Refined Probabilistic Abstraction

Björn Wachter

December 8, 2010
• Bug in control software of power network
  ⇒ 50 million people without electricity
Model checking:

“does a computing system behave as intended?”

- mathematical model $M$ of system
- specification $\varphi$
- automatic proof or refutation of:

\[ M \models \varphi \]

- Example: $\varphi = \text{no arithmetic overflow}$

![Diagram showing a transition from $s_0$ to error state with a label indicating e.g., arithmetic overflow.]
Probabilities are Important

- computer networks
  - performance: $P(\text{message loss}) = 2\%$
  - reliability: $P(\text{node failure}) = 3\%$

- randomized algorithms
  - network protocols
  - sorting algorithms
  - ...

Probabilistic Model checking

- models: Markov chains
- properties: PCTL
Probabilistic Model checking

- models: Markov chains
- properties: PCTL
- Zeroconf protocol
- IP for new member picked probabilistically
- bad: two members have the same IP!

\[ P(\text{same IP}) < 0.01 \]
Why Abstraction?

- Limitations of probabilistic model checking
  - based on state-space exploration
  - state-space explosion problem

\[ M \models \varphi \]

\[ M \uparrow \models \varphi \]
Why Abstraction?

- Limitations of probabilistic model checking
  - based on state-space exploration
  - state-space explosion problem
- Abstraction very successful

\[ M \models \varphi \]
Why Abstraction?

- Limitations of probabilistic model checking
  - based on state-space exploration
  - state-space explosion problem
- Abstraction very successful

$M \models \varphi$  
$M^\# \models \varphi$
Why Abstraction?

- Limitations of probabilistic model checking
  - based on state-space exploration
  - state-space explosion problem
- Abstraction very successful
Why Abstraction?

- Limitations of probabilistic model checking
  - based on state-space exploration
  - state-space explosion problem
- Abstraction very successful

\[ M \models \varphi \quad \text{guarantees} \quad M^\sharp \models \varphi \]
Why Abstraction?

- Limitations of probabilistic model checking
  - based on state-space exploration
  - state-space explosion problem
- Abstraction very successful
  - Refinement fits the abstraction to the property
Contribution

Abstraction refinement for very large probabilistic models

- ... even infinite ones!

- implementation in PASS tool

- successful on various network protocols
  - Wireless LAN
  - IPV4
  - BRP
  - ...


Background
Probabilistic Programs

// parallel composition of modules
module sender
i : int; // variable definition
...
endmodule

module channelK
[aF] (k=0) -> 0.98 : (k’=1) // probabilistic
+ 0.02 : (k’=2); // guarded command
endmodule

init T=false & ... endinit // initial states
Semantics: **Markov Decision Process (MDP)**

states $S$ \[\equiv\] assignments to program variables
Semantics: **Markov Decision Process (MDP)**

- states \( S \)
- probabilistic transitions

\[ s_0 \sim \text{assignments to program variables} \]

\[ \ldots \sim \text{probabilistic choice} \]
Semantics: Markov Decision Process (MDP)

- states $S$ \implies \text{assignments to program variables}
- probabilistic transitions \ldots \text{probabilistic choice}
- non-deterministic choice \ldots \text{concurrency}

- Markov chain \implies \text{deterministic MDP}
Semantics: **Markov Decision Process (MDP)**

- States $S$ = assignments to program variables
- Probabilistic transitions = ... probabilistic choice
- Non-deterministic choice = ... concurrency

- Markov chain $\Rightarrow$ deterministic MDP
Properties: Probabilistic Reachability

- probabilities to reach states $F \subseteq S$
- valuations $[0, 1]^S \cong S \rightarrow [0, 1]$
Properties: Probabilistic Reachability

- probabilities to reach states $F \subseteq S$
- valuations $[0, 1]^S \cong S \rightarrow [0, 1]$

Reachability probability: 1
Properties: Probabilistic Reachability

- probabilities to reach states $F \subseteq S$
- valuations $[0, 1]^S \cong S \rightarrow [0, 1]$

reachability probability: $\frac{1}{2}$
Properties: Probabilistic Reachability

- probabilities to reach states $F \subseteq S$
- valuations $[0, 1]^S \cong S \rightarrow [0, 1]$

adversary $\eta : Paths \rightarrow A$

- resloves non-determinism

$\Rightarrow$ induces a Markov chain
Properties: Probabilistic Reachability

- probabilities to reach states \( F \subseteq S \)
- valuations \([0, 1]^S \cong S \rightarrow [0, 1]\)

reachability probability \( p^\eta_F \)
- depends on adversary \( \eta \)
- minimal/maximal

\[
\begin{align*}
    p^{\min}_F &= \inf_{\eta} p^\eta_F \\
    p^{\max}_F &= \sup_{\eta} p^\eta_F
\end{align*}
\]

adversary \( \eta : Paths \rightarrow A \)
- resolves non-determinism
  \( \Rightarrow \) induces a Markov chain
Properties: Probabilistic Reachability

