Probabilistic Model Checking

Marta Kwiatkowska
Dave Parker

Oxford University Computing Laboratory

ESSLLI’10 Summer School, Copenhagen, August 2010
Course overview

• 5 lectures: Mon–Fri, 11am–12.30pm
  – Introduction
  – 1 – Discrete time Markov chains
  – 2 – Markov decision processes
  – 3 – Continuous–time Markov chains
  – 4 – Probabilistic model checking in practice
  – 5 – Probabilistic timed automata

• Course materials available here:
  – lecture slides, reference list
Part 4

Probabilistic model checking in practice
Overview (Part 4)

- Tool support for probabilistic model checking
  - motivation, existing tools

- The PRISM model checker
  - functionality, features
  - modelling language & property specification
  - PRISM demonstration

- Probabilistic counterexamples
  - (smallest) counterexamples for PCTL + DTMCs

- Probabilistic bisimulation
  - bisimulation equivalences for DTMCs, CTMCs + minimisation
Motivation

- **Complexity of PCTL model checking**
  - generally polynomial in model size (number of states)

- **State space explosion problem**
  - models for realistic case studies are typically huge

- **Clearly tool support is required**

- **Benefits:**
  - fully automated process
  - high-level languages/formalisms for building models
  - visualisation of quantitative results
Tools – Probabilistic model checkers

- **PRISM (Probabilistic Symbolic Model Checker)**
  - DTMCs, MDPs, CTMCs + rewards, [Birmingham/Oxford]

- **MRMC (Markov Reward Model Checker)**
  - DTMCs, CTMCs + reward extensions, [Twente/Aachen]

- **LiQuor**: LTL model checking for MDPs, Probmela language (probabilistic version of SPIN’s Promela), [Dresden]

- **Simulation–based probabilistic model checking:**
  - APMC, Ymer (both based on PRISM language), VESTA

- **Many other related tools/prototypes**
  - RAPTURE, CADP, Möbius, APNN–Toolbox, SMART, GreatSPN, GRIP, CASPA, Premo, PASS, …
The PRISM tool

• **PRISM: Probabilistic symbolic model checker**
  – developed at Birmingham/Oxford University, since 1999
  – free, open source (GPL)
  – versions for Linux, Unix, Mac OS X, Windows, 64-bit OSs

• **Modelling of:**
  – DTMCs, CTMCs, MDPs + costs/rewards

• **Model checking of:**
  – PCTL, CSL, LTL, PCTL* + extensions + costs/rewards
**PRISM functionality**

- High-level modelling language
- Wide range of model analysis methods
  - efficient symbolic implementation techniques
  - also: approximate verification using simulation + sampling
- **Graphical user interface**
  - model/property editor
  - discrete-event simulator – model traces for debugging, etc.
  - easy automation of verification experiments
  - graphical visualisation of results
- **Command-line version**
  - same underlying verification engines
  - useful for scripting, batch jobs
• Many high-level modelling languages, formalisms available

• For example:
  – probabilistic/stochastic process algebras
  – stochastic Petri nets
  – stochastic activity networks

• Custom languages for tools, e.g.:
  – PRISM modelling language
  – Probmela (probabilistic variant of Promela, the input language for the model checker SPIN) – used in LiQuor
PRISM modelling language

- **Simple, textual, state-based language**
  - modelling of DTMCs, CTMCs and MDPs
  - based on Reactive Modules [AH99]

- **Basic components...**

- **Modules:**
  - components of system being modelled
  - composed in parallel

- **Variables**
  - finite (integer ranges or Booleans)
  - local or global
  - all variables public: anyone can read, only owner can modify
PRISM modelling language

- **Guarded commands**
  - describe behaviour of each module
  - i.e. the changes in state that can occur
  - labelled with probabilities (or, for CTMCs, rates)
  - (optional) action labels

\[
\text{[send]} \ (s=2) \rightarrow \ p_{\text{loss}} \ : \ (s'=3) \& \ (\text{lost}'=\text{lost}+1) + (1-p_{\text{loss}}) \ : \ (s'=4);
\]
PRISM modelling language

- **Parallel composition**
  - model multiple components that can execute independently
  - for DTMC models, mostly assume components operate synchronously, i.e. move in lock-step

