Advances in Probabilistic Model Checking

Marta Kwiatkowska

Department of Computer Science, University of Oxford

Marktoberdorf, August 2011
### Probabilistic models

<table>
<thead>
<tr>
<th></th>
<th>Fully probabilistic</th>
<th>Nondeterministic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discrete time</strong></td>
<td>Discrete-time Markov chains (DTMCs)</td>
<td>Markov decision processes (MDPs) (probabilistic automata)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Continuous time</strong></td>
<td>Continuous-time Markov chains (CTMCs)</td>
<td>Probabilistic timed automata (PTAs)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CTMDPs/IMCs</td>
</tr>
</tbody>
</table>
Overview

• Lecture 3
  – Introduction
  – 1 – Discrete time Markov chains
  – 2 – Markov decision processes
  – 3 – Compositional probabilistic verification
  – 4 – Probabilistic timed automata

• Course materials available here:
  – http://www.prismmodelchecker.org/courses/marktoberdorf11/
  – lecture slides, reference list, exercises
Part 3

Compositional probabilistic verification
Overview (Part 3)

- **Compositional verification**
  - assume-guarantee reasoning

- **Markov decision processes**
  - probabilistic safety properties
  - multi-objective model checking

- **Probabilistic assume guarantee**
  - semantics, model checking
  - assume-guarantee proof rules
  - quantitative approaches
  - implementation & experimental results
  - assumption generation with learning
Compositional verification

- **Goal**: scalability through modular verification
  - e.g. decide if $M_1 \parallel M_2 \models G$
  - by analysing $M_1$ and $M_2$ separately

- **Assume–guarantee (AG) reasoning**
  - use assumptions $A$ about the context of a component $M$
  - $\langle A \rangle M \langle G \rangle$ – “whenever $M$ is part of a system that satisfies $A$, then the system must also guarantee $G$”
  - example of asymmetric (non–circular) AG rule:
    
    $\langle true \rangle M_1 \langle A \rangle$
    
    $\langle A \rangle M_2 \langle G \rangle$
    
    $\langle true \rangle M_1 \parallel M_2 \langle G \rangle$

[Pasareanu/Giannakopoulou/et al.]
AG rules for probabilistic systems

• How to formulate AG rules for Markov decision processes?

\[
\langle \text{true} \rangle M_1 \langle A \rangle \\
\langle A \rangle M_2 \langle G \rangle \\
\hline \\
\langle \text{true} \rangle M_1 \ || \ M_2 \langle G \rangle
\]

• Questions:
  – What form do assumptions and guarantees take?
  – What does \( \langle A \rangle M \langle G \rangle \) mean? How to check it?
  – Any restriction on parallel composition \( M_1 || M_2 \)?
  – Can we do this in a “quantitative” way?
  – How do we generate suitable assumptions?
AG rules for probabilistic systems

• How to formulate AG rules for Markov decision processes?

• Questions:
  – What form do assumptions and guarantees take?
    • probabilistic safety properties
  – What does \( \langle A \rangle M \langle G \rangle \) mean? How to check it?
    • reduction to multi-objective probabilistic model checking
  – Any restriction on parallel composition \( M_1 \parallel M_2 \)?
    • no: arbitrary parallel composition
  – Can we do this in a “quantitative” way?
    • yes: generate lower/upper bounds on probabilities
  – How do we generate suitable assumptions?
    • learning techniques (L* algorithm)
Overview (Part 3)

- Compositional verification
  - assume-guarantee reasoning

- Markov decision processes
  - probabilistic safety properties
  - multi-objective model checking

- Probabilistic assume guarantee
  - semantics, model checking
  - assume-guarantee proof rules
  - quantitative approaches
  - implementation & experimental results
  - assumption generation with learning
Recap: Markov decision processes

- **Markov decision processes (MDPs)**
  - model probabilistic and nondeterministic behaviour

- **An MDP is a tuple** $M = (S, s_{\text{init}}, \alpha_M, \delta_M, L)$:
  - $S$ is the state space
  - $s_{\text{init}} \in S$ is the initial state
  - $\alpha_M$ is the action alphabet
  - $\delta_M \subseteq S \times (\alpha_M \cup \tau) \times \text{Dist}(S)$ is the transition probability relation
  - $L : S \rightarrow 2^{\text{AP}}$ labels states with atomic propositions

- **Notes:**
  - $\alpha_M, \delta_M$ have subscripts to avoid confusion with other automata
  - transitions can also be labelled with a “silent” $\tau$ action
  - we write $s^a\mu$ as shorthand for $(s,a,\mu) \in \delta_M$
  - MDPs, here, are identical to probabilistic automata [Segala]
Recap: Adversaries for MDPs

- **Adversaries** resolves the nondeterminism in MDPs
  - also called “schedulers”, “strategies”, “policies”, ...
  - make a (possibly randomised) choice, based on history

- **An adversary** $\sigma$ for an MDP $M$
  - induces probability measure $Pr_{M,s}^{\sigma}$ over (infinite) paths $Path_{M,s}^{\sigma}$
  - we will abbreviate $Pr_{M,sinit}^{\sigma}$ to $Pr_{M}^{\sigma}$ (and $Path_{M,sinit}^{\sigma}$ to $Path_{M}^{\sigma}$)

