The What, Why, and How of Probabilistic Verification
Part 4: Recent Research Developments

Joost-Pieter Katoen

UNIVERSITY OF TWENTE.

CAV Invited Tutorial 2015, San Francisco
Overview

Recent Research Developments
- Parameter Synthesis
- Model Repair
- Counterexample Generation
- Probabilistic Programming

Epilogue
Overview

Recent Research Developments
- Parameter Synthesis
- Model Repair
- Counterexample Generation
- Probabilistic Programming

Epilogue
Motivation

Fact

Probabilistic model checking is applicable to various areas, e.g.:

- fault-tolerant systems
- randomized algorithms
- systems biology
Motivation

Fact

Probabilistic model checking is applicable to various areas, e.g.:

- fault-tolerant systems
- randomized algorithms
- systems biology

Limitation

Probabilities need to be known a priori
Motivation

Fact

Probabilistic model checking is applicable to various areas, e.g.:

▸ fault-tolerant systems
▸ randomized algorithms
▸ systems biology

Limitation

Probabilities need to be known \textit{a priori}

Goal

Treat parametric models, synthesize “safe” parameter values
Motivation

Fact
Probabilistic model checking is applicable to various areas, e.g.:
- fault-tolerant systems
- randomized algorithms
- systems biology

Limitation
Probabilities need to be known \text{a priori}

Goal
Treat parametric models, synthesize “safe” parameter values

New: PROPhESY — A PRObabilistic PharametEr SYntesis Tool
compute

e.g. reachability probabilities, expected rewards, conditional probabilities

Pr(\◇ ) = \frac{1}{6}
**Parametric Markov Chains**

**idea:** enrich discrete-time Markov chains with parameters

- $Pr(\diamond \bullet) = \frac{1}{6}$
- $f_\diamond(p) = \frac{p^2}{p+1}$
- $f_\diamond(0.5) = \frac{1}{6}$

**compute rational functions** representing

- e.g. reachability probabilities, expected rewards, conditional probabilities
Inputs:

1. a (finite) parametric discrete-time Markov chain
2. a property (e.g., reachability, expected reward, conditional reachability)
3. a threshold

Desired output:

For which parameter values does the pMC satisfy the property with the given threshold?
Parameter Synthesis

Inputs:
1. a (finite) parametric discrete-time Markov chain
2. a property (e.g., reachability, expected reward, conditional reachability)
3. a threshold

Desired output:
For which parameter values does the pMC satisfy the property with the given threshold?

Problem instances:
- What is the maximal tolerable message loss?
- What is the maximal tolerable failure rate for program correctness?
- ......
Adapt the automaton-to-regular expression algorithm to parametric MCs.
Adapt the automaton-to-regular expression algorithm to parametric MCs.
Adapt the automaton-to-regular expression algorithm to parametric MCs.
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]
Hierarchical SCC Decomposition

[Jansen et al., 2014]

For which (combinations of) values for $p$ and $q$ is the probability of reaching 5 smaller than $c \in [0, 1]$?

$\Rightarrow$ Evaluate rational function.
1. Determine the rational function $f$ for the given property-of-interest.
   - Use SCC-based state elimination
   - Use dedicated library CaRL for treating rational functions

2. In CEGAR-like style determine parameter sub-spaces for which $f < \text{bound}$
   - Sample the parameter space
   - Automatically generate candidate regions
   - Check whether region is completely safe (unsafe) \(^1\)
   - If sub-space contains an invalid point, refine the region and re-check

\(^1\) Using SMT techniques for non-linear theories, e.g., Z3 or SMT-RAT.
Parameter Synthesis

[Diagram of a scatter plot showing data points and a flowchart labeled PROPhESY with boxes for sampling, region generator, SMT solver, core model, checker, and GUI.]
A Live Demo
Recent Research Developments

Sampling and Regions

Prophesy UI

Input
- Upload Prism
- Upload result
- Select result
- Path to prism file: Browse...
- No file selected.
- Path to PCTL file: Browse...
- No file selected.
- Run with: Prophesy
- Upload

Sampling
- Manual Sampling
- Auto Sampling
- Sampling number: 9
- Number of iterations: 3
- Go

Constraints
- Manual Constraints
- Auto Constraints
- Growing rectangles
- Generate

Settings
- Threshold: 0.5
- SMT Solver: Z3
Experimental Results

competitors
- PARAM [Hahn et al., 2010]
- PRISM [Parker et al., 2011]

models
- Bounded retransmission protocol
- NAND multiplexing
- Zeroconf, Crowds protocol
- $10^4$ to $7.5 \cdot 10^6$ states

experiments:
- best set-up for each tool
- log-scale x- and y-axis

http://moves.rwth-aachen.de/research/tools/prophesy/
Experimental Results

[Dehnert et al., 2015]

competitors

- PARAM [Hahn et al., 2010]
- PRISM [Parker et al., 2011]
- prototype [Baier et al., 2014]

models

- Bounded retransmission protocol
- NAND multiplexing
- Zeroconf, Crowds protocol
- $10^4$ to $7.5 \cdot 10^6$ states

experiments:

- best set-up for each tool
- log-scale $x$- and $y$-axis

http://moves.rwth-aachen.de/research/tools/prophesy/
Overview

Recent Research Developments

- Parameter Synthesis
- Model Repair
- Counterexample Generation
- Probabilistic Programming

Epilogue
Model Repair: Motivation

[Bartocci et al., 2011]

- Assume a reachability probability threshold is not met
Recent Research Developments

Model Repair: Motivation

[Bartocci et al., 2011]

- Assume a reachability probability threshold is not met
- What to do? **No problem!** Counterexamples point to errors
Model Repair: Motivation

[Bartocci et al., 2011]

