Development of probabilistic timed CEGAR

著者	Yamane Satoshi, Shimizu Takaya
journal or publication title	2014 2nd International Conference on Systems and Informatics, ICSAI 2014
number	7009336
page range	482-491
year	2015-01-01
URL	http://hdl.handle.net/2297/41336

doi: 10.1109/ICSAI.2014.7009336

Development of Probabilistic Timed CounterExample Guided Abstraction Refinement

Satoshi Yamane
Kanazawa University
Email: syamane@is.t.kanazawa-u.ac.jp

Takaya Shimizu
Kanazawa University
Email: shimizu@csl.ec.t.kanazawa-u.ac.jp

***Abstract*—In this paper, we present an efficient verification method for probabilistic timed automaton. This method based on predicate abstractions and refinements realizes effective automated verifications for real-time and probabilistic embedded systems.**

***Keywords*-probabilistic timed automaton, model checking, CEGAR(CounterExample Guided Abstraction Refinement), predicate abstraction**

I. Introduction

A. Background

E.M.Clarke has proposed model checking [2] based on CEGAR(CounterExample Guided Abstraction Refinement) [9] of reactive systems. As it is possible to avoid the state space explosion by predicate abstraction [1], CEGAR attracts attention. Of special interests in model checking are safety properties, which assert that the system does not reach a bad state. The safety is verified by reachability analysis.

In this paper, we propose the reachability analysis of a probabilistic timed automaton based on CEGAR, what is called Probabilistic Timed CEGAR. In order to develop the verifier of probabilistic timed automaton based on CEGAR, we must construct a sound abstract model, and analyze counterexamples, refine the abstract model. In this paper, we develop Probabilistic Timed CEGAR by constructing a sound abstract model, and analyzing counterexamples, refining the abstract model for a probabilistic timed automaton. In development of the above-mentioned technique, the simultaneous practicability of two or more paths by a probabilistic branch is judged in Simultaneous Counterexample Analysis, and the refinement technique of the abstract model by the spurious counterexample is realized. Moreover, the experiment of this technique is implemented, and it is shown by comparing with existing techniques [8], [23] that verifications with smaller state spaces are possible.

B. Related works

Various CEGAR verifiers have been studied for various systems. Real-time CEGAR [10] has been developed for real-time systems, Hybrid CEGAR [21], [22] has been developed in hybrid systems. Probabilistic CEGAR [14] has been developed in probabilistic systems. As related works of probabilistic timed CEGAR, H.Hermanns proposes the abstraction of a probabilistic automaton using a stochastic game semantics [15]. But he does not propose refinements, also predicate abstractions. On the other hand, symbolic model checking has been developed for a probabilistic timed automaton [8], [23]. Also, J. Sproston has studied the computational complexity of model checking of a probability time automaton [20]. Our proposed new Probabilistic timed CEGAR is the verification technique reducing the number of states, and can expect an efficient verification.

II. Probabilistic timed automata

A. Preliminaries

First we define distributions and clocks.

Definition 1 (Discrete probability distribution).
A discrete probability distribution over a finite set Q is a function $p : Q \to [0,1]$ such that $\sum_{q\in Q} p(q) = 1$. For a possibly uncountable set Q_∞, let $\mathsf{Dist}(Q_\infty)$ be the set of distributions over finite subsets of Q_∞. ■

Definition 2 (Clocks and clock valuations).
A clock is a real-valued variable which increases at the same rate. Let C be a set of clocks. A valuation of ν is a function $\nu : C \to \mathbb{R}^{\geq 0}$. We denote the set of all clock valuations by $\mathcal{V}_C$. For any $\delta \in \mathbb{R}$ and $\nu \in \mathbb{R}^{\geq 0}$, we use $(\nu+\delta)$ to denote the clock valuation defined as $(\nu+\delta)(x) = \nu(x) + \delta$ for all $x \in C$. We use $\nu[X := 0]$ to denote the clock valuation obtained from ν by resetting all of the clocks in $x \in X$ to 0, and leaving the values of all other clocks unchanged. ■

Next, we define zones as the set of clock valuations.

Definition 3 (Zones).
The set of zones of C, written $Zones(C)$, is defined inductively by the syntax:

$$\zeta ::= x \leq c | x < c | x > c | x \geq c | x_1 - x_2 \leq d | x_1 - x_2 < d | \mathrm{true} | \zeta \wedge \zeta$$

, where $x \in C$, $c \in \mathbb{N}$, $d \in \mathbb{Z}$.

The clock valuation ν satisfies the zone ζ, written $\nu \triangleright \zeta$, if and only if ζ resolves to true after substituting each clock $x \in C$ with the corresponding clock value $\nu(x)$ from ν. The semantics of a zone ζ is the set of clock valuations which *satisfy the zone ζ. The zone ζ satisfies the clock valuation ν_0 , written $\zeta_0 = \{\nu_0\}$.* ■

B. Syntax

We now present the formal syntax of a probabilistic timed automaton.

Definition 4 (Probabilistic timed automaton). *A probabilistic timed automaton G is a tuple $(L, l_0, C, Inv, prob)$ where:*

- *L is a finite set of locations*
- *$l_0 \in L$ is an initial location*
- *C is a finite set of clocks*
- *$Inv : L \to Zones(C)$ is a function called an invariant condition*
- *$prob \subseteq L \times Zones(C) \times \mathsf{Dist}(2^C \times L)$ is a finite set called the probabilistic edge relation* ■

A state s of G is a pair (l, ν). G behaves from an initial state (l_0, ν_0) by discrete transitions or timed transitions.

In any state (l, ν), there is a nondeterministic choice of either (1)letting time pass or (1)making a discrete transition. In case (1), a timed transition is performed within the same location, and when a $t \in \mathbb{R}^{>0}$ timed transition carries out in state (l, ν), it becomes a state $(l, \nu + t)$ by a probability 1, where according to the invariant conditions of a location, t must satisfy $\nu + t \rhd Inv(l)$. In case (2), a discrete transition can be made according to any $(l, \zeta_g, p) \in prob$, where $l \in L$ is a source location, $\zeta_g \in \mathrm{Zones}(C)$ is a zone condition, $p \in \mathsf{Dist}(2^C \times L)$ is a distribution. The probability of moving to the location $l' \in L$ and resetting all of the clocks in the set $X \in 2^C$ to 0 is given by $p(X, l') \in (0, 1]$. When (l, ζ_g, p) is a transition relation and the set of clocks to reset is X, and l moves to l', the state (l, ν) becomes $(l', \nu[X := 0])$ by the discrete transition.

Discrete transitions are possible when filling two of the followings.

- For any $(l, \zeta_g, p) \in prob$, clock valuations in a source state (l, ν) satisfy a zone condition ζ_g.
- Clock valuations in a target state $(l', \nu[X := 0])$ satisfy the invariant condition of l'.

In the result, both $\nu \rhd \zeta_g$ and $\nu[X := 0] \rhd Inv(l')$ must hold true.

Example 1. *We explain behaviors of a probabilistic timed automaton G_1 in Figure 1. $x, y \in C$ are clocks,* $\mathsf{start}, \mathsf{done}, \mathsf{abort} \in L$ *are locations. G_1 makes a transition either from* start *to* start *with the probability* 0.2 *or to* done *with the probability* 0.8. ■

C. Semantics

We give the semantics of a probabilistic timed automaton G defined in terms of a timed probabilistic system $\mathcal{M}$[8], which is a Markov decision process.

