



Automated Compositional Verification for Probabilistic Systems

Dave Parker

Oxford University Computing Laboratory

Technical University of Denmark (DTU), August 2010

Joint work with: Marta Kwiatkowska, Gethin Norman, Hongyang Qu, Lu Feng

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 [TACAS'10]**
 - semantics, model checking, proof rules
 - quantitative approaches
 - implementation & results
- **Automated generation of assumptions [QEST'10]**
 - L*-based learning loop
 - implementation & results
- **Conclusions, current & future work**

Compositional verification

- Goal: scalability through modular verification
 - e.g. decide if $M_1 || M_2 \models G$
 - 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:

$$\langle \text{true} \rangle M_1 \langle A \rangle$$
$$\langle A \rangle M_2 \langle G \rangle$$

$$\langle \text{true} \rangle M_1 || M_2 \langle G \rangle$$

[Pasareanu/Giannakopoulou/et al.]

AG rules for probabilistic systems

- How to formulate AG rules for probabilistic automata?

$$\langle \text{true} \rangle M_1 \langle A \rangle$$
$$\langle A \rangle M_2 \langle G \rangle$$

$$\langle \text{true} \rangle M_1 \parallel M_2 \langle G \rangle$$

- 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 $M_1 \parallel M_2$?
- 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 probabilistic automata?

$$\langle \text{true} \rangle M_1 \langle A \rangle$$
$$\langle A \rangle M_2 \langle G \rangle$$

$$\langle \text{true} \rangle M_1 \parallel M_2 \langle G \rangle$$

- 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

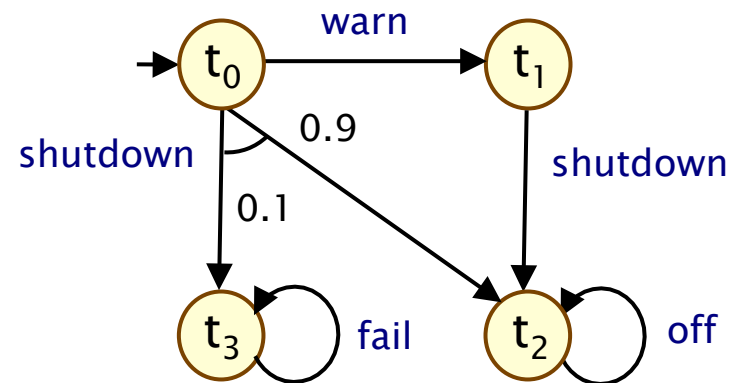
- Compositional verification
 - assume-guarantee reasoning
- **Probabilistic automata**
 - probabilistic safety properties
 - multi-objective model checking
- Probabilistic assume guarantee [TACAS'10]
 - semantics, model checking, proof rules
 - quantitative approaches
 - implementation & results
- Automated generation of assumptions [QEST'10]
 - L*-based learning loop
 - implementation & results
- Conclusions, current & future work

Probabilistic automata (PAs)

- Model nondeterministic as well as probabilistic behaviour
 - very similar to Markov decision processes (MDPs)

- A probabilistic automaton is a tuple $M = (S, s_{init}, \alpha_M, \delta_M)$:

- S is the state space
- $s_{init} \in S$ is the initial state
- α_M is the action alphabet
- $\delta_M \subseteq S \times \alpha_M \times \text{Dist}(S)$ is the transition probability relation



- Parallel composition: $M_1 \parallel M_2$

- CSP style – synchronise over common actions
- (i.e. the intersection of their alphabets)

Property specifications for PAs

- To reason formally about PAs, we use **adversaries**
- An adversary σ resolves nondeterminism in a PA M
 - also called “scheduler”, “strategy”, “policy”, ...
 - makes a (possibly randomised) choice, based on history
 - induces probability measure \Pr_M^σ 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. $\diamond \text{err}$ – “an error eventually occurs”
 - e.g. $\square(\text{req} \rightarrow \diamond \text{ack})$ – “req is always followed by ack”
 - or automata over action labels (see later)
 - e.g. deterministic finite automata (DFAs)

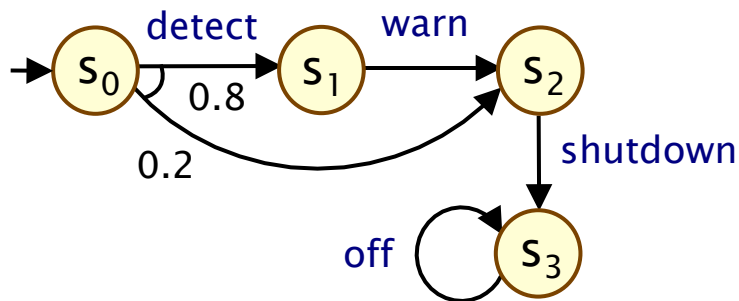
Model checking for PAs

- Property specification: quantify over all adversaries
 - e.g. $M \models P_{\geq p}[\phi] \Leftrightarrow \Pr_M^\sigma(\phi) \geq p$ for all adversaries $\sigma \in \text{Adv}_M$
 - corresponds to best-/worst-case behaviour analysis
 - or in a more quantitative fashion:
 - just compute e.g. $\Pr_M^{\min}(\phi) = \inf \{ \Pr_M^\sigma(\phi) \mid \sigma \in \text{Adv}_M \}$
- Model checking: efficient algorithms exist
 - for reachability, graph-based analysis + linear programming
 - in practice, for scalability, often approximate (value iteration)
 - for LTL, first construct an automaton-PA product
- And tool support is available
 - e.g. PRISM, Liquor, RAPTURE
 - (but scalability is always an issue)

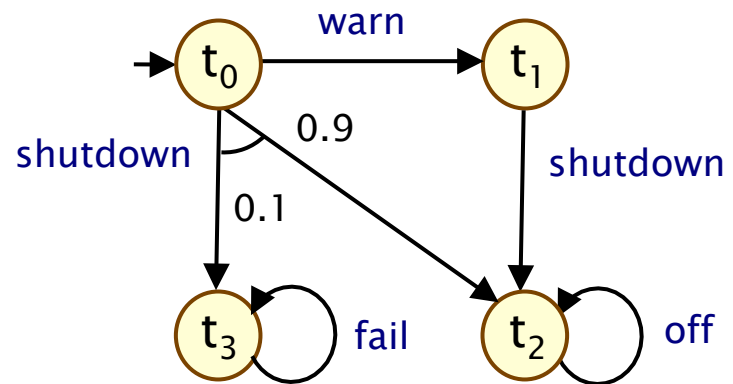
Running example

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

PA M_1 (“controller”)