- probabilities to reach states $F \subseteq S$
- valuations $[0, 1]^S \cong S \rightarrow [0, 1]$

adversary $\eta : Paths \rightarrow A$

- resolves non-determinism
$\Rightarrow$ induces a Markov chain

reachability probability $p^\eta_F$
- depends on adversary $\eta$
- minimal/maximal

$\begin{align*}
    p^{min}_F &= \inf_\eta p^\eta_F \\
    p^{max}_F &= \sup_\eta p^\eta_F
\end{align*}$

least fixpoint of $Pre^{min} : [0, 1]^S \rightarrow [0, 1]^S$

$w \mapsto \lambda s. \begin{cases} 
    1 & ; s \in F \\
    0 & ; s \in F_0 \\
    \min_{a \in A(s) \land (u, t) \in U \times S} \sum R(s, u, t) \cdot w(t) & ; \text{ow.}
\end{cases}$
Properties: Probabilistic Reachability

- probabilities to reach states \( F \subseteq S \)
- valuations \([0, 1]^S \cong S \rightarrow [0, 1]\)

adversary \( \eta : \text{Paths} \rightarrow A \)
- resolves non-determinism
  \[ w \mapsto \lambda_s. \begin{cases} 1 & ; s \in F \\ 0 & ; s \in F_0 \\ \min_{a \in A(s)} \sum_{(u,t) \in U \times S} R(s,u,t) \cdot w(t) & ; \text{ow.} \end{cases} \]

- reachability probability \( p_F^\eta \)
- depends on adversary \( \eta \)
- minimal/maximal
  \[ p_{\min}^F = \inf_\eta p_F^\eta \]
  \[ p_{\max}^F = \sup_\eta p_F^\eta \]

least fixpoint of \( \text{Pre}_{\min}^\eta : [0, 1]^S \rightarrow [0, 1]^S \)
Properties: Probabilistic Reachability

- probabilities to reach states $F \subseteq S$
- valuations $[0, 1]^S \cong S \rightarrow [0, 1]$

\[ w \mapsto \lambda s. \begin{cases} 1 & ; s \in F \\ 0 & ; s \in F_0 \\ \max_{a \in A(s)} \sum_{(u, t) \in U \times S} R(s, u, t) \cdot w(t) & ; \text{ow.} \end{cases} \]

adversary $\eta : Paths \rightarrow A$
- resolves non-determinism
- induces a Markov chain

reachability probability $p_F ^\eta$
- depends on adversary $\eta$
- minimal/maximal

\[ p_{min} ^ F = \inf_{\eta} p_F ^\eta \]
\[ p_{max} ^ F = \sup_{\eta} p_F ^\eta \]

least fixpoint of $Pre^{max} : [0, 1]^S \rightarrow [0, 1]^S$
Abstraction
Abstraction for Probabilistic Reachability

- Problem: many states $S$

reachability probability

<table>
<thead>
<tr>
<th></th>
<th>1.0</th>
<th>0.5</th>
<th>1.0</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.1</td>
<td>1.0</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>0.3</td>
<td>0.7</td>
<td>0.4</td>
<td>0.1</td>
<td></td>
</tr>
<tr>
<td>0.0</td>
<td>0.8</td>
<td>0.3</td>
<td>0.2</td>
<td></td>
</tr>
</tbody>
</table>

In general, hard to compute
Abstraction for Probabilistic Reachability

- Problem: many states $S$

1. merge states to blocks $Q$

### Reachability Probability

<table>
<thead>
<tr>
<th></th>
<th>$B_1$</th>
<th>$B_2$</th>
<th>$B_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.0 0.5</td>
<td>1.0 1.0</td>
<td>0.3 0.7</td>
</tr>
<tr>
<td>0.2 0.1</td>
<td>1.0 1.0</td>
<td>0.4 0.1</td>
<td></td>
</tr>
<tr>
<td>0.0 0.8</td>
<td>0.3 0.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

in general, hard to compute
Abstraction for Probabilistic Reachability

- Problem: many states \( S \)

1. merge states to blocks \( Q \)
   - in example, 16 states but only 3 blocks \( Q = \{B_1, B_2, B_3\} \).

reachability probability

in general, hard to compute
Abstraction for Probabilistic Reachability

- Problem: many states $S$

1. merge states to blocks $Q$
   - in example, 16 states but only 3 blocks $Q = \{B_1, B_2, B_3\}$.

