Quantitative Abstraction Refinement

Dave Parker
Oxford University Computing Laboratory

Joint work with:
Mark Kattenbelt, Marta Kwiatkowska, Gethin Norman

PEPA Club, University of Edinburgh, November 2009
Probabilistic model checking

**Probabilistic model**
e.g. Markov chain

**Probabilistic model checker**
e.g. PRISM

**Probabilistic temporal logic specification**
e.g. PCTL, CSL, LTL

**Result**

\[ P < 0.01 \ [ F \text{ error} ] \]

Quantitative results
Overview

- **Probabilistic model checking**
  - discrete-time Markov chains (DTMCs)
  - Markov decision processes (MDPs)
  - probabilistic reachability

- **Abstraction for probabilistic models**
  - abstractions of DTMCs (MDPs)
  - abstractions of MDPs (two-player stochastic games)

- **Quantitative abstraction refinement**
  - abstraction-refinement loop
  - verification of probabilistic software, real-time systems

- **Current/future work & Conclusions**
Probabilistic models

- **Discrete–time Markov chains (DTMCs)**
  - discrete states, discrete probability distributions

- **Markov decision processes (MDPs)**
  - discrete states, probability and nondeterminism
  - uses: concurrency, under–specification, abstraction
  - models: randomised distributed algorithms, randomised communication protocols, security protocols, …

- **Continuous–time Markov chains (CTMCs)**
  - discrete states, exponentially distributed delays
  - performance modelling, biological reaction systems, …
Discrete–time Markov chains (DTMCs)

- **Model fully probabilistic behaviour**
  - state–transition systems augmented with probabilistic choice

- **Transition probability matrix**
  - over state space $S$
  - $P : S \times S \to [0,1]$

- **Paths through a DTMC**
  - finite/infinite state sequences
  - $\text{Path}(s) = \text{set of all (infinite) paths from state } s$
  - define probability space $\Pr_s$ over $\text{Path}(s)$

- **Probabilistic reachability (for a set of goal states $F \subseteq S$)**
  - probability of reaching $F$ from state $s$
  - $p_s(F) = \Pr_s \{ s_0s_1s_2s_3\ldots \in \text{Path}(s) \mid s_i \in F \text{ for some } i \}$
  - reduces to solution of a linear equation system
Markov decision processes (MDPs)

- Model nondeterministic as well as probabilistic behaviour
  - extension of discrete-time Markov chains
  - nondeterministic choice between probability distributions

- Transition probability function
  - Steps : $S \rightarrow 2^{Act \times \text{Dist}(S)}$
  - maps states to non-empty sets of action–probability distribution pairs

- A (finite or infinite) path through an MDP
  - is a sequence of (connected) states
  - represents an execution of the system
  - resolves both the probabilistic and nondeterministic choices
Strategies of MDPs

- A strategy (aka. “adversary”, “scheduler”, …) of an MDP
  - is a resolution of nondeterminism only
  - is (formally) a mapping from finite paths to distributions

- E.g. strategy that picks $b$ then $c$ in $s_1$
  - $\sigma(s_0s_1) = (b, \mu_b)$, $\sigma(s_0s_1s_1) = (c, \mu_c)$,
    $\sigma(s_0s_1s_0s_1) = (c, \mu_c)$

- A strategy results in a fully probabilistic model
  - i.e. an (infinite-state) DTMC over finite paths
  - on which we can define a probability space over infinite paths

- Strategy $\sigma$ is simple (memoryless) iff $\sigma(s_1...s_n) = \sigma(s_n)$
  - in this case, resulting model reduces to finite Markov chain
Example strategy

• Fragment of DTMC for strategy which picks b then c in $s_1$
Probabilistic reachability for MDPs

- **A strategy** $\sigma$ **induces**, for each state $s$ in the MDP:
  - a set of infinite paths $\text{Path}^\sigma(s)$
  - a probability space $\text{Pr}^\sigma_s$ over $\text{Path}^\sigma(s)$

