PRTS: An Approach for Model Checking Probabilistic Real-time Hierarchical Systems

Songzheng Song

Software Engineering Lab, NUS

IWESSS 2012
February 23, 2012
Outline

1. Motivation
2. Language Syntax of PRTS
3. Operational Semantics
   - Concrete Configurations
   - Abstraction
4. Verification
5. Evaluation
   - Multi-lift System
   - Benchmark Systems Compared with PRISM
6. Conclusion
Model checking real-life systems is usually difficult.
Motivation

Model checking real-life systems is usually difficult.

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

- Serving time/responding time
- Random user behaviors
- Task assign algorithm
- lifts/users/buttons...
Multi-lift System

Scenario: one user presses the lift button, but one lift traveling on the same direction passes by without serving him/her!
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

1. Design an expressive modeling language supporting features like real-time, hierarchy, concurrency, data structures as well as probability.

2. Build a model checker for this language in order to analyze the behavior of such systems.

3. Verify widely used properties such as reachability checking and LTL checking.
Our Approach

1. Design an expressive modeling language supporting features like real-time, hierarchy, concurrency, data structures as well as probability.

2. Build a model checker for this language in order to analyze the behavior of such systems.

3. Verify widely used properties such as reachability checking and LTL checking.

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

http://www.patroot.com
Outline

1. Motivation
2. Language Syntax of PRTS
3. Operational Semantics
   - Concrete Configurations
   - Abstraction
4. Verification
5. Evaluation
   - Multi-lift System
   - Benchmark Systems Compared with PRISM
6. Conclusion
Based on C. A. R. Hoare’s CSP

\[ P = \text{Stop} \]
\[ \quad \text{Skip} \]
\[ \quad e \rightarrow P \]
\[ \quad a\{\text{program}\} \rightarrow P \]
\[ \quad [b]P \]
\[ \quad \text{if} \ (b) \ \{P\} \ \text{else} \ \{Q\} \]
\[ \quad P \sqcap Q \]
\[ \quad P \sqcap Q \]
\[ \quad P \setminus X \]
\[ \quad P; \ Q \]
\[ \quad P \parallel Q \]
\[ \quad Q \]

- in-action
- termination
- event prefixing
- data operation prefixing
- guard condition
- conditional choice
- external choice
- internal choice
- hiding
- sequential composition
- parallel composition
- process referencing
Based on C. A. R. Hoare’s CSP

\[ P = \text{Wait}[d] \quad \text{– delay} \]
\[ | \quad P \text{ timeout}[d] Q \quad \text{– timeout} \]
\[ | \quad P \text{ interrupt}[d] Q \quad \text{– timed interrupt} \]
\[ | \quad P \text{ within}[d] \quad \text{– timed responsiveness} \]
\[ | \quad P \text{ deadline}[d] \quad \text{– deadline} \]

\[ d \text{ is a non-negative integer.} \]
Based on C. A. R. Hoare’s CSP

\[
P = \text{Wait}[d] \quad \text{– delay}
\]

\[
| \quad P \text{ timeout}[d] Q \quad \text{– timeout}
\]

\[
| \quad P \text{ interrupt}[d] Q \quad \text{– timed interrupt}
\]

\[
| \quad P \text{ within}[d] \quad \text{– timed responsiveness}
\]

\[
| \quad P \text{ deadline}[d] \quad \text{– deadline}
\]

\(d\) is a non-negative integer.

\[
P = \text{pcase}\{pr_0 : P_0; \ pr_1 : P_1; \cdots ; \ pr_k : P_k\}
\]

\(pr_i\) is defined as a positive integer. It means with probability

\[
\frac{pr_i}{pr_0 + pr_1 + \cdots + pr_k}, \ P \text{ behaves as } P_i.
\]
Multi-lift System

1. #define NoOfFloors 2;
2. #define NoOfLifts 2;
3. #import "PAT.Lib.Lift";
4. var<LiftControl> ctrl = new LiftControl(NoOfFloors, NoOfLifts);
5. Users() = pcase {
6. 1 : extreq.0.1{ctrl.Assign_External_Up_Request(0)} -> Skip
7. 1 : intreq.0.0.1{ctrl.Add_Internal_Request(0,0)} -> Skip
8. 1 : intreq.1.0.1{ctrl.Add_Internal_Request(1,0)} -> Skip
9. 1 : extreq.1.0{ctrl.Assign_External_Down_Request(1)} -> Skip
10. 1 : intreq.0.1.1{ctrl.Add_Internal_Request(0,1)} -> Skip
11. 1 : intreq.1.1.1{ctrl.Add_Internal_Request(1,1)} -> Skip
12. } within[1]; Users();
13. Lift(i, level, direction) = ...;
14. System = (||| x:{0..NoOfLifts-1} @ Lift(x, 0, 1)) ||| Users();
Multi-lift System

