Probabilistic verification and synthesis

Marta Kwiatkowska

Department of Computer Science, University of Oxford

KTH, Stockholm, August 2015

Lecture plan

- Course slides and lab session
  - http://www.prismmodelchecker.org/courses/kth15/

- 5 sessions: lectures 9-12noon, labs 2.30-5pm
  - 1 – Introduction
  - 2 – Discrete time Markov chains (DTMCs)
  - 3 – Markov decision processes (MDPs)
  - 4 – LTL model checking for DTMCs/MDPs & beyond MDPs
  - 5 – Probabilistic timed automata (PTAs)

- For extended versions of this material
  - and an accompanying list of references
  - see: http://www.prismmodelchecker.org/lectures/

Probabilistic models

	Fully probabilistic	Nondeterministic
Discrete time	Discrete-time Markov chains (DTMCs)	Markov decision processes (MDPs)
		Simple stochastic games (SMGs)
Continuous time	Continuous-time Markov chains (CTMCs)	Probabilistic timed automata (PTAs)
		Interactive Markov chains (IMCs)

Part 4

LTL Model Checking; Beyond MDPs

Overview (Part 4)

- Linear temporal logic (LTL)
- Strongly connected components
- ω-automata (Büchi, Rabin)
- LTL model checking for DTMCs
- LTL model checking for MDPs
- Beyond MDPs: stochastic multiplayer games

Limitations of PCTL

- PCTL, although useful in practice, has limited expressivity
  - essentially: probability of reaching states in X, passing only through states in Y (and within k time-steps)

- One useful approach: extend models with costs/rewards
  - see last two lectures

- Another direction: Use more expressive logics. e.g.:
  - LTL [Pnu77] – (non-probabilistic) linear-time temporal logic
  - PCTL* [ASB+95,BdA95] – which subsumes both PCTL and LTL
  - both allow path operators to be combined
  - (in PCTL, $P_{\sim p}$ [...] always contains a single temporal operator)

LTL – Linear temporal logic

- LTL syntax (path formulae only)
  - $\psi ::= \text{true} \mid a \mid \psi \wedge \psi \mid \neg\psi \mid X\,\psi \mid \psi\; U\; \psi$
  - where $a \in AP$ is an atomic proposition
  - usual equivalences hold: $F\,\phi \equiv \text{true}\; U\; \phi$, $G\,\phi \equiv \neg(F\,\neg\phi)$

- LTL semantics (for a path ω)
  - $\omega \vDash \text{true}$ always
  - $\omega \vDash a \quad \Leftrightarrow \quad a \in L(\omega(0))$
  - $\omega \vDash \psi_1 \wedge \psi_2 \quad \Leftrightarrow \quad \omega \vDash \psi_1$ and $\omega \vDash \psi_2$
  - $\omega \vDash \neg\psi \quad \Leftrightarrow \quad \omega \nvDash \psi$
  - $\omega \vDash X\,\psi \quad \Leftrightarrow \quad \omega[1\ldots] \vDash \psi$
  - $\omega \vDash \psi_1\; U\; \psi_2 \quad \Leftrightarrow \quad \exists k \geq 0 \text{ s.t. } \omega[k\ldots] \vDash \psi_2 \wedge \forall i<k\; \omega[i\ldots] \vDash \psi_1$

where $\omega(i)$ is i^{th} state of ω, and $\omega[i\ldots]$ is suffix starting at $\omega(i)$

LTL examples

- $(F\ tmp\_fail_1) \wedge (F\ tmp\_fail_2)$
  - "both servers suffer temporary failures at some point"
- GF ready
  - "the server always eventually returns to a ready-state"
- FG error
  - "an irrecoverable error occurs"
- G (req → X ack)
  - "requests are always immediately acknowledged"

LTL for DTMCs

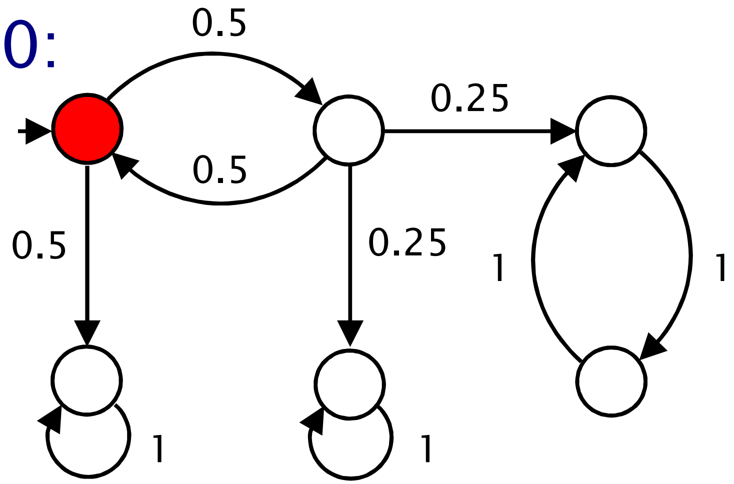
- Same idea as PCTL: probabilities of sets of path formulae
  - for a state s of a DTMC and an LTL formula ψ:
  - $Prob(s, \psi) = Pr_s \{ \omega \in Path(s) \mid \omega \vDash \psi \}$
  - all such path sets are measurable [Var85]

