Rewarding Probabilistic Hybrid Automata

Ernst Moritz Hahn
University of Oxford, United Kingdom*

Holger Hermanns
Saarland University, Germany

ABSTRACT

The joint consideration of randomness and continuous time is important for the formal verification of many real systems. Considering both facets is especially important for wireless sensor networks, distributed control applications, and many other systems of growing importance. Apart from proving the quantitative safety of such systems, it is important to analyse properties related to resource consumption (energy, memory, bandwidth, etc.) and properties that lie more on the economical side (monetary gain, the expected time or cost until termination, etc.). This paper provides a framework to decide such *reward* properties effectively for a generic class of models which have a discrete-continuous behaviour and involve both probabilistic as well as nondeterministic decisions. Experimental evidence is provided demonstrating the applicability of our approach.

Categories and Subject Descriptors: I.6.4 [Computing Methodologies]: Simulation and Modelling - Model Validation and Analysis; G.3 [Mathematics of Computing]: Probability and Statistics.
General Terms: Performance; Reliability; Verification.
Keywords: probabilistic hybrid automaton; abstraction; model checking; expected rewards; probabilistic automaton; performance evaluation; performability; continuous time; nondeterminism; simulation relation.

1. INTRODUCTION

The inclusion of stochastic phenomena in the hybrid systems framework is crucial for a spectrum of application domains, ranging from wireless communication and control to air traffic management and to electric power grid operation [13, 23]. As a consequence, many different stochastic hybrid system models have been proposed [2, 35, 11, 12, 1, 28], together with a vast body of mathematical tools and techniques.

Recently, model checkers for stochastic hybrid systems have emerged [36, 40, 17]. In this context, the model of probabilistic hybrid automata [35] is of particular interest, since it pairs expressiveness and modelling convenience in a way that model checking is indeed possible. In particular, it enables to piggyback [40, 17] the solution of quantitative probabilistic model checking problems on qualitative model checking approaches for hybrid systems. Solvers such as HSolver [31], PHAVer [19], or SpaceEx [20] can be employed for the latter. Thus far, the prime focus in this context has been put on approximating or bounding reach probabilities in probabilistic hybrid automata. This is appropriate for quantifying system safety and reliability, but not for availability, survivability, throughput and resource consumption questions.

In this paper, we aim to overcome this restriction in a framework that is as general as possible, while retaining the idea of piggybacking on existing hybrid system solvers. Taking up initial ideas [22], we decorate probabilistic hybrid automata with *rewards*, which can be considered as costs or bonuses. We discuss a method for handling properties that quantify expected rewards. The properties we consider are either the *minimal or maximal expected total accumulated reward over all executions of a model* or the *minimal or maximal expected time-average reward over all executions of the model*. We will need to postpone the precise formalisation of these notions (cf. Definition 15), until we have defined the semantics of our models. Using appropriate reward structures and property types, this approach allows us to reason about the cost accumulated until termination, the long-run cost of system operation, system availability [16] and survivability [14], time or cost until stabilisation, and many other properties of interest. Proofs backing up the results presented in this paper can be found in [21].

Related Work. Reward properties for classical (i.e. nonstochastic) timed automata have been considered by Bouyer et al. [10, 9]. Rutkowski et al. [32] considered a controller synthesis problem for average-reward properties in classical hybrid automata. Discrete-time stochastic hybrid automata have been considered for the analysis of reward properties [37, 36, 18], and have been studied with importance sampling techniques recently [41]. Methods which approximate continuous-time stochastic hybrid automata by Markov chains [29, 24] also allow for an extension to reward-based properties. To the best of our knowledge, the present paper is the first to address reward-based properties of probabilistic hybrid automata involving nondeterminism, stochastic behaviour as well as continuous time in full generality, harvesting well-understood and effective methods originally developed for the verification of classical hybrid automata.

2. PROBABILISTIC HYBRID AUTOMATA

This section introduces the notion of probabilistic hybrid automata we are going to use, and describes how rewards are integrated into the model. To get started, we first define a generic multi-dimensional *post operator*, which will be used to describe the continuous behaviour of our model. In this operator, we reserve the first two dimensions for the accumulation of reward, respectively the advance of time. In the context of hybrid systems [3, 4], post operators are often described by differential (in)equations. However, our notion is independent of the formalism used.

Definition 1. *A k-dimensional* post operator *with $k \in \mathbb{N}$ and $k \geq 2$ is a function*

$$Post\colon \mathbb{R}^k \to 2^{\mathbb{R}^k}.$$

*Part of this work was done while Ernst Moritz Hahn was with Saarland University.

HSCC'13, April 8–11, 2013, Philadelphia, Pennsylvania, USA.

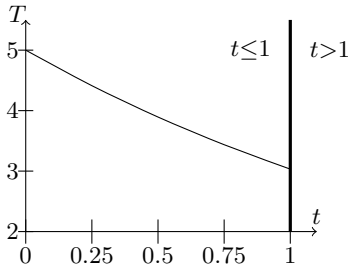

Figure 1: Post operator.

$Post(r, t, v)$ will be used to describe the possible values of the continuous variables after a timed transition. This implies an update of the reward and the time dimension. If there is a constant $c \in \mathbb{R}_{\geq 0}$ satisfying that for any $r, t \in \mathbb{R}$, $v \in \mathbb{R}^{k-2}$,

$$Post(r, t, v) \subseteq \{(r + c\mathsf{t}, t + \mathsf{t}) \mid \mathsf{t} \in \mathbb{R}_{\geq 0}\} \times \mathbb{R}^{k-2},$$

we call the post operator *reward affine*. Models which use only reward affine post operators will turn out to allow for abstractions which are particularly precise.

EXAMPLE 1. *Consider* $Post_{\text{Check}} \colon \mathbb{R}^3 \to 2^{\mathbb{R}^3}$ *with*

$$Post_{\text{Check}}(r, t, T) \stackrel{\text{def}}{=} \{(r, t + \mathsf{t}, T\exp(-0.5\mathsf{t})) \mid \mathsf{t} \in \mathbb{R}_{\geq 0} \wedge \mathsf{t} + t \leq 1\}.$$

For $(r, t, T) = (0, 0, 5)$*, the behaviour is depicted in Figure 1. 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 passed as well as the continuous variable* t *(and here also the value of variable* r*). The axis* T *displays the third dimension. After time* 0.25*, it has a value of* ≈ 4.41*.*

Post operators will appear in the definition of the probabilistic hybrid automaton model we consider. As a preparation, we first define classical hybrid automata.

DEFINITION 2. *A* classical hybrid automaton (HA) *is a tuple*

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

- M *is a finite set of* modes*,*
- $k \in \mathbb{N}$ *with* $k \geq 2$ *is the* dimension*,*
- $\overline{m} \in M$ *is the* initial mode*,*
- $Post_m$ *is a* k*-dimensional post operator for each* m*,*
- $Cmds$ *is a finite set of* guarded commands *of the form*

$$g \to u, \text{ where}$$

  - $g \subseteq M \times \mathbb{R}^k$ *is a* guard*,*
  - $u \colon (M \times \mathbb{R}^k) \to 2^{M \times \mathbb{R}^k}$ *is an* update function *with*
  - $u(s) \subseteq M \times \{(0, 0)\} \times \mathbb{R}^{k-2}$*,*
  - *if* $s \in g$ *then* $u(s) \neq \emptyset$*,*

- *for each* $s = (m, v)$ *with* $Post_m(v) = \emptyset$*, there is a command with guard* g *with* $s \in g$*, and*
- $Rew \colon ((M \times \mathbb{R}^k) \times Cmds) \to \mathbb{R}_{\geq 0}$ *is a* reward structure*.*

The continuous-time behaviour of an HA in a given mode m is determined by the corresponding post operator. Whenever the guard of a guarded command is satisfied, the command can be executed in zero time. If executed, a nondeterministic choice over successor updates and modes results. Multiple guards of commands may be satisfied at the same time, implying a nondeterministic selection over these commands.

Another obvious concept needed for the setting considered is that of a probability distribution.