- **Synchronisation**
  - simultaneous transitions in more than one module
  - guarded commands with matching action-labels
  - probability of combined transition is product of individual probabilities for each component

- **More complex parallel compositions can be defined**
  - using process-algebraic operators
  - other types of parallel composition, action hiding/renaming
module M1
  x : [0..3] init 0;
  [a] x=0 -> (x'=1);
  [b] x=1 -> 0.5:(x'=2) + 0.5:(x'=3);
endmodule

module M2
  y : [0..3] init 0;
  [a] y=0 -> (y'=1);
  [b] y=1 -> 0.4:(y'=2) + 0.6:(y'=3);
endmodule
Example: Leader election

- Randomised leader election protocol
  - due to Itai & Rodeh (1990)

- Set-up: N nodes, connected in a ring
  - communication is synchronous (lock-step)

- Aim: elect a leader
  - i.e. one uniquely designated node
  - by passing messages around the ring

- Protocol operates in rounds. In each round:
  - each node choose a (uniformly) random id \( \in \{0,\ldots,k-1\} \)
  - \( (k \text{ is a parameter of the protocol}) \)
  - all nodes pass their id around the ring
  - if there is maximum unique id, node with this id is the leader
  - if not, try again with a new round
PRISM code
**PRISM property specifications**

- **Based on (probabilistic extensions of) temporal logic**
  - incorporates PCTL, CSL, LTL, PCTL*
  - also includes: quantitative extensions, costs/rewards

- **Leader election properties**
  - \( P_{\geq 1} [ F \text{ elected} ] \)
    - with probability 1, a leader is eventually elected
  - \( P_{>0.8} [ F_{\leq k} \text{ elected} ] \)
    - with probability greater than 0.8, a leader is elected within \( k \) steps

- **Usually focus on quantitative properties:**
  - \( P_{=?} [ F_{\leq k} \text{ elected} ] \)
    - what is the probability that a leader is elected within \( k \) steps?
PRISM property specifications

- **Best/worst-case scenarios**
  - combining “quantitative” and “exhaustive” aspects

- **e.g. computing values for a range of states…**

- **P=? [ F≤t elected {tokens≤k}{min} ] –**
  - “minimum probability of the leader election algorithm completing within $t$ steps from any state where there are at most $k$ tokens”

- **R=? [ F end {“init”}{max} ] –**
  - “maximum expected run-time over all possible initial configurations”
PRISM property specifications

- **Experiments:**
  - ranges of model/property parameters
  - e.g. $P = \exists [ F \leq T \text{ error }]$ for $N=1..5$, $T=1..100$
    where $N$ is some model parameter and $T$ a time bound
  - identify *patterns*, *trends*, *anomalies* in *quantitative results*
More info on PRISM

• **PRISM website:** [http://www.prismmodelchecker.org/](http://www.prismmodelchecker.org/)
  - tool download: binaries, source code (GPL)
  - example repository (50+ case studies)
  - on-line PRISM manual
  - support: help forum, bug tracking, feature requests
  - related publications, talks, tutorials, links

• **Tutorial:** [http://www.prismmodelchecker.org/tutorial/](http://www.prismmodelchecker.org/tutorial/)
Overview (Part 4)

• Tool support for probabilistic model checking
  – motivation, existing tools

• The PRISM model checker
  – functionality, features
  – modelling language & property specification
  – PRISM demonstration

• Probabilistic counterexamples
  – (smallest) counterexamples for PCTL + DTMCs

• Probabilistic bisimulation
  – bisimulation equivalences for DTMCs, CTMCs + minimisation
Non probabilistic counterexamples

- **Counterexamples (for non-probabilistic model checking)**
  - generated when model checking a (universal) property fails
  - trace through model illustrating why property does not hold
  - major advantage of the model checking approach
  - bug finding vs. verification

- **Example:**
  - CTL property \( AG \neg \text{err} \)
  - (or equivalently, \( \neg EF \text{ err} \))
  - ("an error state is never reached")
  - counterexample is a finite trace to a state satisfying \( \text{err} \)
  - alternatively, this is a witness to the satisfaction of formula \( EF \text{ err} \)
Counterexamples for DTMCs?