- A (probabilistic) LTL specification often comprises an LTL (path) formula and a probability bound
  - e.g. $P_{\geq 1}$ [GF ready] – “with probability 1, the server always eventually returns to a ready-state”
  - e.g. $P_{\leq 0.01}$ [FG error] – “with probability at most 0.01, an irrecoverable error occurs”

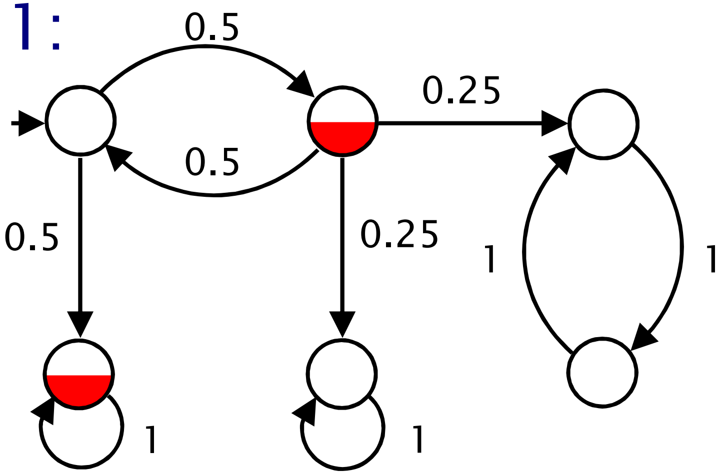
- PCTL* subsumes both LTL and PCTL
  - e.g. $P_{>0.5}$ [GF $crit_1$] $\wedge$ $P_{>0.5}$ [GF $crit_2$]

Long-run behaviour of DTMCs

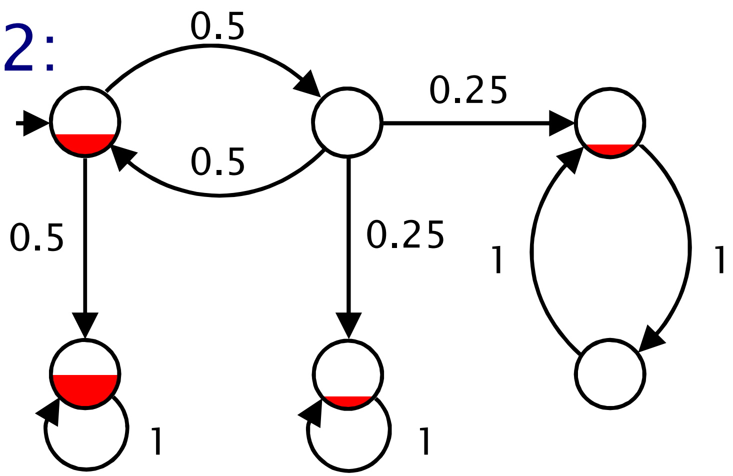
k=0:



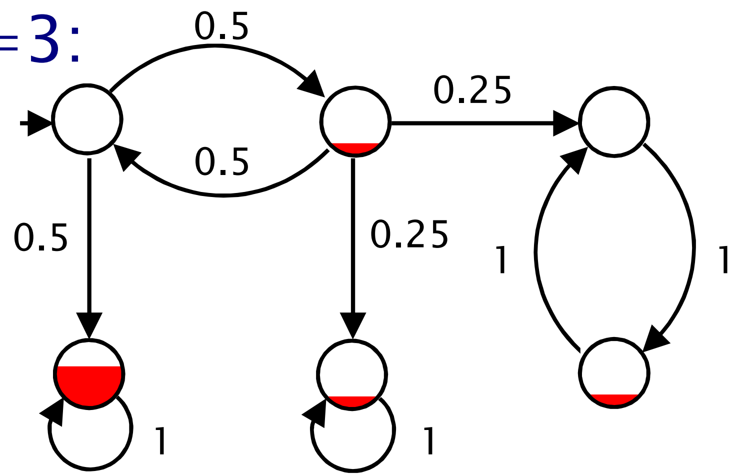
k=1:



k=2:



k=3:



Strongly connected components

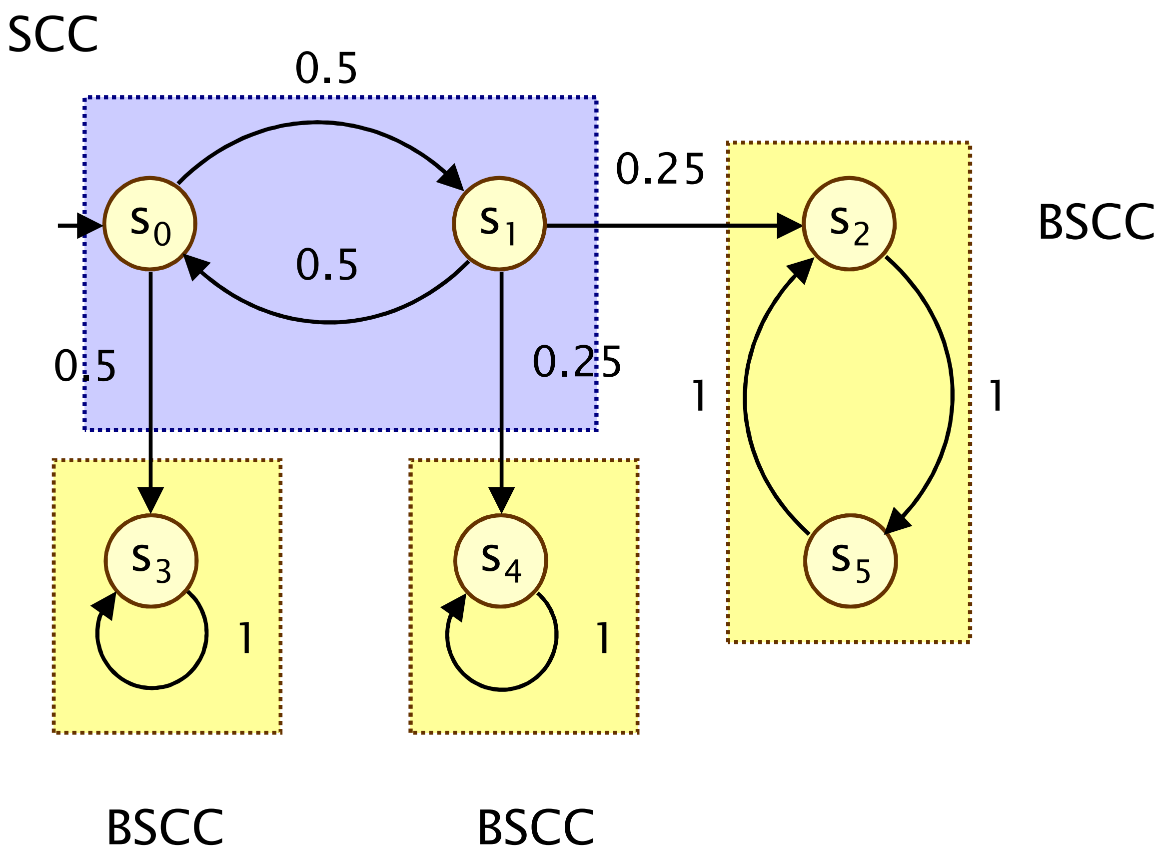
- Long-run properties of DTMCs rely on an analysis of their underlying graph structure (i.e. ignoring probabilities)

- Strongly connected set of states T
  - for any pair of states s and s' in T, there is a path from s to s', passing only through states in T

- Strongly connected component (SCC)
  - a maximally strongly connected set of states (i.e. no superset of it is also strongly connected)

- Bottom strongly connected component (BSCC)
  - an SCC T from which no state outside T is reachable from T