- Assume a reachability probability threshold is not met
- What to do? *No problem!* Counterexamples point to errors
- ....... But they are often large and incomprehensible
Model Repair: Motivation

[Bartocci et al., 2011]

- Assume a reachability probability threshold is not met
- What to do? **No problem!** Counterexamples point to errors
- ....... But they are often large and incomprehensible
- Can we “fix” a model when a bad state is reached too often?
Recent Research Developments

Model Repair: Motivation [Bartocci et al., 2011]

- Assume a reachability probability threshold is not met
- What to do? No problem! Counterexamples point to errors
- ....... But they are often large and incomprehensible
- Can we “fix” a model when a bad state is reached too often?
- What is the minimal change to be made?
Recent Research Developments

Model Repair: Motivation

- Assume a reachability probability threshold is not met

- What to do? **No problem!** Counterexamples point to errors

- ....... But they are often large and incomprehensible

- Can we “fix” a model when a bad state is reached too often?

- What is the **minimal** change to be made?

Here, automated **model repair** algorithms come into the play.
A Robotics Scenario
A Robotics Scenario

- Object moves between position $A$ and $B$ according to MC $\mathcal{M}_1$
A Robotics Scenario

- Object moves between position $A$ and $B$ according to MC $M_1$
- Robot moves according to strategy given by MC $M_2$
A Robotics Scenario

- Object moves between position A and B according to MC $M_1$
- Robot moves according to strategy given by pMC $M_2$
  \[\Rightarrow\] parameters indicate variability of the robot’s strategy
A Robotics Scenario

- Object moves between position $A$ and $B$ according to MC $\mathcal{M}_1$
- Robot moves according to strategy given by pMC $\mathcal{M}_2$
  $\Rightarrow$ parameters indicate variability of the robot’s strategy
- In $\mathcal{M}_1 \parallel \mathcal{M}_2$, robot catches ball at positions $AA$ or $BB$
Recent Research Developments

Repairing Robotics Example

- Assume concrete strategy $\mathcal{M}$ (obtained via reinforcement learning)
- Property $\varphi$: The probability to catch at $B$ shall be smaller than $1/2$

$$\begin{align*}
\mathcal{D}: & \\
\begin{array}{c}
AA \\
1 \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
1 \\
\end{array} & \xrightarrow{\frac{1}{4}} & \xrightarrow{\frac{1}{4}} & \xrightarrow{\frac{1}{4}} & \xrightarrow{\frac{1}{4}}
\begin{array}{c}
BA \\
1 \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
\frac{1}{4} \\
1 \\
\end{array}
\end{align*}$$

Valuation $v(x) = v(y) = 0$
Recent Research Developments

Repairing Robotics Example

- Assume concrete strategy \( \mathcal{M} \) (obtained via reinforcement learning)
- Property \( \varphi \): The probability to catch at \( B \) shall be smaller than \( \frac{1}{2} \)
- \( p_{BB} = \frac{1}{2} \)

Valuation \( \nu(x) = \nu(y) = 0 \Rightarrow \nu \not\models \varphi \)
Repairing Robotics Example

- Assume concrete strategy $\mathcal{M}$ (obtained via reinforcement learning)
- Property $\varphi$: The probability to catch at $B$ shall be smaller than $1/2$
- $p_{BB} = 1/2$ 😞 but $\hat{p}_{BB} = 1/3$ 😊

Valuation $v(x) = v(y) = 0 \Rightarrow v \not\models \varphi$

$\hat{v}(x) = 0.2, \hat{v}(y) = 0 \Rightarrow \hat{v} \models \varphi$
A Local Repair Approach

[Pathak et al., 2015]

Local repair strategy for pMC and $\varphi = \Pr(\diamond B) < p$:

\begin{itemize}
  \item For simple parameter dependencies, $|R| = 1$ can be taken.
\end{itemize}
A Local Repair Approach

[Pathak et al., 2015]

Local repair strategy for pMC and $\varphi = \Pr(\Diamond B) < p$:

- Maintain graph structure of the pMC

\(^2\)For simple parameter dependencies, $|R| = 1$ can be taken.
A Local Repair Approach

[Pathak et al., 2015]

Local repair strategy for pMC and $\varphi = \Pr(\Diamond B) < p$:

- Maintain graph structure of the pMC
- Heuristically select a set $R \subseteq S$ of states\(^2\) to be repaired

\(^2\)For simple parameter dependencies, $|R| = 1$ can be taken.
A Local Repair Approach [Pathak et al., 2015]

Local repair strategy for pMC and $\varphi = \text{Pr}(\Diamond B) < p$:

- Maintain graph structure of the pMC
- Heuristically select a set $R \subseteq S$ of states\(^2\) to be repaired
- Shift probability mass of some edges of $r \in R$ to its other edges, s.t.
  1. $P'(s, u) = P(s, u)$ for all $s \notin R$ and $u \in S$, and
  2. $\sum_{u \in S} P'(r, u) \cdot \Pr(u \models \Diamond B) < \sum_{u \in S} P(r, u) \cdot \Pr(u \models \Diamond B)$ for all $r$ in $R$

\(^2\)For simple parameter dependencies, $|R| = 1$ can be taken.
A Local Repair Approach  

[Pathak et al., 2015]

Local repair strategy for pMC and $\varphi = \Pr(\Diamond B) < p$:

- Maintain graph structure of the pMC
- Heuristically select a set $R \subseteq S$ of states to be repaired
- Shift probability mass of some edges of $r \in R$ to its other edges, s.t.
  1. $P'(s, u) = P(s, u)$ for all $s \notin R$ and $u \in S$, and
  2. $\sum_{u \in S} P'(r, u) \cdot \Pr(u \models \Diamond B) < \sum_{u \in S} P(r, u) \cdot \Pr(u \models \Diamond B)$ for all $r$ in $R$

