Assume-Guarantee Verification for Probabilistic Systems

Dave Parker

University of Oxford

Dagstuhl, February 2010

Joint work with Marta Kwiatkowska, Gethin Norman, Hongyang Qu

Context

Analysis of systems exhibiting:

- probabilistic behaviour (e.g. randomisation, failures)
- nondeterminism (e.g. concurrency, underspecification)
- timed behaviour (e.g. delays, time-outs)

Probabilistic verification

- probabilistic automata, temporal logics, model checking
- emphasis on quantitative properties, e.g. "what is the minimum probability of terminating within k time-units?"

Aim: improve scalability of existing tools/techniques

- compositional approaches: assume-guarantee verification
- focus on efficient, fully-automated techniques

Overview

Compositional verification

- assume-guarantee reasoning

Probabilistic automata

- probabilistic safety properties
- multi-objective model checking

Probabilistic assume-guarantee

- semantics, model checking and proof rules
- quantitative approaches

Implementation, case studies and results

Conclusions, current & future work

Compositional verification

Goal: scalability through modular verification

- e.g. decide if $\mathrm{M}_1 \,\|\, \mathrm{M}_2 \models \mathrm{G}$
- by analysing M_1 and M_2 separately

Assume-guarantee (AG) reasoning

- use assumptions A about the context of a component M
- $\langle A \rangle \, \mathrm{M} \, \langle \mathrm{G} \rangle$ – "whenever M is part of a system that satisfies A, then the system must also guarantee G"
- example of asymmetric (non-circular) AG rule:

$$\frac{\langle \mathit{true} \rangle \, \mathrm{M}_1 \, \langle A \rangle \qquad \langle A \rangle \, \mathrm{M}_2 \, \langle \mathrm{G} \rangle}{\langle \mathit{true} \rangle \, \mathrm{M}_1 \,\|\, \mathrm{M}_2 \, \langle \mathrm{G} \rangle}$$

[Păsăreanu/Giannakopoulou/et al.]

Assume guarantee for probabilistic systems

How to formulate AG rules for probabilistic automata?

$$\frac{\langle true \rangle\, \mathrm{M}_1\, \langle \mathrm{A} \rangle \qquad \langle \mathrm{A} \rangle\, \mathrm{M}_2\, \langle \mathrm{G} \rangle}{\langle true \rangle\, \mathrm{M}_1 \,\|\, \mathrm{M}_2\, \langle \mathrm{G} \rangle}$$

Questions:

- What form do assumptions and guarantees take?
- What does $\langle \mathrm{A} \rangle\, \mathrm{M}\, \langle \mathrm{G} \rangle$ mean? How to check it?
- Any restriction on parallel composition $\mathrm{M}_1 \,\|\, \mathrm{M}_2$?
- Can we do this in a "quantitative" way?

Assume guarantee for probabilistic systems

How to formulate AG rules for probabilistic automata?

$$\frac{\langle true \rangle\, \mathrm{M}_1\, \langle \mathrm{A} \rangle \qquad \langle \mathrm{A} \rangle\, \mathrm{M}_2\, \langle \mathrm{G} \rangle}{\langle true \rangle\, \mathrm{M}_1 \,\|\, \mathrm{M}_2\, \langle \mathrm{G} \rangle}$$

Questions:

- What form do assumptions and guarantees take?
  - probabilistic safety properties
- What does $\langle \mathrm{A} \rangle\, \mathrm{M}\, \langle \mathrm{G} \rangle$ mean? How to check it?
  - reduction to multi-objective model checking
- Any restriction on parallel composition $\mathrm{M}_1 \,\|\, \mathrm{M}_2$?
  - no: arbitrary parallel composition
- Can we do this in a "quantitative" way?
  - yes: generate lower/upper bounds on probabilities

Probabilistic automata (PAs) [Segala/Lynch]