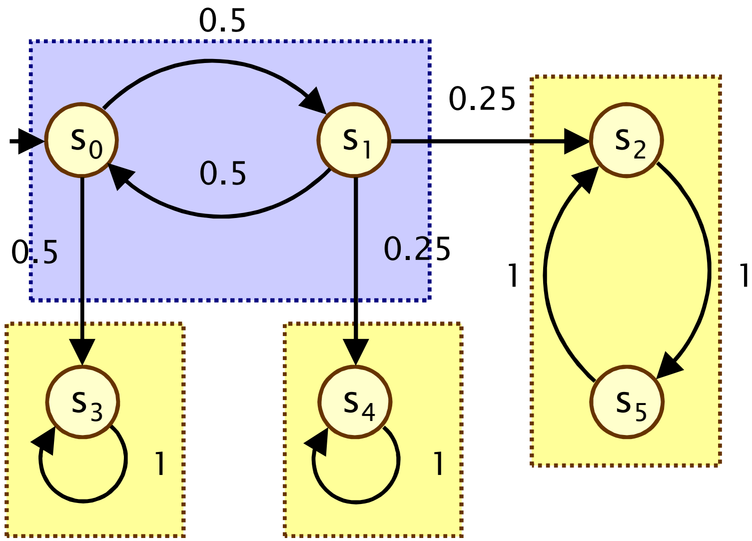
Example – (B)SCCs



Fundamental property of DTMCs

- Fundamental property of (finite) DTMCs…

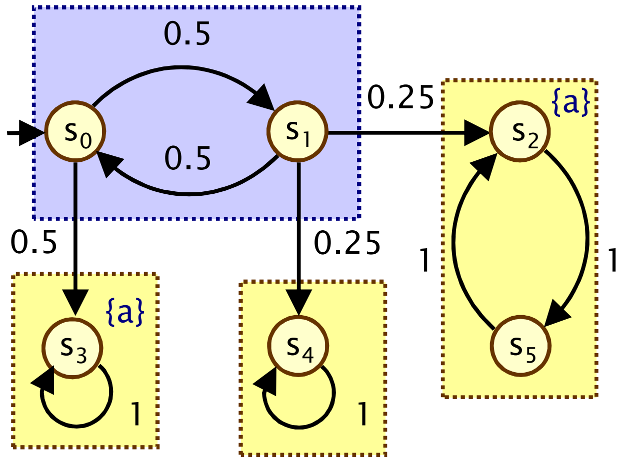
- With probability 1,
  some BSCC will be reached
  and all of its states
  visited infinitely often



- Formally:
  - Pr_{s0} ($s_0 s_1 s_2 \ldots$ | $\exists$ $i \geq 0$, $\exists$ BSCC T such that
    $\forall$ $j \geq i$ $s_j \in T$ and
    $\forall$ $s \in T$ $s_k = s$ for infinitely many k) = 1

LTL model checking for DTMCs

- LTL model checking for DTMCs relies on:
  - computing the probability Prob(s, ψ) for LTL formula ψ
  - reduces to probability of reaching a set of “accepting” BSCCs
  - 2 simple cases: GF a and FG a…
- Prob(s, GF a) = Prob(s, F T_{GFa})
  - where T_{GFa} = union of all BSCCs containing some state satisfying a
- Prob(s, FG a) = Prob(s, F T_{FGa})
  - where T_{FGa} = union of all BSCCs containing only a-states
- To extend this idea to arbitrary LTL formula, we use ω-automata…



Example:

Prob(s_0, GF a)

= Prob(s_0, F T_{GFa})

= Prob(s_0, F {s_3,s_2,s_5})

= 2/3 + 1/6 = 5/6

Overview (Part 3)

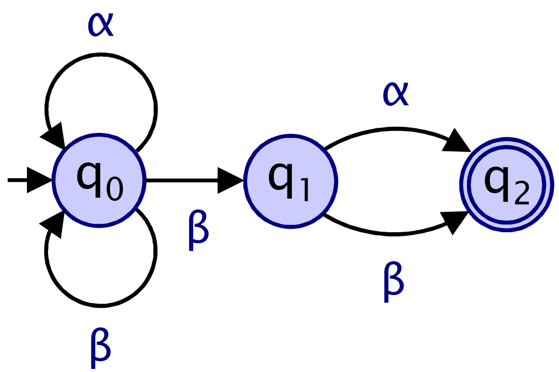
- Linear temporal logic (LTL)
- Strongly connected components
- ω-automata (Büchi, Rabin)
- LTL model checking for DTMCs
- LTL model checking for MDPs
- Beyond MDPs: stochastic multiplayer games

Reminder – Finite automata

- A regular language over alphabet Σ
  - is a set of finite words $L \subseteq \Sigma^*$ such that either:
  - $L = L(E)$ for some regular expression E
  - $L = L(A)$ for some nondeterministic finite automaton (NFA) A
  - $L = L(A)$ for some deterministic finite automaton (DFA) A

- Example:

  Regexp: $(\alpha+\beta)^*\beta(\alpha+\beta)$ NFA A:



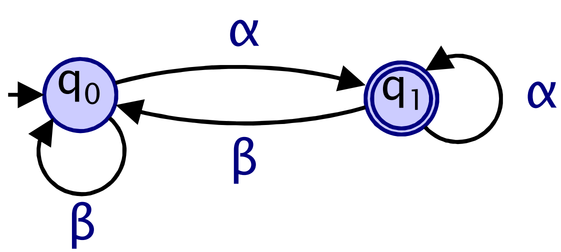
- NFAs and DFAs have the same expressive power
  - we can always determinise an NFA to an equivalent DFA
  - (with a possibly exponential blow-up in size)

Büchi automata

- ω-automata represent sets of infinite words $L \subseteq \Sigma^\omega$
  - e.g. Büchi automata, Rabin automata, Streett, Muller, …

- A nondeterministic Büchi automaton (NBA) is…
  - a tuple $A = (Q, \Sigma, \delta, Q_0, F)$ where:
  - Q is a finite set of states
  - Σ is an alphabet
  - $\delta : Q \times \Sigma \rightarrow 2^Q$ is a transition function
  - $Q_0 \subseteq Q$ is a set of initial states
  - $F \subseteq Q$ is a set of "accept" states