1. `#define NoOfFloors 2;`
2. `#define NoOfLifts 2;`
3. `#import "PAT.Lib.Lift";`
4. `var<LiftControl> ctrl = new LiftControl(NoOfFloors,NoOfLifts);`
5. `Users() = pcase {
   1 : extreq.0.1{ctrl.Assign_External_Up_Request(0)} -> Skip
   1 : intreq.0.0.1{ctrl.Add_Internal_Request(0,0)} -> Skip
   1 : intreq.1.0.1{ctrl.Add_Internal_Request(1,0)} -> Skip
   1 : extreq.1.0{ctrl.Assign_External_Down_Request(1)} -> Skip
   1 : intreq.0.1.1{ctrl.Add_Internal_Request(0,1)} -> Skip
   1 : intreq.1.1.1{ctrl.Add_Internal_Request(1,1)} -> Skip
   } within[1]; Users();
6. `Lift(i, level, direction) = ...;`
7. `System = (||| x:{0..NoOfLifts-1} @ Lift(x, 0, 1)) ||| Users();`

Property: what is the probability that a lift passes by?
Outline

1. Motivation
2. Language Syntax of PRTS
3. Operational Semantics
   - Concrete Configurations
   - Abstraction
4. Verification
5. Evaluation
   - Multi-lift System
   - Benchmark Systems Compared with PRISM
6. Conclusion

Songzheng Song, IWESSS 2012

PRTS@PAT  http://www.patroot.com
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 $Act_\tau$ is $Act \cup \tau$;
- $Pr : S \times (Act_\tau \cup \mathbb{R}_+) \times Distr(S)$ is a transition relation.
Definition (Markov Decision Process)

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

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

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

$$P_D^{\text{max}}(\phi) = \sup_\delta P_D(\{\pi \in \text{paths}(D^\delta) \mid \pi \text{ satisfies } \phi\})$$

$$P_D^{\text{min}}(\phi) = \inf_\delta P_D(\{\pi \in \text{paths}(D^\delta) \mid \pi \text{ satisfies } \phi\})$$
Definition (Concrete System Configuration)

A concrete system configuration is a tuple \( s = (\sigma, P) \) where \( \sigma \) is a variable valuation and \( P \) is a process.
Definition (Concrete System Configuration)

A concrete system configuration is a tuple \( s = (\sigma, P) \) where \( \sigma \) 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.

- \( \text{Wait}[d] \)
- \( \text{pcase} \)
\[ Wait[d] \]

\[ \epsilon \leq d \]
\[ (\sigma, Wait[d]) \xrightarrow{\epsilon} (\sigma, Wait[d - \epsilon]) \]

\[ Wait_1 \]

\[ (\sigma, Wait[0]) \xrightarrow{\tau} (\sigma, Skip) \]

\[ Wait_2 \]
Songzheng Song, IWESSS 2012

Motivation
Language Syntax of PRTS
Operational Semantics
Verification
Evaluation
Conclusion

Concrete Configurations
Abstraction

pcase

\[
\begin{align*}
(\sigma, \text{pcase}\{\text{pr}_0 : P_0; \text{pr}_1 : P_1; \cdots; \text{pr}_k : P_k\}) & \xrightarrow{\tau} \mu \\
\end{align*}
\]

\[
\mu((\sigma, P_i)) = \frac{\text{pr}_i}{\text{pr}_0 + \text{pr}_1 + \cdots + \text{pr}_k} \quad \text{for all } i \in [0, k]
\]
Motivation
Language Syntax of PRTS
Operational Semantics
Verification
Evaluation
Conclusion

Concrete Configurations
Abstraction

pcase transitions are not time-consuming!