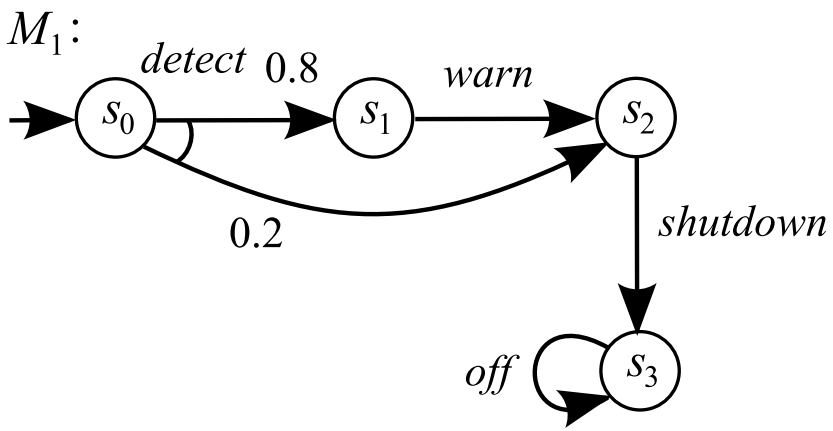
Probabilistic automaton $M = (S, \bar{s}, \alpha_M, \delta_M, L)$ **with:**

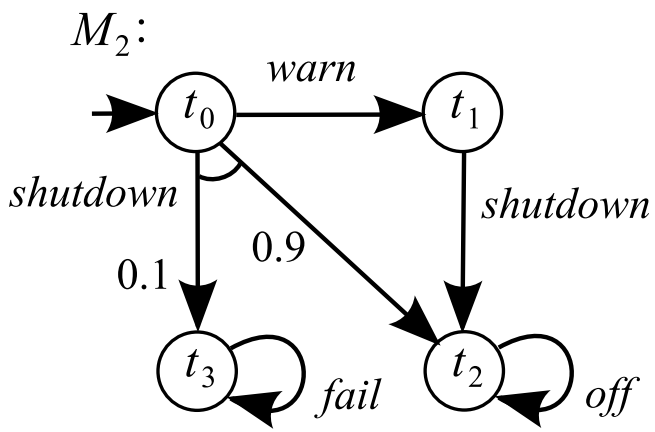
- state space S and initial state $\bar{s}$
- alphabet (actions) α_M and state labelling L
- transition relation: $\delta_M \subseteq S \times (\alpha_M \cup \{\tau\}) \times Dist(S)$

Parallel composition of PAs

- synchronisation over common actions (CSP-style)

Example: $M_1 \parallel M_2$





Probabilistic automata: Model checking

An adversary σ resolves nondeterminism in a PA M

- induces probability measure Pr^{σ}_{M} over (infinite) paths

Property specifications (linear-time)

- specify some measurable property ϕ of paths
- 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

Model checking: quantify over all adversaries Adv_M

- e.g. $M \models P_{\geqslant p}[\,\phi\,] \;\Leftrightarrow\; Pr^{\sigma}_{M}(\phi) \geqslant p$ for all adversaries σ
- or in a more quantitative fashion:
- just compute e.g. $Pr^{\min}_{M}(\,\phi\,) = \inf_{\sigma} Pr^{\sigma}_{M}(\phi)$
- efficient algorithms (and tool support) exist

Safety properties

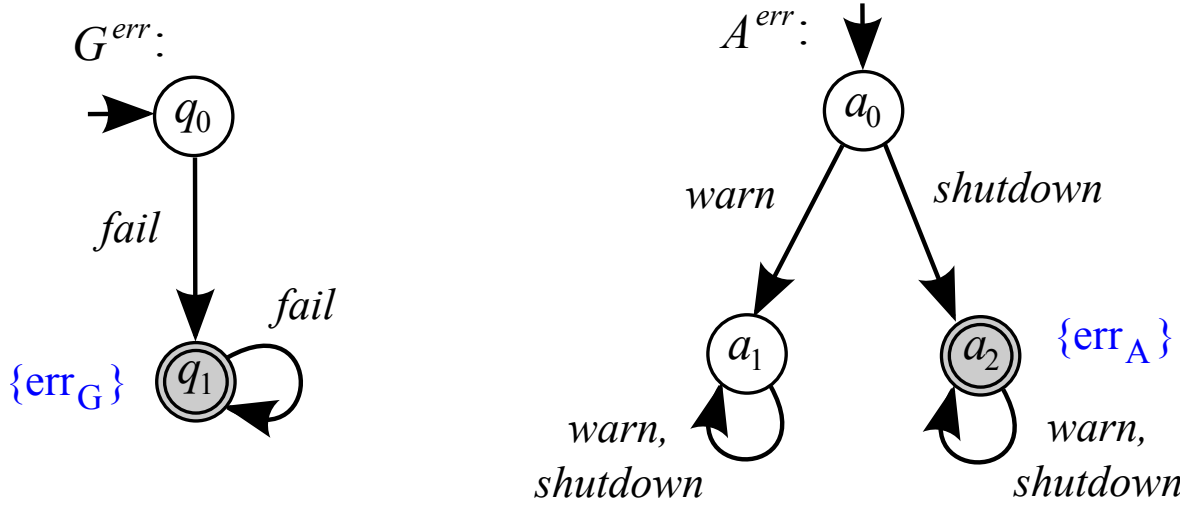
Safety property: language of infinite words (over actions)

