An abstraction framework for mixed non-deterministic and probabilistic systems

Michael Huth, Computing, Imperial College London.

Acknowledgments: Christel Baier, Marta Kwiatkowska, Roberto Segala, and many more.

Objectives of this survey

1. motivate utility of mixed models for design and analysis activities;

Objectives of this survey

1. motivate utility of mixed models for design and analysis activities;
2. motivate need for abstraction in such activities;

Objectives of this survey

1. motivate utility of mixed models for design and analysis activities;
2. motivate need for abstraction in such activities;
3. present design and analysis of mixed models as model-checking problems (MCPs);

Objectives of this survey

1. motivate utility of mixed models for design and analysis activities;
2. motivate need for abstraction in such activities;
3. present design and analysis of mixed models as model-checking problems (MCPs);
4. propose a specification of abstraction for MCPs;

Objectives of this survey

1. motivate utility of mixed models for design and analysis activities;
2. motivate need for abstraction in such activities;
3. present design and analysis of mixed models as model-checking problems (MCPs);
4. propose a specification of abstraction for MCPs;
5. present main abstraction techniques for mixed models as instances of this specification; and

Objectives of this survey

1. motivate utility of mixed models for design and analysis activities;
2. motivate need for abstraction in such activities;
3. present design and analysis of mixed models as model-checking problems (MCPs);
4. propose a specification of abstraction for MCPs;
5. present main abstraction techniques for mixed models as instances of this specification; and
6. formulate research challenges for abstracting MCPs for mixed models.

Outline of talk

- specification of MCPs (notably Markov decision processes);

Outline of talk

- specification of MCPs (notably Markov decision processes);
- motivate need for abstract MCPs;

Outline of talk

- specification of MCPs (notably Markov decision processes);
- motivate need for abstract MCPs;
- survey forms of abstraction for MCPs;

Outline of talk

- specification of MCPs (notably Markov decision processes);
- motivate need for abstract MCPs;
- survey forms of abstraction for MCPs;
- discuss query equivalences for MCPs;

Outline of talk

- specification of MCPs (notably Markov decision processes);
- motivate need for abstract MCPs;
- survey forms of abstraction for MCPs;
- discuss query equivalences for MCPs;
- define approximation problem for abstract MCPs;

Outline of talk

- specification of MCPs (notably Markov decision processes);
- motivate need for abstract MCPs;
- survey forms of abstraction for MCPs;
- discuss query equivalences for MCPs;
- define approximation problem for abstract MCPs;
- stress need for supporting diagnostics;

Outline of talk

- specification of MCPs (notably Markov decision processes);
- motivate need for abstract MCPs;
- survey forms of abstraction for MCPs;
- discuss query equivalences for MCPs;
- define approximation problem for abstract MCPs;
- stress need for supporting diagnostics;
- lists caveats for effective use of abstract MCPs; and

Outline of talk

- specification of MCPs (notably Markov decision processes);
- motivate need for abstract MCPs;
- survey forms of abstraction for MCPs;
- discuss query equivalences for MCPs;
- define approximation problem for abstract MCPs;
- stress need for supporting diagnostics;
- lists caveats for effective use of abstract MCPs; and
- present research challenges for abstract MCPs of mixed models.

Model checking problem (MCP)

A MCP P is a tuple

$$(\mathcal{M}, \Phi, \mathtt{Ans}, \models)$$

Model checking problem (MCP)

A MCP P is a tuple

$$(\mathcal{M}, \Phi, \texttt{Ans}, \models)$$

- where $\mathcal{M}$ is a class of models ($M \in \mathcal{M}$);

Model checking problem (MCP)

A MCP P is a tuple

$$(\mathcal{M}, \Phi, \texttt{Ans}, \models)$$

- where $\mathcal{M}$ is a class of models ($M \in \mathcal{M}$);
- Φ a class of queries about such models ($\phi \in \Phi$);

Model checking problem (MCP)

A MCP P is a tuple

$$(\mathcal{M}, \Phi, \mathtt{Ans}, \models)$$

- where $\mathcal{M}$ is a class of models ($M \in \mathcal{M}$);
- Φ a class of queries about such models ($\phi \in \Phi$);
- $\mathtt{Ans}$ a partial order of answers to such queries ($a \in \mathtt{Ans}$); and

Model checking problem (MCP)

A MCP P is a tuple

$$(\mathcal{M}, \Phi, \texttt{Ans}, \models)$$

- where $\mathcal{M}$ is a class of models ($M \in \mathcal{M}$);
- Φ a class of queries about such models ($\phi \in \Phi$);
- $\texttt{Ans}$ a partial order of answers to such queries ($a \in \texttt{Ans}$); and
- $\models: \mathcal{M} \times \Phi \to \texttt{Ans}$ is a specification of how to compute answers to queries over models.

Markov decision processes...

... determine several MCPs. Here is one:

Markov decision processes...

... determine several MCPs. Here is one:

- $\mathcal{M}$ is the class of Markov decision processes;

Markov decision processes...

... determine several MCPs. Here is one:

- $\mathcal{M}$ is the class of Markov decision processes;
- Φ is linear-time temporal logic (LTL);

Markov decision processes...

... determine several MCPs. Here is one:

- $\mathcal{M}$ is the class of Markov decision processes;
- Φ is linear-time temporal logic (LTL);
- the partial order $\mathtt{Ans}$ is the interval domain

$$\mathcal{I} = \{[r, s] \mid 0 \leq r \leq s \leq 1\}$$

ordered by $[r, s] \leq [r', s']$ iff $r' \leq r$ and $s \leq s'$;

Markov decision processes...

... determine several MCPs. Here is one:

- $\mathcal{M}$ is the class of Markov decision processes;
- Φ is linear-time temporal logic (LTL);
- the partial order `Ans` is the interval domain

$$\mathcal{I} = \{[r, s] \mid 0 \leq r \leq s \leq 1\}$$

  ordered by $[r, s] \leq [r', s']$ iff $r' \leq r$ and $s \leq s'$;
- the answer $[r, s]$ to a model check $(M \models \phi)$ is specified to represent
  - r: the worst-case probability of M satisfying ϕ;
  - s: the best-case probability of M satisfying ϕ.

Markov decision processes...