• **PCTL example:** $P_{<0.01}[F_{\text{err}}]$
  – “the probability of reaching an error state is less than 0.01”
  – what is a counterexample for $s \not \models P_{<0.01}[F_{\text{err}}]$?
  – not necessarily illustrated by a single trace to an err state
  – in fact, “counterexample” is a set of paths satisfying $F_{\text{err}}$ whose combined measure is greater than or equal to 0.01

• **Alternative approach seen so far:**
  – probabilistic model checker provides actual probabilities
  – e.g. queries of the form $P_{=?}[F_{\text{err}}]$
  – anomalous behaviour identified by examining trends
  – e.g. $P_{=?}[F_{\leq T}\text{ err}]$ for $T=0,\ldots,100$

• **This lecture: DTMC counterexamples in style of [HK07]**
  – also some work done on CTMC/MDP counterexamples
DTMC notation

- **DTMC**: \( D = (S, s_{init}, P, \mathcal{L}) \)
- **Path(s)** = set of all infinite paths starting in state \( s \)
- **Pr_s**: \( \Sigma_{Path(s)} \to [0,1] \) = probability measure over infinite paths
  - where \( \Sigma_{Path(s)} \) is the \( \sigma \)-algebra on Path(s)
  - defined in terms of probabilities for finite paths
- **P_s(\omega)** = probability for finite path \( \omega = ss_1...s_n \)
  - \( P_s(s) = 1 \)
  - \( P_s(ss_1...s_n) = P(s, s_1) \cdot P(s_1, s_2) \cdot ... \cdot P(s_{n-1}, s_n) \)
  - extend notation to sets: \( P_s(C) \) for set of finite paths \( C \)
  - \( P_s \) extends uniquely to \( Pr_s \)
- **Path(s, \psi)** = \( \{ \omega \in \text{Path(s)} \mid \omega \models \psi \} \)
  - \( \text{Prob}(s, \psi) = Pr_s(\text{Path(s, \psi)}) \)
- **Path_{fin}(s, \psi)** = set of finite paths from \( s \) satisfying \( \psi \)
Counterexamples for DTMCs

• Consider PCTL properties of the form:
  – $\Pr_{\leq p} [ \Phi_1 \mathbf{U}^{\leq k} \Phi_2 ]$, where $k \in \mathbb{N} \cup \{\infty\}$
  – i.e. bounded or unbounded until formulae with closed upper probability bounds

• Refutation:
  – $s \not\models \Pr_{\leq p} [ \Phi_1 \mathbf{U}^{\leq k} \Phi_2 ]$
  – $\iff \Pr(s, [ \Phi_1 \mathbf{U}^{\leq k} \Phi_2 ]) > p$
  – $\iff \Pr_s(\text{Path}(s, \Phi_1 \mathbf{U}^{\leq k} \Phi_2 )) > p$
  – i.e. total probability mass of $\Phi_1 \mathbf{U}^{\leq k} \Phi_2$ paths exceeds $p$

• Since the property is an until formula
  – this is evidenced by a set of finite paths
Counterexamples for DTMCs

- **A counterexample** for $P_{\leq p} [ \Phi_1 U^{\leq k} \Phi_2 ]$ in state $s$ is:
  - a set $C$ of finite paths such that $C \subseteq \text{Path}_{\text{fin}}(s, \psi)$ and $P_s(C) > p$

- **Example**
  - Consider the PCTL formula:
  - $P_{\leq 0.3} [ F a ]$
  - This is not satisfied in $s_0$
  - $\text{Prob}(s_0, F a) = 1/4 + 1/8 + 1/16 + ... = 1/2$
  - A counterexample: $C = \{ s_0s_2, s_0s_0s_2 \}$
  - $P_{s_0}(C) = 1/4 + (1/2)(1/4) = 3/8 = 0.375$
Finiteness of counterexamples

• There is always a finite counterexample for:
  – $s \not\models P_{\leq p} [ \Phi_1 U \leq k \Phi_2 ]$

• On the other hand, consider this DTMC:
  – and the PCTL formula:
    – $P_{<1/2} [ F a ]$
    – $\text{Prob}(s_0, F a) = 1/4 + 1/8 + 1/16 + \ldots = 1/2$
    – $s_0 \not\models P_{<1/2} [ F a ]$

    – counterexample would require infinite set of paths
    – $\{ (s_0)^i s_2 \}_{i \in \mathbb{N}}$
Counterexamples for DTMCs

