Learning to Use Learning in Verification

Jan Křetínský

Technische Universität München, Germany

joint work with T. Brázdil (Masaryk University Brno),
K. Chatterjee, M. Chmelík, P. Daca, A. Fellner, T. Henzinger, T. Petrov (IST Austria),
V. Forejt, M. Kwiatkowska, M. Ujma (Oxford University)
D. Parker (University of Birmingham)
published at ATVA 2014, CAV 2015, TACAS 2016

Mysore Park Workshop
Trends and Challenges in Quantitative Verification
February 3, 2016
Welcome
Entertainment system will be available shortly

Once the entertainment services are activated, a moving map will appear on your screen. You can adjust volume, select channel and change mode (audio/video/map) according to your preferences, through the control unit in the armrest or by touching the screen.
Service not available
No Signal (Error Code: 6001)

This could be due to a bad connection or bad weather conditions at your end or at broadcast centre.

1. Check the cable connections and remove any obstruction around the dish.
2. Try to change channels.
3. Restart the set top box by switching power off and then on.

For installation setup:

[Instructions in a non-English script]
TO PREVENT DAMAGE TO LAP BELT, ENSURE THE BELT IS FASTENED BEFORE SEAT PAN IS STOWED.

Seat must be occupied during take off and landing
Approaches and their interaction

Formal methods

- precise
- scalability issues
Approaches and their interaction

Formal methods

- precise
- scalability issues

MEM-OUT
Approaches and their interaction
**Approaches and their interaction**

**Formal methods**
- **precise**
- **scalability issues**

**Learning**
- **weaker guarantees**
- **scalable**

**precise computation**

**focus on important stuff**

**Decision tree**, capturing which **ACTION** is ok to play in the current state:

\[
\text{ACTION} = \begin{cases} 
\text{rec} & \text{if } l > 0 \land b = 1 \land ip\_mess = 1 \rightarrow \\
\text{b'=0&z'=0&n1'=min(n1+1,8)&ip\_mess'=0} & \text{otherwise}
\end{cases}
\]

**z \leq 0**

- **Y**: action
- **N**: next state
Approaches and their interaction

Formal methods
  ▶ precise
  ▶ scalability issues
Approaches and their interaction

Formal methods
- precise
- scalability issues

Learning
- weaker guarantees
- scalable

different objectives
Approaches and their interaction

Formal methods
- precise
- scalability issues

Learning
- weaker guarantees
- scalable
Approaches and their interaction

Formal methods
- precise
- scalability issues

Learning
- weaker guarantees
- scalable

precise computation

focus on important stuff
What problems?

- Verification
  - (ε)-optimality \( \rightarrow \) PAC
  - hard guarantees \( \rightarrow \) probably correct

- Controller synthesis
  - convergence is preferable
  - at least probably correct?

- Synthesis
Markov decision processes

\[(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))\]
Markov decision processes

\[(S, s_0 \in S, A, \Delta : S \to A \to D(S))\]
Markov decision processes

\[(S, s_0 \in S, A, \Delta : S \to A \to D(S))\]
Markov decision processes

\((S, s_0 \in S, A, \Delta : S \rightarrow A \rightarrow \mathcal{D}(S))\)

\[
\max_{\sigma} \mathbb{P}^\sigma[\text{Reach err}]
\]
Markov decision processes

\[(S, s_0 \in S, A, \Delta : S \rightarrow A \rightarrow D(S))\]

\[\max_{\text{strategy } \sigma} \mathbb{P}^\sigma[\text{Reach err}]\]
Markov decision processes

\[(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))\]

\[\text{max } \mathbb{P}^\sigma[\text{Reach err}]\]
Markov decision processes

\[(S, s_0 \in S, A, \Delta : S \to A \to \mathcal{D}(S))\]

\[
\begin{align*}
&\text{max}_{\text{strategy } \sigma} \mathbb{P}^\sigma[\text{Reach err}] \\
&\text{ACTION = down}
\end{align*}
\]
Ex. 1: Computing strategies faster: How?