2. compute abstract valuations $[0, 1]^Q$

reachability probability

\[
\begin{array}{ccc}
B_1 & B_2 & B_3 \\
1.0 & 0.5 & 1.0 \\
0.2 & 0.1 & 1.0 \\
0.3 & 0.7 & 0.4 \\
0.0 & 0.8 & 0.3 \\
\end{array}
\]

\[
\begin{array}{ccc}
B_1 & B_2 & B_3 \\
1.0 & 1.0 & 1.0 \\
\end{array}
\]
Abstraction for Probabilistic Reachability

• Problem: many states $S$

1. merge states to blocks $Q$
   • in example, 16 states but only 3 blocks $Q = \{B_1, B_2, B_3\}$.

2. compute abstract valuations $[0, 1]^Q$

reachability probability

lower-bound analysis

upper-bound analysis
Challenge of Analysis Design

complex interplay

• Open Question:

what does an optimal analysis look like?
Challenge of Analysis Design

• Open Question:
  what does an optimal analysis look like?

• Our solution:
  • Recipe: Abstract Interpretation [Cousot77]
Challenge of Analysis Design

- Open Question:
  what does an optimal analysis look like?

- Our solution:
  
  - Recipe: Abstract Interpretation [Cousot77]
  - Ingredients:
    - abstraction functions
Abstraction & concretization

1. abstraction functions:
   - mappings
   \[ [0, 1]^S \rightarrow [0, 1]^Q \]
### Abstraction & concretization

1. **Abstraction functions:**
   - **Mappings**
     
     \[ [0, 1]^S \mapsto [0, 1]^Q \]
   - **Lower bound:**
     \[ \alpha^l(w) = \lambda B. \inf_{s \in B} w(s) \]

---

<table>
<thead>
<tr>
<th>W</th>
<th>1.0</th>
<th>0.5</th>
<th>1.0</th>
<th>0.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>0.2</td>
<td>0.1</td>
<td>1.0</td>
<td>0.0</td>
</tr>
<tr>
<td>B2</td>
<td>0.3</td>
<td>0.7</td>
<td>0.4</td>
<td>0.3</td>
</tr>
<tr>
<td>B3</td>
<td>0.0</td>
<td>0.8</td>
<td>0.3</td>
<td>0.2</td>
</tr>
</tbody>
</table>

### Diagram

- **[0, 1]^S**
- **[0, 1]^Q**
- **B1**
- **B2**
- **B3**

**Q-closure**

<table>
<thead>
<tr>
<th>0.1</th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td></td>
</tr>
</tbody>
</table>
Abstraction & concretization

1 abstraction functions:

- mappings
  \([0, 1]^S \mapsto [0, 1]^Q\)
- lower bound:
  \(\alpha^l(w) = \lambda B. \inf_{s \in B} w(s)\)
- upper bound:
  \(\alpha^u(w) = \lambda B. \sup_{s \in B} w(s)\)
Abstraction & concretization

1 abstraction functions:
   - mappings
     \([0, 1]^S \mapsto [0, 1]^Q\)
   - lower bound:
     \(\alpha^l(w) = \lambda B. \inf_{s \in B} w(s)\)
   - upper bound:
     \(\alpha^u(w) = \lambda B. \sup_{s \in B} w(s)\)

2 concretization function
   (or meaning function)
   - \([0, 1]^Q \mapsto [0, 1]^S\)
   - \(\gamma(w^\#) = \lambda s. w^\#([s]).\)
How do we get an Abstract Analysis?

- abstract analysis
  - function $f^\# : [0, 1]^Q \rightarrow [0, 1]^Q$
  - $\Rightarrow$ lower/upper bound = fixpoint of $f^\#$
- Best-transformer paradigm [Cousot 2002]
How do we get an Abstract Analysis?

- abstract analysis
  - function $f^\# : [0, 1]^Q \to [0, 1]^Q$
    $$\Rightarrow \text{lower/upper bound} = \text{fixpoint of } f^\#$$
- Best-transformer paradigm [Cousot 2002]
How do we get an Abstract Analysis?

- abstract analysis
  - function $f^\# : [0, 1]^Q \rightarrow [0, 1]^Q$
    $\Rightarrow$ lower/upper bound = fixpoint of $f^\#
  - Best-transformer paradigm [Cousot 2002]

![Diagram showing the process of abstract analysis](image)
How do we get an Abstract Analysis?

- abstract analysis
  - function $f^\#: [0, 1]^Q \rightarrow [0, 1]^Q$
    \[ \Rightarrow \text{lower/upper bound} = \text{fixpoint of } f^\# \]
- Best-transformer paradigm [Cousot 2002]
Abstract Transformers = Stochastic Games

\[
(\alpha^u \circ \text{Pre}^{\min}_F \circ \gamma(w^\#))(B) = \sup_{s \in B} \text{Pre}^{\min}_F(\gamma(w^\#))(s) \\
= \sup_{s \in B} \min_{a \in A(s)} \sum_{s' \in S} R(s, a)(s') \cdot (\gamma(w^\#))(s')
\]

