H}$ *are defined as in the nonparametric case (cf. Definition 3.30), except that parametric probability distributions are used.*

Example 6.14. *In Figure 6.3 we depict an abstraction of the PPHA of Figure 6.2. The abstraction is structurally equivalent to the one in Figure 3.4, the difference is that instead of probability distributions we now have parametric probability distributions.* △

With the following theorem, we can bound the values of the induced nonparametric PHAs of a PPHA by using the abstraction of the PPHA.

Theorem 6.15. *Consider a PPHA $\mathcal{H}$, an abstraction $\mathcal{M} \in Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ and an unsafe mode m_{Reach}. Let $Reach \stackrel{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{Reach}\}$. For each $v \in Evals(V)$ we have*

$$\mathrm{val}^{+}_{\mathcal{H}_v, m_{Reach}} \leq \mathrm{val}^{+}_{\mathcal{M}_v, Reach} \text{ and } \mathrm{val}^{-}_{\mathcal{H}_v, m_{Reach}} \geq \mathrm{val}^{-,Act}_{\mathcal{M}_v, Reach}.$$

Proof. $\mathcal{M}_v$ is an abstraction of $[\![\mathcal{H}, \mathbf{T}]\!]_v$, and thus the results follow from Theorem 3.31 and Theorem 3.32. □

Thus, after we have computed the abstraction of a PPHA, we can obtain a bound for a specific induced PHA by inserting the parameter values in the parametric abstraction and then performing the analysis of the concrete abstraction obtained this way. Thus, if we are interested of bounding the properties of the induced models of a limited number of evaluations, we can use this theorem to avoid having to recompute the abstraction for each of them separately.

6.3.1 Computing Abstractions

We can obtain induced HAs for PPHAs in the same way as in Definition 3.34 and also the induced abstractions as in Definition 3.35: in Definition 3.34, the probabilities are handled in a purely symbolic way, which means it does not matter whether they are real numbers or functions. The construction of Definition 3.35 can also be applied, because we have extended the construct $[a_1 \mapsto p_1, \ldots, a_n \mapsto p_n]$ to parametric probability distributions.

6.4 Algorithmic Consideration

With Theorem 6.15 we already have a means of using the abstraction to speed up the analysis of PPHAs, given that we are interested in a limited number of concrete induced models. In this subsection, we aim at developing a method to analyse an uncountably large number of concrete PHAs at once. For this, we will show how we can express reachability properties of PPHA abstractions in terms of rational functions, and synthesise regions, that is, multidimensional intervals of the variables for which the reachability probability respects a given bound. The basic idea is to extend the policy iteration algorithm Algorithm 3.3 of the nonparametric case to the parametric one.

As in Section 3.4, we need to consider models which have a probabilistic behaviour even without a scheduler.

input : finite PMC $\mathcal{M} = (S, \overline{s}, \mathcal{T}, V)$, state set $\mathit{Reach} \subseteq S$.
output: $\mathrm{vals}_{\mathcal{M},\mathit{Reach}}$ where for all $v \in \mathit{Evals}(\mathcal{M})$ and all $s \in S$ we have $\mathrm{vals}_{\mathcal{M},\mathit{Reach}}(s)\langle v\rangle = \mathrm{vals}_{\mathcal{M}_v,\mathit{Reach}}(s)$.

for $s \in S \setminus \mathit{Reach}$ *with* $\mathcal{T}(s)(s) \neq 1$ **do**
  for $(s_1, s_2) \in \mathrm{pre}_{\mathcal{M}}(s) \times \mathrm{post}_{\mathcal{M}}(s)$ *with* $s_1 \neq s \wedge s_2 \neq s$ **do**
    $\mathcal{T}(s_1)(s_2) := \mathcal{T}(s_1)(s_2) + \mathcal{T}(s_1)(s)\frac{1}{1-\mathcal{T}(s)(s)}\mathcal{T}(s)(s_2)$
  for $s' \in \mathrm{pre}_{\mathcal{M}}(s)$ *with* $s' \neq s$ **do** $\mathcal{T}(s')(s) := 0$
  for $s' \in \mathrm{post}_{\mathcal{M}}(s)$ *with* $s' \neq s$ **do** $\mathcal{T}(s)(s') := \frac{\mathcal{T}(s)(s')}{1-\mathcal{T}(s)(s)}$
  $\mathcal{T}(s)(s) := 0$
for $s \in \mathit{Reach}$ **do** $v(s) := 1$
for $s \in S \setminus \mathit{Reach}$ **do** $v(s) := \sum_{s' \in \mathit{Reach}} \mathcal{T}(s)(s')$
return v

Algorithm 6.1: reachParam($\mathcal{M}$, *Reach*).

Definition 6.16. *A* parametric Markov chain (PMC) *is a tuple*

$$\mathcal{M} = (S, \overline{s}, \mathcal{T}, V),$$

where S and $\overline{s}$ are as in Definition 3.40, V is a set of variables, and the transition matrix *$\mathcal{T}\colon S \to \mathit{PDistr}(V, S)$ assigns to each state a parametric distribution over successor states. We also define the adjacency graph in the same way as in the nonparametric case. That is, it is a graph $(S, \overline{s}, \mathcal{T}_{\mathrm{adj}})$ where we have $(s, s') \in \mathcal{T}_{\mathrm{adj}}$ if and only if $\mathcal{T}(s, s') \neq 0$ (where 0 is the constant function f with $f(\cdot) = 0$), and we define $\overline{\mathit{Reach}}$ accordingly.*

As for PPAs, we consider the nonparametric instantiations of PMCs.

Definition 6.17. *Let $\mathcal{M} = (S, \overline{s}, \mathcal{T}, V)$ be a PMC. The MC $\mathcal{M}_v$* induced *by an evaluation $v \in \mathit{Evals}(V)$ is defined as $\mathcal{M}_v \stackrel{\mathrm{def}}{=} (S, \overline{s}, \mathcal{T}_v)$. Here, the transition matrix $\mathcal{T}_v\colon S \to \mathit{Distr}(S)$ is given as $\mathcal{T}_v(s)(s') \stackrel{\mathrm{def}}{=} \mathcal{T}(s)(s')\langle v\rangle$.*

A simple scheduler then induces a PMC from a PPA.

Definition 6.18. *Let $\mathcal{M} = (S, \overline{s}, \mathit{Act}, \mathcal{T}, V)$ be a PPA. The PMC* induced *by a simple scheduler σ of $\mathcal{M}$ is defined as $\mathcal{M}_\sigma \stackrel{\mathrm{def}}{=} (S, \overline{s}, \mathcal{T}_\sigma, V)$, where for $s \in S$ we have $\mathcal{T}_\sigma(s) \stackrel{\mathrm{def}}{=} \mu$ if $\sigma(s) = (a, \mu)$.*

We define predecessors and successors of states in a PMC.

Definition 6.19. *Given a PMC $\mathcal{M} = (S, \overline{s}, \mathcal{T}, V)$, we define the set of* predecessors *and* successors *of a given state $s \in S$ as*

$$\mathrm{pre}_{\mathcal{M}}(s) \stackrel{\mathrm{def}}{=} \{s' \in S \mid \mathcal{T}(s')(s) \neq 0\} \text{ and } \mathrm{post}_{\mathcal{M}}(s) \stackrel{\mathrm{def}}{=} \{s' \in S \mid \mathcal{T}(s)(s') \neq 0\}.$$

In Algorithm 6.1, we give an algorithm which computes parametric reachability probabilities in PMCs. That is, the algorithm computes a function, so that instead of performing

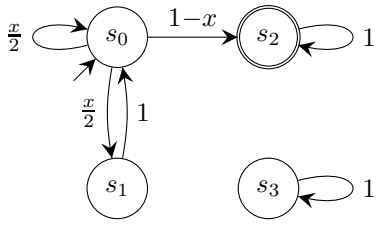


Figure 6.4: Parametric MC.

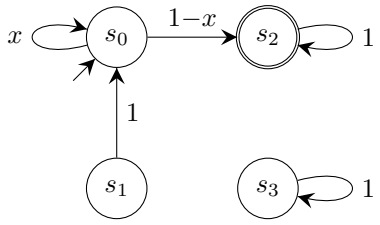


Figure 6.5: Handling s_1.

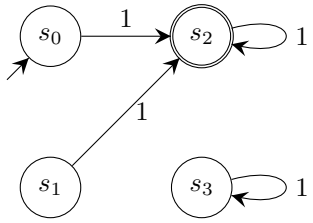


Figure 6.6: Handling s_0.

an analysis on the nonparametric MC induced by a certain evaluation v, we can insert the parameter values of v into the function to obtain the same result. The following states the correctness of the algorithm.

Lemma 6.20. *Let $\mathcal{M} = (S, \overline{s}, \mathcal{T}, V)$ be a PMC, consider $Reach \subseteq S$ and let $\text{vals}_{\mathcal{M},Reach}$ be the result of* reachParam$(\mathcal{M}, Reach)$ *of Algorithm 6.1. Then for all $s \in S$ and all $v \in Evals(\mathcal{M})$ we have*

$$\text{vals}_{\mathcal{M},Reach}(s)\langle v\rangle = \text{vals}_{\mathcal{M}_v,Reach}(s).$$

Proof. Let $\mathcal{M}_0 \stackrel{\text{def}}{=} \mathcal{M}$ and let $\mathcal{M}_i$ be the PMC resulting from the i-th execution of the main loop of Algorithm 6.1 in lines 2 to 6 for $0 \leq i \leq n$ with $n = |S \setminus Reach|$. We have $Evals(\mathcal{M}_i) \subseteq Evals(\mathcal{M}_{i+1})$ for all valid i. Further, for $v \in Evals(\mathcal{M})$ we have $\text{vals}_{\mathcal{M}_i,Reach}(s)\langle v\rangle = \text{vals}_{\mathcal{M}_{i+1},Reach}(s)\langle v\rangle$. This holds, because the unique solution of the equation system from Definition 3.41 for $(\mathcal{M}_i)_v$ is also the unique solution for $(\mathcal{M}_{i+1})_v$. Consider now $\mathcal{M}_n = (S, \overline{s}, \mathcal{T}_n, V)$, that is the model resulting after the complete executing of the main loop. The assignment of reachability probabilities for states of *Reach* in Line 7 is trivial. Consider a state s of $\mathcal{M}_n$ with $s \notin Reach$. Then for all $s' \in S$ with $\mathcal{T}_n(s)(s') \neq 0$ we have either $s' \in Reach$ or $\mathcal{T}_n(s', s') = 1$. Because of this, we have for all $v \in Evals(\mathcal{M}_n)$ that $\text{vals}_{(\mathcal{M}_n)_v,Reach}(s) = \sum_{s' \in Reach}(\mathcal{T}_n)_v(s, s')$. Thus, Line 8 assigns the correct values for each $s \notin Reach$. □

Algorithm 6.1 is a direct algorithm similar to Gaussian elimination, in contrast to Algorithm 3.1. We apply this algorithm, as it has been shown to perform well on models outside the area of hybrid systems [HHZ11b]. Iterative methods to compute reachability probabilities do not perform well, because the functions one would obtain this way can get very complex, as we discuss in the following example.

Example 6.21. *We demonstrate the method on the PMC given in Figure 6.4 with $Reach \stackrel{\text{def}}{=} \{s_2\}$. In the main loop, states s_2 and s_3 are ignored, because $s_2 \in Reach$ and s_3 has a self loop with probability 1. Assume thus that we handle s_1. The resulting PMC is given in Figure 6.5. We then handle s_0, thus to obtain the PMC given in Figure 6.6. The values we obtain are*

$$\text{vals}_{\mathcal{M},Reach}(s_0) = \text{vals}_{\mathcal{M},Reach}(s_1) = \text{vals}_{\mathcal{M},Reach}(s_2) = 1, \text{vals}_{\mathcal{M},Reach}(s_3) = 0.$$

We remark that for $v \notin Evals(\mathcal{M})$ with $v(x) \stackrel{\text{def}}{=} 1$ we cannot use this function to compute the reachability probabilities. We would obtain $\text{vals}_{\mathcal{M},Reach}(s_0)\langle v\rangle = 1$, whereas actually we have $\text{vals}_{\mathcal{M}_v,Reach}(s_0) = 0$. The problem here is that the set $\overline{Reach}$ is $\{s_3\}$ for valid evaluations, but we have $\{s_0, s_1, s_3\}$ for v.

If we had used value iteration in the form of Algorithm 3.1, the sequence of values obtained for $v(s_0)$ would be

$$
\begin{aligned}
&0\\
&1-x\\
&1-\frac{1}{2}x-\frac{1}{2}x^2\\
&1-\frac{3}{4}x^2-\frac{1}{4}x^3\\
&1-\frac{1}{4}x^2-\frac{5}{8}x^3-\frac{1}{8}x^4\\
&1-\frac{1}{2}x^3-\frac{7}{16}x^4-\frac{1}{16}x^5\\
&1-\frac{1}{8}x^3-\frac{9}{16}x^4-\frac{9}{32}x^5-\frac{1}{32}x^6\\
&\ldots.
\end{aligned}
$$

As we see, the length of the function increases in each iteration step. It would thus be hard to decide when the function is precise enough to stop the iteration. Also, the resulting inexact function would then be much more complicated than the exact one obtained by the Algorithm 6.1. △

For now, we have a method to symbolically compute reachability probabilities in PMCs. However, the semantics of PPHAs are PPAs, and the probabilistic behaviour of these models needs to be resolved by a scheduler. Even though we know that simple schedulers suffice to obtain minimal or maximal reachability probabilities for nonparametric PAs (cf. Section 3.4), it is possible that for different variable evaluations different schedulers provide minimal or maximal probabilities. Also, we are often interested in proving that probabilities are bounded by a value $\mathfrak{p}$. Considering the function of reachability probabilities, it is not clear which values maintain this bound. To solve this problem, we divide the space of possible parameter valuations into regions.

Definition 6.22. *A* region *is a high-dimensional rectangle*

$$r = \bigtimes_{x\in V}[l_x, u_x],$$

where for all $x \in V$ we have $[l_x, u_x] \subseteq \text{range}(x) = [L_x, U_x]$. A region represents those evaluations v with $v(x) \in [l_x, u_x]$ for all $x \in V$: in this case, we write $v \in r$, and also use set operations on regions. We define the centre *of a region $r = \bigtimes_{x\in V}[l_x, u_x]$ by $\text{centre}(r)(x) \stackrel{\text{def}}{=} \frac{l_x+u_x}{2}$ for $x \in V$.*

We let

$$\text{split}(r) \stackrel{\text{def}}{=} \text{split}(r, y) \text{ so that } y \in \text{argmax}_{x \in V} \frac{u_x - l_x}{U_x - L_x},$$

with

$$\text{split}(r, y) \stackrel{\text{def}}{=} \left\{ \bigtimes_{x \in V} [l_{x,l}, u_{x,l}], \bigtimes_{x \in V} [l_{x,u}, u_{x,u}] \right\},$$

where we have

$$[l_{y,l}, u_{y,l}] \stackrel{\text{def}}{=} \left[l_y, \frac{l_y + u_y}{2} \right] \text{ and } [l_{y,u}, u_{y,u}] \stackrel{\text{def}}{=} \left[\frac{l_y + u_y}{2}, u_y \right],$$

and for $x \neq y$ we have

$$[l_{x,l}, u_{x,l}] \stackrel{\text{def}}{=} [l_{x,u}, u_{x,u}] \stackrel{\text{def}}{=} [l_x, u_x].$$

We define the volume μ *of a region $r = \bigtimes_{x \in V} [l_x, u_x]$ as $\mu(r) \stackrel{\text{def}}{=} \prod_{x \in V} \frac{u_x - l_x}{U_x - L_x}$. For a set $K = \{r_1, \ldots, r_n\}$ of regions, we define $\mu(K) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \mu(r_i)$.*

A region is thus a multidimensional interval in the space of parameter valuations. The centre of a region is the evaluation in middle of the region. We can use split to divide a region into two smaller ones, and the volume is the relative length of the sides of a region.

To decide properties on regions, we can assume that we are given a semi-decision procedure to prove inequations over them.

Definition 6.23. *A* semi-decision procedure *is a tool deciding the validity of formulae for a given region. Consider a predicate constraint $\stackrel{\text{def}}{=} f \bowtie q$ where f is a rational function over the variables in V, $\bowtie \in \{<, \leq, \geq, >\}$ and $q \in \mathbb{R}$. Let r be a given region. For an evaluation v, with constraint$\langle v \rangle$ we denote $f\langle v \rangle \bowtie q$, i.e., the constraint obtained under the valuation v. We assume that we are given a semi-decision procedure* CHECK$(constraint, r)$ *which*

- *returns true only if for all $v \in r$ we have that constraint$\langle v \rangle$ is true, and*
- *returns false in case this does not hold or the result cannot be decided.*

A number of different semi-decision procedures exist [Rat06; Frä+07; PJ09; Han09]. Some of them work approximately, that is they cannot guarantee correctness in all cases, while others are precise. The definition allows CHECK to return *false* even if all evaluations indeed fulfil the property, as many semi-decision procedures have this property, that is they do not feature completeness.

Using CHECK, we can easily check whether an entire region fulfils a certain probability bound obtained from Algorithm 6.1 when given a PMC and set *Reach*. Algorithm 6.2 targets at a similar problem for PPAs. Here, we also have to find the minimising and maximising scheduler, in case there is such a scheduler for the entire region, and prove that it is indeed minimal or maximal for the entire region. The algorithm handles the minimising case, but the maximising one is similar. The following lemma states the correctness of the algorithm.

input : finite PPA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$, state set $Reach \subseteq S$, region r.
output: either ? or $(\text{vals}^{-}_{\mathcal{M},Reach}, \sigma)$ with $\sigma \in Sched^{\text{simple}}_{\mathcal{M}}$ and for all $v \in Evals(\mathcal{M}) \cap r$ and $s \in S$ we have $\text{vals}^{-}_{\mathcal{M},Reach}(s)\langle v\rangle = \text{vals}^{-}_{\mathcal{M}_v,Reach}(s) = \text{vals}^{\sigma}_{\mathcal{M}_v,Reach}(s)$.

1 $c := \text{centre}(r)$
2 **if** $c \notin Evals(\mathcal{M})$ **then return** ?
3 $(\_, \sigma) :=$ MINREACH$(\mathcal{M}_c, Reach)$ `/* cf. Algorithm 3.3 */`
4 $v :=$ PARAMREACHPROB$(\mathcal{M}_\sigma, Reach)$ `/* cf. Algorithm 6.1 */`
5 *valid* := *true*
6 **forall the** $s \in S \setminus Reach, a \in Act, \mu \in \mathcal{T}(s,a)$ **do**
7 $\quad v_{\text{cmp}}(s) := \sum_{s' \in S} \mu(s')v(s')$
8 $\quad$ *valid* := *valid* $\wedge$ CHECK$(v(s) \leq v_{\text{cmp}}(s), r)$ `/* cf. Definition 6.23 */`
9 **if** *valid* **then return** (v, σ)
10 **else return** ?

Algorithm 6.2: MINREACHPARAMREGION$(\mathcal{M}, Reach, r)$.

Lemma 6.24. *Let $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$ be a PPA, consider $Reach \subseteq S$ and let $(\text{vals}^{-}_{\mathcal{M},Reach}, \sigma)$ be the result of* MINREACHPARAMREGION$(\mathcal{M}, Reach, r)$ *of Algorithm 6.2. Then for all $s \in S$ and all $v \in Evals(\mathcal{M}) \cap r$ we have*

$$\text{vals}^{-}_{\mathcal{M},Reach}(s)\langle v\rangle = \text{vals}^{-}_{\mathcal{M}_v,Reach}(s) = \text{vals}^{\sigma}_{\mathcal{M}_v,Reach}(s).$$

Proof. From the first three lines of Algorithm 6.2 we obtain a scheduler where $\overline{Reach}$ of Lemma 3.44 agrees for all $v \in Evals(\mathcal{M})$. In addition, it is minimal for the centre of the region. By Lemma 6.20, for the value v we obtain in Line 4 we have $v(s)\langle v\rangle = \text{vals}_{\mathcal{M}_\sigma,Reach}$. In lines 5 to 8 we then prove the optimality equation from Lemma 3.44 for all valid evaluations of the region at the same time. □

Example 6.25. *Consider the abstraction $\mathcal{M}$ of the thermostat example from Figure 6.3 and the region r with $r(x) \stackrel{\text{def}}{=} [0,1]$. We want to compute the probability to reach $\mathbf{z}_3$. We have $c = \text{centre}(r)(x) = 0.5$, and the scheduler we obtain from $\mathcal{M}_c$ is σ with*

$$\sigma(\mathbf{z}_0) = (c_{\text{IH}}, [\mathbf{z}_1 \mapsto 1]), \quad \sigma(\mathbf{z}_1) = (c_{\text{HCh}}, [\mathbf{z}_2 \mapsto 1]), \quad \sigma(\mathbf{z}_2) = (c_{\text{Ch}}, [\mathbf{z}_5 \mapsto 0.5, \mathbf{z}_3 \mapsto 0.5]),$$
$$\sigma(\mathbf{z}_4) = (c_{\text{CoH}}, [\mathbf{z}_1 \mapsto 1]), \quad \sigma(\mathbf{z}_5) = (c_{\text{HCh}}, [\mathbf{z}_6 \mapsto 1]), \quad \sigma(\mathbf{z}_6) = (c_{\text{Ch}}, [\mathbf{z}_5 \mapsto 0.5, \mathbf{z}_3 \mapsto 0.5]).$$

The parametric reachability probabilities with this scheduler are

$$v(\mathbf{z}_0) = v(\mathbf{z}_1) = v(\mathbf{z}_2) = v(\mathbf{z}_4) = x + (1-x)x,$$
$$v(\mathbf{z}_5) = v(\mathbf{z}_6) = x, v(\mathbf{z}_3) = 1, v(\mathbf{z}_7) = 0.$$

It turns out that in the loop starting in Line 6 the values v_{cmp} equal those of $v(s)$, so that the calls to CHECK *return true. The algorithm thus returns $v(s)$.* △

Algorithm 6.2 allows to compute the optimal values and scheduler for a given region. With Algorithm 6.3, we can find functions representing minimal or maximal values for

input : finite PPA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$, state set $Reach \subseteq S$, area tolerance ε.
output: $\text{vals}^{-}_{\mathcal{M},Reach} = \{(r_1, b_1, \sigma_1), \ldots, (r_n, b_n, \sigma_n)\}$ with $\mu(\{r_1, \ldots, r_n\}) \geq 1 - \varepsilon$ and for all (r_i, b_i, σ_i) we have $\sigma_i \in Sched^{\text{simple}}_{\mathcal{M}}$ and for $v \in Evals(\mathcal{M}) \cap r_i$ and $s \in S$ it is $b_i(s)\langle v \rangle = \text{vals}^{-}_{\mathcal{M}_v,Reach}(s) = \text{vals}^{\sigma_i}_{\mathcal{M}_v,Reach}(s)$.

1 $unprocessed := \{\bigtimes_{x \in V} \text{range}(x)\}$
2 $result := \emptyset$
3 **while** $\mu(unprocessed) \geq \varepsilon$ **do**
4 choose one largest $r \in unprocessed$
5 $unprocessed := unprocessed \setminus \{r\}$
6 $resRegion :=$ minReachParamRegion$(\mathcal{M}, Reach, r)$
`/* cf. Algorithm 6.2 */`
7 **if** $resRegion =?$ **then** $unprocessed := unprocessed \cup \text{split}(r)$
8 **else** $result := result \cup \{(r, v, \sigma)\}$ with $(v, \sigma) \stackrel{\text{def}}{=} resRegion$
9 **return** $result$

Algorithm 6.3: minReachParam$(\mathcal{M}, Reach, \varepsilon)$.

a significant amount of the parameter space by dividing it into a number of regions. It is in general not possible to provide results for the complete parameter space. Thus, we stop when the volume of the remaining area is below a certain precision.

Theorem 6.26. *Let $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$ be a PPA, consider $Reach \subseteq S$, and a precision $\varepsilon \in (0, 1)$. Further, let $\text{vals}^{-}_{\mathcal{M},Reach} = \{(r_1, b_1, \sigma_1), \ldots, (r_n, b_n, \sigma_n)\}$ be the result of* minReachParam$(\mathcal{M}, Reach, \varepsilon)$ *of Algorithm 6.3. Then we have $\mu(\{r_1, \ldots, r_n\}) > 1 - \varepsilon$ and for all (r_i, b_i, σ_i), $v \in Evals(\mathcal{M}) \cap r_i$ and $s \in S$ we have*

$$b_i(s)\langle v \rangle = \text{vals}^{-}_{\mathcal{M}_v,Reach}(s) = \text{vals}^{\sigma_i}_{\mathcal{M}_v,Reach}(s).$$

Proof. It is $\mu(\{r_1, \ldots, r_n\}) > 1 - \varepsilon$, because the loop starting in Line 3 terminates if $\mu(unprocessed) < \varepsilon$, and we have $\mu(unprocessed) + \mu(\{r_1, \ldots, r_m\}) = 1$ where $\{r_1, \ldots, r_m\}$ are the regions after the m-th iteration of the loop. The correctness of the (r_i, b_i, σ_i) follows from Lemma 6.24. □

Because check is defined as a semi-decision procedure, we can not guarantee termination of Algorithm 6.3. If we are however given a decision procedure, we can indeed guarantee termination.

