IEICE TRANS. FUNDAMENTALS, VOL.E88–A, NO.11 NOVEMBER 2005

PAPER Special Section on Concurrent/Hybrid Systems: Theory and Applications

Symbolic Reachability Analysis of Probabilistic Linear Hybrid Automata

Yosuke MUTSUDA[†a)], Takaaki KATO[†b)], *Nonmembers, and* Satoshi YAMANE[†c)], *Member*

SUMMARY We can model embedded systems as hybrid systems. Moreover, they are distributed and real-time systems. Therefore, it is important to specify and verify randomness and soft real-time properties. For the purpose of system verification, we formally define probabilistic linear hybrid automaton and its symbolic reachability analysis method. It can describe uncertainties and soft real-time characteristics.

key words: *verification, performance evaluation, formal specification, probabilistic hybrid automata, reachability, symbolic methods*

1. Introduction

As ubiquitous computing has progressed, systems are embedded in widespread environments. Then it is important to guarantee their formal correctness, for instance, safety, reliability, dependability, randomization, and soft real-time properties.

In this paper, we propose the formal verification of probabilistic hybrid systems. Probabilistic hybrid systems are digital real-time systems that embedded in analog environment and exhibit probabilistic characteristics.

There have been several formal verification methods based on automaton models as follows:

1. Symbolic model-checking procedure and its implementation HyTech for linear hybrid automata have been developed using manipulating and simplifying $(\mathbb{R}, \leq, +)$-formulae [1].
2. For probabilistic timed automata [5], zone-based symbolic model checking algorithms and tool Prism have been presented [4].
3. Reachability for probabilistic rectangular automata has been mentioned in [6], but the verification methods for general class of probabilistic hybrid automata have not been developed.

We consider probabilistic linear hybrid automata, an extension of linear hybrid automata [1] with discrete probability distributions or probabilistic timed automata [4] with continuous dynamics. This model contains probabilistic rectangular automata [6], moreover, our reachability analysis method differs from [6] on the point that J. Sproston [6] generates a finite-state reachability graph, but our approach uses symbolic computation of logical formulae without graph construction.

To verify probabilistic hybrid systems, we define the polyhedron labeled by probability as the data structure. And we collectively compute state transitions by the symbolic operations.

Probabilistic linear hybrid automata can model uncertain behaviors such as statistical estimates regarding the environment in which a system is embedded. And its verification and performance evaluation allow for soft real-time quantitative properties.

This paper is organized as follows: In Sect. 2, we define probabilistic linear hybrid automata and some preliminary concepts and notations. Section 3 defines the reachability problem of probabilistic linear hybrid automata. The symbolic reachability analysis method and trial examples are presented in Sect. 4, and case study of industrial application is Sect. 5 using prototype tool. Finally, in Sect. 6, we conclude this paper.

2. Probabilistic Linear Hybrid Automata

Probabilistic linear hybrid automata are defined in this section as our model for probabilistic-nondeterministic real-time and hybrid systems. This system description language is an extended linear hybrid automaton [1] by discrete probability distributions.

2.1 Preliminaries

In preparation, we define basic concepts as follows:

Linear constraints Let $\vec{u}$ be a vector of real-valued variables. A *linear term* over $\vec{u}$ is a linear combination of variables from $\vec{u}$ with integer coefficients. A *linear inequality* over $\vec{u}$ is an inequality between linear terms over $\vec{u}$. A *convex linear formula* over $\vec{u}$ is a finite conjunction of linear inequalities over $\vec{u}$. A *linear formula* over $\vec{u}$ is a finite boolean combination of linear inequalities over $\vec{u}$. Let $\mathtt{clf}(\vec{u})$ and $\mathtt{lf}(\vec{u})$ be the set of convex linear formulae over $\vec{u}$ and the set of linear formulae over $\vec{u}$, respectively.

Distributions A discrete probability *distribution* over a finite set Q is a function $p : Q \to [0, 1]$ such that $\sum_{q \in Q} p(q) = 1$. Let $\mathtt{support}(p)$ be the subset of Q such that

Manuscript received March 18, 2005.
Manuscript revised June 1, 2005.
Final manuscript received July 11, 2005.

[†]The authors are with the Graduate School of Natural Science & Technology, Kanazawa University, Kanazawa-shi, 920–1192 Japan.
a) E-mail: 0163muts@is.t.kanazawa-u.ac.jp
b) E-mail: 1118kato@is.t.kanazawa-u.ac.jp
c) E-mail: syamane@is.t.kanazawa-u.ac.jp

DOI: 10.1093/ietfec/e88–a.11.2972

$\texttt{support}(p) = \{q \mid p(q) > 0\}$. For a possibly uncountable set Q', let $\texttt{Dist}(Q')$ be the set of distributions over finite subsets of Q'.

2.2 Syntax

First, we define the syntax of probabilistic linear hybrid automata.

Definition 1: A *probabilistic linear hybrid automaton* $\texttt{PLHA} = \langle \vec{x}, L, init, inv, dif, prob, (grd_l)_{l\in L}, E\rangle$ consists of the following components:

Data variables Let $\vec{x}$ be the finite vector $(x_1, x_2, \ldots, x_n)$ called real-valued *data variables*.

A point $\vec{s} = (s_1, \ldots, s_n) \in \mathbb{R}^n$ is referred to as a *data state*, or, equivalently, a *valuation* of data variables. The convex linear formula $f \in \texttt{clf}(\vec{x})$ defines the convex polyhedron $[\![f]\!] \subseteq \mathbb{R}^n$, where $[\![f]\!] = \{\vec{s} \mid f[\vec{x} := \vec{s}]$ *is true.*$\}$. A *polyhedron* is a finite union of convex polyhedra.

For each data variable x_i, we use the dotted variable $\dot{x}_i$ to denote the *first derivative* of x_i. For data variables $\vec{x}$, we use the primed variable $\vec{x}'$ to denote the *new value* of $\vec{x}$ after a transition. Let *updated variables* X be the subset of $\vec{x}$, similarly, $X' \subseteq \vec{x}'$.

Control locations L is a finite set of *control locations*.

A *state* $(l, \vec{s})$ of the automaton PLHA consists of a control location $l \in L$ and a valuation $\vec{s} \in \mathbb{R}^n$. S_l is the set of data states at location l. A *region* $R = \bigcup_{l\in L}\{(l, S_l)\}$ is a collection of polyhedron $S_l \subseteq \mathbb{R}^n$ with respect to each control location $l \in L$. A *predicate* $\pi = \bigcup_{l\in L}\{(l, f_l)\}$ is a collection of linear formula $f_l \in \texttt{clf}(\vec{x})$. The predicate π defines the region $[\![\pi]\!] = \bigcup_{l\in L}\{(l, [\![f_l]\!])\}$.

Initial state $init = (l_0, \vec{s}_0)$ is an *initial state* of the probabilistic linear hybrid automaton, where $l_0 \in L$ is an initial node, a single point $\vec{s}_0 \in \mathbb{R}^n$ is an initial value.