... determine several MCPs. Here is one:

- $\mathcal{M}$ is the class of Markov decision processes;
- Φ is linear-time temporal logic (LTL);
- the partial order `Ans` is the interval domain

$$\mathcal{I} = \{[r, s] \mid 0 \leq r \leq s \leq 1\}$$

  ordered by $[r, s] \leq [r', s']$ iff $r' \leq r$ and $s \leq s'$;
- the answer $[r, s]$ to a model check $(M \models \phi)$ is specified to represent
  - r : the worst-case probability of M satisfying ϕ;
  - s : the best-case probability of M satisfying ϕ.

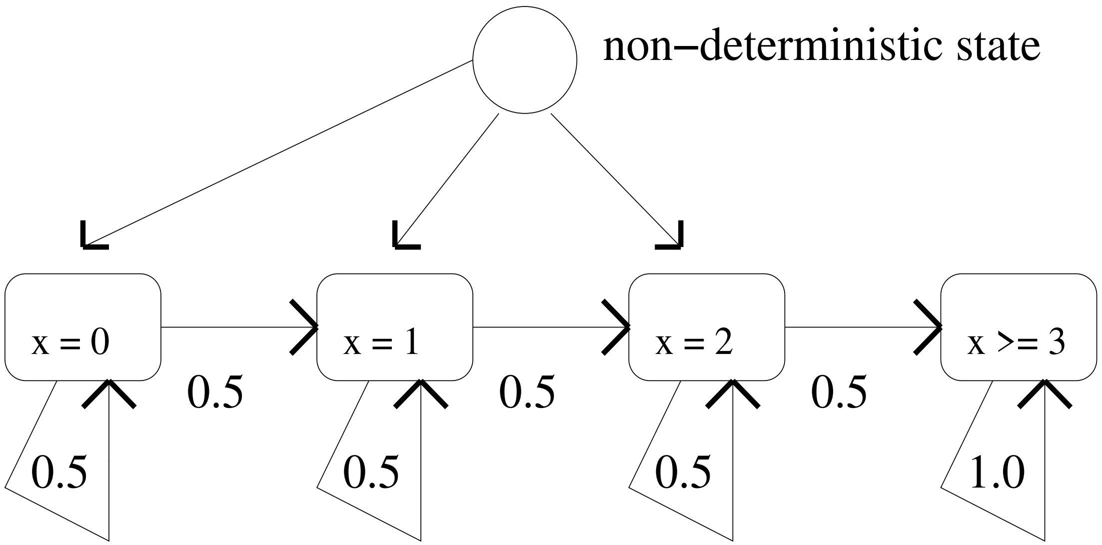
We denote this MCP by MDP subsequently.

An abstraction in MDP

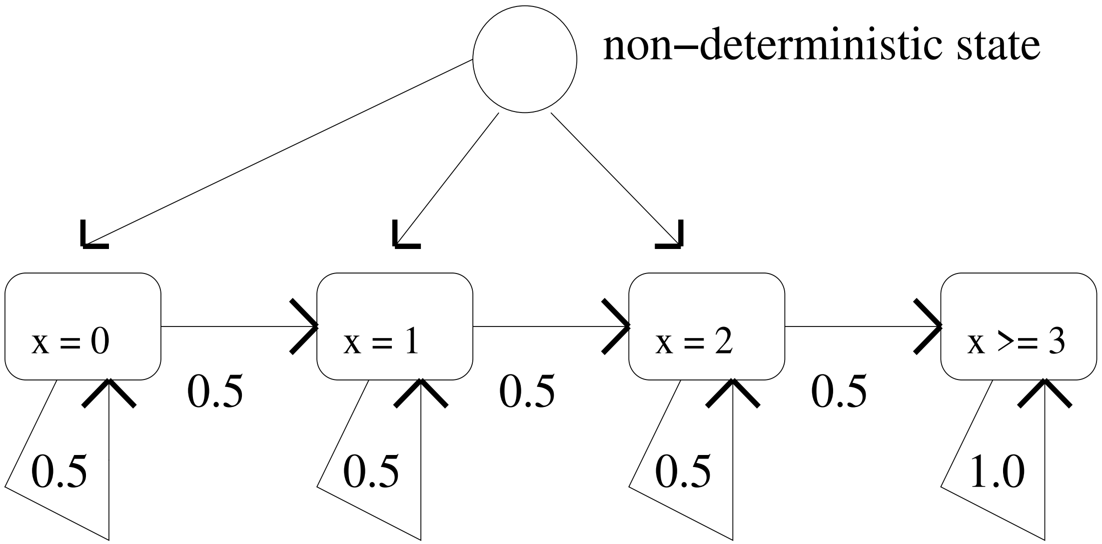
```
int aRandomUpdate(x : int) { // D. Monniaux
// pre-cond.: 0 <= x <= 2
 for (i = 0; i < 5; i++) {
  // coin_flip() returns 0 or 1 with prob. 0.5
  x = x + coin_flip(); }
 return x;
} // post-cond.: x < 3
```

An abstraction in MDP

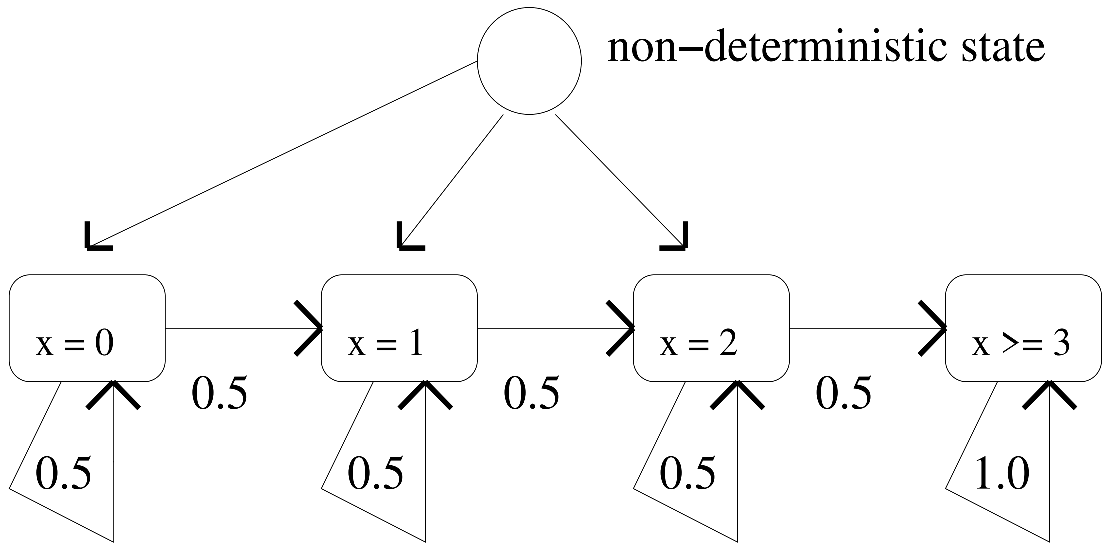
```
int aRandomUpdate(x : int) { // D. Monniaux
// pre-cond.: 0 <= x <= 2
 for (i = 0; i < 5; i++) {
  // coin_flip() returns 0 or 1 with prob. 0.5
  x = x + coin_flip(); }
 return x;
} // post-cond.: x < 3
```