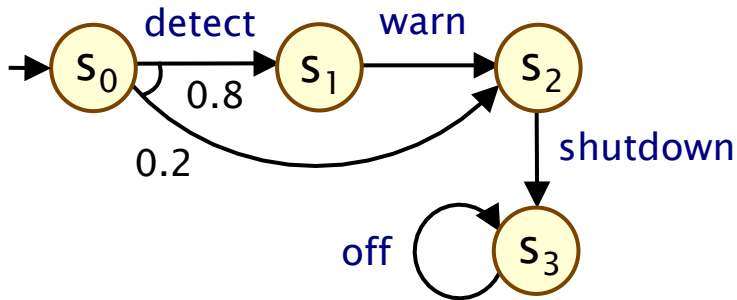


PA M_2 (“device”)

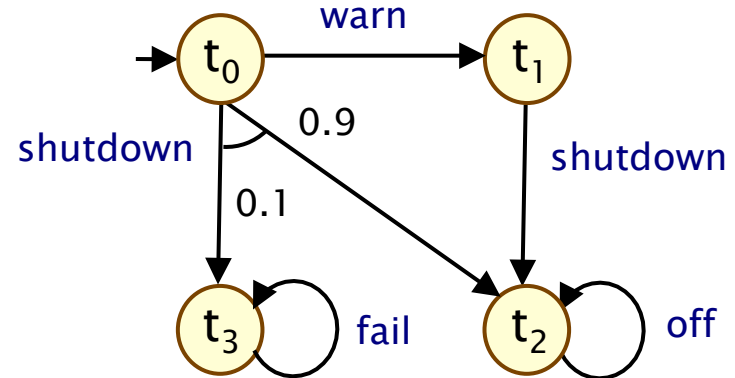


Running example

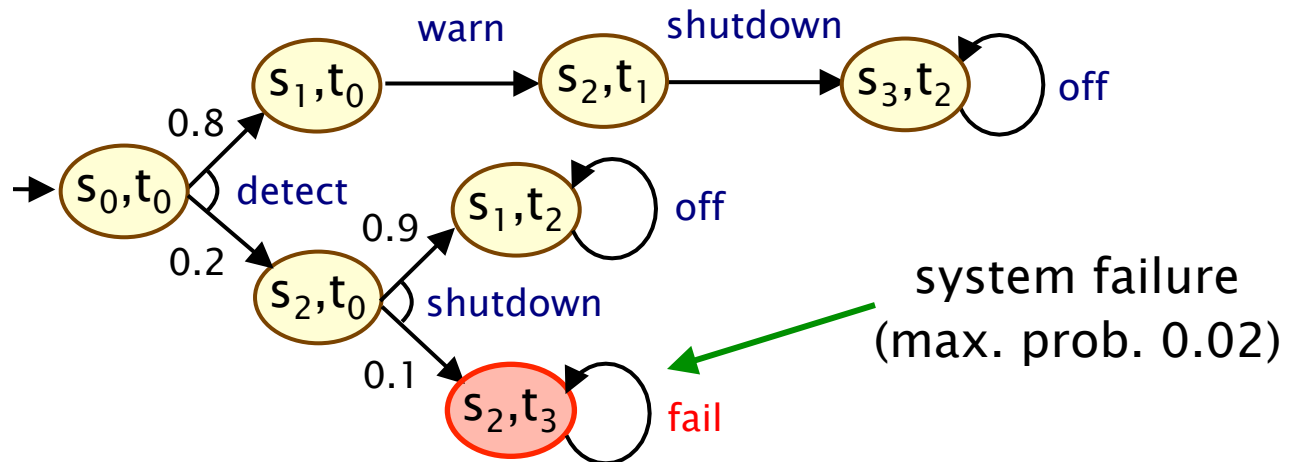
PA M_1 ("controller")



PA M_2 ("device")

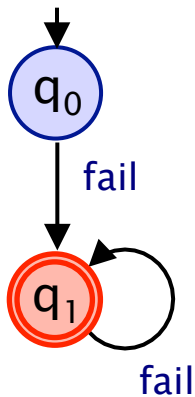


Parallel composition: $M_1 \parallel M_2$

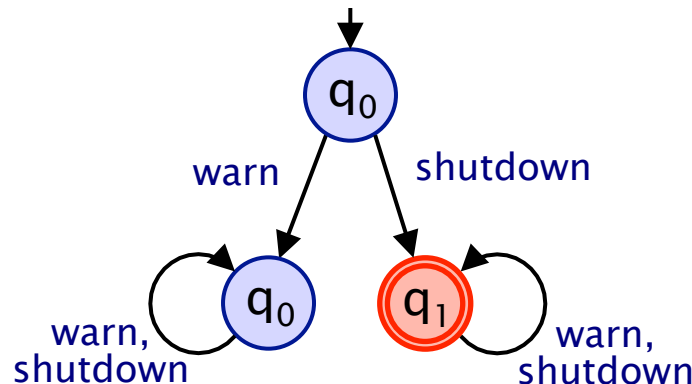


Safety properties

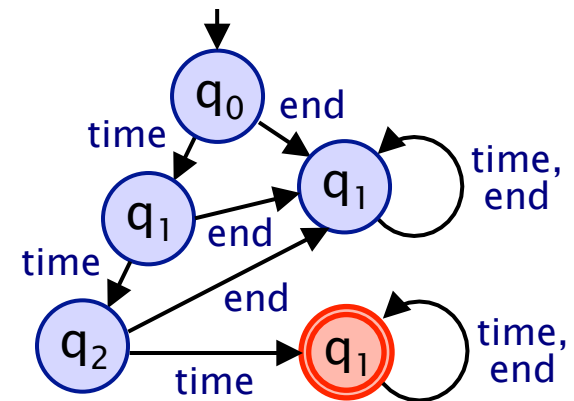
- Safety property: language of infinite words (over actions)
 - characterised by a set of “bad prefixes” (or “finite violations”)
 - i.e. finite words of which any extension violates the property
- Regular safety property
 - bad prefixes are represented by a regular language
 - property A stored as deterministic finite automaton (DFA) A_{err}