Locations invariants The function $inv : L \to \texttt{clf}(\vec{x})$ assigns *invariant* condition to each location. The control of the automaton PLHA may reside in the location l only as long as the invariant $inv(l)$ is true ($\vec{s} \in [\![inv(l)]\!]$).

Continuous flows $dif : L \to \texttt{clf}(\vec{x})$ is a labeling function assigning *flows* to locations. The flows constrain the rates at which the values of data variables change: while the automaton control resides in the location l, the values of first derivatives of all data variables stay within the *differential inclusion* $\dot{\vec{s}} \in [\![dif(l)]\!]$.

The probabilistic linear hybrid automaton PLHA is *time-nondeterministic* if there exists a location $l \in L$ such that $[\![dif(l)]\!]$ is not a single point.

Discrete probability distributions The function $prob : L \to 2^{\texttt{Dist}(2^{\vec{x}}\times\texttt{clf}(\vec{x}\uplus X')\times L)}_{fn}$ assigning to each location a finite, non-empty set of *discrete probability distributions* $prob(l) = \{p_l^1, \ldots, p_l^{|prob(l)|}\} \subseteq \texttt{Dist}(2^{\vec{x}} \times \texttt{clf}(\vec{x} \uplus X') \times L)$.

Enabling conditions The family of functions $(grd_l)_{l\in L}$, where for any $l \in L$, $grd_l : prob(l) \to \texttt{clf}(\vec{x})$ assigns an *enabling condition* (or *guard*) to each $p_l^i \in prob(l)$ at $l \in L$. It may happen that the intersection of multiple guards is not empty. In such a case, nondeterminism on selecting probability distributions arises, i.e. there are a number of possibilities. We solve this by the adversary. We will explain the concept of the adversary in Sect. 2.3.1.

Probabilistic edges For each $l \in L, p_l^i = p \in prob(l), grd_l(p) = g$, we define the *probabilistic edges* $e = (l, g, p, X, updt, l')$ by discrete probability distributions, where $X \subseteq \vec{x}, updt \in \texttt{clf}(\vec{x} \uplus X'), l' \in L$. Let E be the finite set of probabilistic edges such that $E = \{e \mid p(X, updt, l') > 0\}$.

An *update* is a convex linear formula $updt$ over the set $\vec{x}\uplus X'$. The *action* of the update $updt$ is the convex linear formula over the set $\vec{x} \uplus \vec{x}'$, $act = updt \wedge (\bigwedge_{x_i\in\vec{x}\setminus X}(x_i' = x_i))$, all data variables that are not updated remain unchanged. In other words, the update $updt$ defines a function $act : \mathbb{R}^n \to \texttt{clf}(\vec{x}')$ from valuations to convex linear formulae over $\vec{x}'$. For all valuations $\vec{s}, \vec{s}' \in \mathbb{R}^n$, let $\vec{s}' \in [\![act(\vec{s})]\!]$ *iff* $act[\vec{x}, \vec{x}' := \vec{s}, \vec{s}']$ *is true.*

2.2.1 Examples

We will show some simple example as follows:

We consider probabilistic linear hybrid automaton as shown in Fig. 1 and its formal description is below. Guard is assigned to the distribution, and both update formula and the probability are assigned to the edge.

$$\texttt{PLHA} = \langle \vec{x}, L, init, inv, dif, prob, (grd_l)_{l\in L}, E\rangle$$

- $\vec{x} = \{x, y\}$,
- $L = \{l_1, l_2\}$,
- $inv(l_1) = (1 \le y \le 2)$, $inv(l_2) = (y \ge 0)$,
- $dif(l_1) = (1 \le \dot{x} \le 2 \wedge 1 \le \dot{y} \le 2)$, $dif(l_2) = (\dot{x} = 1 \wedge \dot{y} = 2)$,
- $prob(l_1) = \{p_{l_1}\}$, $prob(l_2) = \{p_{l_2}\}$,
- $grd_l(p_{l_1}) = (x \le 3)$,
- $p_{l_1}(\{x, y\}, x' \ge 1 \wedge y' = x, l_2) = 0.6$, $p_{l_1}(\emptyset, \text{-}, l_1) = 0.4$,
  $p_{l_2}(\emptyset, \text{-}, l_2) = 1$.

In Fig. 1, the initial state $init$ is $(l_1, x = 0 \wedge y = 1)$. First, time passes in location 1 or the location changes. If time passes, the values of data variables change at the rate $(1 \le \dot{x} \le 2 \wedge 1 \le \dot{y} \le 2)$. Location might change if the guard $(x \le 3)$ of the distribution is satisfied. If the location changes, the variables are updated according to the action formula of the edge. For example, the transition to location

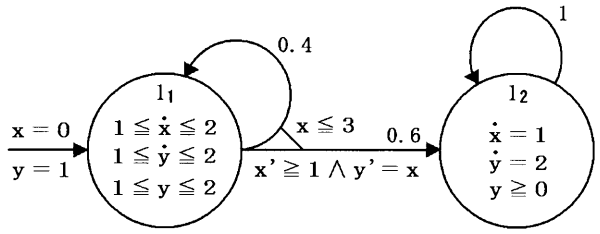


Fig. 1 Probabilistic linear hybrid automaton.

2 with the probability 0.6 updates the values of data variables according to $(x' \geq 1 \wedge y' = x)$.

We verify whether the automaton reaches the target from the initial state or not by tracing transitions. For example, if the target is $(l_2, 1 \leq x \leq 2 \wedge 2 \leq y \leq 3)$, it is possible to reach the target from the initial state $init = (l_1, x = 0 \wedge y = 1)$ with probability 0.6 as follows:

$(l_1,\ x = 0 \wedge y = 1)$
passage of one time unit under $(\dot{x} = 2 \wedge \dot{y} = 2)$
$\rightarrow (l_1,\ x = 2 \wedge y = 2)$
transition to l_2 *under* $(x' = 2 \wedge y' = 2)$
with probability 0.6
$\rightarrow (l_2,\ x = 2 \wedge y = 2)$

We can specify probabilistic hybrid systems, which are reactive systems that intermix discrete and continues components with randomization, using probabilistic linear hybrid automata. Typical examples are digital controllers that interact with continuously changing physical environments, the steam boiler shown in Sect. 5 is the one. Because probabilistic linear hybrid automata can describe the probability, statistical information such as the reliability of the switch can be described, too. Moreover, reset of the values can be described by update formula.

2.3 Semantics

2.3.1 Concurrent Probabilistic Systems

Next, we define concurrent probabilistic systems as a semantic model.

Definition 2: A *concurrent probabilistic system* is a tuple $\mathtt{CPS} = \langle Q, \Sigma, Steps \rangle$ where:

- Q is a set of *states;*
- Σ is a set of *events;*
- $Steps : Q \rightarrow 2^{\Sigma \times \mathtt{Dist}(Q)}$ is a function which assigns to each state a non-empty set $Steps(q)$ of pairs $(\sigma, \mu) \in \Sigma \times \mathtt{Dist}(Q)$ comprising an event and a distribution on Q.