Theorem 6.27. *Assume that* check *is implemented in such a way that instead of the requirements in Definition 6.23 we have for* check$(constraint, r)$ *that it*

- *returns true if and only if for all $v \in r$ we have that $constraint\langle v \rangle$ is true.*

Then minReachParam *terminates for all valid inputs.*

Proof. Assume that the procedure is called as minReachParam$(\mathcal{M}, Reach, \varepsilon)$ with $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$. The termination of Algorithm 6.1 and Algorithm 6.2 is clear,

because in each of them the number of iterations in the loops is bounded by the number of states and transitions in $\mathcal{M}$.

Rational functions g are continuous in $Dom(g)$, and thus for each ε we can find a set A of regions which only overlap at the borders and cover the complete parameter space, and we have for

$$B \stackrel{\text{def}}{=} \{r \mid r \in A \wedge \exists v, v' \in A.\ g\langle v\rangle \leq 0 \wedge g\langle v'\rangle > 0\}$$

that

$$\mu(B) < \varepsilon,$$

which intuitively means that the "measure" of the points where the function is 0 is 0. From this, we can derive a corresponding property for v for which $g\langle v\rangle$ is not defined, and we also obtain a corresponding result for the $v \notin Evals(\mathcal{M})$. We can extend this result to multiple rational functions and thus we can extend the optimality equation of Lemma 3.44 to the parametric case:

$$B \stackrel{\text{def}}{=} \left\{ r \mid \neg\forall s, v \in r.\ \mathrm{vals}^{\sigma}_{\mathcal{M},Reach}(s)\langle v\rangle = \min_{\substack{a\in Act,\\ \mu\in\mathcal{T}(s,a)}} \left(\sum_{s'\in S} \mu(s')\mathrm{vals}^{\sigma}_{\mathcal{M},Reach}(s')\right)\langle v\rangle \right\}.$$

We can also add regions with centre$(r) \notin Evals(r)$ to the above set B because also the measure of these evaluations is 0.

If we apply Algorithm 6.2 to each such region of A, by Theorem 6.26 and because CHECK is assumed to be a complete decision procedure, we will obtain ? if and only if $r \in B$. Algorithm 6.3 splits region until *unprocessed* equals such a B with small enough measure, which means, that by the condition at the main loop in Line 3, it will finally terminate. □

Example 6.28. *We reconsider the thermostat example of Figure 6.2, the region r with $r(x) = [0, 1]$ and apply the variant of Algorithm 6.3 to maximise the probability. As seen from Example 6.21, in this case it is possible to handle the entire region at once, and we obtain a singleton result set.* △

Using Algorithm 6.3, we now have an effective method to bound the reachability values of all induced PHAs of a PPHA.

Corollary 6.29. *Consider a PPHA $\mathcal{H}$, an abstraction $\mathcal{M} \in Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ and an unsafe mode m_{Reach}. Let Reach $\stackrel{\text{def}}{=} \{(m, \zeta) \in \mathbf{A} \mid m = m_{Reach}\}$. Further, let $\mathrm{vals}^{-}_{\mathcal{M},Reach} = \{(r_1, b_1, \sigma_1), \ldots, (r_n, b_n, \sigma_n)\}$ be the result of* minReachParam$(\mathcal{M}, Reach, \varepsilon)$ *given in Algorithm 6.3. Then for all (r_i, b_i, σ_i), $v \in Evals(\mathcal{M}) \cap r_i$ and $s \in S$ we have*

$$\mathrm{vals}^{-}_{\mathcal{H}_v,m_{Reach}} \geq b_i(s)\langle v\rangle,$$

and accordingly for the maximising variant maxReachParam *of the algorithm.*

The corollary follows from Theorem 6.15 and Theorem 6.26.

As stated, to prove that reachability probabilities in a certain region are bounded by $\mathfrak{p}$, we can use CHECK. It is easy to implement an extension of Algorithm 6.3 which also proves bounds on regions, and splits them as necessary in case the bound is not fulfilled for an entire region.

6.5 Case Studies

In this section, we consider parametric variants of the case studies of Section 3.5. For each case study, we describe which probabilities we have parametrised. Results were obtained using our tool PARAM [Hah+10], which implements the algorithms of Section 6.4. As before, experiments were run on an Intel(R) Core(TM)2 Duo CPU with 2.67 GHz and 4 GB RAM. However, as we used the same settings in PHAVER as in Section 3.5, we do not restate the time to obtain the abstraction and number of abstract states, because they are the same as in the corresponding nonparametric models of the former section. We were able to obtain results for parametric versions of all case studies of Chapter 3, of which we state a few in the following.

6.5.1 Thermostat

Consider the parametrised thermostat model of Figure 6.2 (cf. Subsection 3.5.1). Due to the structure of the model, the maximal probability to reach Error within $\mathfrak{T}$ can always be stated as

$$(1-x)^n x$$

for some n depending on the time bound $\mathfrak{T}$: each time the automaton executes the guarded command in Check, there is a chance of x to move to Error, while the execution continues with probability $1-x$. Later on, if time has not yet run out, one might have another chance of reaching Error. Depending on the time, there is a maximal number of chances n to perform the guarded command in Check. Thus, the probability to finally reach Error can be described by a geometric distribution. If we insert the $x = 0.05$ into the corresponding functions, we obtain the same values as in Table 3.3.

6.5.2 Water Level Control

We apply our method on a parametrised version of the water level control case study (cf. Subsection 3.5.3). We denote the probability that we obtain a delay of 3 rather than 2 when filling the tank by x, and the probability that we have a delay of 3 instead of 2 when draining the tank is denoted by y.

In Figure 6.7, we depict bounds for the reachability probabilities dependent on x and y for the different time bounds $\mathfrak{T}$ already used in Table 3.5. We restrict to a maximal value of x and y of $\frac{1}{5}$. As expected, with an increasing time bound or failure probabilities the probability bounds increase. For this case study, there is a single maximising function and we did not have to divide the parameter space to take into account valuation-dependent scheduler decisions.

We ask whether the probability to reach Error can be bounded by $\mathfrak{p} = 0.3$. In Figure 6.8, we divide the parameter space into regions, where for each of them in the abstraction the property holds or does not hold. Thus, in the white area the property is guaranteed to hold, also in the concrete model semantics. In the black boxes, the property does

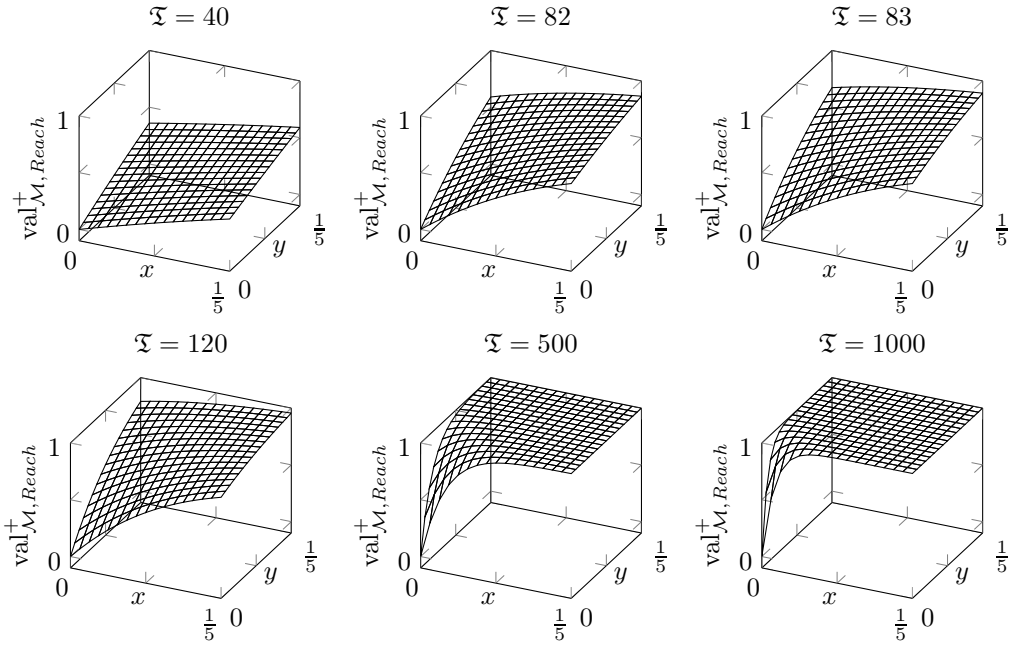


Figure 6.7: Bound for reachability probability in water level control dependent on failure probabilities.

not hold in the abstract model, but it might still hold in the concrete model semantics. The grey areas are mixed and thus inconclusive. The border between the black and the white area is a curved form, and thus we cannot cover the entire parameter space by rectangles. Because of this, we chose a tolerance of 0.03 for the maximal amount of grey regions.

6.6 Related Work

There exist a number of related works in the area of parametric HAs, which however do not consider parametrised probability distributions, but rather parameters on the timed or discrete behaviour. For instance, they allow to specify an invariant ($x \leq b$), where x is a continuous model variable, whereas b is a parameter. The goal is then to find the parameter valuations in which certain properties are fulfilled, in particular safety. Different methods exist for timed automata [And+09; BT09], a subclass of linear HAs [DISS11] and for linear and affine HAs [FK11] and have found successful applications, e.g. [PQ08].

Our initial works in this area [Hah+10; HHZ11a; HHZ11b] was concerned with models given in a variant of the guarded command language of Prism [KNP11], outside the area of HAs. The method of Algorithm 6.1 was originally inspired by a state-elimination approach by Daws [Daw04], which in turn builds on results in the context of regular languages. A similar method has also been used to generate counterexamples for properties

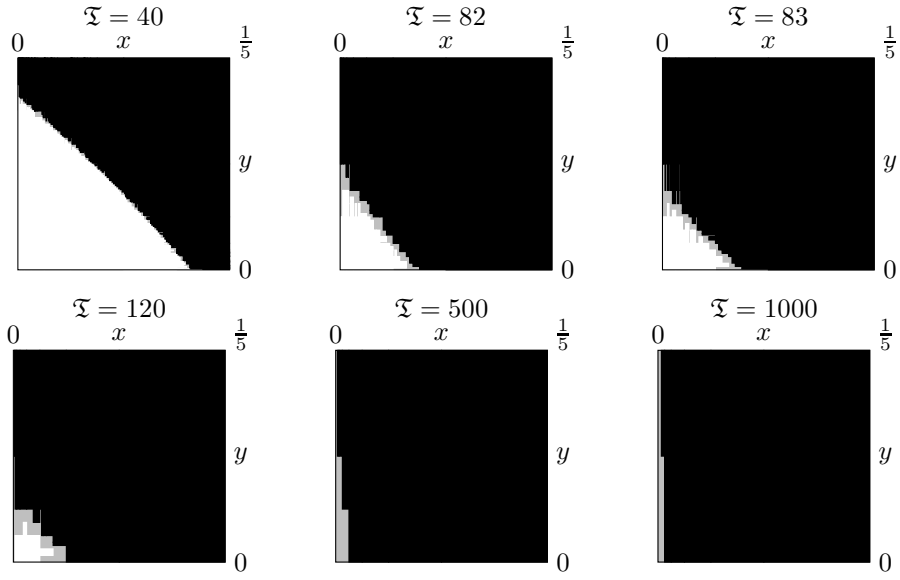


Figure 6.8: Regions for which a reachability probability below 0.3 can or cannot be guaranteed.

of Markov models [Ábr+10]. For continuous-time Markov chains, a similar method dividing the parameter space into regions has been considered by Han [Han09]. We are the first ones to apply this kind of parameter analysis to parametric PHAs models.

A closely related work is due to Fribourg and André [FA09]: for a given PPA and valuation of the parameters, they compute a scheduler for this instantiation which is optimal for a certain property. Afterwards, they compute the set of parameter evaluations for which the scheduler is still optimal. Compared to their work, we do not have a fixed scheduler a priori, but use different optimising schedulers for different regions in case this is necessary. Also, they do not consider parametric probabilities, but parametric reward values (see also Chapter 7), and concentrate on reward-based properties.

The related area of *parameter estimation* targets at finding the parameters of differential equations or MCs which fit best to a given set of measurements which the system is required to be consistent with [BBC08; MW12].

6.7 Conclusion

We have introduced parametric PHAs, in which probabilities are not constant but depend on a number of model parameters. As the semantic basis, we have introduced PPAs. We have discussed restrictions to a certain subset of variable evaluations, which in turn induce nonparametric PAs. Then, we moved on to the high-level model where we have extended the probabilistic guarded commands with parameters, matched by the parametric probability distributions of the previous extension of PAs. Next, we defined abstractions of PPHAs in terms of PPAs. A relevant part of the chapter was

then involved with developing a method to find optimal schedulers of entire regions of parameters, and how to synthesise regions of parameters which fulfil certain properties. For this, we extended the dynamic programming approach previously discussed in the context of Algorithm 3.3 to PPHAs. Doing so allowed us to end up with an efficient means to analyse an entire class of PHAs at once, rather than just a single specific model, and synthesise the set of concrete models satisfying a given property. We then discussed the related works, and applied the model on a number of parametric versions of our case studies, thus to show its practical applicability.

This chapter was mainly involved with algorithmic considerations to analyse the parametric abstractions, while the computation of the abstractions was almost the same as for nonparametric models. Because of this, considerations about the fairness were not necessary here, as they have already been handled for the nonparametric models.

7 Rewards

In this chapter, we decorate our probabilistic hybrid automata with rewards, which are costs or bonuses, and discuss how our solution method can be extended to handle reward-based properties. Doing so allows us to reason about the time until a system terminates, the long-run cost of its operation, percentage of the time in an operational state, time until stabilisation, and many other properties of interest.

This chapter is organised as follows: in Section 7.1 we equip probabilistic automata with reward structures and discuss the properties we can express this way. In Section 7.2, we extend probabilistic hybrid automata with rewards, and show how these are mapped to reward structures of the semantics. Section 7.3 then discusses how we can extend our abstraction technique of Section 3.3 to obtain guarantees on the values of the extended properties we have defined. Section 7.4 will discuss how properties can be decided once the abstraction has been computed. Then in Section 7.5 we apply the abstraction method to compute reward-based properties of two of our case studies. Section 7.7 concludes the chapter.

7.1 Rewards for Probabilistic Automata

In this section, we will extend the PAs with rewards and discuss reward-based properties. As these properties are expected values, we firstly have to define the notion of expectation.

7.1.1 Stochastic Recap

Consider a measurable space (Ω, Σ). The *expectation* E_μ with respect to a probability measure $\mu\colon \Sigma \to [0,1]$ for a Σ-$\mathcal{B}(\mathbb{R})$-measurable function $X\colon \Omega \to \mathbb{R}$ is defined as

$$E_\mu(X) \stackrel{\text{def}}{=} \int_\Omega X(\omega)\,\mu(\mathrm{d}\omega).$$

Given a path probability measure $Pr_{\mathcal{M},\sigma}\colon \Sigma_{\mathcal{M}} \to [0,1]$, we abbreviate

$$E_{\mathcal{M},\sigma} \stackrel{\text{def}}{=} E_{Pr_{\mathcal{M},\sigma}}.$$

7.1.2 Reward Structures and Properties

We equip our PAs with reward structures. We remark that rew_{num} and rew_{den} of the same reward structure rew in the following definition are *not* meant to denote lower or upper bounds on the specification of rewards. Instead, we will later on use them to denote pairs of reward and time.

Definition 7.1. *A* reward structure *for a PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ is a pair $(rew_{\text{num}}, rew_{\text{den}})$ of two functions*

$$rew_{\text{num}}, rew_{\text{den}} \colon (S \times Act \times Distr(S)) \to \mathbb{R}_{\geq 0}.$$

If for all $s \in S$ and $\mu, \mu' \in Distr(S)$ we have $rew_{\text{num}}(s, a, \mu) = rew_{\text{num}}(s, a, \mu')$ and $rew_{\text{den}}(s, a, \mu) = rew_{\text{den}}(s, a, \mu')$, we call this a simple *reward structure and interpret as a pair of functions*

$$rew_{\text{num}}, rew_{\text{den}} \colon (S \times Act) \to \mathbb{R}_{\geq 0}.$$

Given a simple reward structure $(rew_{\text{num}}, rew_{\text{den}})$, we say that it is affine*, if for all $a \in Act$ there are $mul_a \in \mathbb{R}_{\geq 0}$ and $add_a \in \mathbb{R}_{\geq 0}$, where for all $s \in S$ with $\mathcal{T}(s, a) \neq \emptyset$ we have*

$$rew_{\text{num}}(s, a) = mul_a rew_{\text{den}}(s, a) + add_a.$$

We will use reward structures $rew = (rew_{\text{num}}, rew_{\text{den}})$ to specify two different reward-based properties of PAs. For the definition of one of them, we will use both the functions rew_{num} and rew_{den}, whereas for the other one we will only need rew_{num}.

Example 7.2. *In Figure 7.1 we depict a PA along with a reward structure $rew \stackrel{\text{def}}{=} (rew_{\text{num}}, rew_{\text{den}})$. We have*

$$rew_{\text{num}}(s_1, a, [s_1 \mapsto 1]) \stackrel{\text{def}}{=} 0, rew_{\text{den}}(s_1, a, [s_1 \mapsto 1]) \stackrel{\text{def}}{=} 0,$$
$$rew_{\text{num}}(s_1, a, [s_0 \mapsto 1]) \stackrel{\text{def}}{=} 4, rew_{\text{den}}(s_1, a, [s_0 \mapsto 1]) \stackrel{\text{def}}{=} 2, \ldots.$$

Thus, rew is not simple.

Now, consider the PA of Figure 7.2. The reward structure of this PA is indeed simple. Also, rew is affine, where for action b we have the factors

$$mul_b = 2 \text{ and } add_b = 1. \qquad \triangle$$

We can then define properties based on these reward structures.

Definition 7.3. *Given a PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ together with a reward structure $rew = (rew_{\text{num}}, rew_{\text{den}})$ and a fair scheduler σ, the* accumulated reward *is defined as*

$$\text{val}^{\sigma}_{\mathcal{M}, rew, \text{lra}} \stackrel{\text{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty} \sum_{i=0}^{n} rew_{\text{num}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})\right].$$

The fractional long-run average reward *is defined as*

$$\text{val}^{\sigma}_{\mathcal{M}, rew, \text{lra}} \stackrel{\text{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}{\sum_{i=0}^{n} rew_{\text{den}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}\right],$$

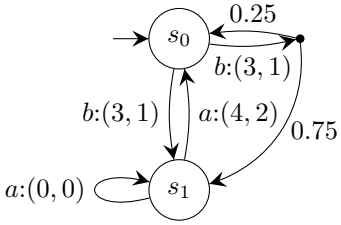


Figure 7.1: Reward-extended PA.

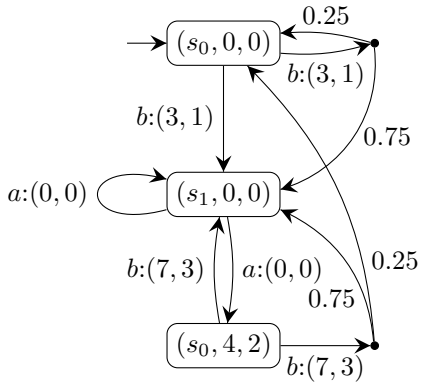


Figure 7.2: Encoded version of the PA of Figure 7.1

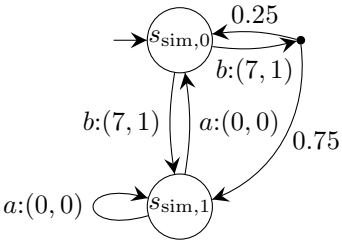


Figure 7.3: PA simulating the one of Figure 7.2.

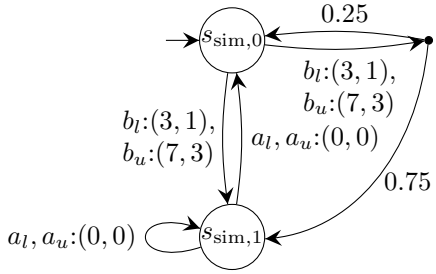


Figure 7.4: PA affinely simulating the one of Figure 7.2.

in case we have

$$Pr_{\mathcal{M},\sigma}(A) = 1,$$

where A is the set of paths on which the property is well-defined. In the above, we let $\frac{0}{0} \stackrel{\text{def}}{=} 0$ and $\frac{x}{0} \stackrel{\text{def}}{=} \infty$ for $x \neq 0$. For $Act_{\text{fair}} \subseteq Act$, we define the Act_{fair}-fair extremal values

$$\text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} = \sup_{\sigma \in Sched^{Act_{\text{fair}}}_{\mathcal{M}}} \text{val}^{\sigma}_{\mathcal{M},rew,\text{acc}} \text{ and } \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} = \sup_{\sigma \in Sched^{Act_{\text{fair}}}_{\mathcal{M}}} \text{val}^{\sigma}_{\mathcal{M},rew,\text{lra}},$$

and accordingly for $\text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},rew,\text{acc}}$ and $\text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}}$. For $\text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}}$ ($\text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}}$) we only take the supremum (infimum) over the schedulers σ for which $Pr_{\mathcal{M},\sigma}(A) = 1$.

We will later on use reward-extended PAs as the semantics of reward-extended PHAs. When considering accumulated reward properties, we add up all rewards we come across along a certain path. The value we consider then is the expected value over all paths. Properties of this kind can for instance be used to reason about the expected time until

system termination, the number of steps until an error is reached, and so on. Fractional long-run average values specify a value that is reached in the long-run operation of a system. The numerator will later on describe the value for which we want to obtain the average. The denominator will describe the time which has passed in a PHA. It is necessary to use a variable denominator here rather than to assume that each step takes one unit of time, because in the semantics of PHAs not all steps take the same amount of time. For instance, executing a guarded command is instantaneous, so that for these transitions no time passes at all.

We remark that the fractional long-run average reward might not be well-defined on all paths of a given model, which is the reason why we require the paths on which it is well-defined to have measure one. Jobstmann et al. [Blo+09; EJ11; EJ12] solve this problem in a different way by using

$$E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty}\liminf_{m\to\infty}\frac{\sum_{i=n}^{m} rew_{\mathrm{num}}(X_i^{\mathcal{M}},Y_i^{\mathcal{M}},Z_i^{\mathcal{M}})}{1+\sum_{i=n}^{m} rew_{\mathrm{den}}(X_i^{\mathcal{M}},Y_i^{\mathcal{M}},Z_i^{\mathcal{M}})}\right]. \tag{7.1}$$

This way, the value on each path always exists. In case the expected fractional long-run average value of Definition 7.3 exists, it agrees with the one defined in Equation 7.1. However, we cannot use this modified definition. Later on, in Lemma 7.6 we have to apply a certain model transformation the validity of which is not valid under the definition by Jobstmann et al.

As for reachability properties, we need to consider fairness conditions here. The reason we have to do this is similar to the basic setting: if a scheduler were allowed to let time stop, it could for instance let the time stop at a moment in which a very high or very low value has been reached. In this case, this value would then form the long-run average value. It would be unrealistic to consider this the extremal average over time, as indeed the time flow has stopped. This phenomenon could occur for both minimal and maximal values, and thus we consider fair schedulers in all cases.

Example 7.4. *Reconsider the PA of Figure 7.1. In this model, the accumulated reward would be infinite, so that we only consider the fractional long-run average reward. We have for* $Act_{\mathrm{fair}} \stackrel{\mathrm{def}}{=} \{b\}$ *that*

$$\mathrm{val}^{+,Act_{\mathrm{fair}}}_{\mathcal{M},rew,\mathrm{lra}} = \frac{12}{5} = 2.4 \text{ and } \mathrm{val}^{-,Act_{\mathrm{fair}}}_{\mathcal{M},rew,\mathrm{lra}} = \frac{7}{3} \approx 2.333,$$

which are for instance obtained by the simple schedulers σ^+ *and* σ^- *with*

$$\sigma^+(s_0) \stackrel{\mathrm{def}}{=} (b,[s_0{\mapsto}0.25, s_1{\mapsto}0.75]), \sigma^+(s_1) \stackrel{\mathrm{def}}{=} (a,[s_1{\mapsto}1]),$$
$$\sigma^-(s_0) \stackrel{\mathrm{def}}{=} (b,[s_1{\mapsto}1]), \sigma^-(s_1) \stackrel{\mathrm{def}}{=} (a,[s_1{\mapsto}1]). \qquad \triangle$$

We now consider variants of PAs where we postpone the rewards of certain actions, in such a way that they are then collected along with the rewards of later actions. For this, we set the rewards resulting from a subset of actions to zero, and instead remember these rewards by encoding them into the state space. Later on, we can collect these values when executing one of the remaining actions.