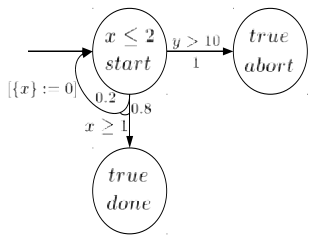

Figure 1. Probabilistic Timed Automaton G_1

Definition 5 (Timed probabilistic system). *Let $G = (L, l_0, C, Inv, prob)$ be a probabilistic timed automaton. The semantics of G is defined as the timed probabilistic system $\mathcal{M} = (S, s_0, Steps)$ where:*

- *$S \subseteq L \times \mathcal{V}_C$ is a set of states*
- *$s_0 = (l_0, \nu_0)$ is an initial state*
- *$Steps \subseteq S \times \mathbb{R}^{>0} \times \mathsf{Dist}(2^C \times S)$ is a finite set of transitions*

$(l, \nu) \in S$ must satisfy $\nu \rhd Inv(l)$. $Steps$ consists of timed transitions and discrete transitions. Let $\mu \in \mathsf{Dist}(2^C \times S)$ be a probability distribution. We define the transition relation $((l, \nu), t, \mu) \in Steps$ from $(l, \nu) \in S$ to $(l', \nu') \in S$ as follows, where $t \in \mathbb{R}^{>0}$ is an elapsed time and $(l, \zeta_g, p) \in prob$ is a probabilistic transition relation of G. Especially let $\mu_\perp$ be a probability distribution of timed transitions.

(1)t-timed transition:

$$(l, \nu) \overset{t, \mu_\perp(\emptyset, (l', \nu'))}{\longrightarrow} (l', \nu')$$

, where

$$\mu_\perp(\emptyset, (l', \nu')) = \begin{cases} 1 & \text{if } l' = l \wedge \forall t'.(0 \leq t' \leq t \wedge \nu' = \nu + t' \wedge \nu' \rhd Inv(l)) \\ 0 & \text{otherwise} \end{cases}$$

(2)Discrete probabilistic transition by (l, ζ_g, p):

$$(l, \nu) \overset{0, \mu(X, (l', \nu'))}{\longrightarrow} (l', \nu')$$

, where

$$\mu(X, (l', \nu')) = \begin{cases} p(X, l') & \text{if } \nu \rhd \zeta_g \wedge \nu' = \nu[X := 0] \\ 0 & \text{otherwise} \end{cases}$$

■

$\mathcal{M}$ moves to (l', ν') by resetting clocks X by the probabilistic transition of $\mu(X, (l', \nu'))$. For discrete transitions, since the probability distributions $\mu(X, (l', \nu'))$ on $\mathcal{M}$ corresponds to the probability distribution $p(X, l')$ on G, $p(X, l')$ is equal to $\mu(X, (l', \nu'))$.

A path ω of $\mathcal{M}$ is a non-empty finite or infinite sequence of transitions from an initial state s_0.

$$\omega = s_0 \overset{t_0, \mu_0(X_0, s_1)}{\longrightarrow} \cdots \overset{t_{i-1}, \mu_{i-1}(X_{i-1}, s_i)}{\longrightarrow} s_i \overset{t_i, \mu_i(X_i, s_{i+1})}{\longrightarrow} \cdots$$

In ith transition, t_i is an elapsed time of a timed transition, and μ_i is a probability distribution, X_i is a set of reset clocks, s_{i+1} is a target state. Here our timed probabilistic system is a strictly divergent [20]. Strict divergence means all the paths have time divergence [3]. Also J. Sproston proposes EXPTIME algorithm for model checking of a timed probabilistic automaton with strict divergence [20]. We say that a finite path ω_{fin} of length $|\omega_{fin}|$ is a prefix of an infinite path ω_{ful}. The last state of a finite path ω_{fin} is denoted by $last(\omega_{fin})$. The sets of all finite and infinite paths starting in state s_0 are denoted $Path_{fin}$ and $Path_{ful}$, respectively. For any path, we denote by $\omega_{fin}(i)$ and $\omega_{ful}(i)$ the $(i+1)$th state.

We define the set of paths reaching the set S_e of states as follows:

- For a finite path
  $Path_{fin}(S_e) = \{\omega_{fin} \in Path_{fin} | last(\omega_{fin}) \in S_e\}$
- For an infinite path
  $Path_{ful}(S_e) = \{\omega_{ful} \in Path_{ful} | \exists i.\omega_{ful}(i) \in S_e\}$

Next we define an adversary to resolve nondeterministic behaviors.

Definition 6 (An adversary of a timed probabilistic system). *An adversary A of a timed probabilistic system $\mathcal{M}$ is a fuction $A : Path_{fin} \rightarrow \mathbb{R}^{\geq 0} \times Dist(2^C \times S)$ such that $(last(\omega_{fin}), t, \mu) \in Steps$. We denote the set of all the adversaries by Adv.* ■

An adversary represents a resolution of nondeterministic behaviors, and maps every finite path to both a timed transition and a probability distribution. Here a timed transition corresponds to both an elapsed time t and a probability distribution $\mu_\perp$. We let $Path^A_{ful}$ (respectively, $Path^A_{fin}$) denote the subset of $Path_{ful}$ (respectively, $Path_{fin}$) induced by A. For any infinite path $\omega_{ful} \in Path^A_{ful}$, we can define $A(\omega_{i\text{-}th}) = (t_i, \mu_i)$ by $\omega_{i\text{-}th} \in Path^A_{fin}$, which is a prefix of an infinite path.

D. Discrete time Markov chains

We construct a discrete time Markov chain $\mathcal{MC}^A = (S, s_0, \mathrm{P}^A)$ from a timed probabilistic system $\mathcal{M} = (S, s_0, Steps)$ by resolving non-deterministic behaviors using any adversary A. The S and s_0 of a timed probabilistic system correspond to the S and s_0 of a discrete time Markov chain. $\mathrm{P}^A : Path^A_{fin} \times Path^A_{fin} \rightarrow [0,1]$ can be described for any finite path of a discrete time Markov chain as follows:

$$\mathrm{P}^A(\omega_{fin}, \omega'_{fin}) = \begin{cases} \mu(X, s') & \text{if } \exists \mu.(A(\omega_{fin}) = (t,\mu) \text{ } and \\ & \omega'_{fin} = \omega_{fin} \xrightarrow{t,\mu(X,s')} s' \\ 0 & \text{otherwise} \end{cases}$$

Next for any finite path ω_{fin}, we define the probability $Prob^A_{fin} : Path^A_{fin} \rightarrow [0,1]$ as follows:

$$Prob^A_{fin}(\omega_{fin}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } |\omega_{fin}| = 0 \\ \prod_{i=0}^{|\omega_{fin}|-1} \mathrm{P}^A(\omega_{i\text{-}th}, \omega_{(i+1)\text{-}th}) & \text{otherwise} \end{cases}$$

A finite path $\omega_{i\text{-}th}$ of length i is a prefix of ω_{fin}.