- **Probabilistic reachability** (for a set of goal states $F \subseteq S$)
  - probability of reaching $F$ from state $s$ under strategy $\sigma$
  - $p^\sigma_s(F) = \text{Pr}^\sigma_s \{ s_0s_1s_2s_3... \in \text{Path}^\sigma(s) \mid s_i \in F$ for some $i \}$

- **Minimum/maximum probabilities** over all strategies
  - $p^\text{min}_s(F) = \inf_\sigma p^\sigma_s(F)$
  - $p^\text{max}_s(F) = \sup_\sigma p^\sigma_s(F)$
  - simple strategies suffice

- **Used to reason about best/worst–case behaviour**
  - e.g. “maximum probability of an error occurring”
Probabilistic model checking for MDPs

- **Also: Bounded reachability properties**
  - e.g. “min. probability of algorithm termination within T steps”

- **Also: Cost– and reward–based properties**
  - augment states/transitions of MDP with real–valued costs
  - define properties as random variables over \( \text{Path}^\sigma(s) \)
  - e.g. “max. expected power consumption for the duration of the protocol”

- **Probabilistic temporal logics**
  - e.g. PCTL extends CTL

- **We focus on quantitative analysis**
  - i.e. just compute \( p_s^{\min}(F) \) and \( p_s^{\max}(F) \)
  - useful to spot patterns/trends
Probabilistic model checking for MDPs

- **Probabilistic reachability is efficiently computable**
  - linear optimisation problem (polynomial complexity)
  - or value iteration (dynamic programming) – simple iterative numerical method; more efficient in practice
  - *best/worst* case simple strategy can also be generated

- **Mature tool support exists, e.g. PRISM**
  - efficient (e.g. symbolic) implementations
  - successful application to wide range of application domains

- **But major challenges remain, e.g.**
  - state-space explosion
  - automating model extraction
Overview

• Probabilistic model checking
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)
  – probabilistic reachability

• Abstraction for probabilistic models
  – abstractions of DTMCs (MDPs)
  – abstractions of MDPs (two–player stochastic games)

• Quantitative abstraction refinement
  – abstraction–refinement loop
  – verification of probabilistic software, real–time systems

• Current/future work & Conclusions
Abstraction

- Very successful in (non-probabilistic) model checking
  - essential for verification of large/infinite-state systems

- Construct abstract model $A$ of concrete model $M$
  - details not relevant to some property of interest removed
  - e.g. partition of state space based on a set of predicates

- Non-probabilistic case: existential abstraction
  - conservative: existence of path in $M$ implies existence in $A$
  - hence can model check $A$ to verify safety properties of $M$

- Abstraction refinement
  - automate process of constructing abstraction
  - start with simple coarse abstraction, then iteratively refine
Abstraction + CEGAR

- **Counterexample-guided abstraction refinement**
  - (non-probabilistic) model checking of reachability properties
Abstraction + CEGAR

- Counterexample-guided abstraction refinement
  - (non-probabilistic) model checking of reachability properties
Abstraction of DTMCs

- We use MDPs as abstractions of DTMCs
  - based on a partition $A$ of the (concrete) state space $S$
  - i.e. each element $a \in A$ is an abstract state

- Analysis of MDP yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

\[
p_a^{\min}(F) \leq p_s(F) \leq p_a^{\max}(F)
\]
• Partition of (concrete) state space gives abstract states

Concrete model (DTMC)
Partition of (concrete) state space gives abstract states
DTMC $\rightarrow$ MDP

- Distributions are lifted to the abstract state space

Concrete model (DTMC)
**DTMC → MDP**

- Choices in abstract states become choices in MDP

Concrete model (DTMC)  Abstraction (MDP)
Abstraction of MDPs

- Abstraction increases degree of nondeterminism
  - i.e. minimum probabilities are lower and maximums higher

  ![Diagram showing minimum and maximum probabilities](image)

  - what form does the abstraction of an MDP take?