An abstraction in MDP (2)

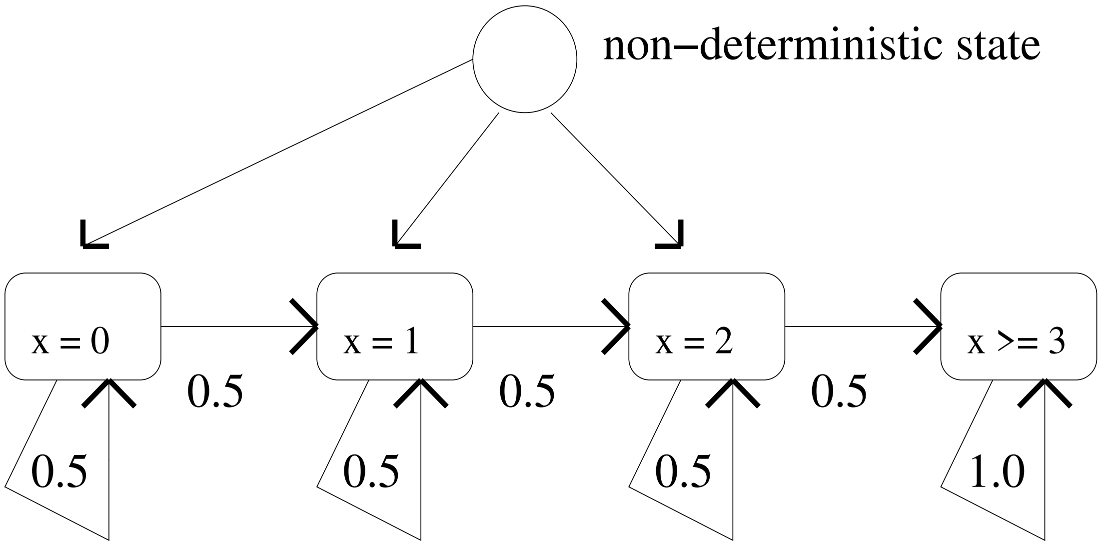


An abstraction in MDP (2)



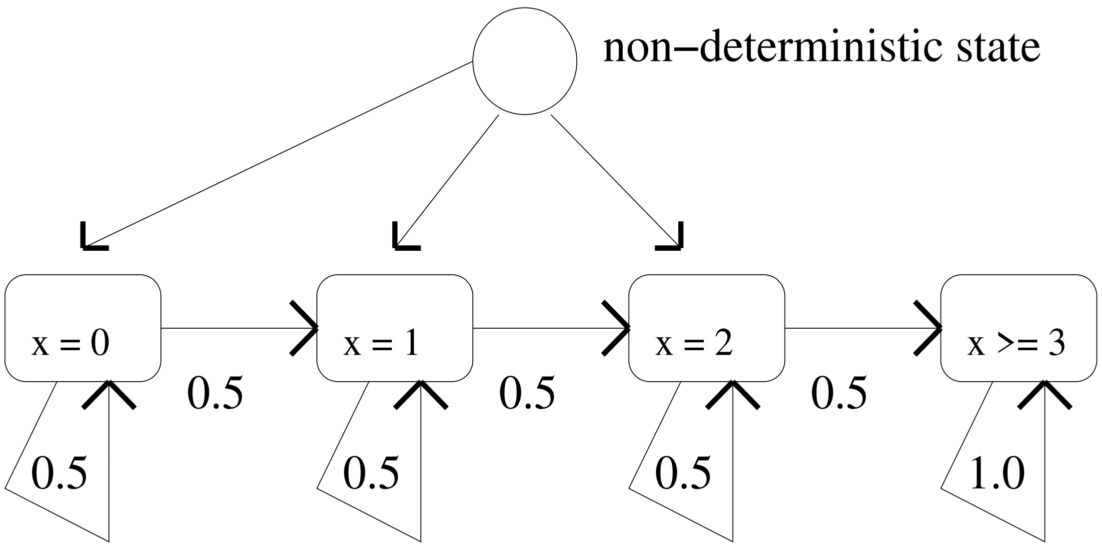
- non-determinism: under-specification of method input;

An abstraction in MDP (2)



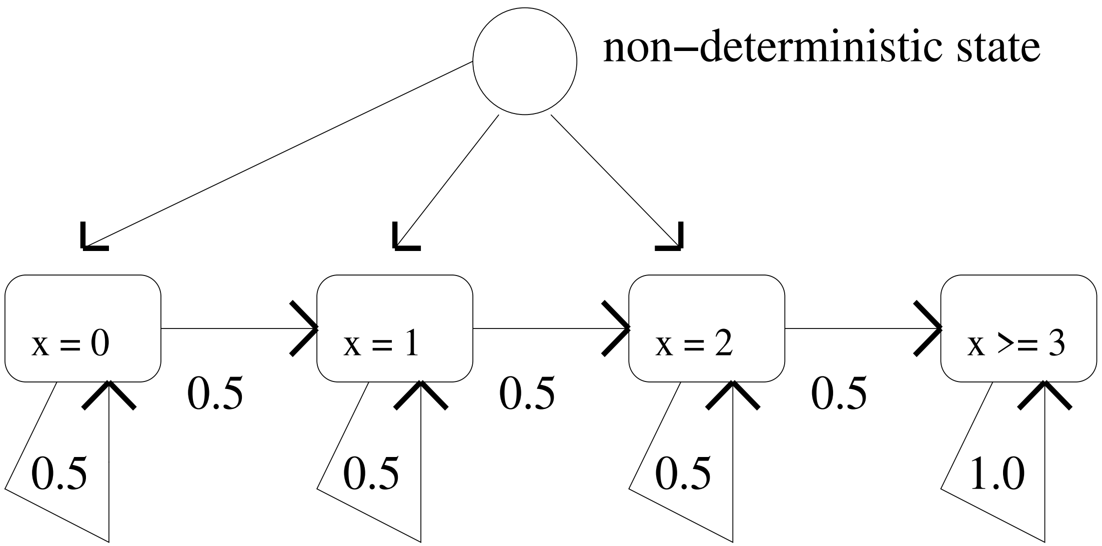
- non-determinism: under-specification of method input;
- probabilistic behavior: random coin flip;

An abstraction in MDP (2)



- non-determinism: under-specification of method input;
- probabilistic behavior: random coin flip;
- abstraction: driven by pre- and post-conditions;

An abstraction in MDP (2)



- non-determinism: under-specification of method input;
- probabilistic behavior: random coin flip;
- abstraction: driven by pre- and post-conditions;
- $(M \models (0 \leq x \leq 2) \rightarrow \mathsf{G}(x < 3)) = [0, 0]$; computed through bounded reachability: $\mathsf{true}\,\mathsf{U}^{\leq 5}\,(x >= 3)$.