- **Aim**: counterexamples should be succinct, comprehensible

- **Set of all counterexamples**:
  - $\text{CX}_p(s, \psi) = \text{set of all counterexamples for } P_{\leq p}[\psi] \text{ in state } s$

- **Minimal counterexample**
  - counterexample $C$ with $|C| \leq |C'|$ for all $C' \in \text{CX}_p(s, \psi)$

- **“Smallest” counterexample**
  - minimal counterexample $C$ with $P(C) \geq P(C')$ for all minimal $C' \in \text{CX}_p(s, \psi)$

- **Strongest (most probable) evidence**
  - finite path $\omega$ in $\text{Path}_{\text{fin}}(s, \psi)$ such that $P(\omega) \geq P(\omega')$ for all $\omega' \in \text{Path}_{\text{fin}}(s, \psi)$
  - i.e. contributes most to violation of PCTL formula
Example

- **PCTL formula:** $P_{\leq 1/2} [ F b ]$
  - $s_0 \not\equiv P_{\leq 1/2} [ F b ]$
  - since $\text{Prob}(s_0, F b) = 0.9$

- **Counterexamples:**
  - $C_1 = \{ s_0s_1s_2, s_0s_1s_4s_2, s_0s_1s_4s_5, s_0s_4s_2 \}$
    - $P_{s_0}(C_1) = 0.2 + 0.2 + 0.12 + 0.15 = 0.67$ (not minimal)
  - $C_2 = \{ s_0s_1s_2, s_0s_1s_4s_2, s_0s_1s_4s_5 \}$
    - $P_{s_0}(C_2) = 0.2 + 0.2 + 0.12 = 0.52$ (not “smallest”)
  - $C_3 = \{ s_0s_1s_2, s_0s_1s_4s_2, s_0s_4s_2 \}$
    - $P_{s_0}(C_3) = 0.2 + 0.2 + 0.15 = 0.55$
Weighted digraphs

• A weighted directed graph is a tuple $G = (V, E, w)$ where:
  – $V$ is a set of vertices
  – $E \subseteq V \times V$ is a set of edges
  – $w : E \to \mathbb{R}_{\geq 0}$ is a weight function

• Finite path $\omega$ in $G$
  – is a sequence of vertices $v_0v_1v_2...v_n$ such that $(v_i,v_{i+1})\in E \ \forall i \geq 0$
  – the distance of $\omega = v_0v_1v_2...v_n$ is: $\sum_{i=0}^{n-1} w(v_i,v_{i+1})$

• Shortest path problem
  – given a weighted digraph, find a path between two vertices $v_1$ and $v_2$ with the smallest distance
  – i.e. a path $\omega$ s.t. $d(\omega) \leq d(\omega')$ for all other such paths $\omega'$
Finding strongest evidences

- **Reduction to graph problem...**
- **Step 1: Adapt the DTMC**
  - make states satisfying $\neg \Phi_1 \land \neg \Phi_2$ absorbing
    - (i.e. replace all outgoing transitions with a single self-loop)
  - add an extra state $t$ and replace all transitions from any $\Phi_2$ state with a single transition to $t$ (with probability 1)
- **Step 2: Convert new DTMC into a weighted digraph**
  - for the (adapted) DTMC $D = (S, s_{\text{init}}, P, L)$:
    - corresponding graph is $G_D = (V, E, w)$ where:
      - $V = S$ and $E = \{(s, s') \in S \times S \mid P(s, s') > 0\}$
      - $w(s, s') = \log(1/P(s, s'))$
  - **Key idea:** for any two paths $\omega$ and $\omega'$ in $D$ (and in $G_D$)
    - $P_S(\omega') \geq P_S(\omega)$ if and only if $d(\omega') \leq d(\omega)$
Example...