Next we define the cylinder of a finite path ω_{fin} for an adversary A as follows:

$$C^A(\omega_{fin}) \stackrel{\text{def}}{=} \{\omega \in Path^A_{ful} | \omega_{fin} \quad is \quad a \quad prefix \quad of \quad \omega\}$$

Let Σ^A be the smallest σ-algebra on $Path_{fin}$ which contains the cylinders $C^A(\omega_{fin})$ for any $\omega_{fin} \in Path_{fin}$. Finally, we define $Prob^A$ on Σ^A as the unique measure for any $\omega_{fin} \in Path_{fin}$ as follows:

$$Prob^A(C^A(\omega_{fin})) \stackrel{\text{def}}{=} Prob^A_{fin}(\omega_{fin})$$

Here a set of infinite paths reaching to a set S_e of states is $\{\omega \in Path^A_{ful} | \exists i \in \mathbb{N}.\omega(i) \in S_e\} \in \Sigma^A$. The probability reaching from an initial s_0 to a set S_e by an adversary A can be given as $Prob^A(\{\omega \in Path^A_{ful} | \exists i \in \mathbb{N}.\omega(i) \in S_e\})$. We denote this probability by $Prob^A(S_e)$. The maximal reachability probability of a Markov decision process can be computed by resolving non-deterministic behaviors using a simple adversary [13]. But the maximal reachability probability of a time probabilistic system can not be computed by resolving non-deterministic behaviors using a simple adversary [20]

E. Probabilistic reachability problem and counterexamples

In this paper, given a probabilistic timed automaton, we verify a reachability problem, which is a safety property.

Definition 7 (Reachability problem). *Let $S_e = \{(l_e, \nu) \in S | l_e \in L_e\}$ be a set of states for a set $L_e \subseteq L$ of locations. Also, let $\lambda \in [0,1]$ be the probability of a set of paths reaching a set S_e of states from s_0.*

Let the reachability problem of G be a tuple (λ, L_e), where $\lambda \in [0,1]$ is the probability of a set of paths reaching a set S_e, and $L_e \subseteq L$ is a set of target locations.

$\forall A.Prob^A(S_e) \leq \lambda$ iff the reachability problem outputs "yes" for any $\mathcal{M}$. ■

The following proposition clearly holds true for a reachability problem (λ, L_e) of a probabilistic timed automaton G.

Proposition 1 (An adversary in reachability problem). *If the probability of a set of paths reaching a set S_e is less than or equal to $\lambda \in [0,1]$ for any adversary A, (λ, L_e) is "yes". Otherwise, (λ, L_e) is "no".* ■

If (λ, L_e) is "no", there is an adversary such that the probability of a set of paths reaching a set S_e is greater than $\lambda \in [0, 1]$, and then probabilistic timed automaton G is not safe.

Next we define a counterexample.

Definition 8 (Counterexamples).
A counterexample for $\forall A.Prob^{A}(S_e) \leq \lambda$ is a pair (A, Ω) where A is an adversary and Ω is a set of finite paths such that

$$\sum_{\omega \in \Omega} Prob^{A}_{fin}(\omega) > \lambda$$

■

The theorem that a finite counterexample for PCTL formula exists [11] is famous. Also, a finite counterexample for $\forall A.Prob^{A}(S_e) \leq \lambda$ exists.

Theorem 1 (A finite counterexample).
A finite counterexample for $\forall A.Prob^{A}(S_e) \leq \lambda$ exists.

It follows that:

$\neg(\forall A.Prob^{A}(S_e) \leq \lambda) = \exists A.Prob^{A}(S_e) > \lambda$

iff $\sum_{\omega \in \Omega} Prob^{A}_{fin}(\omega) > \lambda$

, where a finite set $\Omega \subseteq \{\omega \in Path^{A}_{fin} | last(\omega) \in S_e\}$. ■

By this theorem, we can verify whether a counterexample is spurious or not.

F. A restricted timed probabilistic system

In this paper, a timed probabilistic system(Definition 5) is a strictly divergent system [20], which is a non-zeno system not containing zeno behaviors. But an abstract timed probabilistic system may contain zeno behaviors. Therefore we construct a restricted timed probabilistic system $\mathcal{M}_R$ by extending the predicate abstraction of timed systems [10].

Definition 9 (A restricted timed probabilistic system).
A restricted timed probabilistic system $\mathcal{M}_R = (S, s_0, Steps_R)$ as the semantics of G restricts timed transitions of a timed probabilistic system $\mathcal{M} = (S, s_0, Steps)$.

Consider $t \in \mathbb{R}$ such that

$\exists x \in C. \exists k \in \{0, \cdots, c\}.(\nu(x) = k \vee (\nu(x) < k \wedge \nu(x) + t \geq k))$*, where $c \in N$ is the largest constant in $\mathcal{M}$.*

For the above $t \in \mathbb{R}$, a restricted timed transition is

$(l, \nu) \xrightarrow{t, \mu_{\perp}(\emptyset, (l', \nu'))} (l', \nu')$

, where $\mu_{\perp}$ is as follows:

$$\mu_{\perp}(\emptyset, (l', \nu')) = \begin{cases} 1 & \text{if } l' = l \wedge \nu' = \nu + t \wedge \nu' \in Inv(l) \\ 0 & \text{otherwise} \end{cases}$$

A restricted timed probabilistic system $\mathcal{M}_R$ has only restricted timed transitions in a timed probabilistic system $\mathcal{M}$. ■

Theorem 2 (A restricted timed probabilistic system).
The output of a reachability problem (λ, L_e) of $\mathcal{M}$ is "yes" iff the output of a reachability problem (λ, L_e) of $\mathcal{M}_R$ is "yes". ■

In this paper, we verify the reachability problem of $\mathcal{M}$ by abstracting and refining $\mathcal{M}_R$ instead of $\mathcal{M}$ from the above theorem. We denote a restricted timed probabilistic system $\mathcal{M}_R$ by $\mathcal{M}$.

III. Predicate abstraction

Predicate abstraction [1] is used to compute a finite approximation of a given infinite state transition system in order to avoid the state space explosion. When we verify real-time systems, the method is based on a set of abstraction predicates, which are predicates over clock valuations [10].

A. Abstraction predicates

First we define a set of abstraction predicates.

Definition 10 (Abstraction predicates).
Given a set of clocks C, an abstraction predicate ψ with respect to C is defined as follows:

$$\psi ::= x_1 \leq c | x_1 < c | x_1 - x_2 < d | \text{true}$$

, where $x_1, x_2 \in C$, $c \in \mathbb{N}$, $d \in \mathbb{Z}$.

The value of an abstraction predicate ψ with respect to a clock valuation ν, where both free and bound variables are interpreted in the domain C, is denoted by the juxtaposition $\psi\nu \in \{true, false\}$. Whenever $\psi\nu$ evaluates to $true$, ν satisfies ψ iff the formula obtained as a result of substituting value $\nu(x)$ corresponding to a clock variable $x \in C$ in ψ holds $true$. For example, $x > c$, $x \geq c$, $x_1 - x_2 \geq d$ are predicates. For any $\nu \in \mathcal{V}_C$, $\psi = \text{true}$ is $\psi\nu = true$. ■

In this paper, we define a set $\Psi^{l} = \{\psi^{l}_{0}, \cdots, \psi^{l}_{n-1}\}$ of abstraction predicates per each location, where ψ^{l_i} is an abstraction predicate in location l. A family of abstraction predicates in all the locations is $\mathbf{\Psi} = \{\Psi^{l_0}, \cdots, \Psi^{l_k}\}$. A set $\Psi^{l} = \{\psi^{l}_{0}, \cdots, \psi^{l}_{n-1}\}$ of abstraction predicates determines an abstraction function $\alpha : S \rightarrow S^{\sharp}$, which maps clock valuations (l, ν) to (l, b^{l}), where is a bit-vector b^{l} of length n, such that the ith component of b is set if and only if ψ^{l}_{i} holds for ν. We define a pair (l, b^{l}) as an abstract state $s^{\sharp}$. We denote a set of abstract states by $S^{\sharp}$. Here, we assume that bit-vectors of length n are elements of the set $\mathcal{B}_n$, which are functions of domain $\{0, \cdots, n-1\}$ and codomain $\{0, 1\}$. $\psi^{l}_{i}\nu = b^{l}(i)$ holds true, where i-th $b^{l}(i)$ of b^{l} in an abstract state (l, b^{l}) is given by $\alpha((l, \nu))$. For example, given $\Psi^{l} = \{x \leq 1, x - y < -1\}$, $s = (l, x = 1 \wedge y = 1)$ is transformed into $\alpha(s) = (l, (true, false))$.

