UNIVERSITY OF OXFORD

Automated Verification Techniques for Probabilistic Systems

Vojtěch Forejt
Marta Kwiatkowska
Gethin Norman
Dave Parker

SFM-11:CONNECT Summer School, Bertinoro, June 2011

EU-FP7: CONNECT LSCITS/PSS VERIWARE

Part 4

Compositional probabilistic verification

Overview

- Lectures 1 and 2:
  - 1 – Introduction
  - 2 – Discrete-time Markov chains
  - 3 – Markov decision processes
  - 4 – Compositional probabilistic verification
- PRISM lab session (4.30pm)
  - PC lab downstairs – or install PRISM on your own laptop
- Course materials available here:
  - http://www.prismmodelchecker.org/courses/sfm11connect/
  - lecture slides, reference list, tutorial chapter, lab session

Overview (Part 4)

- 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 $M_1 \,||\, M_2 \models G$
  - by analysing M_1 and M_2 separately
- Assume-guarantee (AG) reasoning
  - use assumption A about the context of a component M_2
  - $\langle A \rangle\, M_2\, \langle G \rangle$ – “whenever M_2 is part of a system satisfying A, then the system must also guarantee G”
  - example of asymmetric (non-circular) A/G rule:

$$\frac{M_1 \models A \qquad \langle A \rangle\, M_2\, \langle G \rangle}{M_1 \,||\, M_2 \models G}$$

[Pasareanu/Giannakopoulou/et al.]

AG rules for probabilistic systems

- How to formulate AG rules for MDPs?

$$\frac{M_1 \models A \qquad \langle A\rangle\, M_2\, \langle G\rangle}{M_1 \,||\, M_2 \models G}$$

- Key questions:
  - 1. What form do assumptions A take?
    - needs to be compositional
    - needs to be efficient to check
    - needs to allow compact assumptions
  - 2. How do we generate suitable assumptions?
    - preferably in a fully automated fashion
  - 3. Can we get "quantitative" results?
    - i.e. numerical values, rather than "yes"/"no"

AG rules for probabilistic systems

- How to formulate AG rules for MDPs?

$$\frac{M_1 \models A \qquad \langle A \rangle\, M_2\, \langle G \rangle}{M_1 \,||\, M_2 \models G}$$

- Key questions:
  - 1. What form do assumptions A take?
    - needs to be compositional
    - needs to be efficient to check
    - needs to allow compact assumptions

  ▷ various compositional relations exist
    - e.g. strong/weak (probabilistic) (bi)simulation
    - but these are either too fine (difficult to get small assumptions) or expensive to check

  ▷ here, we use: probabilistic safety properties [TACAS'10]
    - less expressive, but compact and efficient
    - (see also generalisation to liveness/rewards [TACAS'11])

AG rules for probabilistic systems

- How to formulate AG rules for MDPs?

$$\frac{M_1 \vDash A \qquad \langle A \rangle\, M_2\, \langle G \rangle}{M_1 \,||\, M_2 \vDash G}$$

- Key questions:
  - 2. How do we generate suitable assumptions?
    - preferably in a fully automated fashion

  ▷ algorithmic learning (based on L* algorithm)
  adapt techniques for (non-probabilistic) assumptions

  - 3. Can we get "quantitative" results?
    - i.e. numerical values, rather than "yes"/"no"

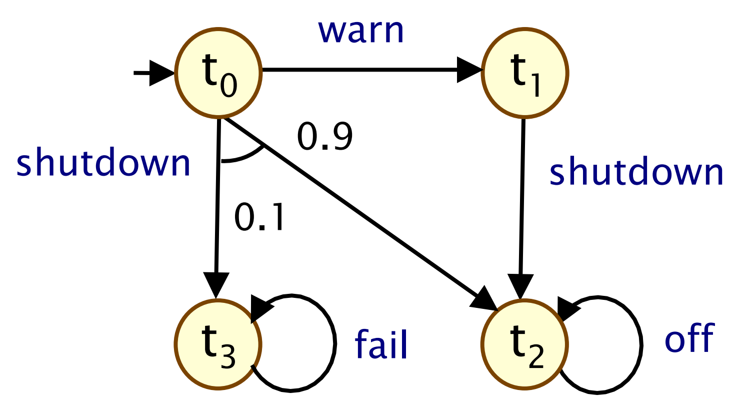
  ▷ yes: generate lower/upper bounds on probabilities

Overview (Part 4)

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



- Notes:
  - α_M, δ_M have subscripts to avoid confusion with other automata
  - transitions can also be labelled with a "silent" τ action
  - we write $s \xrightarrow{a} \mu$ as shorthand for $(s,a,\mu) \in \delta_M$
  - MDPs, here, are identical to probabilistic automata [Segala]

Recap: Model checking for MDPs

- An adversary σ resolves the nondeterminism in an MDP M
  - make a (possibly randomised) choice, based on history
  - induces probability measure Pr_M^σ over (infinite) paths $Path_M^\sigma$
  - can compute probability of some measurable property ϕ
    - e.g. $F\ err \equiv \Diamond err$ – "an error eventually occurs"
    - or automata over action labels (see later)

- Property specifications: quantify over all adversaries
  - e.g. PCTL: $M \vDash P_{\geq p}[\phi] \;\Leftrightarrow\; Pr_M^\sigma(\phi) \geq p$ for all adv.s $\sigma \in Adv_M$
  - corresponds to best-/worst-case behaviour analysis
  - requires computation of $Pr_M^{min}(\phi)$ or $Pr_M^{max}(\phi)$
  - or in a more quantitative fashion:
  - just ask e.g. $P_{min=?}(\phi)$ or $P_{max=?}(\phi)$
  - also extends to (min/max) expected costs & rewards

Parallel composition for MDPs

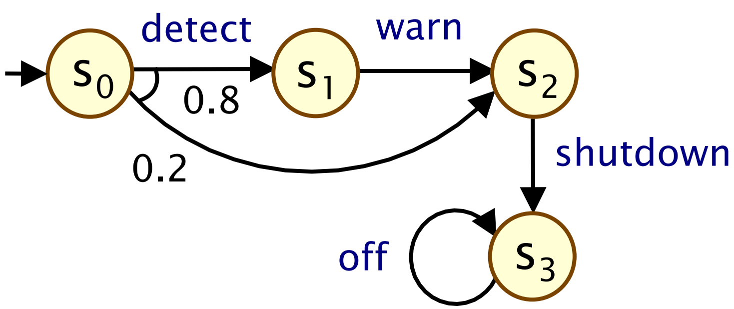
- The parallel composition of M_1 and M_2 is denoted $M_1 \mid\mid M_2$
  - CSP style: synchronise over all common (non-τ) actions
  - when synchronising, transition probabilities are multiplied

- Formally, if $M_i = (S_i, s_{init,i}, \alpha_{M_i}, \delta_{M_i}, L_i)$ for $i=1,2$, then:
- $M_1 \mid\mid M_2 = (S_1 \times S_2, (s_{init,1}, s_{init,2}), \alpha_{M_1} \cup \alpha_{M_2}, \delta_{M_1 \mid\mid M_2}, L_{12})$ where:
  - $L_{12}(s_1,s_2) = L_1(s_1) \cup L_2(s_2)$
  - $\delta_{M_1 \mid\mid M_2}$ 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 $a \in (\alpha_{M_1} \backslash \alpha_{M_2}) \cup \{\tau\}$ (asynchronous)
    - $s_2 \xrightarrow{a} \mu_2$, $\mu_1 = \eta_{s_1}$ and $a \in (\alpha_{M_2} \backslash \alpha_{M_1}) \cup \{\tau\}$ (asynchronous)
  - where $\mu_1 \times \mu_2$ denotes the product of distributions μ_1, μ_2
  - and $\eta_s \in Dist(S)$ is the Dirac (point) distribution on $s \in S$