- characterised by a set of bad prefixes, finite words of which any extension violates the property

Regular safety property

- bad prefixes are represented by a regular language
- property X stored as deterministic finite automaton X^{err}

Examples



G: "*fail* never occurs" A: "*warn* occurs before *shutdown*"

Probabilistic safety properties

A probabilistic safety property $\mathrm{P}_{\geqslant \mathrm{p}}[A]$ comprises:

- regular safety property A + (lower) probability bound p

i.e. "the probability of satisfying A must be at least p"

$$\mathrm{M} \models \mathrm{P}_{\geqslant \mathrm{p}}[A] \quad \Leftrightarrow \quad Pr^{\min}_{\mathrm{M}}(A) \geqslant \mathrm{p}\,.$$

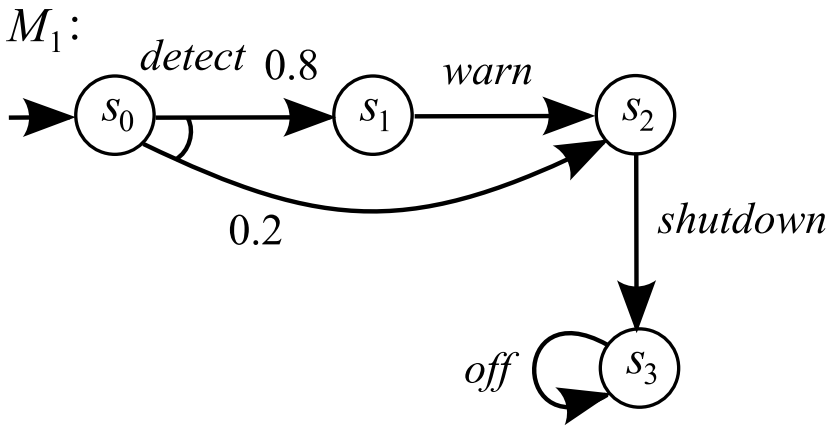
Examples

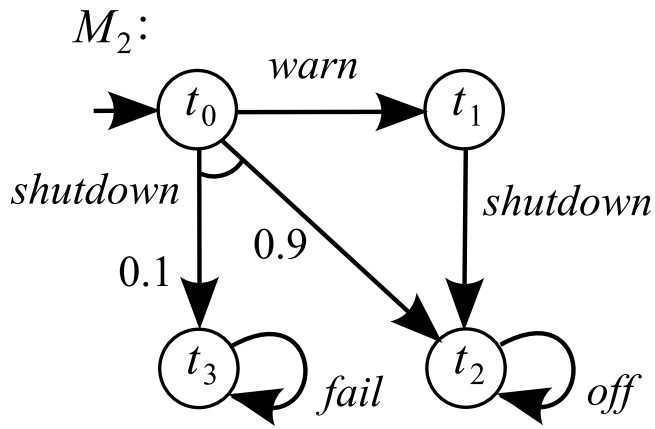
- "event $warn$ always occurs before event $shutdown$ with probability at least 0.98"
- "the probability of $fail$ occurring is at most 0.01"
- "the probability of the protocol terminating within k time-units is at least 0.75"