- This yields a strict partial order on valuations $v' \prec v$

---

$^2$For simple parameter dependencies, $|R| = 1$ can be taken.
A Local Repair Approach

[Pathak et al., 2015]

Local repair strategy for pMC and $\varphi = \Pr(\diamond B) < p$:

- Maintain graph structure of the pMC
- Heuristically select a set $R \subseteq S$ of states\(^2\) to be repaired
- Shift probability mass of some edges of $r \in R$ to its other edges, s.t.
  1. $P'(s, u) = P(s, u)$ for all $s \notin R$ and $u \in S$, and
  2. $\sum_{u \in S} P'(r, u) \cdot \Pr(u \models \diamond B) < \sum_{u \in S} P(r, u) \cdot \Pr(u \models \diamond B)$ for all $r$ in $R$

- This yields a strict partial order on valuations $v' < v$
- Iterations yield a repair sequence $v_n, \ldots, v_0$ with $n > 0$ such that:
  $v_{i+1} < v_i$ and $v_i \not\models \varphi$ for all $i < n$ and $v_n \models \varphi$

---

\(^2\)For simple parameter dependencies, $|R| = 1$ can be taken.
A Local Repair Approach [Pathak et al., 2015]

Local repair strategy for pMC and $\varphi = \Pr(\Diamond B) < p$:

- Maintain graph structure of the pMC
- Heuristically select a set $R \subseteq S$ of states\(^2\) to be repaired
- Shift probability mass of some edges of $r \in R$ to its other edges, s.t.
  1. $P'(s, u) = P(s, u)$ for all $s \notin R$ and $u \in S$, and
  2. $\sum_{u \in S} P'(r, u) \cdot \Pr(u \models \Diamond B) < \sum_{u \in S} P(r, u) \cdot \Pr(u \models \Diamond B)$ for all $r$ in $R$

- This yields a strict partial order on valuations $\nu' < \nu$
- Iterations yield a repair sequence $\nu_n, \ldots, \nu_0$ with $n > 0$ such that:
  $\nu_{i+1} < \nu_i$ and $\nu_i \not\models \varphi$ for all $i < n$ and $\nu_n \models \varphi$
- If no finite repair sequence exists, the pMC is not repairable!

\(^2\)For simple parameter dependencies, $|R| = 1$ can be taken.
Local Repair in Action

Valuation $v(x) = v(y) = 0 \Rightarrow v \not\models \phi$  
$\hat{v}(x) = 0.2, \hat{v}(y) = 0 \Rightarrow \hat{v} \models \phi$

We have $\hat{v} < v$ and BA is only repaired state.
Local Repair Algorithm

- Pick state
- Find local repair
- Model checking (e.g., MRMC)
- Repaired
  \[ \text{pMC } \hat{M} \models \varphi \]
Local Repair Achievements

Soundness
A local repair step repairs at least one state and does not “un-repair” others.

Intuition: Reachability probabilities cannot be increased by local decreasing. (Induction over transient probabilities).

Completeness
If each repair has a certain mass, termination with a minimal result is ensured.

Intuition: The repair mass of infinite sequences converges to zero.
Cost-Optimal Repairs
Cost-Optimal Repairs

- Greedy strategy implies cost functions depending on local changes
- Non-linear cost functions would significantly affect computation time
Cost-Optimal Repairs

- Greedy strategy implies cost functions depending on local changes
- Non-linear cost functions would significantly affect computation time
- Aim: keep changes in parameter values of initial valuation $v_0$ small:

$$\sum_{x_i \in \text{Var}} |v(x_i) - v_0(x_i)|$$
Cost-Optimal Repairs

- Greedy strategy implies cost functions depending on local changes
- Non-linear cost functions would significantly affect computation time
- Aim: keep changes in parameter values of initial valuation \( v_0 \) small:

\[
\sum_{x_i \in \text{Var}} |v(x_i) - v_0(x_i)|
\]

- Prefer local repair step in state \( s \) that minimises

\[
\sum_{x_i \in \text{Var}(s)} |(v(x_i) + \delta_i) - v_0(x_i)|
\]
Cost-Optimal Repairs

- Greedy strategy implies cost functions depending on local changes
- Non-linear cost functions would significantly affect computation time
- Aim: keep changes in parameter values of initial valuation $v_0$ small:

$$\sum_{x_i \in \text{Var}} |v(x_i) - v_0(x_i)|$$

- Prefer local repair step in state $s$ that minimises

$$\sum_{x_i \in \text{Var}(s)} |(v(x_i) + \delta_i) - v_0(x_i)|$$

As we use heuristics to pick a state, no global optimal cost is achieved.
## Experimental Results for Robot Case Study

Strategies obtained via reinforcement learning from MDP environment

| $N$ | states | trans | mc | pick | $\text{Pr}^D$ | $\text{Pr}^\Delta$ | $\sum \Delta$ | $mx_\Delta$ | $|E|$ | steps |
|-----|--------|-------|----|------|-------------|---------------|---------------|-------------|-----|------|
| 48  | 2305   | 17859 | 1.05 | .159 | .001        | 33.4          | .72           | 621         |     |     |
| 64  | 4097   | 32003 | 1.66 | .182 | .001        | 18.0          | .65           | 427         |     |     |
| 96  | 9217   | 72579 | 6.00 | .189 | .001        | 28.0          | .68           | 657         |     |     |
| 128 | 16385  | 129539| 8.46 | .150 | .001        | 20.2          | .45           | 640         |     |     |
| 256 | 65537  | 521219| 63.6 | .130 | .000        | 28.0          | .27           | 888         |     |     |
| 512 | 262145 | 2091011| –   | .168 | .101        | 19.9          | .26           | 480         |     |     |
| 512$^a$ | 262145 | 2091011| 21.7 | .168 | .000        | 54.8          | .26           | 1760        |     |     |
| 1024| 1048577| 8376323| –   | .105 | .104        | 1.1           | .19           | 24          |     |     |
| 1024$^a$ | 1048577| 8376323| 28.9 | .105 | .036        | 80.8          | .25           | 2400        |     |     |