A *probabilistic transition* $q \xrightarrow{\sigma,\mu} q'$ is made from a state q by nondeterministically selecting an event-distribution pair $(\sigma, \mu) \in Steps(q)$, and then making a probabilistic choice of target state q' according to μ, such that $\mu(q') > 0$.

An execution of a concurrent probabilistic system is represented by a *path* ω, that is, a non-empty sequence of transitions $\omega = q_0 \xrightarrow{\sigma_0,\mu_0} q_1 \xrightarrow{\sigma_1,\mu_1} q_2 \xrightarrow{\sigma_2,\mu_2} \cdots$. We denote by $\omega(i)$ the ith state of a path ω, $step(\omega, i)$ the ith transition of ω, $|\omega|$ the length of ω and if ω is finite, the last state by $last(\omega)$. We say that a finite path $\omega^{(k)}$ of length k $(\leq |\omega|)$ is a *prefix* of ω if $\omega^{(k)}(i) = \omega(i)$ for all $0 \leq i \leq k$, and $step(\omega^{(k)}, i) = step(\omega, i)$ for all $0 \leq i \leq k - 1$. $Path_{fin}$ is the set of all finite paths.

Event-distribution pair $(\sigma, \mu) \in Steps(q)$, is nondeterministically chosen. According to the general technique [9]–[11], we represent concurrency by the nondeterministic choice.

We now introduce adversaries of concurrent probabilistic system as functions which resolve all of the nondeterministic choices of the model.

Definition 3: (Adversaries) A deterministic *adversary* (or scheduler) of concurrent probabilistic system is a function $\mathcal{A} : Path_{fin} \rightarrow \Sigma \times \mathtt{Dist}(Q)$ which assigns to each finite path $\omega \in Path_{fin}$ an event-distribution pair (σ, μ) *deterministically* such that $\mathcal{A}(\omega) \in Steps(last(\omega))$.

For an adversary $\mathcal{A}$, we define $Path_{fin}^{\mathcal{A}}$ to be the set of finite paths such that $step(\omega, k) = \mathcal{A}(\omega^{(k)})$ for all $0 \leq k < |\omega|$. $Path_{fin}^{\mathcal{A}}$ means the one (deterministic) computation tree labeled by probabilities. Let Adv be the set of adversaries.

An adversary decides the nondeterministically selecting performed event-distribution pairs in concurrent probabilistic system. Therefore, given an adversary, the nondeterministic model under the adversary can be described by a deterministic model.

With each adversary, we associate a sequential markov chain, which can be regarded as a set of paths in concurrent probabilistic system. Formally, if $\mathcal{A}$ is an adversary, then $\mathtt{MC}^{\mathcal{A}}$ is a markov chain.

Definition 4: (Markov Chains) An infinite-state *Markov chain* which corresponds to $\mathcal{A}$ is $\mathtt{MC}^{\mathcal{A}} = \langle Path_{fin}^{\mathcal{A}}, \mathbf{P}^{\mathcal{A}} \rangle$, where:

- A set of states of $\mathtt{MC}^{\mathcal{A}}$ is $Path_{fin}^{\mathcal{A}}$.
- $\mathbf{P}^{\mathcal{A}} : Path_{fin}^{\mathcal{A}} \times Path_{fin}^{\mathcal{A}} \rightarrow [0, 1]$ is a *transition probability matrix*, such that:

$$\mathbf{P}^{\mathcal{A}}(\omega, \omega') = \begin{cases} \mu(q) & \text{if } \mathcal{A}(\omega) = (\sigma, \mu) \text{ and } \omega' = \omega \xrightarrow{\sigma,\mu} q \\ 0 & \text{otherwise.} \end{cases}$$

Definition 5: (Probabilities over Paths) Let $\mathcal{P}^{\mathcal{A}}$ be the mapping inductively defined on the length of paths in $Path_{fin}^{\mathcal{A}}$ as follows. *If* $|\omega| = 0$, *then* $\mathcal{P}^{\mathcal{A}}(\omega') = 1$. *If* $|\omega| > 0$ *and* $\omega' = \omega \xrightarrow{\sigma,\mu} q$ *for some* $\omega \in Path_{fin}^{\mathcal{A}}$, *then we let*:

$$\mathcal{P}^{\mathcal{A}}(\omega') = \mathcal{P}^{\mathcal{A}}(\omega) \cdot \mathbf{P}^{\mathcal{A}}(\omega, \omega').$$

2.3.2 Semantics of Probabilistic Linear Hybrid Automata

The following notation ([6]) is used to reason about the next states of the probabilistic edges of PLHA. For the distribution p, if $\mathtt{support}(p) = \{(X^1, updt^1, l'^1), (X^2, updt^2, l'^2), \ldots, (X^m, updt^m, l'^m)\}$, then we let the tuple of actions $\mathtt{extract}(p) = (act^1, act^2, \ldots, act^m)$ and generate the tuple of valuations $\langle v \rangle = (\vec{v}^1, \vec{v}^2, \ldots, \vec{v}^m)$ in the following way: for each $1 \leq j \leq m$, we choose a valuation $\vec{v}^j \in \mathbb{R}^n$ such that $\vec{v}^j \in [\![act(\vec{s})^j]\!]$. Observe that, for any $1 \leq i, j \leq m$ such that $i \neq j$, it may be the case that $[\![act(\vec{s})^i]\!]$ and $[\![act(\vec{s})^j]\!]$ have non-empty intersection, and therefore it is possible that $\vec{v}^i = \vec{v}^j$, where $\vec{v}^i \in [\![act(\vec{s})^i]\!]$ and $\vec{v}^j \in [\![act(\vec{s})^j]\!]$. Let $\mathtt{Combinations}(\vec{s}, \mathtt{extract}(p))$ be the set of all such tuples $\langle v \rangle$ for a given state $(l, \vec{s})$ and the distribution p.

Definition 6: The *concurrent probabilistic system* $\mathrm{CPS}_{\mathrm{PLHA}} = \langle Q_{\mathrm{PLHA}}, \Sigma_{\mathrm{PLHA}}, Steps_{\mathrm{PLHA}} \rangle$ *of probabilistic linear hybrid automaton* PLHA is defined as following infinite-state transition system:

- $Q_{\mathrm{PLHA}} \subseteq L \times \mathbb{R}^n$ is the set of states, defined such that $(l, \vec{s}) \in Q_{\mathrm{PLHA}}$ *if* $\vec{s} \in [\![inv(l)]\!]$;
- $\Sigma_{\mathrm{PLHA}} = \mathbb{R}_{\geq 0}$ is the set of events. $\mathrm{CPS}_{\mathrm{PLHA}}$ is not event-driven system using the alphabet but a time-driven system that uses time as a trigger. Therefore, time becomes an event;
- For each state $(l, \vec{s}) \in Q_{\mathrm{PLHA}}$, let $Steps_{\mathrm{PLHA}}((l, \vec{s})) = Cont(l, \vec{s}) \cup Disc(l, \vec{s})$ be the smallest set of event-distribution pairs such that:
  - **Time transition** for each duration $\delta \in \mathbb{R}_{\geq 0}$, there exists $(\delta, \mu_{(l,\vec{s}')}) \in Cont(l, \vec{s})$ such that $\mu_{(l,\vec{s}')}(l, \vec{s}') = 1$ if and only if either
    1. $\delta = 0$ *and* $\vec{s}' = \vec{s}$, *or*
    2. $\delta > 0$ *and* $\frac{\vec{s}' - \vec{s}}{\delta} \in [\![dif(l)]\!]$;
  - **Discrete transition**
    $Disc(l, s) = \bigcup_{p \in prob(l)} Disc(l, \vec{s}, p)$, where for each distribution $p \in prob(l)$, if $\vec{s} \in [\![grd_l(p)]\!]$, then, for each $\langle v \rangle \in \mathtt{Combinations}(\vec{s}, \mathtt{extract}(p))$, there exists the pair $(0, \mu_{p,\langle v \rangle}) \in Disc(l, \vec{s}, p)$ such that

$$\mu_{p,\langle v \rangle}(l', \vec{s}') = \sum_{\substack{i \in \{1,\ldots,m=|\mathtt{support}(p)|\} \\ \& l' = l'^j \& \vec{s}' = \vec{v}^j}} p(X^j, updt^j, l'^j). \quad (1)$$

Expression (1) resolves the case of probabilities summation as the same way [4], [6].

An adversary chooses the event-distribution pair $(\sigma, \mu) \in Steps_{\mathrm{PLHA}}((l, \vec{s})) = Cont(l, \vec{s}) \cup Disc(l, \vec{s})$ that can be performed in $\mathrm{CPS}_{\mathrm{PLHA}}$. In other words, it chooses one from various possibilities as follows. At any time, if the system is in a location, then the system can either remain in its current location and let time advance, or make a discrete transition if there exists a distribution. Discrete transitions are instantaneous and consist of the two steps performed in succession: firstly, the system makes a nondeterministic choice between the set of distributions. Secondly, supposing that the distribution is chosen, the system then makes a probabilistic transition according to the distribution. In this nondeterministic choice between the set of event-distribution pairs, if we define an adversary, the nondeterministic model under the adversary can be described by a deterministic model.

3. Reachability Problem

We now formally define our reachability problem.

Definition 7: (Probabilistic Reachability Problem) Given a probabilistic linear hybrid automaton $\mathtt{PLHA} = \langle \vec{x}, L, init, inv, dif, prob, (grd_l)_{l \in L}, E \rangle$, let T be a predicate called the *target*, let $\sqsupseteq \in \{\geq, >\}$, and let $\lambda \in [0, 1]$ be the *target probability*. Then *probabilistic reachability problem* for PLHA can be defined as the tuple $(\mathrm{T}, \sqsupseteq, \lambda)$, the answer to this problem is "YES, reachable" if and only if there exists an adversary $\mathcal{A} \in Adv$ of $\mathrm{CPS}_{\mathrm{PLHA}}$ (or, equivalently, a series of nondeterministic choices) and a path $\omega \in Path_{fin}^{\mathcal{A}}$ starting in an initial state of PLHA $init = (l_0, \vec{s}_0)$ such that $last(\omega)$ in $(l, [\![f_l]\!]) \in [\![\mathrm{T}]\!]$ with probability (over path) $\sqsupseteq \lambda$, and "No" otherwise.

We now review two subclasses of reachability properties: *time bounded reachability* and *invariance* which are particularly relevant for the verification and the performance evaluation of probabilistic real-time and hybrid systems [5], [6].

About the former, PLHA has certain time deadlines. On the other hand, about the latter, PLHA is required that does *not leave* an *invariant* region $[\![\mathcal{I}]\!] \subseteq Q_{\mathrm{PLHA}}$, or, equivalently, *always satisfies* some properties. (e.g. $[\![\mathcal{I}]\!]$ as desirable or expected region for safety property).

4. Verification: Symbolic Reachability Analysis

The following extended expression is used to express the probabilities of the transitions of $\mathrm{CPS}_{\mathrm{PLHA}}$.

Definition 8: (Polyhedra labeled by Probabilities) For a linear formula $f \in \mathtt{lf}(\vec{x})$ and a corresponding polyhedron $[\![f]\!]$, we define the *probabilistic polyhedron* $([\![f]\!], P)$ to be the pair comprising a polyhedron $[\![f]\!] \subseteq \mathbb{R}^n$ and its probability $P \in (0, 1]$.

The function $\mathtt{plf} : L \to 2^{\mathtt{lf}(\vec{x}) \times (0,1]}$ assigning to each location a set of *probabilistic linear formulae*, where a probabilistic linear formula is the pair of a linear formula f and its probability P such as $(f, P) \in \mathtt{lf}(\vec{x}) \times (0, 1]$. Let $[\![\mathtt{plf}(l)]\!] \subseteq 2^{\mathbb{R}^n} \times (0, 1]$ be the finite set of probabilistic polyhedra in the location l such that $[\![\mathtt{plf}(l)]\!] = \{([\![f_l]\!], P) \mid \textit{for some } [\![f_l]\!], P > 0\}$. Note that, for any $([\![f_l{}^a]\!], P^a), ([\![f_l{}^b]\!], P^b) \in [\![\mathtt{plf}(l)]\!]$ such that $[\![f_l{}^a]\!] = [\![f_l{}^b]\!]$, it is the case that $P^a \neq P^b$.

In the sequel, we use $R = \bigcup_{l \in L} \{(l, [\![\mathtt{plf}(l)]\!])\}$ as a region, and a predicate π corresponding to the region R is $\pi = \bigcup_{l \in L} \{(l, \mathtt{plf}(l))\}$, where $[\![\pi]\!] = \bigcup_{l \in L} \{(l, [\![\mathtt{plf}(l)]\!])\}$.

We introduce extra edge relations to deal with summing up probabilities with respect to the same next state (cf. Sect. 2.3.2 and expression (1)).

Definition 9: (Extra edge relations) For a set E, let $\mathcal{E}$ be the set of *extra probabilistic edges* such that:

$$\mathcal{E} = \bigcup_{\substack{l\in L, p\in prob(l),\\ Act\in 2_{ne}^{\mathtt{extract}(p)}}} \{e = (l, g, pr, act, l') \mid \mathtt{condition}\};$$

$$\mathtt{condition} \equiv \forall act^j \in Act \text{ such that } |Act| \geq 2,$$
$$l' = l'^j \text{ and } (\bigcap_{act^j\in Act} [\![act(\vec{s})^j]\!]) \neq \emptyset, \text{ where}$$
$$pr = \sum_{act^j\in Act} p(X^j, updt^j, l'^j),$$
$$act = \bigwedge_{act^j\in Act} act^j.$$

All the cases of duplication of act or all the nondeterministic combination in addition of probability is treated by $2_{ne}^{\mathtt{extract}(p)}$, where notation *ne* means non-empty set.

We define the following precondition operators to calculate the state transition relation in probabilistic linear hybrid automata symbolically and collectively.