Example:
words $w \in \{\alpha,\beta\}^\omega$
with infinitely many α



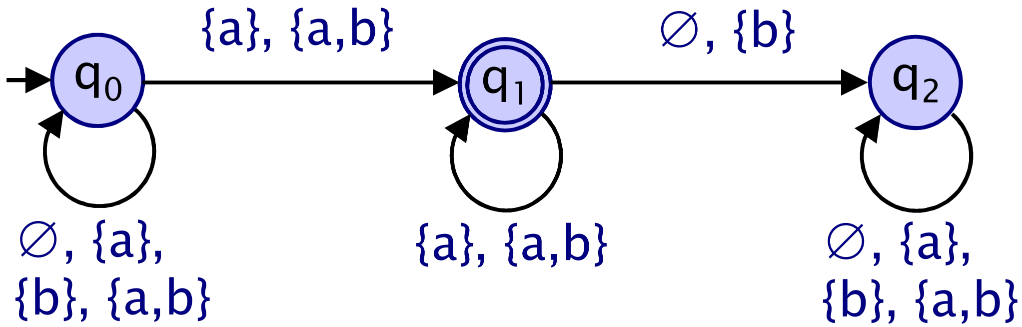
- NBA acceptance condition
  - language $L(A)$ for A contains $w \in \Sigma^\omega$ if there is a corresponding run in A that passes through states in F infinitely often

ω-regular properties

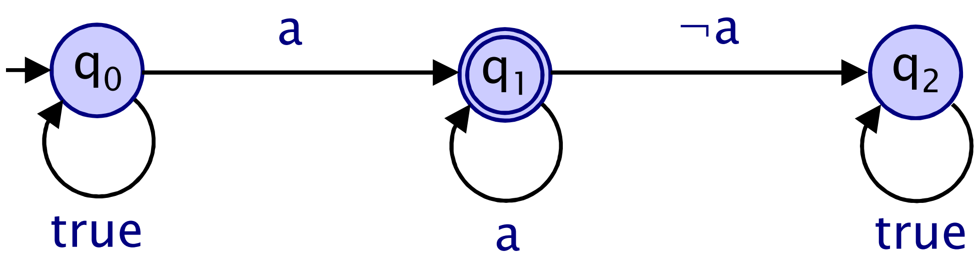
- Consider a model, i.e. an LTS/DTMC/MDP/…
  - for example: DTMC $D = (S, s_{init}, P, Lab)$
  - where labelling Lab uses atomic propositions from set AP
- We can capture properties of these using ω-automata
  - let $\omega \in Path(s)$ be some infinite path in D
  - $trace(\omega) \in (2^{AP})^\omega$ denotes the projection of state labels of ω
  - i.e. $trace(s_0 s_1 s_2 s_3 \ldots) = Lab(s_0)Lab(s_1)Lab(s_2)Lab(s_3)\ldots$
  - can specify a set of paths of D with an ω-automaton over 2^{AP}
- Let $Prob^D(s, A)$ denote the probability…
  - from state s in a discrete-time Markov chain D
  - of satisfying the property specified by automaton A
  - i.e. $Prob^D(s, A) = Pr^D_s\{\ \omega \in Path(s) \mid trace(\omega) \in L(A)\ \}$

Example

- Nondeterministic Büchi automaton
  - for LTL formula FG a, i.e. “eventually always a”
  - for a DTMC with atomic propositions AP = {a,b}



- We abbreviate this to just:



Büchi automata + LTL

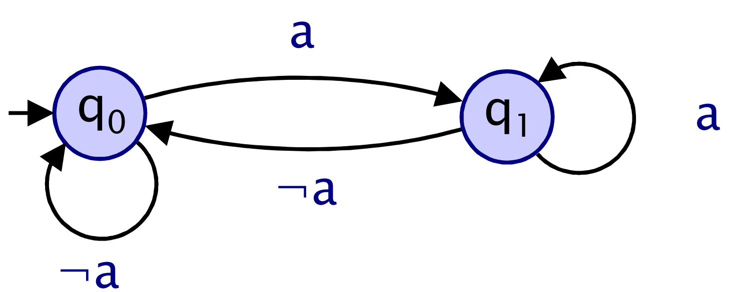
- Nondeterministic Büchi automata (NBAs)
  - define the set of ω-regular languages
- ω-regular languages are more expressive than LTL
  - can convert any LTL formula ψ over atomic propositions AP
  - into an equivalent NBA A_ψ over 2^{AP}
  - i.e. $\omega \vDash \psi \Leftrightarrow \text{trace}(\omega) \in L(A_\psi)$ for any path ω
  - for LTL-to-NBA translation, see e.g. [VW94], [DGV99], [BK08]
  - worst-case: exponential blow-up from $|\psi|$ to $|A_\psi|$
- But deterministic Büchi automata (DBAs) are less expressive
  - e.g. there is no DBA for the LTL formula FG a
  - for probabilistic model checking, need deterministic automata
  - so we use deterministic Rabin automata (DRAs)

Deterministic Rabin automata

- A deterministic Rabin automaton is a tuple $(Q, \Sigma, \delta, q_0, Acc)$:
  - Q is a finite set of states, $q_0 \in Q$ is an initial state
  - Σ is an alphabet, $\delta : Q \times \Sigma \to Q$ is a transition function
  - $Acc = \{ (L_i, K_i) \}_{i=1..k} \subseteq 2^Q \times 2^Q$ is an acceptance condition