- For adversary $\sigma$, we can compute the probability...
  - ... of some measurable property $\phi$ of paths
  - here, we use either temporal logic (LTL) over state labels
    - e.g. $\Diamond$err – “an error eventually occurs”
    - e.g. $\Box$(req $\rightarrow$ $\Diamond$ack) – “req is always followed by ack”
  - or automata over action labels (see later)
    - e.g. deterministic finite automata (DFAs)
Recap: Model checking for MDPs

- **Property specifications**: quantify over all adversaries
  - e.g. $M \models P_{\geq p}[\phi] \iff Pr_M^\sigma(\phi) \geq p$ for all adversaries $\sigma \in \text{Adv}_M$
  - corresponds to best-/worst-case behaviour analysis
  - requires computation of $Pr_M^{\min}(\phi) = \inf_{\sigma} \{ Pr_{M,s}^{\sigma}(\phi) \}$ or $Pr_M^{\max}(\phi) = \sup_{\sigma} \{ Pr_{M,s}^{\sigma}(\phi) \}$
  - or in a more quantitative fashion:
    - just ask e.g. $P_{\min=?}(\phi)$ or $P_{\max=?}(\phi)$

- **Model checking**: efficient algorithms exist
  - for reachability, graph-based analysis + linear programming
  - in practice, for scalability, often approximate (value iteration)
  - for LTL, first do reachability an automaton–MDP product
  - implemented in tools like PRISM, Liquor, RAPTURE
Parallel composition for MDPs

• The parallel composition of $M_1$ and $M_2$ is denoted $M_1 \parallel M_2$
  – CSP style: synchronise over all common (non-τ) actions
  – when synchronising, transition probabilities are multiplied

• Formally, if $M_i = (S_i, s_{init,i}, \alpha_{M_i}, \delta_{M_i}, L_i)$ for $i=1,2$, then:
  • $M_1 \parallel M_2 = (S_1 \times S_2, (s_{init,1}, s_{init,2}), \alpha_{M_1} \cup \alpha_{M_2}, \delta_{M_1 \parallel M_2}, L_{12})$ where:
    – $L_{12}(s_1, s_2) = L_1(s_1) \cup L_2(s_2)$
    – $\delta_{M_1 \parallel M_2}$ is defined such that $(s_1, s_2) \xrightarrow{a} \mu_1 \times \mu_2$ iff one of:
      • $s_1 \xrightarrow{a} \mu_1$, $s_2 \xrightarrow{a} \mu_2$ and $a \in \alpha_{M_1} \cap \alpha_{M_2}$ (synchronous)
      • $s_1 \xrightarrow{a} \mu_1$, $\mu_2 = \eta_{s_2}$ and $a \in (\alpha_{M_1} \setminus \alpha_{M_2}) \cup \{\tau\}$ (asynchronous)
      • $s_2 \xrightarrow{a} \mu_2$, $\mu_1 = \eta_{s_1}$ and $a \in (\alpha_{M_2} \setminus \alpha_{M_1}) \cup \{\tau\}$ (asynchronous)
    – where $\mu_1 \times \mu_2$ denotes the product of distributions $\mu_1, \mu_2$
    – and $\eta_s \in \text{Dist}(S)$ is the Dirac (point) distribution on $s \in S$
Running example

- Two components, each a Markov decision process:
  - $M_1$: controller which shuts down devices (after warning first)
  - $M_2$: device to be shut down (may fail if no warning sent)

**MDP $M_1$ (“controller”)**

**MDP $M_2$ (“device”)**
Running example

MDP $M_1$ ("controller")

$\begin{align*}
{s_0} & \xrightarrow{\text{detect}, 0.8} {s_1} \\
{s_1} & \xrightarrow{\text{warn}} {s_2} \\
{s_2} & \xrightarrow{\text{shutdown}, 0.2} {s_3} \\
{s_3} & \text{off}
\end{align*}$

MDP $M_2$ ("device")

$\begin{align*}
{t_0} & \xrightarrow{\text{warn}, 0.1} {t_3} \\
{t_3} & \xrightarrow{\text{fail}, 0.9} {t_1} \\
{t_1} & \text{shutdown} \\
{t_2} & \text{off}
\end{align*}$

Parallel composition: $M_1 \parallel M_2$

$\begin{align*}
{s_0, t_0} & \xrightarrow{\text{detect}, 0.8} {s_1, t_0} \\
{s_1, t_0} & \xrightarrow{\text{warn}, 0.9} {s_1, t_2} \\
{s_2, t_0} & \xrightarrow{\text{shutdown}, 0.2} {s_2, t_0} \\
{s_2, t_0} & \xrightarrow{\text{shutdown}, 0.1} {s_2, t_3} \\
{s_2, t_3} & \xrightarrow{\text{fail}, 0.1} \{\text{err}\} \\
{s_3, t_2} & \text{off}
\end{align*}$

System failure:

$\Pr_{M_1 \parallel M_2}^{\max}(\Diamond \text{err}) = 0.02$
Safety properties

• Safety property: language of infinite words (over actions)
  – characterised by a set of “bad prefixes” (or “finite violations”)
  – i.e. finite words of which any extension violates the property

• Regular safety property
  – bad prefixes are represented by a regular language
  – property $A$ stored as deterministic finite automaton (DFA) $A_{err}$
Probabilistic safety properties

- A probabilistic safety property $P_{\geq p}[A]$ comprises
  - a regular safety property $A$ + a rational probability bound $p$
  - “the probability of satisfying $A$ must be at least $p$”
  - $M \models P_{\geq p}[A] \iff \Pr_M^\sigma(A) \geq p$ for all $\sigma \in \text{Adv}_M \iff \Pr_M^{\min}(A) \geq p$

- Examples:
  - “warn occurs before shutdown with probability at least 0.8”
  - “the probability of a failure occurring is at most 0.02”
  - “probability of terminating within $k$ time-steps is at least 0.75”

- Model checking: $\Pr_M^{\min}(A) = 1 - \Pr_{M \otimes A_{\text{err}}}^{\max}(\Diamond \text{err}_A)$
  - where $\text{err}_A$ denotes “accept” states for DFA $A$
  - i.e. construct (synchronous) MDP–DFA product $M \otimes A_{\text{err}}$
  - then compute reachability probabilities on product MDP
• Does probabilistic safety property $P_{\geq 0.8} [A]$ hold in $M_1$?
• Does probabilistic safety property $P_{\geq 0.8} [A]$ hold in $M_1$?

**Running example**

**MDP $M_1$ (“controller”)**

$S_0 \xrightarrow{\text{detect}} S_1 \xrightarrow{0.8} S_2 \xrightarrow{\text{warn}} S_3 \xrightarrow{0.2} S_3$

**A (“warn occurs before shutdown”)**

$q_0 \xrightarrow{\text{warn}} q_1 \xrightarrow{\text{shutdown}} q_2$

**Product MDP $M_1 \otimes A_{\text{err}}$**

$S_0, q_0 \xrightarrow{\text{detect}} S_1, q_0 \xrightarrow{0.8} S_2, q_1 \xrightarrow{\text{warn}} S_3, q_1 \xrightarrow{\text{shutdown}} S_3, q_2 \xrightarrow{\{\text{err}_A\}} S_3, q_2 \xrightarrow{\text{off}}$

$\Pr_{M_1} \min(A)$

$= 1 - \Pr_{M_1 \otimes A_{\text{err}}} \max(\Diamond \text{err}_A)$

$= 1 - 0.2$

$= 0.8$

$\rightarrow M_1 \models P_{\geq 0.8} [A]$
Multi-objective MDP model checking

- Consider multiple (linear-time) objectives for an MDP $M$
  - LTL formulae $\Phi_1, \ldots, \Phi_k$ and probability bounds $\sim_1 p_1, \ldots, \sim_k p_k$
  - question: does there exist an adversary $\sigma \in \text{Adv}_M$ such that:
    $$\Pr_M^{\sigma}(\phi_1) \sim_1 p_1 \land \ldots \land \Pr_M^{\sigma}(\phi_k) \sim_k p_k$$

- Motivating example:
  - $\Pr_M^{\sigma}(\square(\text{queue_size}<10)) > 0.99 \land \Pr_M^{\sigma}(\Diamond \text{flat_battery}) < 0.01$

- Multi-objective MDP model checking [EKVY07]
  - construct product of automata for $M$, $\Phi_1, \ldots, \Phi_k$
  - then solve linear programming (LP) problem
  - the resulting adversary $\sigma$ can obtained from LP solution
  - note: $\sigma$ may be randomised (unlike the single objective case)
• Consider the objectives ◊D and ◊E in the MDP below
  – i.e. the probability of reaching either state D or E
  – a (randomised) adversary resolves the choice between a/b/c
  – increasing the probability of reaching one target decreases the probability of reaching the other
Multi-objective MDP model checking

- Consider the objectives $\Diamond D$ and $\Diamond E$ in the MDP below
  - i.e. the probability of reaching either state D or E
  - a (randomised) adversary resolves the choice between a/b/c
  - increasing the probability of reaching one target decreases the probability of reaching the other

- Considering also randomised adversaries...
  - we obtain a Pareto curve, showing trade-off of optimal solutions
Overview (Part 3)

• Compositional verification
  – assume-guarantee reasoning

• Markov decision processes
  – probabilistic safety properties
  – multi-objective model checking

• Probabilistic assume guarantee
  – semantics, model checking
  – assume-guarantee proof rules
  – quantitative approaches
  – implementation & experimental results
  – assumption generation with learning
Probabilistic assume guarantee

- **Assume–guarantee triples** $\langle A \rangle \geq_{p_A} M \langle G \rangle \geq_{p_G}$ where:
  - $M$ is a Markov decision process
  - $P \geq_{p_A} [A]$ and $P \geq_{p_G} [G]$ are probabilistic safety properties