C++ implementation, Intel I7 CPU 3.4 GHz with 32GB RAM; $\text{TO} = 2700$ sec

$^a =$ repairing $|R| = 20$ states simultaneously.
# Experimental Results for Robot Case Study

Strategies obtained via reinforcement learning from MDP environment

<table>
<thead>
<tr>
<th>( N )</th>
<th>states</th>
<th>trans</th>
<th>( \text{mc} )</th>
<th>( \text{pick} )</th>
<th>( \text{Pr}^{D} )</th>
<th>( \text{Pr}^\hat{D} )</th>
<th>( \sum \Delta )</th>
<th>( mx_\Delta )</th>
<th>( \text{E} )</th>
<th>steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>2305</td>
<td>17859</td>
<td>0.76</td>
<td>1.05</td>
<td>.159</td>
<td>.001</td>
<td>33.4</td>
<td>.72</td>
<td>621</td>
<td></td>
</tr>
<tr>
<td>64</td>
<td>4097</td>
<td>32003</td>
<td>1.58</td>
<td>1.66</td>
<td>.182</td>
<td>.001</td>
<td>18.0</td>
<td>.65</td>
<td>427</td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>9217</td>
<td>72579</td>
<td>7.17</td>
<td>6.00</td>
<td>.189</td>
<td>.001</td>
<td>28.0</td>
<td>.68</td>
<td>657</td>
<td></td>
</tr>
<tr>
<td>128</td>
<td>16385</td>
<td>129539</td>
<td>15.3</td>
<td>8.46</td>
<td>.150</td>
<td>.001</td>
<td>20.2</td>
<td>.45</td>
<td>640</td>
<td></td>
</tr>
<tr>
<td>256</td>
<td>65537</td>
<td>521219</td>
<td>129</td>
<td>63.6</td>
<td>.130</td>
<td>.000</td>
<td>28.0</td>
<td>.27</td>
<td>888</td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>262145</td>
<td>2091011</td>
<td>TO</td>
<td>–</td>
<td>.168</td>
<td>.101</td>
<td>19.9</td>
<td>.26</td>
<td>480</td>
<td></td>
</tr>
<tr>
<td>512(^a)</td>
<td>262145</td>
<td>2091011</td>
<td>145</td>
<td>21.7</td>
<td>.168</td>
<td>.000</td>
<td>54.8</td>
<td>.26</td>
<td>1760</td>
<td></td>
</tr>
<tr>
<td>1024</td>
<td>1048577</td>
<td>8376323</td>
<td>TO</td>
<td>–</td>
<td>.105</td>
<td>.104</td>
<td>1.1</td>
<td>.19</td>
<td>24</td>
<td></td>
</tr>
<tr>
<td>1024(^a)</td>
<td>1048577</td>
<td>8376323</td>
<td>378</td>
<td>28.9</td>
<td>.105</td>
<td>.036</td>
<td>80.8</td>
<td>.25</td>
<td>2400</td>
<td></td>
</tr>
</tbody>
</table>

C++ implementation, Intel I7 CPU 3.4 GHz with 32GB RAM; TO = 2700 sec

\(^a\) = repairing \( |R| = 20 \) states simultaneously.
Experimental Results for Robot Case Study

Strategies obtained via reinforcement learning from MDP environment

<table>
<thead>
<tr>
<th>model</th>
<th>time</th>
<th>quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>states</td>
<td>trans</td>
</tr>
<tr>
<td>48</td>
<td>2305</td>
<td>17859</td>
</tr>
<tr>
<td>64</td>
<td>4097</td>
<td>32003</td>
</tr>
<tr>
<td>96</td>
<td>9217</td>
<td>72579</td>
</tr>
<tr>
<td>128</td>
<td>16385</td>
<td>129539</td>
</tr>
<tr>
<td>256</td>
<td>65537</td>
<td>521219</td>
</tr>
<tr>
<td>512</td>
<td>262145</td>
<td>2091011</td>
</tr>
<tr>
<td>512$^a$</td>
<td>262145</td>
<td>2091011</td>
</tr>
<tr>
<td>1024</td>
<td>1048577</td>
<td>8376323</td>
</tr>
<tr>
<td>1024$^a$</td>
<td>1048577</td>
<td>8376323</td>
</tr>
</tbody>
</table>

C++ implementation, Intel I7 CPU 3.4 GHz with 32GB RAM; TO = 2700 sec

$^a$ = repairing $|R| = 20$ states simultaneously.
### Experimental Results for Robot Case Study

Strategies obtained via reinforcement learning from MDP environment

<table>
<thead>
<tr>
<th>Model</th>
<th>Time</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>states</td>
<td>trans</td>
</tr>
<tr>
<td>48</td>
<td>2305</td>
<td>17859</td>
</tr>
<tr>
<td>64</td>
<td>4097</td>
<td>32003</td>
</tr>
<tr>
<td>96</td>
<td>9217</td>
<td>72579</td>
</tr>
<tr>
<td>128</td>
<td>16385</td>
<td>129539</td>
</tr>
<tr>
<td>256</td>
<td>65537</td>
<td>521219</td>
</tr>
<tr>
<td>512</td>
<td>262145</td>
<td>2091011</td>
</tr>
<tr>
<td>512$^a$</td>
<td>262145</td>
<td>2091011</td>
</tr>
<tr>
<td>1024</td>
<td>1048577</td>
<td>8376323</td>
</tr>
<tr>
<td>1024$^a$</td>
<td>1048577</td>
<td>8376323</td>
</tr>
</tbody>
</table>