The inverse image of α is the concretization function $\gamma : S^{\sharp} \rightarrow 2^{S}$.

Definition 11 (Abstraction and concretization).
Let C be a set of clocks and $\mathcal{V}_C$ be the corresponding set of clock valuations. Given a finite set $\mathbf{\Psi} = \{\Psi^{l_0}, \cdots, \Psi^{l_k}\}$ of predicates , the abstraction function $\alpha : S \rightarrow S^{\sharp}$ is defined by

$$\alpha((l, \nu)) = (l, b^{l}) \text{ s.t. } \forall i. b^{l}(i) = \psi^{l}_{i}\nu$$

and the concretization function $\gamma : S^\sharp \to 2^S$ is defined by

$$\gamma((l, b^l)) = \{(l, \nu) \in L \times \mathcal{V}_C | Inv(l) \wedge \bigwedge_{i=0}^{n-1} b^l(i) = \psi_i^l \nu\}.$$

Moreover, $b^l \Psi^l$ is defined as a zone by

$$b^l \Psi^l = \{\nu \in \mathcal{V}_C | \bigwedge_{i=0}^{n-1} b^l(i) = \psi_i^l \nu\}.$$

■

B. Abstract model

We abstract a timed probabilistic system using abstraction predicates, abstraction and concretization function, a set $\mathbf{\Psi}$. The abstract model is a Markov decision process. The abstract model is the over-approximation of a timed probabilistic system such as the abstraction of timed systems [10].

Definition 12 (Construction of Abstract model). *By a set $\mathbf{\Psi}$ of predicates, we construct the abstract model $\mathcal{M}^\sharp = (S^\sharp, s_0^\sharp, Steps^\sharp)$ from a timed probabilistic system $\mathcal{M} = (S, s_0, Steps)$, which is transformed from a probabilistic timed system $G = (L, l_0, C, Inv, prob)$ as follows:*

- $S^\sharp = L \times \mathcal{B}_n$
- $s_0^\sharp = \alpha(s_0)$
- $Steps^\sharp \subseteq S^\sharp \times \mathsf{Dist}(2^C \times S^\sharp)$

If $\exists (l, \nu) \in \gamma((l, b)).((l, \nu), \mu) \in Steps$, a transition $((l, b), \mu^\sharp) \in Steps^\sharp$ exists, where $\mu^\sharp$ in $((l, b), \mu^\sharp)$ corresponding to $((l, \nu), t, \mu)$ is defined as follows:

$$\mu^\sharp(X, (l', b')) = \mu(X, (l', \nu'))$$

, where $\alpha((l, \nu)) = (l, b), \alpha((l', \nu')) = (l', b')$. ■

$Steps^\sharp$ does not include timed transitions. On the other hand, $Steps$ includes timed transitions. But we define the probability distribution $\mu_\perp^\sharp$ of $Steps^\sharp$, which is abstracted from the probability distribution $\mu_\perp$ of a timed transition of $Steps$. Also, we define the path of $\mathcal{M}^\sharp$ as follows:

$$\omega^\sharp = s_0^\sharp \overset{\mu_0^\sharp(X_0, s_1^\sharp)}{\longrightarrow} s_1^\sharp \overset{\mu_1^\sharp(X_1, s_2^\sharp)}{\longrightarrow} s_2^\sharp \overset{\mu_2^\sharp(X_2, s_3^\sharp)}{\longrightarrow} \cdots$$

An adversary $A^\sharp$ of $\mathcal{M}^\sharp$ is a function $A^\sharp : Path_{fin}^\sharp \to Dist(2^C \times S^\sharp)$.

Next a basis is a set of abstraction predicates that is expressive enough to distinguish between two clock regions [10]. If a basis is used for predicate abstraction, then the approximation is exact.

Definition 13 (Basis). *[10]*
Let $\mathcal{M}$ be a timed probabilistic system. Then Ψ is a basis with respect to $\mathcal{M}$ iff for all clock valuations $\nu_1, \nu_2 \in \mathcal{V}_C$

$$(\forall \psi \in \Psi \in \mathbf{\Psi}. \nu_1 \quad satisfies \quad \psi \quad iff \quad \nu_2 \quad satisfies \quad \psi)$$

implies

$$\psi \nu_1 \iff \psi \nu_2$$

■

When we biuld $\mathcal{M}^\sharp$ by a basis, $\mathcal{M}^\sharp$ is a region graph [3].

In Probabilistic Timed CEGAR, a set of abstraction predicates is a subset of basis. Since the loop of Probabilistic Timed CEGAR is the technique of the ability to add new predicates certainly each time, the loop of Probabilistic Timed CEGAR halts.

Example 2. *We show the abstract model $\mathcal{M}^\sharp$ from a probabilistic timed automaton G_1(in Figure 1.) by a set of abstract predicates $\{\psi_0^{\mathsf{st}} = y \le 8, \psi_1^{\mathsf{st}} = x - y < -8\}$ in Figure 2.* ■

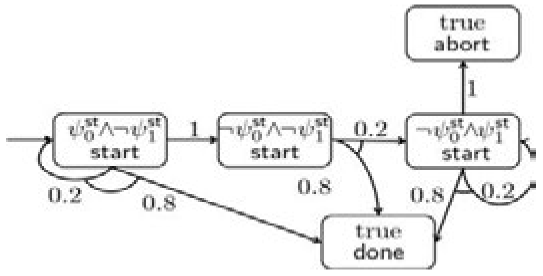
true
abort
1
$\psi_0^{\mathsf{st}} \wedge \neg \psi_1^{\mathsf{st}}$
start
1
$\neg \psi_0^{\mathsf{st}} \wedge \neg \psi_1^{\mathsf{st}}$
start
0.2
$\neg \psi_0^{\mathsf{st}} \wedge \psi_1^{\mathsf{st}}$
start
0.2
0.8
0.8
0.8
0.2
true
done

Figure 2. Abstract Model of $G_1 : \mathcal{M}^\sharp$

C. Correspondence relations between an abstract model and a timed probabilistic system

The abstract model may have a case where it does not have the property of the original model. In this subsection, the correspondence relation between an abstract model and a timed probabilistic system about paths and adversaries is described, moreover the candidate of the counterexample on $\mathcal{M}^\sharp$ corresponding to the counterexample on $\mathcal{M}$ is defined.

1) Path: The timed transition between s_1 and s_2 on $\mathcal{M}$ is performed in an abstract state $s^\sharp$, when $\alpha(s_1) = \alpha(s_2) = s^\sharp$ holds true. Therefore, when we derive ω from $\omega^\sharp$ by Counterexample Analysis, a discrete transition $s^\sharp \xrightarrow{\mu^\sharp}$ in $\omega^\sharp$ is derived into $s_1 \xrightarrow{t, \mu_\perp} s_2 \xrightarrow{0, \mu}$. Thus, the timed transition in this one abstract state is not included in $\omega^\sharp$ corresponding to ω.

