Some recent advances in probabilistic model checking

Jan Křetínský

Faculty of Informatics, Masaryk University, Brno, Czech Republic

IST Austria

PUMA
TUM, November 29, 2013
Probabilistic model checking: Why & What
Markov decision process (MDP)
Markov decision process (MDP) + Linear temporal logic (LTL)
Probabilistic model checking: How

e.g. PEPA models or text
→ PRISM lang. → MTBDD

MDP $M$

e.g. specification patterns
or text → formula

LTL $\varphi$

Non-deterministic
Büchi automaton

determinisation – Safra,…

Deterministic
Rabin automaton

Product to be analysed

tableaux (Spin), altern. automata (LTL2BA)…

$\Pr_{\text{max}}\left[ M \mid \varphi \right]$
Probabilistic model checking: How

e.g. PEPA models or text → PRISM lang. → MTBDD

e.g. specification patterns or text → formula

MDP $M$

LTL $\varphi$

tableaux (Spin), altern. automata (LTL2BA)...

Non-deterministic Büchi automaton
determinisation – Safra, ...

Deterministic Rabin automaton

Product to be analysed

MEC decomposition + evaluation

MEC collapse

reachability – lin. prog., value iteration, ...

$Pr_{\text{max}}[M \models \varphi]$
Solution I: Speed up computations!

Parallel computation

- ProbDiVinE-MC [Barnat et al.–Masaryk University Brno]
  - distributed memory
  - decompose LP
    - 1 per SCC
    - solve independently/iteratively
  - SCC decomposition on CUDA?

Abstraction refinement

- RAPTURE [D’Argenio,Jeannet,Jensen,Larsen’01]
- PRISM [Kattenbelt,Kwiatkowska,Norman,Parker’08]
- PASS [Hahn,Hermanns,Wachter,Zhang’10]
Solution II: Make the product smaller!

MC, MDP, game, ...

LTL formula

Spin, LTL2BA, Spot...

Non-deterministic Büchi automaton

Deterministic Rabin automaton

Product to be analysed

\[ \bigwedge_{i \in \{1, \ldots, n\}} GFa_i \Rightarrow GFb_i \]

<table>
<thead>
<tr>
<th>NBA</th>
<th>LTL2BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>n = 1</td>
<td>4</td>
</tr>
<tr>
<td>n = 2</td>
<td>14</td>
</tr>
<tr>
<td>n = 3</td>
<td>40</td>
</tr>
</tbody>
</table>
Solution II: Make the product smaller!

- MC, MDP, game, ...
- LTL formula
  - Spin, LTL2BA, Spot...
  - Non-deterministic Büchi automaton
  - Deterministic Rabin automaton

Product to be analysed

\[ \bigwedge_{i \in \{1, \ldots, n\}} \mathbf{GF} a_i \Rightarrow \mathbf{GF} b_i \]

<table>
<thead>
<tr>
<th>Method</th>
<th>NBA</th>
<th>DRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>LTL2BA</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ltl2dstar</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| n = 1          | 4   | 4   |
| n = 2          | 14  | > 10^4 |
| n = 3          | 40  | > 10^6 |
Solution II: Make the product smaller!

- MC, MDP, game, ...
- LTL formula
- Non-deterministic Büchi automaton
- Deterministic generalized Rabin automaton

Product to be analysed

\[ \bigwedge_{i \in \{1, \ldots, n\}} \text{GF}a_i \Rightarrow \text{GF}b_i \]

<table>
<thead>
<tr>
<th>( n )</th>
<th>NBA</th>
<th>DRA</th>
<th>DRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LTL2BA</td>
<td>ltl2dstar</td>
<td>Rabinizer</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>&gt; 10^4</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>&gt; 10^6</td>
<td>462</td>
<td></td>
</tr>
</tbody>
</table>
Solution II: Make the product smaller!

MC, MDP, game, ...

LTL formula

Spin, LTL2BA, Spot...

Non-deterministic Büchi automaton

Deterministic generalized Rabin automaton

Product to be analysed

\[ \bigwedge_{i \in \{1, \ldots, n\}} GFa_i \Rightarrow GFB_i \]

<table>
<thead>
<tr>
<th></th>
<th>NBA</th>
<th>DRA</th>
<th>DRA</th>
<th>DGRA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LTL2BA</td>
<td>ltl2dstar</td>
<td>Rabinizer</td>
<td>Rabinizer 2</td>
</tr>
<tr>
<td>(n = 1)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>(n = 2)</td>
<td>14</td>
<td>&gt; (10^4)</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td>(n = 3)</td>
<td>40</td>
<td>&gt; (10^6)</td>
<td>462</td>
<td>64</td>
</tr>
</tbody>
</table>
Solution II: Make the product smaller!