Definition 7.5. *Consider a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *with the reward structure* $rew = (rew_{\text{num}}, rew_{\text{den}})$ *and a subset* $Act_{\text{enc}} \subseteq Act$ *of the actions. The* Act_{enc}-encoded PA and reward structure

$$\text{enc}(\mathcal{M}, rew, Act_{\text{enc}}) \stackrel{\text{def}}{=} (\mathcal{M}_{\text{enc}}, rew_{\text{enc}})$$

are defined as $\mathcal{M}_{\text{enc}} \stackrel{\text{def}}{=} (S_{\text{enc}}, \overline{s}_{\text{enc}}, Act, \mathcal{T}_{\text{enc}})$, $rew_{\text{enc}} \stackrel{\text{def}}{=} (rew_{\text{enc,num}}, rew_{\text{enc,den}})$ *where*

- $S_{\text{enc}} \stackrel{\text{def}}{=} S \times \mathbb{R}^2$,
- $\overline{s}_{\text{enc}} \stackrel{\text{def}}{=} (\overline{s}, 0, 0)$,
- *for* $s_{\text{enc}} = (s, v_{\text{num}}, v_{\text{den}})$ *and* $a \in Act_{\text{enc}}$ *we have*

$$\mathcal{T}_{\text{enc}}(s_{\text{enc}}, a) \stackrel{\text{def}}{=} \{[(s'_1, 0, 0) \mapsto p_1, \ldots, (s'_n, 0, 0) \mapsto p_n] \mid [s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \in \mathcal{T}(s, a)\},$$

- *for* $s_{\text{enc}} = (s, v_{\text{num}}, v_{\text{den}})$ *and* $a \in Act \setminus Act_{\text{enc}}$ *we have*

$$\begin{aligned}\mathcal{T}_{\text{enc}}(s_{\text{enc}}, a) \stackrel{\text{def}}{=} \{&[(s'_1, v'_{\text{num}}, v'_{\text{den}}) \mapsto p_1, \ldots, (s'_n, v'_{\text{num}}, v'_{\text{den}}) \mapsto p_n] \mid \\ &[s'_1 \mapsto p_1, \ldots, s'_n \mapsto p_n] \in \mathcal{T}(s, a) \\ &\wedge v'_{\text{num}} = v_{\text{num}} + rew_{\text{num}}(s, a, \mu) \wedge v'_{\text{den}} = v_{\text{den}} + rew_{\text{den}}(s, a, \mu)\},\end{aligned}$$

- *for* $s_{\text{enc}} = (s, v_{\text{num}}, v_{\text{den}})$, $a \in Act_{\text{enc}}$ *and* $\mu \in \mathcal{T}_{\text{enc}}(s_{\text{enc}}, a)$ *we have*

$$\begin{aligned}rew_{\text{enc,num}}(s_{\text{enc}}, a, \mu) &\stackrel{\text{def}}{=} rew_{\text{num}}(s, a, \mu) + v_{\text{num}} \text{ and} \\ rew_{\text{enc,den}}(s_{\text{enc}}, a, \mu) &\stackrel{\text{def}}{=} rew_{\text{den}}(s, a, \mu) + v_{\text{den}},\end{aligned}$$

- *for* $s_{\text{enc}} = (s, v_{\text{num}}, v_{\text{den}})$, $a \in Act \setminus Act_{\text{enc}}$ *and* $\mu \in \mathcal{T}_{\text{enc}}(s, a)$ *we have*

$$rew_{\text{enc,num}}(s_{\text{enc}}, a, \mu) \stackrel{\text{def}}{=} 0 \text{ and } rew_{\text{enc,den}}(s_{\text{enc}}, a, \mu) \stackrel{\text{def}}{=} 0.$$

Thus, the rewards are stored in the continuous dimensions v_{num} and v_{den} of the state space. Each time we come across $a \in Act_{\text{enc}}$, we collect the original reward from this command plus the values of v_{num} and v_{den}, and reset these variables to zero. When executing a command $a \notin Act_{\text{enc}}$, we collect a reward of 0, but increase v_{num} and v_{den} accordingly. In Section 7.2, we will use this definition to postpone rewards of timed actions, to overcome the loss of timing information in the computed abstractions.

We can show that this construction maintains reward values, under the condition that $Act_{\text{enc}} = Act_{\text{fair}}$ is the set of fair actions.

Lemma 7.6. *Let* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *be a PA with reward structure rew and consider the* Act_{fair}*-encoded model and reward structure* $(\mathcal{M}_{\text{enc}}, rew_{\text{enc}}) = \text{enc}(\mathcal{M}, rew, Act_{\text{fair}})$ *for* $Act_{\text{fair}} \subseteq Act$*. Then we have*

$$\text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}, rew, \text{acc}} = \text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}_{\text{enc}}, rew_{\text{enc}}, \text{acc}} \text{ and } \text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}, rew, \text{lra}} = \text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}_{\text{enc}}, rew_{\text{enc}}, \text{lra}},$$

and accordingly for the minimal values.

Proof. We only consider the case of maximal fractional rewards, because the other cases are similar (but simpler for the accumulated rewards). For this, we have to show that for each $\sigma \in Sched_{\mathcal{M}}^{Act_{\text{fair}}}$ we have $\sigma_{\text{enc}} \in Sched_{\mathcal{M}_{\text{enc}}}^{Act_{\text{fair}}}$ with

$$\text{val}_{\mathcal{M},rew,\text{lra}}^{\sigma} = \text{val}_{\mathcal{M}_{\text{enc}},rew_{\text{enc}},\text{lra}}^{\sigma_{\text{enc}}},$$

and vice versa when starting with $\sigma_{\text{enc}} \in Sched_{\mathcal{M}_{\text{enc}}}^{Act_{\text{fair}}}$.

For $\beta \in Path_{\mathcal{M}}^{\text{fin}} \uplus Path_{\mathcal{M}}^{\text{inf}}$, denote by $\text{enc}(\beta)$ the corresponding path in $Path_{\mathcal{M}_{\text{enc}}}^{\text{inf}}$, in which we have encoded the rewards of $a \in Act \setminus Act_{\text{fair}}$ into the state space, as in the definition of $\mathcal{T}_{\text{enc}}$ in Definition 7.5. Assume $rew = (rew_{\text{num}}, rew_{\text{den}})$ and $rew_{\text{enc}} = (rew_{\text{enc,num}}, rew_{\text{enc,den}})$. We then define the measurable functions $f_n, f \colon Path_{\mathcal{M}}^{\text{inf}} \to \mathbb{R}_{\geq 0}$ and $f_{\text{enc},n}, f_{\text{enc}} \colon Path_{\mathcal{M}_{\text{enc}}}^{\text{inf}} \to \mathbb{R}_{\geq 0}$ for $n \in \mathbb{N}$ as

$$f_n \stackrel{\text{def}}{=} \frac{\sum_{i=0}^{n} rew_{\text{num}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}{\sum_{i=0}^{n} rew_{\text{den}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})} \text{ and } f_{\text{enc},n} \stackrel{\text{def}}{=} \frac{\sum_{i=0}^{n} rew_{\text{enc,num}}(X_i^{\mathcal{M}_{\text{enc}}}, Y_i^{\mathcal{M}_{\text{enc}}}, Z_i^{\mathcal{M}_{\text{enc}}})}{\sum_{i=0}^{n} rew_{\text{enc,den}}(X_i^{\mathcal{M}_{\text{enc}}}, Y_i^{\mathcal{M}_{\text{enc}}}, Z_i^{\mathcal{M}_{\text{enc}}})},$$

$$f \stackrel{\text{def}}{=} \lim_{n\to\infty} f_n \text{ and } f_{\text{enc}} \stackrel{\text{def}}{=} \lim_{n\to\infty} f_{\text{enc},n}.$$

For $\sigma \in Sched_{\mathcal{M}}^{Act_{\text{fair}}}$, we define $\sigma_{\text{enc}} \in Sched_{\mathcal{M}_{\text{enc}}}^{Act_{\text{fair}}}$ where for all $\beta \in Path_{\mathcal{M}}^{\text{fin}}$ we let $\sigma_{\text{enc}}(\text{enc}(\beta))$ be defined so that it corresponds to the decision of $\sigma(\beta)$, in that it chooses the same actions and the corresponding distributions for each path. For $A \in \Sigma_{\mathcal{M}}$ and $A_{\text{enc}} \stackrel{\text{def}}{=} \{\text{enc}(\beta) \mid \beta \in A\}$, we have

$$Pr_{\mathcal{M},\sigma}(A) = Pr_{\mathcal{M}_{\text{enc}},\sigma_{\text{enc}}}(A_{\text{enc}}) \text{ and } Pr_{\mathcal{M},\sigma}(Path_{\mathcal{M}}^{Act_{\text{fair}}}) = Pr_{\mathcal{M}_{\text{enc}},\sigma_{\text{enc}}}(Path_{\mathcal{M}_{\text{enc}}}^{Act_{\text{fair}}}) = 1.$$

Because of this, and by the definition of the expectation as an integral, to show that $\text{val}_{\mathcal{M},rew,\text{lra}}^{\sigma} = \text{val}_{\mathcal{M}_{\text{enc}},rew_{\text{enc}},\text{lra}}^{\sigma_{\text{enc}}}$, it suffices to show that for each $\beta \in Path_{\mathcal{M}}^{Act_{\text{fair}}}$ we have $f(\beta) = f_{\text{enc}}(\text{enc}(\beta))$.

Consider thus an arbitrary $\beta \in Path_{\mathcal{M}}^{Act_{\text{fair}}}$. We define

$$Act_{\text{fair}}(\beta) \stackrel{\text{def}}{=} \{i \in \mathbb{N} \mid \beta[i] \in Act_{\text{fair}}\},$$

that is the set of the positions in a path which contain actions of the fairness set Act_{fair}. By the definition of a fair path (cf. Definition 3.5), this set is infinitely large. Then, by definition of the encoding, for all $n \in Act_{\text{fair}}(\beta)$ we have

$$f_n(\beta) = f_{\text{enc},n}(\text{enc}(\beta)),$$

and thus also

$$f(\beta) = f_{\text{enc}}(\text{enc}(\beta)).$$

For the other direction, we start with $\sigma_{\text{enc}} \in Sched_{\mathcal{M}_{\text{enc}}}^{Act_{\text{fair}}}$, construct $\sigma \in Sched_{\mathcal{M}}^{Act_{\text{fair}}}$ and show equality in the same way. □

As seen from the proof, Lemma 7.6 indeed requires to restrict to Act_{fair} schedulers. Otherwise, we could have a situation where in the original model we obtain some reward which we do not obtain in the encoded model, because we just encode this reward in the state space but never actually collect it later.

Example 7.7. *The PA of Figure 7.2 is the Act_{fair}-encoded PA of the one in Figure 7.1, for $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$. Because of this, the maximal and minimal fair average values of $\frac{12}{5} = 2.4$ and $\frac{7}{3} \approx 2.333$ are the same in both models.* △

To prove the validity of abstractions of PHA for reward-based properties, we extend the definition of a simulation relation from Definition 3.15 to take into account reward structures. We only consider simple reward structures, because we will only need this restricted form for the analysis of PHAs.

Definition 7.8. *Consider two PAs $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ and $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ related by a simulation relation R and for which we are given simple reward structures $rew = (rew_{\text{num}}, rew_{\text{den}})$ and $rew_{\text{sim}} = (rew_{\text{sim,num}}, rew_{\text{sim,den}})$. We say that R is* upper-bound compatible*, if in case we have $(s, s_{\text{sim}}) \in R$ then for all $a \in Act$ we have*

$$rew_{\text{num}}(s,a) \leq rew_{\text{sim,num}}(s_{\text{sim}}, a) \text{ and } rew_{\text{den}}(s,a) \geq rew_{\text{sim,den}}(s_{\text{sim}}, a).$$

If there exists such a relation, we write

$$(\mathcal{M}, rew) \stackrel{\text{up}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}}).$$

We define lower-bound compatible *simulations R accordingly by swapping $\leq$ and $\geq$ above and write*

$$(\mathcal{M}, rew) \stackrel{\text{lo}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}}).$$

With simulations, we can establish upper and lower bounds on the reward properties of simulated models by considering the corresponding property in the simulating model.

Lemma 7.9. *For PAs $\mathcal{M}$ and $\mathcal{M}_{\text{sim}}$ with corresponding simple reward structures rew and rew_{sim}, if $(\mathcal{M}, rew) \stackrel{\text{up}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}})$ then*

$$\text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{acc}} \leq \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}},rew_{\text{sim}},\text{acc}} \text{ and } \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} \leq \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}},rew_{\text{sim}},\text{lra}}.$$

Likewise, if $(\mathcal{M}, rew) \stackrel{\text{lo}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}})$ then

$$\text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},rew,\text{acc}} \geq \text{val}^{-,Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}},rew_{\text{sim}},\text{acc}} \text{ and } \text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} \geq \text{val}^{-,Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}},rew_{\text{sim}},\text{lra}}.$$

Proof. We restrict to the proof of

$$\text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} \leq \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}},rew_{\text{sim}},\text{lra}}, \tag{7.2}$$

the other three cases follow along the lines.

For $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ and $(rew_{\text{num}}, rew_{\text{den}}) = rew$, we define the PA and reward structure

$$S \times \mathcal{M} \stackrel{\text{def}}{=} (S, \overline{s}, S \times Act, S \times \mathcal{T}) \text{ and } S \times rew \stackrel{\text{def}}{=} (S \times rew_{\text{num}}, S \times rew_{\text{den}}),$$

where

- for $s \in S$ and $a \in Act$ we have $(S \times \mathcal{T})(s,(s,a)) \stackrel{\text{def}}{=} \mathcal{T}(s,a)$ and $(S \times \mathcal{T})(\cdot,\cdot) \stackrel{\text{def}}{=} \emptyset$ else,
- for $s \in S$ and $a \in Act$ we have $(S \times rew_{\text{num}})(s,(s,a)) \stackrel{\text{def}}{=} rew_{\text{num}}(s,a)$, accordingly for $S \times rew_{\text{den}}$.

Then, for $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ and $(rew_{\text{sim,num}}, rew_{\text{sim,den}}) = rew_{\text{sim}}$, we define the PA and reward structure

$$S \times \mathcal{M}_{\text{sim}} \stackrel{\text{def}}{=} (S_{\text{sim}}, \overline{s}_{\text{sim}}, S \times Act, S \times \mathcal{T}_{\text{sim}}) \text{ and } S \times rew_{\text{sim}} \stackrel{\text{def}}{=} (S \times rew_{\text{sim,num}}, S \times rew_{\text{sim,den}}),$$

where

- for $s_{\text{sim}} \in S_{\text{sim}}$, $s \in S$ and $a \in Act$, we have $(S \times \mathcal{T}_{\text{sim}})(s_{\text{sim}},(s,a)) \stackrel{\text{def}}{=} \mathcal{T}_{\text{sim}}(s_{\text{sim}},a)$,
- for $s_{\text{sim}} \in S_{\text{sim}}$, $s \in S$ and $a \in Act$, we specify $(S \times rew_{\text{sim,num}})(s_{\text{sim}},(s,a)) \stackrel{\text{def}}{=} rew_{\text{sim,num}}(s_{\text{sim}},a)$, and accordingly for $S \times rew_{\text{sim,den}}$.

Consider arbitrary schedulers $\sigma \in Sched_{\mathcal{M}}$ and $\sigma_{\text{sim}} \in Sched_{\mathcal{M}_{\text{sim}}}$. We can construct corresponding schedulers $S \times \sigma \in Sched_{S \times \mathcal{M}}$ and $S \times \sigma_{\text{sim}} \in Sched_{S \times \mathcal{M}_{\text{sim}}}$ with

$$\text{val}^{\sigma}_{\mathcal{M},rew,\text{lra}} = \text{val}^{S\times\sigma}_{S\times\mathcal{M},S\times rew,\text{lra}} \text{ and } \text{val}^{\sigma_{\text{sim}}}_{\mathcal{M}_{\text{sim}},rew_{\text{sim}},\text{lra}} = \text{val}^{S\times\sigma_{\text{sim}}}_{S\times\mathcal{M}_{\text{sim}},S\times rew_{\text{sim}},\text{lra}}, \tag{7.3}$$

and vice versa when starting with schedulers $S \times \sigma \in Sched_{S\times\mathcal{M}}$ and $S \times \sigma_{\text{sim}} \in Sched_{S\times\mathcal{M}_{\text{sim}}}$. We have (we remark that $(rew_{\text{num}}, rew_{\text{den}})$ is a reward structure in $\mathcal{M}$ and not in $S \times \mathcal{M}$)

$$\text{val}^{S\times\sigma}_{S\times\mathcal{M},S\times rew,\text{lra}} = E_{S\times\mathcal{M},S\times\sigma}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(Y_i^{S\times\mathcal{M}})}{\sum_{i=0}^{n} rew_{\text{den}}(Y_i^{S\times\mathcal{M}})}\right], \tag{7.4}$$

and accordingly for $S \times \mathcal{M}_{\text{sim}}$. This means, that the average reward can be derived in this extended model by only looking at the sequence of actions, rather than requiring to consider also states and distributions. It can be shown that $S \times \mathcal{M}_{\text{sim}}$ simulates $S \times \mathcal{M}$ by using a simulation relation between $\mathcal{M}$ and $\mathcal{M}_{\text{sim}}$, which we already know to exist. Because of this, from $S \times \sigma$ we can construct a scheduler $S \times \sigma_{\text{sim}}$ for which measures on traces agree [Seg95, Proposition 7.7.1]. From Equation 7.4 and by the definition of the reward structures, we have

$$\begin{aligned}
&E_{S\times\mathcal{M},S\times\sigma}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(Y_i^{S\times\mathcal{M}})}{\sum_{i=0}^{n} rew_{\text{den}}(Y_i^{S\times\mathcal{M}})}\right] \\
&= E_{S\times\mathcal{M}_{\text{sim}},S\times\sigma_{\text{sim}}}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(Y_i^{S\times\mathcal{M}_{\text{sim}}})}{\sum_{i=0}^{n} rew_{\text{den}}(Y_i^{S\times\mathcal{M}_{\text{sim}}})}\right] \\
&\leq E_{S\times\mathcal{M}_{\text{sim}},S\times\sigma_{\text{sim}}}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{sim,num}}(X_i^{S\times\mathcal{M}_{\text{sim}}}, Y_i^{S\times\mathcal{M}_{\text{sim}}})}{\sum_{i=0}^{n} rew_{\text{sim,den}}(X_i^{S\times\mathcal{M}_{\text{sim}}}, Y_i^{S\times\mathcal{M}_{\text{sim}}})}\right].
\end{aligned} \tag{7.5}$$

By Equation 7.3 and Equation 7.5, for each $\sigma \in Sched_{\mathcal{M}}$ we can construct $\sigma_{\text{sim}} \in Sched_{\mathcal{M}_{\text{sim}}}$ with

$$\text{val}^{\sigma}_{\mathcal{M},rew,\text{lra}} \leq \text{val}^{\sigma_{\text{sim}}}_{\mathcal{M}_{\text{sim}},rew_{\text{sim}},\text{lra}},$$

which implies Equation 7.2. □

Thus, the principal idea of this simulation is that the simulating automaton can mimic the behaviour of the simulated one, while overapproximating or underapproximating respectively rew_{num} and rew_{den}.

Example 7.10. *In Figure 7.3 we give a PA with corresponding reward structures which simulates the one of Figure 7.2 by an upper-bound compatible simulation relation. The maximal fractional long-run average reward of the latter is indeed much higher than the former, namely, 7 rather than* $\frac{12}{5} = 2.4$. *To obtain a lower-bound compatible simulation relation, we would replace the rewards* $(7, 1)$ *by* $(3, 3)$, *thus obtaining a reward of* 1 *which is considerably lower than the minimal reward* $\frac{7}{3} \approx 2.333$ *of the original model.* △

In the case of affine reward structures, we can define a simulation relation with which we can obtain more precise results.

Definition 7.11. *Consider two PAs* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and* $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ *which are related by a simulation relation* R. *Further, consider three affine reward structures*

$$\begin{aligned} rew &= (rew_{\text{num}}, rew_{\text{den}}), \\ rew_{\text{sim,up}} &= (rew_{\text{sim,up,num}}, rew_{\text{sim,up,den}}), \\ rew_{\text{sim,lo}} &= (rew_{\text{sim,lo,num}}, rew_{\text{sim,lo,den}}). \end{aligned}$$

We require that rew, $rew_{\text{sim,up}}$ *and* $rew_{\text{sim,lo}}$ *are affine with the same factors* mul_a, add_a *(cf. Definition 7.1) for each action* a, *that is for* $s \in S$ *and* $\mathbf{z} \in \mathbf{A}$ *we have*

$$\begin{aligned} rew_{\text{num}}(s, a) &= mul_a rew_{\text{den}}(s, a) + add_a, \\ rew_{\text{sim,up,num}}(\mathbf{z}, a) &= mul_a rew_{\text{sim,up,den}}(\mathbf{z}, a) + add_a, \\ rew_{\text{sim,lo,num}}(\mathbf{z}, a) &= mul_a rew_{\text{sim,lo,den}}(\mathbf{z}, a) + add_a. \end{aligned}$$

Then, we define R *as being* affine compatible *if for all* $(s, s_{\text{sim}}) \in R$ *and* $a \in Act$ *we have*

$$rew_{\text{sim,lo,num}}(s_{\text{sim}}, a) \leq rew_{\text{num}}(s, a) \leq rew_{\text{sim,up,num}}(s_{\text{sim}}, a).$$

If there exists such a relation, we write

$$(\mathcal{M}, rew) \overset{\text{aff}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim,up}}, rew_{\text{sim,lo}}).$$

As before, affine simulations maintain reward properties.

Lemma 7.12. *Consider the PAs* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *with the reward structure* rew *and* $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ *with reward structures* $rew_{\text{sim,up}}$ *and* $rew_{\text{sim,lo}}$ *with* $(\mathcal{M}, rew) \overset{\text{aff}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim,up}}, rew_{\text{sim,lo}})$. *We define*

$$\mathcal{M}'_{\text{sim}} \overset{\text{def}}{=} (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act \times \{\text{lo}, \text{up}\}, \mathcal{T}'_{\text{sim}}) \text{ and } rew'_{\text{sim}} = (rew'_{\text{sim,num}}, rew'_{\text{sim,den}}),$$

where

- *for* $s \in S_{\text{sim}}$ *and* $a \in Act$ *we have* $\mathcal{T}'_{\text{sim}}(s,(a,\text{lo})) \stackrel{\text{def}}{=} \mathcal{T}'_{\text{sim}}(s,(a,\text{up})) \stackrel{\text{def}}{=} \mathcal{T}_{\text{sim}}(s,a)$,
- *for* $s \in S_{\text{sim}}$ *and* $a \in Act$ *we have* $rew'_{\text{sim,num}}(s,(a,\text{lo})) \stackrel{\text{def}}{=} rew_{\text{sim,lo,num}}(s,a)$, *and accordingly for* $rew'_{\text{sim,num}}(s,(a,\text{up}))$, $rew'_{\text{sim,den}}(s,(a,\text{lo}))$ *and* $rew'_{\text{sim,den}}(s,(a,\text{up}))$.

Then we have

$$\text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{acc}} \leq \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M}'_{\text{sim}},rew'_{\text{sim}},\text{acc}} \text{ and } \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} \leq \text{val}^{+,Act_{\text{fair}}}_{\mathcal{M}'_{\text{sim}},rew'_{\text{sim}},\text{lra}},$$

and accordingly for the minimising cases.

Proof. Firstly, we remark that we also have

$$rew_{\text{sim,lo,den}}(s_{\text{sim}},a) \leq rew_{\text{den}}(s,a) \leq rew_{\text{sim,up,den}}(s_{\text{sim}},a)$$

which follows from the definition of affine simulations. We define $S \times \mathcal{M}$ and $S \times \mathcal{M}_{\text{sim}}$ as in the proof of Lemma 7.9. As there, for $\sigma \in Sched_{\mathcal{M}}$ we can construct $S \times \sigma \in Sched_{S\times\mathcal{M}}$ and from this $S \times \sigma_{\text{sim}} \in Sched_{S\times\mathcal{M}_{\text{sim}}}$ so that measures on the traces agree. We also consider a corresponding model $S \times \mathcal{M}'_{\text{sim}} \stackrel{\text{def}}{=} (S_{\text{sim}}, \overline{s}_{\text{sim}}, S \times Act \times \{\text{lo},\text{up}\}, S \times \mathcal{T}'_{\text{sim}})$. From $S \times \sigma_{\text{sim}}$, we construct a scheduler $S \times \sigma'_{\text{sim}} \in Sched_{S\times\mathcal{M}'_{\text{sim}}}$ as follows. For $\beta \in Path^{\text{fin}}_{S\times\mathcal{M}'_{\text{sim}}} \uplus Path^{\text{inf}}_{S\times\mathcal{M}'_{\text{sim}}}$ let $\overline{\beta} \in Path^{\text{fin}}_{S\times\mathcal{M}_{\text{sim}}}$ be so that for all $\beta[i] = (a,\text{lo})$ or $\beta[i] = (a,\text{up})$ we have $\overline{\beta}[i] = a$. For $s \in S, s_{\text{sim}} \in S_{\text{sim}}$ and $a \in Act$, let $p_{\text{up},s,s_{\text{sim}},a}, p_{\text{lo},s,s_{\text{sim}},a} \in [0,1]$ be so that

- $p_{\text{up},s,s_{\text{sim}},a} + p_{\text{lo},s,s_{\text{sim}},a} = 1$ and
- $rew_{\text{num}}(s,a) = p_{\text{up},s,s_{\text{sim}},a} rew_{\text{sim,up,num}}(s_{\text{sim}},a) + p_{\text{lo},s,s_{\text{sim}},a} rew_{\text{sim,lo,num}}(s_{\text{sim}},a)$,
- (which by the definition of affine rewards in Definition 7.11 also implies that $rew_{\text{den}}(s,a) = p_{\text{up},s,s_{\text{sim}},a} rew_{\text{sim,up,den}}(s_{\text{sim}},a) + p_{\text{lo},s,s_{\text{sim}},a} rew_{\text{sim,lo,den}}(s_{\text{sim}},a)$).