Fixed-point computation

\[ V(s) := \max_{a \in \Delta(s)} V(s, a) \]

\[ V(s, a) := \sum_{s' \in S} \Delta(s, a, s') \cdot V(s') \]
Ex.1: Computing strategies faster: How?

Fixed-point computation

\[ V(s) := \max_{a \in \Delta(s)} V(s, a) \]

\[ V(s, a) := \sum_{s' \in S} \Delta(s, a, s') \cdot V(s') \]

Order of evaluation?
Fixed-point computation

\[
V(s) := \max_{a \in \Delta(s)} V(s, a)
\]

\[
V(s, a) := \sum_{s' \in S} \Delta(s, a, s') \cdot V(s')
\]

Order of evaluation? [ATVA’14]

More frequently evaluate those states that are \textit{visited} more frequently
Fixed-point computation

\[ V(s) := \max_{a \in \Delta(s)} V(s, a) \]

\[ V(s, a) := \sum_{s' \in S} \Delta(s, a, s') \cdot V(s') \]

Order of evaluation? [ATVA’14]

More frequently evaluate those states that are visited more frequently by reasonably good schedulers
Ex. 1: Computing strategies faster: How?

**Reinforcement learning**

- **Fixed-point computation**
  
  \[
  V(s) := \max_{a \in \Delta(s)} V(s, a)
  \]

  \[
  V(s, a) := \sum_{s' \in S} \Delta(s, a, s') \cdot V(s')
  \]

- **Order of evaluation?** [ATVA’14]
  
  More frequently evaluate those states that are **visited** more frequently by reasonably good schedulers
Ex.1: Computing strategies faster: Algorithm

1: $U(\cdot, \cdot) \leftarrow 1$, $L(\cdot, \cdot) \leftarrow 0$
2: $L(1, \cdot) \leftarrow 1$, $U(0, \cdot) \leftarrow 0$
3: repeat

7: until $U(s_0) - L(s_0) < \epsilon$
Ex. 1: Computing strategies faster: Algorithm

1: $U(\cdot, \cdot) \leftarrow 1, L(\cdot, \cdot) \leftarrow 0$
2: $L(1, \cdot) \leftarrow 1, U(0, \cdot) \leftarrow 0$

3: repeat
4: sample a path from $s_0$ to $\{1, 0\}$
   > actions uniformly from $\arg \max_a U(s, a)$
   > states according to $\Delta(s, a, s')$

7: until $U(s_0) - L(s_0) < \epsilon$
Ex. 1: Computing strategies faster: Algorithm

1: \( U(\cdot, \cdot) \leftarrow 1, L(\cdot, \cdot) \leftarrow 0 \)
2: \( L(1, \cdot) \leftarrow 1, U(0, \cdot) \leftarrow 0 \)

3: repeat
4: sample a path from \( s_0 \) to \( \{1, 0\} \)
   \[ \text{actions uniformly from } \arg \max_a U(s, a) \]
   \[ \text{states according to } \Delta(s, a, s') \]
5: for all visited transitions \((s, a, s')\) do
6: \( \text{UPDATE}(s, a, s') \)
7: until \( U(s_0) - L(s_0) < \epsilon \)
Ex. 1: Computing strategies faster: Algorithm

1. \( U(\cdot, \cdot) \leftarrow 1, \ L(\cdot, \cdot) \leftarrow 0 \)
2. \( L(1, \cdot) \leftarrow 1, \ U(0, \cdot) \leftarrow 0 \)

3. repeat
4. sample a path from \( s_0 \) to \( \{1, 0\} \)
   \( \triangleright \) actions uniformly from \( \arg \max_a U(s, a) \)
   \( \triangleright \) states according to \( \Delta(s, a, s') \)

5. for all visited transitions \((s, a, s')\) do
6. \( \text{UPDATE}(s, a, s') \)

7. until \( U(s_0) - L(s_0) < \epsilon \)

------------------------------------------

1. procedure \( \text{UPDATE}(s, a, \cdot) \)
2. \( U(s, a) := \sum_{s' \in S} \Delta(s, a, s') \cdot U(s') \)
3. \( L(s, a) := \sum_{s' \in S} \Delta(s, a, s') \cdot L(s') \)
Ex. 1: Computing strategies faster