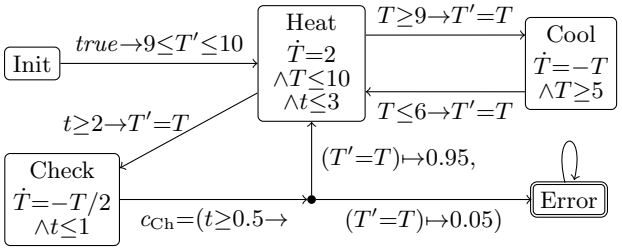

Figure 2: PHA modelling a thermostat.

DEFINITION 3. *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 it is* $\sum_{a \in \Omega} \mu(a) = 1$*. 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 *pairwise different 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 use* $[a_1 \mapsto p_1, \ldots, a_n \mapsto p_n]$ *to denote the probability distribution* μ *with* $\mu(a_i) = p_i$*.*

With this extension, we can now specify probabilistic hybrid automata (similar to Sproston [35, Section 2]).

DEFINITION 4. *A* probabilistic hybrid automaton (PHA) *is a tuple*

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

where all components of this tuple are as in Definition 2, and satisfy the same constraints, except for

- $Cmds$*, which is a finite set of* probabilistic guarded commands *of the form*

$$g \to [u_1 \mapsto p_1, \ldots, u_n \mapsto p_n], \text{ where}$$

  - $g \subseteq M \times \mathbb{R}^k$ *is a* guard*,*
  - $u_i \colon (M \times \mathbb{R}^k) \to 2^{M \times \mathbb{R}^k}$ *is an* update function*,*
  - $u_i(s) \subseteq M \times \{(0, 0)\} \times \mathbb{R}^{k-2}$*,*
  - *if* $s \in g$ *then* $u_i(s) \neq \emptyset$ *for* $1 \leq i \leq n$*.*

$\mathcal{H}$ *is* reward affine *if all its post operators are reward affine and if* $Rew(\cdot, c)$ *is constant for all* $c \in Cmds$*. Under these assumptions we can consider* Rew *as a function*

$$Rew \colon Cmds \to \mathbb{R}_{\geq 0}.$$

Probabilities are incorporated in this definition as part of the commands that update mode and variables according to the probabilities p_i associated with the ith update option u_i.

A HA can now be viewed as a PHA where each guarded command has only a single update option, to be chosen by a Dirac distribution, or a (possibly uncountable) nondeterministic choice over Dirac distributions.

EXAMPLE 2. *Figure 2 depicts a PHA model of a simple unreliable thermostat, where* $\dot{r} = 0$ *and* $\dot{t} = 1$ *in each mode. The model can switch between modes* Heat *and* Cool *to adjust the temperature of its environment. At certain occasions the system may enter a* Check *mode. The post operator of* Check *has been described in Example 1, the other ones are similar. On execution of the command* c_{Ch}*, the system moves to mode* Heat *with probability* 0.95*, and to mode* Error *with probability* 0.05*. We thus have*

$$c_{\text{Ch}} = (g \to [u_{\text{ChH}} \mapsto 0.95, u_{\text{ChE}} \mapsto 0.05]), \text{ where}$$

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

The formalisation of the other commands is similar, but does not include nontrivial probabilities.

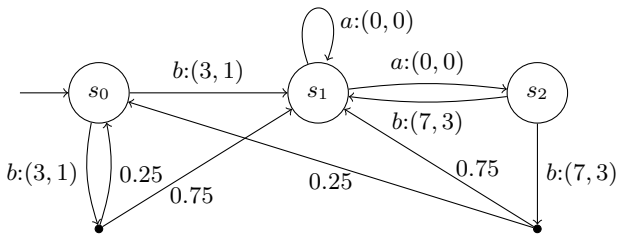

Figure 3: Probabilistic automaton.

In the post operators, we have already integrated means to refer to reward and time. Indeed, the first two dimensions of a PHA are used to record reward accumulation, and time advances as the system lifetime time progresses. We will "collect" the reward and times whenever a command is executed, and therefore reset these dimensions whenever executing commands. The component *Rew* associates rewards to discrete transitions of a PHA.

As time and reward are present as explicit dimensions in our construction, guards and invariants can relate to them. This is in contrast to e.g. priced timed automata [10], where this is forbidden, so as to avoid crossing the undecidability boundary. In the setting considered here, this makes no difference because we build on machinery (for HAs) that is developed for undecidable theories, in the form of heuristics.

EXAMPLE 3. *Consider the thermostat of Figure 2. Here it is not possible to leave the* Error *mode once entered. We define a reward structure* Rew_{acc} *with* $Rew_{\text{acc}}(c) \stackrel{\text{def}}{=} 0$ *if* $c \in \{c_{\text{E}}, c_{\text{IH}}\}$ *and* $Rew_{\text{acc}}(c) \stackrel{\text{def}}{=} 1$ *else. The system thus earns a reward for executing any command, except for the one of* Error*, and the one to initialise the system. With this reward structure, the minimal sum of reward values accumulated, expresses the minimal expected number of commands executed until an error happens.*

Now, assume we have extended mode Error *so that the system can recover after a certain time (e.g. by adding a reset transition back to the initial state). In such a system, it makes sense to consider the long-run behaviours. We can for instance look at a reward structure* Rew_{lra} *assigning constant* 0 *to each command. If in addition we modify the post operator in such a way that* r *is increased by* 1 *per time unit in mode* Error*, we can use this to reason about the percentage of time the system is not operational on the long run.*

3. PHA SEMANTICS

In this section, we describe the semantics of PHAs. The semantics maps on variations of infinite-state Markov decision processes [30], known as probabilistic automata [33].

DEFINITION 5. *A* probabilistic automaton (PA) *is a tuple*

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

- 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 state-action pairs.*

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

PAs contain a (possibly uncountable) set of states, whereof one is initial. In each $s \in S$, there is a nondeterministic choice of actions $a \in Act$ and distributions $\mu \in \mathcal{T}(s,a)$ over successor states.

EXAMPLE 4. *In Figure 3, we depict a finite example PA* $\mathcal{M} \stackrel{\text{def}}{=} (S, \overline{s}, Act, \mathcal{T})$*. Here, we have* $S \stackrel{\text{def}}{=} \{s_0, s_1, s_2\}$*,* $\overline{s} \stackrel{\text{def}}{=} s_0$*,* $Act = \{a, b\}$*, and*

$$\mathcal{T}(s_0, a) \stackrel{\text{def}}{=} \emptyset, \mathcal{T}(s_0, b) \stackrel{\text{def}}{=} \{[s_0 \mapsto 0.25, s_1 \mapsto 0.75], [s_1 \mapsto 1]\},$$
$$\mathcal{T}(s_1, a) \stackrel{\text{def}}{=} \{[s_1 \mapsto 1], [s_2 \mapsto 1]\}, \mathcal{T}(s_1, b) \stackrel{\text{def}}{=} \emptyset,$$
$$\mathcal{T}(s_2, a) \stackrel{\text{def}}{=} \emptyset, \mathcal{T}(s_2, b) \stackrel{\text{def}}{=} \{[s_1 \mapsto 1], [s_0 \mapsto 0.25, s_1 \mapsto 0.75]\}.$$

DEFINITION 6. *A* finite path *of a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *is a tuple*

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

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

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

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* $\mathcal{M}$*.*

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'*.*

By $\text{trace}(\beta_{\text{fin}}) = a_0 a_1 \dots a_{n-1}$ *we denote the* trace *of a finite path, 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* $\gamma = a_0 a_1 \dots \in Trace^*_{\mathcal{M}} \uplus Trace^\omega_{\mathcal{M}}$*, 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 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 5. *A finite path in the PA of Figure 3 is*

$$\beta_{\text{fin}} \stackrel{\text{def}}{=} s_0 b [s_0 \mapsto 0.25, s_1 \mapsto 0.75] s_0,$$

and with $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$*, an* Act_{fair}*-fair infinite path is*

$$\beta_{\text{inf}} \stackrel{\text{def}}{=} s_0 (b [s_0 \mapsto 0.25, s_1 \mapsto 0.75] s_0)^\omega.$$

We have

$$\text{trace}(\beta_{\text{fin}}) = b, \text{trace}(\beta_{\text{inf}}) = b^\omega,$$
$$\text{last}(\beta_{\text{fin}}) = s_0, \beta_{\text{fin}}[0] = s_0, \beta_{\text{inf}}[15] = s_0.$$

Path sets are by themselves not sufficient to describe the properties of PAs. This is because nondeterministic behaviour is intertwined with probabilistic behaviour. This asks for instances to resolve the nondeterminism. These instances, called *schedulers*, induce a purely probabilistic behaviour, which can be subjected to stochastic analyses.