For $\beta \in Path^{\text{fin}}_{S\times\mathcal{M}'_{\text{sim}}}$, $s \in S$, $a \in Act$ and $\mu \in Distr(S_{\text{sim}})$, we let

$$(S \times \sigma'_{\text{sim}})(\beta)((s,a,\text{up}),\mu) \stackrel{\text{def}}{=} p_{\text{up},s,\text{last}(\beta),a}(S \times \sigma_{\text{sim}})(\overline{\beta})((s,a),\mu) \text{ and}$$
$$(S \times \sigma'_{\text{sim}})(\beta)((s,a,\text{lo}),\mu) \stackrel{\text{def}}{=} p_{\text{lo},s,\text{last}(\beta),a}(S \times \sigma_{\text{sim}})(\overline{\beta})((s,a),\mu).$$

With

$$f_n \stackrel{\text{def}}{=} \frac{\sum_{i=0}^{n} rew_{\text{num}}(Y_i^{S\times\mathcal{M}_{\text{sim}}})}{\sum_{i=0}^{n} rew_{\text{den}}(Y_i^{S\times\mathcal{M}_{\text{sim}}})} \text{ and } f'_n \stackrel{\text{def}}{=} \frac{\sum_{i=0}^{n} rew_{\text{num}}(Y_i^{S\times\mathcal{M}'_{\text{sim}}})}{\sum_{i=0}^{n} rew_{\text{den}}(Y_i^{S\times\mathcal{M}'_{\text{sim}}})},$$

we have for $\beta \in Path^{\text{fin}}_{S\times\mathcal{M}_{\text{sim}}}$ with $|\beta| \geq n$ that

$$f_n(\beta) = \sum_{\substack{\beta' \in Path^{\text{fin}}_{S\times\mathcal{M}'_{\text{sim}}}, \\ \overline{\beta'}=\beta}} f'_n(\beta') \frac{Pr_{S\times\mathcal{M}'_{\text{sim}},S\times\sigma'_{\text{sim}}}(\beta')}{Pr_{S\times\mathcal{M}_{\text{sim}},S\times\sigma_{\text{sim}}}(\beta)}. \tag{7.6}$$

Further, for $A \subseteq Path^{\text{inf}}_{S\times\mathcal{M}_{\text{sim}}}$ and $A' \stackrel{\text{def}}{=} \{\beta' \in Path^{\text{inf}}_{S\times\mathcal{M}'_{\text{sim}}} \mid \overline{\beta'} \in A\}$ we have

$$Pr_{S\times\mathcal{M}_{\text{sim}},S\times\sigma_{\text{sim}}}(A) = Pr_{S\times\mathcal{M}'_{\text{sim}},S\times\sigma'_{\text{sim}}}(A'). \tag{7.7}$$

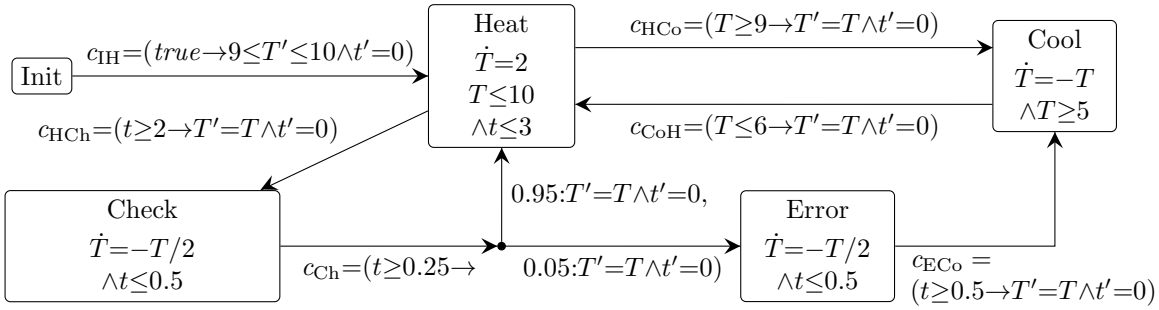


Figure 7.5: Thermostat modified for long-run average rewards.

By the definition of the expectation as an integral, from Equation 7.6 and Equation 7.7, we can conclude that

$$\mathrm{val}^{S\times\sigma}_{S\times\mathcal{M},S\times rew,\mathrm{lra}} = \mathrm{val}^{S\times\sigma'_{\mathrm{sim}}}_{\mathcal{M}'_{\mathrm{sim}},rew',\mathrm{lra}},$$

which proves the lemma. □

Similarly as for upper-bound and lower-bound compatible simulations, the affinely simulating automaton $\mathcal{M}_{\mathrm{sim}}$ can mimic the behaviours of the simulated one. In $\mathcal{M}'_{\mathrm{sim}}$ we also mimic the behaviours of the original model, but we use randomised choices over (a, up) and (a, lo) to obtain exactly the same reward as when choosing a in the original model. The reason is that we will obtain results which are more precise, that is for both rew_{num} and rew_{den} we either minimise or maximise, whereas for nonaffine reward structures we had to minimise and optimise in the opposite directions.

Example 7.13. *In Figure 7.4 we give a PA which affinely simulates the one of Figure 7.2. Maximal and minimal fractional long-run averages are* 3 *and* $\frac{7}{3} \approx 2.333$*, which is more precise than the values obtained from Figure 7.3 as discussed in Example 7.10.* △

7.2 Rewards for Probabilistic Hybrid Automata

We can now define reward structures of PHAs.

Definition 7.14. *A* reward structure *for a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m\rangle_{m\in M}, Cmds)$ *is a tuple*

$$rew = (rew_{\mathrm{dis}}, rew_{\mathrm{cts}})$$

of two functions, the discrete reward function $rew_{\mathrm{dis}} \colon ((M \times \mathbb{R}^k) \times Cmds) \to \mathbb{R}_{\geq 0}$ *and the* continuous reward function $rew_{\mathrm{cts}} \colon ((M \times \mathbb{R}^k) \times \mathbb{R}_{\geq 0}) \to \mathbb{R}_{\geq 0}$*. We require that for each* $m \in M$ *and* $v \in \mathbb{R}^k$ *we have*

- $rew_{\mathrm{cts}}((m, v), \cdot)$ *monotonically increasing,*
- $rew_{\mathrm{cts}}((m, v), \mathsf{t}) = rew_{\mathrm{cts}}((m, v), \mathsf{t}_1) + rew_{\mathrm{cts}}((m, v), \mathsf{t}_2)$ *for* $\mathsf{t}, \mathsf{t}_1, \mathsf{t}_2 \geq 0$ *with* $\mathsf{t} = \mathsf{t}_1 + \mathsf{t}_2$*.*

A discrete reward function is called affine *if* $rew_{\text{dis}}(\cdot, c)$ *is constant for all* $c \in Cmds$*. In this case, we can write it as* $rew_{\text{dis}} \colon Cmds \to \mathbb{R}_{\geq 0}$*. A continuous reward function is called* affine *if for all* $m \in M$ *there is* $mul_m \in \mathbb{R}_{\geq 0}$ *where for all* $v \in \mathbb{R}^k$ *and* $\mathtt{t} \in \mathbb{R}_{\geq 0}$ *we have* $rew_{\text{cts}}((m, v), \mathtt{t}) = mul_m \mathtt{t}$*. In this case, we also write it as* $rew_{\text{cts}} \colon M \to \mathbb{R}_{\geq 0}$*, where* $rew_{\text{cts}}(m) = mul_m$*. If both the discrete and the continuous part are affine, we call* rew affine*.*

Example 7.15. *Consider the thermostat of Figure 3.3 in which it is not possible to leave the* Error *mode. As we do not consider time-bounded properties, we assume that the global clock* c *is no longer part of the model. We define a reward structure* $rew_{\text{acc}} \stackrel{\text{def}}{=} (rew_{\text{acc,dis}}, rew_{\text{acc,cts}})$ *with*

$$rew_{\text{acc,cts}}(\cdot, \cdot) \stackrel{\text{def}}{=} 0 \text{ and } rew_{\text{acc,dis}}(s, c) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } c \in \{c_{\text{E}}, c_{\text{IH}}\}, \\ 1 & \text{else.} \end{cases}$$

We thus obtain a reward for executing a command except the one of Error *or the one to initialise the system. Thus, using the minimal accumulated reward we can reason about the minimal expected number of commands executed until an error happens.*

Now consider the modified thermostat of Figure 7.5. We have extended the Error *mode so that the system can recover from an error after a certain time. We define another reward structure* $rew_{\text{lra}} = (rew_{\text{lra,dis}}, rew_{\text{lra,cts}})$ *where*

$$rew_{\text{lra,dis}}(\cdot, \cdot) \stackrel{\text{def}}{=} 0 \text{ and } rew_{\text{lra,cts}}((m, t, T), \mathtt{t}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } m = \text{Error}, \\ 0 & \text{else.} \end{cases}$$

Thus, we obtain a reward of 1 *per time unit when being in mode* Error*. By considering the maximal fractional long-run average reward, we can reason about the maximal percentage of time the system can spend in the* Error *mode.* △

In the definition of the semantics of a PHA, it is possible to choose any nonnegative time duration to wait. However, depending on the corresponding post operator, the actual time for which the model will wait might be lower. Indeed, by Requirement 3 of Definition 2.12 there is always a fixed upper bound for the maximal time to wait. Also, depending on which state the automaton actually moves to, the timed transition might stop earlier or later. To specify time-dependent rewards correctly, we must thus have some means to find out how long the PHA has actually waited, if we intended to wait for a certain amount of time and reach a certain state.

Definition 7.16. *Consider a* k*-dimensional post operator* $Post$*. For* $v \in \mathbb{R}^k$*,* $\mathtt{t} \in \mathbb{R}_{\geq 0}$ *and* $v' \in Post(v, \mathtt{t})$ *we define* $\text{dur}_{Post}(v, \mathtt{t}, v') \stackrel{\text{def}}{=}$

$$\sup\left(\{0\} \cup \{t_1 + t_2 \mid t_1 + t_2 \leq \mathtt{t} \wedge \exists v'' \in Post(v, t_1).\ v'' \neq v' \wedge v' \in Post(v'', t_2)\}\right).$$

We shortly write $\text{dur}_m(v, \mathtt{t}, v')$ *for* $\text{dur}_{Post_m}(v, \mathtt{t}, v')$*, where* $Post_m$ *is the post operator of a given mode* m *in a PHA.*

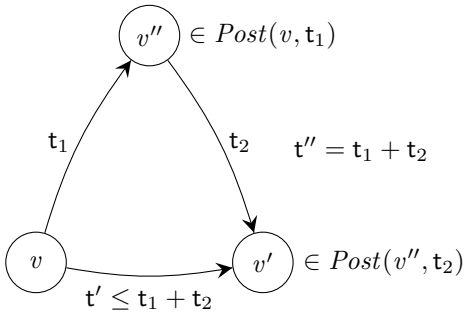


Figure 7.6: Illustration of dur.

Thus, $\mathrm{dur}_m(v, \mathsf{t}, v')$ is the latest time in which variable configuration v' is reached from v in mode m if we intend to wait for t. This means that if we have $\mathrm{dur}_m(v, \mathsf{t}, v') \lneqq \mathsf{t}$, we have a time lock in v'.

Example 7.17. *Consider again the model of Figure 7.5. For* $(t', T') \in Post_{\mathrm{Check}}((t, T), \mathsf{t})$ *we have*

$$\mathrm{dur}_{\mathrm{Check}}((t, T), \mathsf{t}, (t', T')) = \begin{cases} 0 & \text{if } t \geq 0.5, \\ t' - t & \text{else.} \end{cases}$$

The definition for the other modes is similar. △

We use the somewhat complicated Definition 7.16 to take into account cases in which we start in a certain v, wait for t and might reach v' at two different points of time $\mathsf{t}' < \mathsf{t}'' \leq \mathsf{t}$, as illustrated in Figure 7.6. As we intended to wait for t, we would want dur to return t'' rather than t'. If we had defined dur just to take the minimal time at which v' is visited, we would have obtained t_1 instead. On the other hand, we also cannot define dur using the maximum time, because in case of a time lock this value would be equal to t, although the given value might have been reached long before t.

Example 7.18. *We illustrate the issues by a simple example. Consider a post operator* $Post\colon (\mathbb{R} \times \mathbb{R}_{\geq 0}) \to 2^{\mathbb{R}^k}$ *where for* $0 \leq t \leq 1$ *we have*

$$Post(t, \mathsf{t}) \stackrel{\text{def}}{=} \{\min\{t', 1\} \mid t' \in [t + \mathsf{t}, t + 2\mathsf{t}]\}.$$

Then we have

$$Post(0, 0.5) = [0.5, 1], Post(0.5, 0.5) = \{1\}, Post(0, 1) = \{1\}, \text{ and } Post(0, 2) = \{1\}.$$

Because of this, we have $\mathrm{dur}((0), 2, (1)) = 1$ *rather than* 0.5 *or* 2. △

With these preparations, we can define the semantics of PHAs reward structures.

Definition 7.19. *Given a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *with semantics* $[\![\mathcal{H}]\!] = (S, \overline{s}, Act, \mathcal{T})$ *and a corresponding reward structure* $rew = (rew_{\mathrm{dis}}, rew_{\mathrm{cts}})$*, the* semantics *of the reward structures is the PA reward structure* $[\![rew]\!] \stackrel{\text{def}}{=} ([\![rew]\!]_{\mathrm{num}}, [\![rew]\!]_{\mathrm{den}})$*. For* $s \in S$*,* $c \in Cmds$ *and* $\mu \in \mathcal{T}(s, c)$ *we have*

$$[\![rew]\!]_{\mathrm{num}}(s, c, \mu) \stackrel{\text{def}}{=} rew_{\mathrm{dis}}(c) \text{ and } [\![rew]\!]_{\mathrm{den}}(s, c, \mu) \stackrel{\text{def}}{=} 0.$$

For $s = (m, v) \in S$, $\mathsf{t} \in \mathbb{R}_{\geq 0}$ *and* $\mu = [(m, v') \mapsto 1] \in \mathcal{T}(s, t)$ *we define*

$$[\![rew]\!]_{\text{num}}(s, \mathsf{t}, \mu) \stackrel{\text{def}}{=} rew_{\text{cts}}(s, \text{dur}_m(v, \mathsf{t}, v')) \text{ and } [\![rew]\!]_{\text{den}}(s, \mathsf{t}, \mu) \stackrel{\text{def}}{=} \text{dur}_m(v, \mathsf{t}, v').$$

Example 7.20. *Reconsider the reward structure* $rew_{\text{lra}} = (rew_{\text{lra,dis}}, rew_{\text{lra,cts}})$ *given in Example 7.15. For a state* $s = (\text{Error}, t, T)$ *of the mode* Error *and* $\mu = [(\text{Error}, t', T') \mapsto 1]$, *we have*

$$[\![rew_{\text{lra}}]\!](s, \mathsf{t}, \mu) = \text{dur}_{\text{Error}}((t, T), \mathsf{t}, (t', T')),$$

which means that we cannot obtain a reward of more than 0.5. *If we had not taken* $\text{dur}_{\text{Error}}$ *into account, and simply defined*

$$[\![rew_{\text{lra}}]\!](s, \mathsf{t}, \mu) = \mathsf{t},$$

the maximal fractional long-run average reward in the semantics with this reward structure would be $\lim_{\mathsf{t}\to\infty} \frac{\mathsf{t}}{r+\mathsf{t}} = 1$ *for some constant* r, *because the reward would be the same as if we could stay in* Error *for an arbitrarily long time.* △

We can then define the values of reward properties using the semantics of PHAs and their reward structures.

Definition 7.21. *Given a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *with semantics* $[\![\mathcal{H}]\!] = (S, Act, \mathcal{T})$ *and a corresponding reward structure* rew, *we define the* maximal accumulated reward *as*

$$\text{val}^{+}_{\mathcal{H}, rew, \text{acc}} \stackrel{\text{def}}{=} \text{val}^{+, Cmds}_{[\![\mathcal{H}]\!], [\![rew]\!], \text{acc}},$$

and define the minimal accumulated reward *accordingly. We define the* maximal time-average reward *as*

$$\text{val}^{+}_{\mathcal{H}, rew, \text{lra}} \stackrel{\text{def}}{=} \text{val}^{+, Cmds}_{[\![\mathcal{H}]\!], [\![rew]\!], \text{lra}},$$

and accordingly the minimal time-average reward.

As for reachability properties, it does not matter whether or not we apply a time restriction. Because we use abstractions which actually abstract the time-restricted semantics as in Chapter 3, we need the following lemma.

Lemma 7.22. *For a PHA* $\mathcal{H}$ *given time restriction* $\mathbf{T}$ *we have*

$$\text{val}^{+}_{\mathcal{H}, rew, \text{acc}} = \text{val}^{+, Cmds}_{[\![\mathcal{H}, \mathbf{T}]\!], [\![rew]\!], \text{acc}},$$
$$\text{val}^{+}_{\mathcal{H}, rew, \text{lra}} = \text{val}^{+, Cmds}_{[\![\mathcal{H}, \mathbf{T}]\!], [\![rew]\!], \text{lra}},$$

and accordingly for the minimal values.

The proof follows along the lines of Lemma 3.26.

As already prepared for a similar form for the semantical model in Definition 7.5, we now define a transformation to PHAs, in which we postpone the rewards obtained by timed transitions. This way, we collect the rewards only at the execution of commands. This will turn out to be advantageous in the later abstraction in which we lose the information about the duration of timed transitions.

Definition 7.23. *Given a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *with the reward structure* $rew = (rew_{\text{dis}}, rew_{\text{cts}})$*, the* reward-encoded *PHA and reward structures*

$$\text{enc}(\mathcal{H}, rew) \stackrel{\text{def}}{=} (\mathcal{H}_{\text{enc}}, rew_{\text{enc}}, rew_{\mathtt{t}}),$$

with $\mathcal{H}_{\text{enc}} \stackrel{\text{def}}{=} (M, k+2, \overline{m}, \langle Post_{\text{enc},m} \rangle_{m \in M}, Cmds_{\text{enc}})$*,* $rew_{\text{enc}} = (rew_{\text{enc,dis}}, rew_{\text{enc,cts}})$ *and* $rew_{\mathtt{t}} = (rew_{\mathtt{t},\text{dis}}, rew_{\mathtt{t},\text{cts}})$*, are defined so that*

- *for each* $s = (m, v, v_{\text{num}}, v_{\text{den}})$ *with* $m \in M$*,* $v \in \mathbb{R}^k$*,* $v_{\text{num}}, v_{\text{den}} \in \mathbb{R}$*, and* $\mathtt{t} \in \mathbb{R}_{\geq 0}$*, we have* $(v', v'_{\text{num}}, v'_{\text{den}}) \in Post_{\text{enc},m}((v, v_{\text{num}}, v_{\text{den}}), \mathtt{t})$ *if and only if*
  - $v' \in Post_m(v, \mathtt{t})$*,*
  - $v'_{\text{num}} = v_{\text{num}} + rew_{\text{cts}}(m, v, \text{dur}_m(v, \mathtt{t}, v'))$*,*
  - $v'_{\text{den}} = v_{\text{den}} + \text{dur}_m(v, \mathtt{t}, v')$*,*
- *for each* $c = (g \to p_1 : u_1 + \ldots + p_n : u_n)$*, we have exactly one corresponding* $c_{\text{enc}} = (g_{\text{enc}} \to p_1 : u_{\text{enc},1} + \ldots + p_n : u_{\text{enc},n})$ *and for all* $s \in M \times \mathbb{R}^k$*,* $v_{\text{num}}, v_{\text{den}} \in \mathbb{R}$*, we have*
  - $g_{\text{enc}} = g \times \mathbb{R}^2$*,*
  - *for all* i *with* $1 \leq i \leq n$ *we have* $u_{\text{enc},i}(s, v_{\text{num}}, v_{\text{den}}) = u_i(s) \times \{(0,0)\}$*,*
- *we have* $rew_{\text{enc,cts}}(\cdot,\cdot) \stackrel{\text{def}}{=} rew_{\mathtt{t},\text{cts}}(\cdot,\cdot) \stackrel{\text{def}}{=} 0$*,*
- *for each* $s = (m, v, v_{\text{num}}, v_{\text{den}})$ *and* $c_{\text{enc}} \in Cmds_{\text{enc}}$ *we have*

$$rew_{\text{enc,dis}}(s, c) \stackrel{\text{def}}{=} rew_{\text{dis}}(s, c) + v_{\text{num}} \text{ and } rew_{\mathtt{t},\text{dis}}(s, c) \stackrel{\text{def}}{=} v_{\text{den}}.$$

In this definition, rew_{enc} describes the rewards we obtain by executing a command plus the reward stored in the state space by timed transitions since the current mode was entered. This way, it can be used to substitute $[\![rew]\!]_{\text{num}}$ of Definition 7.19. The reward structure $rew_{\mathtt{t}}$ describes the time since a mode was entered, implying it can replace the denominator reward $[\![rew]\!]_{\text{den}}$.

As this transformation corresponds to a transformation of the PA semantics, it does not change the values of reward properties.

Lemma 7.24. *Consider a PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds)$ *with a reward structure* rew *and the reward-encoded variants* $\text{enc}(\mathcal{H}, rew) = (\mathcal{H}_{\text{enc}}, rew_{\text{enc}}, rew_{\mathtt{t}})$ *as well as a time restriction* $\mathbf{T}$*. For* $[\![rew_{\text{enc}}]\!] = ([\![rew_{\text{enc}}]\!]_{\text{num}}, [\![rew_{\text{enc}}]\!]_{\text{den}})$ *and* $[\![rew_{\mathtt{t}}]\!] = ([\![rew_{\mathtt{t}}]\!]_{\text{num}}, [\![rew_{\mathtt{t}}]\!]_{\text{den}})$ *we have*

$$\text{val}^{+}_{\mathcal{H},rew,\text{acc}} = \text{val}^{+,Cmds_{\text{enc}}}_{[\![\mathcal{H}_{\text{enc}},\mathbf{T}]\!],([\![rew_{\text{enc}}]\!]_{\text{num}},[\![rew_{\mathtt{t}}]\!]_{\text{num}}),\text{acc}} \text{ and}$$
$$\text{val}^{+}_{\mathcal{H},rew,\text{lra}} = \text{val}^{+,Cmds_{\text{enc}}}_{[\![\mathcal{H}_{\text{enc}},\mathbf{T}]\!],([\![rew_{\text{enc}}]\!]_{\text{num}},[\![rew_{\mathtt{t}}]\!]_{\text{num}}),\text{lra}},$$

and accordingly for the minimal values.

Proof. If in the semantical models we equate commands of $Cmds$ with corresponding ones of $Cmds_{\text{enc}}$, we have

$$([\![\mathcal{H}_{\text{enc}}, \mathbf{T}]\!], ([\![rew_{\text{enc}}]\!]_{\text{num}}, [\![rew_{\mathtt{t}}]\!]_{\text{num}})) = \text{enc}([\![\mathcal{H}, \mathbf{T}]\!], [\![rew]\!], Cmds),$$

and thus the statement follows from Lemma 7.22 and Lemma 7.6. □

Example 7.25. *The PHA of Figure 7.5 already contains the necessary variable t to encode the reward rew_{Error}. As we obtain a reward of 1 per time unit, it can be used to represent both time and rewards. We thus obtain the encoded model by defining*

$$rew_{\text{enc,dis}}((m,t,T),c) \stackrel{\text{def}}{=} \begin{cases} t & \text{if } m = \text{Error}, \\ 0 & \text{else} \end{cases} \quad \text{and } rew_{\mathsf{t},\text{dis}}((m,t,T),c) \stackrel{\text{def}}{=} t. \qquad \triangle$$

7.3 Abstraction

We can now equip our abstractions of PHAs with corresponding rewards structures.