Definition 14 (Correspondence relation of paths). *For $\mathcal{M}^\sharp$ constituted by $\mathcal{M}$ and $\mathbf{\Psi}$, a path $\omega^\sharp$ of $\mathcal{M}^\sharp$ corresponding to a path ω of $\mathcal{M}$ is constructed for all the transitions of ω by the following procedures.*

1) *If a transition of ω is a discrete transition $s \overset{0, \mu(X, s')}{\longrightarrow} s'$, a transition of $\omega^\sharp$ is $\alpha(s) \overset{\mu^\sharp(X, \alpha(s'))}{\longrightarrow} \alpha(s')$*
2) *If a transition of ω is a timed transition $s \overset{t, \mu_\perp(\emptyset, s')}{\longrightarrow} s'$ and $\alpha(s) = \alpha(s')$, a transition of $\omega^\sharp$ does not exist.*

3) If a transition of ω is a timed transition $s \xrightarrow{t,\mu_\perp(\emptyset,s')} s'$ and $\alpha(s) \neq \alpha(s')$, a transition of $\omega^\sharp$ is $\alpha(s) \xrightarrow{\mu^\sharp_\perp(\emptyset,s')} \alpha(s')$.

Also, we define the correspondence $\omega \in Path_{fin}$ and $\omega^\sharp \in Path^\sharp_{fin}$ by $\alpha_{\mathrm{Path}} : Path_{fin} \to Path^\sharp_{fin}$. ■

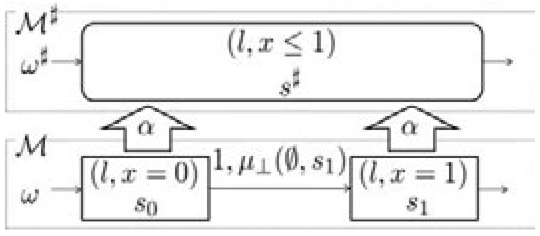

Figure 3. Correspondence relation of paths

We show an example of correspondence relation of an abstract path and a concrete path in Figure 3. A path ω of $\mathcal{M}$ is one timed transition from s_0 to s_1, but the corresponding path $\omega^\sharp$ of $\mathcal{M}^\sharp$ does not have a timed transition when $s^\sharp = \alpha(s_0) = \alpha(s_1)$ holds true.

2) Adversary: We define a correspondence relation of adversaries by a correspondence relation of paths.

Definition 15 (Correspondence relation of adversaries). *An adversary $A^\sharp : Path^\sharp_{fin} \to Dist(2^C \times S^\sharp)$ of $\mathcal{M}^\sharp$ corresponds to an adversary $A : Path_{fin} \to \mathbb{R}^{\geq 0} \times Dist(2^C \times S)$ of $\mathcal{M}$ when the following condition is satisfied:*

For $\forall \omega \in Path^A_{fin}.\forall s \in S.\forall X \subseteq C.A(\omega) = (t, \mu) \wedge A^\sharp(\alpha_{Path}(\omega)) = \mu^\sharp,\ \ \mu(X, s) = \mu^\sharp(X, \alpha(s))$.

Also, we define a correspondence between $A \in Adv$ and $A^\sharp \in Adv^\sharp$ by $\alpha_{\mathrm{Adv}} : Adv \to Adv^\sharp$. ■

For an adversary $A^\sharp$, we define the probability $\mathrm{P}^{A^\sharp} : S^\sharp \times S^\sharp \to [0, 1]$ as follows:

$$\mathrm{P}^{A^\sharp}(s^\sharp, s^{\sharp\prime}) \overset{\mathrm{def}}{=} \begin{cases} \displaystyle\sum_{X \subseteq C} \mu^\sharp(X, s^{\sharp\prime}) \qquad \text{if } \exists \omega^\sharp \in Path^{A^\sharp}_{fin}. \\ (last(\omega^\sharp) = s^\sharp \wedge \exists \mu^\sharp.A^\sharp(\omega^\sharp) = \mu^\sharp)) \\ 0 \qquad \text{otherwise} \end{cases}$$

Moreover, for any finite path $\omega^\sharp_{fin} \in Path^{A^\sharp}_{fin}$, we define the probability $Prob^{A^\sharp}_{fin}(\omega^\sharp_{fin})$ as follows:

$$Prob^{A^\sharp}_{fin}(\omega^\sharp_{fin}) \overset{\mathrm{def}}{=} \begin{cases} 1 & \text{if } |\omega^\sharp_{fin}| = 0 \\ \displaystyle\prod_{i=0}^{|\omega^\sharp_{fin}|-1} \mathrm{P}^{A^\sharp}(\omega^\sharp_{fin}(i), \omega^\sharp_{fin}(i+1)) & \text{otherwise} \end{cases}$$

By Definition 14 and Definition 15, the following theorem holds true:

Theorem 3 (Correctness of paths).
For any $\omega \in Path^A_{fin}$ and $\mathbf{\Psi}$, $\alpha_{Path}(\omega) = \omega^\sharp \in Path^{A^\sharp}_{fin}$ and $\alpha_{\mathrm{Adv}}(A) = A^\sharp \in Adv^\sharp$ exist such that $Prob^A_{fin}(\omega) = Prob^{A^\sharp}_{fin}(\omega^\sharp)$. ■

Theorem 4 (Abstraction of a path).
For $\mathcal{M}$, we denote a set $Path^A_{fin}$ of paths induced by an adversary A. For two paths $\omega_1, \omega_2 \in Path^A_{fin}$, the following holds true:

$$\alpha_{Path}(\omega_1) \neq \alpha_{Path}(\omega_2)$$

■

3) Candidate of counterexamples: We define a candidate of counterexamples by correspondence relations of paths and adversaries.

Definition 16 (Candidate of counterexamples).
For G and a reachability problem (λ, L_e), a candidate of counterexamples of $\mathcal{M}^\sharp$ is a tuple of an adversary $A^\sharp$ and a set $\Omega^\sharp$ of paths such that

$$\Sigma_{\omega^\sharp \in \{\omega'^\sharp \in Path^{A^\sharp}_{fin} | last(\omega'^\sharp) \in S^\sharp_e\}} Prob^{A^\sharp}_{fin}(\omega^\sharp) > \lambda$$

, where $S^\sharp_e \subseteq S^\sharp$ is a set of states of $\mathcal{M}^\sharp$, which include $l_e \in L_e$. A counterexample (A, Ω) corresponds to a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ when the following relation is satisfied:

$$\alpha_{Adv}(A) = A^\sharp \wedge \forall \omega \in \Omega.\exists \omega^\sharp \in \Omega^\sharp.\alpha_{Path}(\omega) = \omega^\sharp$$

■

We denote

$$\Sigma_{\omega^\sharp \in \{\omega'^\sharp \in Path^{A^\sharp}_{fin} | last(\omega'^\sharp) \in S^\sharp_e\}} Prob^{A^\sharp}_{fin}(\omega^\sharp)$$

by

$$Prob^{A^\sharp}_{fin}(S^\sharp_e).$$

Here we show two theorems about a counterexample and a candidate of counterexamples.