- Markov models
  - Markov chain = \( \frac{1}{2} \) player
  - MDP = \( 1 \frac{1}{2} \) player
  - stochastic game = \( 2 \frac{1}{2} \) player
- minimum / maximum over all strategies for both players

\[ p_{F}^{\min, \min} \quad p_{F}^{\max, \min} \quad p_{F}^{\min, \max} \quad p_{F}^{\max, \max} \]

\[ 0 \quad \text{concurrency player} \quad 1 \]

\[ \text{abstraction player} \quad \text{abstraction player} \]
Abstract Transformers = Stochastic Games

\[(\alpha^u \circ \text{Pre}^\text{min}_F \circ \gamma(w^\#))(B) = \sup_{s \in B} \text{Pre}^\text{min}_F(\gamma(w^\#))(s)\]

\[= \sup_{s \in B} \min_{a \in A(s)} \sum_{s' \in S} R(s, a)(s') \cdot (\gamma(w^\#))(s')\]

- Markov models
  - Markov chain = 1 \(\frac{1}{2}\) player
  - MDP = 1 \(\frac{1}{2}\) player
  - stochastic game = 2 \(\frac{1}{2}\) player
- minimum / maximum over all strategies for both players

\[p^\text{min, min}_F, p^\text{max, min}_F, p^\text{min, max}_F, p^\text{max, max}_F\]

concurrency
player
abstraction player
abstraction player

0 \quad 1
Wrap-up

MDP \rightarrow \text{abstract MDP}

reachability probability

\[ \ell_b, \ell_u \in \text{probabilistic program semantics} \]
Wrap-up

MDP  \rightarrow  abstract MDP

reachability probability

\[ \text{reachability probability} \in \text{probabilistic program semantics} \]

Next: how to compute this
Wrap-up

• Diagram defines an abstract semantics (mathematics)

MDP \rightarrow \text{abstract MDP}

reachability probability $[lb, ub]$
Wrap-up

A diagram defines an abstract semantics (mathematics). The reachability probability \( \epsilon \) belongs to the interval \([lb, ub]\). 

Next: how to compute this.
Wrap-up

• diagram defines an abstract semantics (mathematics)

probabilistic program

semantics

MDP

abstract MDP

reachability probability \( \in [lb, ub] \)
Wrap-up

- diagram *defines* an abstract semantics (mathematics)

![Diagram](image-url)

- probabilistic program
- semantics
- MDP
- abstract MDP
- reachability probability ∈ \([lb, ub]\)
Wrap-up

- diagram *defines* an abstract semantics (mathematics)

Next: how to *compute* this

![Diagram](image_url)

- $\exists$ reachability probability $\in [lb, ub]$
State of the Art before Thesis

- de Alfaro, Roy.
  Magnifying-Lens Abstraction for Markov Decision Processes.
  CAV 2007

- Chatterjee, Henzinger, Jhala, Majumdar.
  Counterexample Guided Planning.
  UAI 2005

- D’Argenio, Jeannet, Jensen, Larsen.
  Reduction and Refinement Strategies for Probabilistic Analysis.
  PAPM-PROBMIV 2002

- ... build full semantics
State of the Art before Thesis

- de Alfaro, Roy.  
  Magnifying-Lens Abstraction for Markov Decision Processes.  
  CAV 2007

- Chatterjee, Henzinger, Jhala, Majumdar.  
  Counterexample Guided Planning.  
  UAI 2005

- D’Argenio, Jeannet, Jensen, Larsen.  
  Reduction and Refinement Strategies for Probabilistic Analysis.  
  PAPM-PROBMIV 2002

- ... build full semantics  
  ⇒ expensive or even impossible
State of the Art before Thesis

- Chatterjee, Henzinger, Jhala, Majumdar. Counterexample Guided Planning. UAI 2005

- ... build full semantics
  ⇒ expensive or even impossible

Premiere: 1st symbolic abstraction for probabilistic programs
- abstraction at the language level
Predicate Abstraction

- Predicates $\approx$ expressions over program variables
  - e.g., $x > 0$, $x < y$

\[
\begin{array}{llll}
\text{x = 0} & \text{y = 0} & \text{x = 5} & \text{y = 3} \\
\hline
\text{x = -2} & \text{y = 0} & \text{x = -5} & \text{y = 3} \\
\text{x = 0} & \text{y = -1} & \text{x = 5} & \text{y = -7} \\
\text{x = -2} & \text{y = -1} & \text{x = -8} & \text{y = -7} \\
\end{array}
\]
Predicate Abstraction

- Predicates \( \approx \) expressions over program variables
  - e.g., \( x > 0, x < y \)

\[ \begin{align*}
  x &= 0 \\
  y &= 0 \\
  x &= 5 \\
  y &= 3 \\
  x &= 0 \\
  y &= -1 \\
  x &= 5 \\
  y &= -7 \\
  x &= -2 \\
  y &= 0 \\
  x &= -5 \\
  y &= 3 \\
  x &= -2 \\
  y &= -1 \\
  x &= -8 \\
  y &= -7
\]
Predicate Abstraction