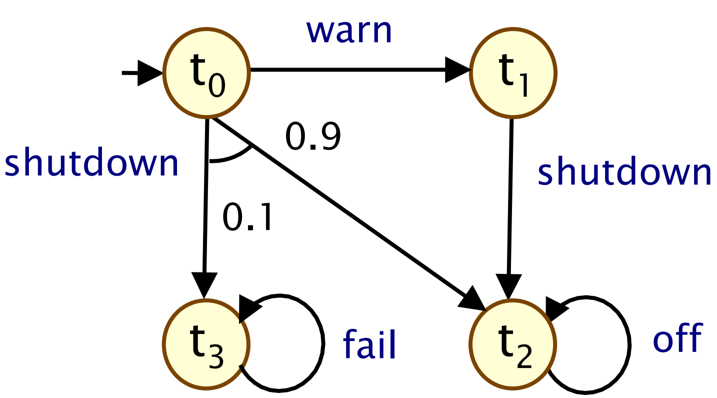
Running example

- 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")

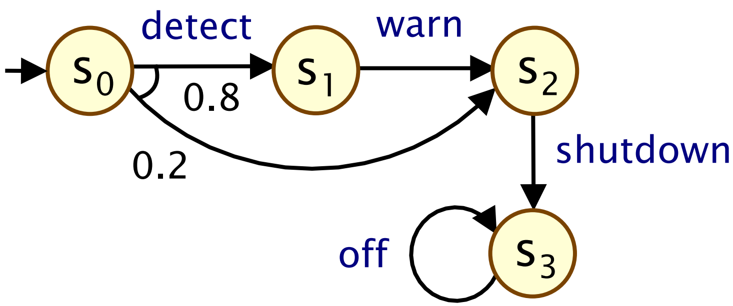


MDP M_2 ("device")

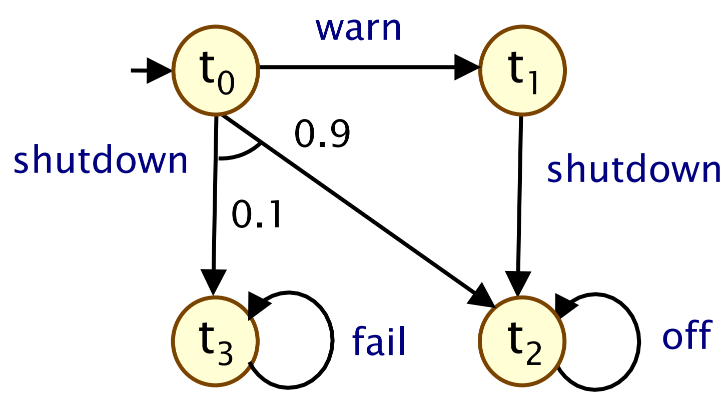


Running example

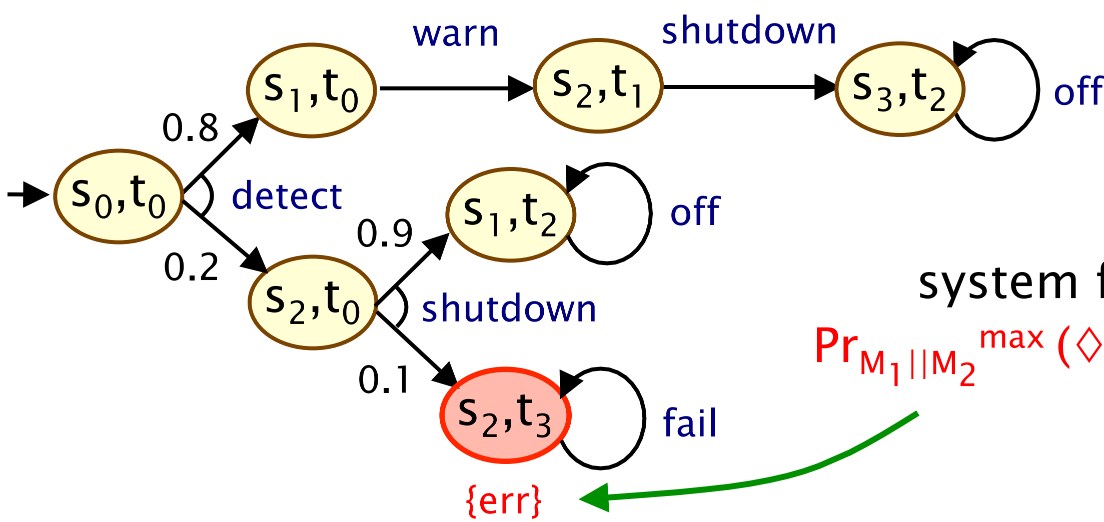
MDP M_1 ("controller")



MDP M_2 ("device")



Parallel composition: $M_1 \mid\mid M_2$

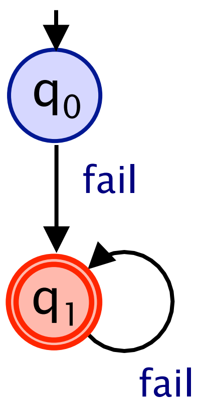


system failure:

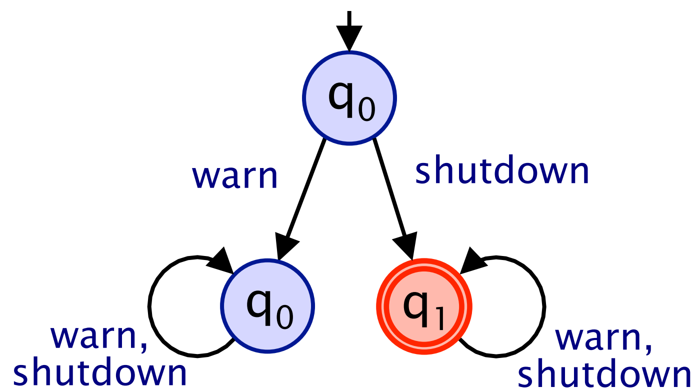
$Pr_{M_1 || M_2}^{\max}(\Diamond err) = 0.02$

Safety properties

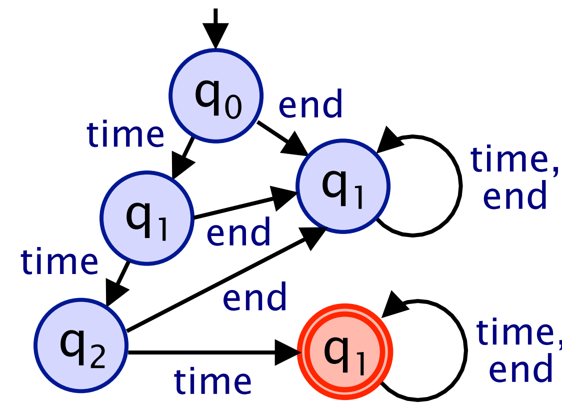
- Safety property: language of infinite words (over actions)
  - characterised by a set of "bad prefixes" (or "finite violations")
  - i.e. finite words of which any extension violates the property
- Regular safety property
  - bad prefixes are represented by a regular language
  - property A stored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