- MC, MDP, game, ...
- LTL formula
- Non-deterministic Büchi automaton
- Deterministic generalized Rabin automaton
- Product to be analysed

\[ \bigwedge_{i \in \{1, \ldots, n\}} \text{GF}a_i \Rightarrow \text{GF}b_i \]

<table>
<thead>
<tr>
<th>( n )</th>
<th>NBA</th>
<th>DRA</th>
<th>DRA</th>
<th>DGRA</th>
<th>LTL((X, F, G, U)) (\setminus) GU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>( &gt; 10^4 )</td>
<td>18</td>
<td>16</td>
<td>64</td>
</tr>
<tr>
<td>3</td>
<td>40</td>
<td>( &gt; 10^6 )</td>
<td>462</td>
<td>64</td>
<td>64</td>
</tr>
</tbody>
</table>
DRA: eager non-determinism vs. lazy determinism

\[ \varphi = Ga \lor Fb \]
Deterministic **generalized** Rabin automata

- **Büchi** set $I$ of states: visit $I$ inf. often
- **Rabin** pairs
  \[
  \bigvee_{j=1..k} (F^j, I^j):
  \]
  for some $j$ visit $F^j$ fin. often and $I^j$ inf. often
Deterministic generalized Rabin automata

Büchi set $I$ of states: visit $I$ inf. often

generalized Büchi $\bigwedge_{i=1..n} I_i$: visit each $I_i$ inf. often

Rabin pairs $\bigvee_{j=1..k} (F^j, I^j)$:

for some $j$ visit $F^j$ fin. often and $I^j$ inf. often
Example motivating DGRA

Example: atomic propositions $a$ and $b$, formula $\varphi = \text{GF}a$
Example motivating DGRA

Example: atomic propositions $a$ and $b$, formula $\varphi = GFa \land GFb$

The trick: save using a better acceptance condition
Generalized Rabin automata

Büchi set $I$ of states: visit $I$ inf. often

generalized Büchi

$\bigwedge_{i=1..n} I_i$: visit each $I_i$ inf. often

Rabin pairs

$\bigvee_{j=1..k} (F^j, I^j)$:

for some $j$ visit $F^j$ fin. often and $I^j$ inf. often
**Generalized Rabin automata**

Büchi set $I$ of states: visit $I$ inf. often

generalized Büchi $\bigwedge_{i=1..n} I_i$: visit each $I_i$ inf. often

Rabin pairs $\bigvee_{j=1..k} (F^j, I^j)$: for some $j$ visit $F^j$ fin. often and $I^j$ inf. often

generalized Rabin pairs $\bigvee_{j=1..k} (F^j, \bigwedge_{i=1..n} I^j_i)$: for some $j$ visit $F^j$ fin. often and each $I^j_i$ inf. often
Theoretically...

How to use DGRA $\mathcal{A}$ with $\bigvee_{j=1..k} (F^j, \exists^j)$ where $\exists^j = \bigwedge_{i=1..n_j} l^j_i$?
Theoretically...

How to use DGRA $\mathcal{A}$ with \( \bigvee_{j=1..k} (F^j, \exists^j) \) where \( \exists^j = \bigwedge_{i=1..n_j} l^j_i \) ?

- **De-generalize** into Rabin automata
  - create copies of $\mathcal{A}$ to track which $l^j_i$'s you are now waiting for
  - *de-generalization index* \( D := | \prod_{j=1}^k \exists^j | = n_1 \cdot \ldots \cdot n_k \)
  - Examples:
    - for $\mathbf{GF}a \land \mathbf{GF}b$ we have $D = 2$
Theoretically...

How to use DGRA $\mathcal{A}$ with $\bigvee_{j=1..k} (F^j, \exists^j)$ where $\exists^j = \bigwedge_{i=1..n_j} I^j_i$?

- **De-generalize** into Rabin automata
  - create copies of $\mathcal{A}$ to track which $l^j_i$’s you are now waiting for
  - *de-generalization index* $D := |\prod_{j=1}^k \exists^j| = n_1 \cdot \ldots \cdot n_k$
  - Examples:
    - for $\mathbf{GF}a \land \mathbf{GF}b$ we have $D = 2$
    - for conjunction of 3 fairness constraints $D = 24$
How to use DGRA $\mathcal{A}$ with $\bigvee_{j=1..k} (F^j, \exists^j)$ where $\exists^j = \bigwedge_{i=1..n_j} l_i^j$ ?