C++ implementation, Intel I7 CPU 3.4 GHz with 32GB RAM; TO = 2700 sec

$^a$ = repairing $|R| = 20$ states simultaneously.
Experiments on Parametric PRISM Benchmarks

<table>
<thead>
<tr>
<th>Model</th>
<th>time</th>
<th>quality</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N</td>
<td>K</td>
</tr>
<tr>
<td>Crowds</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Crowds</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>Crowds</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>Crowds</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Crowds</td>
<td>10</td>
<td>6</td>
</tr>
<tr>
<td>Crowds</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>NAND</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>NAND</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>NAND</td>
<td>12</td>
<td>6</td>
</tr>
<tr>
<td>NAND</td>
<td>12</td>
<td>8</td>
</tr>
<tr>
<td>NAND</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>NAND</td>
<td>12</td>
<td>12</td>
</tr>
</tbody>
</table>

- “False” valuations introduced, repaired towards original result
- Correct model probabilities and the repaired ones are quite close
Recent Research Developments

Parameter Synthesis
Model Repair
Counterexample Generation
Probabilistic Programming

Epilogue
It is impossible to overestimate the importance of counterexamples. The counterexamples are invaluable in debugging complex systems. Some people use model checking just for this feature.

Ed Clarke, 25 Years of Model Checking, FLOC 2008

Relevance for CEGAR, model repair, scheduling problems, model analysis ...
Counterexamples

- **LTL counterexamples are finite paths**
  - $\Box \Phi$: a path ending in a $\neg \Phi$-state
  - $\Diamond \Phi$: a $\neg \Phi$-path leading to a $\neg \Phi$ cycle
  - BFS yields shortest counterexamples

- **CTL counterexamples are (mostly) finite trees**
  - universal CTL $\setminus$ LTL: trees or proof-like counterexample
  - existential CTL: witnesses, annotated counterexample

- **What are counterexamples for probabilistic reachability?**
  - a set of finite paths whose probability mass exceeds a threshold
  - represented as minimal (critical) sub-models
Minimal Critical Subsystems

Given a DTMC $\mathcal{D}$ refuting $\Pr(\square G) > p$, that is $\Pr(\Diamond \neg G) \leq 1 - p$
Minimal Critical Subsystems

Given a DTMC $\mathcal{D}$ refuting $\Pr(\Box G) > p$, that is $\Pr(\Diamond \neg G) \leq 1 - p$

Critical subsystem

A subset $C \subseteq S$ such that the probability of reaching a $\neg G$-state by only visiting states in $C$ is already beyond $1 - p$. 
Recent Research Developments

Minimal Critical Subsystems

Given a DTMC $\mathcal{D}$ refuting $\Pr(\square G) > p$, that is $\Pr(\Diamond \neg G) \leq 1-p$

Critical subsystem

A subset $C \subseteq S$ such that the probability of reaching a $\neg G$-state by only visiting states in $C$ is already beyond $1-p$.

Goal

Compute a critical subsystem with a minimum number of states. This is a minimal critical subsystem.
Recent Research Developments

Minimal Critical Subsystems

Given a DTMC $\mathcal{D}$ refuting $\Pr(\square G) > p$, that is $\Pr(\Diamond \neg G) \leq 1 - p$

Critical subsystem

A subset $C \subseteq S$ such that the probability of reaching a $\neg G$-state by only visiting states in $C$ is already beyond $1 - p$.

Goal

Compute a critical subsystem with a minimum number of states. This is a minimal critical subsystem.

For arbitrary PCTL-formulas, finding a minimal critical subsystem is NP-complete.
Example MCS

Property: target is reachable with probability $< \frac{7}{10}$
Recent Research Developments

Example MCS

MCS for which target is reachable with probability $> \frac{7}{10}$
MILP formulation for MCS

[Jansen et al., 2012]
MILP formulation for MCS

[Jansen et al., 2012]

**Variables**

- $x_s \in \{0, 1\}$, a decision variable for each state $s$
- $p_s \in [0, 1]$, reachability probability for state $s$ within the subsystem
MILP formulation for MCS

[Jansen et al., 2012]

**Variables**

- \( x_s \in \{0, 1\} \), a decision variable for each state \( s \)
- \( p_s \in [0, 1] \), reachability probability for state \( s \) within the subsystem

**Constraints**

\[
\text{minimize } \sum_{s \in S} x_s \\
\text{such that} \\
\text{initial state } s_0 : \quad p_{s_0} > 1 - p \\
\text{target states } s : \quad p_s = x_s \\
\text{non-target states } s : \quad p_s \leq x_s \\
\text{non-target states } s : \quad p_s \leq \sum_{u \in S} P(s, u) \cdot p_u
\]
MILP formulation for MCS

- This yields only a lower bound on required probability $p_{s_0}$
- Additionally, we want to obtain an MCS with a maximal probability
This yields only a lower bound on required probability $p_{s_0}$
- Additionally, we want to obtain an MCS with a maximal probability

**Adapted constraints (for some $0 < c < 1$)**

\[
\text{minimize} \quad \sum_{s \in S} -c \cdot p_{s_0} + x_s \\
\text{such that} \\
\text{initial state } s_0 : \quad p_{s_0} > 1 - p \\
\text{target states } s : \quad p_s = x_s \\
\text{non-target states } s : \quad p_s \leq x_s \\
\text{non-target states } s : \quad p_s \leq \sum_{u \in S} P(s, u) \cdot p_u \]
MILP formulation for MCS [Jansen et al., 2012]