"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"
  - $M \vDash P_{\geq p}[A] \;\Leftrightarrow\; Pr_M^{\sigma}(A) \geq p$ for all $\sigma \in Adv_M \;\Leftrightarrow\; Pr_M^{min}(A) \geq p$

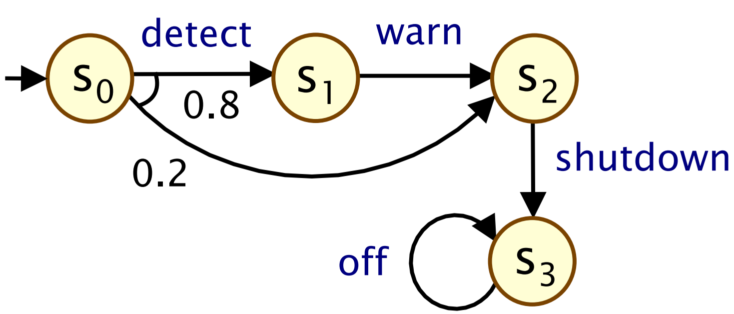
- Examples:
  - "*warn* occurs before *shutdown* with probability at least 0.8"
  - "the probability of a failure occurring is at most 0.02"
  - "probability of terminating within k time-steps is at least 0.75"

- Model checking: $Pr_M^{min}(A) = 1 - Pr_{M \otimes A_{err}}^{max}(\Diamond err_A)$
  - where err_A denotes "accept" states for DFA A
  - i.e. construct (synchronous) MDP-DFA product $M \otimes A_{err}$
  - then compute reachability probabilities on product MDP

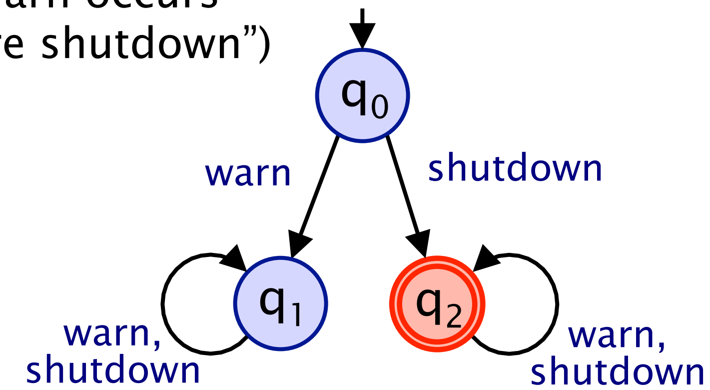
Running example

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

MDP M_1 (“controller”)



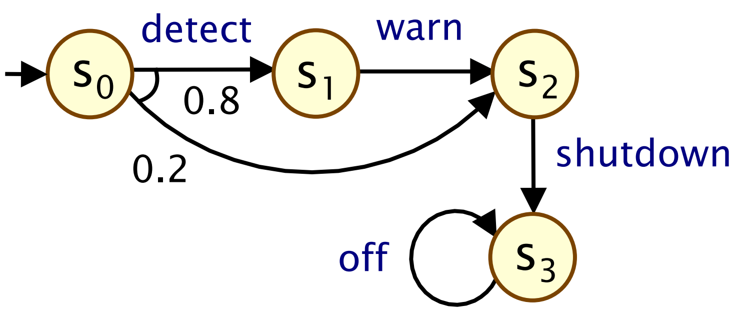
A (“warn occurs before shutdown”)



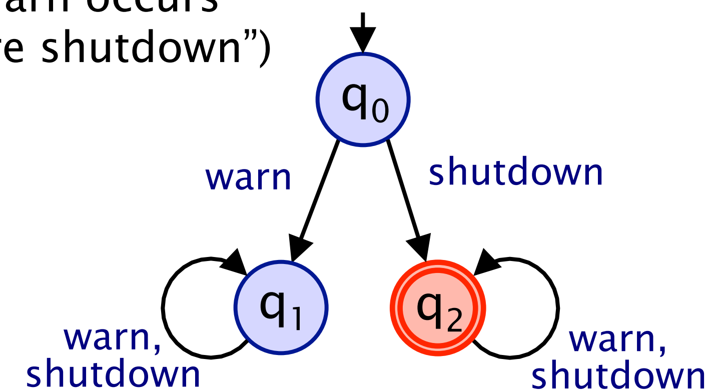
Running example

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

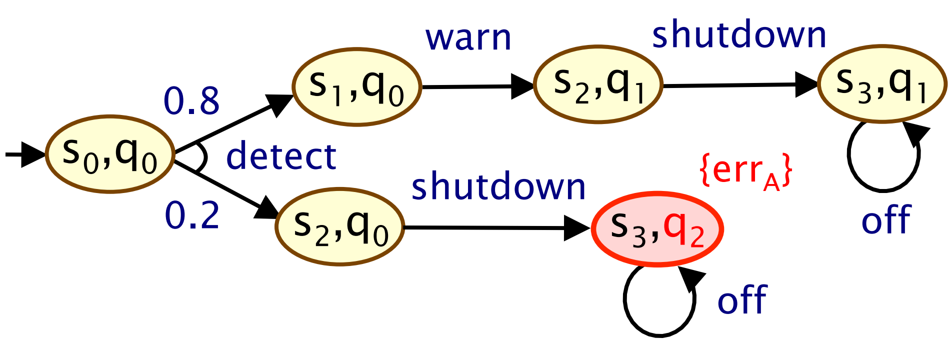
MDP M_1 ("controller")



A ("warn occurs before shutdown")



Product MDP $M_1 \otimes A_{err}$



$$Pr_{M_1}^{min}(A)$$
$$= 1 - Pr_{M_1 \otimes A_{err}}^{max}(\Diamond err_A)$$
$$= 1 - 0.2$$
$$= 0.8$$
$$\rightarrow M_1 \models P_{\geq 0.8} [A]$$

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 <u>exist</u> an adversary $\sigma \in Adv_M$ such that:

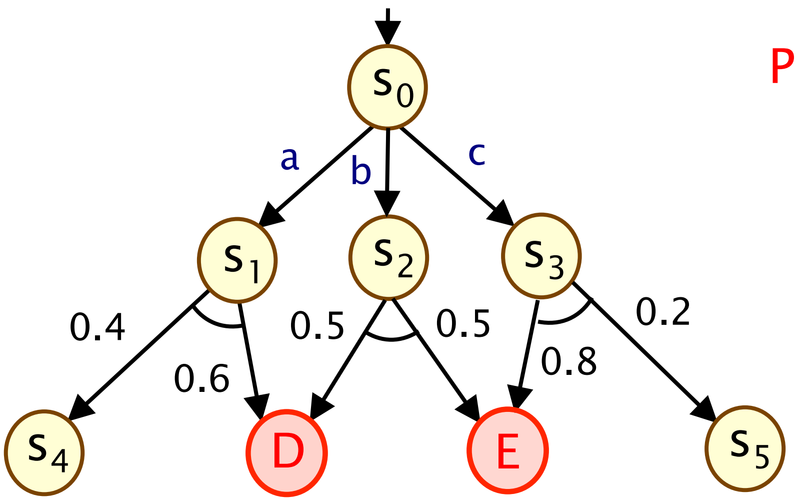
    $Pr_M^{\sigma}(\phi_1) \sim_1 p_1 \wedge \ldots \wedge Pr_M^{\sigma}(\phi_k) \sim_k p_k$