- **De-generalize** into Rabin automata
  - create copies of $\mathcal{A}$ to track which $l_i^j$'s you are now waiting for
  - *de-generalization index* $\mathcal{D} := | \prod_{j=1}^k \exists^j | = n_1 \cdot \ldots \cdot n_k$
  - Examples:
    - for $\textbf{GF} a \land \textbf{GF} b$ we have $\mathcal{D} = 2$
    - for conjunction of 3 fairness constraints $\mathcal{D} = 24$
    - for conjunction of 4 fairness constraints $\mathcal{D} = 20736$

- Use directly by model checking/synthesis algorithms
- extend algorithms from DRA to DGRA
- for probabilistic model checking almost the same complexity $\Rightarrow$ speed up by factor $\mathcal{D}^5/3$ (exponential in # pairs, 2-exp. in # fairness constraints)
Theoretically...

How to use DGRA $\mathcal{A}$ with $\bigvee_{j=1..k} (F^j, Z^j)$ where $Z^j = \bigwedge_{i=1..n_j} I^j_i$ ?

- **De-generalize** into Rabin automata
  - create copies of $\mathcal{A}$ to track which $I^j_i$’s you are now waiting for
  - *de-generalization index* $D := |\prod_{j=1}^{k} Z^j| = n_1 \cdot \ldots \cdot n_k$
  - Examples:
    - for $\text{GF}a \land \text{GF}b$ we have $D = 2$
    - for conjunction of 3 fairness constraints $D = 24$
    - for conjunction of 4 fairness constraints $D = 20736$

- Use **directly** by model checking/synthesis algorithms
  - extend algorithms from DRA to DGRA
  - for probabilistic model checking almost the same complexity

$\Rightarrow$ **speed up by factor** $D^{5/3}$ (exponential in # pairs, 2-exp. in # fairness constraints)
Case study: Pnueli-Zuck randomized mutual exclusion protocol

- 2 368 states for 3 participants
- 27 600 states for 4 participants
- 308 800 states for 5 participants

▶ standard method for Rabin automata
▶ Rabin via generalized Rabin (optimized)
▶ generalized Rabin directly
Case study: Pnueli-Zuck randomized mutual exclusion protocol
- 2,368 states for 3 participants
- 27,600 states for 4 participants
- 308,800 states for 5 participants

- standard method for Rabin automata
- Rabin via generalized Rabin (optimized)
- generalized Rabin directly

PRISM running times in seconds, time-out after 30 minutes

<table>
<thead>
<tr>
<th>Formula</th>
<th>#</th>
<th>t</th>
<th>t</th>
<th>t</th>
<th>t/t</th>
<th>D</th>
<th>t/t</th>
</tr>
</thead>
<tbody>
<tr>
<td>( GFp_1 = 10 \land GFp_2 = 10 \land GFp_3 = 10 )</td>
<td>3</td>
<td>1.2</td>
<td>0.4</td>
<td>0.2</td>
<td>2.2</td>
<td>3</td>
<td>6.8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>17.4</td>
<td>1.8</td>
<td>0.3</td>
<td>6.4</td>
<td>3</td>
<td>60.8</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>257.5</td>
<td>15.2</td>
<td>0.6</td>
<td>26.7</td>
<td>3</td>
<td>447.9</td>
</tr>
<tr>
<td>( (FGp_1 \neq 0 \lor FGp_2 \neq 0 \lor GFp_3 = 0) \lor )</td>
<td>3</td>
<td>289.7</td>
<td>12.6</td>
<td>3.4</td>
<td>3.7</td>
<td>12</td>
<td>84.3</td>
</tr>
<tr>
<td>( (FGp_1 \neq 0 \land GFp_2 = 10 \land GFp_3 = 10) )</td>
<td>4</td>
<td>–</td>
<td>194.5</td>
<td>33.2</td>
<td>5.9</td>
<td>12</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>–</td>
<td>–</td>
<td>543</td>
<td>–</td>
<td>12</td>
<td>–</td>
</tr>
<tr>
<td>( (GFp_1 = 0 \lor GFp_2 \neq 0) \land )</td>
<td>3</td>
<td>–</td>
<td>122.1</td>
<td>7.1</td>
<td>17.2</td>
<td>24</td>
<td>–</td>
</tr>
<tr>
<td>( (GFp_2 = 0 \lor GFp_3 \neq 0) \land )</td>
<td>4</td>
<td>–</td>
<td>–</td>
<td>75.6</td>
<td>–</td>
<td>24</td>
<td>–</td>
</tr>
<tr>
<td>( (GFp_3 = 0 \lor GFp_1 \neq 0) )</td>
<td>5</td>
<td>–</td>
<td>–</td>
<td>1219.5</td>
<td>–</td>
<td>24</td>
<td>–</td>
</tr>
</tbody>
</table>
K., Esparza (CAV 2012)