Theorem 5 (Sum total of reachability probability).
When an adversary of $\mathcal{M}$ corresponds to an adversary $A^\sharp$ of $\mathcal{M}^\sharp$, the following holds true;

$$Prob^A_{fin}(S_e) \leq Prob^{A^\sharp}_{fin}(S^\sharp_e)$$

, where $Prob^A_{fin}(S_e)$ is the probability of a set of paths reaching a set of target states S_e for an adversary A. ■

Theorem 5 shows that the maximum reachability probability of $\mathcal{M}$ turns into below the maximum reachability probability on an abstract model.

Theorem 6 (Existence of a counterexample).
If a counterexample (A, Ω) of $\mathcal{M}$ exists, a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ always exists. ■

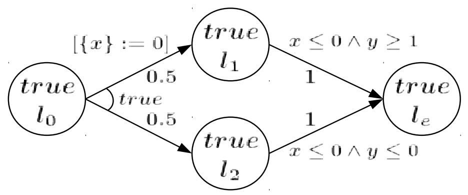

Figure 4. Probabilistic Timed Automaton G_2

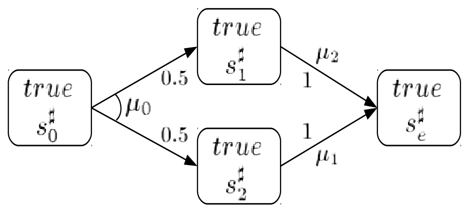

Figure 5. Abstract Model $\mathcal{M}_0^\sharp$

This theorem shows that if there is not a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ of $\mathcal{M}^\sharp$, there is a counterexample (A, Ω) of $\mathcal{M}$. Therefore, if there is not a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ of $\mathcal{M}^\sharp$, we can say "yes" for a probabilistic reachability problem.

D. Simultaneous execution

We mention simultaneous executions of a probabilistic timed automaton. By Theorem 6, we mention if a counterexample (A, Ω) of $\mathcal{M}$ exists, a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ always exists. But if a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ exists, a counterexample (A, Ω) of $\mathcal{M}$ does not necessarily exist. Therefore, in Counterexample Analysis, we propose how to compute a counterexample (A, Ω) from a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$. But it is insufficient just to consider it as a counterexample in quest of a set Ω of the paths corresponding to each path of $\Omega^\sharp$, because simultaneous executions of a set Ω may be impossible. We show this fact by both a probabilistic timed automaton G_2 in Figure 4 and an abstract model $\mathcal{M}_0^\sharp$ using abstraction predicates $\mathbf{\Psi}$ $(\Psi^{l_0} = \{\text{true}\}, \Psi^{l_1} = \{\text{true}\}, \Psi^{l_2} = \{\text{true}\}, \Psi^{l_e} = \{\text{true}\})$ in Figure 5, where a reachability problem is $(0.5, \{l_e\})$.

We compute a set $\Omega^\sharp$ of paths as candidates of counterexamples of $\mathcal{M}_0^\sharp$ such that $\omega_1^\sharp = s_0^\sharp \to s_1^\sharp \to s_e^\sharp$ and $\omega_2^\sharp = s_0^\sharp \to s_2^\sharp \to s_e^\sharp$. Two paths of $\mathcal{M}$ corresponding to the candidates of two counterexamples exist. For a path ω_1 corresponding to $\omega_1^\sharp$, l_e cannot be reached if the timed transition of 1 or more unit time is not carried out in l_0. On the other hand, for a path ω_2 corresponding to $\omega_2^\sharp$, l_e cannot be reached if the timed transition is carried out in l_0. In ω_1, a timed transition is carried out by an initial state (l_0, ν_0), and a discrete transition is carried out in ω_2. That is, the behaviors chosen, respectively differ and they are behaving by different adversaries. Therefore, it is necessary to investigate not only Ω from $(A^\sharp, \Omega^\sharp)$ but the correspondence of A. In this paper, by Counterexample Analysis, we resolve this problem.

IV. Probabilistic Timed CEGAR

The approach which applies predicate abstraction and refinement by a counterexample to a verification is a framework of CEGAR(CounterExample-Guided Abstraction Refinement) [9]. The flow of the verification of Probabilistic Timed CEGAR is explained according to Figure 6.

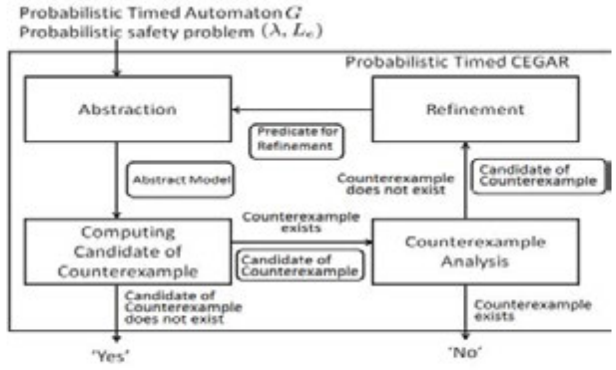
Probabilistic Timed Automaton G
Probabilistic safety problem (λ, L_e)
Probabilistic Timed CEGAR
Abstraction
Refinement
Predicate for Refinement
Abstract Model
Counterexample does not exist
Candidate of Counterexample
Counterexample exists
Computing Candidate of Counterexample
Candidate of Counterexample
Counterexample Analysis
Candidate of Counterexample does not exist
Counterexample exists
'Yes'
'No'

Figure 6. Verification by Probabilistic Timed CEGAR

1) Abstraction: We compute an abstract model $\mathcal{M}^\sharp$ by a set $\mathbf{\Psi}$ of predicates. We start computing $\mathcal{M}^\sharp$ by a set $\mathbf{\Psi}$ of predicates such that $\forall l \in L. \Psi^l = \{\text{true}\}$.
2) Computing Candidates of Counterexamples: We compute a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ of $\mathcal{M}_{\mathbf{\Psi}}^\sharp$. If there is no candidate of a counterexample, we output "yes", and then finish the verification.
3) Counterexample Analysis: We decide whether a real counterexample (A, Ω) corresponding to a candidate $(A^\sharp, \Omega^\sharp)$ of a counterexample exists. If a real counterexample exists, we output "no", and then finish the verification. If a real counterexample does not exist, $(A^\sharp, \Omega^\sharp)$ does not exist in the concrete model. The counterexample is a spurious counterexample.
4) Refinement: If it is a spurious counterexample as a result of Counterexample Analysis, it is the spurious counterexample. The new predicate $\mathbf{\Psi}$ for removing $(A^\sharp, \Omega^\sharp)$ from $\mathcal{M}^\sharp$ is derived.
5) return to 1).

Finally by repeating these loops, Probabilistic Timed CEGAR system judges "yes"or "no" to a reachability problem.

The following techniques are needed in order to realize the verification by this Probabilistic Timed CEGAR.

- The technique of deriving a candidate $(A^\sharp, \Omega^\sharp)$ of a counterexample from $\mathcal{M}^\sharp$.

- The counterexample analysis technique to compute whether a counterexample (A, Ω) corresponding to a candidate of a counterexample $(A^\sharp, \Omega^\sharp)$ exists.
- The refinement technique which derives predicate Ψ which removes $(A^\sharp, \Omega^\sharp)$ from $\mathcal{M}^\sharp$.

Here, we mention the theorem of Probabilistic Timed CEGAR.

