A Model Checker for Hierarchical Probabilistic Real-time Systems

Songzheng Song[1] Jun Sun[2] Yang Liu[1] Jin Song Dong[1]

[1]National University of Singapore

[2]Singapore University of Technology and Design

CAV 2012
July 11, 2012

Motivation

Model checking real-life systems is usually difficult since such systems usually have the following characteristics:

- quantitative timing factors
- unreliable/random environment
- complex data operations
- hierarchical control flows

Existing Approach

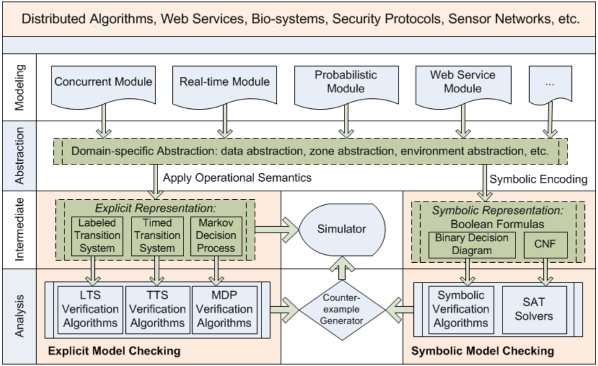
Probabilistic Timed Automata (PTA) is widely used to specify systems having stochastic and real-time characteristics.

Existing Approach

Probabilistic Timed Automata (PTA) is widely used to specify systems having stochastic and real-time characteristics.

1. PTA models often have a simple structure, e.g. a network of automata without hierarchy;
2. Verifying PTA models is not very efficient.

Our Approach: PAT



We propose **PRTS** for probabilistic real-time systems and it has been integrated into our framework **PAT**.

Based on C. A. R. Hoare's CSP

$P = Stop$	– in-action
$\mid\ Skip$	– termination
$\mid\ e \rightarrow P$	– event prefixing
$\mid\ a\{program\} \rightarrow P$	– data operation prefixing
$\mid\ [b]P$	– guard condition
$\mid\ \textbf{if}\ (b)\ \{P\}\ \textbf{else}\ \{Q\}$	– conditional choice
$\mid\ P \Box Q$	– external choice
$\mid\ P \sqcap Q$	– internal choice
$\mid\ P \setminus X$	– hiding
$\mid\ P;\ Q$	– sequential composition
$\mid\ P \parallel Q$	– parallel composition
$\mid\ Q$	– process referencing

Based on C. A. R. Hoare's CSP

$$
\begin{array}{rll}
P = & Wait[d] & \text{– delay} \\
\mid & P \ timeout[d] \ Q & \text{– timeout} \\
\mid & P \ interrupt[d] \ Q & \text{– timed interrupt} \\
\mid & P \ within[d] & \text{– timed responsiveness} \\
\mid & P \ deadline[d] & \text{– deadline}
\end{array}
$$

Based on C. A. R. Hoare's CSP

$$
\begin{array}{rll}
P = & Wait[d] & \text{– delay} \\
| & P\ timeout[d]\ Q & \text{– timeout} \\
| & P\ interrupt[d]\ Q & \text{– timed interrupt} \\
| & P\ within[d] & \text{– timed responsiveness} \\
| & P\ deadline[d] & \text{– deadline}
\end{array}
$$

$$P = pcase\{pr_0 : P_0;\ pr_1 : P_1;\ \cdots;\ pr_k : P_k\}$$

pr_i is defined as a positive integer. It means with probability $\frac{pr_i}{pr_0+pr_1+\cdots+pr_k}$, P behaves as P_i.

Note here we assume** pcase **must happen immediately when it is enabled.

Example

```
1. P = (pcase{  1 : Q
2.              3 : R }) timeout[3] S;
3. Q = Wait[2];
4. R = Wait[5];
5. S = exit -> P;
//assertions:
6. #assert P deadlockfree;
```

- The semantic model of our language is Markov Decision Process (MDP).