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

\[
\mu((\sigma, P_i)) = \frac{pr_i}{pr_0 + pr_1 + \cdots + pr_k} \text{ for all } i \in [0, k]
\]
Using the firing rules, the transition relation of the MDP could be defined. However, since PRTS model has a dense-time semantics, the underlying MDP has infinite states.
Using the firing rules, the transition relation of the MDP could be defined. However, since PRTS model has a dense-time semantics, the underlying MDP has infinite states.

\[
(\sigma, \text{Wait}[1]) \xrightarrow{0.1} (\sigma, \text{Wait}[0.9]) \xrightarrow{0.01} (\sigma, \text{Wait}[0.89]) \xrightarrow{0.001} \ldots
\]
Using the firing rules, the transition relation of the MDP could be defined. However, since PRTS model has a dense-time semantics, the underlying MDP has infinite states.

\[ (\sigma, \text{Wait}[1]) \xrightarrow{0.1} (\sigma, \text{Wait}[0.9]) \xrightarrow{0.01} (\sigma, \text{Wait}[0.89]) \xrightarrow{0.001} ... \]

Abstraction is required!
Dynamic Zone Abstraction

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

- $P \text{ timeout}[d] Q \rightarrow P \text{ timeout}[d_c] Q$
- Constraint over clock: $c \leq 5$ represents any process $P \text{ timeout}[d'] Q$ with $d' \leq 5$
Dynamic Zone Abstraction

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

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

A zone $D$ is the conjunction of multiple primitive constraints over a set of clocks.

- $c \sim d$ 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 $\geq, \leq, =$
Given a concrete system configuration \((\sigma, P)\), the corresponding abstract system configuration is a triple \((\sigma, P_T, D)\) 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 \((\sigma, P)\), the corresponding abstract system configuration is a triple \((\sigma, P_T, D)\) 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**[$d$]

\[
\begin{align*}
(\sigma, \text{Wait}[d]_c, D) & \xrightarrow{\tau} (\sigma, \text{Skip}, D^{\uparrow} \land c = d) \\

\end{align*}
\]

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

_Songzheng Song, IWESSS 2012_
\[
(\sigma, \text{pcase} \{pr_0 : P_0; \ pr_1 : P_1; \cdots; \ pr_k : P_k\}, D) \sim \mu
\]

\[
\mu((\sigma, P_i, D)) = \frac{pr_i}{pr_0 + pr_1 + \cdots + pr_k} \quad \text{for } i \in [0, k]; \text{ zone is unchanged.}
\]
1. abstract model is suitable for standard probabilistic model checking techniques.
2. abstract model preserves the verification result of given property $\phi$ in concrete model.

- $M$ : PRTS model;
- $D_M$ : concrete MDP of $M$;
- $D^a_M$ : abstract MDP of $M$. 
Theorem 1

Theorem

\[ D^a_M \text{ is finite for any model } M. \]
Theorem 1

Theorem

\[ D^a_M \text{ is finite for any model } M. \]

1. Variable valuations are finite\textit{by assumption}.
2. Process expressions are finite\textit{by assumption and clock reuse}.
3. Zones are finite.

J. Bengtsson and Y. Wang.
Definition

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

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

C2: If $(s_c, s_a) \in R$ 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 $(s'_c, s'_a) \in R$;

C3: If $(s_c, s_a) \in R$ 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 $(s'_c, s'_a) \in R$;
Theorem 2

Theorem

\[ \mathcal{P}^{\text{max}}_{\mathcal{D}^a_M}(\phi) = \mathcal{P}^{\text{max}}_{\mathcal{D}_M}(\phi) \quad \text{and} \quad \mathcal{P}^{\text{min}}_{\mathcal{D}^a_M}(\phi) = \mathcal{P}^{\text{min}}_{\mathcal{D}_M}(\phi). \]
Theorem 2

\[ P_{D_M}^{\text{max}}(\phi) = P_{D_M}^{\text{max}}(\phi) \, \text{and} \, P_{D_M}^{\text{min}}(\phi) = P_{D_M}^{\text{min}}(\phi). \]

1. For any scheduler \( \delta \) in \( D_M^a \), there is a scheduler \( \xi \) in \( D_M \) such that \((D_M^a)^\delta\) and \((D_M)^\xi\) are bisimilar Markov Chains.

2. For any scheduler \( \eta \) in \( D_M \), there is a scheduler \( \vartheta \) in \( D_M^a \) such that \((D_M)^\eta\) and \((D_M^a)^\vartheta\) are bisimilar Markov Chains.

Songzheng Song, IWESSS 2012

PRTS@PAT http://www.patroot.com
Theorem 2

\[ P_{D^a_M}^{\text{max}}(\phi) = P_{D_M}^{\text{max}}(\phi) \quad \text{and} \quad P_{D^a_M}^{\text{min}}(\phi) = P_{D_M}^{\text{min}}(\phi). \]

1. For any scheduler \( \delta \) in \( D^a_M \), there is a scheduler \( \xi \) in \( D_M \) such that \( (D^a_M)^\delta \) and \( (D_M)^\xi \) are bisimilar Markov Chains.
2. For any scheduler \( \eta \) in \( D_M \), there is a scheduler \( \vartheta \) in \( D^a_M \) such that \( (D_M)^\eta \) and \( (D^a_M)^\vartheta \) are bisimilar Markov Chains.