Abstracting source code

Driven by

- pre- and post-conditions;

Abstracting source code

Driven by

- pre- and post-conditions;
- (object) invariants;

Abstracting source code

Driven by

- pre- and post-conditions;
- (object) invariants;
- control-flow constraints (e.g. boolean guards);

Abstracting source code

Driven by

- pre- and post-conditions;
- (object) invariants;
- control-flow constraints (e.g. boolean guards);
- explicit bounded fixed-points (for-statements);

Abstracting source code

Driven by

- pre- and post-conditions;
- (object) invariants;
- control-flow constraints (e.g. boolean guards);
- explicit bounded fixed-points (for-statements);
- bounded model-checking (while-statements): e.g. replace $\mathsf{F}\,p$ with $p \vee \mathsf{X}\,p \vee \mathsf{XX}\,p$ etc.

Abstracting source code

Driven by

- pre- and post-conditions;
- (object) invariants;
- control-flow constraints (e.g. boolean guards);
- explicit bounded fixed-points (for-statements);
- bounded model-checking (while-statements): e.g. replace $\mathsf{F}\,p$ with $p \vee \mathsf{X}\,p \vee \mathsf{XX}\,p$ etc.

Well known for non-probabilistic programs, but transfer to mixed models requires more (e.g. Markov kernels, measure theory etc).

Abstracting source code

Driven by

- pre- and post-conditions;
- (object) invariants;
- control-flow constraints (e.g. boolean guards);
- explicit bounded fixed-points (for-statements);
- bounded model-checking (while-statements): e.g. replace $\mathsf{F}\, p$ with $p \vee \mathsf{X}\, p \vee \mathsf{XX}\, p$ etc.

Well known for non-probabilistic programs, but transfer to mixed models requires more (e.g. Markov kernels, measure theory etc).

Recommended reading: D. Monniaux's doctoral thesis: abstract interpretation for Markov decision processes; and Monto Carlo methods.

Why abstraction? Decidability.

```
int Collatz(x : int) {   // pre-condition: 1 <= x
  int c = x;
  while (c != 1) {
    if (c%2 == 0) { c = c/2; } // %2 is ``modulo 2''
    else { c = 3*c + 1; }
  }
  return c;
}   // post-condition: c = 1.
```

Why abstraction? Decidability.

```
int Collatz(x : int) {   // pre-condition: 1 <= x
  int c = x;
  while (c != 1) {
    if (c%2 == 0) { c = c/2; } // %2 is ‘‘modulo 2’’
    else { c = 3*c + 1; }
  }
  return c;
}   // post-condition: c = 1.
```

- semantics of code: infinite-state Kripke structure M in x and c;

Why abstraction? Decidability.

```
int Collatz(x : int) {   // pre-condition: 1 <= x
  int c = x;
  while (c != 1) {
    if (c%2 == 0) { c = c/2; } // %2 is ``modulo 2''
    else { c = 3*c + 1; }
  }
  return c;
}   // post-condition: c = 1.
```

- semantics of code: infinite-state Kripke structure M in x and c;

- answer to CTL check $(M \models (1 \leq x) \to \mathsf{AF}(c = 1))$ unknown;

Why abstraction? Decidability.

```
int Collatz(x : int) {   // pre-condition: 1 <= x
  int c = x;
  while (c != 1) {
    if (c%2 == 0) { c = c/2; } // %2 is ``modulo 2''
    else { c = 3*c + 1; }
  }
  return c;
}   // post-condition: c = 1.
```

- semantics of code: infinite-state Kripke structure M in x and c;

- answer to CTL check $(M \models (1 \leq x) \to \mathsf{AF}(c = 1))$ unknown;
- infinite Kripke structures as infinite Markov decision processes;

Why abstraction? Decidability.

```
int Collatz(x : int) {   // pre-condition: 1 <= x
  int c = x;
  while (c != 1) {
    if (c%2 == 0) { c = c/2; } // %2 is ``modulo 2''
    else { c = 3*c + 1; }
  }
  return c;
}   // post-condition: c = 1.
```

- semantics of code: infinite-state Kripke structure M in x and c;
- answer to CTL check $(M \models (1 \leq x) \to \mathsf{AF}(c = 1))$ unknown;
- infinite Kripke structures as infinite Markov decision processes;
- checks in MDP undecidable in general.

Why abstraction? Complexity.

- STOCHASTIC SCHEDULING, scheduling tasks with completion times determined by Poisson distributions: PSPACE-complete;

Why abstraction? Complexity.

- STOCHASTIC SCHEDULING, scheduling tasks with completion times determined by Poisson distributions: PSPACE-complete;
- deciding whether the synchronous composition of processes can deadlock: PSPACE-complete;

Why abstraction? Complexity.

- STOCHASTIC SCHEDULING, scheduling tasks with completion times determined by Poisson distributions: PSPACE-complete;
- deciding whether the synchronous composition of processes can deadlock: PSPACE-complete;
- for mixed models in MDP, the worst-case upper bound time complexity of $M \models \phi$ is doubly exponential in the size of ϕ and polynomial in the size of M if M is finite;

Why abstraction? Complexity.

- STOCHASTIC SCHEDULING, scheduling tasks with completion times determined by Poisson distributions: PSPACE-complete;
- deciding whether the synchronous composition of processes can deadlock: PSPACE-complete;
- for mixed models in MDP, the worst-case upper bound time complexity of $M \models \phi$ is doubly exponential in the size of ϕ and polynomial in the size of M if M is finite;
- conjunction $\phi_1 \wedge \phi_2$ complexity bottleneck in MDP: conditional probabilities, non-compositional.