Definition 7.26. *Let $\mathcal{H}$ be a PHA with rewards rew, and consider its encoded PHA with reward structures $(\mathcal{H}_{\text{enc}}, (rew_{\text{enc,dis}}, rew_{\text{enc,cts}}), (rew_{\mathsf{t},\text{dis}}, rew_{\mathsf{t},\text{cts}}))$ and an abstraction $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathcal{H}_{\text{enc}}, \mathbf{A}, \mathbf{T})$ of $\mathcal{H}_{\text{enc}}$. The* abstract upper-bound reward structure *is defined as* $\text{rabs}_{\text{up}}(\mathcal{H}, \mathcal{M}, rew) \stackrel{\text{def}}{=} (rew_{\text{num}}, rew_{\text{den}})$ *where*

- *for all $\mathbf{z} \in \mathbf{A}$ we have $rew_{\text{num}}(\mathbf{z}, \tau) \stackrel{\text{def}}{=} rew_{\text{den}}(\mathbf{z}, \tau) \stackrel{\text{def}}{=} 0$,*
- *for all $\mathbf{z} \in \mathbf{A}$ and $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ we have*

$$rew_{\text{num}}(\mathbf{z}, c) \stackrel{\text{def}}{=} \sup_{s \in \mathbf{z}} rew_{\text{enc,dis}}(s, c) \text{ and } rew_{\text{den}}(\mathbf{z}, c) \stackrel{\text{def}}{=} \inf_{s \in \mathbf{z}} rew_{\mathsf{t},\text{dis}}(s, c).$$

The abstract lower-bound reward structure rabs_{lo} *is defined accordingly by swapping* sup *and* inf*.*

We can use these reward structures to safely bound the reward values of PHAs semantics.

Theorem 7.27. *Consider a PHA $\mathcal{H}$ with reward structure rew and an abstraction $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathcal{H}_{\text{enc}}, \mathbf{A}, \mathbf{T})$ of its encoded PHA $\mathcal{H}_{\text{enc}}$, with the abstract upper-bound reward structure $rew_{\text{up}} \stackrel{\text{def}}{=} \text{rabs}_{\text{up}}(\mathcal{H}, \mathcal{M}, rew)$. Then*

$$\text{val}^{+}_{\mathcal{H},rew,\text{acc}} \leq \text{val}^{+,Cmds}_{\mathcal{M},rew_{\text{up}},\text{acc}} \text{ and } \text{val}^{+}_{\mathcal{H},rew,\text{lra}} \leq \text{val}^{+,Cmds}_{\mathcal{M},rew_{\text{up}},\text{lra}},$$

and accordingly for abstract lower-bound reward structures by changing $\leq$ to $\geq$.

Proof. We only consider the case $\text{val}^{+}_{\mathcal{H},rew,\text{lra}} \leq \text{val}^{+}_{\mathcal{M},rew_{\text{up}},\text{lra}}$ as the other cases are similar. Using Lemma 7.24, we assume that $\mathcal{H}$ is already reward encoded, and by Lemma 7.22 it suffices to show that

$$\text{val}^{+,Cmds}_{[\![\mathcal{H},\mathbf{T}]\!],[\![rew]\!],\text{lra}} \leq \text{val}^{+,Cmds}_{\mathcal{M},rew_{\text{up}},\text{lra}}.$$

For this, we can use the same simulation relation R as in the proofs of Theorem 3.31 and Theorem 3.32, with corresponding intermediate models. The additional requirements on the simulation concerning the reward values by Definition 7.8 are fulfilled (timed self loops get assigned rewards of 0), and so the result follows by Lemma 7.9. □

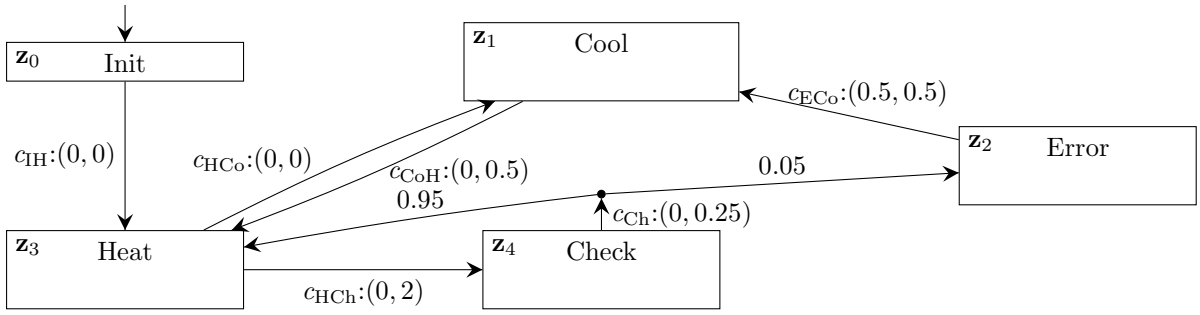


Figure 7.7: PHA abstraction for rewards.

Example 7.28. *In Figure 7.7 we sketch an abstraction of Figure 7.5 with an abstract upper-bound reward structure for the PHA reward structure* rew_{Error} *of Example 7.25. Thus, we have a reward of* 1 *per time in* Error *and* 0 *otherwise. In the abstract states, we left out constraints to restrict to states which are actually reachable. Consider the abstract state* $\mathbf{z}_4$*. As the mode of this state is* Check*, we obtain a reward of* 0 *when executing* c_{Ch}*. According to the guard of this command, we have to wait at least until* $t \geq 0.25$ *to execute it. Now consider* $\mathbf{z}_2$*. We can leave this state at point* $t = 0.25$*, and we thus obtain a reward and time of* 0.25*.* △

In case we are given affine reward structures, we can use more precise reward structures in the abstraction.

Definition 7.29. *Consider a PHA* $\mathcal{H}$ *with commands Cmds and an affine reward structure* $rew = (rew_{\text{dis}}, rew_{\text{cts}})$ *and an abstraction* $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathcal{H}, \mathbf{A}, \mathbf{T})$ *of* $\text{enc}(\mathcal{H}, rew)$*. We define the* affine abstraction

$$\mathcal{M}_{\text{aff}} \stackrel{\text{def}}{=} (\mathbf{A}, \overline{\mathbf{z}}, Cmds_{\text{num}} \uplus Cmds_{\text{den}} \uplus \{\tau_{\text{num}}, \tau_{\text{den}}\}, \mathcal{T}_{\text{aff}}),$$

where we have a one-to-one correspondence between actions a *and* a_{num} *as well as* a_{den}*, and define* $\mathcal{T}_{\text{aff}}$ *as* $\mathcal{T}_{\text{aff}}(s, a_{\text{num}}) \stackrel{\text{def}}{=} \mathcal{T}_{\text{aff}}(s, a_{\text{den}}) \stackrel{\text{def}}{=} \mathcal{T}(s, a)$*. Then, the* abstract affine reward structure *is defined as* $rew_{\text{aff}} = (rew_{\text{num}}, rew_{\text{den}})$ *where*

- *for all* $\mathbf{z} \in \mathbf{A}$ *we have*

$$rew_{\text{num}}(\mathbf{z}, \tau_{\text{num}}) \stackrel{\text{def}}{=} rew_{\text{num}}(\mathbf{z}, \tau_{\text{den}}) \stackrel{\text{def}}{=} rew_{\text{den}}(\mathbf{z}, \tau_{\text{num}}) \stackrel{\text{def}}{=} rew_{\text{den}}(\mathbf{z}, \tau_{\text{den}}) \stackrel{\text{def}}{=} 0,$$

- *for all* $\mathbf{z} \in \mathbf{A}$ *with mode* m *and* $c = (g \to p_1 : u_1 + \ldots + p_n : u_n) \in Cmds$ *we have*

$$rew_{\text{num}}(\mathbf{z}, c_{\text{num}}) \stackrel{\text{def}}{=} rew_{\text{dis}}(c) + rew_{\text{cts}}(m) t_{\text{sup}} \text{ and } rew_{\text{den}}(\mathbf{z}, c_{\text{num}}) \stackrel{\text{def}}{=} t_{\text{sup}},$$

$$rew_{\text{num}}(\mathbf{z}, c_{\text{den}}) \stackrel{\text{def}}{=} rew_{\text{dis}}(c) + rew_{\text{cts}}(m) t_{\text{inf}} \text{ and } rew_{\text{den}}(\mathbf{z}, c_{\text{den}}) \stackrel{\text{def}}{=} t_{\text{inf}},$$

with

$$t_{\text{sup}} \stackrel{\text{def}}{=} \sup\{v_{\text{den}} \mid (m, v, v_{\text{num}}, v_{\text{den}}) \in \mathbf{z}\} \text{ and } t_{\text{inf}} \stackrel{\text{def}}{=} \inf\{v_{\text{den}} \mid (m, v, v_{\text{num}}, v_{\text{den}}) \in \mathbf{z}\}.$$

We then define $\mathrm{rabs}_{\mathrm{aff}}(\mathcal{H}, \mathcal{M}, rew) \stackrel{\mathrm{def}}{=} (\mathcal{M}_{\mathrm{aff}}, rew_{\mathrm{aff}})$.

Theorem 7.30. *Consider a PHA* $\mathcal{H}$ *with reward structure rew and an abstraction* $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathcal{H}_{\mathrm{enc}}, \mathbf{A}, \mathbf{T})$ *of its encoded PHA* $\mathcal{H}_{\mathrm{enc}}$*, with the affine abstraction and reward structure* $\mathrm{rabs}_{\mathrm{aff}}(\mathcal{H}, \mathcal{M}, rew) = (\mathcal{M}_{\mathrm{aff}}, rew_{\mathrm{aff}})$*. Then*

$$\mathrm{val}^{+}_{\mathcal{H},rew,\mathrm{acc}} \leq \mathrm{val}^{+,Cmds}_{\mathcal{M}_{\mathrm{aff}},rew_{\mathrm{aff}},\mathrm{acc}} \text{ and } \mathrm{val}^{+}_{\mathcal{H},rew,\mathrm{lra}} \leq \mathrm{val}^{+,Cmds}_{\mathcal{M}_{\mathrm{aff}},rew_{\mathrm{aff}},\mathrm{lra}},$$

$$\mathrm{val}^{-}_{\mathcal{H},rew,\mathrm{acc}} \geq \mathrm{val}^{-,Cmds}_{\mathcal{M}_{\mathrm{aff}},rew_{\mathrm{aff}},\mathrm{acc}} \text{ and } \mathrm{val}^{-}_{\mathcal{H},rew,\mathrm{lra}} \geq \mathrm{val}^{-,Cmds}_{\mathcal{M}_{\mathrm{aff}},rew_{\mathrm{aff}},\mathrm{lra}}.$$

Proof. The reward structure $rew_{\mathcal{M}}$ is affine, as is the reward structure in the semantics of $\mathcal{H}_{\mathrm{enc}}$. The proof then follows along the lines of the one of Theorem 7.27 using Definition 7.11 and Lemma 7.12. □

Example 7.31. *If using an affine abstraction, we replace the actions of Figure 7.7 by*

$c_{\mathrm{IH},u}$:$(0,0)$, $c_{\mathrm{IH},l}$:$(0,0)$, $c_{\mathrm{CoH},u}$:$(0,2)$, $c_{\mathrm{CoH},l}$:$(0,0.5)$, $c_{\mathrm{ECo},u}$:$(0.5,0.5)$, $c_{\mathrm{ECo},l}$:$(0.5,0.5)$, $c_{\mathrm{HCo},u}$:$(0,3)$, $c_{\mathrm{HCo},l}$:$(0,0)$, $c_{\mathrm{Ch},u}$:$(0,0.5)$, $c_{\mathrm{Ch},l}$:$(0,0.25)$. △

7.3.1 Computing Abstractions

We have already discussed in Subsection 3.3.1 how we can obtain abstractions of PHAs. What remains is the computation of the reward-encoded PHAs and the one of the reward structures of the abstraction.

Obtaining the reward-encoded PHAs is simple: we just have to add two new variables in the PHA, one for the continuous reward function and one for the time component, according to Definition 7.23. We then use the specification language of the hybrid solver used to specify the behaviour of these additional variables. That is, we have to reset them to zero on mode changes and take care that they increase their value during timed transitions, for instance by using differential equations. We can then compute the abstraction from the induced HA as before.

It is a bit more complicated to obtain the abstract reward structures. In case we only have rewards which depend on the commands and which are constant for each state, the reward values can be obtained directly from the abstraction of the induced HA. Otherwise, we have to take a deeper look at the abstract states. According to Definition 7.26 and Definition 7.29, we have to find suprema and infima of the variables for rewards and time. How this can be done depends on the hybrid solver used. We consider the case of PHAVer which we use throughout the thesis. This solver uses polyhedra to represent abstract states, which can be represented as sets of linear inequations. Because of this, we can use a linear programming tool to find the minimal and maximal values of reward and time variables.

If we do not have timed rewards and only a constant reward value for each command and want to consider the expected accumulated reward, we do not need to consider the encoded PHA, as we need not keep track of continuous rewards or time. Also, for

affine abstractions, we only need a variable to remember the time since a mode change, because we can compute the rewards from these values; in Definition 7.29 the values v_{num} are not used.

7.4 Algorithmic Consideration

After we have obtained finite abstractions and have computed the according reward structure using linear programming, it remains to compute expected accumulated or long-run average rewards in the abstraction. As in Section 3.4, we briefly state how such values can be obtained.

We firstly restate some results for accumulated rewards [Put94; FV96] (given for the slightly different model of Markov decision processes). In addition to the induced MCs we have used in Section 3.4, here we will also need to consider induced reward structures on these MCs.

Definition 7.32. *Let $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ be a PA and consider a reward structure $rew = (rew_{\text{num}}, rew_{\text{den}})$ with $rew_{\text{num}}, rew_{\text{den}} \colon (S \times Act \times Distr(S)) \to \mathbb{R}_{\geq 0}$ and a simple scheduler $\sigma \colon S \to (Act \times Distr(S))$. The* MC reward structure *$rew_\sigma = (rew_{\sigma,\text{num}}, rew_{\sigma,\text{den}})$ with $rew_{\sigma,\text{num}}, rew_{\sigma,\text{den}} \colon S \to \mathbb{R}_{\geq 0}$* induced *by σ in $\mathcal{M}$ is defined so that for $s \in S$ we have*

$$rew_{\sigma,\text{num}}(s) \stackrel{\text{def}}{=} rew_{\text{num}}(s, \sigma(s)) \text{ and } rew_{\sigma,\text{den}}(s) \stackrel{\text{def}}{=} rew_{\text{den}}(s, \sigma(s)).$$

Given an MC and a reward structure for this MC, we can specify their accumulated reward.

Definition 7.33. *Given an MC $\mathcal{M} = (S, \overline{s}, \mathcal{T})$ and an MC reward structure $rew = (rew_{\text{num}}, rew_{\text{den}})$, we define*

$$\text{vals}_{\mathcal{M},rew,\text{acc}} \colon S \to \mathbb{R}_{\geq 0}$$

as the smallest nonnegative solution of the equation system where for all $s \in S$ we have

$$v(s) = rew_{\text{num}}(s) + \sum_{s' \in S} \mathcal{T}(s)(s') v(s').$$

Then, we can use Algorithm 7.1 to compute minimal expected accumulated rewards on finite PAs. As in Section 3.4, $\text{vals}^{-}_{\mathcal{M},rew,\text{acc}}(s)$ is the value in case s were the initial state. The basic principle of the algorithm is very similar to Algorithm 3.3: an initial scheduler is chosen by MINACCINIT, which is then improved until no further improvements are possible. As in Algorithm 3.3, we have to perform a graph-based search to find states for which the expected accumulated reward is 0 when starting there. If we want to compute the minimal values instead, we have to take care about the initial scheduler, similarly to what has been done in the algorithm for reachability. As Algorithm 3.3, that algorithm does not take fairness into account, but we can still apply it for the same reasons as discussed in Section 3.4. Other algorithms based on value iterations or linear programming also exist.

In case we want to compute long-run average values, there are also algorithms using linear programming [Alf97; EJ11] or policy iteration [EJ12].

input : finite PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ with reward structure $rew = (rew_{\text{num}}, rew_{\text{den}})$.
output: $(\text{vals}^{-}_{\mathcal{M},rew,\text{acc}}, \sigma)$ with $\sigma \in Sched^{\text{simple}}_{\mathcal{M}}$ and $\text{vals}^{-}_{\mathcal{M},rew,\text{acc}} = \text{vals}^{\sigma}_{\mathcal{M},rew,\text{acc}}$.

$\sigma' :=$ MINACCINIT($\mathcal{M}$)
repeat
  $\sigma := \sigma'$
  $v := \text{vals}_{\mathcal{M}_\sigma, rew_\sigma, \text{acc}}$
  forall the $s \in S$ **do**
    $A(s) := \text{argmin}_{\substack{(a,\mu),\\ \mu \in \mathcal{T}(s,a)}} rew_{\text{num}}(s,a,\mu) + \sum_{s' \in S} \mu(s')v(s')$
    if $\sigma(s) \in A(s)$ **then** $\sigma'(s) := \sigma(s)$ **else** choose $\sigma'(s)$ from $A(s)$
until $\sigma' = \sigma$
return (v, σ)

Algorithm 7.1: MINACC($\mathcal{M}, rew$).

constraint length	PHAVER (s)	states	time until error			commands until error		
			constr. (s)	ana. (s)	result	constr. (s)	ana. (s)	result
–	0	7	0	0	0.0000	0	0	42.0000
1	1	102	0	0	45.3347	0	0	42.0000
0.5	0	288	0	0	53.4934	0	1	42.0000
0.1	18	5909	1	8	61.3732	1	14	50.7820
0.05	151	24593	6	65	62.0170	1	103	48.8410
0.04	296	37511	10	154	62.0633	1	189	48.8410
0.03	2566	93979	25	584	62.7227	2	722	50.7820
0.02	6628	156864	44	1509	62.6245	5	1867	50.7820

Table 7.1: Accumulated rewards in thermostat.

7.5 Case Studies

We implemented the analysis methods to compute expected accumulated and long-run average rewards in our tool PROHVER, and have applied them on two of the case studies. As before, experiments were run on an Intel(R) Core(TM)2 Duo CPU with 2.67 GHz and 4 GB RAM.

7.5.1 Thermostat

The first properties we consider for our running thermostat example are the minimal expected time and the expected number of commands (except the one from Init to Heat) it takes until the Error mode is reached. For this, we consider the original thermostat model of Figure 3.3, with the difference that we no longer need the global time c. Instead, we add reward structures, where the one for the expected number of commands agrees with the one of Example 7.15. To improve the quality of the overapproximation, we apply the PHAVER-specific constraint on the variable T modelling the temperature.

Results are given in Table 7.1. We provide the constraint lengths, the time PHAVER needed to construct the model and the number of states. Because we use the same abstraction of the PHA to compute both accumulated and long-run average rewards, these values are the same for both properties. The runtime of PROHVER is no longer negligible. We give the time this tool needed to compute the reward by using linear programming on the polyhedra built up by PHAVER, and the time needed to compute the result value once the model with its reward structure is available. We also state the final result rounded to four decimal digits.

PHAVER computes more abstract states than for the larger time bounds of Table 3.3, though, because of the differences in the property type, the number of continuous variables and analysis settings, those results are hard to compare. The time needed in terms of the runtime of PHAVER and linear programming is relatively low, taking into account the large number of abstract states.

Initially, we performed these experiments with constraints on the local time t. Doing so, we obtained much worse reward bounds, indeed always 42 for the expected number of commands until Error. Manual analysis showed that this refinement only excluded the first direct transition from Heat to Check without a previous visit to Cool. The resulting lower bound on the expected number of commands therefore is

$$\begin{aligned} & rew_{\text{num}}(c_{\text{HCo}}) + rew_{\text{num}}(c_{\text{CoH}}) + (rew_{\text{num}}(c_{\text{HCh}}) + rew_{\text{num}}(c_{\text{ChH}}))\frac{1}{0.05} \\ &= 1 + 1 + (1 + 1)\frac{1}{0.05} \\ &= 42. \end{aligned}$$

Table 7.1 shows that this is not the ultimate answer to the minimal expected number of executed commands, which means that also at later points of time a direct transition from Heat to Check without visiting Cool in between is not always possible.

The time needed to construct the reward structures for the expected number of commands is much lower than the one for the expected time, because in the first case the reward is constant, and we thus can avoid solving a large number of linear programming problems. The time for the analysis is higher, though.

The analysis of the abstract model needed considerably more time than the one for reachability in Subsection 3.5.1. In the abstract models which we obtain, there are often large chains of states connected only by timed transitions. Techniques building on bisimulation-like methods could in the future speed up the analysis by taking this fact into account. This might be necessary for larger case studies.

Next, we consider the maximal expected time spent in the Error mode of the modified thermostat model in Figure 7.5. We also provide results when using the more precise method taking advantage of the fact that the reward structure is affine by using a reward structure as in Example 7.31. Results are given in Table 7.2. As seen, using affine reward structures does not improve the result: this was expected by comparing Example 7.28 and Example 7.31. Some of the actions of Example 7.31 have the same values as actions in Example 7.28 (e.g. $c_{\text{CoH},l} : (0, 0.5)$ corresponds to $c_{\text{CoH}} : (0, 0.5)$), while the remaining

constraint length	PHAVer (s)	states	time in error			time in error (lin)		
			constr. (s)	ana. (s)	result	constr. (s)	ana. (s)	result
1	0	126	0	0	0.0131	0	0	0.0131
0.5	1	382	0	0	0.0111	1	0	0.0111
0.1	28	7072	4	9	0.0094	6	14	0.0094
0.05	185	29367	8	124	0.0093	25	313	0.0093

Table 7.2: Long-run average rewards in thermostat.

constraint length	PHAVer (s)	states	time until error			commands until error		
			constr. (s)	ana. (s)	result	constr. (s)	ana. (s)	result
–	1	10	0	0	0.0000	0	0	40.0000
1	0	53	0	0	137.0320	0	0	40.0000
0.5	0	95	0	0	150.1600	0	0	40.0000
0.1	3	411	0	0	163.0980	0	1	40.0000
0.05	5	814	1	1	164.8010	0	2	40.0000
0.02	15	2151	0	11	165.7080	0	11	40.0000
0.01	31	4294	1	42	166.0560	0	46	40.0000
0.005	74	8580	1	167	166.2350	0	186	40.0000

Table 7.3: Accumulated rewards in water level control.

ones are suboptimal choices for a maximising scheduler (e.g. $c_{\mathrm{CoH},u} : (0, 2)$). Thus, a scheduler in the affine abstraction basically takes the same choices as in the previous abstraction.

By the definition of affine simulation, it is necessary to solve twice as many linear optimisation problems to obtain an abstraction as when using an abstract upper-bound reward structure, so that the time needed by ProHVer is greater.

7.5.2 Water Level Control

Similarly to the thermostat case, for our water level control model (cf. Subsection 3.5.3) we considered the minimal expected time and number of command executions until the Error mode is reached. As this model features only affine continuous dynamics, we successfully obtained results using constraints on the local timers t. Results are given in Table 7.3.

For the second property, we remove the error mode and assume that the operational bounds of the system are safe. We are interested in the average energy consumption of the system. We assume that no energy is spent in the modes where the pump is switched off. While the pump is running, 2 units of energy are consumed per time unit, starting the pump takes 10 units of energy and switching it off again takes 6 units. The reward structure thus features both rewards obtained from timed transitions as well as those obtained from command executions. Results are given in Table 7.4. As seen, using affine reward structures improves the result more than in the thermostat case. This happens

constraint length	PHAVer (s)	states	average energy			average energy (lin)		
			constr. (s)	ana. (s)	result	constr. (s)	ana. (s)	result
1	0	51	0	0	1.9854	0	0	1.8145
0.5	0	93	0	0	1.7995	0	0	1.7069
0.1	2	409	0	0	1.6564	0	0	1.6397
0.05	4	812	0	0	1.6395	0	0	1.6314
0.02	11	2149	0	0	1.6303	1	0	1.6268
0.01	24	4292	0	1	1.6268	2	1	1.6250
0.005	58	8578	2	4	1.6250	2	7	1.6242

Table 7.4: Long-run average rewards in water level control.

because in a larger percentage of the time a nonzero reward in the numerator is obtained. In the thermostat case study, this was only possible in mode Error. As the time spent in Error in the thermostat is also rather small, the difference between the lower and upper values with the two kinds of reward abstractions was rather small.

7.6 Related Work

Reward-based properties for (nonstochastic) timed automata have been considered by Bouyer et al. [BM07; Bou+07]. Jurdziński et al. [JLR09; RLJ11] consider a problem of controller synthesis for average-reward properties in classical HAs. The discrete-time stochastic hybrid automata by Abate et al. (cf. also Section 4.5) can be used for the analysis of reward-based properties as sketched e.g. in [TA12]. Methods which approximate continuous-time stochastic hybrid automata by Markov chains [PH06; HLS00] also allow for an extension to reward-based properties. Another related work is by Fränzle et al. [FTE10b], which considers approximations of the *step-bounded* expected accumulated reward. To the best of our knowledge, this thesis is the first work to address reward-based properties of PHAs involving nondeterminism, continuous-time as well as stochastic behaviour by building upon well-understood and effective abstraction methods for classical hybrid automata.

For the underlying PAs, solution methods for various reward-based properties exist, see e.g. [Put94] for an overview. Fractional long-run average properties have been discussed by de Alfaro [Alf97] and Jobstmann et al. [EJ11; EJ12].