- This yields only a lower bound on required probability $p_{s_0}$
- Additionally, we want to obtain an MCS with a maximal probability

Adapted constraints (for some $0 < c < 1$)

\[
\begin{align*}
\text{minimize} & \quad \sum_{s \in S} -c \cdot p_{s_0} + x_s \\
\text{such that} & \\
\text{initial state } s_0 : & \quad p_{s_0} > 1 - p \\
\text{target states } s : & \quad p_s = x_s \\
\text{non-target states } s : & \quad p_s \leq x_s \\
\text{non-target states } s : & \quad p_s \leq \sum_{u \in S} P(s, u) \cdot p_u
\end{align*}
\]

This can be generalised for PCTL, and rewards. MDPs: $\omega$-regular properties.
Experiments

A bar chart showing the increase in size with respect to the MCS for various benchmarks. The x-axis represents different benchmarks such as brp32-2, brp512-2, crowds5-4, crowds5-6, crowds5-8, sleader4-4, sleader4-6, sleader4-8, sleader8-4, aleader3, aleader4, consensus2-2, consensus2-4, csma2-2, csma2-4, and csma2-6. The y-axis represents the increase in size, with values ranging from 0 to 8. The chart compares different verification methods, indicated by different colors and markers.
Recent Research Developments

Experiments

- Hard to proof optimality (NP complete for most of the settings)
- Using intermediate results of MILP solvers gives good heuristic method

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>States</th>
<th>λ</th>
<th>Subsystem</th>
<th>Time (s)</th>
<th>Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>crowds</td>
<td>68740</td>
<td>0.1</td>
<td>83</td>
<td>343</td>
<td>&lt; 1 GB</td>
</tr>
<tr>
<td>sleader</td>
<td>12302</td>
<td>0.5</td>
<td>6150</td>
<td>22</td>
<td>&lt; 1 GB</td>
</tr>
<tr>
<td>consensus</td>
<td>272</td>
<td>0.1</td>
<td>15</td>
<td>733</td>
<td>&lt; 1 GB</td>
</tr>
<tr>
<td>csma</td>
<td>66718</td>
<td>0.1</td>
<td>415</td>
<td>2364</td>
<td>&lt; 1 GB</td>
</tr>
</tbody>
</table>
PRISM Counterexamples

- Counterexamples on level of state space are typically hard to grasp
- Idea: generate counterexamples directly on model description level!

```
module coin
  f: bool init 0;
  c: bool init 0;
  [flip] ¬f → 0.5 : (f' = 1) & (c' = 1) + 0.5 : (f' = 1) & (c' = 0);
  [reset] f ∧ ¬c → 1 : (f' = 0);
  [proc] f → 0.99 : (f' = 1) + 0.01 : (c' = 1);
endmodule

module processor
  p: bool init 0;
  [proc] ¬p → 1 : (p' = 1);
  [loop] p → 1 : (p' = 1);
  [reset] true → 1 : (p' = 0)
endmodule
```
PRISM Model’s State Space

\[ P_{\leq 1-\lambda}(\Diamond C) \iff P_{\text{max}}^{A}(s_{\text{init}} \models \Diamond C) \leq 1 - \lambda \]
Obtaining PRISM Counterexamples

Minimal set of the PRISM commands whose induced MDP is already buggy!

MILP Approach [Jansen et al., 2013]

1. Assign a unique label to each command.
2. Construct state space, label transitions their originating commands.
3. Use a MILP formulation to minimize the number of commands.
Obtaining PRISM Counterexamples

Minimal set of the PRISM commands whose induced MDP is already buggy!

**MAXSAT Approach**

[Dehnert et al., 2014]

1. Use MAX-SAT solver to enumerate possible combinations of commands of minimal size
2. Check their criticality by model checking
3. Analyse non-critical command sets to infer constraints for a “better” solution.
MAX-SAT Approach
Experimental Results
Experimental Results
Recent Research Developments

Experiments: Conclusion

- Complimentary technique to minimal critical subsystems
- Extendible to Continuous-time Markov Chains

<table>
<thead>
<tr>
<th>Model</th>
<th>Comm.</th>
<th>Cex.</th>
<th>Time (s)</th>
<th>Lower bound</th>
<th>Removed branches</th>
</tr>
</thead>
<tbody>
<tr>
<td>consensus</td>
<td>14</td>
<td>≤ 9</td>
<td>&gt; 600</td>
<td>7</td>
<td>1 / 12</td>
</tr>
<tr>
<td>consensus</td>
<td>28</td>
<td>≤ 20</td>
<td>&gt; 600</td>
<td>5</td>
<td>2 / 24</td>
</tr>
<tr>
<td>csma</td>
<td>38</td>
<td>36</td>
<td>184.05</td>
<td>—</td>
<td>20 / 90</td>
</tr>
<tr>
<td>firewire</td>
<td>68</td>
<td>28</td>
<td>545.68</td>
<td>—</td>
<td>38 / 68</td>
</tr>
<tr>
<td>wlan</td>
<td>76</td>
<td>8</td>
<td>0.04</td>
<td>—</td>
<td>6 / 14</td>
</tr>
<tr>
<td>wlan</td>
<td>76</td>
<td>≤ 38</td>
<td>&gt; 600</td>
<td>32</td>
<td>31 / 72</td>
</tr>
</tbody>
</table>
Recent Research Developments

Parameter Synthesis
Model Repair
Counterexample Generation
Probabilistic Programming

Epilogue
Motivation

Probabilistic Programming for Advancing Machine Learning (PPAML)
Probabilistic Programs
What are probabilistic programs?