- **Informally:**
  - “whenever $M$ is part of a system satisfying $A$ with probability at least $p_A$, then the system is guaranteed to satisfy $G$ with probability at least $p_G$”

- **Formally:**

$$
\langle A \rangle \geq_{p_A} M \langle G \rangle \geq_{p_G}
\iff
\forall \sigma \in \text{Adv}_{M[\alpha_A]} ( Pr_{M[\alpha_A]}^\sigma (A) \geq p_A \rightarrow Pr_{M[\alpha_A]}^\sigma (G) \geq p_G )
$$

  - where $M[\alpha_A]$ is $M$ with its alphabet extended to include $\alpha_A$
Assume–guarantee model checking

• Checking whether $\langle A \rangle \geq p_A \ M \langle G \rangle \geq p_G$ is true
  – reduces to multi–objective model checking
  – on the product MDP $M' = M[\alpha_A] \otimes A_{err} \otimes G_{err}$

• More precisely:
  – check no adv. of $M$ satisfying $\Pr_M \sigma (A) \geq p_A$ but not $\Pr_M \sigma (G) \geq p_G$

  $\langle A \rangle \geq p_A \ M \langle G \rangle \geq p_G$
  $\iff$

  $\neg \exists \sigma' \in \text{Adv}_{M'} \ ( \Pr_{M'}^{\sigma'} (\diamond \text{err}_A) \leq 1 - p_A \land \Pr_{M'}^{\sigma'} (\diamond \text{err}_G) > 1 - p_G )$

  – solve via LP problem, i.e. in time polynomial in $|M| \cdot |A_{err}| \cdot |G_{err}|$

• Note: $\langle \text{true} \rangle \ M \langle G \rangle \geq p_G$ denotes the absence of an assumption
  – reduces to standard model checking (since a safety property)
An assume-guarantee rule

- The following asymmetric proof rule holds
  - (symmetric = uses a single assumption about one component)

\[
\begin{align*}
\langle \text{true} \rangle M_1 \langle A \rangle \geq p_A \\
\langle A \rangle \geq p_A \quad M_2 \langle G \rangle \geq p_G \quad \text{(ASYM)} \\
\langle \text{true} \rangle M_1 || M_2 \langle G \rangle \geq p_G
\end{align*}
\]

- So, verifying \( M_1 || M_2 \models P \geq p_c [G] \) requires:
  - premise 1: \( M_1 \models P \geq p_A [A] \) (standard model checking)
  - premise 2: \( \langle A \rangle \geq p_A M_2 \langle G \rangle \geq p_G \) (multi-objective model checking)

- Potentially much cheaper if \( |A| \) much smaller than \( |M_1| \)
Running example

- Does probabilistic safety property $P_{\geq 0.98} [G]$ hold in $M_1 \parallel M_2$?

**MDP $M_1$ (“controller”)**

- $s_0$ to $s_1$: detect, 0.8
- $s_1$ to $s_2$: warn
- $s_2$ to $s_3$: shutdown, 0.2
- $s_3$: off

**MDP $M_2$ (“device”)**

- $t_0$ to $t_1$: warn, 0.9
- $t_1$ to $t_2$: shutdown, 0.1
- $t_2$: off
- $t_3$: fail

$G$ (“a fail action never occurs”)
Running example

- Does probabilistic safety property \( P_{\geq 0.98} [G] \) hold in \( M_1 || M_2 \)?

**MDP \( M_1 \) (“controller”)**

- \( s_0 \) to \( s_1 \): `detect` with 0.8
- \( s_1 \) to \( s_2 \): `warn`
- \( s_2 \) to \( s_3 \): `shutdown` with 0.2
- \( s_3 \) self-loop: `off`

**MDP \( M_2 \) (“device”)**

- \( t_0 \) to \( t_1 \): `warn`
- \( t_1 \) to \( t_2 \): `shutdown` with 0.9
- \( t_2 \) to \( t_0 \): `fail`
- \( t_0 \) self-loop: `off`

**MDP \( G \) (“a fail action never occurs”)**

- \( q_0 \) to \( q_1 \): `fail`, \( q_1 \) self-loop: `fail`

- Use AG with assumption \( \langle A \rangle_{\geq 0.8} \) about \( M_1 \)

\[
\langle true \rangle M_1 \langle A \rangle_{\geq 0.8} \Rightarrow \langle true \rangle M_1 \langle A \rangle_{\geq 0.8} \Rightarrow \langle true \rangle M_1 \langle G \rangle_{\geq 0.98}
\]

\[
\langle true \rangle M_1 || M_2 \langle G \rangle_{\geq 0.98}
\]
Running example

- **Premise 1**: Does $M_1 \models P_{\geq 0.8} [A]$ hold? (same as earlier ex.)