- **PCTL formula**: \( P_{\leq 1/2} [ F b ] \)
Finding strongest evidences

- To find strongest evidence in DTMC D
  - analyse corresponding digraph
- For unbounded until formula $P_{\leq p} \left[ \Phi_1 U \Phi_2 \right]$
  - solve shortest path problem in digraph (target t)
  - polynomial time algorithms exist
    - e.g. Dijsktra’s algorithm can be implemented in $O(|E| + |V| \cdot \log|V|)$
- For bounded until formula $P_{\leq p} \left[ \Phi_1 U^{\leq k} \Phi_2 \right]$
  - solve special case of the constrained shortest path problem
  - also solvable in polynomial time
- Generation of smallest counterexamples
  - based on computation of k shortest paths
  - k can be computed on the fly
Other cases

- **Lower bounds on probabilities**
  - i.e. $s \not\equiv P_{\geq p} [ \Phi_1 U^{\leq k} \Phi_2 ]$
  - negate until formula to reverse probability bound
  - solvable with BSCC computation + probabilistic reachability
  - for details, see [HK07]

- **Continuous-time Markov chains**
  - these techniques can be extended to CTMCs and CSL [HK07b]
  - naïve approach: apply DTMC techniques to uniformised DTMC
  - modifications required to get smaller counterexamples
  - another possibility: directed search based techniques [AHL05]
Overview (Part 4)

• Tool support for probabilistic model checking
  – motivation, existing tools

• The PRISM model checker
  – functionality, features
  – modelling language & property specification
  – PRISM demonstration

• Probabilistic counterexamples
  – (smallest) counterexamples for PCTL + DTMCs

• Probabilistic bisimulation
  – bisimulation equivalences for DTMCs, CTMCs + minimisation
Bisimulation

- Identifies models with the same branching structure
  - i.e. the same stepwise behaviour
  - each model can simulate the actions of the other
  - guarantees that models satisfy many of the same properties

- Uses of bisimulation:
  - show equivalence between a model and its specification
  - state space reduction: bisimulation minimisation

- Formally, bisimulation is an equivalence relation over states
  - bisimilar states must have identical labelling
  and identical stepwise behaviour
Bisimulation on DTMCs

• Consider a DTMC $D = (S, s_{\text{init}}, P, L)$

• Some notation:
  - $P(s, T) = \sum_{s' \in T} P(s, s')$ for $T \subseteq S$

• An equivalence relation $R$ on $S$ is a probabilistic bisimulation on $D$ if and only if for all $s_1 \ R \ s_2$:
  - $L(s_1) = L(s_2)$
  - $P(s_1, T) = P(s_2, T)$ for all $T \in S/R$ (i.e. for all equivalence classes of $R$)

• States $s_1$ and $s_2$ are bisimulation-equivalent (or bisimilar)
  - if there exists a probabilistic bisimulation $R$ on $D$ with $s_1 \ R \ s_2$
  - denoted $s_1 \sim \ s_2$
Simple example

- **Bisimulation relation ~**

- **Quotient of S under ~**
  - denoted $S/\sim$
  - $\{\{s_1\}, \{u_1, u_2\}, \{v_1, v_2\}\}$

- **Bisimilar states:**
  - $u_1 \sim u_2$
  - $v_1 \sim v_2$
Bisimulation on DTMCs

• **Bisimulation between DTMCs** $D_1$ and $D_2$
  – $D_1 \sim D_2$ if they have bisimilar initial states

• **Formally:**
  – state labellings for $D_1$ and $D_2$ over same set of atomic prop.s
  – bisimulation relation is over disjoint union of $D_1$ and $D_2$
Simple example

- **Bisimilar states:**
  - \( u_1 \sim u_2 \sim u \)
  - \( v_1 \sim v_2 \sim v \)
  - \( s_1 \sim s \)

- **Bisimilar DTMCs:** \( D_1 \sim D_2 \)
Quotient DTMC

- For a DTMC \( D = (S, s_{\text{init}}, P, L) \) and probabilistic bisimulation \( \sim \)

- Quotient DTMC is
  \[ D/\sim = (S', s'_{\text{init}}, P', L') \]

- where:
  \[ S' = S/\sim = \{ [s]_\sim \mid s \in S \} \]
  \[ s'_{\text{init}} = [s_{\text{init}}]_\sim \]
  \[ P'([s]_\sim, [s']_\sim) = P(s, [s']_\sim) \]
  \[ L'([s]_\sim) = L(s) \]

well defined since bisimulation ensures \( P(s, [s']_\sim) \) same for all \( s \) in \( [s]_\sim \)
Bisimulation and PCTL