“a fail action never occurs”



“warn occurs before shutdown”



“at most 2 time steps pass before termination”

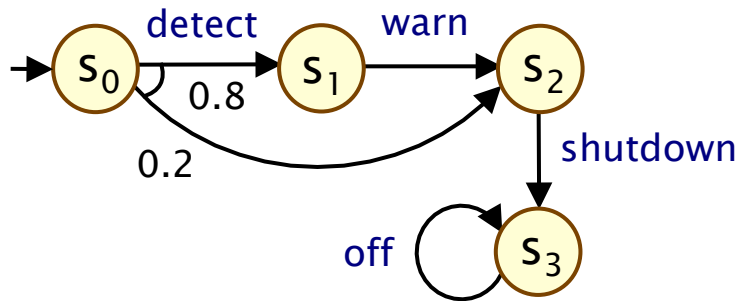
Probabilistic safety properties

- 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 ”
 - $M \models P_{\geq p}[A] \Leftrightarrow \Pr_M^\sigma(A) \geq p$ for all $\sigma \in \text{Adv}_M \Leftrightarrow \Pr_M^{\min}(A) \geq p$
- 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 err_A denotes “accept” states for DFA A
 - i.e. construct (synchronous) PA-DFA product $M \otimes A_{\text{err}}$
 - then compute reachability probabilities on product PA

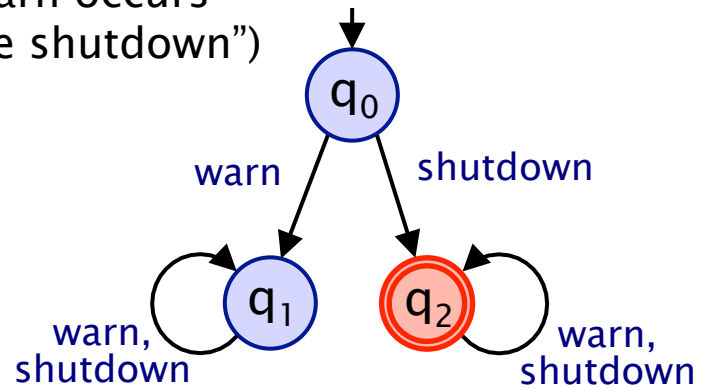
Running example

- Does probabilistic safety property $P_{\geq 0.8} [A]$ hold in M_1 ?

PA M_1 (“controller”)



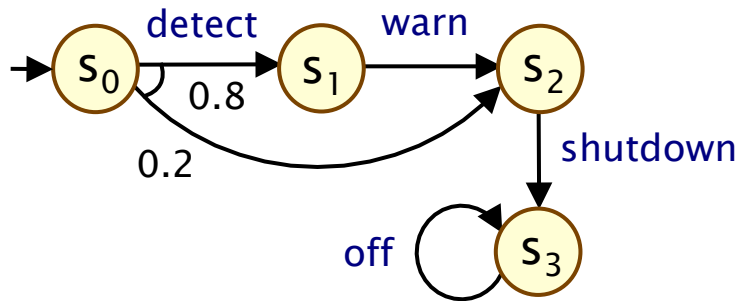
A (“warn occurs before shutdown”)



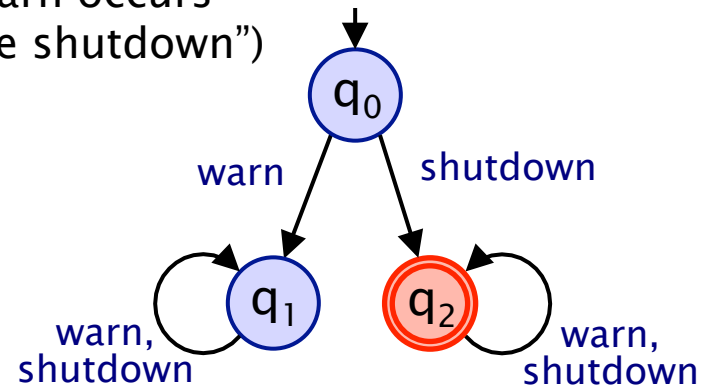
Running example

- Does probabilistic safety property $P_{\geq 0.8} [A]$ hold in M_1 ?

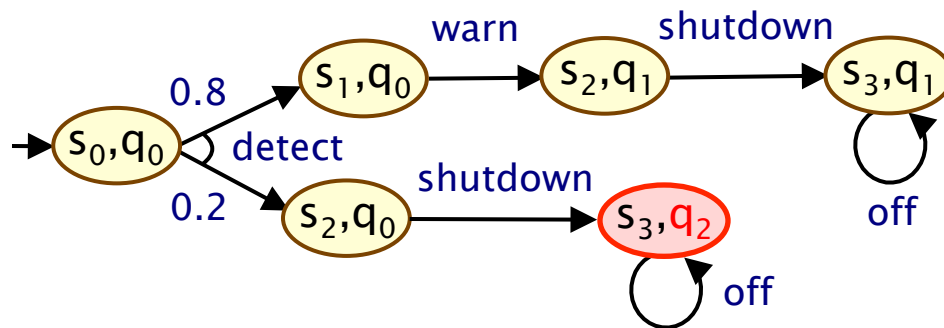
PA M_1 (“controller”)



A (“warn occurs before shutdown”)