Complexity-driven abstraction in MDP

- Baier et al. tame MDP complexity with sound approximation of conjunctive probabilistic meaning;

Complexity-driven abstraction in MDP

- Baier et al. tame MDP complexity with sound approximation of conjunctive probabilistic meaning;
- use $c\colon [0,1] \times [0,1] \to [0,1]$ with

$$c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}];$$

Complexity-driven abstraction in MDP

- Baier et al. tame MDP complexity with sound approximation of conjunctive probabilistic meaning;
- use $c\colon [0,1] \times [0,1] \to [0,1]$ with

$$c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}];$$

- For $[r', s'] = c([r_1, s_1], [r_2, s_2])$ soundness follows since $r' \leq r_1, r_2$ and $s_1, s_2 \leq s'$ by the modularity axiom of measures;

Complexity-driven abstraction in MDP

- Baier et al. tame MDP complexity with sound approximation of conjunctive probabilistic meaning;
- use $c\colon [0,1] \times [0,1] \to [0,1]$ with

$$c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}];$$

- For $[r', s'] = c([r_1, s_1], [r_2, s_2])$ soundness follows since $r' \leq r_1, r_2$ and $s_1, s_2 \leq s'$ by the modularity axiom of measures;
- modifies $\models$ to $\models'$;

Complexity-driven abstraction in MDP

- Baier et al. tame MDP complexity with sound approximation of conjunctive probabilistic meaning;
- use $c\colon [0,1] \times [0,1] \to [0,1]$ with

$$c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}];$$

- For $[r', s'] = c([r_1, s_1], [r_2, s_2])$ soundness follows since $r' \leq r_1, r_2$ and $s_1, s_2 \leq s'$ by the modularity axiom of measures;
- modifies $\models$ to $\models'$;
- makes time complexity single exponential in size of formula, has linear space complexity.

Complexity-driven abstraction of MDP

- For feasible average complexity of $\models$ in a MCP, $\models$ may be approximated with randomized algorithms;

Complexity-driven abstraction of MDP

- For feasible average complexity of $\models$ in a MCP, $\models$ may be approximated with randomized algorithms;
- this is unsound, but potentially useful;

Complexity-driven abstraction of MDP

- For feasible average complexity of $\models$ in a MCP, $\models$ may be approximated with randomized algorithms;
- this is unsound, but potentially useful;
- is a form of probabilistic testing;

Complexity-driven abstraction of MDP

- For feasible average complexity of $\models$ in a MCP, $\models$ may be approximated with randomized algorithms;
- this is unsound, but potentially useful;
- is a form of probabilistic testing;
- randomization can improve complexity: trace equivalence in labeled transition systems is PSPACE complete, but trace equivalence in fully probabilistic systems is time polynomial.

Abstracting environments: adversaries

- randomize interaction with environment, e.g. put probability distribution on input values: probabilistic predicate transformers [Kozen, Morgan et al];

Abstracting environments: adversaries

- randomize interaction with environment, e.g. put probability distribution on input values: probabilistic predicate transformers [Kozen, Morgan et al];
- for next move in mixed model, adversary has access to
  - current state only: Markov condition of history-free strategy, used in MDP;
  - execution trace from initial to present state (history-dependent strategy); or
  - may see all the system's random choices during its unfolding, e.g. Monte Carlo methods.

Abstracting environments: adversaries

- randomize interaction with environment, e.g. put probability distribution on input values: probabilistic predicate transformers [Kozen, Morgan et al];
- for next move in mixed model, adversary has access to
  - current state only: Markov condition of history-free strategy, used in MDP;
  - execution trace from initial to present state (history-dependent strategy); or
  - may see all the system's random choices during its unfolding, e.g. Monte Carlo methods.
- **caveat abstractor**: e.g. semantics of safety properties in MDP does not change if strategies are history-*dependent*; *not* true for liveness properties.

Caveat abstractor (2)

- soundness of abstraction context-dependent;

Caveat abstractor (2)

- soundness of abstraction context-dependent;
- e.g. sound abstraction of universal-path properties may interfere with existential-path properties (e.g. fairness assumptions);

Caveat abstractor (2)

- soundness of abstraction context-dependent;
- e.g. sound abstraction of universal-path properties may interfere with existential-path properties (e.g. fairness assumptions);
- debugging diagnostics may be unsound (e.g. spurious counter-examples);

Caveat abstractor (2)

- soundness of abstraction context-dependent;
- e.g. sound abstraction of universal-path properties may interfere with existential-path properties (e.g. fairness assumptions);
- debugging diagnostics may be unsound (e.g. spurious counter-examples);
- sound abstraction for first-order logic with transitive closure possible for Markov decision processes: three-valued Kleene semantics on top of probabilistic one [Huth'02].

Abstracting with Occam's razor

- For MCPs, Occam's razor specializes to

  "Whatever aspects of M, $\models$, and ϕ are not needed in computing $(M \models \phi)$ should be discarded."

Abstracting with Occam's razor

- For MCPs, Occam's razor specializes to

  "Whatever aspects of M, $\models$, and ϕ are not needed in computing $(M \models \phi)$ should be discarded."

- e.g. software slicing abstracts M — the model of a program p — to M' and then checks $M' \models \phi$;

Abstracting with Occam's razor

- For MCPs, Occam's razor specializes to

  "Whatever aspects of M, $\models$, and ϕ are not needed in computing $(M \models \phi)$ should be discarded."

- e.g. software slicing abstracts M — the model of a program p — to M' and then checks $M' \models \phi$;

- M' computed by
  - defining the program points C of interest, typically determined by the query ϕ;
  - computing all program points C' on which points in C depend; and
  - by using C' to compress the model M to some M';

Abstracting with Occam's razor

- For MCPs, Occam's razor specializes to

  "Whatever aspects of M, $\models$, and ϕ are not needed in computing $(M \models \phi)$ should be discarded."

- e.g. software slicing abstracts M — the model of a program p — to M' and then checks $M' \models \phi$;

- M' computed by
  - defining the program points C of interest, typically determined by the query ϕ;
  - computing all program points C' on which points in C depend; and
  - by using C' to compress the model M to some M';

- existing work for mixed models?