7.7 Conclusion

We have extended our abstraction framework to handle properties of PHAs based on rewards, thus considerably enlarging the class of properties we are able to analyse. For this, we firstly had to consider the semantical model of PAs. We defined reward structures, costs or bonuses associated with taking a transition in a PA. We then defined properties, reasoning over the expected total or long-run average reward with a given

reward structure. We also extended probabilistic simulation relations, thus maintaining the correctness of reward-based properties rather than just probabilistic reachability. Then, we have decorated PHAs with reward structures, have discussed how these structures carry over to the semantical model and how properties of PHAs are defined. To allow for the automatic analysis of reward-based properties, we extended our framework accordingly. In case we have rewards depending on the time, as for instance the average time the system is operational, it was necessary to take a closer look at the form of the abstract states, to find out about minimal or maximal reward values. We compared our conceptual approach to similar ones from the literature. The effectiveness of our approach was demonstrated by using it to compute reward-based values on two of our case studies.

8 Orthogonal Combinations

In this chapter, we discuss orthogonal combinations of the models and techniques developed in the preceding four chapters. This way, we can handle very general models and properties. In the following, we will consider each pair of two different extensions and describe what has to be done to integrate them. The integration of more than two of the extensions is then straightforward.

8.1 Continuous Distributions and Partial Control

If we combine continuous distributions with partial control, the semantic model of an SHA $\mathcal{H}$ is still an NLMP

$$\mathcal{M} = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T}),$$

as is Definition 4.14 on page 83. However, we interpret it as a two-player game with continuous distributions. In this setting, we can define strategies similarly to Definition 5.1 on page 104. However, we have to take care to apply restrictions to enforce that the joint scheduler of Definition 5.1 is a measurable function, as required in Definition 4.5 on page 75. For this, we can define the set of *measurable player*-con *strategies* $Strat^{\text{con}}_{\mathcal{M},\text{meas}}$ as functions of the form

$$\sigma_{\text{con}} \colon Path^{\text{fin}}_{\mathcal{M}} \to \Delta(Act),$$

for which for all $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ we have

$$\sigma_{\text{con}}(\beta)(\{a \in Act \mid \nexists \mu \in \Delta(S).\ (a,\mu) \in \mathcal{T}(\text{last}(\beta))\}) = 0,$$

which are additionally required to be $\Sigma_{Path^{\text{fin}}_{\mathcal{M}}}$-$\mathcal{H}(\Delta(Act))$-measurable. The *measurable player*-env *strategies* $Strat^{\text{env}}_{\mathcal{M},\text{meas}}$ are then defined as the set of functions

$$\sigma_{\text{env}} \colon (Path^{\text{fin}}_{\mathcal{M}} \times Act) \to \Delta(\Delta(S)),$$

where for all $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ and $a \in Act$ for which there is $\mu \in \Delta(S)$ with $(a,\mu) \in \mathcal{T}(\text{last}(\beta))$ we have

$$\sigma_{\text{env}}(\beta, a)(\{\mu \in \Delta(S) \mid \nexists a \in Act.\ (a,\mu) \in \mathcal{T}(\text{last}(\beta))\}) = 0,$$

and which are $(\Sigma_{Path^{\mathrm{fin}}_{\mathcal{M}}} \otimes \mathcal{H}(Act))$-$\mathcal{H}(\Delta(\Delta(S)))$-measurable.
We can then define the *joint scheduler* $\mathrm{join}(\sigma_{\mathrm{con}}, \sigma_{\mathrm{env}}) \in Sched_{\mathcal{M}}$ of $\sigma_{\mathrm{con}} \in Strat^{\mathrm{con}}_{\mathcal{M},\mathrm{meas}}$ and $\sigma_{\mathrm{env}} \in Strat^{\mathrm{env}}_{\mathcal{M},\mathrm{meas}}$, where for $\beta \in Path^{\mathrm{fin}}_{\mathcal{M}}$ and $A \subseteq Act \times \Delta(S)$ we have

$$\mathrm{join}(\sigma_{\mathrm{con}}, \sigma_{\mathrm{env}})(\beta)(A) \stackrel{\mathrm{def}}{=} \int_{Act} \sigma_{\mathrm{env}}(\beta, a)(A_{|a})\, \sigma_{\mathrm{con}}(\beta)(\mathrm{d}a).$$

We further change the definition of the maximal value $\mathrm{val}^{+,-}_{\mathcal{M},Reach}$ which a controller can enforce against any malevolent environment in Definition 5.3 of page 104 to

$$\mathrm{val}^{+,-}_{\mathcal{M},Reach} \stackrel{\mathrm{def}}{=} \sup_{\sigma_{\mathrm{con}} \in Strat^{\mathrm{con}}_{\mathcal{M},\mathrm{meas}}} \inf_{\sigma_{\mathrm{env}} \in Strat^{\mathrm{env}}_{\mathcal{M},\mathrm{meas}}} \mathrm{val}^{\sigma_{\mathrm{con}},\sigma_{\mathrm{env}}}_{\mathcal{M},Reach}.$$

The definition of the case $\mathrm{val}^{-,+,Act_{\mathrm{fair}}}_{\mathcal{M},Reach}$ in which the controller minimises is similar, but we still have the requirement that σ_{con} is only chosen from strategies where for all $\sigma_{\mathrm{env}} \in Strat^{\mathrm{env}}_{\mathcal{M},\mathrm{meas}}$ the joint scheduler is Act-fair.
Analogously to Definition 4.5, we define *finitely probabilistic player*-con *strategies* σ_{con} by requiring that for each $\beta \in Path^{\mathrm{fin}}_{\mathcal{M}}$ there is a finite number of actions $a_1, \ldots, a_n \in Act$ with

$$\sigma_{\mathrm{con}}(\beta)(\{a_1, \ldots, a_n\}) = 1.$$

For *finitely probabilistic player*-env *strategies* σ_{env} we require that for each $\beta \in Path^{\mathrm{fin}}_{\mathcal{M}}$ and $a \in Act$ there is a finite number of distributions $\mu_1, \ldots, \mu_n \in \Delta(S)$ with

$$\sigma_{\mathrm{con}}(\beta, a)(\{\mu_1, \ldots, \mu_n\}) = 1.$$

Semi finitely probabilistic strategies are defined in the same manner. After we abstract the measurable continuous guarded commands as in Definition 4.20 on page 88 by a family of command abstractions $\mathbf{F}$, we implicitly have three types of nondeterminism: the nondeterminism of the controller in the original hybrid system, the one of the environment, and the one resulting from the abstraction of continuous distributions. However, in the current method of abstraction, the latter two types of nondeterminism are indistinguishable and are thus automatically combined. Because of this, we end up with a two-player game in which the controller player plays against the environment and the abstraction. Thus, the abstraction is in favour of the environment player. Similarly to Theorem 4.22 on page 89, the environment player can simulate the continuous distributions, and thus for each strategy of the original model it can construct an equivalent strategy of the abstract model. It might be able to choose a more powerful one, which was not possible in the original model. The controller has the same choices as before. In turn, the abstraction is advantageous for the environment, and thus for an unsafe or desirable mode m_{Reach} of $\mathcal{H}$ we have

$$\mathrm{val}^{+,-}_{\mathcal{H},m_{Reach}} \geq \mathrm{val}^{+,-}_{\mathrm{abs}(\mathcal{H},\mathbf{F}),m_{Reach}} \text{ and } \mathrm{val}^{-,+}_{\mathcal{H},m_{Reach}} \leq \mathrm{val}^{-,+}_{\mathrm{abs}(\mathcal{H},\mathbf{F}),m_{Reach}}.$$

Similar to Lemma 4.10 at page 78, one can show that the environment player can obtain its optimal value by only using finitely probabilistic strategies. Because of this, the result

also holds if we restrict to this class of strategies, which are the ones also valid in the interpretation of finitely probabilistic NLMPs as PAs.

If we then apply the abstraction of Section 5.3 on page 107, the abstraction is again in favour of the environment player. Because of this, if we establish a certain bound in the abstraction of such a system, the controller is guaranteed to achieve at least this bound in the original semantics.

8.2 Continuous Distributions and Parametric Models

We can consider PHAs which are parametric and also feature continuous distributions. What we have to do is to extend the definition of post operators, measurable finitely-probabilistic guarded commands, and measurable continuous guarded commands in Definition 4.18 on page 87. In this extension, moments of the continuous distributions, for instance their mean or variation, can also be given as parameters rather than constants. If we do so, we can define a *parametric SHA (PSHA)* as a model of the form

$$\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}}, Cmds_{\text{cts}}, V),$$

where the parameters V are as in Definition 6.8 on page 126. The commands $Cmds_{\text{fin}}$ are as the commands $Cmds$ of Definition 6.8. Continuous commands $c \in Cmds_{\text{cts}}$ are of the form $g \to M$ where g is as for finite commands, and $M \colon S \to (Evals(V) \to \Delta(S))$ is a function mapping states to parametric probability distributions. We define the set of valid evaluations $Evals(\mathcal{H})$ as in Definition 6.8. The post operators are required to fulfil the measurability restrictions of Definition 4.11 on page 80, and for each valid evaluation $v \in Evals(\mathcal{H})$ and command $c \in Cmds_{\text{fin}} \uplus Cmds_{\text{cts}}$ we must have that the induced nonparametric guarded command fulfils Definition 4.11.

As in Definition 6.9 on page 126, the set of nonparametric models a parametric SHA represents, are the models obtained by inserting the parameter values in both finitely-probabilistic as well as continuous guarded commands.

To analyse such a model, we have to use an adapted version of the methods described in Section 4.3 on page 87. Afterwards, we have a PPHA which overapproximates the original PSHA, and can thus continue with the abstraction scheme on Section 6.3 on page 127, the validity of which we have already shown before. The part we have to adapt is Definition 4.18 on page 87. Here, we overapproximated a continuous command c by a finitely-probabilistic guarded command $\text{abs}(c, \mathbf{f}) = (g \to p_1 : u_1 + \ldots + p_n : u_n)$. Being in state s, we divided the possible successor states of the commands into a number of sets $u_i(s)$. The finitely-probabilistic guarded command was then defined in such a way that each of the set is chosen by its probability p_i, but the successor within the set was selected nondeterministically. For the parametric setting, we abstract parametric continuous commands to parametric finitely-probabilistic commands. For the correctness to hold, we thus have to make sure that for each $v \in Evals(\mathcal{H})$ the probabilities p_i are chosen to guarantee that the finitely-probabilistic command induced by v overapproximates the continuous command induced by v.

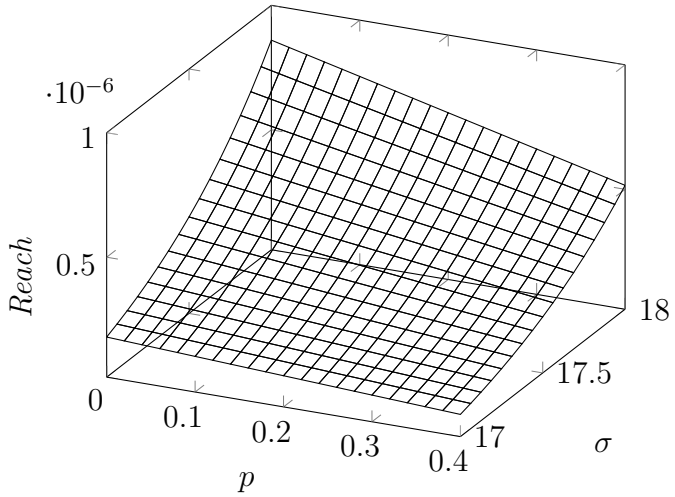


Figure 8.1: Bounds for probability of crashing as a function of probability of movement authority loss p and standard deviation σ of distance measurement. A time bound of 100s and Abstraction A was used.

As an example of this combination, reconsider the moving-block train controller case study from Subsection 4.4.3 on page 96. The graph in Figure 8.1 displays the expected positive correlation between measurement error (standard deviation) and risk, but also the effectiveness of the fault-tolerance mechanism handling communication loss. We see that crashes due to communication losses are effectively avoided, rooted in the principle of maintaining the last received movement authority whenever no fresh authority is at hand. In fact, risk correlates negatively with the likelihood of communication loss: if no new movement authority is received, the follower train cannot proceed further to the next block, and will thus avoid crashing into the leader train.

The model we analysed here is thus a parametric SHA with two parameters, the loss probability p and the standard deviation σ. To obtain the graph, we applied (a straightforward extensions of) the abstraction mechanism of Chapter 4 to obtain a parametric PHA from the original parametric SHA. The function correlating risk to measurement error and probability of communication loss has been computed using the methods of Chapter 6. This way, we were able to save a lot of time. Without using the parametric method here, we would have had to perform a separate PHAVer run for each of the 357 points used to draw the graph, each of which already takes several minutes, as can be seen from Table 4.3 on page 98.

8.3 Continuous Distributions and Rewards

If we want to consider reward-based properties of an SHA

$$\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds_{\text{fin}}, Cmds_{\text{cts}}),$$

at first we have to ensure that the value of interest is measurable in the NLMP semantics

$$\mathcal{M} = (S, \Sigma_S, \overline{s}, Act, \Sigma_{Act}, \mathcal{T}).$$

We discuss the case of long-run average rewards, the other one is similar but simpler. Given a scheduler σ and a reward structure $(rew_{\text{num}}, rew_{\text{den}})$ of $\mathcal{M}$, for the fractional long-run average reward of Definition 7.3 on page 142 we have

$$\text{val}^{\sigma}_{\mathcal{M},rew,\text{lra}} \stackrel{\text{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}{\sum_{i=0}^{n} rew_{\text{den}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}\right].$$

If the reward functions $rew_{\text{num}}, rew_{\text{den}} \colon (S \times Act \times \Delta(S)) \to \mathbb{R}_{\geq 0}$ are $(\Sigma_S \otimes \Sigma_{Act} \otimes \Delta(\Sigma_S))$-$\mathcal{B}(\mathbb{R}_{\geq 0})$-measurable, we can derive the measurability of

$$\lim_{n\to\infty} \liminf_{m\to\infty} \frac{\sum_{i=n}^{m} rew_{\text{num}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}{1 + \sum_{i=n}^{m} rew_{\text{den}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}. \tag{8.1}$$

As in the discussion of Definition 7.3 on page 142, we then assume σ is chosen so that

$$\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}{\sum_{i=0}^{n} rew_{\text{den}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}$$

exists for a set of paths with probability 1. This value agrees with the one of Equation 8.1, which means that $\text{val}^{\sigma}_{\mathcal{M},rew,\text{lra}}$ exists. We remark that, in case of PAs, we would in principle also have to take into account measurability considerations. However, we only had to show that the reward functions are $(2^S \otimes 2^{Act} \otimes 2^{Distr(S)})$-$\mathcal{B}(\mathbb{R}_{\geq 0})$-measurable, which is trivially the case.

To ensure the measurability of the semantics, we have to show that the reward structures of $\mathcal{H}$ as in Definition 7.14 on page 151 are measurable. Because we can encode continuous reward functions in discrete reward functions (cf. Definition 7.23 on page 155), we only consider the latter form. Let $Cmds \stackrel{\text{def}}{=} Cmds_{\text{fin}} \uplus Cmds_{\text{cts}}$. To guarantee the measurability of rew_{num} and rew_{den}, we have to make sure that $rew_{\text{dis}} \colon ((M \times \mathbb{R}^k) \times Cmds) \to \mathbb{R}_{\geq 0}$ is $(\Sigma_S \otimes 2^{Cmds})$-$\mathcal{B}(\mathbb{R}_{\geq 0})$-measurable where $\Sigma_S = 2^M \otimes \mathcal{B}(\mathbb{R}^k)$. Because there are only finitely many modes and commands, this is equivalent to showing that for each $m \in M$ and $c \in Cmds$ we have that $rew_{\text{dis}}(m, \cdot, c) \colon \mathbb{R}^k \to \mathbb{R}_{\geq 0}$ is $\mathcal{B}(\mathbb{R}^k)$-$\mathcal{B}(\mathbb{R}_{\geq 0})$-measurable. Similarly to the discussion after Definition 4.11 on page 80, we can argue that this measurability requirement is fulfilled when formulating the rewards in common mathematical theories, for instance o-minimal ones.

To perform the analysis, one can perform the same abstraction as in Section 4.3 on page 87 to obtain a PHA. With this PHA and the same reward structure as for the original SHA, we can then use the solution methods for PHAs of Section 7.3 on page 156 to obtain valid bounds, the correctness of which we have already shown before. To show that the abstraction from SHAs to PHAs is correct, we have to extend Lemma 4.10 on page 78. This lemma states that finitely probabilistic schedulers, which are the ones valid in PAs, suffice to obtain optimal reachability probabilities, even under fairness.

input : finite PPA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$, state set $Reach \subseteq S$, region r.
output: either ? or $(\text{vals}^{+,-}_{\mathcal{M},Reach}, \sigma_{\text{con}}, \sigma_{\text{env}})$ with $\sigma_{\text{con}} \in Strat^{\text{con,simple}}_{\mathcal{M}}$ and $\sigma_{\text{env}} \in Strat^{\text{env,simple}}_{\mathcal{M}}$ and for all $v \in Evals(\mathcal{M}) \cap r$ and $s \in S$ we have $\text{vals}^{+,-}_{\mathcal{M},Reach}(s)\langle v\rangle = \text{vals}^{\text{join}(\sigma_{\text{con}},\sigma_{\text{env}})}_{\mathcal{M},Reach}(s)\langle v\rangle$ $= \inf_{\sigma'_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{vals}^{\sigma_{\text{con}},\sigma'_{\text{env}}}_{\mathcal{M},Reach}(s)\langle v\rangle = \sup_{\sigma'_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}}} \text{vals}^{\sigma'_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M},Reach}(s)\langle v\rangle$.

$c := \text{centre}(r)$
if $c \notin Evals(\mathcal{M})$ **then return** ?
$(\_, \sigma_{\text{con}}, \sigma_{\text{env}}) :=$ maxMinValSched$(\mathcal{M}_c, Reach)$ `/* cf. Algorithm 5.1 */`
$v :=$ reachParam$(\mathcal{M}_{\text{join}(\sigma_{\text{con}},\sigma_{\text{env}})}, Reach)$ `/* cf. Algorithm 6.1 */`
$valid := true$
forall the $s \in S \setminus Reach, a \in Act$ *with* $\mathcal{T}(s,a) \neq \emptyset$ **do**
  $v_{\text{cmp}}(s) := \sum_{s' \in S} \sigma_{\text{env}}(s,a)(s')v(s')$
  $valid := valid \wedge$ check$(v(s) \geq v_{\text{cmp}}(s), r)$ `/* cf. Definition 6.23 */`
forall the $s \in S \setminus Reach, \mu \in \mathcal{T}(s, \sigma_{\text{con}}(s))$ **do**
  $v_{\text{cmp}}(s) := \sum_{s' \in S} \mu(s')v(s')$
  $valid := valid \wedge$ check$(v(s) \leq v_{\text{cmp}}(s), r)$ `/* cf. Definition 6.23 */`
if $valid$ **then return** $(v, \sigma_{\text{con}}, \sigma_{\text{env}})$
else return ?

Algorithm 8.1: maxMinReachParamRegion$(\mathcal{M}, Reach, r)$.

To adapt the proof, we basically have to replace the reachability probabilities within the inequations connecting semantics and abstractions by the expected reward values. Theorem 4.22 on page 89 does not need to be modified much. Its proof shows that we can simulate any stochastic behaviour of the original model by the abstract model, which also includes reward-based properties.

8.4 Partial Control and Parametric Models

The definition of a parametric PHA $\mathcal{H}$ with partial control is the same as in Definition 6.8 on page 126, because the definition of nonparametric PHAs with partial control is the same as the one for PHAs without control in Definition 3.20 on page 46. A parametric game abstraction $\mathcal{M}$ can then be defined similarly to the nonparametric game-based abstraction in Definition 5.8 on page 107. The main difference is, that we use parametric probability distributions in the same way as in Definition 6.13 on page 127. Given an evaluation $v \in Evals(\mathcal{H})$, the nonparametric automaton induced by v on $\mathcal{H}$ is then abstracted by the induced abstraction of $\mathcal{M}$ by v. The correctness for these nonparametric models follows by Theorem 5.16 on page 113.

To compute functions which bound the mutually optimal values in the abstraction, we have to adapt Algorithm 6.2 on page 133. This algorithm computes a function which depends on the model parameters and represents the optimal value for an entire parameter region. Using techniques from Algorithm 5.1 on page 118, we have to adapt

Algorithm 6.2 to compute the mutually optimal values rather than the minimal one. We give a sketch of the modified algorithm in Algorithm 8.1. As Algorithm 6.3, the algorithm operates on regions of parameter evaluations. In the first two lines it computes the centre evaluation v of the region. Then, in Line 3, it uses Algorithm 5.1 to compute the player-con and player-env strategies $\sigma_{\text{con}}, \sigma_{\text{env}}$ which are mutually optimal for v, in contrast to Algorithm 6.2 which uses Algorithm 3.3 on page 63. In Line 4, it computes the reachability values obtained if the two players play according to $\sigma_{\text{con}}, \sigma_{\text{env}}$. In lines 6 to 8 the algorithm verifies that these strategies are mutually optimal for the entire parameter region. As in Algorithm 6.3, we might find out that we cannot find mutually optimal strategies for the entire region, and thus the algorithm might return ?, which means that when returning to Algorithm 6.3 the region will be split further.

8.5 Partial Control and Rewards

It is possible to analyse PHAs with partial control for reward-based properties. To do so, we have to adapt the definition of mutually optimal values in PAs with a given reward structure in Definition 5.3 on page 104 to the reward-based value of Definition 7.3 on page 142. The definition of player-con and player-env strategies, fairness and so on can stay the same. If we want to synthesise a controller which maximises the long-run average rewards, the objective becomes

$$\text{val}^{+,-}_{\mathcal{M},rew,\text{lra}} \stackrel{\text{def}}{=} \sup_{\sigma_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}}} \inf_{\sigma_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M},rew,\text{lra}} \text{ where } \text{val}^{\sigma_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M},rew,\text{lra}} \stackrel{\text{def}}{=} \text{val}^{\text{join}(\sigma_{\text{con}},\sigma_{\text{env}})}_{\mathcal{M},rew,\text{lra}},$$

with

$$\text{val}^{\sigma}_{\mathcal{M},rew,\text{lra}} \stackrel{\text{def}}{=} E_{\mathcal{M},\sigma}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}{\sum_{i=0}^{n} rew_{\text{den}}(X_i^{\mathcal{M}}, Y_i^{\mathcal{M}}, Z_i^{\mathcal{M}})}\right],$$

as in Definition 7.3. The definitions for a minimising controller and accumulated rewards are similar.

The definition of reward structures (cf. Definition 7.14 on page 151) of PHAs can stay the same, as can the definition of their semantics in terms of reward structures of PAs. To define long-run average values in PHAs, we can combine Definition 7.21 on page 154 with Definition 5.6 on page 106. Both definitions only map the definition of values to the PA semantics.

For the abstraction, one firstly has to consider the reward-encoded PHA of Definition 7.23 on page 155, in which a part of the reward is encoded in the state space and certain rewards are obtained later than in the original model. The mutually optimal values stay the same in the reward-encoded PHA, because for each pair of player-con and player-env strategies $\sigma_{\text{con}}, \sigma_{\text{env}}$ the value functions using the joint scheduler $\text{join}(\sigma_{\text{con}}, \sigma_{\text{env}})$ stays the same, as we have already shown this for general schedulers in Lemma 7.24 on page 155. In the proof, we only considered maximal and minimal values, but the result holds indeed for all fair schedulers, as it relies on Lemma 7.6 on page 145 in the proof of which we considered values of all fair schedulers.

input : finite PA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$, reward structure $rew = (rew_{\text{num}}, rew_{\text{den}})$.
output: $(\text{vals}^{+,-}_{\mathcal{M},rew,\text{acc}}, \sigma_{\text{con}}, \sigma_{\text{env}})$ with $\sigma_{\text{con}} \in Strat^{\text{con,simple}}_{\mathcal{M}}$ and
$\sigma_{\text{env}} \in Strat^{\text{env,simple}}_{\mathcal{M}}$ and
$\text{vals}^{+,-}_{\mathcal{M},rew,\text{acc}} = \text{vals}^{\text{join}(\sigma_{\text{con}},\sigma_{\text{env}})}_{\mathcal{M},rew,\text{acc}}$
$= \inf_{\sigma'_{\text{env}} \in Strat^{\text{env}}_{\mathcal{M}}} \text{vals}^{\sigma_{\text{con}},\sigma'_{\text{env}}}_{\mathcal{M},rew,\text{acc}} = \sup_{\sigma'_{\text{con}} \in Strat^{\text{con}}_{\mathcal{M}}} \text{vals}^{\sigma'_{\text{con}},\sigma_{\text{env}}}_{\mathcal{M},rew,\text{acc}}$.

```
(σ_con, σ_env) := maxMinAccInit(M)
v := vals_{M_{join(σ_con,σ_env)},rew,acc}
repeat
    v' := v
    forall the s ∈ S do
        A(s) := argmax_{a∈Act, T(s,a)≠∅} rew_num(s,a,σ_env(s,a)) + Σ_{s'∈S} σ_env(s,a)(s')v(s')
        if σ_con(s) ∉ A(s) then choose σ_con(s) from A(s)
    (v, σ_env) := minAcc(M_{σ_con}, rew)        /* cf. Algorithm 7.1 */
until v = v'
return (v, σ_con, σ_env)
```

Algorithm 8.2: maxMinAcc($\mathcal{M}, rew$).