**MDP $M_1$ (“controller”)**

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>detect</td>
<td>$s_1$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$s_2$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>warn</td>
<td>$s_2$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>shutdown</td>
<td>$s_3$</td>
</tr>
<tr>
<td></td>
<td>off</td>
<td>$s_3$</td>
</tr>
</tbody>
</table>

**Event $A$ (“warn occurs before shutdown”)**

<table>
<thead>
<tr>
<th>State</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>warn</td>
</tr>
<tr>
<td></td>
<td>shutdown</td>
</tr>
<tr>
<td>$q_1$</td>
<td>warn, shutdown</td>
</tr>
<tr>
<td>$q_2$</td>
<td>warn, shutdown</td>
</tr>
</tbody>
</table>

**Product MDP $M_1 \otimes A_{err}$**

<table>
<thead>
<tr>
<th>State</th>
<th>Action</th>
<th>Next State</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0, q_0$</td>
<td>detect</td>
<td>$s_1, q_0$</td>
</tr>
<tr>
<td>$s_1, q_0$</td>
<td>warn</td>
<td>$s_2, q_1$</td>
</tr>
<tr>
<td>$s_2, q_0$</td>
<td>shutdown</td>
<td>$s_3, q_2$</td>
</tr>
<tr>
<td>$s_2, q_0$</td>
<td>shutdown</td>
<td>$s_3, q_2$</td>
</tr>
<tr>
<td>$s_3, q_1$</td>
<td>shutdown</td>
<td>$s_3, q_2$</td>
</tr>
<tr>
<td>$s_3, q_2$</td>
<td>off</td>
<td>$s_3, q_2$</td>
</tr>
</tbody>
</table>

**Pr$_{M_1}^\text{min}(A)$**

$$= 1 - Pr_{M_1 \otimes A_{err}}^\text{max}(\diamond \text{err}_A)$$

$$= 1 - 0.2$$

$$= 0.8$$

$$\rightarrow M_1 \models P_{\geq 0.8} [A]$$
• Premise 2: Does $\langle A \rangle \geq 0.8$ $M_2$ $\langle G \rangle \geq 0.98$ hold?

**MDP $M_2$ (“device”)**

$\begin{align*}
&\text{t}_0 \quad \text{t}_1 \\
\text{warn} & \quad \text{shutdown} \\
\text{t}_3 & \quad \text{t}_2 \\
\text{fail} & \quad \text{off}
\end{align*}$

$\begin{align*}
\text{A (“warn occurs before shutdown”)}
\end{align*}$

$\begin{align*}
\text{G (“a fail action never occurs”)}
\end{align*}$

**Product MDP**

$M' = M_2[\alpha_A] \otimes A_{err} \otimes G_{err}$
Running example

• Premise 2: Does \( \langle A \rangle \geq 0.8 \) \( M_2 \langle G \rangle \geq 0.98 \) hold?

Product MDP
\[ M' = M_2[\alpha_A] \otimes A_{\text{err}} \otimes G_{\text{err}} \]

• \( \exists \) an adversary of \( M_2 \) satisfying \( Pr_{M^\sigma}(A) \geq 0.8 \) but not \( Pr_{M^\sigma}(G) \geq 0.98 \)?
  \[ \Leftrightarrow \]
  • \( \exists \) an adversary of \( M' \) with \( Pr_{M'^{\sigma'}}(\Diamond \text{err}_A) \leq 0.2 \) and \( Pr_{M'^{\sigma'}}(\Diamond \text{err}_G) > 0.02 \)?

• To satisfy \( Pr_{M'^{\sigma'}}(\Diamond \text{err}_A) \leq 0.2 \), adversary \( \sigma' \) must choose shutdown in initial state with probability \( \leq 0.2 \), which means \( Pr_{M'^{\sigma'}}(\Diamond \text{err}_G) \leq 0.02 \)

• So, there is no such adversary and \( \langle A \rangle \geq 0.8 \) \( M_2 \langle G \rangle \geq 0.98 \) does hold
Other assume-guarantee rules

Multiple assumptions:
\[
\langle \text{true} \rangle \ M_1 \langle A_1, \ldots, A_k \rangle \geq p_1, \ldots, p_k
\]
\[
\langle A_1, \ldots, A_k \rangle \geq p_1, \ldots, p_k \ M_2 \langle G \rangle \geq p_G
\]
\[
\langle \text{true} \rangle \ M_1 || M_2 \langle G \rangle \geq p_G
\]

Circular rule:
\[
\langle \text{true} \rangle \ M_2 \langle A_1 \rangle \geq p_2
\]
\[
\langle A_2 \rangle \geq p_2 \ M_1 \langle A_1 \rangle \geq p_1
\]
\[
\langle A_1 \rangle \geq p_1 \ M_2 \langle G \rangle \geq p_G
\]
\[
\langle \text{true} \rangle \ M_1 || M_2 \langle G \rangle \geq p_G
\]

Multiple components (chain):
\[
\langle \text{true} \rangle \ M_1 \langle A_1 \rangle \geq p_1
\]
\[
\langle A_1 \rangle \geq p_1 \ M_2 \langle A_2 \rangle \geq p_2
\]
\[
\ldots
\]
\[
\langle A_n \rangle \geq p_n \ M_n \langle G \rangle \geq p_G
\]
\[
\langle \text{true} \rangle \ M_1 || \ldots || M_n \langle G \rangle \geq p_G
\]

Asynchronous components:
\[
\langle A_1 \rangle \geq p_1 \ M_1 \langle G_1 \rangle \geq q_1
\]
\[
\langle A_2 \rangle \geq p_2 \ M_2 \langle G_2 \rangle \geq q_2
\]
\[
\langle A_1, A_2 \rangle \geq p_{12} \ M_1 || M_2 \langle G_1 \lor G_2 \rangle \geq \left( q_1 + q_2 - q_{12} \right)
\]
A quantitative approach

- For (non-compositional) probabilistic verification
  - prefer quantitative properties: $\Pr_{M}^{\min}(G)$, not $M \models P_{\geq p_{G}}[G]$
  - can we do this for compositional verification?