Specifying abstract MCPs

A MCP $\mathcal{P}' = (\mathcal{M}', \Phi', \mathtt{Ans}', \models')$ abstracts the MCP $\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ with witness (α, β, γ) iff

Specifying abstract MCPs

A MCP $\mathcal{P}' = (\mathcal{M}', \Phi', \mathtt{Ans}', \models')$ abstracts the MCP $\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ with witness (α, β, γ) iff

1. $\alpha \subseteq \mathcal{M} \times \mathcal{M}'$ is a relation where $(M, M') \in \alpha$ means that model M' abstracts model M;

Specifying abstract MCPs

A MCP $\mathcal{P}' = (\mathcal{M}', \Phi', \texttt{Ans}', \models')$ abstracts the MCP $\mathcal{P} = (\mathcal{M}, \Phi, \texttt{Ans}, \models)$ with witness (α, β, γ) iff

1. $\alpha \subseteq \mathcal{M} \times \mathcal{M}'$ is a relation where $(M, M') \in \alpha$ means that model M' abstracts model M;
2. $\beta \subseteq \Phi \times \Phi'$ is a relation where $(\phi, \phi') \in \beta$ means that the query ϕ' abstracts the query ϕ; and

Specifying abstract MCPs

A MCP $\mathcal{P}' = (\mathcal{M}', \Phi', \mathtt{Ans}', \models')$ abstracts the MCP $\mathcal{P} = (\mathcal{M}, \Phi, \mathtt{Ans}, \models)$ with witness (α, β, γ) iff

1. $\alpha \subseteq \mathcal{M} \times \mathcal{M}'$ is a relation where $(M, M') \in \alpha$ means that model M' abstracts model M;
2. $\beta \subseteq \Phi \times \Phi'$ is a relation where $(\phi, \phi') \in \beta$ means that the query ϕ' abstracts the query ϕ; and
3. $\gamma \colon \mathtt{Ans}' \to \mathtt{Ans}$ is a total function such that $\gamma(a')$ is the concrete interpretation of the abstract answer $a' \in \mathtt{Ans}'$.

Specifying abstract MCPs

A MCP $\mathcal{P}' = (\mathcal{M}', \Phi', \texttt{Ans}', \models')$ abstracts the MCP $\mathcal{P} = (\mathcal{M}, \Phi, \texttt{Ans}, \models)$ with witness (α, β, γ) iff

1. $\alpha \subseteq \mathcal{M} \times \mathcal{M}'$ is a relation where $(M, M') \in \alpha$ means that model M' abstracts model M;
2. $\beta \subseteq \Phi \times \Phi'$ is a relation where $(\phi, \phi') \in \beta$ means that the query ϕ' abstracts the query ϕ; and
3. $\gamma \colon \texttt{Ans}' \to \texttt{Ans}$ is a total function such that $\gamma(a')$ is the concrete interpretation of the abstract answer $a' \in \texttt{Ans}'$.

Soundness:

$$(M, M') \in \alpha \;\&\; (\phi, \phi') \in \beta \;\Rightarrow\; \gamma(M' \models' \phi') \leq (M \models \phi).$$

Examples of abstract MCPs of MDP

- $\texttt{Ans} = \texttt{Ans}'$, $\models = \models'$, β and γ identities, and $(M, M') \in \alpha$ iff M' is abstract interpretation of M as in Monniaux's thesis: sound;

Examples of abstract MCPs of MDP

- $\texttt{Ans} = \texttt{Ans}'$, $\models = \models'$, β and γ identities, and $(M, M') \in \alpha$ iff M' is abstract interpretation of M as in Monniaux's thesis: sound;
- as in previous item, but α specifies probabilistic simulation (sound for LTL) or bisimulation (sound for branching time);

Examples of abstract MCPs of MDP

- $\texttt{Ans} = \texttt{Ans}'$, $\models = \models'$, β and γ identities, and $(M, M') \in \alpha$ iff M' is abstract interpretation of M as in Monniaux's thesis: sound;
- as in previous item, but α specifies probabilistic simulation (sound for LTL) or bisimulation (sound for branching time);
- as previous item, but $\models'$ interprets $\wedge$ compositionally: $c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}]$: sound;

Examples of abstract MCPs of MDP

- $\texttt{Ans} = \texttt{Ans}'$, $\models \, = \, \models'$, β and γ identities, and $(M, M') \in \alpha$ iff M' is abstract interpretation of M as in Monniaux's thesis: sound;
- as in previous item, but α specifies probabilistic simulation (sound for LTL) or bisimulation (sound for branching time);
- as previous item, but $\models'$ interprets $\wedge$ compositionally: $c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}]$: sound;
- as in first or previous item, but F, U, and G turn into bounded instances: potentially unsound;

Examples of abstract MCPs of MDP

- $\mathtt{Ans} = \mathtt{Ans}'$, $\models \, = \, \models'$, β and γ identities, and $(M, M') \in \alpha$ iff M' is abstract interpretation of M as in Monniaux's thesis: sound;
- as in previous item, but α specifies probabilistic simulation (sound for LTL) or bisimulation (sound for branching time);
- as previous item, but $\models'$ interprets $\wedge$ compositionally: $c([r_1, s_1], [r_2, s_2]) = [\max\{0, r_1 + r_2 - 1\}, \min\{s_1, s_2\}]$: sound;
- as in first or previous item, but F, U, and G turn into bounded instances: potentially unsound;
- possibilistic abstractions: M' qualitative model, $\mathtt{Ans}'$ discrete domain; often sound for "probability $= 1$" type questions.

MDP as sound abstraction

$$\phi ::= \text{true} \mid o \mid \neg\phi \mid \phi \wedge \phi \mid [\mathsf{X}\,\phi]_{\geq p} \mid [\phi\,\mathsf{U}\,\phi]_{\geq p} \qquad (\text{LTL}_{\geq})$$

MDP as sound abstraction

$$\phi ::= \text{true} \mid o \mid \neg\phi \mid \phi \wedge \phi \mid [\mathsf{X}\,\phi]_{\geq p} \mid [\phi\,\mathsf{U}\,\phi]_{\geq p} \qquad (\text{LTL}_{\geq})$$

$\models_{\geq}$ models worst-case probability of the underlying trace set be $\geq p$; MDP abstracts the MCP $(\mathcal{M}', \models, \Phi, \texttt{Ans})$, where

MDP as sound abstraction

$$\phi ::= \mathrm{true} \mid o \mid \neg\phi \mid \phi \wedge \phi \mid [\mathsf{X}\,\phi]_{\geq p} \mid [\phi\,\mathsf{U}\,\phi]_{\geq p} \qquad (\mathrm{LTL}_{\geq})$$

$\models_{\geq}$ models worst-case probability of the underlying trace set be $\geq p$; MDP abstracts the MCP $(\mathcal{M}', \models, \Phi, \mathtt{Ans})$, where