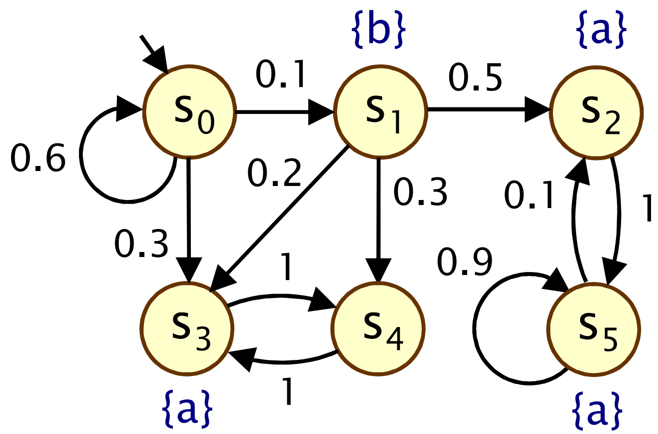
- A run of a word on a DRA is accepting iff:
  - for some pair (L_i, K_i), the states in L_i are visited finitely often and (some of) the states in K_i are visited infinitely often
  - or in LTL: $\bigvee_{1 \leq i \leq k} (\mathrm{FG}\, \neg L_i \wedge \mathrm{GF}\, K_i)$

- Example: DRA for FG a
  - acceptance condition is $Acc = \{ (\{q_0\},\{q_1\}) \}$



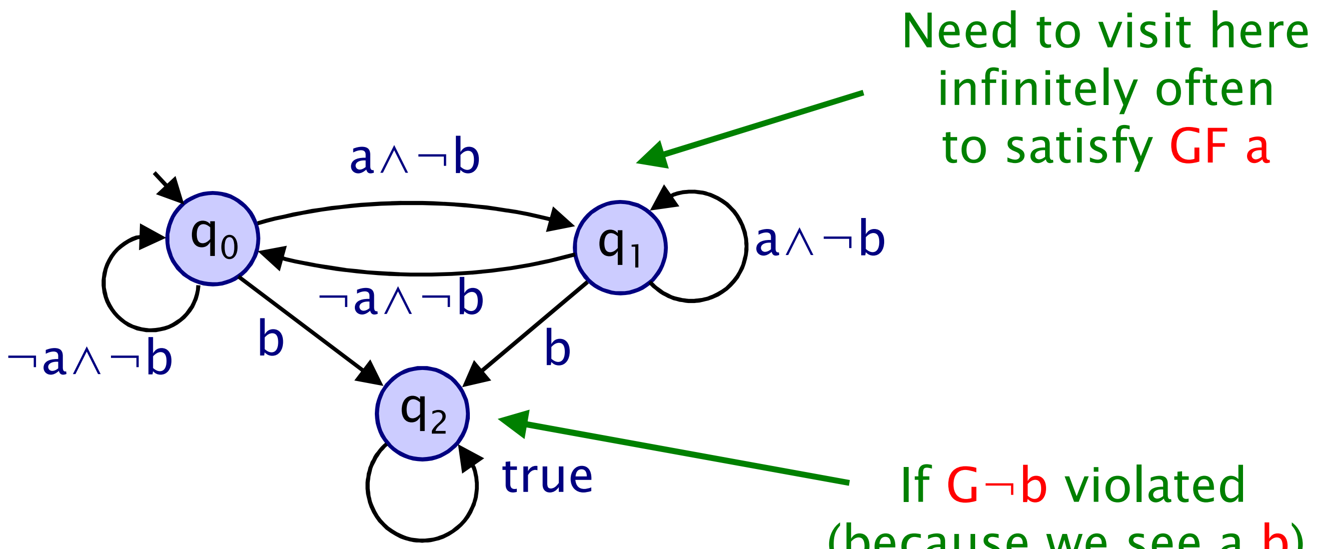
LTL model checking for DTMCs

- LTL model checking for DTMC D and LTL formula ψ
- 1. Construct DRA A_ψ for ψ
- 2. Construct product D ⊗ A of DTMC D and DRA A_ψ
- 3. Compute $\mathrm{Prob}^D(s, \psi)$ from DTMC D ⊗ A
- Running example:
  - compute probability of satisfying LTL formula $\psi = G\neg b \wedge GF\ a$ on:



Example – DRA

- DRA A_ψ for $\psi = G\neg b \wedge GF\ a$
  - acceptance condition is Acc = { ({},{q_1}) }
  - (i.e. this is actually a deterministic Büchi automaton)