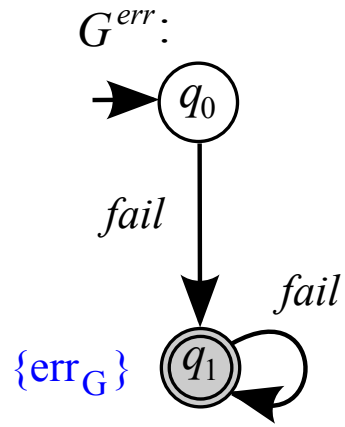
Model checking: reachability on (synchronous) product

- $Pr^{\min}_{\mathrm{M}}(A) = 1 - Pr^{\max}_{\mathrm{M}\otimes \mathrm{A}^{err}}(\Diamond err_{\mathrm{A}})$

Example







G: "*fail* never occurs"

$$Pr^{\max}_{(M_1 \| M_2) \otimes G^{err}}(\Diamond err_G) = 0.02$$

$$\Rightarrow Pr^{\min}_{M_1 \| M_2}(G) = 0.98$$

$$\Rightarrow M_1 \| M_2 \models P_{\geqslant 0.98}[G]$$

Multi-objective model checking for PAs

Multiple (linear-time) objectives for a PA M

- [Etessami/Kwiatkowska/Vardi/Yannakakis, TACAS'07]
- LTL formulae $\phi_1, \ldots, \phi_k$, prob. bounds $\sim_1 p_1, \ldots, \sim_k p_k$
- does there exist an adversary σ such that $\bigwedge_{i=1}^{k} Pr_M^{\sigma}(\phi_i) \sim_i p_i$?

Example:

- $Pr_M^{\sigma}(\Box(queue\_size < 10)) \geqslant 0.99 \wedge$ $Pr_M^{\sigma}(\Diamond(flat\_battery)) < 0.01$

Multi-objective model checking

- construct product of automata for M, ϕ_i
- then solve linear programming problem

Assume-guarantee semantics

Assume-guarantee triples $\langle A \rangle_{\geqslant p_A} M \langle G \rangle_{\geqslant p_G}$ where:

- M is a probabilistic automaton
- $P_{\geqslant p_A}[A]$ and $P_{\geqslant p_G}[G]$ are probabilistic safety properties

Informally:

- "whenever M is part of a system satisfying A with probability at least p_A, then the system will satisfy G with probability at least p_G"

Formally:

$$\langle A \rangle_{\geqslant p_A} M \langle G \rangle_{\geqslant p_G}$$

$$\Leftrightarrow$$

$$\forall \sigma \in Adv_{M[\alpha_A]} \cdot \left(Pr^{\sigma}_{M[\alpha_A]}(A) \geqslant p_A \rightarrow Pr^{\sigma}_{M[\alpha_A]}(G) \geqslant p_G \right)$$

(where $M[\alpha_A]$ is M with its alphabet extended to α_A)

Assume-guarantee model checking

Checking whether $\langle A \rangle_{\geqslant p_A} M \langle G \rangle_{\geqslant p_G}$ is true

- reduces to multi-objective model checking
- on the product PA $M' = M[\alpha_A] \otimes A^{err} \otimes G^{err}$

More precisely:

- check no adversary satisfying $P_{\geqslant p_A}[A]$ but not $P_{\geqslant p_G}[G]$

$$\langle A \rangle_{\geqslant p_A} M \langle G \rangle_{\geqslant p_G}$$
$$\Leftrightarrow$$
$$\neg \exists \sigma' \in Adv_{M'} \left(Pr^{\sigma'}_{M'}(\Diamond err_A) \leqslant 1 - p_A \wedge Pr^{\sigma'}_{M'}(\Diamond err_G) > 1 - p_G \right)$$

- can be checked by solving a linear programming problem
- i.e. in time polynomial in $|M| \cdot |A^{err}| \cdot |G^{err}|$

An assume-guarantee rule

The following asymmetric proof rule holds:

- i.e. uses a single assumption about one component

$$\frac{\langle \mathit{true} \rangle\, \mathrm{M}_1\, \langle \mathrm{A} \rangle_{\geqslant \mathrm{p}_\mathrm{A}} \qquad \langle \mathrm{A} \rangle_{\geqslant \mathrm{p}_\mathrm{A}}\, \mathrm{M}_2\, \langle \mathrm{G} \rangle_{\geqslant \mathrm{p}_\mathrm{G}}}{\langle \mathit{true} \rangle\, \mathrm{M}_1 \parallel \mathrm{M}_2\, \langle \mathrm{G} \rangle_{\geqslant \mathrm{p}_\mathrm{G}}} \quad (\textsc{Asym})$$