4.1 Precondition Operators

We define the time-precondition operator and the discrete-precondition operator based on the non-probabilistic precedent of [1].

Non-probabilistic hybrid automata case was showed in [1]. So, emphasis is placed on probabilities and the definition of precondition operators follows from probabilistic transition of $\mathsf{CPS}_{\mathtt{PLHA}}$, we can define the following precondition operators according to Definition 6. Because we use the backward algorithm later, the operations are inverse image computations defined as follows.

Definition 10: (Time Precondition)
We write $\mathtt{tpre}(\mathtt{plf}(l))$ for the probabilistic linear formula such that from any state in the corresponding region $(l, [\![\mathtt{tpre}(\mathtt{plf}(l)]\!])$ a state in $(l, [\![\mathtt{plf}(l)]\!])$ can be reached in a single time transition.

$$\mathtt{tpre}(\mathtt{plf}(l)) = \bigcup_{(f_l,P)\in\mathtt{plf}(l)} \{(inv(l) \wedge (\exists\delta \geq 0.\exists\vec{c}.$$
$$(((\delta\cdot dif(l))[\vec{x} := \vec{c}] \wedge (f_l \wedge inv(l))[\vec{x} := \vec{x} + \vec{c}])), P\cdot 1)\}.$$

Definition 11: (Discrete Precondition)
We write $\mathtt{dpre}(l', \mathtt{plf}(l'))$ and $\mathtt{expre}(l', \mathtt{plf}(l'))$ for the predicate, the corresponding region of states from which a state in $(l', [\![\mathtt{plf}(l')]\!])$ can be reached in a single discrete transition according to probabilistic edges E and extra edges $\mathcal{E}$, respectively.

$$\mathtt{dpre}(l', \mathtt{plf}(l')) = \bigcup_{e=(l,g,p,X,updt,l')\in E} \{(l, \bigcup_{\substack{(f_{l'},P)\\ \in\mathtt{plf}(l')}} \{(inv(l) \wedge$$
$$\exists\vec{x}'.(g \wedge act \wedge (f_{l'} \wedge inv(l'))[\vec{x} := \vec{x}']),$$
$$P \cdot p(X, updt, l'))\})\}.$$
$$\mathtt{expre}(l', \mathtt{plf}(l')) = \bigcup_{e=(l,g,pr,act,l')\in\mathcal{E}} \{(l, \bigcup_{(f_{l'},P)\in\mathtt{plf}(l')}$$
$$\{(inv(l) \wedge \exists\vec{x}'.(g \wedge act \wedge (f_{l'} \wedge inv(l'))[\vec{x} := \vec{x}']),$$
$$P \cdot pr)\})\}.$$

Definition 11 enables us to calculate the state transitions that follows the same probability distribution symbolically and collectively. By one calculation, the length of path increases by one, (refer to Definition 5). Since calculation of probability is multiplication, we can calculate it reversely.

Finally, we define precondition operator $\mathtt{pre}$ consisting of time and discrete precondition operator. For a predicate π, any state in the corresponding region $[\![\mathtt{pre}(\pi)]\!]$ can reach to some states in the region $[\![\pi]\!]$ with single probabilistic transitions.

Definition 12: (Precondition Operators)

$$\mathtt{pre}(\pi) = \mathtt{Tpre}(\pi) \cup \mathtt{Dpre}(\pi), \text{ where}$$
$$\mathtt{Tpre}(\pi) = \bigcup_{l\in L} \{(l, \mathtt{tpre}(\mathtt{plf}(l)))\},$$
$$\mathtt{Dpre}(\pi) = \bigcup_{l'\in L} (\mathtt{dpre}(l', \mathtt{plf}(l')) \cup \mathtt{expre}(l', \mathtt{plf}(l'))).$$

It is well known [1]–[3] that we can solve the reachability problem by repeating inverse image computations and calculating all the states where it can follow from the target. We use the backward algorithm, because it is said that the backward algorithm is more efficient than the forward algorithm. We can trace all the operation of probabilistic linear hybrid automaton by using precondition operator $\mathtt{pre}$ previously defined when we perform inverse image computations.

4.2 Symbolic Backward Reachability Analysis Procedure

We propose the symbolic backward reachability analysis procedure as below:

Procedure SRA:
Input: a probabilistic linear hybrid automaton PLHA;
an initial state $init = (l_0, \vec{s}_0)$;
a target predicate
and a probabilistic requirement $(\mathrm{T}, \sqsupseteq, \lambda)$.
Output: YES, *reachable* / NO;
(a region $[\![Q_i]\!]$ which can reach $[\![\mathrm{T}]\!]$.)

Y := T /* Q_0 */
Z := T

repeat
/* computation of a region $[\![Q_i]\!]$ which can reach $[\![T]\!]$. */
Z := pre(Z) \ Y /* $\mathrm{pre}^i \setminus Q_{i-1}$ */
Y := Y ∪ Z /* Q_i */

/* judgment of reachability every time i, Z is the form of $\bigcup_{l\in L}\{(l, \mathtt{plf}(l))\} = \bigcup_{l\in L}\{(l, \bigcup_{(f_l,P)\in \mathtt{plf}(l)}\{(f_l, P)\})\}$. */
for each z = $(l, \mathtt{plf}(l)) \in$ Z wrt $l \in L$
if $l == l_0$
for each $(f_l, P) \in \mathtt{plf}(l)$
if $\vec{s}_0 \in [\![f_l]\!]$
if $P \sqsupseteq \lambda$
return YES, *reachable.*
halt SRA
end if
end if
end for each
end if
end for each

until Z == ∅
Y == Y ∪ Z)

return NO.

In general, the convergence of the least fixed point, and thus the termination of this reachability analysis procedure is not guaranteed, as already the reachability problem for constant-slope hybrid systems is undecidable [1], [2].

Afterwards, we transform predicates into logical formulae with the aim of implementing the above procedure by symbolic computation of logical formulae. Let l_c and P_c be *control variables* that ranges over the set of locations L and the set of real numbers $\mathbb{R}$, respectively. The predicate $\pi = \bigcup_{l\in L}\{(l, \mathtt{plf}(l))\} = \bigcup_{l\in L}\{(l, \bigcup_{(f_l,P)\in \mathtt{plf}(l)}\{(f_l, P)\})\}$ defines the logical formula ϕ in the following manner:

$$\phi = \bigvee_{l\in L}(l_c = l \wedge \mathtt{plf}(l))$$
$$= \bigvee_{l\in L}(l_c = l \wedge (\bigvee_{(f_l,P)\in \mathtt{plf}(l)} (f_l \wedge P_c = P)))$$

In precondition operators, union operators $\bigcup$ and $\cup$ are replaced by disjunctions $\bigvee$ and $\vee$, respectively.

4.3 Examples

We will show some simple example as follows:

We consider probabilistic linear hybrid automaton as shown in Fig. 1.

Probabilistic reachability problem $(T, \geq, 0.35)$.