- Predicates \( \simeq \) expressions over program variables
  - e.g., \( x > 0, x < y \)

\[\text{Blocks define partition}\]

- Blocks
Predicate Abstraction

- Predicates \( \approx \) expressions over program variables
  - e.g., \( x > 0 \), \( x < y \)

\[\begin{align*}
\text{define} & \Rightarrow \text{partition} \\
\text{Blocks} & \Rightarrow \text{abstract model} \\
\text{stochastic game}
\end{align*}\]

\[\begin{align*}
x & \geq 0 \checkmark \quad y \geq 0 \checkmark \\
 x & = 0 \\
y & = 0 \\
x & = 5 \\
y & = 3
\end{align*}\]
Predicate Abstraction

- Predicates \( \equiv \) expressions over program variables
  - e.g., \( x > 0, x < y \)

\( \Rightarrow \) partition
  - Blocks

\( \Rightarrow \) abstract model
  - stochastic game

- reduce abstraction to \textit{satisfiability} of logical formulas
\( \Rightarrow \) implemented by \textit{SMT solver}
  - SMT = Satisfiability Modulo Theories
Example

- Consider program

```verbatim
module main
s : [0..2]; // control flow
x,y : int; // integer variables
[a] s=0 -> 1.0:(s'=1) & (x'=y);
[b] s=0 & x >10 -> 0.5:(s'=0) + 0.5:(s'=2);
endmodule
```

- predicates $s = 0, s = 1, s = 2, x = 0, x > 0, x < 0$

```
s = 0, x > 0
a

sl

b

s = 1, x < 0

s = 1, x = 0

s = 1, x > 0

s = 2, x > 0
```
Example

- Consider program

```
module main
  s  : [0..2]; // control flow
  x,y : int;  // integer variables
[a] s=0 -> 1.0:(s'=1) & (x'=y);
[b] s=0 & x>10 -> 0.5:(s'=0)+ 0.5:(s'=2);
endmodule
```

- predicates $s = 0, s = 1, s = 2, x = 0, x > 0, x < 0$

```
s = 0, x > 0
    a

s = 1, x < 0

s = 1, x = 0

s = 1, x > 0

s = 2, x > 0
```
Example

- Consider program

```plaintext
module main
  s : [0..2]; // control flow
  x, y : int; // integer variables
  [a] s=0 -> 1.0:(s'=1) & (x'=y);
  [b] s=0 & x>10 -> 0.5:(s'=0) + 0.5:(s'=2);
endmodule
```

- predicates $s = 0$, $s = 1$, $s = 2$, $x = 0$, $x > 0$, $x < 0$
Example

- Consider program

```plaintext
module main
    s : [0..2];  // control flow
    x, y : int;  // integer variables

[a] s=0 -> 1.0:(s' = 1) & (x' = y);
[b] s=0 & x > 10 -> 0.5:(s' = 0) + 0.5:(s' = 2);
endmodule
```

- predicates $s = 0, s = 1, s = 2, x = 0, x > 0, x < 0$
Example

- Consider program

```plaintext
module main
  s : [0..2];  // control flow
  x,y : int;   // integer variables
  [a] s=0 -> 1.0:(s'=1) & (x'=y);
  [b] s=0 & x>10 -> 0.5:(s'=0) + 0.5:(s'=2);
endmodule
```

- predicates $s = 0, s = 1, s = 2, x = 0, x > 0, x < 0$
Example

- Consider program

```plaintext
module main
    s  : [0..2]; // control flow
    x,y : int; // integer variables
    [a] s=0 -> 1.0:(s'=1) & (x'=y);
    [b] s=0 & x>10 -> 0.5:(s'=0) + 0.5:(s'=2);
endmodule
```

- Predicates $s = 0, s = 1, s = 2, x = 0, x > 0, x < 0$
Example

- Consider program

```plaintext
module main
    s : [0..2];  // control flow
    x, y : int;  // integer variables
    [a] s=0 -> 1.0:(s'=1) & (x'=y);
    [b] s=0 & x>10 -> 0.5:(s'=0) + 0.5:(s'=2);
endmodule
```

- predicates \( s = 0, s = 1, s = 2, x = 0, x > 0, x < 0 \)
Example

- Consider program

```plaintext
module main
    s : [0..2]; // control flow
    x, y : int; // integer variables
    s = 0 -> 1.0:(s' = 1) & (x' = y);
    s = 0 & x > 10 -> 0.5:(s' = 0) + 0.5:(s' = 2);
endmodule
```

- predicates $s = 0, s = 1, s = 2, x = 0, x > 0, x < 0$

![stochastic two-player game diagram]

```plaintext
stochastic two-player game
```

```plaintext
s = 0, x > 0

s = 1, x < 0
s = 1, x = 0
s = 1, x > 0

s = 2, x > 0
```

21 / 1
Wrap-up

- fully automatic and symbolic abstraction
  - for given **predicate** set
Wrap-up

- fully automatic and symbolic abstraction
  - for given predicate set

probabilistic program

semantics

predicate abstraction

MDP

abstract MDP
Wrap-up

- fully automatic and symbolic abstraction
  - for given **predicate** set
- ... but where do predicates come from?