- Our approach: two-player stochastic games [QEST'06]
- Key idea: separate two forms of nondeterminism
  - (a) from abstraction and (b) from original MDP
  - then generate separate lower/upper bounds for min/max

  ![Diagram showing separate bounds for minimum and maximum probabilities](image)
Two-player stochastic games

- **Subclass of simple stochastic games** [Shapley], [Condon]
  - two nondeterministic players and probabilistic choice
  - vertices $V$ of game partitioned into $V_1$ and $V_2$ (player 1/2 vertices)
  - player 1 has choices between $V_2$ vertices
  - player 2 has choices between distributions over $V_1$ vertices

- **A resolution of the nondeterminism in the game**
  - corresponds to a pair of strategies for players 1 and 2: $(\sigma_1,\sigma_2)$
  - $p_{v}^{\sigma_1,\sigma_2}(F) = \text{probability of reaching } F \text{ from } v \text{ under } (\sigma_1,\sigma_2)$

- **Optimal reachability probabilities for player 1 and player 2**
  - informally: the maximum probability of reaching $F$ a player can guarantee, no matter what the other player does
  - formally: $\sup_{\sigma_1} \inf_{\sigma_2} p_{v}^{\sigma_1,\sigma_2}(F)$ and $\sup_{\sigma_2} \inf_{\sigma_1} p_{v}^{\sigma_1,\sigma_2}(F)$
  - computable (and simple strategies) with value iteration
Games as abstractions of MDPs

- Abstraction of an MDP is a two-player stochastic game
  - based on a partition $A$ of the concrete state space
- where:
  - player 1 controls the nondeterminism of the abstraction
  - player 2 controls the nondeterminism of the MDP

- Player 1 vertices: abstract states
  - elements of partition $A$

- Player 2 vertices: choices of MDP lifted to abstract states
  - sets of distributions over $A$
  - or equivalently, sets of concrete states with the same abstract choices
MDP $\rightarrow$ Game
MDP $\rightarrow$ Game

- Player 1 vertices are partition elements (abstract states)
MDP → Game

- (Sets of) distributions are lifted to the abstract state space
MDP → Game

• States with same (sets of) choices form player 2 vertices
• Complete transformation:

Concrete model (MDP)  Abstraction (game)
Analysis of the abstraction

- Analysis of game yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

\[
\inf_{\sigma_1,\sigma_2} p_{v}^{\sigma_1,\sigma_2}(F) \leq p_s^{\min}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_{v}^{\sigma_1,\sigma_2}(F)
\]

\[
\sup_{\sigma_2} \inf_{\sigma_1} p_{v}^{\sigma_1,\sigma_2}(F) \leq p_s^{\max}(F) \leq \sup_{\sigma_1,\sigma_2} p_{v}^{\sigma_1,\sigma_2}(F)
\]
Analysis of the abstraction

- Analysis of game yields lower/upper bounds:

  for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

\[
\inf_{\sigma_1,\sigma_2} p_{a}^{\sigma_1,\sigma_2}(F) \leq p_{s}^{\min}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_{a}^{\sigma_1,\sigma_2}(F)
\]

\[
\sup_{\sigma_2} \inf_{\sigma_1} p_{a}^{\sigma_1,\sigma_2}(F) \leq p_{s}^{\max}(F) \leq \sup_{\sigma_1,\sigma_2} p_{a}^{\sigma_1,\sigma_2}(F)
\]

min/max reachability probabilities for original MDP
Analysis of the abstraction

- Analysis of game yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

$$\inf_{\sigma_1, \sigma_2} p_{a^{\sigma_1, \sigma_2}}(F) \leq p_s^{\min}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_{a^{\sigma_1, \sigma_2}}(F)$$

$$\sup_{\sigma_2} \inf_{\sigma_1} p_{a^{\sigma_1, \sigma_2}}(F) \leq p_s^{\max}(F) \leq \sup_{\sigma_1, \sigma_2} p_{a^{\sigma_1, \sigma_2}}(F)$$