- Probabilistic linear hybrid automaton Fig. 1,
- $init = (l_c = l_1 \wedge x = 0 \wedge y = 1)$,
- $T = (l_c = l_2 \wedge 1 \leq x \leq 2 \wedge 2 \leq y \leq 3 \wedge P_c = 1)$.

Predecessor calculation using quantifier elimination [1], [12].

$\mathtt{tpre}(1 \leq x \leq 2 \wedge 2 \leq y \leq 3 \wedge P_c = 1)$
$= (y \geq 0 \wedge (\exists\delta \geq 0.\exists c_x, c_y.(c_x = \delta \wedge c_y = 2\cdot\delta$
$\wedge 1 \leq x + c_x \leq 2 \wedge 2 \leq y + c_y \leq 3 \wedge y + c_y \geq 0))$
$\wedge P_c = 1 \cdot 1)$
$= (y \geq 0 \wedge (\exists\delta \geq 0.(1 \leq x + \delta \leq 2 \wedge 2 \leq y + 2\delta \leq 3))$
$\wedge P_c = 1)$
$= (x \leq 2 \wedge 0 \leq y \leq 3 \wedge -2 \leq y - 2x \leq 1 \wedge P_c = 1).$
see Fig. 2

$\mathtt{dpre}(l_c = l_2 \wedge 1 \leq x \leq 2 \wedge 2 \leq y \leq 3 \wedge P_c = 1)$
$= (l_c = l_1 \wedge$
$(1 \leq y \leq 2 \wedge \exists x'.\exists y'.(x \leq 3 \wedge x' \geq 1 \wedge y' = x$
$\wedge 1 \leq x' \leq 2 \wedge 2 \leq y' \leq 3 \wedge y' \geq 0))$
$\wedge P_c = 1 \cdot 0.6)$
$= (l_c = l_1 \wedge (1 \leq y \leq 2$
$\wedge \exists x'.(x \leq 3 \wedge 1 \leq x' \leq 2 \wedge 2 \leq x \leq 3)) \wedge P_c = 0.6)$
$= (l_c = l_1 \wedge 2 \leq x \leq 3 \wedge 1 \leq y \leq 2 \wedge P_c = 0.6) = \phi_1.$
seeFig. 3

$\mathtt{pre}(T) = \mathtt{Tpre}(T) \vee \mathtt{Dpre}(T)$
$= (l_c = l_2 \wedge x \leq 2 \wedge 0 \leq y \leq 3 \wedge -2 \leq y - 2x \leq 1 \wedge P_c = 1)$
$\vee \phi_1 \vee T$
$= \phi_1 \vee \phi_2 \vee T.$

Reachability determination.

For $\mathtt{pre}^2 = \mathtt{pre}(\mathtt{pre}(T) \setminus T) = \mathtt{pre}(\phi_1 \vee \phi_2) = \mathtt{pre}(\phi_1) \vee \mathtt{pre}(\phi_2)$, $\mathtt{cpre}(\phi_1) = (l_c = l_1 \wedge x \leq 3 \wedge 1 \leq y \leq 2 \wedge 2x - y \leq 5 \wedge 2y - x \leq 2 \wedge P_c = 0.6)$ and *init* have a non-empty intersection. Then probabilistic requirement is satisfied ($P_c = 0.6 \geq 0.35$), and therefore we conclude this reachability problem with "YES."

The example of operation shown in Sect. 2.2.1 is correctly calculated in Figs. 2 and 3 that uses the procedure described in Sect. 4.2. This calculation is symbolically performed.

We have implemented a prototype of verifier based on MATHEMATICA.

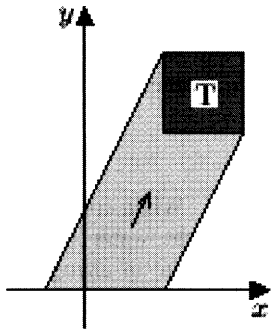


Fig. 2 Time precondition.

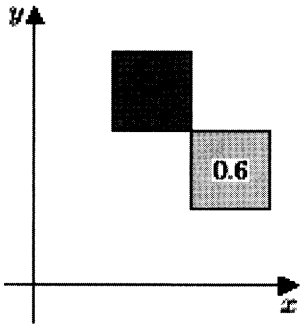


Fig. 3 Discrete precondition.

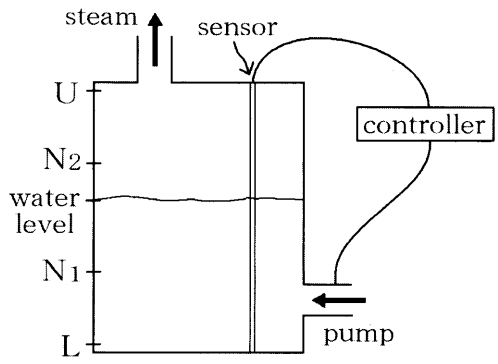


Fig. 4 Steam boiler.

5. Case Study

5.1 Probabilistic Steam Boiler

We now consider the problem of modelling an industrial application [13], [14], namely that of a *steam boiler* (Fig. 4), using probabilistic linear hybrid automata. The system consists of a number of physical units, namely a vessel containing an amount of water, a water pump, a series of sensors, and a message transmission system. The vessel is continuously heated in order to produce steam. The constant L denotes the minimal limit level of water, with U denoting the corresponding maximal limit level. When the water level remains within the normal operating interval of $[N_1, N_2]$, the controller need not intervene. However, if the water level falls below N_1, then the pump is switched on, and if the water level rises above N_1, the pump is switched off.

There is the possibility of a failure in the water level sensor [6], [15]. Given the occurrence of such a failure, the controller uses an approximate guess of the actual water level when deciding whether to switch the pump on or off. Periodically, there is the possibility of the water level sensor being repaired.

5.2 Modelling

Probabilistic linear hybrid automaton to model the steam boiler system described above is given in Fig. 5. We ease the graphical notation by enclosing the locations in the dotted boxes, and draw a single edge from each box to the location. The variable w denotes the water level, t and cl are clocks, g_l represents the lower bound on the current guess on the water level, g_u represents the corresponding upper bound.

Location Off and On. When control resides in these two locations, the value of the water level is affected by the steam. We express the rate of change of the steam emission volume as $0 \leq \dot{s} \leq e$ for some positive integer constant e. In the location `On` the pump being on. The pump water the vessel at any rate between 0 and p litres per time units.

Both location have the invariant condition $t \leq \Delta$; therefore, control must leave either of these locations if the value of the clock t is equal to Δ.

Location Urgent off and Urgent on. The purpose of the two locations is to ease the graphical notation of probabilistic linear hybrid automaton. In these locations no time is permitted to elapse. They correspond to the controller making estimates of the water level.

All of the distributions available in these locations reset the *guess* variables, g_l and g_u.

Location Off/failure and On/failure. Naturally, these two locations correspond to the case in which the water level sensor has failed. As the controller is now maintaining an estimate of the water level, the flow conditions of the variables representing the bounds on the guess of the water level, g_l and g_u, are altered to take into account the fact that the real water level may change as time elapses.