Theorem 7 (Validity of Probabilistic Timed CEGAR). *The verification result computed by Probabilistic Timed CEGAR terminates and is valid.*

Proof: (1)First we will show that the loop of Probabilistic Timed CEGAR terminates. When we build an abstract model $\mathcal{M}^\sharp$ by a basis, $\mathcal{M}^\sharp$ is a region graph [3]. In Probabilistic Timed CEGAR, a set of abstraction predicates is a subset of basis. Since the loop of Probabilistic Timed CEGAR is the technique of the ability to add new predicates in basis certainly each time, the loop of Probabilistic Timed CEGAR can be said after certainly ending by the number of times of limited.

(2)Next we will show that Probabilistic Timed CEGAR is valid. When Probabilistic Timed CEGAR system outputs "yes" , it is a verification result which is in agreement with the verification result of $\mathcal{M}$ from theorem 6. When Probabilistic Timed CEGAR system outputs "no", the candidate of a counterexample is computed, and the counterexample on $\mathcal{M}$ is derived, and $\mathcal{M}^\sharp$ is refined. If the above is repeated until Probabilistic Timed CEGAR system outputs 'yes", while $\mathcal{M}^\sharp$ overapproximates a region graph [3], $\mathcal{M}^\sharp$ will be converged on a region graph. Therefore, the verification result of $\mathcal{M}$ is equal to the verification result of Probabilistic Timed CEGAR. In addition, in the case of being the worst, $\mathcal{M}^\sharp$ becomes a region graph, and the verification result of $\mathcal{M}$ is equal to the verification result of Probabilistic Timed CEGAR. ∎

In this paper, both Counterexample Analysis and Refinement are omitted on account of space.

V. Experiments

A. Outline of experiments

We experiment by implementing Probabilistic Timed CEGAR verifier by Java SE 1.7.0.09, and compare with existing techniques about the number of states. Experiment environment is Windows 7 Home Premium SP1 on Intel Core i3-2120T 2.60GHz and 3980MB. The source code by Java is about 2000 lines. Since all the zones are convex zones, zones are implemented by DBM [16]. Probabilistic Time CEGAR uses the technique of forward analysis in Counterexample Analysis. When expressing a zone like DBM using the maximum constant which appears in a model, it is known that forward analysis cannot be conducted correctly [18]. If it is a model which fills any one or more of the followings, it is also known that forward analysis can be conducted correctly [18].

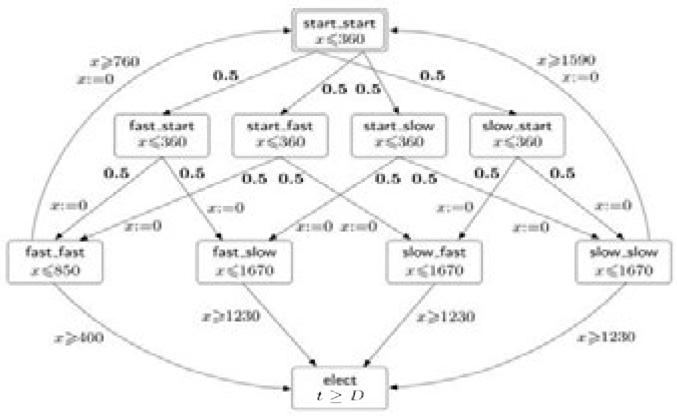

Figure 7. FireWire Root Contention Protocol

- A diagonal-free model which uses only clock constraints of the form $x \sim d$, $\sim \in \{<, >, \leq, \geq, =\}$.
- A model which uses no more than three clocks.

So, we will verify models which use no more than three clocks in this experiment. In addition, we conducted two experiments such as FireWire Root Contention Protocol and CSMA/CD.

B. FireWire Root Contention Protocol

The IEEE1394 FireWire root contention protocol concerns the election of a leader between two contending nodes of a network. The protocol consists of a number of rounds in which each of the contending nodes flips a coin; given the result of the coin flip, a node may decide to wait for a short amount of time or a long amount of time. After this amount of time has elapsed, a node then checks to see if the other node has already deferred, and declares itself to be the leader if so; otherwise, this node defers. The timing constraints are derived from those given in the standard when the communication delay is 360ns. The properties we consider concern the minimum probability to elect a leader with and without a deadline [8], [23].

The model is shown in Figure 7. In the model, an invariant condition $t \geq D$ is added to the location $elect$ which shows that leader election was completed. Here, t is the newly added clock variable, in order to express the time which leader election took. Moreover, D is a constant showing a deadline. Therefore, reaching to $elect$ means that the time more than a deadline may be required and leader election may be executed. Such change is added because this technique is aimed only at the reachability over a location in Symbolic Model Checking or PRISM while a deadline can be specified as property to verify. In addition, in order to perform the comparison with existing techniques, a deadline prepares the models from 2000 to 60000. A reachability

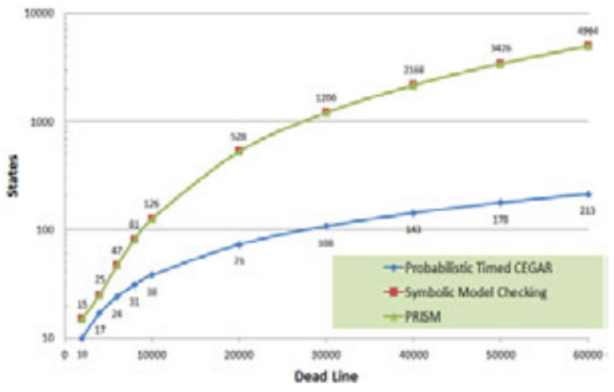

Figure 8. Comparison about FireWire Root Contention Protocol

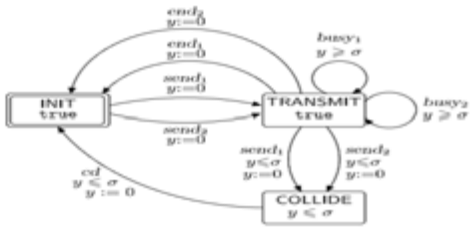

Figure 9. CSMA/CD: The Medium

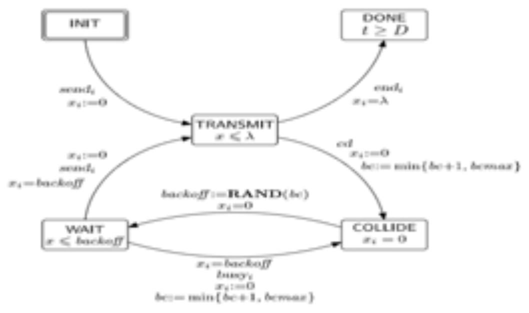

Figure 10. CSMA/CD: The Stations

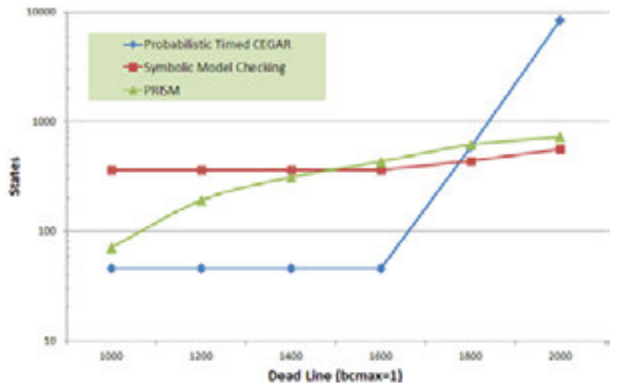

Figure 11. CSMA/CD(bcmax=1)

problem equivalent to the property verified by both Symbolic Model Checking and PRISM is defined as follows.