DEFINITION 7. *A* scheduler *for PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *is a function* $\rho \colon Path^{\text{fin}}_{\mathcal{M}} \to Distr(Act \times Distr(S))$*. For* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$*, we require* $\rho(\beta)(a, \mu) > 0$ *implies* $\mu \in \mathcal{T}(\text{last}(\beta), a)$*. With* $Sched_{\mathcal{M}}$ *we denote the set of schedulers of* $\mathcal{M}$*.*

A scheduler ρ *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* $\rho(\beta) = \rho(\beta')$*. We can interpret it as being of the form* $\rho \colon S \to (Act \times Distr(S))$*.*

In this paper, we develop results valid for general schedulers, thus not restricted to simple schedulers. But since simple schedulers are simpler to describe and understand, they appear in illustrating examples. We are now in the position to define probability measures on paths.

DEFINITION 8. *We define* $Pr_{\mathcal{M},\rho} \colon Path^{\text{fin}}_{\mathcal{M}} \to [0,1]$ *for PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and scheduler* $\rho \colon Path^{\text{fin}}_{\mathcal{M}} \to Distr(Act \times$

Distr(S)*): Given* $\beta = s_0 a_0 \mu_0 s_1 a_1 \mu_1 \dots s_n \in Path^{\text{fin}}_{\mathcal{M}}$*, let*

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

We define the cylinder *of* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$

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

as the set of infinite paths which start with β*. Then, we use the generated* σ*-algebra*

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

to obtain the measurable space $(Path^{\text{inf}}_{\mathcal{M}}, \Sigma_{\mathcal{M}})$*.*

There is a unique extension [27] of $Pr_{\mathcal{M},\rho}\colon Path^{\text{fin}}_{\mathcal{M}} \to [0,1]$ *to* $Pr_{\mathcal{M},\rho}\colon \Sigma_{\mathcal{M}} \to [0,1]$ *where for all* $\beta \in Path^{\text{fin}}_{\mathcal{M}}$ *it is*

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

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

Definition 9. *A scheduler* $\rho \in Sched_{\mathcal{M}}$ *of a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *is called* Act_{fair}-fair *for* $Act_{\text{fair}} \subseteq Act$ *if*

$$Pr_{\mathcal{M},\rho}(Path^{Act_{\text{fair}}}_{\mathcal{M}}) = 1.$$

$Sched^{Act_{\text{fair}}}_{\mathcal{M}}$ *denotes the set of* Act_{fair}*-fair schedulers of* $\mathcal{M}$*.*

We define two stochastic processes associated to a PA.

Definition 10. *Let* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *be a PA. We define the* state process *and the* action process *of* $\mathcal{M}$ *as*

$$X^{\mathcal{M}}\colon (Path^{\text{inf}}_{\mathcal{M}} \times \mathbb{N}) \to S \text{ with } X^{\mathcal{M}}(\beta, n) \stackrel{\text{def}}{=} \beta[n],$$
$$Y^{\mathcal{M}}\colon (Path^{\text{inf}}_{\mathcal{M}} \times \mathbb{N}) \to Act \text{ with } Y^{\mathcal{M}}(\beta, n) \stackrel{\text{def}}{=} \text{trace}(\beta)[n]$$

for $\beta \in Path^{\text{inf}}_{\mathcal{M}}$ *and* $n \in \mathbb{N}$*.*

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.

Definition 11. *Given PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$*, a* reward structure *is a pair* $(rew_{\text{num}}, rew_{\text{den}})$ *of two functions*

$$rew_{\text{num}}, rew_{\text{den}}\colon (S \times Act) \to \mathbb{R}_{\geq 0}.$$

Given a 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* $\mathcal{T}(s,a) \neq \emptyset$ *it is*

$$rew_{\text{num}}(s,a) = mul_a rew_{\text{den}}(s,a) + add_a.$$

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}, for the other one we only need rew_{num}.

Example 6. *In Figure 3 we depict a PA along with a reward structure* $rew \stackrel{\text{def}}{=} (rew_{\text{num}}, rew_{\text{den}})$*. We have*

$$rew_{\text{num}}(s_0,b) \stackrel{\text{def}}{=} 3, rew_{\text{den}}(s_0,b) \stackrel{\text{def}}{=} 1,$$
$$rew_{\text{num}}(s_1,a) \stackrel{\text{def}}{=} 0, rew_{\text{den}}(s_1,a) \stackrel{\text{def}}{=} 0,$$
$$rew_{\text{num}}(s_2,b) \stackrel{\text{def}}{=} 7, rew_{\text{num}}(s_2,b) \stackrel{\text{def}}{=} 3.$$

The reward structure is affine, and for action b *we have the factors* $mul_b = 2$ *and* $add_b = 1$*.*

We define properties based on these reward structures.

Definition 12. *Given a PA* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *together with a reward structure* $rew = (rew_{\text{num}}, rew_{\text{den}})$ *and* $\rho \in Sched_{\mathcal{M}}$*, the* accumulated reward *is the expectation*

$$\text{val}^{\rho}_{\mathcal{M},rew,\text{acc}} \stackrel{\text{def}}{=} E_{\mathcal{M},\rho}\left[\lim_{n\to\infty} \sum_{i=0}^{n} rew_{\text{num}}(X^{\mathcal{M}}_i, Y^{\mathcal{M}}_i)\right]$$

under the probability measure $Pr_{\mathcal{M},\rho}$*. The* fractional long-run average reward *is defined as*

$$\text{val}^{\rho}_{\mathcal{M},rew,\text{lra}} \stackrel{\text{def}}{=} E_{\mathcal{M},\rho}\left[\lim_{n\to\infty} \frac{\sum_{i=0}^{n} rew_{\text{num}}(X^{\mathcal{M}}_i, Y^{\mathcal{M}}_i)}{\sum_{i=0}^{n} rew_{\text{den}}(X^{\mathcal{M}}_i, Y^{\mathcal{M}}_i)}\right],$$

in case we have $Pr_{\mathcal{M},\rho}(A) = 1$*, 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 maximal value*

$$\text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} = \sup_{\rho \in Sched^{Act_{\text{fair}}}_{\mathcal{M}}} \text{val}^{\rho}_{\mathcal{M},rew,\text{lra}},$$

and accordingly for accumulated rewards and minimal values. 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* ρ *for which* $Pr_{\mathcal{M},\rho}(A) = 1$*.*

There are more complicated notions of fractional long-run averages which are well-defined on all paths [38]. They agree with the definition above if it exists, which we use for clarity. We will later on use reward-extended PAs as the semantics of 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 or the number of steps until an error is reached. 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 of 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 such models, not all steps take the same duration of time to be performed.

Later on, the timed transitions will correspond to the non-fair actions. If a scheduler would be allowed to choose an infinite number of timed actions, it could let the time flow 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, which we consider as being unrealistic. Fairness can prevent this problem.