In Definition 5.8 on page 107 we have defined a game-based abstraction for reachability probabilities, and in Definition 7.26 on page 156 and Definition 7.29 on page 157, we have defined two different abstractions of rewards on abstractions of PHAs without synthesis, as in Definition 3.30 on page 52. The difference between Definition 5.8 and Definition 3.30 is basically that in Definition 5.8 we have a few additional transitions, which are used to model the fact that the controller cannot execute each command in each situation. For transitions already existing in the abstraction of Definition 3.30 we assign a reward as it was given in Definition 7.26 or Definition 7.29, and assign a reward which is bad for player con to the additional transitions. This way, the abstraction is guaranteed to disadvantage player con, which is necessary to obtain bounds on the values the controller can obtain. To show the correctness of this abstraction, one has to adapt Theorem 5.16 on page 113, which uses a chain of (in)equations obtained by showing a number of lemmas. The basic argumentation when using reward-based value functions stays the same for each of these lemmas.

To analyse properties of accumulated rewards in the abstractions, we can use an existing algorithm [FV96], which we sketch in Algorithm 8.2. It combines the ideas of Algorithm 5.1 on page 118 and methods of Algorithm 7.1 on page 160.

An algorithm to solve fractional long-run average games was described by Bloem et al. [Blo+09]. Its basic idea is to reduce such games to the case of long-run average rewards in which each step takes one unit of time.

input : finite PMC $\mathcal{M} = (S, \overline{s}, \mathcal{T}, V)$, reward structure $rew = (rew_{\text{num}}, rew_{\text{den}})$.
output: $\text{vals}_{\mathcal{M},rew,\text{acc}}$ where for all $v \in Evals(\mathcal{M})$ and all $s \in S$ we have
$\text{vals}_{\mathcal{M},rew,\text{acc}}(s)\langle v\rangle = \text{vals}_{\mathcal{M}_v,rew,\text{acc}}(s)$.

for $s \in S$ *with* $\mathcal{T}(s)(s) \neq 1$ **do**
  for $s' \in \text{pre}_{\mathcal{M}}(s)$ *with* $s' \neq s$ **do**
    $rew_{\text{num}}(s') := rew_{\text{num}}(s') + \frac{\mathcal{T}(s')(s)}{1-\mathcal{T}(s)(s)} rew_{\text{num}}(s)$
  for $(s_1, s_2) \in \text{pre}_{\mathcal{M}}(s) \times \text{post}_{\mathcal{M}}(s)$ *with* $s_1 \neq s \wedge s_2 \neq s$ **do**
    $\mathcal{T}(s_1)(s_2) := \mathcal{T}(s_1)(s_2) + \mathcal{T}(s_1)(s)\frac{1}{1-\mathcal{T}(s)(s)}\mathcal{T}(s)(s_2)$
  for $s' \in \text{pre}_{\mathcal{M}}(s)$ *with* $s' \neq s$ **do** $\mathcal{T}(s')(s) := 0$
  for $s' \in \text{post}_{\mathcal{M}}(s)$ *with* $s' \neq s$ **do** $\mathcal{T}(s)(s') := \frac{\mathcal{T}(s)(s')}{1-\mathcal{T}(s)(s)}$
  $rew_{\text{num}}(s) := \frac{1}{1-\mathcal{T}(s)(s)} rew_{\text{num}}(s)$
  $\mathcal{T}(s)(s) := 0$
for $s \in S$ **do** $v(s) := rew_{\text{num}}(s)$
return v

Algorithm 8.3: accParam($\mathcal{M}$, *Reach*).

8.6 Parametric Models and Rewards

If we consider reward-based properties of parametric PHA, the adaption of notations is rather straightforward. As the semantical model, we can use the PAs of Definition 6.4 on page 124. We can take over the definition of the reward structures on transitions of these models from Definition 7.1 on page 142. The definitions for the high-level PHA model is also simple. We use Definition 6.8 on page 126 and can take over the definition of rewards from Definition 7.14 on page 151.

The definition of reward-encoded PAs, etc. is as in the nonparametric case. We can use Definition 6.13 on page 127 to abstract the parametric PHAs, and use Definition 7.26 on page 156 or Definition 7.29 on page 157 as abstractions for the rewards. We have to make sure that rewards in abstractions are always higher (or lower, respectively) than certain rewards in the semantics. In the nonparametric reward-based analysis, we used linear programming for this issue. In case the reward structures stay nonparametric, the abstraction can be done as before. In the case of reward structures which depend on the model parameters, we have to adapt the method accordingly to ensure that for all valid parameter evaluations the requirements on the reward structures of the induced PHA are fulfilled.

To compute the values in the abstract model, we have to adapt the algorithm which computes values in Markov chains in Algorithm 6.1 on page 129, as well as the algorithm to verify optimality of a given value function in Algorithm 6.2 on page 133. We consider the case of expected accumulated rewards here. The main algorithm Algorithm 6.3 on page 134 responsible for dividing regions can remain basically as it is. As an adaption of Algorithm 6.1, we use a method already described in a previous publication [HHZ09; HHZ11a]. The values in induced PMCs are solved by a variant of Algorithm 6.1, where

input : finite PPA $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T}, V)$, reward structure $rew = (rew_{\text{num}}, rew_{\text{den}})$, region r.
output: either ? or $(\text{vals}^{-}_{\mathcal{M},rew,\text{acc}}, \sigma)$ with $\sigma \in Sched^{\text{simple}}_{\mathcal{M}}$ and for all $v \in Evals(\mathcal{M}) \cap r$ and $s \in S$ we have $\text{vals}^{-}_{\mathcal{M},rew,\text{acc}}(s)\langle v \rangle = \text{vals}^{-}_{\mathcal{M}_v,rew,\text{acc}}(s) = \text{vals}^{\sigma}_{\mathcal{M}_v,Reach,\text{acc}}(s)$.

$c := \text{centre}(r)$
if $c \notin Evals(\mathcal{M})$ **then return** ?
$(\_, \sigma) := \text{MINACC}(\mathcal{M}_c, Reach)$ `/* cf. Algorithm 7.1 */`
$v := \text{ACCPARAM}(\mathcal{M}_\sigma, Reach)$ `/* cf. Algorithm 8.3 */`
$valid := true$
forall the $s \in S, a \in Act, \mu \in \mathcal{T}(s,a)$ **do**
  $v_{\text{cmp}}(s) := rew_{\text{num}}(s,a,\mu) + \sum_{s' \in S} \mu(s')v(s')$
  $valid := valid \wedge \text{CHECK}(v(s) \leq v_{\text{cmp}}(s), r)$ `/* cf. Definition 6.23 */`
if *valid* **then return** (v, σ)
else return ?

Algorithm 8.4: MINACCPARAMREGION$(\mathcal{M}, Reach, r)$.

processing one state maintains the accumulated reward value instead of the reachability probability. To prove the optimality we apply an algorithm of a previous publication [HHZ11a]. The algorithm there works in a similar way as in Algorithm 6.3 by proving that a certain scheduler is optimal for an entire region.

Bloem et al. [Blo+09] describe an algorithm to solve fractional long-run average games in the nonparametric case. Its basic idea is to reduce such games to the case of long-run average rewards in which each step takes one unit of time. Because this algorithm is an iteration based on policy iteration, it can also be adapted to the parametric case.

As an example for this combination we consider the parametric water level control of Subsection 6.5.2 on page 136 and consider the minimal expected number of commands executed until termination as in Subsection 7.5.2 on page 162. It turns out that there is a single optimal scheduler for all variable evaluations and the expected reward is at least as large as

$$\frac{2x-4}{xy-x-y}.$$

In Figure 8.2 we give a plot for this function on the left. On the right, we check whether the minimal expected number of commands executed until termination is at least as large as 15. As in Subsection 6.5.2, the white regions are the ones guaranteed to fulfil the bound. The expected number of commands executed decreases with increasing failure probability, because a higher failure probability increases the chance of an earlier termination of a system, which means that fewer commands can be executed.

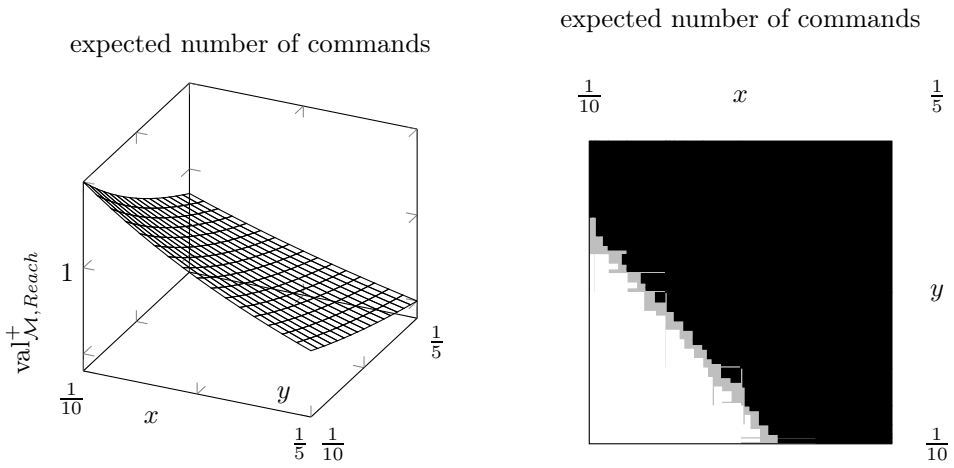


Figure 8.2: Results for reward-based properties in parametric water level control.

8.7 Conclusion

We have shown how we can combine the results of the previous chapters to handle more general models and properties. To show how such a combination can take place, we consider all pairs of different extensions, and discuss the necessary steps to combine them. Combinations of more than two of them are then straightforward. We also exemplified two of the orthogonal combinations by sketching corresponding extensions of our case studies.

9 Summary and Future Work

The analysis of hybrid systems is intricate, as reflected by the undecidability results for most models, except restricted subclasses of such systems. Integrating stochastic behaviours further complicates the situation; in addition to the discrete-continuous dynamics on the infinite state space, abstractions have to take probabilistic behaviours into account. Although previous works have provided significant results for many important classes of such models, including exact solution methods, prior to our work it was not clear how to develop a generic analysis framework for the automatic analysis of the generic class of stochastic hybrid systems we operate on.

9.1 Summary of Contributions

In this thesis, we have developed an efficient analysis framework for a generic class of stochastic hybrid systems, based on solution methods for classical nonprobabilistic hybrid automata. After having developed the basic method, we have expanded it into a number of different directions, thus increasing the expressivity of the models and properties we can analyse.

Probabilistic Reachability

We have developed the foundation of our framework in terms of a flexible method to analyse two kinds of reachability properties of probabilistic hybrid automata. To do so, we combined methods for the abstraction of classical hybrid automata with results from the analysis of Markov models, both to show the correctness results—for which we used probabilistic simulation—as well as to compute value bounds in the abstract models—by using policy iteration and related methods. Doing so, we were able to conservatively overapproximate reachability probabilities in abstract versions of models under considerations, and thus develop the first automatic and generic framework for this generic class of stochastic hybrid systems.

Continuous Distributions

To extend the set of models we can analyse, we extended the class of probability distributions which can appear in a stochastic hybrid system to continuous distributions. We solved not only theoretical issues concerning the well-definedness of such models, but also extended our existing analysis framework in order to handle this more general model class.

Partial Control

For cases in which the nondeterminism of a system is under our partial control, we have developed an extension of our method to synthesise controllers which resolve this nondeterminism. When used to steer an actual system, they are then guaranteed to maintain the validity of reachability properties by the bounds computed in the abstraction. The proof of correctness turned out to be more complicated than for the case without control, because we could no longer use probabilistic simulation as a means of relating two system models.

Parametric Probability Distributions

We have considered systems in which probabilities are not fixed, but given by a set of parameters. Combining our results on stochastic hybrid systems with our previous works in parametric probabilistic model checking, we were able to provide an extension of our framework which can handle this kinds of models. Instead of ending up with a single number describing a bound on the reachability probability, we are now able to compute parameter instantiations which fulfil a given property, and to obtain functions describing reachability probabilities. These functions are subject to further analysis, for instance to find optimal parameter values for a given property using for instance nonlinear optimisation tools or computer algebra systems.

Rewards

Extending the previous analysis of reachability probabilities, we have assigned rewards to timed and discrete transitions of our hybrid automata. In turn, we were able to reason about expected accumulated reward values, or expected long-run reward values. Doing so allowed us to reason about properties which are related to resource consumption or are more on the economical side, like for instance the energy consumption or cost of a system until termination, or the long-run cost of running a given system.

9.2 Conclusions

By the orthogonal combination of concepts from the area of classical hybrid automata with those from the area of Markov models, we arrive at an analysis framework that can handle a significant class of stochastic hybrid systems, involving nondeterminism, general probability distributions, continuous time and complex properties. Our framework is implemented in a way which allows for a completely automatic analysis without the need of manual intervention. This admits to automatically verify stochastic hybrid systems far beyond the reach of any previous method.

9.3 Future Works

The work we have presented in this thesis gives path to several ideas of future works.

State-dependent probabilities. In this thesis, all probabilities of the models considered were indeed fixed or parametric in a set of model variables, but they could not depend on the current system state. For instance, we could not analyse a model which involves a probabilistic guarded command $g \to x : u_1 + (1-x) : u_2$ where x is not a model parameter in the sense of Chapter 6 but a continuous variable of the model. However, we have developed a solution method for systems in the area of probabilistic discrete-time, discrete-state systems [Fio+12]. Here, similarly to the methods of Chapter 7, we have to find certain minima and maxima over the abstract states, like x in the example above. Having done so, we use an extension of *abstract (or interval) Markov chains* [Kat+12] as the abstract model. This method can be extended to probabilistic hybrid automata. In particular, if we are using polyhedra-based representations of abstract states, we can again use linear programming to reason about these extrema, which is indeed simpler than the optimisation problem we had to solve in the previous publication.

Richer properties. We already considered an extension from reachability to reward-based properties. In the future, it might be of interest to consider other classes, such as for instance specifications in logics such as PCTL [HJ94] or properties based on words, e.g. given as Büchi automata [Büc62]. A similar method has proven successful in the area of discrete-time stochastic hybrid systems [AKM11]. Depending on the property under consideration, as in for the extensions discussed above, we might have to take a closer look at the abstract states.

Improving time-bounded reachability. In the case studies we considered so far, we have handled time-bounded reachability by using an additional variable, which increases the complexity of the model and thus the cost of the analysis. If we instead use a timer which is reset in each mode, as in Chapter 7, we can find out about the minimal and maximal times for staying in each abstract state. Using an approach related to a method

by Andova et al. [AHK03], we could use an abstraction of the latter model to handle time-bounded properties, by building a new model consisting of several copies of the abstract state space in a chain. This has the advantage that the computational cost within the hybrid solver is independent of the time bound, though results obtained this way might become less precise for larger time bounds.

Stochastic differential equations. In the hybrid automata we were considering, we have used differential (in)equations, but have thus far not included stochastic differential equations. Althoff et al. [ASB08] use zonotopes to compute overapproximations of the states in which the system will remain with a guaranteed probability. The systems they consider do so far not contain nondeterminism, but, differently from our method, they do contain stochastic differential equations. It would be interesting to combine with their methods thus to handle systems which contain both features.

State-space refinement. In our approach so far, we relied on solvers computing the abstraction of the stochastic hybrid automaton. In many cases, this method turned out to work very well, but in other cases, it did not. For instance, we also tried to use the tool HSolver [RS07] instead of ProHVer to compute an overapproximation. This failed, because HSolver always tries to prove the complete safety of the system. As in our models there are indeed some paths leading to unsafe states, HSolver never stopped trying to disprove these paths in favour of trying to disprove paths which indeed are only present in the abstract model. Because HSolver has indeed been successful in showing the safety of completely safe systems, this indicates that it might be worthwhile to adapt existing methods to our needs, rather than relying on an existing solver to compute the abstraction. Doing so decreases the generality of the method, as new abstraction methods have to be adapted to our probabilistic setting. It has however the potential to provide more precise results and reduce the cost of analyses, as we might be able to work with a fewer number of abstract states.

Refined continuous distributions. In Chapter 4, we handle continuous probability distributions by dividing the possible successor states into several sets, already on the high-level description of the automaton. The approach does not take into account automatically that certain parts of the possible successor state set might be more important than others, and furthermore relies on a good guess for the splitting. Along with the state-space refinement of the previous paragraph, one could implement a method to split the distributions in a more target-oriented way. It might also be worthwhile to consider whether an extension of interval Markov chains we discussed previously could be used as an alternative to the splitting of distributions. Intervals would then overapproximate the possible probabilities over all concrete states of a given abstract state to move to a certain successor abstract state.

Bibliography

[Aba+08] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. "Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems". In: *Automatica* 44.11 (2008), pp. 2724–2734.

[Aba+10] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini. "Approximate model checking of stochastic hybrid systems". In: *European Journal of Control* (2010).

[ACD93] R. Alur, C. Courcoubetis, and D. L. Dill. "Model-checking in dense real-time". In: *Inf. Comput.* 104.1 (1993), pp. 2–34.

[AD11] J. Assouramou and J. Desharnais. "Analysis of non-linear probabilistic hybrid systems". In: *QAPL*. Ed. by M. Massink and G. Norman. Vol. 57. Electronic Proceedings in Theoretical Computer Science. 2011, pp. 104–119.

[AD90] R. Alur and D. L. Dill. "Automata for modeling real-time systems". In: *ICALP*. Ed. by M. Paterson. Vol. 443. Lecture Notes in Computer Science. Springer, 1990, pp. 322–335. ISBN: 3-540-52826-1.

[AD94] R. Alur and D. L. Dill. "A theory of timed automata". In: *Theor. Comput. Sci.* 126.2 (1994), pp. 183–235.

[ADD00] R. B. Ash and C. Doléans-Dade. *Probability & Measure Theory*. Academic Press, 2000.

[ADI06] R. Alur, T. Dang, and F. Ivančić. "Predicate abstraction for reachability analysis of hybrid systems". In: *ACM Trans. Embedded Comput. Syst.* 5.1 (2006), pp. 152–199.

[AG97] E. Altman and V. Gaitsgory. "Asymptotic optimization of a nonlinear hybrid system governed by a Markov decision process". In: *SIAM Journal of Control and Optimization* 35.6 (1997), pp. 2070–2085.

[AHH93] R. Alur, T. A. Henzinger, and P.-H. Ho. "Automatic symbolic verification of embedded systems". In: *IEEE Real-Time Systems Symposium*. IEEE Computer Society, 1993, pp. 2–11.

[AHH96] R. Alur, T. A. Henzinger, and P.-H. Ho. "Automatic symbolic verification of embedded systems". In: *IEEE Trans. Software Eng.* 22.3 (1996), pp. 181–201.

[AHK03] S. Andova, H. Hermanns, and J.-P. Katoen. "Discrete-time rewards model-checked". In: *FORMATS*. Ed. by K. G. Larsen and P. Niebert. Vol. 2791. Lecture Notes in Computer Science. Springer, 2003, pp. 88–104. ISBN: 3-540-21671-5.

[AKM11] A. Abate, J.-P. Katoen, and A. Mereacre. "Quantitative automata model checking of autonomous stochastic hybrid systems". In: *HSCC*. Ed. by M. Caccamo, E. Frazzoli, and R. Grosu. ACM, 2011, pp. 83–92. ISBN: 978-1-4503-0629-4.

[Alf97] L. de Alfaro. "Formal Verification of Probabilistic Systems". PhD thesis. Stanford University, 1997.

[Alu+95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. "The algorithmic analysis of hybrid systems". In: *Theor. Comput. Sci.* 138.1 (1995), pp. 3–34.

[AMP94] E. Asarin, O. Maler, and A. Pnueli. "Symbolic controller synthesis for discrete and timed systems". In: *Hybrid Systems*. Ed. by P. J. Antsaklis, W. Kohn, A. Nerode, and S. Sastry. Vol. 999. Lecture Notes in Computer Science. Springer, 1994, pp. 1–20. ISBN: 3-540-60472-3.

[And+09] É. André, T. Chatain, L. Fribourg, and E. Encrenaz. "An inverse method for parametric timed automata". In: *Int. J. Found. Comput. Sci.* 20.5 (2009), pp. 819–836.

[Arn74] L. Arnold. *Stochastic Differential Equations: Theory and Applications*. Wiley - Interscience, 1974.

[Asa+00] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. "Effective synthesis of switching controllers for linear systems". In: *Proceedings of the IEEE, Special Issue on "Hybrid Systems"* 88.7 (2000), pp. 1011–1025.

[ASB08] M. Althoff, O. Stursberg, and M. Buss. "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization". In: *CDC*. IEEE, 2008, pp. 4042–4048.

[Bak+92] J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg, eds. *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*. Vol. 600. Lecture Notes in Computer Science. Springer, 1992. ISBN: 3-540-55564-1.

[BBC08] J. R. Banga and E. Balsa-Canto. "Parameter estimation and optimal experimental design". In: *Essays in biochemistry* 45 (2008), pp. 195–209. ISSN: 0071-1365.

[Beh+07] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. "UPPAAL-Tiga: time for playing games!" In: *CAV*. Ed. by W. Damm and H. Hermanns. Vol. 4590. Lecture Notes in Computer Science. Springer, 2007, pp. 121–125. ISBN: 978-3-540-73367-6.

[Bel57] R. Bellman. "A Markovian decision process". In: *Indiana Univ. Math. J.* 6 (1957), pp. 679–684.

[BK08] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008, pp. I–XVII, 1–975. ISBN: 978-0-262-02649-9.

[BK85] J. A. Bergstra and J. W. Klop. "Algebra of communicating processes with abstraction". In: *Theor. Comput. Sci.* 37 (1985), pp. 77–121.

[BK98] C. Baier and M. Z. Kwiatkowska. "Model checking for a probabilistic branching time logic with fairness". In: *Distributed Computing* 11.3 (1998), pp. 125–155.

[BL06a] H. A. P. Blom and J. Lygeros. *Stochastic Hybrid Systems: Theory and Safety Critical Applications*. Vol. 337. Lecture Notes in Control and Information Sciences. Springer, 2006.

[BL06b] M. L. Bujorianu and J. Lygeros. "Toward a general theory of stochastic hybrid systems". In: *Stochastic Hybrid Systems: Theory and Safety Critical Applications*. Ed. by H. Blom and J. Lygeros. Vol. 337. Lecture Notes in Control and Information Sciences. Springer, 2006, pp. 3–30.

[BLB05] M. L. Bujorianu, J. Lygeros, and M. C. Bujorianu. "Bisimulation for general stochastic hybrid systems". In: *HSCC*. Ed. by M. Morari and L. Thiele. Vol. 3414. Lecture Notes in Computer Science. Springer, 2005, pp. 198–214. ISBN: 3-540-25108-1.

[BLL08] M. L. Bujorianu, J. Lygeros, and R. Langerak. "Reachability analysis of stochastic hybrid systems by optimal control". In: *HSCC*. Ed. by M. Egerstedt and B. Mishra. Vol. 4981. Lecture Notes in Computer Science. Springer, 2008, pp. 610–613. ISBN: 978-3-540-78928-4.

[Blo+09] R. Bloem, K. Greimel, T. A. Henzinger, and B. Jobstmann. "Synthesizing robust systems". In: *FMCAD*. IEEE, 2009, pp. 85–92. ISBN: 978-1-4244-4966-8.

[BM07] P. Bouyer and N. Markey. "Costs are expensive!" In: *FORMATS*. Ed. by J.-F. Raskin and P. S. Thiagarajan. Vol. 4763. Lecture Notes in Computer Science. Springer, 2007, pp. 53–68. ISBN: 978-3-540-75453-4.

[BO04] A. Berarducci and M. Otero. "An additive measure in o-minimal expansions of fields". In: *The Quarterly Journal of Mathematics* 55.4 (2004), pp. 411–419.

[Bou+07] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. "The cost of punctuality". In: *LICS*. IEEE Computer Society, 2007, pp. 109–120.

[BP98] Y. Baisalov and B. Poizat. "Paires de structures o-minimales". In: *J. Symb. Log.* 63.2 (1998), pp. 570–578.

[BT09] L. Bozzelli and S. L. Torre. "Decision problems for lower/upper bound parametric timed automata". In: *Formal Methods in System Design* 35.2 (2009), pp. 121–151.

[Buj04] M. L. Bujorianu. "Extended stochastic hybrid systems and their reachability problem". In: *HSCC*. Ed. by R. Alur and G. J. Pappas. Vol. 2993. Lecture Notes in Computer Science. Springer, 2004, pp. 234–249. ISBN: 3-540-21259-0.