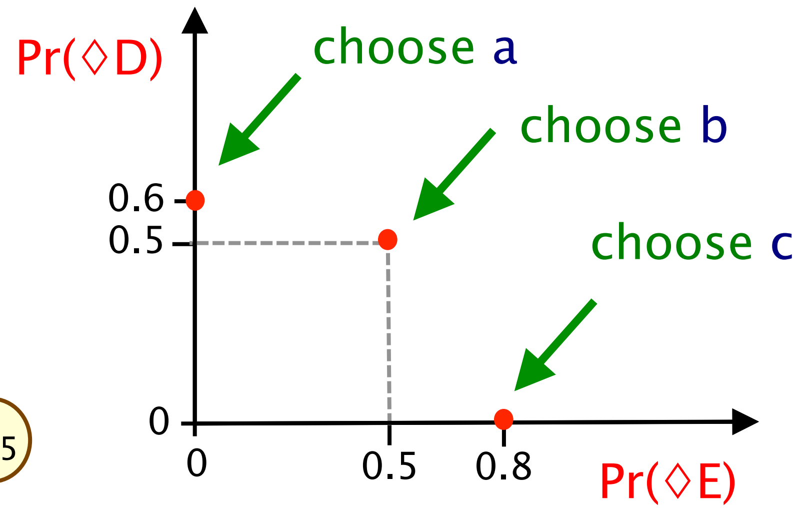
- Motivating example:
  - $Pr_M^{\sigma}(\Box(\text{queue\_size}<10)) > 0.99 \wedge Pr_M^{\sigma}(\Diamond \text{flat\_battery}) < 0.01$

- Multi-objective MDP model checking [EKVY07]
  - construct product of automata for M, $\Phi_1,\ldots,\Phi_k$
  - then solve linear programming (LP) problem
  - the resulting adversary σ can obtained from LP solution
  - note: σ may be randomised (unlike the single objective case)

Multi-objective MDP model checking

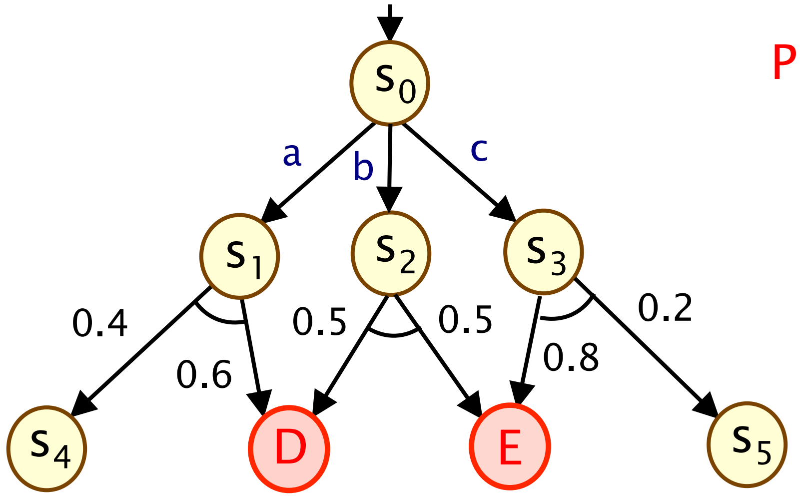
- Consider the two objectives ◊D and ◊E in the MDP below
  - i.e. the trade-off between the probabilities Pr(◊D) and Pr(◊E)
  - an adversary resolves the choice between a/b/c
  - increasing the probability of reaching one target decreases the probability of reaching the other

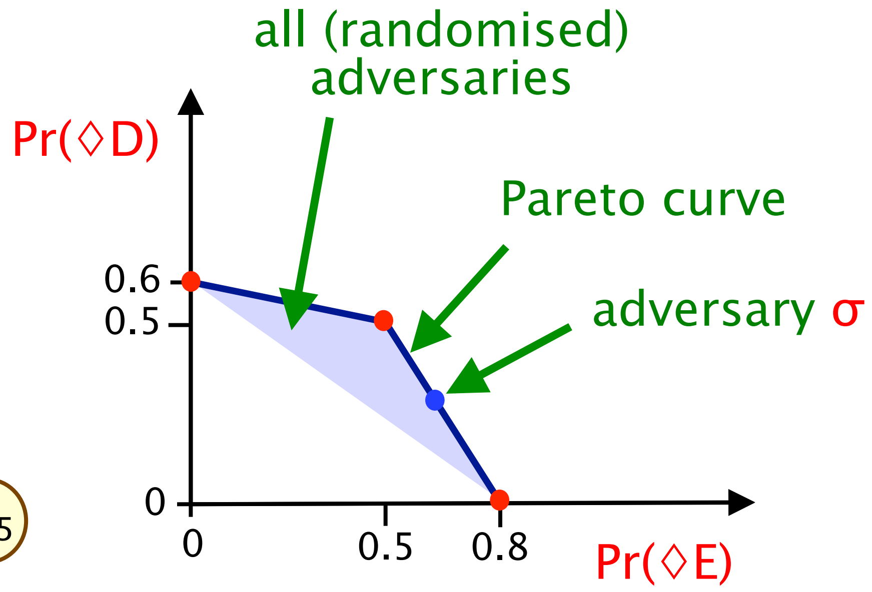




Multi-objective MDP model checking

- Need to consider all randomised adversaries
  - for example, is there an adversary σ such that:
  - $Pr(\Diamond D) > 0.2 \wedge Pr(\Diamond E) > 0.6$





Overview (Part 4)

- 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 an MDP
  - $P_{\geq p_A}[A]$ 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:
  - $\forall \sigma \in Adv_{M'}\ (\ Pr_{M'}^{\sigma}(A) \geq p_A \rightarrow Pr_{M'}^{\sigma}(G) \geq p_G\ )$
  - where M' is M with its alphabet extended to include α_A
  - reduces to multi-objective model checking on M'
  - look for adversary satisfying assumption but not guarantee
  - i.e. can check $\langle A \rangle_{\geq p_A}\ M\ \langle G \rangle_{\geq p_G}$ efficiently via LP problem

An assume-guarantee rule

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

$$\frac{M_1 \models P_{\geq p_A}[A] \qquad \langle A\rangle_{\geq p_A}\, M_2\, \langle G\rangle_{\geq p_G}}{M_1 \,||\, M_2 \models P_{\geq p_G}[G]} \quad \text{(ASYM)}$$

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

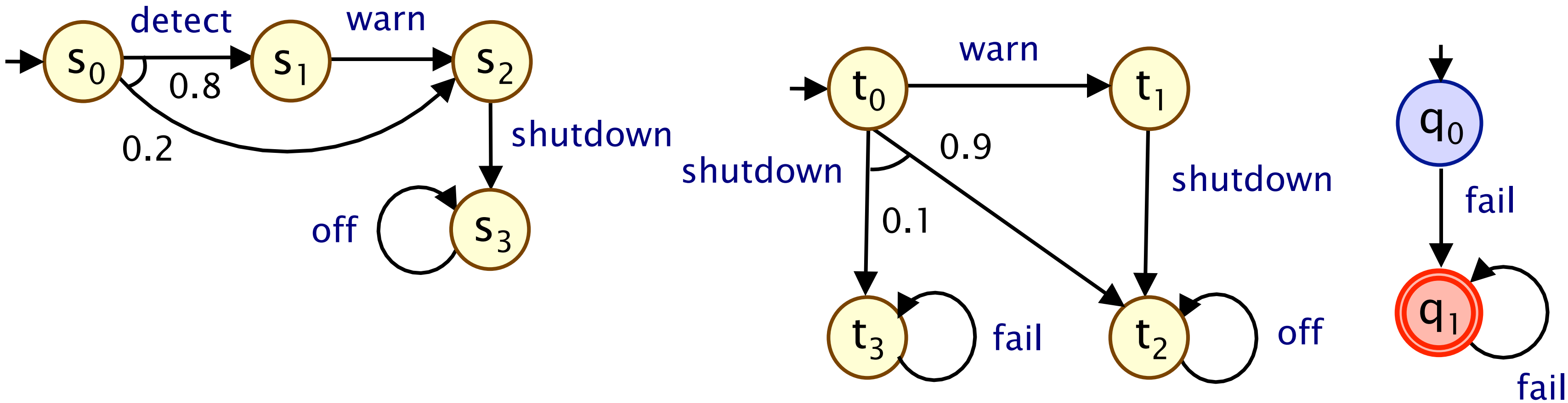
Running example

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

MDP M_1 (“controller”)