Product PA $M_1 \otimes A_{err}$



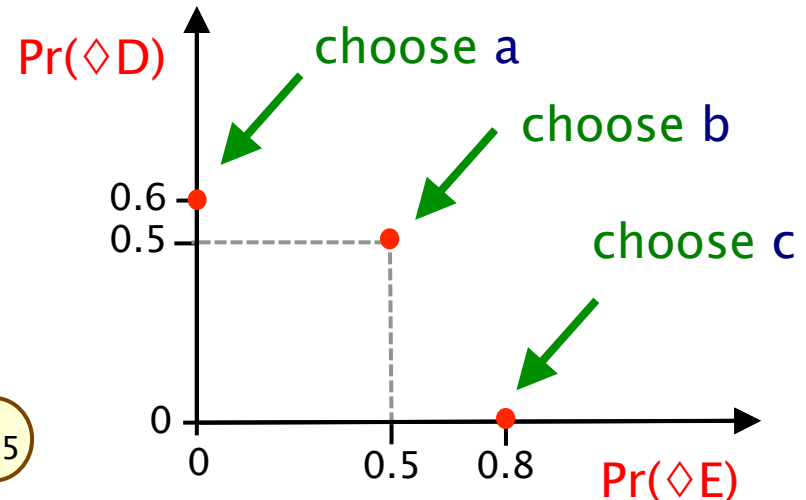
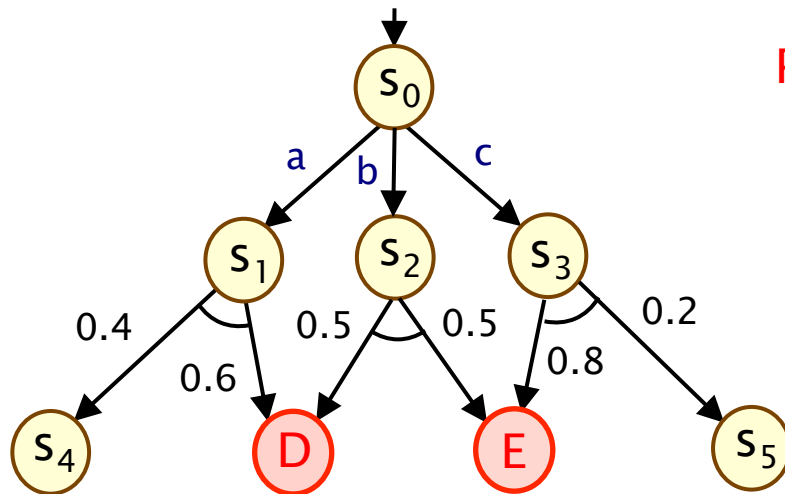
$$\begin{aligned}
 & \Pr_{M_1}^{\min}(A) \\
 &= 1 - \Pr_{M_1 \otimes A_{err}}^{\max}(\diamond \text{err}_A) \\
 &= 1 - 0.2 \\
 &= 0.8 \\
 &\rightarrow M_1 \models P_{\geq 0.8} [A]
 \end{aligned}$$

Multi-objective PA model checking

- Consider multiple (linear-time) objectives for a PA M
 - LTL formulae ϕ_1, \dots, ϕ_k and probability bounds $\sim_1 p_1, \dots, \sim_k p_k$
 - question: does there exist an adversary $\sigma \in \text{Adv}_M$ such that:
$$\Pr_M^\sigma(\phi_1) \sim_1 p_1 \wedge \dots \wedge \Pr_M^\sigma(\phi_k) \sim_k p_k$$
- Motivating example:
 - $\Pr_M^\sigma(\Box(\text{queue_size} < 10)) > 0.99 \wedge \Pr_M^\sigma(\Diamond \text{flat_battery}) < 0.01$
- Multi-objective PA model checking
 - [Etessami/Kwiatkowska/Vardi/Yannakakis, TACAS'07]
 - construct product of automata for M, ϕ_1, \dots, ϕ_k
 - then solve linear programming (LP) problem
 - the resulting adversary σ can be obtained from LP solution
 - note: σ may be randomised (unlike the single objective case)

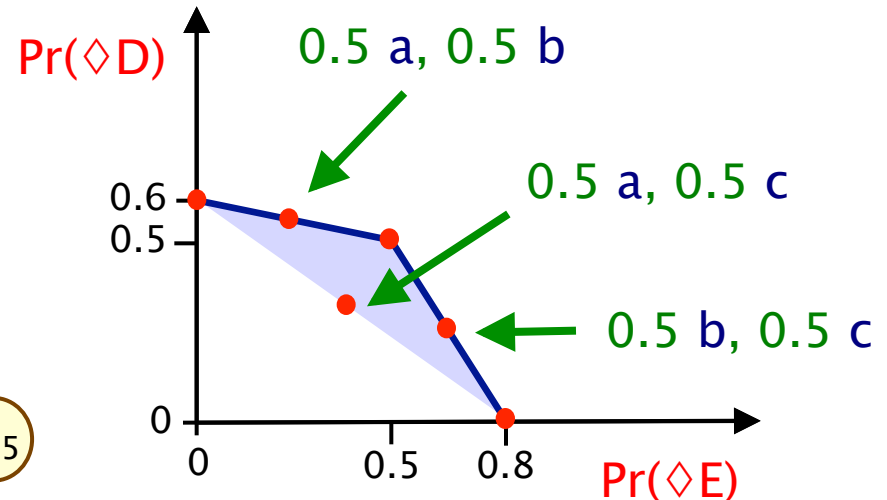
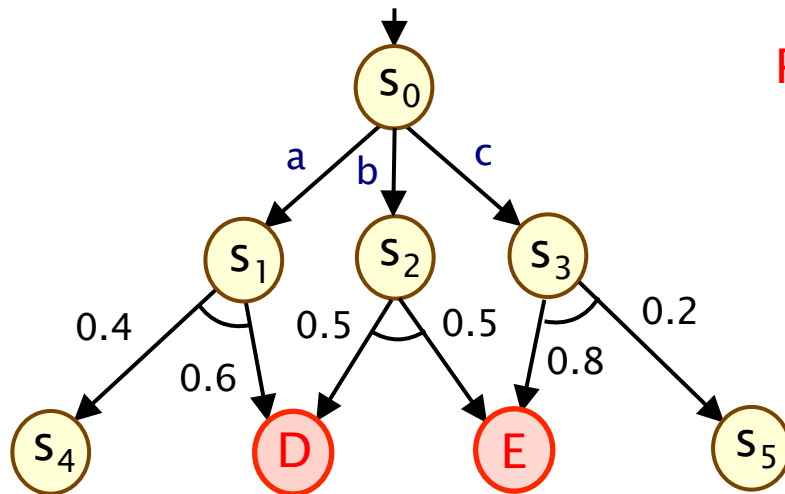
Multi-objective PA model checking