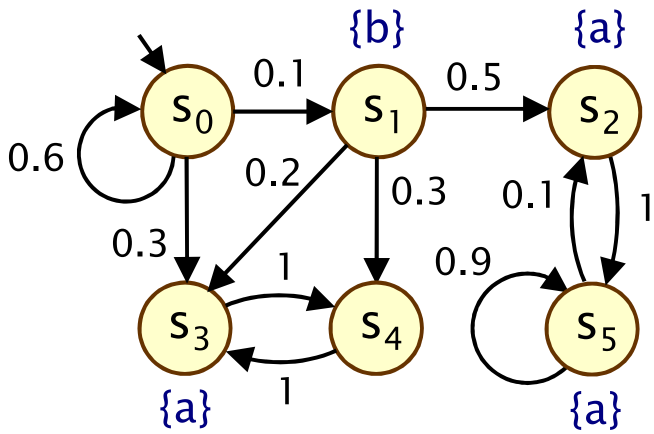
Product DTMC for a DRA

- We construct the product DTMC
  - for DTMC D and DRA A, denoted $D \otimes A$
  - $D \otimes A$ can be seen as an unfolding of D with states (s,q), where q records state of automaton A for path fragment so far
  - since A is deterministic, $D \otimes A$ is a also a DTMC
  - each path in D has a corresponding (unique) path in $D \otimes A$
  - the probabilities of paths in D are preserved in $D \otimes A$

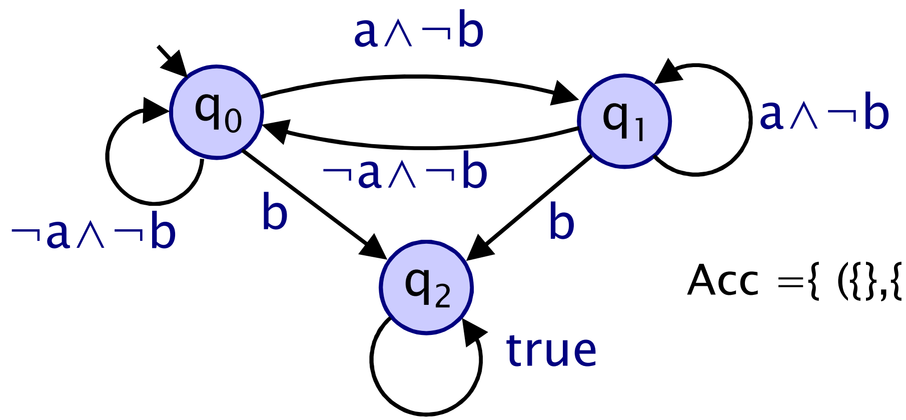
- Formally, for $D = (S, s_{init}, P, L)$ and $A = (Q, \Sigma, \delta, q_0, \{(L_i, K_i)\}_{i=1..k})$
  - $D \otimes A$ is the DTMC $(S \times Q, (s_{init}, q_{init}), P', L')$ where:
  - $q_{init} = \delta(q_0, L(s_{init}))$
  - $P'((s_1,q_1),(s_2,q_2)) = \begin{cases} P(s_1,s_2) & \text{if } q_2 = \delta(q_1, L(s_2)) \\ 0 & \text{otherwise} \end{cases}$
  - $l_i \in L'(s,q)$ if $q \in L_i$ and $k_i \in L'(s,q)$ if $q \in K_i$