MDP M_2 (“device”)

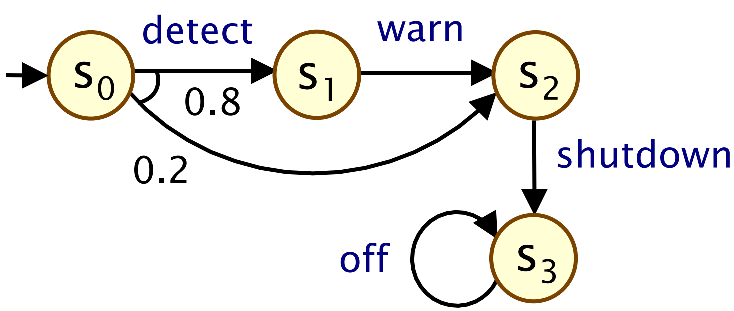
G (“a fail action never occurs”)



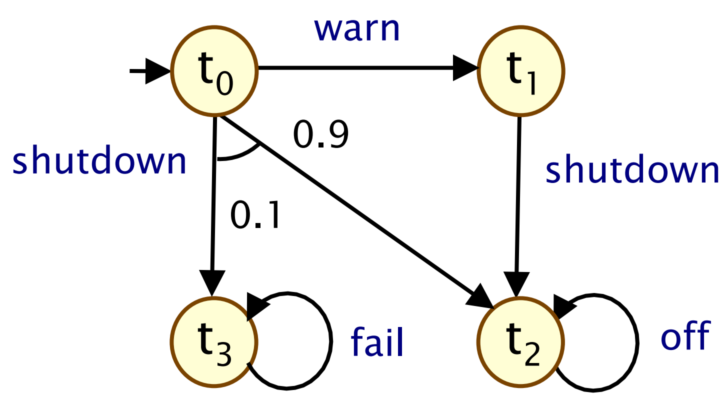
Running example

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

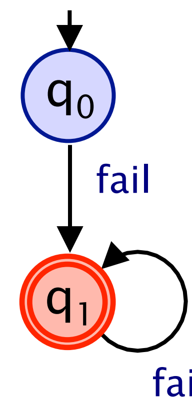
MDP M_1 ("controller")



MDP M_2 ("device")



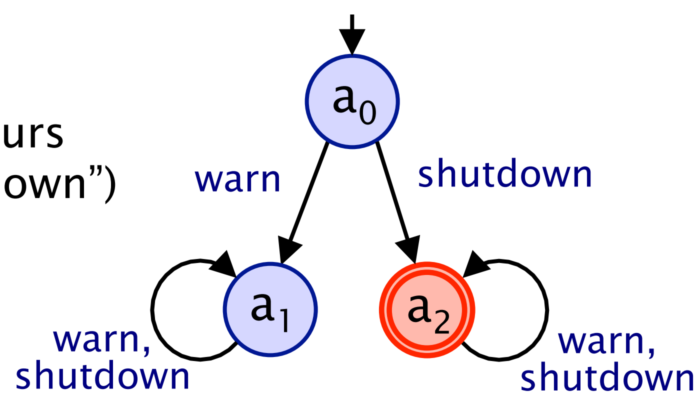
G ("a fail action never occurs")



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

$$\frac{\langle \text{true}\rangle\ M_1\ \langle A\rangle_{\geq 0.8} \qquad \langle A\rangle_{\geq 0.8}\ M_2\ \langle G\rangle_{\geq 0.98}}{\langle \text{true}\rangle\ M_1 \,||\, M_2\ \langle G\rangle_{\geq 0.98}}$$

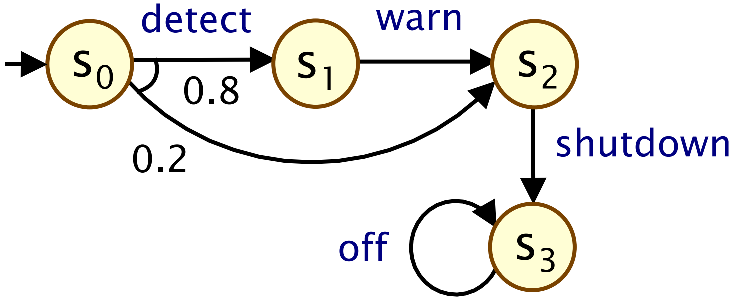
A ("warn occurs before shutdown")



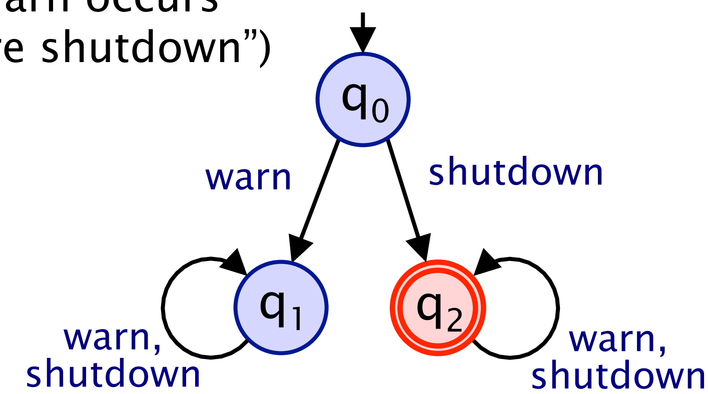
Running example

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

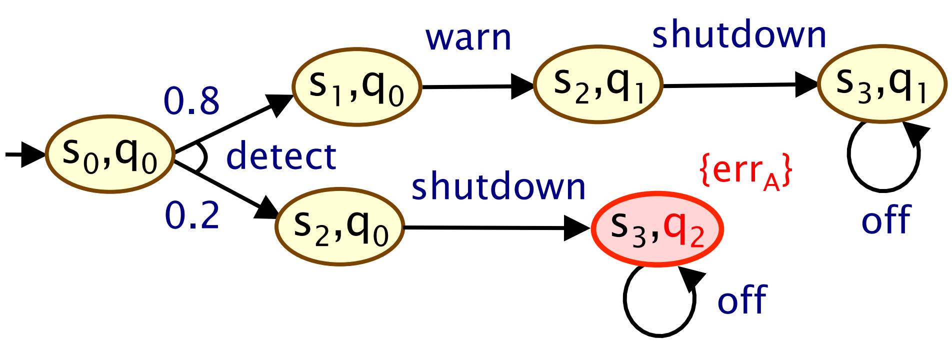
MDP M_1 ("controller")