- Consider the objectives $\diamond D$ and $\diamond E$ in the PA 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 PA model checking

- Consider the objectives $\diamond D$ and $\diamond E$ in the PA 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

- Compositional verification
 - assume-guarantee reasoning
- Probabilistic automata
 - probabilistic safety properties
 - multi-objective model checking
- **Probabilistic assume guarantee [TACAS'10]**
 - semantics, model checking, proof rules
 - quantitative approaches
 - implementation & results
- Automated generation of assumptions [QEST'10]
 - L*-based learning loop
 - implementation & results
- Conclusions, current & future work

Probabilistic assume guarantee

- Assume-guarantee triples $\langle A \rangle_{\geq p_A} M \langle G \rangle_{\geq p_G}$ where:
 - M is a probabilistic automaton
 - $P_{\geq p_A}[A]$ 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:
$$\langle A \rangle_{\geq p_A} M \langle G \rangle_{\geq p_G} \Leftrightarrow \forall \sigma \in \text{Adv}_{M[\alpha_A]} (\text{Pr}_{M[\alpha_A]}^\sigma (A) \geq p_A \rightarrow \text{Pr}_{M[\alpha_A]}^\sigma (G) \geq p_G)$$
 - where $M[\alpha_A]$ is M with its alphabet extended to include α_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 PA $M' = M[\alpha_A] \otimes A_{err} \otimes G_{err}$
- More precisely:
 - check no adv. of M satisfying $\Pr_{M^\sigma}(A) \geq p_A$ but not $\Pr_{M^\sigma}(G) \geq p_G$

$$\langle A \rangle_{\geq p_A} M \langle G \rangle_{\geq p_G} \\ \Leftrightarrow$$

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

- 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)

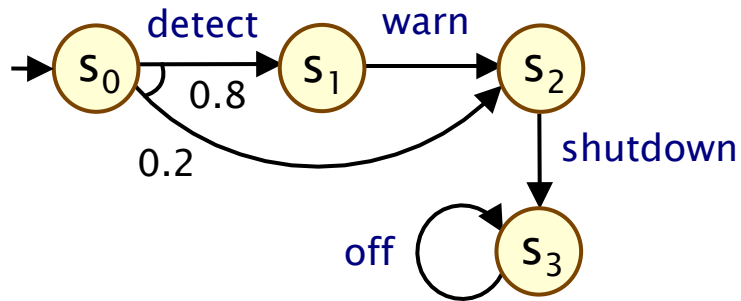
$$\frac{\langle \text{true} \rangle M_1 \langle A \rangle_{\geq p_A} \quad \langle A \rangle_{\geq p_A} M_2 \langle G \rangle_{\geq p_G}}{\langle \text{true} \rangle M_1 \parallel M_2 \langle G \rangle_{\geq p_G}} \quad (\text{ASYM})$$

- So, verifying $M_1 \parallel M_2 \models P_{\geq p_G} [G]$ 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 $|M_1|$

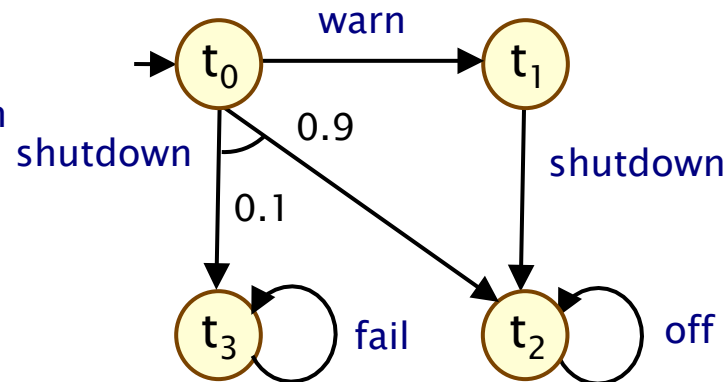
Running example

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

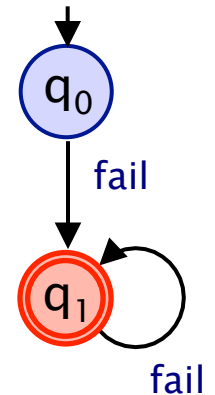
PA M_1 (“controller”)



PA M_2 (“device”)



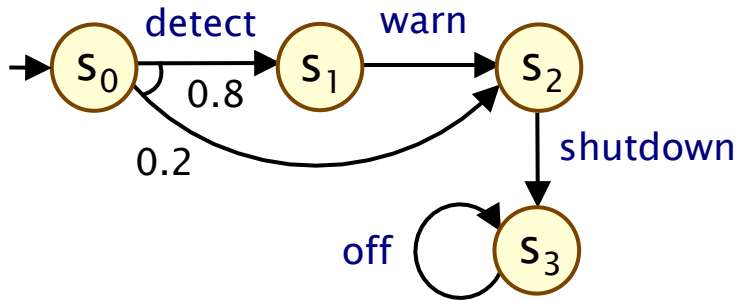
G (“a fail action never occurs”)



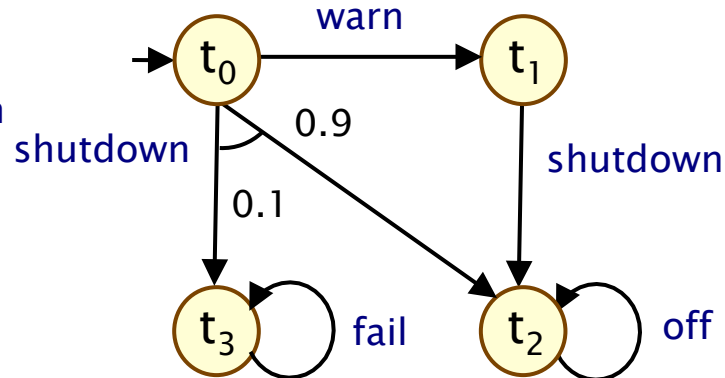
Running example

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

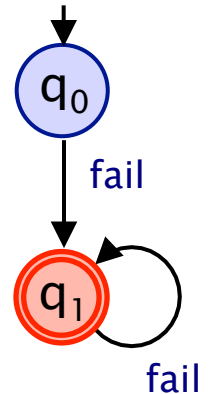
PA M_1 (“controller”)