- **Reinforcement Learning**
  - Important parts of the system
  - Faster & sure updates

- **Value Iteration**

Guaranteed upper & lower bounds at all times + practically fast convergence
Ex. 1: Computing strategies faster

Guaranteed upper & lower bounds at all times + practically fast convergence

Remark:
- PAC SMC for MDP and (unbounded) LTL \([\text{ATVA’14}]: |S|, p_{\text{min}}\)
- practical PAC SMC for MC and (unbounded) LTL + mean payoff \([\text{TACAS’16}]: p_{\text{min}}\)
### Ex.1: Experimental results

<table>
<thead>
<tr>
<th>Example</th>
<th>Visited states</th>
<th>PRISM</th>
<th>BRTDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3,001,911</td>
<td>760</td>
</tr>
<tr>
<td></td>
<td>zeroconf</td>
<td>4,427,159</td>
<td>977</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5,477,150</td>
<td>1411</td>
</tr>
<tr>
<td></td>
<td>wlan</td>
<td>345,000</td>
<td>2018</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1,295,218</td>
<td>2053</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5,007,548</td>
<td>1995</td>
</tr>
<tr>
<td></td>
<td>firewire</td>
<td>6,719,773</td>
<td>26,508</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13,366,666</td>
<td>25,214</td>
</tr>
<tr>
<td></td>
<td></td>
<td>19,213,802</td>
<td>32,214</td>
</tr>
<tr>
<td></td>
<td>mer</td>
<td>17,722,564</td>
<td>1950</td>
</tr>
<tr>
<td></td>
<td></td>
<td>17,722,564</td>
<td>2902</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26,583,064</td>
<td>1950</td>
</tr>
<tr>
<td></td>
<td></td>
<td>26,583,064</td>
<td>2900</td>
</tr>
</tbody>
</table>
Further examples on reinforcement learning

Sebastian Junges, Nils Jansen, Christian Dehnert, Ufuk Topcu, Joost-Pieter Katoen:  
*Safety-Constrained Reinforcement Learning for MDPs.* TACAS 2016.  
▶ safe and cost-optimizing strategies  
▶ (1) compute safe, permissive strategies  
▶ (2) learn cost-optimal strategies (convergence) among them

Alexandre David, Peter GjJ Jensen, Kim Guldstrand Larsen, Axel Legay, Didier Lime, Mathias Grund Srensen, Jakob Haahr Taankvist:  
*On Time with Minimal Expected Cost!* ATVA 2014.  
▶ priced timed games → priced timed MDPs  
▶ time-bounded cost-optimal (convergence) strategies  
▶ (1) Uppaal TiGa for safe strategies  
▶ (2) Uppaal SMC and learning for cost-optimal strategies
Importance of a node $s$ with respect to $\diamond target$ and strategy $\sigma$:

$$P^\sigma[\diamond s]$$
Importance of a node $s$ with respect to $\Diamond target$ and strategy $\sigma$:

$$P^\sigma[\Diamond s \mid \Diamond target]$$
Ex.2: Computing small strategies: Which decisions?

Importance of a node $s$ with respect to $\Diamond target$ and strategy $\sigma$:

$$P^\sigma[\Diamond s \mid \Diamond target]$$

Cut off states with zero importance (unreachable or useless)
Importance of a node $s$ with respect to $\diamond target$ and strategy $\sigma$:

$$P^\sigma[\diamond s \mid \diamond target]$$

Cut off states with zero importance (unreachable or useless)

Cut off states with low importance (small error, $\varepsilon$-optimal strategy)
Ex.2: Small strategies: Which representation?

How to make use of the exact importance?
Ex.2: Small strategies: Which representation?

How to make use of the exact \textit{importance}?

\textbf{Learn} decisions in $s$ in proportion to importance of $s$
Ex.2: Small strategies: Which representation?

How to make use of the exact importance?