A ("warn occurs before shutdown")



Product MDP $M_1 \otimes A_{err}$

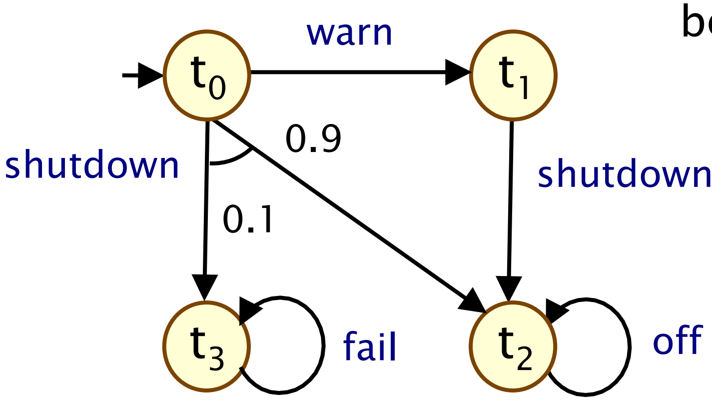


$$
\begin{aligned}
&Pr_{M_1}^{\min}(A) \\
&= 1 - Pr_{M_1 \otimes A_{err}}^{\max}(\Diamond err_A) \\
&= 1 - 0.2 \\
&= 0.8 \\
&\rightarrow M_1 \models P_{\geq 0.8}\,[A]
\end{aligned}
$$

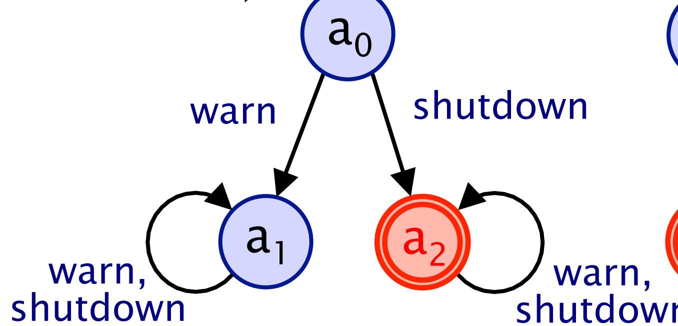
Running example

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

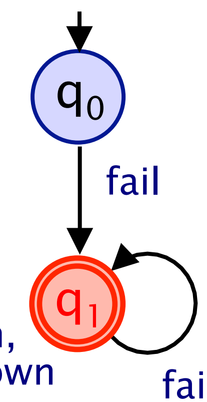
MDP M_2 ("device")



A ("warn occurs before shutdown")

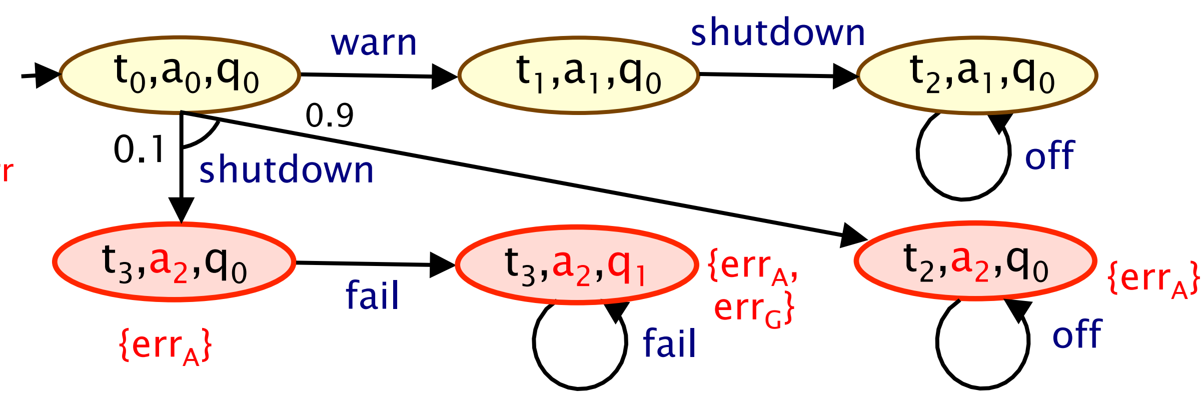


G ("a fail action never occurs")



Product MDP

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

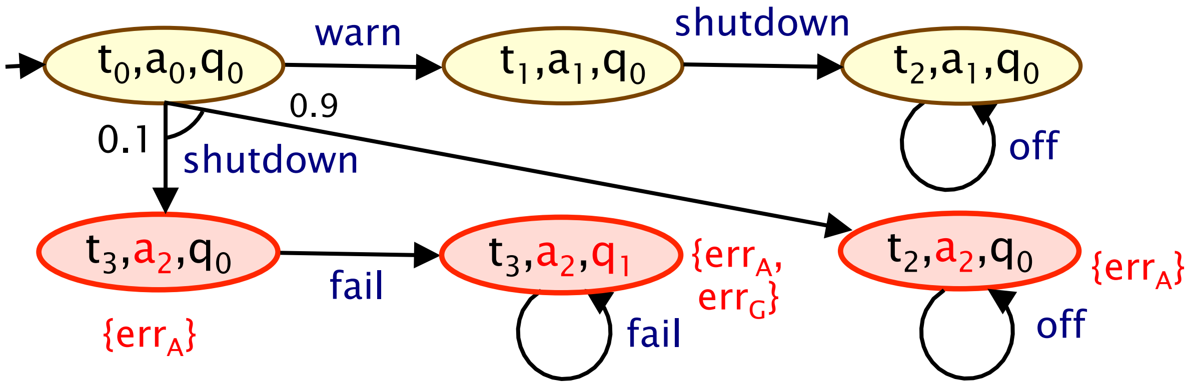


Running example

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

Product MDP

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



- $\exists$ 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 $Pr_{M'}{}^{\sigma'}(\lozenge err_A) \leq 0.2$ and $Pr_{M'}{}^{\sigma'}(\lozenge err_G) > 0.02$?
- To satisfy $Pr_{M'}{}^{\sigma'}(\lozenge err_A) \leq 0.2$, adversary σ' must choose shutdown in initial state with probability $\leq$ 0.2, which means $Pr_{M'}{}^{\sigma'}(\lozenge err_G) \leq 0.02$
- So, there is no such adversary and $\langle A \rangle_{\geq 0.8}\ M_2\ \langle G \rangle_{\geq 0.98}$ <u>does</u> hold

Other assume-guarantee rules

- Multiple assumptions:

$$\frac{M_1 \models P_{\geq p_1}[A_1] \wedge \ldots \wedge P_{\geq p_k}[A_k] \qquad \langle A_1,\ldots,A_k\rangle_{\geq p_{1,\ldots,}p_k}\ M_2\ \langle G\rangle_{\geq p_G}}{M_1 \mid\mid M_2 \models P_{\geq p_G}[G]} \quad \text{(ASYM-MULT)}$$

Multiple components (chain):

$$\frac{M_1 \models P_{\geq p_1}[A_1] \qquad \langle A_1\rangle_{\geq p_1}\ M_2\ \langle A_2\rangle_{\geq p_2} \qquad \ldots \qquad \langle A_n\rangle_{\geq p_n}\ M_n\ \langle G\rangle_{\geq p_G}}{M_1 \mid\mid \ldots \mid\mid M_n \models P_{\geq p_G}[G]} \quad \text{(ASYM-N)}$$