Example 7. *Reconsider the PA of Figure 3. In this model, the accumulated reward would be infinite, so that we only consider the fractional long-run average reward. We derive for* $Act_{\text{fair}} \stackrel{\text{def}}{=} \{b\}$ *that*

$$\text{val}^{+,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} = \frac{12}{5} = 2.4, \text{val}^{-,Act_{\text{fair}}}_{\mathcal{M},rew,\text{lra}} = \frac{7}{3} \approx 2.333,$$

which is for instance obtained by the simple schedulers ρ^+ *and* ρ^- *with*

$$\rho^+(s_0) \stackrel{\text{def}}{=} (b, [s_0 \mapsto 0.25, s_1 \mapsto 0.75]), \rho^+(s_1) \stackrel{\text{def}}{=} (a, [s_2 \mapsto 1]),$$
$$\rho^+(s_2) \stackrel{\text{def}}{=} (b, [s_0 \mapsto 0.25, s_1 \mapsto 0.75]),$$
$$\rho^-(s_0) \stackrel{\text{def}}{=} (b, [s_1 \mapsto 1]), \rho^-(s_1) \stackrel{\text{def}}{=} (a, [s_2 \mapsto 1]),$$
$$\rho^-(s_2) \stackrel{\text{def}}{=} (a, [s_1 \mapsto 1]).$$

We now define the semantics of PHAs as PAs formally.

Definition 13. *For PHA* $\mathcal{H} = (M, k, \overline{m}, \langle Post_m \rangle_{m \in M}, Cmds, Rew)$*, the* semantics *is a PA*

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

- $S \stackrel{\text{def}}{=} M \times \mathbb{R}^k$*,*
- $\overline{s} \stackrel{\text{def}}{=} (\overline{m}, 0, \dots, 0)$*,*
- $Act \stackrel{\text{def}}{=} Cmds \uplus \{\tau\}$*,*
- *for* $s = (m, v) \in S$ *we have that*

– *for* $c = (g \to [u_1 \mapsto p_1, \ldots, u_n \mapsto p_n]) \in \mathit{Cmds}$ *it is* $\mathcal{T}(s,c) \stackrel{\text{def}}{=} \emptyset$ *if* $s \notin g$ *and else:*

$$\mathcal{T}(s,c) \stackrel{\text{def}}{=} \{\mu \in \mathit{Distr}(S) \mid \exists s'_1 \in u_1(s), \ldots, s'_n \in u_n(s).\ \forall s' \in S.\ \mu(s') = \sum_{s'_i = s'} p_i\},$$

– *it is* $\mathcal{T}(s,\tau) \stackrel{\text{def}}{=} \{[(m,v') \mapsto 1] \mid v' \in \mathit{Post}_m(v)\}$.

The semantics is similar to usual notions of HAs, which are usually given in terms of *labelled transition systems* [26]. The difference is in the probabilistic guarded commands, where we can have a probabilistic choice over successor states in addition to nondeterminism.

EXAMPLE 8. *Consider the PHA* $\mathcal{H}$ *of Figure 2. Then*

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

- $S = M \times \mathbb{R}^3$,
- $\overline{s} = (\text{Init}, 0, 0, 0)$,
- $\mathit{Act} = \{c_{\text{IH}}, c_{\text{HCo}}, c_{\text{HCh}}, c_{\text{CoH}}, c_{\text{E}}, c_{\text{Ch}}, \tau\}$,
- $\mathcal{T}\colon (S \times \mathit{Act}) \to 2^{\mathit{Distr}(S)}$.

For $s = (\text{Check}, r, t, T) \in \{\text{Check}\} \times \mathbb{R}^3$, *we have*

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

it is $\mathcal{T}(s, c_{\text{Ch}}) = \emptyset$ *if* $t < 0.5$ *and we have* $\mathcal{T}(s, c_{\text{Ch}}) = \{[(\text{Heat}, 0, 0, T) \mapsto 0.95, (\text{Error}, 0, 0, T) \mapsto 0.05]\}$ *else. Further,*

$$\mathcal{T}(s,\tau) = \{[(\text{Check}, r, t + \mathtt{t}, T\exp(-0.5\mathtt{t})) \mapsto 1] \mid \mathtt{t} \in \mathbb{R}_{\geq 0} \wedge \mathtt{t} + t \leq 1\}.$$

With these preparations, we can define the semantics of reward structures of PHAs.

DEFINITION 14. *Given PHA* $\mathcal{H} = (M, k, \overline{m}, \langle \mathit{Post}_m \rangle_{m \in M}, \mathit{Cmds}, \mathit{Rew})$ *with semantics* $[\![\mathcal{H}]\!] = (S, \overline{s}, \mathit{Act}, \mathcal{T})$, *the* reward semantics *is the reward structure* $\text{rew}(\mathcal{H}) \stackrel{\text{def}}{=} (\mathit{rew}_{\text{num}}, \mathit{rew}_{\text{den}})$ *associated to* $[\![\mathcal{H}]\!]$. *For* $s = (m, r, t, v) \in S$ *and* $c \in \mathit{Cmds}$, *let*

$$\mathit{rew}_{\text{num}}(s,c) \stackrel{\text{def}}{=} \mathit{Rew}(s,c) + r, \mathit{rew}_{\text{den}}(s,c) \stackrel{\text{def}}{=} t,$$

and $\mathit{rew}_{\text{num}}(s,\tau) \stackrel{\text{def}}{=} \mathit{rew}_{\text{den}}(s,\tau) \stackrel{\text{def}}{=} 0$.

In this definition, whenever a command is executed, the reward of this command is obtained. Additionally, the timed rewards and the time accumulated until the execution of this command become effective here. As in our model only paths of infinitely many commands are relevant, this is equivalent to a reward semantics in which rewards and passage of time are attached directly to timed transitions. By postponing the collection of timed rewards to the execution of subsequent commands, we will be able to simplify the computation of abstractions of PHAs.

We are now in the position to define the values of reward properties, the technical core of our approach, using the semantics of PHAs and their reward structures.

DEFINITION 15. *Given PHA* $\mathcal{H} = (M, k, \overline{m}, \langle \mathit{Post}_m \rangle_{m \in M}, \mathit{Cmds}, \mathit{Rew})$, *we define the* maximal and minimal time-average reward *as*

$$\text{val}^{+}_{\mathcal{H},\text{lra}} \stackrel{\text{def}}{=} \text{val}^{+,\mathit{Cmds}}_{[\![\mathcal{H}]\!],\text{rew}(\mathcal{H}),\text{lra}} \textit{ and } \text{val}^{-}_{\mathcal{H},\text{lra}} \stackrel{\text{def}}{=} \text{val}^{-,\mathit{Cmds}}_{[\![\mathcal{H}]\!],\text{rew}(\mathcal{H}),\text{lra}}$$

and define the accumulated rewards *accordingly as*

$$\text{val}^{+}_{\mathcal{H},\text{acc}} \stackrel{\text{def}}{=} \text{val}^{+,\mathit{Cmds}}_{[\![\mathcal{H}]\!],\text{rew}(\mathcal{H}),\text{acc}} \textit{ and } \text{val}^{-}_{\mathcal{H},\text{acc}} \stackrel{\text{def}}{=} \text{val}^{-,\mathit{Cmds}}_{[\![\mathcal{H}]\!],\text{rew}(\mathcal{H}),\text{acc}}.$$

We only optimise over fair schedulers, because otherwise, we could assign a relevant probability mass to paths which are *time convergent* [8, Chapter 9], that is their trace will end in a sequence $\tau\tau\tau\ldots$ corresponding to time durations $\mathtt{t}_0\mathtt{t}_1\mathtt{t}_2, \ldots$ with $\sum_{i=0}^{\infty} \mathtt{t}_i < \infty$. This way, time effectively stops, which means that only the reward up to this point of time will be taken into account, which is unrealistic. Now assume that infinitely many commands are executed, and that the automaton is *structurally (strongly) nonzeno* [6][5, Definition 6], that is the guards are defined so that they cannot be executed without a minimal fixed delay. Then this ensures the time divergence of the path.

One point is worth noting. One might be interested in models in which it is a legal behaviour to eventually reside in a mode m of the model without further executing any commands. However, the definition above requires that infinitely many commands are executed on each legal path. Because of this, such models have to be adapted accordingly. This can be done, e.g. by adding a new auxiliary command c_m which can be executed infinitely often after a given delay whenever residing in m.