(probabilistic program)

(MDP) → **predicate abstraction**

(abstract MDP) → semantics
Reachability Properties

\[ P_{\eta}^n (s_0 \leadsto e) < 0.03 \]
Abstraction Refinement

reachability probability

\[ P_{\eta}^{\#} \xrightarrow{\text{due to abstraction}} \]

refinement steps
Abstraction Refinement

Due to abstraction, the reachability probability decreases as the refinement steps increase.
Abstraction Refinement

Due to abstraction, the reachability probability decreases over refinement steps.
Abstraction Refinement

Reachability probability vs. refinement steps
Abstraction Refinement

\[ P_B^{\eta^\#}(\sim e) < p \]
Abstraction Refinement

\[ P^n_\eta (\sim \varepsilon) < p \]
Abstraction Refinement

\[ \mathcal{P}_B^{\eta^\#}(\sim e) < p \]
Property shown

\[ P^\eta_B(\sim \varepsilon) < p \]
Property shown

\[ \mathcal{P}_B^{\eta#}(\sim e) < p \]
Property refuted

\[ P_B^{\#}(\sim e) < p \]
Property refuted

\[ P^\eta_B (\sim e) < p \]

\[ \times \]
Inconclusive: refinement due
Inconclusive: refinement due to refinement steps

\[ P^{\eta \#}_{B}(\sim \square e) < p \]
Inconclusive: refinement due to refinement steps

\[ P_{B}^{\eta \#}(\rightsquigarrow \Sigma) < p \]
CEGAR: Counterexample-Guided Abstraction Refinement

- refinement technique in software model checking
  - SLAM project at Microsoft [Ball/Rajamani 2002,...]
  - Blast
  - ...

What is an abstract CE?

- pioneering work in probabilistic verification
CEGAR: Counterexample-Guided Abstraction Refinement

- refinement technique in software model checking
  - SLAM project at Microsoft [Ball/Rajamani 2002,...]
  - Blast
  - ...

What is an abstract CE?

- pioneering work in probabilistic verification
Counterexamples

Safety: Error State Unreachable

Probabilistic Reachability

Transition System
Counterexamples

Safety: Error State Unreachable

Probabilistic Reachability

Transition System

CE is a Path

“error state is reachable”
Counterexamples

Safety: Error State Unreachable

Probabilistic Reachability

Transition System

CE is a Path

“error state is reachable”
Counterexamples

Safety: Error State Unreachable

Transition System

CE is a Path

“error state is reachable”

Probabilistic Reachability

Stochastic Game

resolve nondet. choice
Counterexamples

Safety: Error State Unreachable

Transition System

CE is a Path

“error state is reachable”

Probabilistic Reachability

Stochastic Game

Markov Chain

“probability to reach error state = 1/3”
Counterexamples

Safety: Error State Unreachable

Transition System

CE is a Path

“error state is reachable”

Probabilistic Reachability

Stochastic Game

Markov Chain

“probability to reach error state = \frac{1}{3}”
Conventional counterexample analysis

• check if the abstract counterexample is realisable ...
Conventional counterexample analysis

- check if the abstract counterexample is realisable ...
  - if so, we’ve found a bug
Conventional counterexample analysis

- check if the abstract counterexample is **realisable** ...
  - if so, we’ve found a **bug**

- ... or **spurious**
  - abstraction too coarse, i.e., needs **refinement**
Conventional counterexample analysis

- check if the abstract counterexample is realisable ...
  - if so, we’ve found a bug

... or spurious
- abstraction too coarse, i.e., needs refinement

Implementation with SMT solver:
- convert path to formula
- path realisable $\iff$ formula satisfiable
- generate splitting predicate, e.g., by interpolation
Analysis of Probabilistic CE

- Is there a matching real counterexample?
  - replay decisions of abstract counterexample

abstract
CE

\[
\begin{align*}
  B_0 & \xrightarrow{c_0, u_1, \frac{1}{3}} B_1 \\
  & \xrightarrow{u_2, \frac{2}{3}} B_1 \\
  & \xrightarrow{c_1, u_4} B_2
\end{align*}
\]
Analysis of Probabilistic CE

- Is there a matching real counterexample?
  - replay decisions of abstract counterexample

abstract CE

- Challenge: CE correspond to many paths
  - Markov chain: cyclic & has probabilistic branching
- Goal 1: Leverage conventional counterexample analysis
  - path analysis based on SMT and interpolation
- Goal 2: avoid exploring too many paths
Probabilistic CEGAR