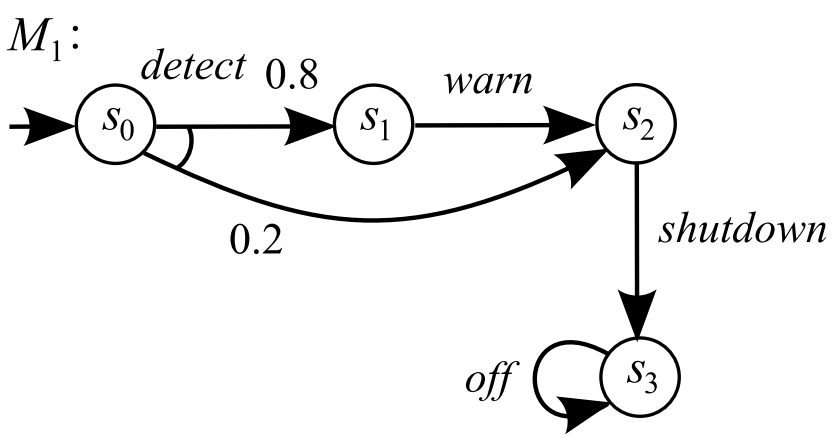
So, verifying $\mathrm{M}_1 \parallel \mathrm{M}_2 \models \mathrm{P}_{\geqslant \mathrm{p}_\mathrm{G}}[\,\mathrm{G}\,]$ requires:

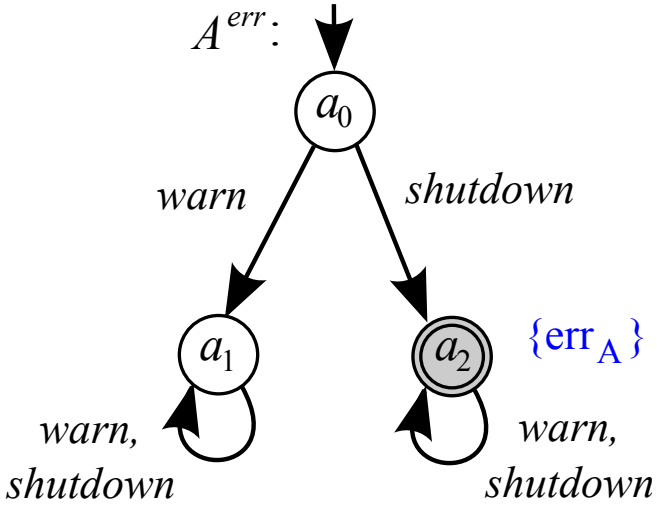
- premise 1: $\mathrm{M}_1 \models \mathrm{P}_{\geqslant \mathrm{p}_\mathrm{A}}[\,\mathrm{A}\,]$ (prob. reachability)
- premise 2: $\langle \mathrm{A} \rangle_{\geqslant \mathrm{p}_\mathrm{A}}\, \mathrm{M}_2\, \langle \mathrm{G} \rangle_{\geqslant \mathrm{p}_\mathrm{G}}$ (multi-objective)

(potentially much cheaper if $|\mathrm{A}|$ much smaller than $|\mathrm{M}_1|$)