PA M_2 (“device”)



G (“a fail action never occurs”)



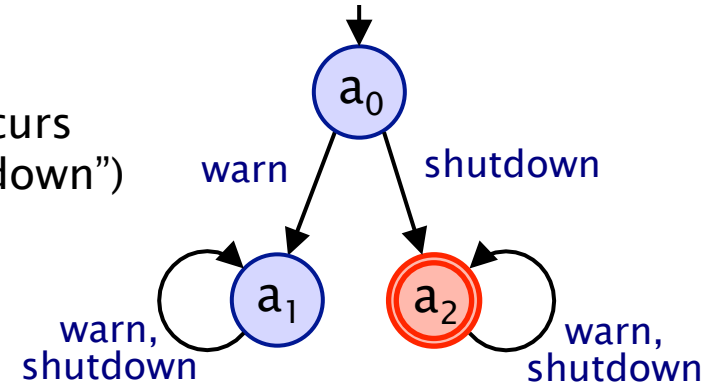
- Use AG with assumption $\langle A \rangle_{\geq 0.8}$ about M_1

$$\langle \text{true} \rangle M_1 \langle A \rangle_{\geq 0.8}$$

$$\langle A \rangle_{\geq 0.8} M_2 \langle G \rangle_{\geq 0.98}$$

$$\langle \text{true} \rangle M_1 || M_2 \langle G \rangle_{\geq 0.98}$$

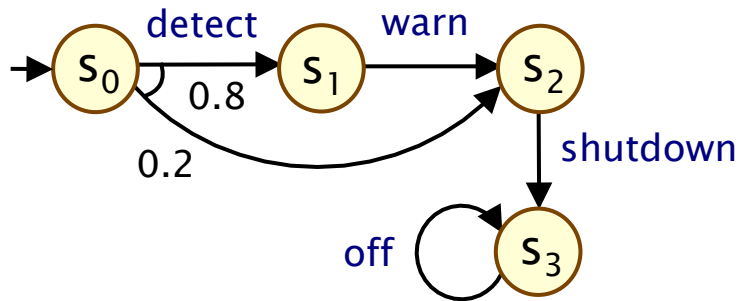
A (“warn occurs before shutdown”)



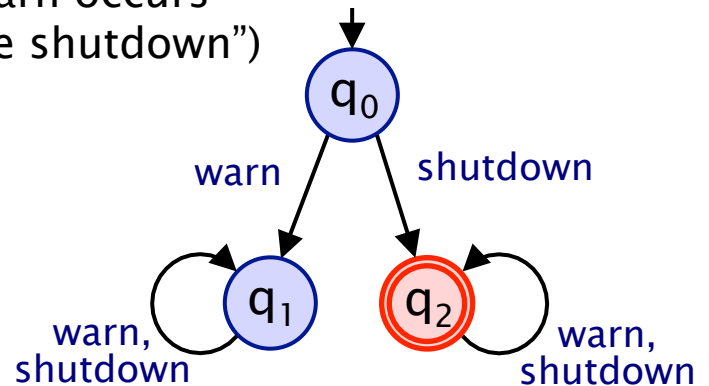
Running example

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

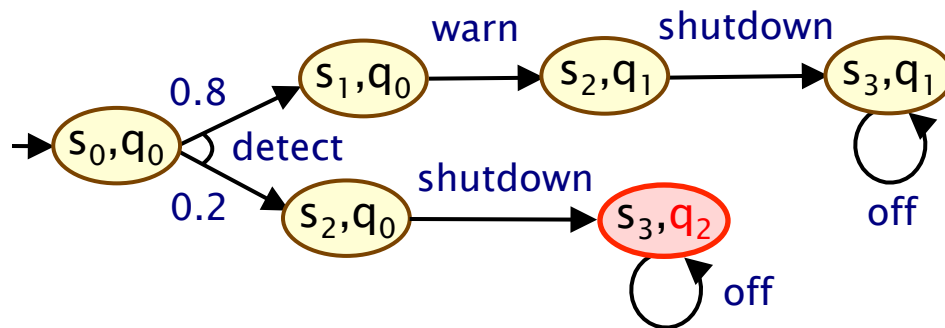
PA M_1 (“controller”)



A (“warn occurs before shutdown”)