- $\models$ equals $\models_{\geq} \times \models_{\geq}$,
- Φ equals $\mathrm{LTL}_{\geq} \times \mathrm{LTL}_{\geq}$,
- $\mathtt{Ans}$ equals $\{\mathrm{false} < \mathrm{true}\} \times \{\mathrm{true} < \mathrm{false}\}$,
- α is the identity,
- β contains only pairs $((\phi^\flat, \phi^\sharp), \phi)$ — $[r, s]$-semantics provides thresholds,
- and $\gamma\colon \mathtt{Ans}' \to \mathtt{Ans}$ maps intervals $[r, s]$ to $(\alpha^\forall(r), \alpha^\exists(s)) \in \mathtt{Ans}$; $\alpha^\forall, \alpha^\exists\colon [0,1] \to \{\mathrm{false} < \mathrm{true}\}$, $(\alpha^\forall)^{-1}(\mathrm{false}) = \{0\}$ and $(\alpha^\exists)^{-1}(\mathrm{true}) = \{1\}$.

Abstraction through query equivalence

$M \sim M'$ iff $\lambda\phi\colon \Phi.(M \models \phi) = \lambda\phi\colon \Phi.(M' \models \phi)$.

Abstraction through query equivalence

$$M \sim M' \qquad \text{iff} \qquad \lambda\phi\colon \Phi.(M \models \phi) = \lambda\phi\colon \Phi.(M' \models \phi).$$

$\sim$ partitions $\mathcal{M}$; induces abstract MCP (as α); applications:

Abstraction through query equivalence

$$M \sim M' \qquad \text{iff} \qquad \lambda\phi\colon \Phi.(M \models \phi) = \lambda\phi\colon \Phi.(M' \models \phi).$$

$\sim$ partitions $\mathcal{M}$; induces abstract MCP (as α); applications:

- replace M in $M \models \phi$ with some M' such that $M \sim M'$ and the computation of $(M' \models \phi)$ has "minimal" cost — e.g. size of state space;

Abstraction through query equivalence

$$M \sim M' \qquad \text{iff} \qquad \lambda\phi\colon \Phi.(M \models \phi) = \lambda\phi\colon \Phi.(M' \models \phi).$$

$\sim$ partitions $\mathcal{M}$; induces abstract MCP (as α); applications:

- replace M in $M \models \phi$ with some M' such that $M \sim M'$ and the computation of $(M' \models \phi)$ has "minimal" cost — e.g. size of state space;
- alas, some continuous-space labeled Markov chains aren't equivalent to any finite-state model for a simple probabilistic modal logic Φ;

Abstraction through query equivalence

$$M \sim M' \qquad \text{iff} \qquad \lambda\phi\colon \Phi.(M \models \phi) = \lambda\phi\colon \Phi.(M' \models \phi).$$

$\sim$ partitions $\mathcal{M}$; induces abstract MCP (as α); applications:

- replace M in $M \models \phi$ with some M' such that $M \sim M'$ and the computation of $(M' \models \phi)$ has "minimal" cost — e.g. size of state space;
- alas, some continuous-space labeled Markov chains aren't equivalent to any finite-state model for a simple probabilistic modal logic Φ;
- probabilistic bisimulations, establish instances of query equivalences for suitable Φ;

Abstraction through query equivalence

$$M \sim M' \qquad \text{iff} \qquad \lambda\phi\colon \Phi.(M \models \phi) = \lambda\phi\colon \Phi.(M' \models \phi).$$

$\sim$ partitions $\mathcal{M}$; induces abstract MCP (as α); applications:

- replace M in $M \models \phi$ with some M' such that $M \sim M'$ and the computation of $(M' \models \phi)$ has "minimal" cost — e.g. size of state space;
- alas, some continuous-space labeled Markov chains aren't equivalent to any finite-state model for a simple probabilistic modal logic Φ;
- probabilistic bisimulations, establish instances of query equivalences for suitable Φ;
- weak probabilistic bisimulations may/may not be consistent with query semantics.

Precise abstract MCPs

$$\mathrm{A}_{\phi}^{M} = \{\gamma(M' \models \phi') \mid (M, M') \in \alpha,\ (\phi, \phi') \in \beta\}$$

collects all information about $(M \models \phi)$ we could ever infer from abstract and sound checks.

Precise abstract MCPs

$$\mathrm{A}_\phi^M = \{\gamma(M' \models \phi') \mid (M, M') \in \alpha,\ (\phi, \phi') \in \beta\}$$

collects all information about $(M \models \phi)$ we could ever infer from abstract and sound checks.

- For $\phi \in \Phi$, $\mathcal{P}'$ is ϕ-*precise* for $\mathcal{P}$ iff for all $M \in \mathcal{M}$, A_ϕ^M has $(M \models \phi)$ as a least upper bound;

Precise abstract MCPs

$$\mathrm{A}_\phi^M = \{\gamma(M' \models \phi') \mid (M, M') \in \alpha,\ (\phi, \phi') \in \beta\}$$

collects all information about $(M \models \phi)$ we could ever infer from abstract and sound checks.

- For $\phi \in \Phi$, $\mathcal{P}'$ is ϕ-*precise* for $\mathcal{P}$ iff for all $M \in \mathcal{M}$, A_ϕ^M has $(M \models \phi)$ as a least upper bound;
- $\mathcal{P}'$ is *precise* for $\mathcal{P}$ iff it is ϕ-precise for $\mathcal{P}$ for all $\phi \in \Phi$.

Precise abstract MCPs

$$\mathrm{A}_\phi^M = \{\gamma(M' \models \phi') \mid (M, M') \in \alpha,\ (\phi, \phi') \in \beta\}$$

collects all information about $(M \models \phi)$ we could ever infer from abstract and sound checks.

- For $\phi \in \Phi$, $\mathcal{P}'$ is ϕ-*precise* for $\mathcal{P}$ iff for all $M \in \mathcal{M}$, A_ϕ^M has $(M \models \phi)$ as a least upper bound;
- $\mathcal{P}'$ is *precise* for $\mathcal{P}$ iff it is ϕ-precise for $\mathcal{P}$ for all $\phi \in \Phi$.
- Example: **finite-model property** for Kripke structures; $(M, M') \in \alpha$ implies M' finite; $\mathtt{Ans} = \{\text{false} < \text{true}\}$; if ϕ holds in M, does it hold in a finite abstraction M'?