- Circular rule:

$$\frac{M_2 \models P_{\geq p_2}[A_2] \qquad \langle A_2\rangle_{\geq p_2}\ M_1\ \langle A_1\rangle_{\geq p_1} \qquad \langle A_1\rangle_{\geq p_1}\ M_2\ \langle G\rangle_{\geq p_G}}{M_1 \mid\mid M_2 \models P_{\geq p_G}[G]} \quad \text{(CIRC)}$$

Asynchronous components:

$$\frac{\langle A_1\rangle_{\geq p_1}\ M_1\ \langle G_1\rangle_{\geq q_1} \qquad \langle A_2\rangle_{\geq p_2}\ M_2\ \langle G_2\rangle_{\geq q_2}}{\langle A_1,A_2\rangle_{\geq p_1 p_2}\ M_1 \mid\mid M_2\ \langle G_1 \vee G_2\rangle_{\geq (q_1+q_2-q_1 q_2)}} \quad \text{(ASYNC)}$$

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

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

Implementation + Case studies

- Prototype extension of PRISM model checker
  - already supports LTL for 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

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 (Part 4)

- 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

- Can model check $M_1 || M_2$ compositionally
  - but this relies on the existence of a suitable assumption $P_{\geq p_A}[A]$

$$\frac{M_1 \models P_{\geq p_A}[A] \qquad \langle A \rangle_{\geq p_A} \, M_2 \, \langle G \rangle_{\geq p_G}}{M_1 \,||\, M_2 \models P_{\geq p_G}[G]}$$

- 1. Does such an assumption always exist?

- 2. When it does exist, can we generate it automatically?

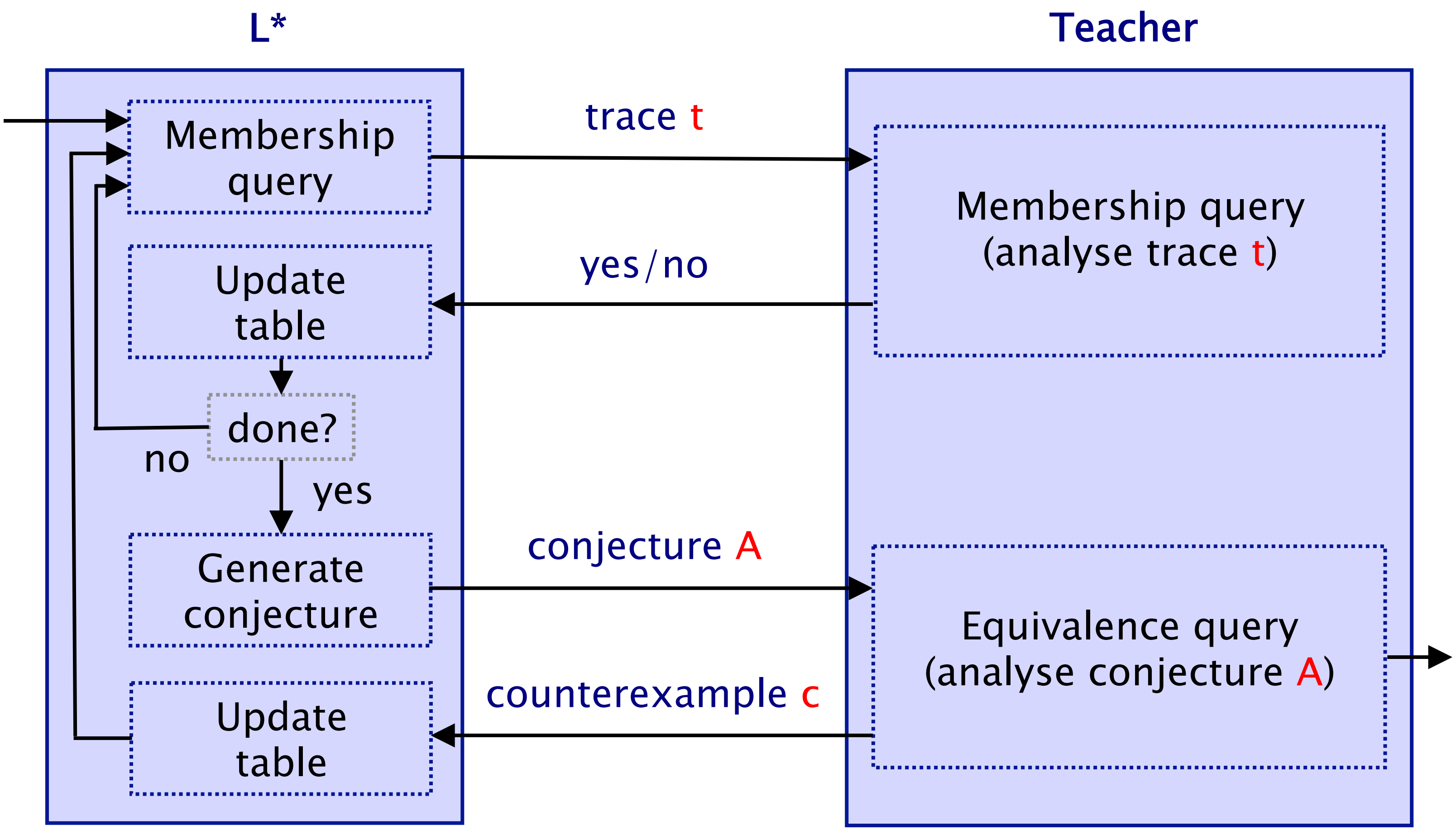
- Our approach: use algorithmic learning techniques
  - inspired by non-probabilistic AG work of [Pasareanu et al.]
  - uses L* algorithm to learn finite automata for assumptions
  - we use a modified version of L*
  - to learn probabilistic assumptions for rule (ASYM) [QEST'10]

The L* learning algorithm

- The L* algorithm [Angluin]
  - learns an unknown regular language L, as a (minimal) DFA

- Based on "active" learning
  - relies on existence of a "teacher" to guide the learning
  - answers two type of queries: "membership" and "equivalence"
  - membership: "is trace (word) t in the target language L?"
    - stores results of membership queries in observation table
    - based on these, generates conjectures A for the automata
  - equivalence: "does automata A accept the target language L"?
    - if not, teacher must return counterexample c
    - (c is a word in the symmetric difference of L and L(A))

The L* learning algorithm



L* for assume-guarantee

- Breakthrough in automated compositional verification
  - use of L* to learn assumptions for A/G reasoning
  - [Pasareanu/Giannakopoulou/et al.]
  - uses notion of "weakest assumption" about a component that suffices for compositional verification (always exists)
  - weakest assumption is the target regular language
- Fully automated L* learning loop
  - 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
- Successfully applied to several large case studies
  - does particularly well when assumption/alphabet are small
  - much recent interest in learning for verification…