Location Emergency stop and Shutdown. If the real water level falls below the lower limit L or exceed the upper limit U, then control can pass to the terminal location `Emergency stop`.

If the lower bound on the estimate of the water level is below the lower normal water level at the same time as the upper bound on the estimate is above the upper normal level, then control can pass to a terminal location `Shutdown`.

An example run of the probabilistic steam boiler control is shown in Fig. 6. We assume that the system constants are such that $L < N_1 < G_1 < G_2 < N_2 < U$, and that the initial water level of the boiler, denoted by w_0, is some point in the interval (G_1, G_2). When the system commences operation, the pump is switched off.

5.3 Analysis

5.3.1 Property Description

In this paper, the property that we wish to check is *whether or not the system can reach the terminal location* (`Emergency stop`, `Shutdown`) *in* 10 *time units have*

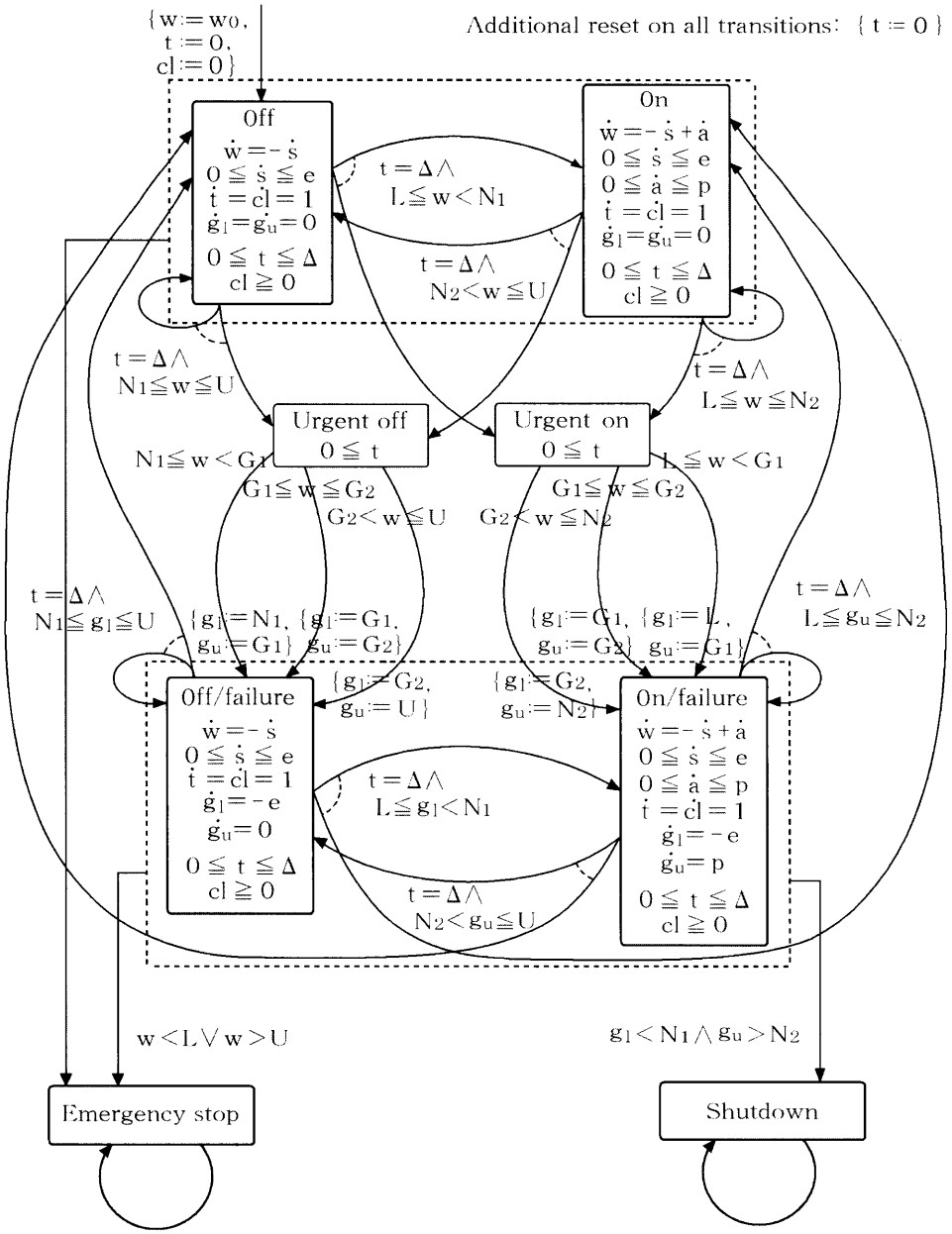


Fig. 5 Probabilistic linear hybrid automaton for probabilistic steam boiler.

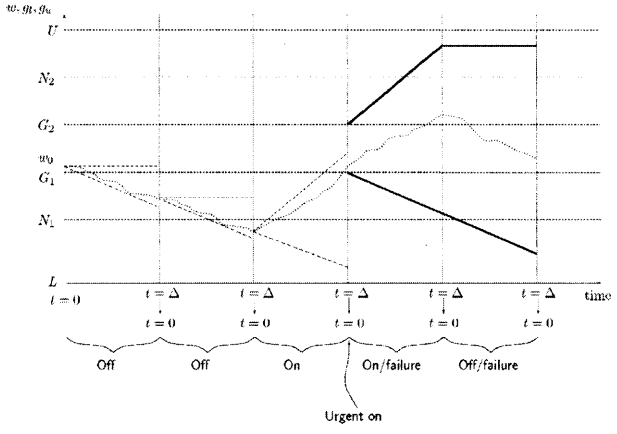


Fig. 6 An example run of the probabilistic steam boiler.

elapsed.

However, it is possible to ignore the location `Emergency stop` by doing a little arithmetic as follows; if the amount of the steam emission volume s during Δ time units is not more than N_1 *minus* L at the same time as the amount of water absorption a during Δ time units is not more than U *minus* N_2, then the system never reaches the location `Emergency stop`. Therefore, we consider only the location `Shutdown` the terminal location. We note states T for which it is possible to make a single transition in order to reach `Shutdown`; we omit the actual target states (`Shutdown`) for simplicity.

Finally, the probabilistic reachability property that requires that the system reach the location `Shutdown` within

10 time units, with the probability strictly greater than 0, can be expressed as the tuple (T, >, 0).

Our first task is to specify the probabilities of the discrete transitions of the steam boiler control. For convenience, we select very simple distributions. We decide that the possibility of a failure in the water level sensor is 0.1, otherwise 0.9.

5.3.2 Implementation and Results

The results are as follows. "The system *can reach* the states for which it is possible to make a single transition in order to reach `Shutdown` within 10 time units, with probability 0.1 at 4th transition."

This automatic analysis took about 14 minutes and up to 100 MB memory. (Intel Pentium 4 CPU 3.00 GHz with 2.00 GB RAM under MS Windows XP OS on MATHEMATICA 5.0). MATHEMATICA can execute the commands such as *quantifier elimination*,QE [12]. We can transform the formula into another formula that is equivalent to the former, and does not contain $\exists$ by using QE. We use this QE when we compute the `tpre()`, `dpre()`, and `expre()`. This is fully automated in analysis.