Example

Premise 1: $M_1 \models P_{\geqslant 0.8}[A]$ **?**



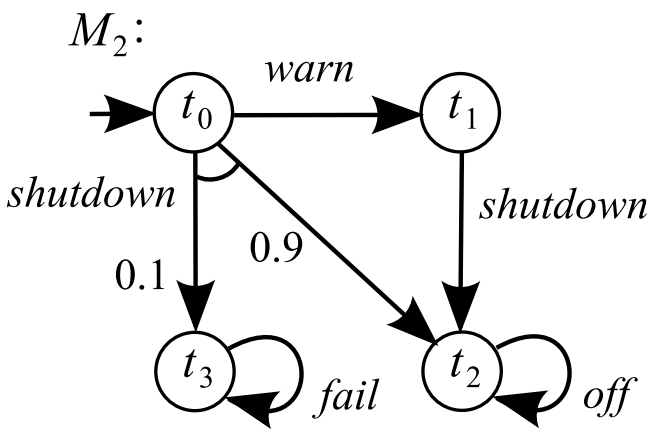


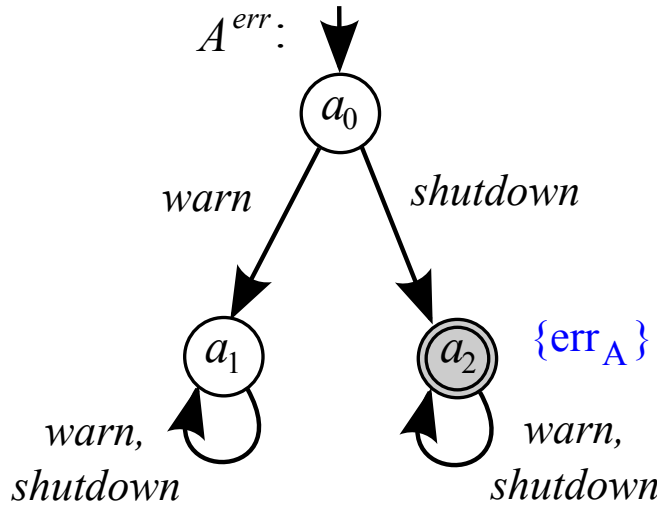
$$Pr^{\min}_{M_1}(A) = 0.8$$

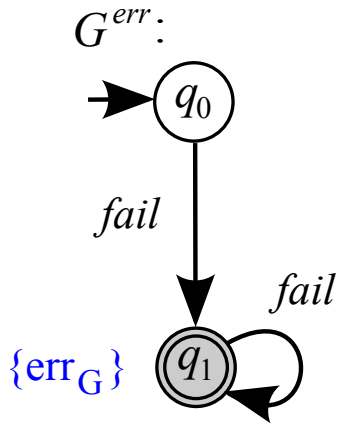
$$\Rightarrow M_1 \models P_{\geqslant 0.8}[A]$$

Example...

Premise 2: $\langle A \rangle_{\geqslant 0.8}\, M_2\, \langle G \rangle_{\geqslant 0.98}$ **?**



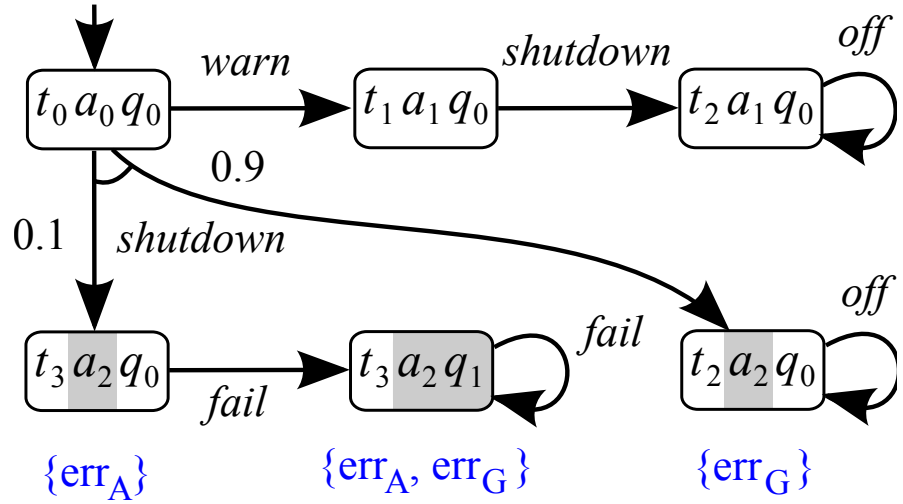




Example...