- The semantic model of our language is Markov Decision Process (MDP).
- Since PRTS model has a dense-time semantics, the underlying MDP has infinite states.

- The semantic model of our language is Markov Decision Process (MDP).
- Since PRTS model has a dense-time semantics, the underlying MDP has infinite states.

$$(\sigma,\ \mathit{Wait}[1]) \stackrel{0.1}{\rightarrow} (\sigma,\ \mathit{Wait}[0.9]) \stackrel{0.01}{\rightarrow} (\sigma,\ \mathit{Wait}[0.89]) \stackrel{0.001}{\rightarrow} \ldots$$

- The semantic model of our language is Markov Decision Process (MDP).
- Since PRTS model has a dense-time semantics, the underlying MDP has infinite states.

$$(\sigma,\ \mathit{Wait}[1]) \stackrel{0.1}{\rightarrow} (\sigma,\ \mathit{Wait}[0.9]) \stackrel{0.01}{\rightarrow} (\sigma,\ \mathit{Wait}[0.89]) \stackrel{0.001}{\rightarrow} \ldots$$

Abstraction is required!

Dynamic Zone Abstraction

The first step of abstraction is to associate timed process constructs with implicit **clocks**.

- $P\ timeout[d]\ Q \rightarrow P\ timeout[d]_c\ Q$
- Constraint over clock : $c \leq 5$ represents any process $P\ timeout[d']\ Q$ with $d' \leq 5$

Dynamic Zone Abstraction

The first step of abstraction is to associate timed process constructs with implicit **clocks**.

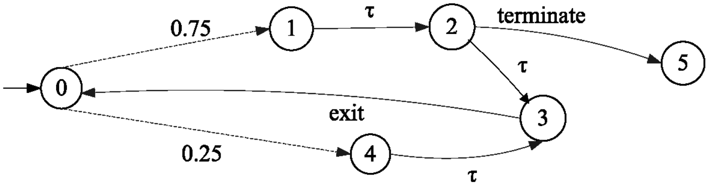
- $P\ timeout[d]\ Q \rightarrow P\ timeout[d]_c\ Q$
- Constraint over clock : $c \leq 5$ represents any process $P\ timeout[d']\ Q$ with $d' \leq 5$

A **zone** D is the conjunction of multiple primitive constraints over a set of clocks, which is calculated by Difference Bound Matrix(DBM).

- $c \sim d$ or $c_i - c_j \sim d$ where c, c_i, c_j are values of clocks and d is a constant integer. $\sim$ represents $\geq, \leq, =$

Example Revisit

```
1. P = (pcase{  1 : Q
2.              3 : R }) timeout[3] S;
3. Q = Wait[2];
4. R = Wait[5];
5. S = exit -> P;
//assertions:
6. #assert P deadlockfree;
```



Verification

Properties supported:

1. Reachability Checking
2. Reward Checking
3. LTL Checking
4. Refinement Checking

Verification

Properties supported:

1. Reachability Checking
2. Reward Checking
3. LTL Checking
4. Refinement Checking

Key difference between PTA and our approach:

- PTA's reachability checking through zone abstraction can just supplies an upper or lower bound;
- We can get a precise result after zone abstraction.

Benchmark Systems Compared with PRISM

System	Result	PAT		PRISM		
		States	Time(s)	States	Iterations	Time(s)
FA(10K)	0.94727	1352	0.15	1065	19	1.98
FA(20K)	0.99849	5030	0.13	8663	34	65.08
FA(30K)	0.99994	11023	0.45	34233	45	575.03
FA(300K)	$>$0.99999	726407	30.74	-	-	-
ZC(100)	0.49934	404	0.15	135	0	0.28
ZC(300)	0.01291	4813	0.65	2129	26	2.73
ZC(500)	0.00027	12840	2.39	10484	44	63.19
ZC(700)	1E-5	24058	5.78	31717	60	427.70

One is the *firewire abstraction* (*FA*) for IEEE 1394 FireWire root contention protocol and the other is *zeroconf* (*ZC*) for Zeroconf network configuration protocol.