3.1 Expressing Properties

Table 1 provides an overview of common properties that are expressible using the mechanisms described. Here, **F** denotes the set of failed states of the PHA, and **T** describes states in which operation has terminated. The *availability* [16] of a system can then be expressed as a time-average reward value, by specifying the reward of the PHA under consideration so that in each mode in which the system is available the reward increases with rate 1 per time unit and zero else. *Survivability* [14] is the ability of a system to recover a certain quality of service level in a timely manner after a disaster. Here, we consider the maximal expected time needed to recover from an error condition. This can be expressed using expected total rewards, by maximising over all states of the system (or, an abstraction of the system) in which it is not available.

4. ABSTRACTION

In this section, we develop the necessary tools for the abstraction for PHAs and their reward structures. For the theoretical justification of our abstraction method in its entirety we use simulation relations. The relations are actually never constructed during the verification process, just like the full semantics of PHAs, which is our reference semantics, but its construction is prohibitive.

To prove the validity of abstractions of PHAs for reward-based properties, we extend the definition of simulation relations [34, 33] to take into account reward structures. 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* [25, Definition 4.3].

DEFINITION 16. *Let* $\mu \in \mathit{Distr}(S)$ *and* $\mu_{\text{sim}} \in \mathit{Distr}(S_{\text{sim}})$ *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.

Using weight functions, we define simulations.

DEFINITION 17. *Given the two PAs* $\mathcal{M} = (S, \overline{s}, \mathit{Act}, \mathcal{T})$ *and* $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, \mathit{Act}, \mathcal{T}_{\text{sim}})$, *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*

	property	*Rew*	$\dot{r}$	type	remark	
(a)	time until failure	0	$\begin{cases}0 & s \in \mathbf{F}\\ 1 & \text{else}\end{cases}$	minimum accumulated reward	**F** absorbing	[7]
(b)	cost until termination	$\begin{cases}0 & s \in \mathbf{F}\\ \text{any} & \text{else}\end{cases}$	$\begin{cases}0 & s \in \mathbf{T}\\ \text{any} & \text{else}\end{cases}$	maximum accumulated reward	**T** absorbing	[7]
(c)	long-run cost of operation	any	any	maximum long-run average reward	-	[7]
(d)	system availability	0	$\begin{cases}0 & s \in \mathbf{F}\\ 1 & \text{else}\end{cases}$	minimum long-run average reward	-	[16]
(e)	survivability	0	$\begin{cases}1 & s \in \mathbf{F}\\ 0 & \text{else}\end{cases}$	maximum accumulated reward	maximum over $s \in \mathbf{F}$	[14]

Table 1: Overview of expressible reward properties.

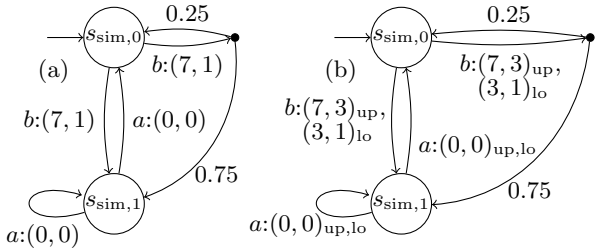

Figure 4: (a) PA simulating the one of Figure 3. (b) PA affinely simulating the one of Figure 3.

1. *it is* $(\overline{s}, \overline{s}_{\text{sim}}) \in R$,
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 our purposes, we must consider rewards of PAs.

Definition 18. *Consider two PAs* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and* $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ *with reward structures* $rew = (rew_{\text{num}}, rew_{\text{den}})$, $rew_{\text{sim}} = (rew_{\text{sim,num}}, rew_{\text{sim,den}})$, *and a simulation relation* R *between the PAs. We say that* R *is* upper-bound compatible, *if in case we have* $(s, s_{\text{sim}}) \in R$ *then for all* $a \in Act$ *it is*

$$rew_{\text{num}}(s,a) \leq rew_{\text{sim,num}}(s_{\text{sim}}, a),$$
$$rew_{\text{den}}(s,a) \geq rew_{\text{sim,den}}(s_{\text{sim}}, a).$$

If there exists such a relation R, *we write*

$$(\mathcal{M}, rew) \overset{\text{up}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}}).$$

We define lower-bound compatible *simulations* R *accordingly by swapping* $\leq$ *and* $\geq$ *above and write*

$$(\mathcal{M}, rew) \overset{\text{lo}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}}).$$

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 1. *For PAs* $\mathcal{M}$ *and* $\mathcal{M}_{\text{sim}}$ *with reward structures* rew *and* rew_{sim}, *if* $(\mathcal{M}, rew) \overset{\text{up}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}})$ *then*

$$\text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}, rew, \text{lra}} \leq \text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}_{\text{sim}}, rew_{\text{sim}}, \text{lra}},$$

accordingly for the accumulated rewards. In case $(\mathcal{M}, rew) \overset{\text{lo}}{\preceq} (\mathcal{M}_{\text{sim}}, rew_{\text{sim}})$, *we obtain lower bounds for minimal values.*

We can thus bound the maximal (minimal) reward values from above (below). The principle 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 9. *In Figure 4 (a) we give a PA with corresponding reward structures which simulates the one of Figure 3 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 to obtain 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 different simulation relation to obtain more precise results.

Definition 19. *Consider two PAs* $\mathcal{M} = (S, \overline{s}, Act, \mathcal{T})$ *and* $\mathcal{M}_{\text{sim}} = (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act, \mathcal{T}_{\text{sim}})$ *between which there is a simulation relation* R. *Consider affine reward structures*

$$rew = (rew_{\text{num}}, rew_{\text{den}}),$$
$$rew_{\text{sim,up}} = (rew_{\text{sim,up,num}}, rew_{\text{sim,up,den}}),$$
$$rew_{\text{sim,lo}} = (rew_{\text{sim,lo,num}}, rew_{\text{sim,lo,den}}).$$

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 11) for each action* a, *that is for* $s \in S$ *and* $\mathbf{z} \in \mathbf{A}$ *it is*

$$rew_{\text{num}}(s,a) = mul_a\, rew_{\text{den}}(s,a) + add_a,$$
$$rew_{\text{sim,up,num}}(\mathbf{z},a) = mul_a\, rew_{\text{sim,up,den}}(\mathbf{z},a) + add_a,$$
$$rew_{\text{sim,lo,num}}(\mathbf{z},a) = mul_a\, rew_{\text{sim,lo,den}}(\mathbf{z},a) + add_a.$$

Then, we define R *as* affine compatible *if for all* $(s, s_{\text{sim}}) \in R$ *and* $a \in Act$ *it is*

$$rew_{\text{sim,lo,num}}(s_{\text{sim}}, a) \leq rew_{\text{num}}(s,a) \leq rew_{\text{sim,up,num}}(s_{\text{sim}}, a),$$
$$rew_{\text{sim,lo,den}}(s_{\text{sim}}, a) \leq rew_{\text{den}}(s,a) \leq rew_{\text{sim,up,den}}(s_{\text{sim}}, a).$$

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 2. *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*

$$\mathcal{M}_{\text{aff}} \overset{\text{def}}{=} (S_{\text{sim}}, \overline{s}_{\text{sim}}, Act \times \{\text{up}, \text{lo}\}, \mathcal{T}_{\text{aff}}),$$
$$rew_{\text{aff}} \overset{\text{def}}{=} (rew_{\text{aff,num}}, rew_{\text{aff,den}}), \text{ where}$$

- *for* $s \in S_{\text{sim}}$ *and* $a \in Act$ *it is* $\mathcal{T}_{\text{aff}}(s, (a, \text{up})) \overset{\text{def}}{=} \mathcal{T}_{\text{aff}}(s, (a, \text{lo})) \overset{\text{def}}{=} \mathcal{T}_{\text{sim}}(s, a)$,
- *for* $s \in S_{\text{sim}}$ *and* $a \in Act$ *it is* $rew_{\text{aff,num}}(s, (a, \text{up})) \overset{\text{def}}{=} rew_{\text{sim,up,num}}(s, a)$, *accordingly for* $rew_{\text{aff,num}}(s, (a, \text{lo}))$, $rew_{\text{aff,den}}(s, (a, \text{lo}))$ *and* $rew_{\text{aff,den}}(s, (a, \text{lo}))$.