**Deterministic automata for the \((F,G)\)-fragment of LTL.**
- novel translation of \(LTL(F,G)\) to Rabin automata
- generalized Rabin pairs acceptance condition introduced

Gaiser, K., Esparza (ATVA 2012)

**Rabinizer: Small deterministic automata for \(LTL(F,G)\).**
- optimized implementation for \(LTL(F,G)\)

Chatterjee, Gaiser, K. (CAV 2013)

**Automata with generalized Rabin pairs for probabilistic model checking and LTL synthesis.**
- verification algorithms extended from Rabin to generalized Rabin
- theoretical speed ups \((D^{5/3})\) and experimental speed ups (see table)

K., Ledesma Garza (ATVA 2013)

**Rabinizer 2: Small deterministic automata for \(LTL \setminus GU\).**
- translation to generalized Rabin for \(LTL(X,F,G,U)\) without \(U\) in scope of any \(G\)

Current and future work:
- extending the approach to the whole \(LTL\)
- implementation downloadable as a **plug-in** for PRISM
Probabilistic model checking

MDP $M$ → LTL $\varphi$ → tableaux (Spin), altern. automata (LTL2BA) →
Non-deterministic Büchi automaton → determinisation – Safra,…
Deterministic Rabin automaton

Product to be analysed → MEC decomposition + evaluation

MEC collapse → reachability – lin. prog., value iteration,…

$Pr_{\text{max}}[M \models \varphi]$
**Solution III: Ignore parts of the state space!**

Statistical model checking \[\ldots,\text{Younes’02, Clarke’anytime he likes,}\ldots\]

- simple
- fast
- black-box
- low memory requirements

<table>
<thead>
<tr>
<th></th>
<th>MC</th>
<th>MDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>bounded</td>
<td>convergence + rate</td>
<td>convergence</td>
</tr>
<tr>
<td>unbounded</td>
<td>convergence</td>
<td>?</td>
</tr>
</tbody>
</table>
Solution III: Ignore parts of the state space!

Statistical model checking \([\ldots, \text{Younes’02, Clarke’ anytime he likes,}\ldots]\)

- simple
- fast
- black-box
- low memory requirements

<table>
<thead>
<tr>
<th></th>
<th>MC</th>
<th>MDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>bounded</td>
<td>convergence + rate</td>
<td>convergence</td>
</tr>
<tr>
<td></td>
<td>↑</td>
<td></td>
</tr>
<tr>
<td>unbounded</td>
<td>convergence</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>precompute 0 probability vertices</td>
<td></td>
</tr>
<tr>
<td></td>
<td>in each step stop with (p), go on with (1 - p)</td>
<td></td>
</tr>
</tbody>
</table>

\[\text{MC MDP} \xrightarrow{\text{reinforcement learning}}\]
Probably approximately correct (PAC) reinforcement learning

- with prob. $1 - \delta$ is the strategy $\varepsilon$-optimal after $n(\delta, \varepsilon, |M|)$ trials
- try many runs before concluding the value is lower
Probably approximately correct (PAC) reinforcement learning

- with prob. $1 - \delta$ is the strategy $\varepsilon$-optimal after $n(\delta, \varepsilon, |M|)$ trials
- try many runs before concluding the value is lower

Further, PAC and value iteration together:
Probably approximately correct (PAC) reinforcement learning

- with prob. $1 - \delta$ is the strategy $\varepsilon$-optimal after $n(\delta, \varepsilon, |M|)$ trials
- try many runs before concluding the value is lower

Further, PAC and value iteration together:

VI → PAC → VI
Model checking **huge** probabilistic systems? Some solutions:

1. **Speed up computation**: parallelism, abstraction, …
2. **Make the system/automaton smaller**:

**Future work**: whole LTL, more complex acceptance, PRISM plugin

3. **Ignore parts of the system**: statistical model checking + machine learning/AI (reinforcement learning, PAC)

**Future work**: trading guarantees for speed, more advanced ML/AI methods, succinct (**sublinear**) space representation
Model checking huge probabilistic systems? Some solutions:

1. **Speed up computation**: parallelism, abstraction, ...

2. Make the system/automaton smaller:

**Future work**: whole LTL, more complex acceptance, PRISM plugin

3. Ignore parts of the system:
statistical model checking + machine learning/AI
(reinforcement learning, PAC)

**Future work**: trading guarantees for speed, more advanced ML/AI methods, succinct (sublinear) space representation

Thank you!