Conclusion

1. Modeling language PRTS is proposed for hierarchical probabilistic real-time systems.
2. Zone abstraction is used in order to apply probabilistic model checking techniques. Evaluations demonstrate the efficiency of our approach.
3. Model checking framework PAT is extended to support PRTS.

Statistics of PAT:

- 5 years history
- 30+ researchers
- 1 million LOC
- 2000+ downloads

Our tool can be freely downloaded at

`http://www.patroot.com`

Statistics of PAT:

- 5 years history
- 30+ researchers
- 1 million LOC
- 2000+ downloads

Our tool can be freely downloaded at
`http://www.patroot.com`

THANK YOU!

Definition (Markov Decision Process)

An MDP is a tuple $\mathcal{D} = (S, init, Act, Pr)$ where

- S is a set of states;
- $init \in S$ is the initial state;
- Act is a set of actions and Act_τ is $Act \cup \tau$;
- $Pr : S \times (Act_\tau \cup \mathbb{R}_+) \times Distr(S)$ is a transition relation.

A Markov Chain can be defined given an MDP $\mathcal{D}$ and a scheduler δ, which is denoted as $\mathcal{D}^\delta$.

A *path* of $\mathcal{D}^\delta$ is defined as $\omega = s_0 \xrightarrow{x_0} s_1 \xrightarrow{x_1} s_2 \xrightarrow{x_2} \ldots$

Given a property ϕ:

$$\mathcal{P}_{\mathcal{D}}^{max}(\phi) = \sup_{\delta} \mathcal{P}_{\mathcal{D}}(\{\pi \in \textit{paths}(\mathcal{D}^{\delta}) \mid \pi \text{ satisfies } \phi\})$$

$$\mathcal{P}_{\mathcal{D}}^{min}(\phi) = \inf_{\delta} \mathcal{P}_{\mathcal{D}}(\{\pi \in \textit{paths}(\mathcal{D}^{\delta}) \mid \pi \text{ satisfies } \phi\})$$

Definition (Concrete System Configuration)

A concrete system configuration is a tuple $s = (\sigma, P)$ where σ is a variable valuation and P is a process.

Definition (Concrete System Configuration)

A concrete system configuration is a tuple $s = (\sigma, P)$ where σ is a variable valuation and P is a process.

The probabilistic transition relation of a model's MDP semantics is defined by a set of firing rules with every process construct.

- $Wait[d]$
- *pcase*

$Wait[d]$

$$\frac{\epsilon \leq d}{(\sigma, Wait[d]) \xrightarrow{\epsilon} (\sigma, Wait[d - \epsilon])} \quad [\ wait_1\ ]$$

$$\frac{}{(\sigma, Wait[0]) \xrightarrow{\tau} (\sigma, Skip)} \quad [\ wait_2\ ]$$

pcase

$$\frac{}{(\sigma, pcase\ \{pr_0 : P_0;\ pr_1 : P_1;\ \cdots;\ pr_k : P_k\}) \xrightarrow{\tau} \mu}\ [\ pcase\ ]$$

$$\mu((\sigma, P_i)) = \frac{pr_i}{pr_0 + pr_1 + \cdots + pr_k} \text{ for all } i \in [0,\ k]$$

pcase

$$\frac{}{(\sigma, pcase\ \{pr_0 : P_0;\ pr_1 : P_1;\ \cdots;\ pr_k : P_k\}) \xrightarrow{\tau} \mu}\ [\ pcase\ ]$$

$$\mu((\sigma, P_i)) = \frac{pr_i}{pr_0 + pr_1 + \cdots + pr_k} \text{ for all } i \in [0,\ k]$$

pcase transitions are not time-consuming!

Abstract Configurations

Definition (Abstract System Configuration)

Given a concrete system configuration (σ, P), the corresponding abstract system configuration is a triple (σ, P_T, D) such that P_T is a process obtained by associating P with a set of clocks; and D is a zone over the clocks.