- Consider, for example, AG rule (ASYM)
  - this proves $\Pr_{M_{1} \parallel M_{2}}^{\min}(G) \geq p_{G}$ for certain values of $p_{G}$
  - i.e. gives lower bound for $\Pr_{M_{1} \parallel M_{2}}^{\min}(G)$
  - for a fixed assumption $A$, we can compute the maximal lower bound obtainable, through a simple adaption of the multi-objective model checking problem
  - we can also compute upper bounds using generated adversaries as witnesses
  - furthermore: can explore trade-offs in parameterised models by approximating Pareto curves
Implementation + Case studies

- **Prototype extension of PRISM model checker**
  - already supports LTL for Markov decision processes
  - automata can be encoded in modelling language
  - added support for multi-objective LTL model checking, using LP solvers (ECLiPSe/COIN-OR CBC)

- **Two large case studies**
  - randomised consensus algorithm (Aspnes & Herlihy)
    - minimum probability consensus reached by round $R$
  - Zeroconf network protocol
    - maximum probability network configures incorrectly
    - minimum probability network configured by time $T$
Case study: Randomised consensus

- **Distributed consensus protocol**
  - algorithm run by a collection of distributed processes
  - processes each have some (nondeterministic) initial value
  - processes must eventually terminate, agreeing on same value

- **Aspnes/Herlihy randomised distributed consensus [AH90]**
  - consensus algorithm for N processes, operates in rounds
  - each round uses a shared coin protocol, parameterised by K

- **We check:**
  - “minimum probability consensus reached by round R”
  - captured as a probabilistic safety property with DFA representing any run where a process enters round R+1
Case study: Randomised consensus

- **Model structure: parallel composition of:**
  - N MDPs, each representing one process
  - R MDPs, one for the shared coin protocol of each round

- **Compositional verification:**
  - model check a probabilistic safety property for each coin protocol from rounds 1, ..., R−2
  - safety property: minimum probability that the coin protocol returns the same coin value for all processes
  - combine these results through R−2 applications of the “asynchronous” rule, proving a probabilistic safety property about the parallel composition of the R−2 coin protocols
  - this probabilistic safety property is used as the assumption for an application of the (ASYM) rule, yielding the final property
## Experimental results

<table>
<thead>
<tr>
<th>Case study [parameters]</th>
<th>Non–compositional</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>States</td>
<td>Time (s)</td>
</tr>
<tr>
<td><strong>Randomised consensus</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3 processes) [R,K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3, 2</td>
<td>1,418,545</td>
<td>18,971</td>
</tr>
<tr>
<td>3, 20</td>
<td>39,827,233</td>
<td>time–out</td>
</tr>
<tr>
<td>4, 2</td>
<td>150,487,585</td>
<td>78,955</td>
</tr>
<tr>
<td>4, 20</td>
<td>2,028,200,209</td>
<td>mem–out</td>
</tr>
<tr>
<td><strong>ZeroConf [K]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>313,541</td>
<td>103.9</td>
</tr>
<tr>
<td>6</td>
<td>811,290</td>
<td>275.2</td>
</tr>
<tr>
<td>8</td>
<td>1,892,952</td>
<td>592.2</td>
</tr>
<tr>
<td><strong>ZeroConf time–bounded</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[K, T]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, 10</td>
<td>65,567</td>
<td>46.3</td>
</tr>
<tr>
<td>2, 14</td>
<td>106,177</td>
<td>63.1</td>
</tr>
<tr>
<td>4, 10</td>
<td>976,247</td>
<td>88.2</td>
</tr>
<tr>
<td>4, 14</td>
<td>2,288,771</td>
<td>128.3</td>
</tr>
</tbody>
</table>
### Experimental results