Premise 2: $\langle A \rangle_{\geqslant 0.8}\, M_2\, \langle G \rangle_{\geqslant 0.98}$ **?**

$M' = M_2 \otimes A^{err} \otimes G^{err}$:



– there is no adversary $\sigma' \in Adv_{M'}$ under which $Pr^{\sigma'}_{M'}(\Diamond err_A) \leqslant 1-0.8$ and $Pr^{\sigma'}_{M'}(\Diamond err_G) > 1-0.98$

$\Rightarrow \langle A \rangle_{\geqslant 0.8}\, M_2\, \langle G \rangle_{\geqslant 0.98}$

Other assume-guarantee rules

Multiple assumptions:

$$\frac{\langle true\rangle\ \mathrm{M}_1\ \langle \mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{k}\rangle_{\geqslant \mathrm{p}_1,\ldots,\mathrm{p}_\mathrm{k}} \qquad \langle \mathrm{A}_1, \ldots, \mathrm{A}_\mathrm{k}\rangle_{\geqslant \mathrm{p}_1,\ldots,\mathrm{p}_\mathrm{k}}\ \mathrm{M}_2\ \langle \mathrm{G}\rangle_{\geqslant \mathrm{p}_\mathrm{G}}}{\langle true\rangle\ \mathrm{M}_1 \parallel \mathrm{M}_2\ \langle \mathrm{G}\rangle_{\geqslant \mathrm{p}_\mathrm{G}}} \qquad (\textsc{Asym-Mult})$$

Circular rule:

$$\frac{\begin{array}{c}\langle true\rangle\ \mathrm{M}_2\ \langle \mathrm{A}_2\rangle_{\geqslant \mathrm{p}_2} \\ \langle \mathrm{A}_2\rangle_{\geqslant \mathrm{p}_2}\ \mathrm{M}_1\ \langle \mathrm{A}_1\rangle_{\geqslant \mathrm{p}_1} \\ \langle \mathrm{A}_1\rangle_{\geqslant \mathrm{p}_1}\ \mathrm{M}_2\ \langle \mathrm{G}\rangle_{\geqslant \mathrm{p}_\mathrm{G}}\end{array}}{\langle true\rangle\ \mathrm{M}_1 \parallel \mathrm{M}_2\ \langle \mathrm{G}\rangle_{\geqslant \mathrm{p}_\mathrm{G}}} \qquad (\textsc{Circ})$$

Other rules:

- chains of components; asynchronous components

Quantitative approaches

For (non-compositional) probabilistic verification

- prefer quantitative properties: $Pr^{\min}_{\mathrm{M}}(\mathrm{G})$, not $\mathrm{P}_{\geqslant \mathrm{p_G}}[\mathrm{G}]$
- can we do this for compositional verification?

Consider, for example, AG rule (ASym)

- proves $Pr^{\min}_{\mathrm{M_1} \| \mathrm{M_2}}(\mathrm{G}) \geqslant \mathrm{p_G}$ for certain values of $\mathrm{p_G}$
- i.e. gives lower bounds for probability $Pr^{\min}_{\mathrm{M_1} \| \mathrm{M_2}}(\mathrm{G})$
- 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 probabilistic automata
- 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

Results

Case study		Non-compositional		Compositional	
		States	Time (s)	LP size	Time (s)
rand. cons. [R K]	3, 2	1,418,545	18,971	40,542	**29.6**
	3, 20	39,827,233*	time-out	40,542	**125.3**
	4, 2	150,487,585	78,955	141,168	**376.1**
	4, 20	2,028,200,209*	mem-out	141,168	**471.9**
zero-conf [K]	2	91,041	39.0	6,910	**9.3**
	4	313,541	103.9	20,927	**21.9**
	6	811,290	275.2	40,258	**54.8**
	8	1,892,952	592.2	66,436	**107.6**
zero-conf timed [K T]	2, 10	665,567	**46.3**	62,188	89.0
	2, 14	106,177	**63.1**	101,313	170.8
	4, 10	976,247	**88.2**	74,484	170.8
	4, 14	2,288,771	**128.3**	166,203	430.6

* These models can be constructed, but not model checked, in PRISM.

Conclusions

Compositional probabilistic verification based on:

- probabilistic automata, with arbitrary parallel composition
- assumptions/guarantees: probabilistic safety properties
- reduction to multi-objective model checking
- multiple proof rules; adapted to quantitative approach

Encouraging experimental results

- verified safety/performance on several large case studies
- cases where infeasible using non-compositional verification

Current/future work

- automatic generation of assumptions: learning
- integration of other quantitative properties, e.g. rewards
- further (e.g. symmetric) proof rules