Abstract Configurations

Definition (Abstract System Configuration)

Given a concrete system configuration (σ, P), the corresponding abstract system configuration is a triple (σ, P_T, D) such that P_T is a process obtained by associating P with a set of clocks; and D is a zone over the clocks.

Abstract firing rules are defined in order to get the abstract MDP. $Wait[d]$ and *pcase* are listed as examples.

$Wait[d]$

$$\frac{}{(\sigma, Wait[d]_c, D) \overset{\tau}{\rightsquigarrow} (\sigma, Skip, D^{\uparrow} \wedge c = d)} \quad [\ await\ ]$$

- $D^{\uparrow}$ denotes the zone obtained by delaying arbitrary amount of time. e.g. $(c \leq 5)^{\uparrow}$ is $c \leq \infty$.

pcase

$$\frac{}{(\sigma, \mathit{pcase}\ \{pr_0 : P_0;\ pr_1 : P_1;\ \cdots;\ pr_k : P_k\}, D) \overset{\tau}{\rightsquigarrow} \mu}$$

$\mu((\sigma, P_i, D)) = \frac{pr_i}{pr_0 + pr_1 + \cdots + pr_k}$ for $i \in [0, k]$; **zone is unchanged.**

Theorem 1

Theorem

$\mathcal{D}^a_M$ is finite for any model M. □

1. Variable valuations are finite[**by assumption**].
2. Process expressions are finite[**by assumption and clock reuse**].
3. Zones are finite.

J. Bengtsson and Y. Wang.
Timed Automata: Semantics, Algorithms and Tools.
In *Lectures on Concurrency and Petri Nets*, pages 87-124, 2003.

Definition

A *probabilistic time-abstract bi-simulation relation* between a DTMC $\mathcal{C} = (S_c, init_c, Act, Pr_c)$ and an abstract DTMC $\mathcal{C}_a = (S_a, init_a, Act, Pr_a)$ is a relation $\mathcal{R} \subseteq S_c \times S_a$ satisfying the following condition.

C1: If $(s_c, s_a) \in \mathcal{R}$, then s_c and s_a have the same variable valuation.

C2: If $(s_c, s_a) \in \mathcal{R}$ and $(s_c, (\epsilon, e, p), s'_c) \in Pr_c$ for some $\epsilon \geq 0$, $e \in Act_\tau$ and $p \in [0, 1]$, then there exists s'_a such that $(s_a, (e, p), s'_a) \in Pr_a$ and $(s'_c, s'_a) \in \mathcal{R}$;

C3: If $(s_c, s_a) \in \mathcal{R}$ and $(s_a, (e, p), s'_a) \in Pr_a$ for some $e \in Act_\tau$ and $p \in [0, 1]$, then there exists some $\epsilon \geq 0$ and s'_c such that $(s_c, (\epsilon, e, p), s'_c) \in Pr_c$ and $(s'_c, s'_a) \in \mathcal{R}$;

Theorem 2

Theorem

$\mathcal{P}^{max}_{\mathcal{D}^a_M}(\phi) = \mathcal{P}^{max}_{\mathcal{D}_M}(\phi)$ and $\mathcal{P}^{min}_{\mathcal{D}^a_M}(\phi) = \mathcal{P}^{min}_{\mathcal{D}_M}(\phi)$. □

1. For any scheduler δ in $\mathcal{D}^a_M$, there is a scheduler ξ in $\mathcal{D}_M$ such that $(\mathcal{D}^a_M)^\delta$ and $(\mathcal{D}_M)^\xi$ are bisimilar Markov Chains.
2. For any scheduler η in $\mathcal{D}_M$, there is a scheduler ϑ in $\mathcal{D}^a_M$ such that $(\mathcal{D}_M)^\eta$ and $(\mathcal{D}^a_M)^\vartheta$ are bisimilar Markov Chains.

pcase transitions are not time-consuming!