<table>
<thead>
<tr>
<th>Case study [parameters]</th>
<th>Non-compositional</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>States</td>
<td>Time (s)</td>
</tr>
<tr>
<td><strong>Randomised consensus</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3 processes) [R,K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3, 2</td>
<td>1,418,545</td>
<td>18,971</td>
</tr>
<tr>
<td>3, 20</td>
<td>39,827,233</td>
<td>time-out</td>
</tr>
<tr>
<td>4, 2</td>
<td>150,487,585</td>
<td>78,955</td>
</tr>
<tr>
<td>4, 20</td>
<td>2,028,200,209</td>
<td>mem-out</td>
</tr>
<tr>
<td><strong>ZeroConf [K]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>313,541</td>
<td>103.9</td>
</tr>
<tr>
<td>6</td>
<td>811,290</td>
<td>275.2</td>
</tr>
<tr>
<td>8</td>
<td>1,892,952</td>
<td>592.2</td>
</tr>
<tr>
<td><strong>ZeroConf time-bounded</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[K, T]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, 10</td>
<td>65,567</td>
<td>46.3</td>
</tr>
<tr>
<td>2, 14</td>
<td>106,177</td>
<td>63.1</td>
</tr>
<tr>
<td>4, 10</td>
<td>976,247</td>
<td>88.2</td>
</tr>
<tr>
<td>4, 14</td>
<td>2,288,771</td>
<td>128.3</td>
</tr>
</tbody>
</table>

- Faster than conventional model checking in a number of cases
## Experimental results

<table>
<thead>
<tr>
<th>Case study [parameters]</th>
<th>Non-compositional</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>States</td>
<td>Time (s)</td>
</tr>
<tr>
<td>Randomised consensus (3 processes) [R,K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3, 2</td>
<td>1,418,545</td>
<td>18,971</td>
</tr>
<tr>
<td>3, 20</td>
<td>39,827,233</td>
<td>time-out</td>
</tr>
<tr>
<td>4, 2</td>
<td>150,487,585</td>
<td>78,955</td>
</tr>
<tr>
<td>4, 20</td>
<td>2,028,200,209</td>
<td>mem-out</td>
</tr>
<tr>
<td>ZeroConf [K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>313,541</td>
<td>103.9</td>
</tr>
<tr>
<td>6</td>
<td>811,290</td>
<td>275.2</td>
</tr>
<tr>
<td>8</td>
<td>1,892,952</td>
<td>592.2</td>
</tr>
<tr>
<td>ZeroConf time-bounded [K, T]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, 10</td>
<td>65,567</td>
<td>46.3</td>
</tr>
<tr>
<td>2, 14</td>
<td>106,177</td>
<td>63.1</td>
</tr>
<tr>
<td>4, 10</td>
<td>976,247</td>
<td>88.2</td>
</tr>
<tr>
<td>4, 14</td>
<td>2,288,771</td>
<td>128.3</td>
</tr>
</tbody>
</table>

- Verified instances where conventional model checking is infeasible
## Experimental results

<table>
<thead>
<tr>
<th>Case study [parameters]</th>
<th>Non-compositional</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>States</td>
<td>Time (s)</td>
</tr>
<tr>
<td>Randomised consensus (3 processes) [R,K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3, 2</td>
<td>1,418,545</td>
<td>18,971</td>
</tr>
<tr>
<td>3, 20</td>
<td>39,827,233</td>
<td>time–out</td>
</tr>
<tr>
<td>4, 2</td>
<td>150,487,585</td>
<td>78,955</td>
</tr>
<tr>
<td>4, 20</td>
<td>2,028,200,209</td>
<td>mem–out</td>
</tr>
<tr>
<td>ZeroConf [K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>313,541</td>
<td>103.9</td>
</tr>
<tr>
<td>6</td>
<td>811,290</td>
<td>275.2</td>
</tr>
<tr>
<td>8</td>
<td>1,892,952</td>
<td>592.2</td>
</tr>
<tr>
<td>ZeroConf time–bounded [K, T]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, 10</td>
<td>65,567</td>
<td>46.3</td>
</tr>
<tr>
<td>2, 14</td>
<td>106,177</td>
<td>63.1</td>
</tr>
<tr>
<td>4, 10</td>
<td>976,247</td>
<td>88.2</td>
</tr>
<tr>
<td>4, 14</td>
<td>2,288,771</td>
<td>128.3</td>
</tr>
</tbody>
</table>

- LP problem generally much smaller than full state space (but still the limiting factor)
Overview (Part 3)

- Compositional verification
  - assume-guarantee reasoning

- Markov decision processes
  - probabilistic safety properties
  - multi-objective model checking

- Probabilistic assume guarantee
  - semantics, model checking
  - assume-guarantee proof rules
  - quantitative approaches
  - implementation & experimental results
  - assumption generation with learning
Generating assumptions

• We can verify $M_1 || M_2$ compositionally
  – but this relies on the existence of a suitable assumption $\langle A \rangle \geq p_A$

• 1. Does such an assumption always exist?
• 2. When it does exist, can we generate it automatically?

• One possibility: use algorithmic learning techniques
  – inspired by non–probabilistic AG work of [Pasareanu et al.]
  – uses L* algorithm to learn finite automata for assumptions
  – successful implementations using Boolean functions [Chen/Clarke/et al.] and BDD–based techniques [Alur et al.]