- enumerate paths of CE Markov chain
- visit paths with highest probability first [Han&Katoen 2007]
  - path $\sigma_1^\#$
    - if spurious generate predicate (interpolation)

abstract probability mass $\mathcal{P}^\eta_B(\leadsto e)$
Probabilistic CEGAR

- enumerate paths of CE Markov chain
- visit paths with highest probability first [Han&Katoen 2007]
  - path $\sigma_1^\#$
    - if spurious generate predicate (interpolation)
  - path $\sigma_2^\#$

abstract probability mass $\mathcal{P}_B^{\eta^\#} (\rightsquigarrow e)$
Probabilistic CEGAR

- enumerate paths of CE Markov chain
- visit paths with highest probability first [Han & Katoen 2007]
  - path $\sigma_1^\#$
    - if spurious generate predicate (interpolation)
  - path $\sigma_2^\#$
  - ...

Abstract probability mass $\mathcal{P}_{B}^{\eta^\#}(\sim e)$
Probabilistic CEGAR

- enumerate paths of CE Markov chain
- visit paths with highest probability first [Han & Katoen 2007]
  - path $\sigma_1^\#$
    - if spurious generate predicate (interpolation)
  - path $\sigma_2^\#$
  - ...

abstract probability mass $P_{B}^{\eta^\#}(\sim e)$
Probabilistic CEGAR

- enumerate paths of CE Markov chain
- visit paths with highest probability first [Han & Katoen 2007]
  - path $\sigma_1$
    - if spurious generate predicate (interpolation)
  - path $\sigma_2$
  - ...

abstract probability mass $P_B^{\eta}(\rightsquigarrow e)$
Probabilistic CEGAR

- enumerate paths of CE Markov chain
- visit paths with highest probability first [Han & Katoen 2007]
  - path $\sigma_1^*$
    - if spurious generate predicate (interpolation)
  - path $\sigma_2^*$
  - ...
- realisable probability mass $> p$?

abstract probability mass $\mathcal{P}_{\eta}^B (\sim \varepsilon)$
Computing realisable probability is harder than it seems

realisable probability ≠ \( \sum \{ P(\sigma^\#) \mid \text{realizable path} \} = 1 \)
Computing realisable probability is harder than it seems

\[
\frac{2}{3} = \text{realisable probability} \neq \sum \{ P(\sigma^\#) \mid \text{realizable path} \} = 1
\]

\[P_{s_0}(\sim \rightarrow e) = \frac{2}{3}\]

\[P_{s_1}(\sim \rightarrow e) = \frac{1}{3}\]
Computing realisable probability is harder than it seems

\[ \frac{2}{3} = \text{realisable probability} \neq \sum \{ P(\sigma) \mid \text{realizable path} \} = 1 \]

Theorem (Realisable probability)

**Realisable probability as optimisation problem:**

\[ \text{MaxSmt}(exp_1, \ldots, exp_n) = \max \{ \sum_{i=1}^{n} [exp_i]_s \cdot p_i \mid s \in [I \land F(B)] \} \]

where \( exp_i \) characterizes path \( \sigma_i \)
Probabilistic CEGAR

- analyse paths in Markov chain with decreasing probability
  - spurious paths give predicates
  - realizable paths can improve lower bound
    - MaxSMT

\[ \begin{align*}
\sigma_1^\# & \quad \sigma_2^\# & \quad \sigma_3^\# & \quad \sigma_4^\# & \quad \sigma_5^\# \\
\end{align*} \]

\[ \mathcal{P}_{B}^{\eta^\#}(C) \]

\[ \text{MaxSMT}(C_{\text{real}}) \]
Probabilistic CEGAR

- analyse paths in Markov chain with decreasing probability
  - spurious paths give predicates
  - realizable paths can improve lower bound
    - MaxSMT

\[ \sigma_1^\# \quad \sigma_2^\# \quad \sigma_3^\# \quad \sigma_4^\# \quad \sigma_5^\# \]

\[ \mathcal{P}_{B}^{\eta^\#}(C) \quad \text{MaxSMT}(C_{\text{real}}) \]

- Semi-decision procedure for probabilistic CE analysis
  - always terminates returning either
    - \( \times \) CE realizable
    - \( \checkmark \) CE spurious and predicate
    - ? don’t know and predicate
  - incomplete to ensure termination
    - limit on number of spurious paths
Justification for Incompleteness: Undecidability

• A CE analysis problem consists of
  • probabilistic program $M$
  • threshold $p$
  • abstraction $G$ of $M$
  • CE $(B, \eta^\#)$ in $G$, i.e., $\mathcal{P}_{B}^{\eta^\#}(\leadsto e) > p$

• decide if the CE is real
  • there is a corresponding concrete CE $(s, \eta)$ with $\mathcal{P}_{s}^{\eta}(\leadsto e) > p$
Justification for Incompleteness: Undecidability