Sequential, possibly non-deterministic, programs with \textit{random assignments}. 
What are probabilistic programs?
Sequential, possibly non-deterministic, programs with random assignments.

Applications
Security, machine learning, quantum computing, approximate computing
Probabilistic Programs

What are probabilistic programs?
Sequential, possibly non-deterministic, programs with random assignments.

Applications
Security, machine learning, quantum computing, approximate computing

The scientific challenge
- Such programs are small, but hard to understand and analyse.
What are probabilistic programs?
Sequential, possibly non-deterministic, programs with random assignments.

Applications
Security, machine learning, quantum computing, approximate computing

The scientific challenge
- Such programs are small, but hard to understand and analyse.
- Problems: infinite variable domains, parameters, and loops.
What are probabilistic programs?
Sequential, possibly non-deterministic, programs with random assignments.

Applications
Security, machine learning, quantum computing, approximate computing

The scientific challenge

- Such programs are small, but hard to understand and analyse.
- Problems: infinite variable domains, parameters, and loops.

⇒ **Aim:** push the limits of their automated analysis
Recent Research Developments

Program Equivalence  

[Keifer et al., 2012]

int XminY1(float p, q)
{
    int x, f := 0, 0;
    while (f = 0) {
        (x += 1 [p] f := 1);
    }
    f := 0;
    while (f = 0) {
        (x -= 1 [q] f := 1);
    }
    return x;
}

int XminY2(float p, q)
{
    int x, f := 0, 0;
    (f := 0 [0.5] f := 1);
    if (f = 0) {
        while (f = 0) {
            (x += 1 [p] f := 1);
        }
    } else {
        f := 0;
        while (f = 0) {
            while (f = 0) {
                x := 1;
                (skip [q] f := 1);
            }
        }
    }
    return x;
}
Program Equivalence

The programs are equivalent for \((p, q) = \left(\frac{1}{2}, \frac{2}{3}\right)\).
Recent Research Developments

Program Equivalence

[Kiefer et al., 2012]

The programs are equivalent for \((p, q) = \left(\frac{1}{2}, \frac{2}{3}\right)\). Q: No other ones?
Probabilistic Guarded Command Language

- skip
- abort
- $x := E$
- prog1 ; prog2
- if (G) prog1 else prog2
- prog1 [] prog2
- prog1 [p] prog2
- while (G) prog
The decision problem whether a pGCL program almost surely terminate on one given input is as hard as the problem whether an ordinary program terminates on all possible inputs.
int cowboyDuel(float a, b) {
    int t := A [] t := B;
    bool c := true;
    while (c) {
        if (t = A) {
            (c := false [a] t := B);
        } else {
            (c := false [b] t := A);
        }
    }
    return t;
}

This MDP is parameterized but finite. Once we count the number of shots before one of the cowboys dies, the MDP becomes infinite. Our approach however allows to determine e.g., the expected number of shots before success.
Weakest Preconditions

### Syntax

- `skip`
- `abort`
- `x := E`
- `P1 ; P2`
- `if (G)P1 else P2`
- `P1 [] P2`
- `P1 [p] P2`
- `while (G)P`

### Semantics `wp(P, f)`

- `f`
- `0`
- `f[x := E]`
- `wp(P_1, wp(P_2, f))`
- `\[G\] \cdot wp(P_1, f) + [\neg G] \cdot wp(P_2, f)`
- `\min (wp(P_1, f), wp(P_2, f))`
- `p \cdot wp(P_1, f) + (1-p) \cdot wp(P_2, f)`
- \( \mu X. ([G] \cdot wp(P, X) + [\neg G] \cdot f) \)

\( \mu \) is the least fixed point operator wrt. the ordering \( \leq \) on expectations.
Determining Weakest Preconditions is Hard

Correspondence [Gretz et al., 2013]

For pGCL-program $P$, variable valuation $\eta$, and post-expectation $f$, it holds that $wpP, f(\eta)$ equals the expected reward of reaching a terminal state in $P$'s MDP.\(^3\)

\(^3\)All states have reward 0, except terminal states $\langle \varepsilon, \eta' \rangle$ have reward $f(\eta')$. 
Determining Weakest Preconditions is Hard

Correspondence

For pGCL-program $P$, variable valuation $\eta$, and post-expectation $f$, it holds that $wpP, f(\eta)$ equals the expected reward of reaching a terminal state in $P$’s MDP.\(^3\)

Thus: weakest pre-conditions can be obtained as expected rewards in infinite-state parametric MDPs.

This is as hard as solving the universal halting problem!

---

\(^3\)All states have reward 0, except terminal states $\langle \varepsilon, \eta' \rangle$ have reward $f(\eta')$. 
Determining Weakest Preconditions is Hard

Correspondence [Gretz et al., 2013]

For pGCL-program $P$, variable valuation $\eta$, and post-expectation $f$, it holds that $wpP, f(\eta)$ equals the expected reward of reaching a terminal state in $P$’s MDP.\(^3\)

Thus: weakest pre-conditions can be obtained as expected rewards in infinite-state parametric MDPs.

This is as hard as solving the universal halting problem!

Is there no hope for automation? Well, semi-automation.

---

\(^3\)All states have reward 0, except terminal states $\langle \varepsilon, \eta' \rangle$ have reward $f(\eta')$.\)
Main steps
1. Speculatively annotate a program with linear expressions:

\[
[\alpha_1 \cdot x_1 + \ldots + \alpha_n \cdot x_n + \alpha_{n+1} \ll 0] \cdot (\beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n + \beta_{n+1})
\]

with real parameters \(\alpha_i, \beta_i\), program variable \(x_i\), and \(\ll \in \{<, \leq\}\).
Recent Research Developments

Loop-Invariant Synthesis

Main steps

1. Speculatively annotate a program with linear expressions:

\[
[\alpha_1 \cdot x_1 + \ldots + \alpha_n \cdot x_n + \alpha_{n+1} \ll 0] \cdot (\beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n + \beta_{n+1})
\]

with real parameters \(\alpha_i, \beta_i\), program variable \(x_i\), and \(\ll \in \{<, \leq\}\).