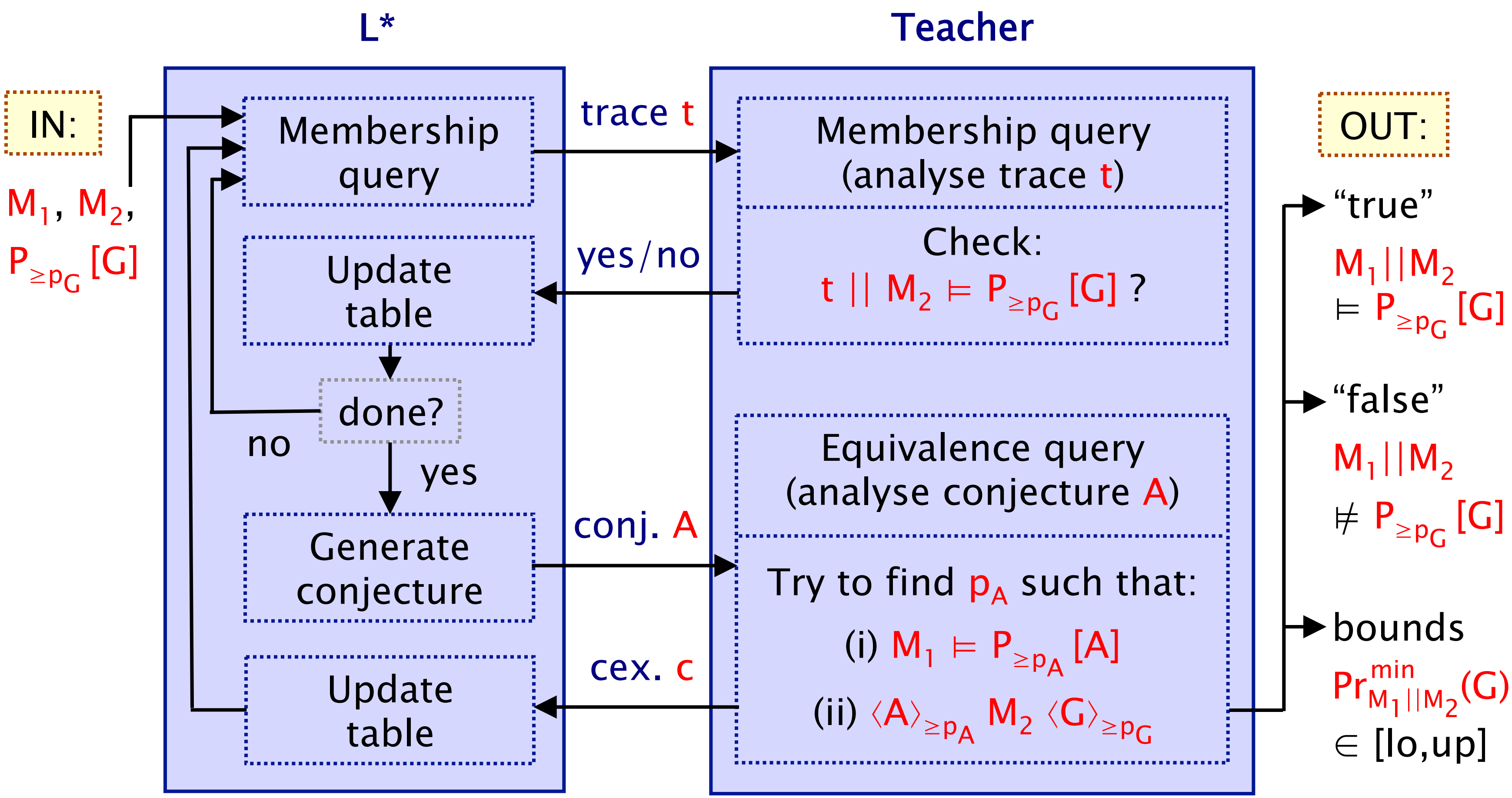
Probabilistic assumption generation

- Goal: automate A/G rule (ASYM)
  - generate probabilistic assumption $P_{\geq p_A}[A]$
  - for checking property $P_{\geq p_G}[G]$ on $M_1 \,||\, M_2$

$$\frac{M_1 \models P_{\geq p_A}[A] \qquad \langle A \rangle_{\geq p_A} M_2 \langle G \rangle_{\geq p_G}}{M_1 \,||\, M_2 \models P_{\geq p_G}[G]}$$

- Reduce problem to generation of non-probabilistic assumption A
  - then (if possible) find lowest p_A such that premises 1 & 2 hold
  - in fact, for fixed A, we can generate lower and upper bounds on $Pr_{M_1||M_2}^{min}(G)$, which may suffice to verify/refute $P_{\geq p_G}[G]$
- Use adapted L* to learn non-probabilistic assumption A
  - note: there is no "weakest assumption" (AG rule is incomplete)
  - but can generate sequence of conjectures for A in similar style
  - "teacher" based on a probabilistic model checker (PRISM), feedback is from probabilistic counterexamples [Han/Katoen]
  - three outcomes of loop: "true", "false", lower/upper bounds

Probabilistic assumption generation



Implementation + Case studies

- Implemented using:
  - extension of PRISM model checker
  - libalf learning library [Bollig et al.]

- Several case studies
  - client-server (A/G model checking benchmark + failures)
    - minimum probability mutual exclusion not violated
  - randomised consensus algorithm [Aspnes & Herlihy]
    - minimum probability consensus reached by round R
  - sensor network [QEST'10]
    - minimum probability of processor error occurring
  - Mars Exploration Rovers (MER) [NASA]
    - minimum probability mutual exclusion not violated in k cycles

Experimental results (learning)

Case study [parameters]		Component sizes		Compositional	
		$\lvert M_2 \otimes G_{err} \rvert$	$\lvert M_1 \rvert$	$\lvert A^{err} \rvert$	Time (s)
Client-server (N failures) [N]	3	229	16	5	6.6
	4	1,121	25	6	26.1
	5	5,397	36	7	191.1
Randomised consensus [N,R,K]	2, 3, 20	391	3,217	6	24.2
	2, 4, 4	573	431,649	12	413.2
	3, 3, 20	8,843	38,193	11	438.9
Sensor network [N]	2	42	1,184	3	3.7
	3	42	10,662	3	4.6
MER [N R]	2, 5	5,776	427,363	4	31.8
	3, 2	16,759	171	4	210.5

Experimental results (learning)

Case study [parameters]		Component sizes		Compositional	
		$\|M_2 \otimes G_{err}\|$	$\|M_1\|$	$\|A^{err}\|$	Time (s)
Client-server (N failures) [N]	3	229	16	5	6.6
	4	1,121	25	6	26.1
	5	5,397	36	7	191.1
Randomised consensus [N,R,K]	2, 3, 20	391	3,217	6	24.2
	2, 4, 4	573	431,649	12	413.2
	3, 3, 20	8,843	38,193	11	438.9
Sensor network [N]	2	42	1,184	3	3.7
	3	42	10,662	3	4.6
MER [N R]	2, 5	5,776	427,363	4	31.8
	3, 2	16,759	171	4	210.5

- Successfully learnt (small) assumptions in all cases

Experimental results (learning)

Case study [parameters]		Component sizes		Compositional	
		$\lvert M_2 \otimes G_{err} \rvert$	$\lvert M_1 \rvert$	$\lvert A^{err} \rvert$	Time (s)
Client-server (N failures) [N]	3	229	16	5	6.6
	4	1,121	25	6	26.1
	5	5,397	36	7	191.1
Randomised consensus [N,R,K]	2, 3, 20	391	3,217	6	$\boxed{24.2}$
	2, 4, 4	573	431,649	12	413.2
	3, 3, 20	8,843	38,193	11	$\boxed{438.9}$
Sensor network [N]	2	42	1,184	3	3.7
	3	42	10,662	3	4.6
MER [N R]	2, 5	5,776	427,363	4	31.8
	3, 2	16,759	171	4	210.5

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

Summary (Part 4)

- 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], [FKN+11]

- Next: PRISM lab session…

Thanks for your attention

More info here:
www.prismmodelchecker.org