Precise abstractions in MDP

- A_{ϕ}^{M} may not be directed,

Precise abstractions in MDP

- A_{ϕ}^{M} may not be directed,
- its information content may lead to directed approximations;

Precise abstractions in MDP

- A_{ϕ}^{M} may not be directed,
- its information content may lead to directed approximations;
- e.g. in MDP, if $[r', s']$ and $[r'', s'']$ are in A_{ϕ}^{M} for $\mathtt{Ans} = \mathcal{I}$, then $[\max(r', r''), \min(s', s'')]$ is a rational re-construction of an upper bound that still approximates $(M \models \phi)$ — although $[\max(r', r''), \min(s', s'')]$ may not be in A_{ϕ}^{M};

Precise abstractions in MDP

- A_ϕ^M may not be directed,
- its information content may lead to directed approximations;
- e.g. in MDP, if $[r', s']$ and $[r'', s'']$ are in A_ϕ^M for $\mathtt{Ans} = \mathcal{I}$, then $[\max(r', r''), \min(s', s'')]$ is a rational re-construction of an upper bound that still approximates $(M \models \phi)$ — although $[\max(r', r''), \min(s', s'')]$ may not be in A_ϕ^M;
- finite acyclic labeled Markov processes are precise for continuous-state labeled Markov processes [Deharnais et al.'00];

Precise abstractions in MDP

- A_{ϕ}^{M} may not be directed,
- its information content may lead to directed approximations;
- e.g. in MDP, if $[r', s']$ and $[r'', s'']$ are in A_{ϕ}^{M} for $\texttt{Ans} = \mathcal{I}$, then $[\max(r', r''), \min(s', s'')]$ is a rational re-construction of an upper bound that still approximates $(M \models \phi)$ — although $[\max(r', r''), \min(s', s'')]$ may not be in A_{ϕ}^{M};
- finite acyclic labeled Markov processes are precise for continuous-state labeled Markov processes [Deharnais et al.'00];

- metric $d([r, s], [r', s']) \stackrel{\text{def}}{=} |r - r'| + |s - s'|$ measures accuracy of $\gamma(M' \models' \phi')$ with respect to $(M \models \phi)$;

Precise abstractions in MDP

- A^M_ϕ may not be directed,
- its information content may lead to directed approximations;
- e.g. in MDP, if $[r', s']$ and $[r'', s'']$ are in A^M_ϕ for $\texttt{Ans} = \mathcal{I}$, then $[\max(r', r''), \min(s', s'')]$ is a rational re-construction of an upper bound that still approximates $(M \models \phi)$ — although $[\max(r', r''), \min(s', s'')]$ may not be in A^M_ϕ;
- finite acyclic labeled Markov processes are precise for continuous-state labeled Markov processes [Deharnais et al.'00];

- metric $d([r, s], [r', s']) \stackrel{\text{def}}{=} |r - r'| + |s - s'|$ measures accuracy of $\gamma(M' \models' \phi')$ with respect to $(M \models \phi)$;
- begs the question of what $(M \models \phi)$ is, blurs under-specification of adversary and approximation.

Approximation Problem via Measurements

- answer domains `Ans` have a measurement [Keye'00]
  $\mu\colon \texttt{Ans} \to [0, \infty)$: $\mu(a)$ degree of partiality of a;

Approximation Problem via Measurements

- answer domains `Ans` have a measurement [Keye'00]
  $\mu\colon \texttt{Ans} \to [0, \infty)$: $\mu(a)$ degree of partiality of a;
- allow measuring precision of abstraction without knowing un-abstracted value;

Approximation Problem via Measurements

- answer domains $\texttt{Ans}$ have a measurement [Keye'00] $\mu\colon \texttt{Ans} \to [0,\infty)$: $\mu(a)$ degree of partiality of a;
- allow measuring precision of abstraction without knowing un-abstracted value;
- reformulated approximation problem:

  "Given $\epsilon > 0$, $M \in \mathcal{M}$, and $\phi \in \Phi$, are there $M' \in \mathcal{M}'$ and $\phi' \in \Phi'$ such that $(M, M') \in \alpha$, $(\phi, \phi') \in \beta$, and

$$\mu(\gamma(M' \models' \phi')) < \epsilon?''$$

Limitations of approach

- **diagnostics:** need ability to attach information to extensional replies (e.g. $[r, s]$ for MDP);

Limitations of approach

- **diagnostics:** need ability to attach information to extensional replies (e.g. $[r, s]$ for MDP);
- **cost effectiveness:**
  - specifying/computing executable abstractions cost more than executing un-abstracted models?
  - heuristics or other support to guide choice of abstraction?

Limitations of approach

- **diagnostics:** need ability to attach information to extensional replies (e.g. $[r, s]$ for MDP);
- **cost effectiveness:**
  - specifying/computing executable abstractions cost more than executing un-abstracted models?
  - heuristics or other support to guide choice of abstraction?
- **abandoning soundness:** limited bounded model checking, probabilistic testing, Monte Carlo methods, lack of finite-model property, approximative bisimilarity and information flow [Di Pierro et el.'02], etc.

Research challenges

- techniques for query-driven (ϕ) abstraction of models such that $(M \models \phi) = (M' \models \phi)$;

Research challenges

- techniques for query-driven (ϕ) abstraction of models such that $(M \models \phi) = (M' \models \phi)$;
- prove or disprove that abstract MCPs are precise for their concrete counterparts;

Research challenges

- techniques for query-driven (ϕ) abstraction of models such that $(M \models \phi) = (M' \models \phi)$;
- prove or disprove that abstract MCPs are precise for their concrete counterparts;
- heuristics or formal techniques for solving the abstract MCP approximation problem;

Research challenges

- techniques for query-driven (ϕ) abstraction of models such that $(M \models \phi) = (M' \models \phi)$;
- prove or disprove that abstract MCPs are precise for their concrete counterparts;
- heuristics or formal techniques for solving the abstract MCP approximation problem;
- develop diagnostics for the evidence of answers; develop probabilistic games as such diagnostics;

Research challenges

- techniques for query-driven (ϕ) abstraction of models such that $(M \models \phi) = (M' \models \phi)$;
- prove or disprove that abstract MCPs are precise for their concrete counterparts;
- heuristics or formal techniques for solving the abstract MCP approximation problem;
- develop diagnostics for the evidence of answers; develop probabilistic games as such diagnostics;
- push established methods and results from qualitative model checking into the model checking of mixed models.

Conclusions

(This page is intentionally left blank.)