Optimal probabilities for player 1, player 2 in game
Analysis of the abstraction

- Analysis of game yields lower/upper bounds:
  - for target $F \in A$, $s \in S$ and $a \in A$ with $s \in a$

\[
\inf_{\sigma_1, \sigma_2} p_{a(\sigma_1, \sigma_2)}(F) \leq p_s^{\text{min}}(F) \leq \sup_{\sigma_1} \inf_{\sigma_2} p_{a(\sigma_1, \sigma_2)}(F)
\]

\[
\sup_{\sigma_2} \inf_{\sigma_1} p_{a(\sigma_1, \sigma_2)}(F) \leq p_s^{\text{max}}(F) \leq \sup_{\sigma_1, \sigma_2} p_{a(\sigma_1, \sigma_2)}(F)
\]

min/max reachability probabilities, treating game as MDP (i.e. assuming that players 1 and 2 cooperate)
Abstraction: Results

- Israeli & Jalfon’s Self Stabilisation [IJ90]
  - protocol for obtaining a stable state in a token ring
  - minimum probability of reaching a stable state by time $T$

![Graph showing minimum probability stabilised by time $T$](image)

- Concrete states: 1,048,575
- Abstract states: 627
Abstraction: Results

- IPv4 Zeroconf [CAG02]
  - protocol for obtaining an IP address for a new host
  - maximum probability the new host not configured by $T$

![Graph showing the relationship between maximum probability not configured by $T$ and time $T$. The graph includes lines for upper bound, actual value, and lower bound.]

- Concrete states: 838,905
- Abstract states: 881
Abstraction: Results

- **Sliding Window Protocol**
  - protocol for sending data over an insecure medium
  - maximum probability of $K$ timeouts

<table>
<thead>
<tr>
<th>$D$</th>
<th>Concrete States</th>
<th>Abstract States</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>189,952</td>
<td>742</td>
</tr>
<tr>
<td>10</td>
<td>987,136</td>
<td>964</td>
</tr>
<tr>
<td>20</td>
<td>??</td>
<td>2,074</td>
</tr>
</tbody>
</table>
Overview

• Probabilistic model checking
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)
  – probabilistic reachability

• Abstraction for probabilistic models
  – abstractions of DTMCs (MDPs)
  – abstractions of MDPs (two-player stochastic games)

• **Quantitative abstraction refinement**
  – abstraction–refinement loop
  – verification of probabilistic software, real–time systems

• Current/future work & Conclusions
Abstraction refinement

- Consider (max) difference between lower/upper bounds
  - gives a quantative measure of the abstraction’s precision

- If the difference ("error") is too great, refine the abstraction
  - a finer partition yields a more precise abstraction
  - model checking results (bounds/strategies) guide refinement
Abstraction refinement

- Probabilistic reachability yields *simple* strategies

- Consider abstract state \( a \) with “error” 0.2:

  - Abstract state \( a \) represents a set of concrete states
    - each choice in \( a \) corresponds to a *subset* of these

- Refine (split) \( a \) accordingly:
  - “strategy-based” refinement or “value-based” refinement
Abstraction–refinement loop

- **Quantitative abstraction–refinement loop for MDPs**

Diagram:
- Initial partition
- Abstraction
- Bounds and strategies
- Return bounds
- New partition

Arrows:
- Abstract
- Model check
- Refine
- Error $\geq \epsilon$
- Error $< \epsilon$
Abstraction–refinement loop

- **Quantitative abstraction–refinement loop for MDPs**

![Diagram]

- Refinements yield strictly finer partition
- Guaranteed to converge for finite models
- Guaranteed to converge for infinite models with finite bisimulation
Abstraction–refinement loop

- Implementations of quantitative abstraction refinement...

  1. Explicit–state prototype implementation
     - confirms viability of game–based abstraction/refinement

  2. Compositional abstraction of PRISM models [QAPL’08]
     - efficient construction of (manually specified) abstractions

  3. Verification of probabilistic software [VMCAI’09]
     - (predicate) abstraction/refinement of infinite–state MDPs

  4. Verification of probabilistic timed automata [FORMATS’09]
     - (DBM–based) abstraction/refinement of infinite–state MDPs
Probabilistic software

- Consider sequential ANSI C programs
  - support functions, pointers, arrays, but not dynamic memory allocation, unbounded recursion, floating point operations
  - probabilistic functions (for failures, randomisation)
  - nondeterministic functions (e.g. to abstract system calls)
  - infinite-state MDP semantics