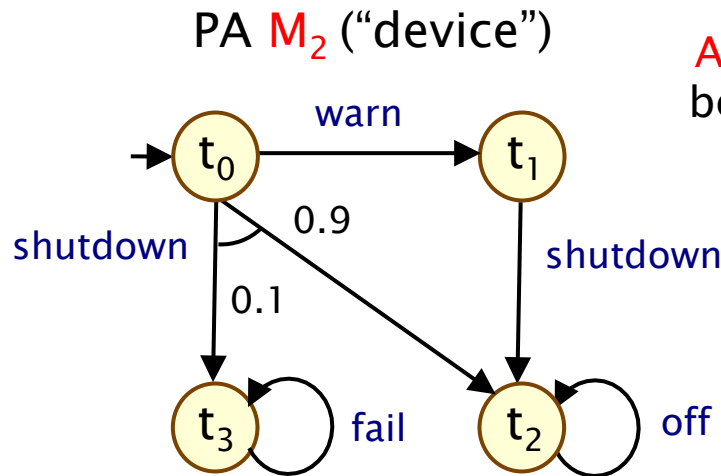
Product PA $M_1 \otimes A_{err}$



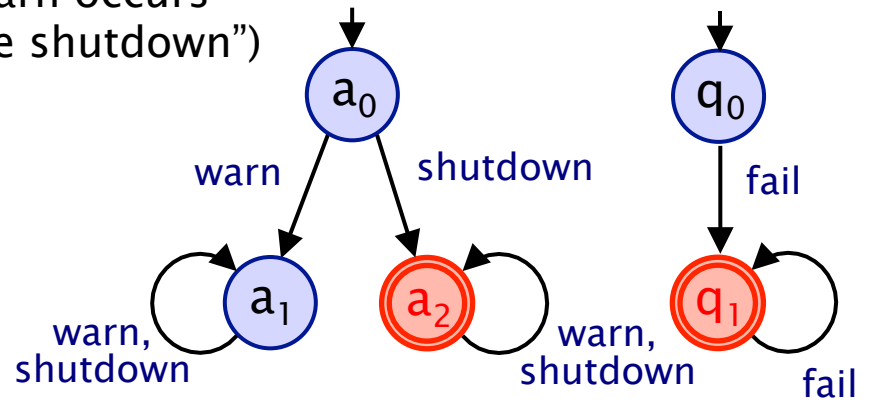
$$\begin{aligned}
 & \Pr_{M_1}^{\min}(A) \\
 &= 1 - \Pr_{M_1 \otimes A_{err}}^{\max}(\diamond \text{err}_A) \\
 &= 1 - 0.2 \\
 &= 0.8 \\
 &\rightarrow M_1 \models P_{\geq 0.8} [A]
 \end{aligned}$$

Running example

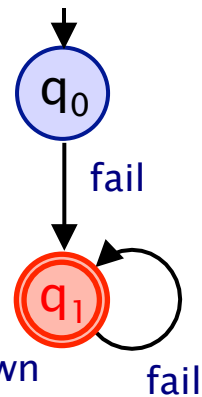
- Premise 2: Does $\langle A \rangle_{\geq 0.8} M_2 \langle G \rangle_{\geq 0.98}$ hold?



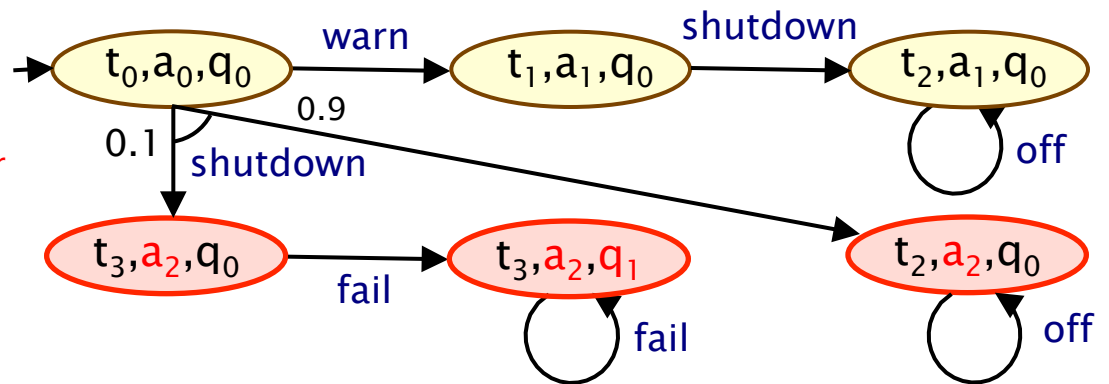
A ("warn occurs before shutdown")



G ("a fail action never occurs")



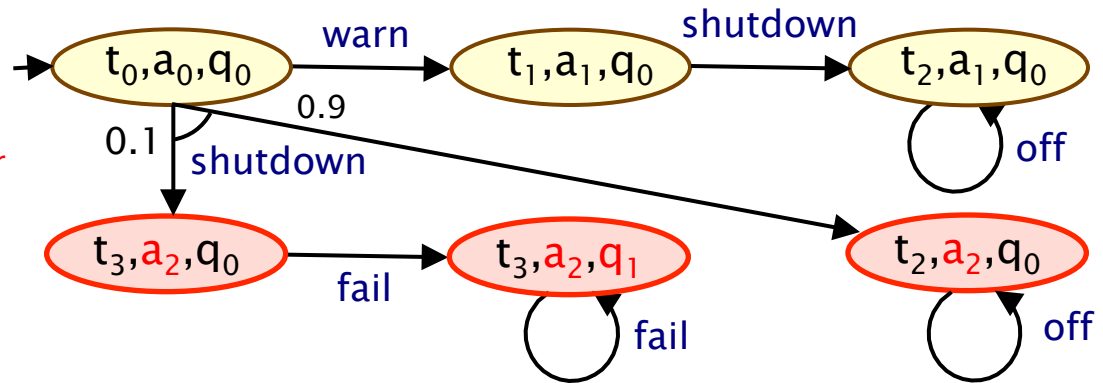
Product PA
 $M' = M_2[\alpha_A] \otimes A_{err} \otimes G_{err}$



Running example

- Premise 2: Does $\langle A \rangle_{\geq 0.8} M_2 \langle G \rangle_{\geq 0.98}$ hold?

Product PA
 $M' = M_2[\alpha_A] \otimes A_{err} \otimes G_{err}$



- \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 an adversary of M' with $\Pr_{M,\sigma'}(\diamond a_2) \leq 0.2$ and $\Pr_{M,\sigma'}(\diamond q_1) > 0.02$?
- To satisfy $\Pr_{M,\sigma'}(\diamond a_2) \leq 0.2$, adversary σ' must choose **shutdown** in initial state with probability ≤ 0.2 , which means $\Pr_{M,\sigma'}(\diamond q_1) \leq 0.02$
- 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:

$$\frac{\langle \text{true} \rangle M_1 \langle A_1, \dots, A_k \rangle_{\geq p_1, \dots, p_k} \quad \langle A_1, \dots, A_k \rangle_{\geq p_1, \dots, p_k} M_2 \langle G \rangle_{\geq p_G}}{\langle \text{true} \rangle M_1 \parallel M_2 \langle G \rangle_{\geq p_G}}$$

- Circular rule:

$$\frac{\langle \text{true} \rangle M_2 \langle A_1 \rangle_{\geq p_2} \quad \langle A_2 \rangle_{\geq p_2} M_1 \langle A_1 \rangle_{\geq p_1} \quad \langle A_1 \rangle_{\geq p_1} M_2 \langle G \rangle_{\geq p_G}}{\langle \text{true} \rangle M_1 \parallel M_2 \langle G \rangle_{\geq p_G}}$$