Then we have

$$\text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}, rew, \text{lra}} \leq \text{val}^{+, Act_{\text{fair}}}_{\mathcal{M}_{\text{aff}}, rew_{\text{aff}}, \text{lra}},$$

and accordingly for the accumulated rewards and the minimising cases.

Similar to upper-bound and lower-bound compatible simulations, the affinely simulating automaton $\mathcal{M}_{\text{sim}}$ can mimic the behaviours of the simulated one. In $\mathcal{M}_{\text{aff}}$ then, it also mimics the behaviours of the original model, but can 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 that we will obtain results which are more precise is, intuitively, that for nonaffine reward structures we had to bound rew_{num} and rew_{den} from opposite directions.

EXAMPLE 10. *In Figure 4 (b) we give a PA which affinely simulates the one of Figure 3. Maximal and minimal long-run averages are* 3 *and* $\frac{7}{3} \approx 2.333$*, which is more precise than the values obtained from Figure 4 (a) in Example 9.*

We describe abstract state spaces to subsume uncountably many states of the infinite semantics of PHAs.

DEFINITION 20. *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 it is $\bigcup_{(m,\zeta)\in\mathbf{A}} \zeta = \mathbb{R}^k$ 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 [31] we may have overlapping borders, whereas for PHAVER [19] we may also have common interiors of abstract states.

An abstraction of a PHA is defined as follows. There, we will need to transfer probability distributions over the states of the PHA semantics to the states of abstractions.

DEFINITION 21. *Consider an arbitrary PHA $\mathcal{H} = (M, k, \overline{m}, \langle Post_m\rangle_{m\in M}, Cmds, Rew)$, and abstract state space $\mathbf{A} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\}$ of corresponding dimension and modes. We say that*

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

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

- $(\overline{m}, 0, \ldots, 0) \in \overline{\mathbf{z}}$,
- *for all $\mathbf{z} \in \mathbf{A}$, $s \in \mathbf{z}$, $c = (g \to [u_1 \mapsto p_1, \ldots, u_n \mapsto p_n]) \in Cmds$, if $s \in \mathbf{z} \cap g$, then for all*

  $$(s'_1, \ldots, s'_n) \in u_1(s) \times \cdots \times u_n(s)$$

  there are

  $$(\mathbf{z}'_1, \ldots, \mathbf{z}'_n) \in \mathbf{A}^n \text{ with } s_i \in \mathbf{z}_i, 1 \le i \le n$$

  so that there is $\mu \in Distr(\mathbf{A})$ with $\mu(\mathbf{z}') = \sum_{\mathbf{z}'=\mathbf{z}'_i} p_i$ and $\mu \in \mathcal{T}(\mathbf{z}, c)$,
- *for all $\mathbf{z} \in \mathbf{A}$, $s = (m, v) \in \mathbf{z}$ and all $s' = (m, v') \in \{m\} \times Post_m(v)$, 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 $Abs(\mathcal{H}, \mathbf{A})$ we denote the set of all such abstractions.

Next, we equip PHA abstractions with rewards.

DEFINITION 22. *Let $\mathcal{H}$ be a PHA with rewards Rew and consider $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathcal{H}, \mathbf{A})$. The* abstract upper-bound reward structure *is defined as*

$$\text{absup}(\mathcal{H}, \mathcal{M}) \stackrel{\text{def}}{=} (rew_{\text{num}}, rew_{\text{den}}), \text{ where}$$

- *for all $\mathbf{z} \in \mathbf{A}$ it is $rew_{\text{num}}(\mathbf{z}, \tau) \stackrel{\text{def}}{=} rew_{\text{den}}(\mathbf{z}, \tau) \stackrel{\text{def}}{=} 0$,*
- *for all $\mathbf{z} \in \mathbf{A}$ and $c = (g \to [u_1 \mapsto p_1, \ldots, u_n \mapsto p_n]) \in Cmds$ it is*

$$rew_{\text{num}}(\mathbf{z}, c) \stackrel{\text{def}}{=} \sup_{s=(m,r,t,v)\in\mathbf{z}} r + Rew(s, c),$$
$$rew_{\text{den}}(\mathbf{z}, c) \stackrel{\text{def}}{=} \inf_{(m,r,t,v)\in\mathbf{z}} t.$$

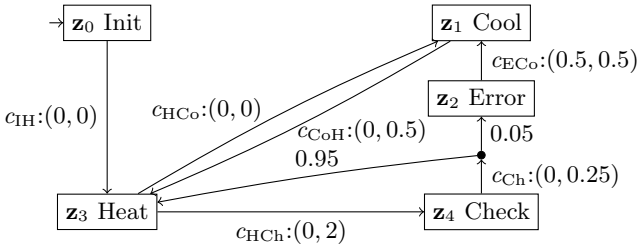

Figure 5: PHA abstraction for rewards.

The abstract lower-bound reward structure abslo *is defined accordingly by swapping* sup *and* inf.

We can use these reward structures to safely bound the reward values of PHAs semantics.

THEOREM 1. *Consider a PHA $\mathcal{H}$ with reward structure Rew and $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathcal{H}, \mathbf{A})$ with $rew_{\text{up}} \stackrel{\text{def}}{=} \text{absup}(\mathcal{H}, \mathcal{M})$. Then*

$$\text{val}^{+}_{\mathcal{H},\text{lra}} \le \text{val}^{+,Cmds}_{\mathcal{M},rew_{\text{up}},\text{lra}},$$

and accordingly for accumulated rewards and minimal values.

The theorem follows by Lemma 1, because abstractions simulate the semantics of PHAs.

EXAMPLE 11. *In Figure 5 we sketch an abstraction of Figure 2 with an abstract upper-bound reward structure for the PHA reward structure Rew_{lra} of Example 3. Thus, we have a reward of* 1 *per time in* Error *and* 0 *else. 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 (which we slightly modify when using this reward structure), we have to wait at least until $t \ge 0.25$ 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 reward affine PHAs, we can use more precise reward structures in the abstraction.

DEFINITION 23. *Consider a reward affine PHA $\mathcal{H}$ with commands Cmds and the reward structure Rew and $\mathcal{M} = (\mathbf{A}, \overline{\mathbf{z}}, Cmds \uplus \{\tau\}, \mathcal{T}) \in Abs(\mathcal{H}, \mathbf{A})$. We define the* affine abstraction

$$\mathcal{M}_{\text{aff}} \stackrel{\text{def}}{=} (\mathbf{A}, \overline{\mathbf{z}}, Act_{\text{aff}}, \mathcal{T}_{\text{aff}}),$$

where $Act_{\text{aff}} \stackrel{\text{def}}{=} \{(\tau, \text{up}), (\tau, \text{lo})\} \uplus \{(c, \text{up}) \mid c \in Cmds\} \uplus \{(c, \text{lo}) \mid c \in Cmds\}$, and define $\mathcal{T}_{\text{aff}}$ as $\mathcal{T}_{\text{aff}}(\mathbf{z}, (a, \text{up})) \stackrel{\text{def}}{=} \mathcal{T}_{\text{aff}}(\mathbf{z}, (a, \text{lo})) \stackrel{\text{def}}{=} \mathcal{T}(\mathbf{z}, a)$. Then, the abstract affine reward structure *is defined as $rew_{\text{aff}} = (rew_{\text{num}}, rew_{\text{den}})$ where*

- *for all $\mathbf{z} \in \mathbf{A}$, let*

  $$rew_{\text{num}}(\mathbf{z}, (\tau, \text{up})) \stackrel{\text{def}}{=} rew_{\text{num}}(\mathbf{z}, (\tau, \text{lo}))$$
  $$\stackrel{\text{def}}{=} rew_{\text{den}}(\mathbf{z}, (\tau, \text{up})) \stackrel{\text{def}}{=} rew_{\text{den}}(\mathbf{z}, (\tau, \text{lo})) \stackrel{\text{def}}{=} 0,$$