- Probabilistic bisimulation preserves all PCTL formulae

- For all states $s$ and $s'$:

\[
s \sim s' \iff \text{for all PCTL formulae } \Phi, s \models \Phi \text{ if and only if } s' \models \Phi
\]

- Note also:
  - every pair of non-bisimilar states can be distinguished with some PCTL formula
  - $\sim$ is the coarsest relation with this property
  - in fact, bisimulation also preserves all PCTL* formulae
CTMC bisimulation

• Check equivalence of rates, not probabilities…

• An equivalence relation $R$ on $S$ is a probabilistic bisimulation on CTMC $C=(S,s_{\text{init}},R,L)$ if and only if for all $s_1 R s_2$:
  − $L(s_1) = L(s_2)$
  − $R(s_1, T) = R(s_2, T)$ for all classes $T$ in $S/R$

• Alternatively, check:
  − $L(s_1) = L(s_2)$, $P^{\text{emb}}(C)(s_1, T) = P^{\text{emb}}(C)(s_2, T)$, $E(s_1) = E(s_2)$

• Bisimulation on CTMCs preserves CSL
  − (see [BHHK03] and also [DP03])
Bisimulation minimisation

- More efficient to perform PCTL/CSL model checking on the quotient DTMC/CTMC
  - assuming quotient model can be constructed efficiently
  - (see [KKZJ07] for experimental results on this)

- Bisimulation minimisation
  - algorithm to construct quotient model
  - based on partition refinement
  - repeated splitting of an initially coarse partition
  - final partition is coarsest bisimulation wrt. initial partition
  - (optimisations/variants possible by changing initial partition)
  - complexity: $O(|P| \cdot \log|S| + |AP| \cdot |S|)$ [DHS’03]
    - assuming suitable data structure used (splay trees)
Bisimulation minimisation

1. Start with initial partition
   - say $\Pi = \{ \{ s \in S \mid a \in L(s) \} \mid a \in AP \}$

2. Find a splitter $T \in \Pi$ for some block $B \in \Pi$
   - a splitter $T$ is a block such that probability of going to $T$
     differs for some states in block $B$
   - i.e. $\exists s, s' \in B . P(s, T) \neq P(s', T)$

3. Split $B$ into sub-blocks
   - such that $P(s, T)$ is the same for all states in each sub-block

4. Repeat steps 2/3 until no more splitters exist
   - i.e. no change to partition $\Pi$

replace $P$ with $R$ for CTMCs
• Consider model checking $P_{\sim p}[ F^{[0,t]} a ]$ on this CTMC:

Minimisation:

\[ \Pi_0: B_1 = \{s_0, s_1, s_2, s_3, s_4, s_5\}, B_2 = \{s_6\} \]

$B_2$ is a splitter for $B_1$
(since e.g. $R(s_1, B_2) = 0 \neq 2 = R(s_2, B_2)$)

\[ \Pi_1: B_1 = \{s_0, s_1, s_4, s_5\}, B_2 = \{s_6\}, B_3 = \{s_2, s_3\} \]

$B_3$ is a splitter for $B_1$
(since e.g. $R(s_1, B_3) = 0 \neq 4 = R(s_0, B_3)$)

\[ \Pi_2: B_1 = \{s_1, s_5\}, B_2 = \{s_6\}, B_3 = \{s_2, s_3\}, B_4 = \{s_0, s_4\} \]

No more splitters...

\[ S/\sim = \{ \{s_1, s_5\}, \{s_6\}, \{s_2, s_3\}, \{s_0, s_4\} \} \]
CTMC example…

\[ \text{Prob}^C(s, F^{[0,t]} a) = \text{Prob}^{C/\sim}(\{s_0,s_4\}, F^{[0,t]} a) \]
Summary

- **PRISM: Probabilistic model checker**
  - for DTMCs, MDPs, CTMCs, …
  - high-level modelling language, property specifications
  - graphical user interface

- **Counterexamples**
  - essential ingredient of non-probabilistic model checking
  - for PCTL + DTMCs, need set of finite paths/evidences
  - computation: reduction to well-known graph problems

- **Bisimulation**
  - relates states/Markov chains with identical labelling and identical stepwise behaviour, preserves PCTL, CSL, …
  - minimisation: automated construction of quotient model

- **Tomorrow: probabilistic timed automata (PTAs)**