- Multiple components (chain)

$$\frac{\langle \text{true} \rangle M_1 \langle A_1 \rangle_{\geq p_1} \quad \langle A_1 \rangle_{\geq p_1} M_2 \langle A_2 \rangle_{\geq p_2} \quad \dots \quad \langle A_n \rangle_{\geq p_n} M_n \langle G \rangle_{\geq p_G}}{\langle \text{true} \rangle M_1 \parallel \dots \parallel M_n \langle G \rangle_{\geq p_G}}$$

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 $\Pr_{M_1 || M_2}^{\min}(G) \geq p_G$
for certain values of p_G
- i.e. gives lower bound for $\Pr_{M_1 || M_2}^{\min}(G)$
- 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

$$\frac{\langle \text{true} \rangle M_1 \langle A \rangle_{\geq p_A} \quad \langle A \rangle_{\geq p_A} M_2 \langle G \rangle_{\geq p_G}}{\langle \text{true} \rangle M_1 || M_2 \langle G \rangle_{\geq p_G}}$$

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

Experimental results

Case study [parameters]		Non-compositional		Compositional	
		States	Time (s)	LP size	Time (s)
Randomised consensus (3 processes) [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
ZeroConf [K]	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
ZeroConf time-bounded [K, T]	2, 10	65,567	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

Experimental results

Case study [parameters]		Non-compositional		Compositional	
		States	Time (s)	LP size	Time (s)
Randomised consensus (3 processes) [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
ZeroConf [K]	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
ZeroConf time-bounded [K, T]	2, 10	65,567	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

- Faster than conventional model checking in a number of cases

Experimental results

Case study [parameters]		Non-compositional		Compositional	
		States	Time (s)	LP size	Time (s)
Randomised consensus (3 processes) [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
ZeroConf [K]	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
ZeroConf time-bounded [K, T]	2, 10	65,567	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

- Verified instances where conventional model checking is infeasible

Experimental results

Case study [parameters]		Non-compositional		Compositional	
		States	Time (s)	LP size	Time (s)
Randomised consensus (3 processes) [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
ZeroConf [K]	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
ZeroConf time-bounded [K, T]	2, 10	65,567	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

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

Overview

- Compositional verification
 - assume-guarantee reasoning
- Probabilistic automata
 - probabilistic safety properties
 - multi-objective model checking
- Probabilistic assume guarantee [TACAS'10]
 - semantics, model checking, proof rules
 - quantitative approaches
 - implementation & results
- **Automated generation of assumptions** [QEST'10]
 - L*-based learning loop
 - implementation & results
- Conclusions, current & future work

Generating assumptions

- We can model check $M_1 || M_2$ compositionally

- but this relies on the existence of a suitable assumption $\langle A \rangle_{\geq p_A}$

$$\langle \text{true} \rangle M_1 \langle A \rangle_{\geq p_A}$$

$$\langle A \rangle_{\geq p_A} M_2 \langle G \rangle_{\geq p_G}$$

$$\langle \text{true} \rangle M_1 || M_2 \langle G \rangle_{\geq p_G}$$

- 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
 - 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*

- “Membership” query (for trace t):

- does $t \parallel M_2 \models P_{\geq p_G} [G]$ hold?

$$\langle \text{true} \rangle M_1 \langle A \rangle_{\geq p_A}$$

$$\langle A \rangle_{\geq p_A} M_2 \langle G \rangle_{\geq p_G}$$

$$\langle \text{true} \rangle M_1 \parallel M_2 \langle G \rangle_{\geq p_G}$$

- “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

- Counterexamples to “refine A ”

- are probabilistic counterexamples [Han/Katoen]

Experimental results (learning)

Case study [parameters]		Component sizes		Compositional	
		$ M_2 \otimes G_{err} $	$ M_1 $	$ A $	Time (s)
Client-server (N failures) [N]	3	229	16	4	6.6
	4	1,121	25	5	13.1
	5	5,397	36	6	87.5
Randomised consensus [N,R,K]	2, 3, 20	391	3,217	5	24.2
	2, 4, 2	573	113,569	10	108.4
	3, 3, 2	8,843	4,065	14	681.7
	3, 3, 20	8,843	38,193	14	863.8
Sensor network [N]	1	42	72	2	3.5
	2	42	1,184	2	3.7
	3	42	10,662	2	4.6

Experimental results (learning)

Case study [parameters]		Component sizes		Compositional	
		$ M_2 \otimes G_{err} $	$ M_1 $	$ A $	Time (s)
Client-server (N failures) [N]	3	229	16	4	6.6
	4	1,121	25	5	13.1
	5	5,397	36	6	87.5
Randomised consensus [N,R,K]	2, 3, 20	391	3,217	5	24.2
	2, 4, 2	573	113,569	10	108.4
	3, 3, 2	8,843	4,065	14	681.7
	3, 3, 20	8,843	38,193	14	863.8
Sensor network [N]	1	42	72	2	3.5
	2	42	1,184	2	3.7
	3	42	10,662	2	4.6

- Successfully learnt (small) assumptions in all cases

Experimental results (learning)

Case study [parameters]		Component sizes		Compositional	
		$ M_2 \otimes G_{\text{err}} $	$ M_1 $	$ A $	Time (s)
Client-server (N failures) [N]	3	229	16	4	6.6
	4	1,121	25	5	13.1
	5	5,397	36	6	87.5
Randomised consensus [N,R,K]	2, 3, 20	391	3,217	5	24.2
	2, 4, 2	573	113,569	10	108.4
	3, 3, 2	8,843	4,065	14	681.7
	3, 3, 20	8,843	38,193	14	863.8
Sensor network [N]	1	42	72	2	3.5
	2	42	1,184	2	3.7
	3	42	10,662	2	4.6

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

Conclusions

- Compositional probabilistic verification based on:
 - probabilistic automata, 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
- Encouraging experimental results
 - verified safety/performance on several large case studies
 - cases where infeasible using non-compositional verification
- Current/future work
 - prove (lack of) completeness
 - other types of assumptions/properties, e.g. liveness, rewards
 - further (e.g. symmetric/circular) proof rules
 - continuous-time models