- Quantitative properties based on probabilistic reachability
  - e.g. maximum probability of unsuccessful data transmission
  - e.g. minimum expected number of packets sent

- Prototype tool qprover
  - components from PRISM (model checking of stochastic games)
  - components from SATABS (predicate abstraction, SAT solvers)
  - analysed Linux network utilities (ping, tftp) – approx 1KLOC
Example – sample target program

```
bool fail = false;
int c = 0;
int main ()
{
    // nondeterministic
    c = num_to_send ();
    while (! fail && c > 0)
    {
        // probabilistic
        fail = send_msg ();
        c --;
    }
}
```

Φ: “what is the minimum/maximum probability of the program terminating with fail being true?”
Abstraction-refinement loop

- Probabilistic program
  - Abstraction (based on SAT)
  - Model extraction
- Boolean probabilistic program
- Abstraction (game)
  - Model construction
- Predicates
  - [error $\geq \varepsilon$
  - Refinement (weakest precondition)
- Bounds and strategies
  - [error $< \varepsilon$
- Return bounds

Software verification abstraction-refinement loop [VMCAI’09]
Verification of PTAs

• Verification of probabilistic timed automata (PTAs)
  – discrete states, discrete probabilistic choice, nondeterminism and real-valued clocks
  – infinite-state MDP semantics

• Timed automata + probabilities

• PRISM modelling language + clocks

```
module M
  s : [0..3];
  x : clock;
  [a] s=0 & x<10 → (s’=1);
  [b] s=1 → 0.5:(s’=2) + 0.5:(s’=3)&(x’=0);
endmodule
```
Verification of PTAs

- **Finite-state (game) abstractions using zones**
  - efficiently represented as difference bound matrices (DBMs)

- **Initial abstraction from forwards reachability**
  - subsequent refinements through zone splitting

- **Properties**: 
  - guaranteed to converge in finite time for any $\epsilon \geq 0$
  - i.e. exact verification of PTAs

- **Outperforms existing PTA verification techniques**
  - on several large case studies
Overview

• Probabilistic model checking
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)
  – probabilistic reachability

• Abstraction for probabilistic models
  – abstractions of DTMCs (MDPs)
  – abstractions of MDPs (two-player stochastic games)

• Quantitative abstraction refinement
  – abstraction-refinement loop
  – verification of probabilistic software, real-time systems

• Current/future work & Conclusions
We can consider a general class of “nondeterministic” abstractions for probabilistic models

<table>
<thead>
<tr>
<th>Concrete model:</th>
<th>Abstraction:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTMC</td>
<td>MDP</td>
</tr>
<tr>
<td>MDP</td>
<td>TPSG</td>
</tr>
<tr>
<td>CTMC</td>
<td>CTMDP</td>
</tr>
<tr>
<td>CTMDP</td>
<td>CTTPSG</td>
</tr>
</tbody>
</table>

• CTMDP = continuous-time Markov decision process
• CTTPSG = continuous-time two-player stochastic game
Abstraction for CTMCs

• A useful subset of CTMC properties are untimed
  – unbounded probabilistic reachability
  – steady-state (long-run) probabilities
  – so can abstract embedded DTMC as an MDP

• More interesting: time-bounded (transient properties)
  – can abstract CTMCs as CTMDPs, yielding lower/upper bounds
  – [Smith], [Katoen et al.], …
  – moreover, abstractions are uniform CTMDPs

• But: How to refine?
  – time-bounded reachability does not yield simple strategies

• How to build abstraction? What high-level model?
Conclusions

- **Abstraction approach for probabilistic models**
  - DTMCs abstracted as MDPs
  - MDPs abstracted as two-player stochastic games
  - abstraction yields lower/upper bounds on probabilities

- **Quantitative abstraction refinement**
  - bounds give quantitative measure of utility of abstraction
  - bounds/strategies can be used to guide refinement
  - quantitative abstraction–refinement loop (for error < $\varepsilon$)
  - fully automatic generation of abstraction
  - works in practice: probabilistic software & timed automata

- **Current & future work**
  - CTMCs, timed properties
  - probabilistic/stochastic hybrid systems
  - improved refinement heuristics, imprecise abstractions