Learn decisions in $s$ in proportion to importance of $s$

Advantages of decision trees over BDDs:
- more readable: predicates
- smaller due to good ordering: entropy
- yet smaller at a cost of an error: pruning
Ex.2: Small strategies: Which representation?

How to make use of the exact **importance**?

**Learn** decisions in $s$ in proportion to importance of $s$

Advantages of decision trees over BDDs:
- more readable: predicates
- smaller due to good ordering: entropy
- yet smaller at a cost of an error: pruning

```
ACTION = rec

Y    N

ACTION = l>0&b=1&ip.mess=1 -> 
b'=0&z'=0&n1'=min(n1+1,8)&ip.mess'=0

Y    N

z ≤ 0

Y    N

ACTION = rec
```
### Ex.2: Experimental results

<table>
<thead>
<tr>
<th>Example</th>
<th>#states</th>
<th>Value</th>
<th>Explicit</th>
<th>BDD</th>
<th>DT</th>
<th>Rel.err(DT) %</th>
</tr>
</thead>
<tbody>
<tr>
<td>firewire</td>
<td>481,136</td>
<td>1.0</td>
<td>479,834</td>
<td>4233</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>investor</td>
<td>35,893</td>
<td>0.958</td>
<td>28,151</td>
<td>783</td>
<td>27</td>
<td>0.886</td>
</tr>
<tr>
<td>mer</td>
<td>1,773,664</td>
<td>0.200016</td>
<td>MEM-OUT</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zeroconf</td>
<td>89,586</td>
<td>0.00863</td>
<td>60,463</td>
<td>409</td>
<td>7</td>
<td>0.106</td>
</tr>
</tbody>
</table>

* MEM-OUT in PRISM, whereas BRTDP yields: 1887 619 13 0.00014
### Example

<table>
<thead>
<tr>
<th>Example</th>
<th>#states</th>
<th>Value</th>
<th>Explicit</th>
<th>BDD</th>
<th>DT</th>
<th>Rel.err(DT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>firewire</td>
<td>481,136</td>
<td>1.0</td>
<td>479,834</td>
<td>4233</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>investor</td>
<td>35,893</td>
<td>0.958</td>
<td>28,151</td>
<td>783</td>
<td>27</td>
<td>0.886</td>
</tr>
<tr>
<td>mer</td>
<td>1,773,664</td>
<td>0.200016</td>
<td>——— MEM-OUT ——— *</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>zeroconf</td>
<td>89,586</td>
<td>0.00863</td>
<td>60,463</td>
<td>409</td>
<td>7</td>
<td>0.106</td>
</tr>
</tbody>
</table>

* MEM-OUT in PRISM, whereas BRTDP yields: 1887 619 13 0.00014
Further examples on decision trees


- positive examples from runs of the program
- negative examples from modifications
- implication examples


- positive examples: states reachable when preconditions holds
- negative examples: states leaving loop and violating a postcondition
Machine learning in verification

- **Scalable heuristics**
- Example 1: **Speeding up** value iteration
  - TECHNIQUE: reinforcement learning, BRTDP
  - IDEA: focus on updating “most important parts” = most often visited by good strategies
- Example 2: **Small and readable strategies**
  - TECHNIQUE: decision tree learning
  - IDEA: based on the importance of states, feed the decisions to the learning algorithm
Verification using machine learning

- How far do we want to compromise?
- Do we have to compromise?
  - BRTDP, invariant generation, strategy representation don’t
- Don’t we want more than ML?
  - $(\varepsilon)$-optimal controllers?
  - arbitrary controllers – is it still verification?
- What do we actually want?
  - scalability shouldn’t overrule guarantees?
    - SMC should be PAC; when is it enough?
- Oracle usage seems fine
Verifying using machine learning

- How far do we want to compromise?
- Do we have to compromise?
  - BRTDP, invariant generation, strategy representation don’t
- Don’t we want more than ML?
  - $(\varepsilon)$-optimal controllers?
  - arbitrary controllers – is it still verification?
- What do we actually want?
  - scalability shouldn’t overrule guarantees?
    - SMC should be PAC; when is it enough?
- Oracle usage seems fine

Thank you