[Büc62] J. R. Büchi. "On a decision method in restricted second order arithmetic". In: *Proceedings of the 1960 International Congress on Logic, Methodology and Philosophy of Science (LMPS'60)*. Ed. by E. Nagel, P. Suppes, and A. Tarski. Stanford University Press, 1962, pp. 1–11.

[CFG11] M. Caccamo, E. Frazzoli, and R. Grosu, eds. *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*. ACM, 2011. ISBN: 978-1-4503-0629-4.

[Cla+00] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. "Counterexample-guided abstraction refinement". In: *CAV*. Ed. by E. A. Emerson and A. P. Sistla. Vol. 1855. Lecture Notes in Computer Science. Springer, 2000, pp. 154–169. ISBN: 3-540-67770-4.

[Cla+03] E. M. Clarke, A. Fehnker, Z. Han, B. H. Krogh, O. Stursberg, and M. Theobald. "Verification of hybrid systems based on counterexample-guided abstraction refinement". In: *TACAS*. Ed. by H. Garavel and J. Hatcliff. Vol. 2619. Lecture Notes in Computer Science. Springer, 2003, pp. 192–207. ISBN: 3-540-00898-5.

[Con93] A. Condon. "On algorithms for simple stochastic games". In: *Advances in Computational Complexity Theory, volume 13 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1993, pp. 51–73.

[D'A+09] P. R. D'Argenio, N. Wolovick, P. S. Terraf, and P. Celayes. "Nondeterministic labeled Markov processes: bisimulations and logical characterization". In: *QEST*. IEEE Computer Society, 2009, pp. 11–20. ISBN: 978-0-7695-3808-2.

[Dav84] M. H. A. Davis. "Piecewise-deterministic Markov processes: a general class of non-diffusion stochastic models". In: *Journal of the Royal Statistical Society*. B (Methodological) 46.3 (1984), pp. 353–388.

[Daw04] C. Daws. "Symbolic and parametric model checking of discrete-time Markov chains". In: *ICTAC*. Ed. by Z. Liu and K. Araki. Vol. 3407. Lecture Notes in Computer Science. Springer, 2004, pp. 280–294. ISBN: 3-540-25304-1.

[Dbl] *Eighth International Conference on Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5-8 September, 2011*. IEEE Computer Society, 2011. ISBN: 978-1-4577-0973-9.

[Dil89] D. L. Dill. "Timing assumptions and verification of finite-state concurrent systems". In: *Automatic Verification Methods for Finite State Systems*. Ed. by J. Sifakis. Vol. 407. Lecture Notes in Computer Science. Springer, 1989, pp. 197–212. ISBN: 3-540-52148-8.

[DISS11] W. Damm, C. Ihlemann, and V. Sofronie-Stokkermans. "PTIME parametric verification of safety properties for reasonable linear hybrid automata". In: *Mathematics in Computer Science* 5.4 (2011), pp. 469–497.

[DM12] T. Dang and I. M. Mitchell, eds. *Hybrid Systems: Computation and Control (part of CPS Week 2012), HSCC'12, Beijing, China, April 17-19, 2012*. ACM, 2012. ISBN: 978-1-4503-1220-2.

[DTW12] P. R. D'Argenio, P. S. Terraf, and N. Wolovick. "Bisimulations for nondeterministic labeled Markov processes". In: *Math. Struct. in Comp. Science* 22.1 (2012), pp. 43–68.

[EJ11] C. von Essen and B. Jobstmann. "Synthesizing systems with optimal average-case behavior for ratio objectives". In: *iWIGP*. Ed. by J. Reich and B. Finkbeiner. Vol. 50. Electronic Proceedings in Theoretical Computer Science. 2011, pp. 17–32.

[EJ12] C. von Essen and B. Jobstmann. "Synthesizing efficient controllers". In: *VMCAI*. Ed. by V. Kuncak and A. Rybalchenko. Vol. 7148. Lecture Notes in Computer Science. Springer, 2012, pp. 428–444. ISBN: 978-3-642-27939-3.

[EM08] M. Egerstedt and B. Mishra, eds. *Hybrid Systems: Computation and Control, 11th International Workshop, HSCC 2008, St. Louis, MO, USA, April 22-24, 2008. Proceedings*. Vol. 4981. Lecture Notes in Computer Science. Springer, 2008. ISBN: 978-3-540-78928-4.

[FA09] L. Fribourg and É. André. "An inverse method for policy iteration based algorithms". In: *INFINITY*. Electronic Proceedings in Theoretical Computer Science. Open Publishing Association, 2009, pp. 44–61.

[FHT08] M. Fränzle, H. Hermanns, and T. Teige. "Stochastic satisfiability modulo theory: a novel technique for the analysis of probabilistic hybrid systems". In: *HSCC*. Ed. by M. Egerstedt and B. Mishra. Vol. 4981. Lecture Notes in Computer Science. Springer, 2008, pp. 172–186. ISBN: 978-3-540-78928-4.

[Fio+12] L. M. F. Fioriti, E. M. Hahn, H. Hermanns, and B. Wachter. "Variable probabilistic abstraction refinement". In: *ATVA*. 2012, pp. 300–316.

[FK11] L. Fribourg and U. Kühne. *Parametric Verification of Hybrid Automata Using the Inverse Method*. Research Report LSV-11-04. 25 pages. Laboratoire Spécification et Vérification, ENS Cachan, France, Mar. 2011.

[Fre+11] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. "SpaceEx: scalable verification of hybrid systems". In: *CAV*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 379–395. ISBN: 978-3-642-22109-5.

[Fre05] G. Frehse. "PHAVer: algorithmic verification of hybrid systems past HyTech". In: *HSCC*. Ed. by M. Morari and L. Thiele. Vol. 3414. Lecture Notes in Computer Science. Springer, 2005, pp. 258–273. ISBN: 3-540-25108-1.

[Fre08] G. Frehse. "PHAVer: algorithmic verification of hybrid systems past HyTech". In: *STTT* 10.3 (2008), pp. 263–279.

[Frä+07] M. Fränzle, C. Herde, T. Teige, S. Ratschan, and T. Schubert. "Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure". In: *JSAT* 1.3-4 (2007), pp. 209–236.

[Frä+11] M. Fränzle, E. M. Hahn, H. Hermanns, N. Wolovick, and L. Zhang. "Measurability and safety verification for stochastic hybrid systems". In: *HSCC*. Ed. by M. Caccamo, E. Frazzoli, and R. Grosu. ACM, 2011, pp. 43–52. ISBN: 978-1-4503-0629-4.

[FTE10a] M. Fränzle, T. Teige, and A. Eggers. "Engineering constraint solvers for automatic analysis of probabilistic hybrid automata". In: *J. Log. Algebr. Program.* 79.7 (2010), pp. 436–466.

[FTE10b] M. Fränzle, T. Teige, and A. Eggers. "Satisfaction meets expectations - computing expected values of probabilistic hybrid systems with smt". In: *IFM*. 2010, pp. 168–182.

[FV96] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. New York, NY, USA: Springer-Verlag New York, Inc., 1996. ISBN: 0-387-94805-8.

[Gan+04] H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli. "DPLL(T): fast decision procedures". In: *CAV*. Ed. by R. Alur and D. Peled. Vol. 3114. Lecture Notes in Computer Science. Springer, 2004, pp. 175–188. ISBN: 3-540-22342-8.

[GG09] C. L. Guernic and A. Girard. "Reachability analysis of hybrid systems using support functions". In: *CAV*. Ed. by A. Bouajjani and O. Maler. Vol. 5643. Lecture Notes in Computer Science. Springer, 2009, pp. 540–554. ISBN: 978-3-642-02657-7.

[Gir82] M. Giry. "A categorical approach to probability theory". In: *Categorical Aspects of Topology and Analysis*. Springer, 1982, pp. 68–85.

[GKR04] S. Gupta, B. H. Krogh, and R. A. Rutenbar. "Towards formal verification of analog designs". In: *ICCAD*. IEEE Computer Society / ACM, 2004, pp. 210–217. ISBN: 0-7803-8702-3.

[GQ11] G. Gopalakrishnan and S. Qadeer, eds. *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011. ISBN: 978-3-642-22109-5.

[GT04] R. Ghosh and C. Tomlin. "Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling". In: *Systems Biology* 1 (2004), pp. 170–183.

[Hah+10] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang. "PARAM: a model checker for parametric Markov models". In: *CAV*. Ed. by T. Touili, B. Cook, and P. Jackson. Vol. 6174. Lecture Notes in Computer Science. Springer, 2010, pp. 660–664. ISBN: 978-3-642-14294-9.

[Hah+11] E. M. Hahn, G. Norman, D. Parker, B. Wachter, and L. Zhang. "Game-based abstraction and controller synthesis for probabilistic hybrid systems". In: *QEST*. IEEE Computer Society, 2011, pp. 69–78. ISBN: 978-1-4577-0973-9.

[Hal98] N. Halbwachs. "Synchronous programming of reactive systems". In: *CAV*. Ed. by A. J. Hu and M. Y. Vardi. Vol. 1427. Lecture Notes in Computer Science. Springer, 1998, pp. 1–16. ISBN: 3-540-64608-6.

[Han09] T. Han. "Diagnosis, Synthesis and Analysis of Probabilistic Models". PhD thesis. RWTH Aachen University/University of Twente, 2009.

[Har87] D. Harel. "Statecharts: a visual formalism for complex systems". In: *Sci. Comput. Program.* 8.3 (1987), pp. 231–274.

[Hen+92] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. "Symbolic model checking for real-time systems". In: *LICS*. IEEE Computer Society, 1992, pp. 394–406. ISBN: 0-8186-2735-2.

[Hen+94] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. "Symbolic model checking for real-time systems". In: *Inf. Comput.* 111.2 (1994), pp. 193–244.

[Hen+98] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. "What's decidable about hybrid automata?" In: *J. Comput. Syst. Sci.* 57.1 (1998), pp. 94–124.

[Her+08] C. Herde, A. Eggers, M. Fränzle, and T. Teige. "Analysis of hybrid systems using HySAT". In: *ICONS*. IEEE Computer Society, 2008, pp. 196–201.

[HHWT97a] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. "HYTECH: a model checker for hybrid systems". In: *STTT* 1.1-2 (1997), pp. 110–122.

[HHWT97b] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. "HYTECH: a model checker for hybrid systems". In: *CAV*. Ed. by O. Grumberg. Vol. 1254. Lecture Notes in Computer Science. Springer, 1997, pp. 460–463. ISBN: 3-540-63166-6.

[HHZ09] E. M. Hahn, H. Hermanns, and L. Zhang. "Probabilistic reachability for parametric Markov models". In: *SPIN*. Ed. by C. S. Pasareanu. Vol. 5578. Lecture Notes in Computer Science. Springer, 2009, pp. 88–106. ISBN: 978-3-642-02651-5.

[HHZ11a] E. M. Hahn, T. Han, and L. Zhang. "Synthesis for PCTL in parametric Markov decision processes". In: *NASA Formal Methods*. Ed. by M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi. Vol. 6617. Lecture Notes in Computer Science. Springer, 2011, pp. 146–161. ISBN: 978-3-642-20397-8.

[HHZ11b] E. M. Hahn, H. Hermanns, and L. Zhang. "Probabilistic reachability for parametric Markov models". In: *STTT* 13.1 (2011), pp. 3–19.

[HJ94] H. Hansson and B. Jonsson. "A logic for reasoning about time and reliability". In: *Formal Asp. Comput.* 6.5 (1994), pp. 512–535.

[HK66] A. J. Hoffman and R. M. Karp. "On nonterminating stochastic games". In: *Management Science* 12.5 (1966), pp. 359–370.

[HK99] T. A. Henzinger and P. W. Kopke. "Discrete-time control for rectangular hybrid automata". In: *Theor. Comput. Sci.* 221.1-2 (1999), pp. 369–392.

[HLS00] J. Hu, J. Lygeros, and S. Sastry. "Towards a theory of stochastic hybrid systems". In: *HSCC*. Ed. by N. A. Lynch and B. H. Krogh. Vol. 1790. Lecture Notes in Computer Science. Springer, 2000, pp. 160–173. ISBN: 3-540-67259-1.

[How60] R. A. Howard. *Dynamic Programming and Markov Processes*. Cambridge, Massachusetts: MIT Press, 1960.

[JL91] B. Jonsson and K. G. Larsen. "Specification and refinement of probabilistic processes". In: *LICS*. IEEE Computer Society, 1991, pp. 266–277. ISBN: 0-8186-2230-X.

[JLR09] M. Jurdziński, R. Lazić, and M. Rutkowski. "Average-price-per-reward games on hybrid automata with strong resets". In: *VMCAI*. Ed. by N. D. Jones and M. Müller-Olm. Vol. 5403. Lecture Notes in Computer Science. Springer, 2009, pp. 167–181. ISBN: 978-3-540-93899-6.

[JP09] A. A. Julius and G. J. Pappas. "Approximations of stochastic hybrid systems". In: *IEEE Trans. Automat. Contr.* 54.6 (2009), pp. 1193–1203.

[Kat+12] J.-P. Katoen, D. Klink, M. Leucker, and V. Wolf. "Three-valued abstraction for probabilistic systems". In: *J. Log. Algebr. Program.* 81.4 (2012), pp. 356–389.

[Kec95] A. S. Kechris. *Classical Descriptive Set Theory*. Vol. 156. Graduate Texts in Mathematics. Springer, 1995.

[Kel76] R. M. Keller. "Formal verification of parallel programs". In: *Commun. ACM* 19.7 (1976), pp. 371–384.

[KNP06] M. Z. Kwiatkowska, G. Norman, and D. Parker. "Game-based abstraction for Markov decision processes". In: *QEST*. IEEE Computer Society, 2006, pp. 157–166. ISBN: 0-7695-2665-9.

[KNP09] M. Z. Kwiatkowska, G. Norman, and D. Parker. "Stochastic games for verification of probabilistic timed automata". In: *FORMATS*. Ed. by J. Ouaknine and F. W. Vaandrager. Vol. 5813. Lecture Notes in Computer Science. Springer, 2009, pp. 212–227. ISBN: 978-3-642-04367-3.

[KNP11] M. Z. Kwiatkowska, G. Norman, and D. Parker. "PRISM 4.0: verification of probabilistic real-time systems". In: *CAV*. Ed. by G. Gopalakrishnan and S. Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 585–591. ISBN: 978-3-642-22109-5.

[Kro99] T. Kropf. *Introduction to Formal Hardware Verification: Methods and Tools for Designing Correct Circuits and Systems*. 1st. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999.

[KSB01] S. Kowalewski, O. Stursberg, and N. Bauer. "An experimental batch plant as a case example for the verification of hybrid systems". In: *European Journal of Control* 7 (2001), pp. 366–381.

[KSK66] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.

[Kwi+02] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. "Automatic verification of real-time systems with discrete probability distributions". In: *Theor. Comput. Sci.* 282.1 (2002), pp. 101–150.

[Lew90] H. R. Lewis. "A logic of concrete time intervals (extended abstract)". In: *LICS*. IEEE Computer Society, 1990, pp. 380–389. ISBN: 0-8186-2073-0.

[LGS95] J. Lygeros, D. N. Godbole, and S. Sastry. "A game-theoretic approach to hybrid system design". In: *Hybrid Systems*. Ed. by R. Alur, T. A. Henzinger, and E. D. Sontag. Vol. 1066. Lecture Notes in Computer Science. Springer, 1995, pp. 1–12. ISBN: 3-540-61155-X.

[LPS98] G. Lafferriere, G. J. Pappas, and S. Sastry. "O-minimal hybrid systems". In: *Mathematics of Control, Signals, and Systems* 13 (1998), pp. 1–21.

[LPY99] G. Lafferriere, G. J. Pappas, and S. Yovine. "A new class of decidable hybrid systems". In: *HSCC*. Ed. by F. W. Vaandrager and J. H. van Schuppen. Vol. 1569. Lecture Notes in Computer Science. Springer, 1999, pp. 137–151. ISBN: 3-540-65734-7.

[Mil71] R. Milner. "An algebraic definition of simulation between programs". In: *IJCAI*. Ed. by D. C. Cooper. William Kaufmann, 1971, pp. 481–489. ISBN: 0-934613-34-6.

[Mil89] R. Milner. *Communication and Concurrency*. PHI Series in computer science. Prentice Hall, 1989. ISBN: 978-0-13-115007-2.

[MMP91] O. Maler, Z. Manna, and A. Pnueli. "From timed to hybrid systems". In: *REX Workshop*. Ed. by J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg. Vol. 600. Lecture Notes in Computer Science. Springer, 1991, pp. 447–484. ISBN: 3-540-55564-1.

[MT05] M. Morari and L. Thiele, eds. *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005, Zurich, Switzerland, March 9-11, 2005, Proceedings*. Vol. 3414. Lecture Notes in Computer Science. Springer, 2005. ISBN: 3-540-25108-1.

[MW12] L. Mikeev and V. Wolf. "Parameter estimation for stochastic hybrid models of biochemical reaction networks". In: *HSCC*. Ed. by T. Dang and I. M. Mitchell. ACM, 2012, pp. 155–166. ISBN: 978-1-4503-1220-2.

[Nai03] S. Naimpally. "What is a hit-and-miss topology?" In: *TopCom* 8.1 (2003).

[Nic+92] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. "An approach to the description and analysis of hybrid systems". In: *Hybrid Systems*. Ed. by R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel. Vol. 736. Lecture Notes in Computer Science. Springer, 1992, pp. 149–178. ISBN: 3-540-57318-6.

[NSY91] X. Nicollin, J. Sifakis, and S. Yovine. "From ATP to timed graphs and hybrid systems". In: *REX Workshop*. Ed. by J. W. de Bakker, C. Huizing, W. P. de Roever, and G. Rozenberg. Vol. 600. Lecture Notes in Computer Science. Springer, 1991, pp. 549–572. ISBN: 3-540-55564-1.

[NSY93] X. Nicollin, J. Sifakis, and S. Yovine. "From ATP to timed graphs and hybrid systems". In: *Acta Inf.* 30.2 (1993), pp. 181–202.

[Pet81] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1981. ISBN: 0136619835.

[PH06] M. Prandini and J. Hu. "A stochastic approximation method for reachability computations". In: *Stochastic Hybrid Systems*. Ed. by H. Blom and J. Lygeros. Vol. 337. Lecture Notes in Control and Information Sciences. Springer, 2006, pp. 107–139.

[PH09] M. Prandini and J. Hu. "Application of reachability analysis for stochastic hybrid systems to aircraft conflict prediction". In: *IEEE Trans. Automat. Contr.* 54.4 (2009), pp. 913–917.

[PJ09] G. O. Passmore and P. B. Jackson. "Combined decision techniques for the existential theory of the reals". In: *Calculemus/MKM*. Ed. by J. Carette, L. Dixon, C. S. Coen, and S. M. Watt. Vol. 5625. Lecture Notes in Computer Science. Springer, 2009, pp. 122–137. ISBN: 978-3-642-02613-3.

[PJP07] S. Prajna, A. Jadbabaie, and G. J. Pappas. "A framework for worst-case and stochastic safety verification using barrier certificates". In: *IEEE TAC* 52.8 (2007), pp. 1415–1429.

[PQ08] A. Platzer and J.-D. Quesel. "Logical verification and systematic parametric analysis in train control". In: *HSCC*. Ed. by M. Egerstedt and B. Mishra. Vol. 4981. Lecture Notes in Computer Science. Springer, 2008, pp. 646–649. ISBN: 978-3-540-78928-4.

[Pre+98] J. Preußig, S. Kowalewski, H. Wong-Toi, and T. A. Henzinger. "An algorithm for the approximative analysis of rectangular automata". In: *FTRTFT*. Ed. by A. P. Ravn and H. Rischel. Vol. 1486. Lecture Notes in Computer Science. Springer, 1998, pp. 228–240. ISBN: 3-540-65003-2.

[Put94] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[PW06] A. Podelski and S. Wagner. "Model checking of hybrid systems: from reachability towards stability". In: *HSCC*. Ed. by J. P. Hespanha and A. Tiwari. Vol. 3927. Lecture Notes in Computer Science. Springer, 2006, pp. 507–521. ISBN: 3-540-33170-0.

[Rat06] S. Ratschan. "Efficient solving of quantified inequality constraints over the real numbers". In: *ACM Trans. Comput. Log.* 7.4 (2006), pp. 723–748.

[Rat10] S. Ratschan. "Safety verification of non-linear hybrid systems is quasi-semidecidable". In: *TAMC*. Ed. by J. Kratochvíl, A. Li, J. Fiala, and P. Kolman. Vol. 6108. Lecture Notes in Computer Science. Springer, 2010, pp. 397–408. ISBN: 978-3-642-13561-3.

[RLJ11] M. Rutkowski, R. Lazić, and M. Jurdziński. "Average-price-per-reward games on hybrid automata with strong resets". In: *STTT* 13.6 (2011), pp. 553–569.

[RS07] S. Ratschan and Z. She. "Safety verification of hybrid systems by constraint propagation-based abstraction refinement". In: *ACM Trans. Embedded Comput. Syst.* 6.1 (2007).

[Seg95] R. Segala. "Modeling and Verification of Randomized Distributed Real-time Systems". PhD thesis. Massachusetts Institute of Technology, 1995.

[SL95] R. Segala and N. A. Lynch. "Probabilistic simulations for probabilistic processes". In: *Nord. J. Comput.* 2.2 (1995), pp. 250–273.

[Spr00] J. Sproston. "Decidable model checking of probabilistic hybrid automata". In: *FTRTFT*. Ed. by M. Joseph. Vol. 1926. Lecture Notes in Computer Science. Springer, 2000, pp. 31–45. ISBN: 3-540-41055-4.

[Spr01] J. Sproston. "Model Checking for Probabilistic Timed and Hybrid Systems". PhD thesis. School of Computer Science, The University of Birmingham, 2001.

[Spr11] J. Sproston. "Discrete-time verification and control for probabilistic rectangular hybrid automata". In: *QEST*. IEEE Computer Society, 2011, pp. 79–88. ISBN: 978-1-4577-0973-9.

[TA12] I. Tkachev and A. Abate. "Regularization of Bellman equations for infinite-horizon probabilistic properties". In: *HSCC*. Ed. by T. Dang and I. M. Mitchell. ACM, 2012, pp. 227–236. ISBN: 978-1-4503-1220-2.

[TCJ10] T. Touili, B. Cook, and P. Jackson, eds. *Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings*. Vol. 6174. Lecture Notes in Computer Science. Springer, 2010. ISBN: 978-3-642-14294-9.

[Tei12] T. Teige. "Stochastic Satisfiability Modulo Theories: A Symbolic Technique for the Analysis of Probabilistic Hybrid Systems". PhD thesis. Carl von Ossietzky Universität Oldenburg, 2012.

[TF09] T. Teige and M. Fränzle. "Constraint-based analysis of probabilistic hybrid systems". In: *adhs*. Ed. by A. Giua, C. Mahulea, M. Silva, and J. Zaytoon. IFAC, 2009, pp. 162–167.

[Var85] M. Y. Vardi. "Automatic verification of probabilistic concurrent finite-state programs". In: *FOCS*. IEEE Computer Society, 1985, pp. 327–338.

[Wac11] B. Wachter. "Refined Probabilistic Abstraction". PhD thesis. Saarland University, 2011. ISBN: 978-3-8325-2764-8.

[Wol12] N. Wolovick. "Continuous Probability and Nondeterminism in Labeled Transition Systems". PhD thesis. FaMAF - UNC, Córdoba, Argentina, 2012.

[WZ10] B. Wachter and L. Zhang. "Best probabilistic transformers". In: *VMCAI*. Ed. by G. Barthe and M. V. Hermenegildo. Vol. 5944. Lecture Notes in Computer Science. Springer, 2010, pp. 362–379. ISBN: 978-3-642-11318-5.

[Ye+06] P. Ye, E. Entcheva, S. A. Smolka, M. R. True, and R. Grosu. "Hybrid automata as a unifying framework for modeling excitable cells". In: *Conf Proc IEEE Eng Med Biol Soc* 1 (2006), pp. 4151–4. ISSN: 1557-170X.

[ZBL07] S. M. Zanoli, L. Barboni, and T. Leo. "An application of hybrid automata for the MIMO model identification of a gasification plant". In: *SMC*. IEEE, 2007, pp. 1427–1432.

[Zha+10] L. Zhang, Z. She, S. Ratschan, H. Hermanns, and E. M. Hahn. "Safety verification for probabilistic hybrid systems". In: *CAV*. Ed. by T. Touili, B. Cook, and P. Jackson. Vol. 6174. Lecture Notes in Computer Science. Springer, 2010, pp. 196–211. ISBN: 978-3-642-14294-9.

[Zha+11] L. Zhang, Z. She, S. Ratschan, H. Hermanns, and E. M. Hahn. "Safety verification for probabilistic hybrid systems". In: *European Journal of Control* (2011).

[Ábr+10] E. Ábrahám, N. Jansen, R. Wimmer, J.-P. Katoen, and B. Becker. "DTMC model checking by SCC reduction". In: *QEST*. IEEE Computer Society, 2010, pp. 37–46. ISBN: 978-0-7695-4188-4.