• We use a modified version of L*
  – to learn probabilistic assumptions for rule (ASYM)
L* for assume-guarantee

- **L* algorithm [Angluin]** – learns regular languages (as a DFA)
  - relies on existence of a “teacher” to guide the learning
  - answers two type of queries: “membership” and “conjecture”
  - membership: “is word $w$ in the target language $L$?”
  - conjecture: “does automata $A$ accept the target language $L$”?!
  - if not, teacher must return counterexample $w'$
  - L* produces minimal DFA, runs in polynomial time

- **Successfully applied to the of learning assumptions for AG**
  - uses notion of “weakest assumption” about a component that suffices for compositional verification (always exists)
  - weakest assumption is the target regular language
  - model checker plays role of teacher, returns counterexamples
  - in practice, can usually stop early: either with a simpler (stronger) assumption or by refuting the property
Key steps of (modified) L*

- **Key idea:** learn probabilistic assumption $\langle A \rangle \geq p_A$
  - via non-probabilistic assumption $A$

- **Membership** query (for trace $t$):
  - does $t \parallel M_2 \models P_{\geq p_G} [G]$ hold?

- **“Conjecture”** query (for assumption $A$)
  - 1. compute lowest value of $p_A$ such that $\langle A \rangle \geq p_A$ $M_2 \langle G \rangle \geq p_G$ holds
    - if no such value, need to refine $A$
  - 2. check if $M_1 \models P_{\geq p_A} [A]$ holds
    - if yes, successfully verified $\langle G \rangle \geq p_G$ for $M_1 \parallel M_2$ (with $\langle A \rangle \geq p_A$)
  - 3. check if counterexample from 2 is real
    - if yes, have refuted $\langle G \rangle \geq p_G$ for $M_1 \parallel M_2$
    - if no, need to refine $A$
  - (use probabilistic counterexamples [HK07] to “refine $A$”)
**Experimental results (learning)**

<table>
<thead>
<tr>
<th>Case study [parameters]</th>
<th>Component sizes</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>M_2 \otimes G_{err}</td>
</tr>
<tr>
<td>Client-server (N failures) [N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>1,121</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>5,397</td>
<td>36</td>
</tr>
<tr>
<td>Randomised consensus [N,R,K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, 3, 20</td>
<td>391</td>
<td>3,217</td>
</tr>
<tr>
<td>2, 4, 2</td>
<td>573</td>
<td>113,569</td>
</tr>
<tr>
<td>3, 3, 2</td>
<td>8,843</td>
<td>4,065</td>
</tr>
<tr>
<td>3, 3, 20</td>
<td>8,843</td>
<td>38,193</td>
</tr>
<tr>
<td>Sensor network [N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>42</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>1,184</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>10,662</td>
</tr>
</tbody>
</table>
## Experimental results (learning)

<table>
<thead>
<tr>
<th>Case study [parameters]</th>
<th>Component sizes</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>M_2 \otimes G_{err}</td>
</tr>
<tr>
<td><strong>Client-server (N failures) [N]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>1,121</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>5,397</td>
<td>36</td>
</tr>
<tr>
<td><strong>Randomised consensus [N,R,K]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, 3, 20</td>
<td>391</td>
<td>3,217</td>
</tr>
<tr>
<td>2, 4, 2</td>
<td>573</td>
<td>113,569</td>
</tr>
<tr>
<td>3, 3, 2</td>
<td>8,843</td>
<td>4,065</td>
</tr>
<tr>
<td>3, 3, 20</td>
<td>8,843</td>
<td>38,193</td>
</tr>
<tr>
<td><strong>Sensor network [N]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>42</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>1,184</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>10,662</td>
</tr>
</tbody>
</table>

- Successfully learnt (small) assumptions in all cases
### Experimental results (learning)

<table>
<thead>
<tr>
<th>Case study [parameters]</th>
<th>Component sizes</th>
<th>Compositional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$</td>
<td>M_2 \otimes G_{err}</td>
</tr>
<tr>
<td><strong>Client-server</strong> (N failures) [N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>1,121</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>5,397</td>
<td>36</td>
</tr>
<tr>
<td><strong>Randomised consensus</strong> [N,R,K]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2, 3, 20</td>
<td>391</td>
<td>3,217</td>
</tr>
<tr>
<td>2, 4, 2</td>
<td>573</td>
<td>113,569</td>
</tr>
<tr>
<td>3, 3, 2</td>
<td>8,843</td>
<td>4,065</td>
</tr>
<tr>
<td>3, 3, 20</td>
<td>8,843</td>
<td>38,193</td>
</tr>
<tr>
<td><strong>Sensor network</strong> [N]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>42</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>1,184</td>
</tr>
<tr>
<td>3</td>
<td>42</td>
<td>10,662</td>
</tr>
</tbody>
</table>

- In some cases, learning + compositional verification is faster (than non-compositional verification, using PRISM)
Summary (Part 3)

- Compositional verification, e.g. assume–guarantee
  - decompose verification problem based on system structure

- Compositional probabilistic verification based on:
  - Markov decision processes, with arbitrary parallel composition
  - assumptions/guarantees are probabilistic safety properties
  - reduction to multi–objective model checking
  - multiple proof rules; adapted to quantitative approach
  - automatic generation of assumptions: L* learning

- Can work well in practice
  - verified safety/performance on several large case studies
  - cases where infeasible using non–compositional verification

- For further detail, see [KNPQ10], [FKP10]

- Next: Probabilistic timed automata (PTAs)