2. Transform these numerical constraints into Boolean predicates.
Main steps

1. Speculatively annotate a program with linear expressions:

\[
[\alpha_1 \cdot x_1 + \ldots + \alpha_n \cdot x_n + \alpha_{n+1} \ll 0] \cdot (\beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n + \beta_{n+1})
\]

with real parameters \(\alpha_i, \beta_i\), program variable \(x_i\), and \(\ll \in \{<, \leq\}\).

2. Transform these numerical constraints into Boolean predicates.

3. Transform these predicates into non-linear FO formulas.
Main steps

1. Speculatively annotate a program with linear expressions:

\[ [\alpha_1 \cdot x_1 + \ldots + \alpha_n \cdot x_n + \alpha_{n+1} \ll 0] \cdot (\beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n + \beta_{n+1}) \]

with real parameters \( \alpha_i, \beta_i \), program variable \( x_i \), and \( \ll \in \{<, \leq\} \).

2. Transform these numerical constraints into Boolean predicates.

3. Transform these predicates into non-linear FO formulas.

4. Use constraint-solvers for quantifier elimination (e.g., Redlog).
Main steps

1. Speculatively annotate a program with linear expressions:

\[
[\alpha_1 \cdot x_1 + \ldots + \alpha_n \cdot x_n + \alpha_{n+1} \ll 0] \cdot (\beta_1 \cdot x_1 + \ldots + \beta_n \cdot x_n + \beta_{n+1})
\]

with real parameters \(\alpha_i, \beta_i\), program variable \(x_i\), and \(\ll \in \{<, \leq\}\).

2. Transform these numerical constraints into Boolean predicates.
3. Transform these predicates into non-linear FO formulas.
4. Use constraint-solvers for quantifier elimination (e.g., Redlog).
5. Simplify the resulting formulas (e.g., using Slfq and SMT solving).
6. Exploit resulting assertions to infer program correctness.

Quantitative version of approach by [Colón et al., 2003] for ordinary programs.
For any linear pGCL program annotated with propositionally linear expressions, this method will find all parameter solutions that make the annotation valid, and no others.
Prinsys: Synthesis Tool of Probabilistic Invariants

download from moves.rwth-aachen.de/prinsys
Program Equivalence

```c
int XminY1(float p, q){
  int x, f := 0, 0;
  while (f = 0) {
    (x := 1 [p] f := 1);
  }
  f := 0;
  while (f = 0) {
    (x := 1 [p] f := 1);
  }
  return x;
}
```

```c
int XminY2(float p, q){
  int x, f := 0, 0;
  (f := 0 [0.5] f := 1);
  if (f = 0) {
    while (f = 0) {
      (x := 1 [p] f := 1);
    }
  } else {
    f := 0;
    while (f = 0) {
      (x := 1 [p] f := 1);
    }
    (skip [q] f := 1);
  }
  return x;
}
```
Recent Research Developments

Program Equivalence

Analysis with Prinsys yields:

Both programs are equivalent for any $q$ with $q = \frac{1}{2^p}$. 
Conditioning

\[ P(A | B) = \frac{P(B | A)P(A)}{P(B)} \]
One fish is contained within the confines of an opaque fishbowl. The fish is equally likely to be a piranha or a goldfish. A sushi lover throws a piranha into the fish bowl alongside the other fish. Then, immediately, before either fish can devour the other, one of the fish is blindly removed from the fishbowl. The fish that has been removed from the bowl turns out to be a piranha. What is the probability that the fish that was originally in the bowl by itself was a piranha?
The Piranha Problem

1. \( (f1 := \text{goldfish}[0.5] \ f1 := \text{piranha}); \)
2. \( f2 := \text{piranha}; \)
3. \( (\text{sample} := f1[0.5] \ \text{sample} := f2); \)
4. \( \text{observe}([\text{sample} = \text{piranha}]); \)
The Piranha Problem

What is the probability that the original fish in the bowl was a piranha?

1. \( (f1 := \text{goldfish} [0.5] \ f1 := \text{piranha}); \)
2. \( f2 := \text{piranha}; \)
3. \( (\text{sample} := f1 [0.5] \ \text{sample} := f2); \)
4. \( \text{observe}([\text{sample} = \text{piranha}]); \)
The Piranha Problem

What is the probability that the original fish in the bowl was a piranha?

$$\mathbb{E}(f1 = \text{piranha} \mid P \text{ terminates}) = \frac{1 \cdot \frac{1}{2} + 0 \cdot \frac{1}{4}}{\frac{1}{2} + \frac{1}{4}} = \frac{\frac{1}{2}}{\frac{3}{4}} = \frac{2}{3}.$$
Infeasible Programs

Program $P$ does not terminate. Program $Q$ is infeasible.
Overview

Recent Research Developments
- Parameter Synthesis
- Model Repair
- Counterexample Generation
- Probabilistic Programming

Epilogue
Probabilistic Model Checking ...

- ....... is a mature automated technique
- ....... focuses on quantitative measures
- ....... has a broad range of applications
- ....... is scalable
- ....... is extendible to costs
- ....... offers many interesting challenges!
Probabilistic Model Checking ...

- is a mature automated technique
- focuses on quantitative measures
- has a broad range of applications
- is scalable
- is extendible to costs
- offers many interesting challenges!

Current Research

- probabilistic program analysis
- tight game-based abstractions
- parametric verification and synthesis
- stochastic hybrid systems