\textit{pcase transitions are not time-consuming!}
After abstraction, we obtain the finite states system and standard probabilistic model checking are applied to solve the linear program in MDP.

C. Baier and J. Katoen.  
*Principles of Model checking.*  
Outline

1 Motivation
2 Language Syntax of PRTS
3 Operational Semantics
   - Concrete Configurations
   - Abstraction
4 Verification
5 Evaluation
   - Multi-lift System
   - Benchmark Systems Compared with PRISM
6 Conclusion
Multi-lift System

Property: one user presses the lift button, but one lift traveling on the same direction passes by without serving him/her.

<table>
<thead>
<tr>
<th>System</th>
<th>Random</th>
<th>Nearest</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Result(pmax)</td>
<td>States</td>
</tr>
<tr>
<td>lift=2; floor=2; user=2</td>
<td>0.21875</td>
<td>20120</td>
</tr>
<tr>
<td>lift=2; floor=2; user=3</td>
<td>0.47656</td>
<td>173729</td>
</tr>
<tr>
<td>lift=2; floor=2; user=4</td>
<td>0.6792</td>
<td>777923</td>
</tr>
<tr>
<td>lift=2; floor=2; user=5</td>
<td>0.81372</td>
<td>2175271</td>
</tr>
<tr>
<td>lift=2; floor=3; user=2</td>
<td>0.2551</td>
<td>72458</td>
</tr>
<tr>
<td>lift=2; floor=3; user=3</td>
<td>0.54009</td>
<td>1172800</td>
</tr>
<tr>
<td>lift=2; floor=4; user=2</td>
<td>0.27</td>
<td>170808</td>
</tr>
<tr>
<td>lift=3; floor=2; user=2</td>
<td>0.22917</td>
<td>562309</td>
</tr>
</tbody>
</table>

Songzheng Song, IWESSS 2012  PRTS@PAT  http://www.patroot.com
## Benchmark Systems Compared with PRISM

<table>
<thead>
<tr>
<th>System</th>
<th>Result</th>
<th>PAT</th>
<th>PRISM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>States</td>
<td>Time(s)</td>
</tr>
<tr>
<td>FA(10K)</td>
<td>0.94727</td>
<td>1352</td>
<td>0.15</td>
</tr>
<tr>
<td>FA(20K)</td>
<td>0.99849</td>
<td>5030</td>
<td>0.13</td>
</tr>
<tr>
<td>FA(30K)</td>
<td>0.99994</td>
<td>11023</td>
<td>0.45</td>
</tr>
<tr>
<td>FA(300K)</td>
<td>&gt;0.99999</td>
<td>726407</td>
<td>30.74</td>
</tr>
<tr>
<td>ZC(100)</td>
<td>0.49934</td>
<td>404</td>
<td>0.15</td>
</tr>
<tr>
<td>ZC(300)</td>
<td>0.01291</td>
<td>4813</td>
<td>0.65</td>
</tr>
<tr>
<td>ZC(500)</td>
<td>0.00027</td>
<td>12840</td>
<td>2.39</td>
</tr>
<tr>
<td>ZC(700)</td>
<td>1E-5</td>
<td>24058</td>
<td>5.78</td>
</tr>
</tbody>
</table>

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 checker PAT is extended to support PRTS.


References II

C. Baier and J. Katoen. 
*Principles of Model checking.* 

M. Kwiatkowska, G. Norman, R. Segala and J. Sproston. 
Automatic Verification of Real-time Systems with Discrete Probability Distributions. 

S. Schnerder. 
*Concurrent and Real-time Systems.* 
J. Ouaknine and J. Worrell.  
Timed CSP = Closed Timed Safety Automata.  
Electrical Notes Theoretical Computer Sciences, 68(2), 2002.

J. Sun, Y. Liu, J. S. Dong and X. Zhang.  
Verifying Stateful Timed CSP Using Implicit Clocks and Zone Abstraction.  
ICFEM, pages 581-600, 2009.

J. Sun, S. Song and Y. Liu  
Model Checking Hierarchical Probabilistic Systems.  
THANK YOU!