- *for all $\mathbf{z} \in \mathbf{A}$ with mode m and $c = (g \to [u_1 \mapsto p_1, \ldots, u_n \mapsto p_n]) \in Cmds$, let*

  $$rew_{\text{num}}(\mathbf{z}, (c, \text{up})) \stackrel{\text{def}}{=} cv_{\text{sup}} + Rew(c),$$
  $$rew_{\text{den}}(\mathbf{z}, (c, \text{up})) \stackrel{\text{def}}{=} v_{\text{sup}},$$

$$rew_{\text{num}}(\mathbf{z},(c,\text{lo})) \stackrel{\text{def}}{=} cv_{\text{inf}} + Rew(c),$$
$$rew_{\text{den}}(\mathbf{z},(c,\text{lo})) \stackrel{\text{def}}{=} v_{\text{inf}},$$

with the factor c of Definition 1 and

$$v_{\text{sup}} \stackrel{\text{def}}{=} \sup\{t \mid (m,r,t,v) \in \mathbf{z}\},$$
$$v_{\text{inf}} \stackrel{\text{def}}{=} \inf\{t \mid (m,r,t,v) \in \mathbf{z}\}.$$

We then define $\text{absaff}(\mathcal{H},\mathcal{M}) \stackrel{\text{def}}{=} (\mathcal{M}_{\text{aff}}, rew_{\text{aff}})$.

THEOREM 2. *Consider a reward affine PHA $\mathcal{H}$ with reward structure Rew and abstraction $\mathcal{M} = (\mathbf{A},\bar{\mathbf{z}},Cmds \uplus \{\tau\},\mathcal{T}) \in Abs(\mathcal{H},\mathbf{A})$, with* $\text{absaff}(\mathcal{H},\mathcal{M}) = (\mathcal{M}_{\text{aff}}, rew_{\text{aff}})$. *Then*

$$\text{val}^{+}_{\mathcal{H},\text{lra}} \leq \text{val}^{+,Cmds}_{\mathcal{M}_{\text{aff}},rew_{\text{aff}},\text{lra}},$$

accordingly for accumulated rewards and minimal values.

The theorem follows by Lemma 2.

EXAMPLE 12. *If using an affine abstraction, we replace the actions of Figure 5 by*

$$(c_{\text{IH}},\text{up}){:}(0,0),(c_{\text{IH}},\text{lo}){:}(0,0),$$
$$(c_{\text{CoH}},\text{up}){:}(0,2),(c_{\text{CoH}},\text{lo}){:}(0,0.5),$$
$$(c_{\text{ECo}},\text{up}){:}(0.5,0.5),(c_{\text{ECo}},\text{lo}){:}(0.5,0.5),$$
$$(c_{\text{HCo}},\text{up}){:}(0,3),(c_{\text{HCo}},\text{lo}){:}(0,0),$$
$$(c_{\text{Ch}},\text{up}){:}(0,0.5),(c_{\text{Ch}},\text{lo}){:}(0,0.25).$$

4.1 Computing Abstractions

A practical recipe to compute abstractions of PHAs has been developed in earlier work [39], and is implemented in the tool PROHVER. The abstraction process is piggybacked on solvers for HAs (as in Definition 2). Concretely, the tool applies PHAVER for this purpose. Thus far however there was no need and no means to compute reward structures for the abstraction at hand.

According to Definition 22 and Definition 23, we have to find suprema and infima of the variables for rewards and time. How this can be done depends on the hybrid systems solver used. PHAVER 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.

In some cases, this construction can be simplified. If we do not have time-dependent rewards and only a constant reward value for each command, and only want to consider the expected accumulated reward, we do not need to compute infima or suprema at all. 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 23 the values r are not used.

In usual abstractions of HAs, the information about the time which a continuous transition takes is lost. This is the main reason why we encode reward and time into the first two dimension of a PHA, rather than assigning them directly to the timed transitions.

4.2 Algorithmic Considerations

After we have obtained a finite abstraction and have computed the according reward structure using linear programming, it remains to compute expected accumulated or long-run average rewards in the abstraction. There are several algorithms which we can apply for this purpose. For accumulated rewards we can use algorithms based on policy iteration or linear programming [30]. In case we want to compute time-average average values, there are also algorithms using linear programming [15] or policy iteration [38].

len.	PHAVER	S	uptime			commands		
			constr.	ana.	res.	constr.	ana.	res.
–	0	7	0	0	0.00	0	0	42.00
1	1	102	0	0	45.33	0	0	42.00
0.05	151	24593	6	65	62.02	1	103	48.84
0.03	2566	93979	25	584	62.72	2	722	50.78

Table 2: Accumulated rewards in thermostat.

len.	PHAVER	S	time in error			time in error (lin)		
			constr.	ana.	result	constr.	ana.	result
1	0	126	0	0	0.013	0	0	0.013
0.5	1	382	0	0	0.011	1	0	0.011
0.1	28	7072	4	9	0.009	6	14	0.009
0.05	185	29367	8	124	0.009	25	313	0.009

Table 3: Long-run average rewards in thermostat.

5. EXPERIMENTS

We implemented the analysis methods to compute expected accumulated and time-average rewards in our tool PROHVER, and have applied them on two case studies. Experiments were run on an Intel(R) Core(TM)2 Duo CPU with 2.67 GHz and 4 GB RAM.

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 from Init to Heat) it takes until the error mode is reached, as in Example 3.

Results are given in Table 2. The *constraint length* (len.) is a parameter which influences the precision with which PHAVER builds the abstract state space. It splits the abstract states along a given variable, in our case T. We also provide the times in seconds which PHAVER needed to build the abstraction (PHAVER), the number of states (S), the time we needed to compute the reward structures (constr.) and the computed value bound (result).

Initially, we performed these experiments splitting on 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

$$Rew(c_{\text{HCo}}) + Rew(c_{\text{CoH}}) + (Rew(c_{\text{HCh}}) + Rew(c_{\text{ChH}}))\frac{1}{0.05} = 1 + 1 + (1+1)\frac{1}{0.05} = 42.$$

Table 2 shows that this is not the ultimate answer to the minimal expected number of executed commands. This implies that 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.

Next, we consider the maximal expected time fraction f spent in the Error mode of the modified thermostat variant in which this mode can be left. This allows us to obtain a lower bound for the system long-run availability $1-f$. 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 12. Results are given in Table 3. As seen, using affine reward structures

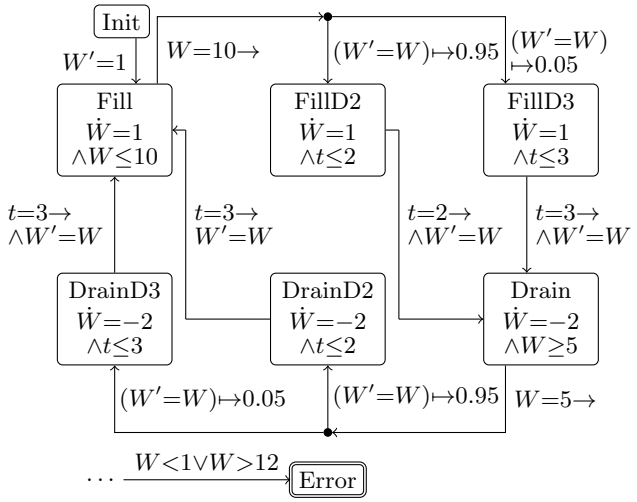

Figure 6: Water level control automaton.

len.	PHAVER	S	uptime			commands		
			constr.	ana.	result	constr.	ana.	result
–	1	10	0	0	0.0	0	0	40
1	0	53	0	0	137.0	0	0	40
0.05	5	814	1	1	164.8	0	2	40
0.01	31	4294	1	42	166.1	0	46	40

Table 4: Accumulated rewards in water control.

does not improve the result: This was expected by comparing Example 11 and Example 12. Some of the actions of Example 12 have the same values as actions in Example 11 (e.g. $(c_{\text{CoH}}, \text{lo}) : (0, 0.5)$ corresponds to $c_{\text{CoH}} : (0, 0.5)$), while the remaining ones are suboptimal choices for a maximising scheduler (e.g. $(c_{\text{CoH}}, , \text{up}) : (0, 2)$). 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 the time needed by PROHVER is larger.