- A CE analysis problem consists of
  - probabilistic program $M$
  - threshold $p$
  - abstraction $G$ of $M$
  - CE $(B, \eta^\#)$ in $G$, i.e., $\mathcal{P}_B^{\eta^\#}(\sim e) > p$

- **decide** if the CE is real
  - there is a corresponding concrete CE $(s, \eta)$ with $\mathcal{P}_s^\eta(\sim e) > p$

- **assume**: expressions language $=$ linear arithmetic over the integers
  $\Rightarrow$ conventional counterexample analysis decidable
Justification for Incompleteness: Undecidability

- A CE analysis problem consists of
  - probabilistic program $M$
  - threshold $p$
  - abstraction $G$ of $M$
  - CE $(B, \eta^\#)$ in $G$, i.e., $P_{B}^{\eta^\#} (\sim e) > p$

- **decide** if the CE is real
  - there is a corresponding concrete CE $(s, \eta)$ with $P_{s}^{\eta} (\sim e) > p$

- assume: expressions language $=$ linear arithmetic over the integers
  $\Rightarrow$ conventional counterexample analysis decidable

**Theorem (Undecidability of Counterexample Analysis)**

*Counterexample analysis for probabilistic programs is undecidable*

**Proof.**

Halting problem for counter machines can be reduced to CE analysis
Backward Refinement (VMCAI 2010)

- Refinement if no threshold is available
- Target uncertainty from abstraction (leverage game strategies)
The PASS tool

program → parser → property

predicate abstraction → game → prob. reachability

refinement
Case Study: BRP

- Probability to reach:
  “receiver does not receive any chunk and sender tried to send a chunk”
- Can be analyzed for an infinite parameter range with PASS
  - for any file size $\geq 16$, probability is $1.6E - 7$
- PASS provides proof for arbitrary file size
- BRP is just one case study:

<table>
<thead>
<tr>
<th>Case study (parameters)</th>
<th>Property</th>
<th>Conventional</th>
<th>Abstraction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>states</td>
<td>trans</td>
</tr>
<tr>
<td>WLAN (BOFF T)</td>
<td>k=3</td>
<td>5,195K</td>
<td>11,377K</td>
</tr>
<tr>
<td></td>
<td>k=6</td>
<td>12,616K</td>
<td>28,137K</td>
</tr>
<tr>
<td></td>
<td>k=6</td>
<td>12,616K</td>
<td>28,137K</td>
</tr>
<tr>
<td>CSMA/CD (BOFF)</td>
<td>p1</td>
<td>41K</td>
<td>52K</td>
</tr>
<tr>
<td></td>
<td>p2</td>
<td>124K</td>
<td>161K</td>
</tr>
<tr>
<td></td>
<td>p2</td>
<td>41K</td>
<td>52K</td>
</tr>
<tr>
<td></td>
<td>p2</td>
<td>124K</td>
<td>161K</td>
</tr>
<tr>
<td>BRP (N MAX)</td>
<td>p1</td>
<td>2K</td>
<td>3K</td>
</tr>
<tr>
<td></td>
<td>p1</td>
<td>5K</td>
<td>7K</td>
</tr>
<tr>
<td></td>
<td>p1</td>
<td>10K</td>
<td>14K</td>
</tr>
<tr>
<td></td>
<td>p4</td>
<td>$\infty$</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>p4</td>
<td>$\infty$</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>p4</td>
<td>$\infty$</td>
<td>–</td>
</tr>
<tr>
<td>SW</td>
<td>goodput</td>
<td>$\infty$</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>timeout</td>
<td>$\infty$</td>
<td>–</td>
</tr>
</tbody>
</table>
Why abstraction works.

- absolute values do matter but only, e.g., differences between variables
- cannot contribute to optimum
Contributions

a novel analysis method for probabilistic programs:

- **symbolic abstraction** to tackle large state spaces
  - **Première**: predicate abstraction in probabilistic verification
    - prior work in qualitative software verification
  - **Challenge**

- **refinement** to achieve full automation
  - **Première**: Probabilistic CEGAR
    - prior work: CEGAR in qualitative software verification
  - **Challenge**
    - counterexamples are Markov chains

- implemented in **PASS** tool
Limitations

- Abstraction is not a panacea / silver bullet
  - can be less efficient for certain finite-state models
- Probabilistic CEGAR:
  - lower thresholds for minimal reachability?
- No support for state-dependent probabilities:

\[
\![ \text{m=1 \& x>0} \rightarrow \frac{1}{x} : (x' = x - 1) + \frac{x-1}{x} : (m' = 3); \]


Avenues for Future Work

• Beyond probabilistic reachability for MDPs
  • rewards and expectations
  • Exponential distributions
  • Support for full PCTL

• Richer input language
  • Modest
Thesis Statement

Abstraction enables automatic verification of probabilistic programs with large and, for the first time, infinite state spaces.