Here, we refer to the symbolic run (set of paths) that has reached `Shutdown`, as follows.

$$
\begin{aligned}
&(l_c = \texttt{Off} \wedge\ 0 \le cl < 1 \wedge 0 \le t \le 5 \wedge 0 \le w < 35 - 3t \\
&\quad \vee 1 \le cl < 6 \wedge -1 + cl < t \le 5 \wedge 0 \le w < 35 - 3t) \\
&\quad \rightarrow (l_c = \texttt{Off} \wedge\ 0 \le w < 20 \wedge 0 \le cl < 6 \wedge t = 5) \\
&\quad \rightarrow (l_c = \texttt{Urgent on} \wedge\ 0 \le w < 30 \wedge 0 \le cl < 6 \wedge t = 0) \\
&\quad \rightarrow (l_c = \texttt{On/failure} \wedge\ 0 \le t \le 5 \wedge \\
&\qquad (0 \le cl \le 5 + t \wedge g_u > 25 + 5t \wedge g_l < 35 - 3t \\
&\quad \vee 5 + t < cl \le 10 \wedge g_u > 5cl \wedge g_l < 50 - 3cl)) \\
&\quad \rightarrow (l_c = \texttt{On/failure} \wedge \\
&\quad g_l < N_1 \wedge g_u > N_2 \wedge 0 \le t \le \Delta = 5 \wedge 0 \le cl \le 10)
\end{aligned}
$$

The probability of these paths is 0.1. Note that the first region contains the system commences operation $init = (l_c = \texttt{Off} \wedge\ w = 0 \wedge t = 0 \wedge cl = 0)$, and the last region is contained the target T.

In this paper, we regarded the states for which it is possible to make a single transition in order to reach `Shutdown` as the target region. Generally, it is undesirable that the system can enter `Shutdown`. So, we should remodel the design parameter of the steam boiler control to avoid the above case.

Now, we change the parameter (initial water level w_0 and guard G_1) and analyze again. Then the system do not enter the undesirable target within 10 time units in four times transition.

6. Conclusions

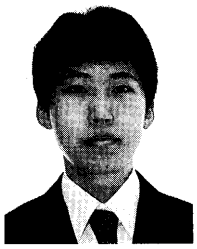
In this paper, we have defined the probabilistic linear hybrid automata as system modelling language, and presented its symbolic reachability analysis method. By our method, we will be able to handle many systems such as distributed control systems, timed randomized protocols and so on. We have implemented an experimental verification system using MATHEMATICA, and demonstrated that our method can help the system designer to choose critical system parameters, via case study. We are now working for an abstraction/approximation method of probabilistic linear hybrid automata to handle complicated realistic problems.

References

[1] R. Alur, T.A. Henzinger, and P.-H. Ho, "Automatic symbolic verification of embedded systems," IEEE Trans. Softw. Eng., vol.22, no.3, pp.181–201, 1996.
[2] R. Alur, C. Coucoubetis, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," Theor. Comput. Sci., vol.138, pp.3–34, 1995.
[3] T.A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine, "Symbolic model checking for real-time systems," Inf. Comput., vol.111, pp.193–244, 1994.
[4] M. Kwiatkowska, G. Norman, and J. Sproston, "Symbolic model checking for probabilistic timed automata," Technical Report CSR-03-10, School of Computer Science, University of Birmingham, 2003.
[5] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, "Automatic verification of real-time systems with discrete probability distributions," Theor. Comput. Sci., vol.282, pp.101–150, 2002.
[6] J. Sproston, Model checking for probabilistic timed and hybrid systems, Ph.D. Thesis, Technical Report CSR-01-04, School of Computer Science, University of Birmingham, 2001.
[7] J. Sproston, "Analyzing subclasses of probabilistic hybrid automata," Technical Report CSR-99-8, School of Computer Science, University of Birmingham, 1999.
[8] J. Sproston, "Decidable model checking of probabilistic hybrid automata," Lecture Notes in Computer Science 1926, pp.31–45, Springer-Verlag, 2000.
[9] S. Hart, M. Sharir, and A. Pnueli, "Termination of probabilistic concurrent program," ACM Trans. Programming Languages and Systems (TOPLAS), vol.5, no.3, pp.356–380, 1983.
[10] E.A. Emerson, "Temporal and modal logic," in Handbook of Theoretical Computer Science (vol.B): Formal Models and Semantics, pp.995–1072, MIT Press, 1991.
[11] C. Courcoubetis and M. Yannakakis, "The complexity of probabilistic verification," J. ACM, vol.42, no.4, pp.857–907, 1995.
[12] A. Tarski, A decision method for elementary algebra and geometry, 2nd ed., University of California Press, Berkeley and Los Angeles, California, 1951.
[13] J.R. Abrial, E. Borger, and H. Langmaack, eds., "Formal methods for industrial applications: Specifying and programming the steam boiler control," Lecture Notes in Computer Science, vol.1165, pp.1–12, Springer-Verlag, 1996.
[14] T.A. Henzinger and H. Wong-Toi, "Using HYTECH to synthesize control parameters for a steam boiler," Lecture Notes in Computer Science, vol.1165, pp.265–282, Springer-Verlag, 1996.
[15] A. McIver, C. Morgan, and E. Troubitsyna, "The probabilistic steam boiler: A case study in probabilistic data refinement," Proc. IRW/FMP'98, Australia, 1998.
[16] R. Alur, T. Dang, and F. Ivancic, "Reachability analysis of hybrid systems using counter-example guided predicate abstraction," Technical Report MS-CIS-02-34, University of Pennsylvania, 2002.
[17] E.M. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald, "Abstraction and counterexample-guided refinement in model checking of hybrid systems," Technical Report

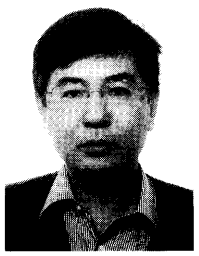
CMU-CS-03-104, School of Computer Science, Carnegie Mellon University, 2003.

[18] E.M. Clarke, O. Grumberg, and D.A. Peled, Model checking, MIT Press, 1999.



Yosuke Mutsuda received B.S. in Information and Systems Engineering from Kanazawa University, in 2004. He is now a master student in Graduate School of Natural Science & Technology in Kanazawa University. His recent research area is formal specification and verification of probabilistic real-time hybrid systems. He is a student member of IPSJ.

Takaaki Kato received B.S. in Information and Systems Engineering from Kanazawa University, in 2005. He is now a master student in Graduate School of Natural Science & Technology in Kanazawa University. His recent research area is formal specification and verification of probabilistic real-time hybrid systems.



Satoshi Yamane received Ph.D. from Kyoto University of Compuer Science. Now, he is a professor of Natural Science & Technology in Kanazawa University. He is interested in formal specification and verification of real-time hybrid systems.