$(\lambda, L_e) = (0, \{elect\})$, that is $\forall A.Prob^A(S_e) \leq 0. S_e = \{(elect, \nu) \in S\}$. The case where this verification result is "no" – namely, when $\exists A.Prob^A(S_e) > 0. S_e = \{(elect, \nu) \in S\}$ is truth, it turns out that leader election may take the time more than a deadline.

A comparative result with existing techniques is shown in Figure 8. For any the Dead Line, verifications are finished with the number of states smaller than existing techniques. Moreover, the number of states shows that the number of states is reduced more with the increase in Dead Line, although there are few effects while Dead Line is small.

C. CSMA/CD

The CSMA/CD protocol is designed for networks with a single channel and specifies the behavior of stations with the aim of minimizing simultaneous use of the channel (data collision). The basic structure of the protocol is as follows: when a station has data to send, it listens to the medium, after which, if the medium was free (no other station is transmitting), the station starts to send its data. On the other hand, if the medium was sensed busy, the station waits a random amount of time, based on the number of failed transmissions of the packet, and then repeats this process [8], [23].

In this experiment, we verify whether the time more than a deadline may be taken for each client to complete communication. The automaton used in this experiment is shown in Figure 9 and Figure 10. Figure 9 expresses the action of the transmission path of communication between clients, and Figure 10 expresses the action of the client which communicates. It assumes communicating by connecting between two clients by a transmission path as an actual model used for this experiment, and becomes the model which carried out parallel composition of these three automata. $bcmax$ used here is a maximum of the number of times of the resending processing performed when collision occurs. For the model constituted in this way, the reachability problem used as verification property is defined as follows

$(\lambda, L_e) = (0, \{DONE\})$. Namely, we verify whether $\forall A.Prob^A(S_e) \leq 0. S_e = \{(DONE, \nu) \in S\}$ is satisfied. The case where this verification result is "no" – namely When $\exists A.Prob^A(S_e) > 0. S_e = \{(DONE, \nu) \in S\}$ holds true, it turns out that the time more than a deadline may be taken for each client to complete communication.

A comparative result with existing techniques is shown in Figure 11 and Figure 12. Although a verification is finished with the number of states smaller than existing techniques for all the Dead Line at the time of $bcmax = 2$, when $bcmax = 1$ and Dead Line are 1800 and 2000, the number of states is large rather than existing techniques. The state was subdivided when seeing the log of this experiment later on,

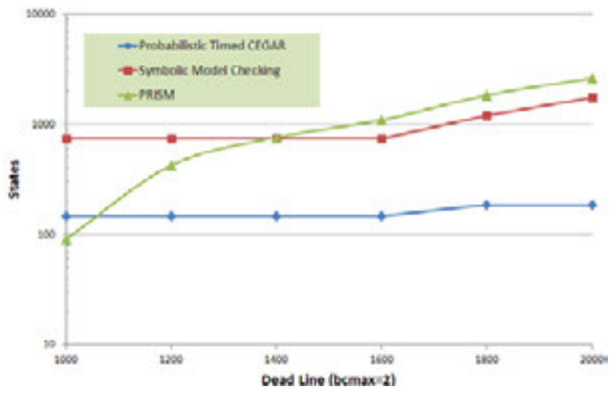

Figure 12. CSMA/CD(bcmax=2)

and many predicates were added by the location containing a loop. The number of loops was two in the model of FireWire Root Contention Protocol. Since the number of loops was 16 in the model of CSMA/CD, the number of times of a CEGAR loop increase, and then the number of states will increase.

VI. Conclusion

We have implemented Probabilistic Timed CEGAR and it was shown in the specific model as compared with existing techniques that a verification with the smaller number of states is possible. We will develop model checking for PTCTL by extending Probabilistic Timed CEGAR, and also will develop new technique in order to avoid the state space explosion.

References

[1] S. Graf, H. Saïdi, ”Construction of Abstract State Graphs with PVS”, LNCS 1254, pp.72-83, 1997.

[2] E. M. Clarke, O. Grumberg, D. Peled, ”Model Checking”, The MIT Press, 1999.

[3] M. Kwiatkowska, G. Norman, R. Segala, J. Sproston, ”Automatic verification of real-time systems with discrete probability distributions”, TCS 282(1), pp.101-150, 2002.

[4] R. Alur, ”Timed Automata”, LNCS 1633, pp.8-22, 1999.

[5] R. Alur, D.L. Dill, ”A theory of timed automata”, TCS 126(2), pp.183-235, 1994.

[6] M. Kwiatkowska, G. Norman, J. Sproston, ”Symbolic Model Checking of Probabilistic Timed Automata Using Backwards Reachability”, CSR-00-01, University of Birmingham, School of Computer Science, 2000.

[7] M. Kwiatkowska, G. Norman, J. Sproston, ”Symbolic Computation of Maximal Probabilistic Reachability”, LNCS 2154, pp.169-183, 2001.

[8] M. Kwiatkowska, G. Norman, J. Sproston, F. Wang, ”Symbolic model checking for probabilistic timed automata”, Information and Computation 205(7), pp.1027-1077, 2007.

[9] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, H. Veith, ”Counterexample-Guided Abstraction Refinement”, LNCS 1855, pp.154-169, 2000.

[10] M. O. Moller, H. Rues, M. Sorea, ”Predicate Abstraction for Dense Real-Time Systems”, ENTCS 65(6), pp.218-237, 2002.

[11] T. Han, J. P. Katoen, ”Counterexamples in probabilistic model checking”, LNCS 4424, pp.72-86, 2007.

[12] T.A. Henzinger, X. Nicollin, J. Sifakis, S. Yovine, ”Symbolic Model Checking for Real-Time Systems”, Information and Computation 111(2), pp.394-406, 1994.

[13] A. Bianco, Luca de Alfaro, ”Model checking of probabilistic and nondeterministic systems”, LNCS 1026, pp.499-513, 1995.

[14] H. Hermanns, W. Björn, L. Zhang, ”Probabilistic CEGAR”, LNCS 5123, pp.162-175, 2008.

[15] L.M.F. Fioriti, H. Hermanns, ”Heuristics for Probabilistic Timed Automata with Abstraction Refinement”, LNCS 7201, pp.151-165, 2012.

[16] J. Bengtsson, Wang Yi, ”Timed Automata: Semantics, Algorithms and Tools”, LNCS 3098, pp.87-124, 2004.

[17] A. Rybalchenko, V. Sofronie-Stokkermans, ”Constraint solving for interpolation”, J. Symb. Comput. 45(11), pp.1212-1233, 2010.

[18] P. Bouyer, ”Untameable Timed Automata!”, LNCS 2607, pp.620-631, 2003.

[19] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, ”Automated Verification Techniques for Probabilistic Systems”, LNCS 6659, pp.53-113, 2011.

[20] J. Sproston, ”Strict Divergence for Probabilistic Timed Automata”, LNCS 5710, pp.620-636, 2009.

[21] E.M. Clarke, A. Fehnker, Z. Han, B.H. Krogh, J. Ouaknine, O. Stursberg, M. Theobald, ”Abstraction and Counterexample-Guided Refinement in Model Checking of Hybrid Systems”, International Journal of Foundations of Computer Science 14(4), pp.583-604, 2003.

[22] R. Alur, T. Dang, F. Ivancic, ”Predicate abstraction for reachability analysis of hybrid systems”, ACM Transactions in Embedded Computing Systems 5(1), pp.152-199, 2006.

[23] PRISM, <http://www.prismmodelchecker.org>