5.2 Water Level Control

We consider a model of a water level control system (extended from the one of Alur et al. [3]) 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 6. The water level W of a tank is controlled by a monitor. Its change is specified by an affine 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 ($\dot{W}$). 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). Similar to the thermostat case, 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 t. Results are given in Table 4.

len.	PHAVER	S	avg. energy			avg. energy (lin.)		
			constr.	ana.	result	constr.	ana.	result
1	0	51	0	0	1.985	0	0	1.814
0.1	2	409	0	0	1.656	0	0	1.640
0.02	11	2149	0	0	1.630	1	0	1.627
0.01	24	4292	0	1	1.627	2	1	1.625

Table 5: Long-run average rewards in water control.

For the second property, we remove Error 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 5. As seen, using affine reward structures improves the result more than in the thermostat case. This happens 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 that mode in the thermostat setting is small, the induced difference between the lower and upper values with the two kinds of reward abstractions is also rather small.

6. CONCLUSION

We have presented a framework to handle probabilistic hybrid automata decorated with costs or bonuses. The resulting abstraction and verification approach considerably enriches the spectrum of properties that are amenable to model checking. It is now possible to check properties including cost accumulated until termination, the long-run cost of system operation, and other popular quantities. This includes long-run availability, as mentioned in Section 5, and many other quantities of interest. For instance, we are now in the position to compute the long-run survivability [14] of a probabilistic hybrid system. This quantity asks for a time bounded reach analysis [40] nested inside a long-run cost analysis.

To arrive at this powerful framework, we have decorated PHAs with reward structures associated to taking a transition, have discussed how these structures carry over to the semantical model (PAs) and how properties of PHAs are defined. These properties can reason about the expected total or long-run average reward.

On the level of the semantical model, we have extended our abstraction framework to work well with reward structures, and this required some nontrivial considerations. Probabilistic simulation relations serve as means to guarantee the correctness of the abstractions we build, and they needed to be augmented to properly cover reward-based properties.

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 has turned out necessary to take a closer look at the form of the abstract states, to find out about minimal or maximal reward values. The effectivity of our approach has been demonstrated by using it to compute reward-based values on two case studies.

Acknowledgements. This work was supported by the Transregional Collaborative Research Centre SFB/TR 14 AVACS, the NWO-DFG bilateral project ROCKS, the European Uni-

on Seventh Framework Programme under grant agreement numbers 295261 (MEALS), and 318490 (SENSATION), and the ERC Advanced Grant VERIWARE.

7. REFERENCES

[1] A. Abate, M. Prandini, J. Lygeros, and S. Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[2] E. Altman and V. Gaitsgory. Asymptotic optimization of a nonlinear hybrid system governed by a Markov decision process. *SICON*, 35(6):2070–2085, 1997.

[3] 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. *TCS*, 138(1):3–34, 1995.

[4] R. Alur, T. Dang, and F. Ivančić. Predicate abstraction for reachability analysis of hybrid systems. *ACM TECS*, 5(1):152–199, 2006.

[5] E. Asarin, O. Bournez, T. Dang, O. Maler, and A. Pnueli. Effective synthesis of switching controllers for linear systems. *Proc. IEEE*, 88(7):1011–1025, 2000.

[6] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems*, pages 1–20, 1994.

[7] C. Baier, L. Cloth, B. R. Haverkort, H. Hermanns, and J.-P. Katoen. Performability assessment by model checking of Markov reward models. *Formal Methods in System Design*, 36:1–36, 2010.

[8] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[9] P. Bouyer, U. Fahrenberg, K. G. Larsen, and N. Markey. Quantitative analysis of real-time systems using priced timed automata. *Commun. ACM*, 54(9):78–87, 2011.

[10] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. The cost of punctuality. In *LICS*, pages 109–120, 2007.

[11] M. L. Bujorianu. Extended stochastic hybrid systems and their reachability problem. In *HSCC*, pages 234–249, 2004.

[12] M. L. Bujorianu, J. Lygeros, and M. C. Bujorianu. Bisimulation for general stochastic hybrid systems. In *HSCC*, pages 198–214, 2005.

[13] C. Cassandras and J. Lygeros. *Stochastic Hybrid Systems*, volume 24. CRC Press and IEEE Press, 2006.

[14] L. Cloth and B. R. Haverkort. Model checking for survivability. In *QEST*, pages 145–154, 2005.

[15] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[16] E. de Souza e Silva and H. R. Gail. Calculating availability and performability measures of repairable computer systems using randomization. *J. ACM*, 36(1):171–193, 1989.

[17] M. Fränzle, E. M. Hahn, H. Hermanns, N. Wolovick, and L. Zhang. Measurability and safety verification for stochastic hybrid systems. In *HSCC*, pages 43–52, 2011.

[18] M. Fränzle, T. Teige, and A. Eggers. Satisfaction meets expectations - computing expected values of probabilistic hybrid systems with SMT. In *IFM*, pages 168–182, 2010.

[19] G. Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. *STTT*, 10(3):263–279, 2008.

[20] 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*, pages 379–395, 2011.

[21] E. M. Hahn. *Model Checking Stochastic Hybrid Systems*. PhD thesis, Saarland University, 2013. To appear.

[22] E. M. Hahn, G. Norman, D. Parker, B. Wachter, and L. Zhang. Game-based abstraction and controller synthesis for probabilistic hybrid systems. In *QEST*, pages 69–78, 2011.

[23] A. Hartmanns, H. Hermanns, and P. Berrang. A comparative analysis of decentralized power grid stabilization strategies. In *WSC*, 2012.

[24] J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In *HSCC*, pages 160–173, 2000.

[25] B. Jonsson and K. G. Larsen. Specification and refinement of probabilistic processes. In *LICS*, pages 266–277, 1991.

[26] R. M. Keller. Formal verification of parallel programs. *Commun. ACM*, 19(7):371–384, 1976.

[27] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.

[28] A. Platzer. Stochastic differential dynamic logic for stochastic hybrid programs. In *CADE*, pages 446–460, 2011.

[29] M. Prandini and J. Hu. A stochastic approximation method for reachability computations. In H. Blom and J. Lygeros, editors, *Stochastic Hybrid Systems*, volume 337 of *LNCIS*, pages 107–139. Springer, 2006.

[30] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[31] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM TECS*, 6(1), 2007.

[32] M. Rutkowski, R. Lazić, and M. Jurdziński. Average-price-per-reward games on hybrid automata with strong resets. *STTT*, 13(6):553–569, 2011.

[33] R. Segala. *Modeling and Verification of Randomized Distributed Real-time Systems*. PhD thesis, MIT, 1995.

[34] R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *NJC*, 2(2):250–273, 1995.

[35] J. Sproston. Decidable model checking of probabilistic hybrid automata. In *FTRTFT*, pages 31–45, 2000.

[36] T. Teige. *Stochastic Satisfiability Modulo Theories: A Symbolic Technique for the Analysis of Probabilistic Hybrid Systems*. PhD thesis, Carl von Ossietzky Universität Oldenburg, 2012.

[37] I. Tkachev and A. Abate. Regularization of Bellman equations for infinite-horizon probabilistic properties. In *HSCC*, pages 227–236, 2012.

[38] C. von Essen and B. Jobstmann. Synthesizing efficient controllers. In *VMCAI*, pages 428–444, 2012.

[39] L. Zhang, Z. She, S. Ratschan, H. Hermanns, and E. M. Hahn. Safety verification for probabilistic hybrid systems. In *CAV*, pages 196–211, 2010.

[40] L. Zhang, Z. She, S. Ratschan, H. Hermanns, and E. M. Hahn. Safety verification for probabilistic hybrid systems. *European Journal of Control*, 18:572–587, 2012.

[41] P. Zuliani, C. Baier, and E. M. Clarke. Rare-event verification for stochastic hybrid systems. In *HSCC*, pages 217–226, 2012.