Example – Product DTMC

DTMC D

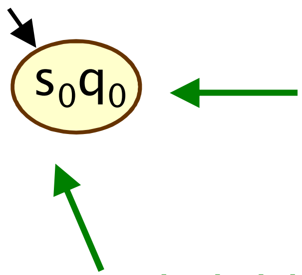


DRA A_ψ for $\psi = G\neg b \wedge GF\, a$



Acc ={ ({},{q_1}) }

Product DTMC $D \otimes A_\psi$

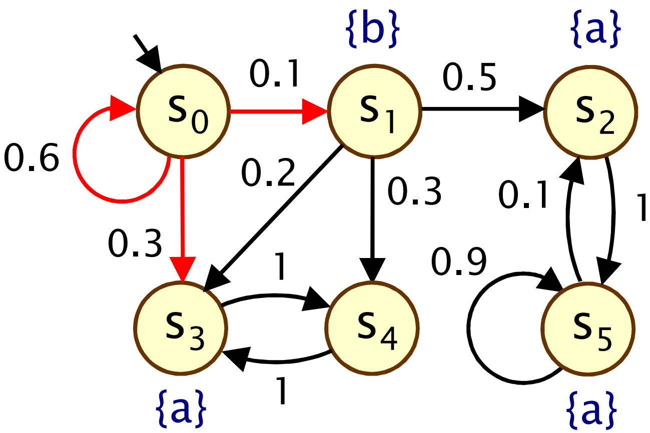


s_0 satisfies neither a or b so we stay in q_0 in DRA A_ψ

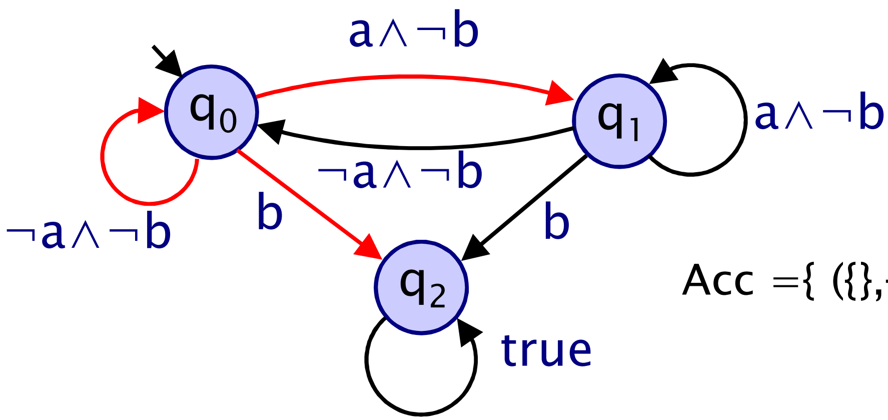
s_0 is initial state of DTMC D

Example – Product DTMC

DTMC D

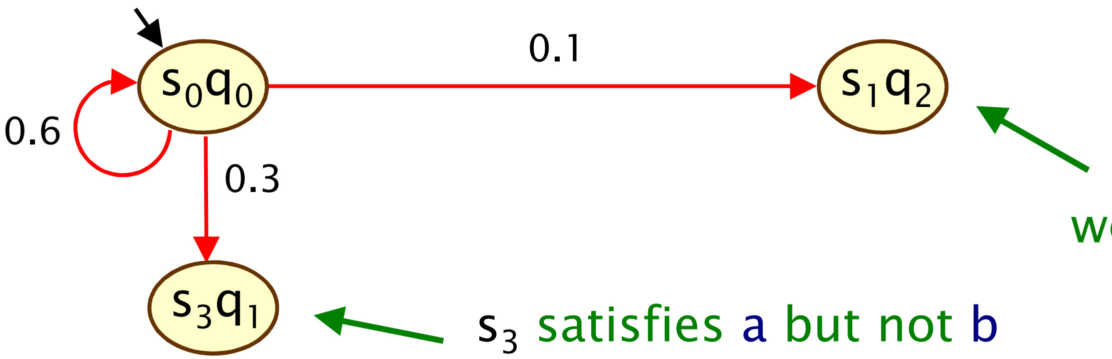


DRA A_ψ for $\psi = G\neg b \wedge GF\ a$



Acc $=\{ (\{\},\{q_1\}) \}$

Product DTMC $D \otimes A_\psi$

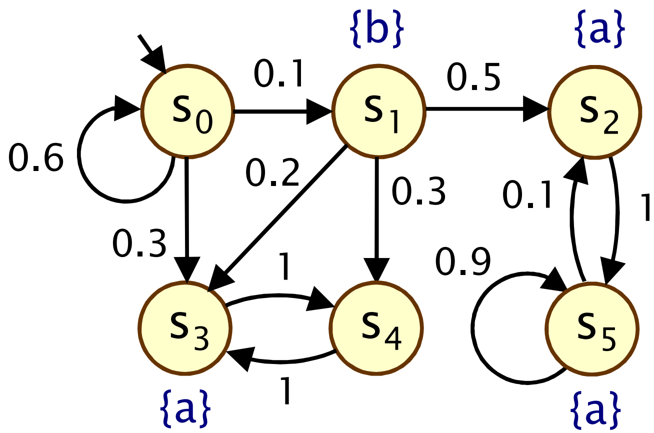


s_1 satisfies b so we move to q_2 in A_ψ

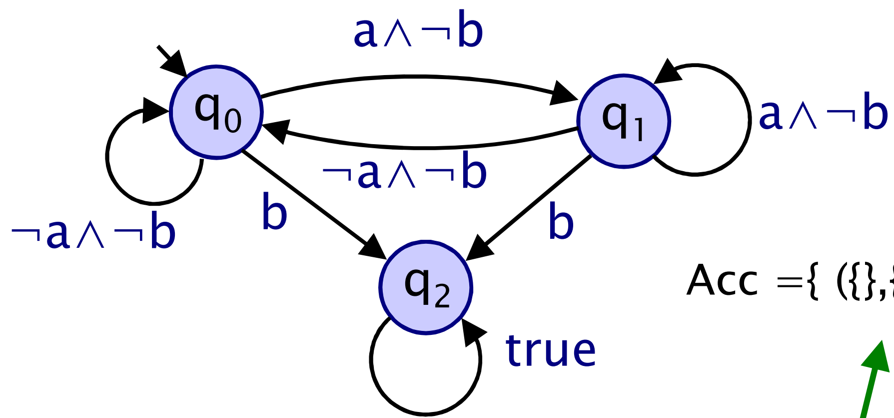
s_3 satisfies a but not b so we move to q_1 in A_ψ

Example – Product DTMC

DTMC D



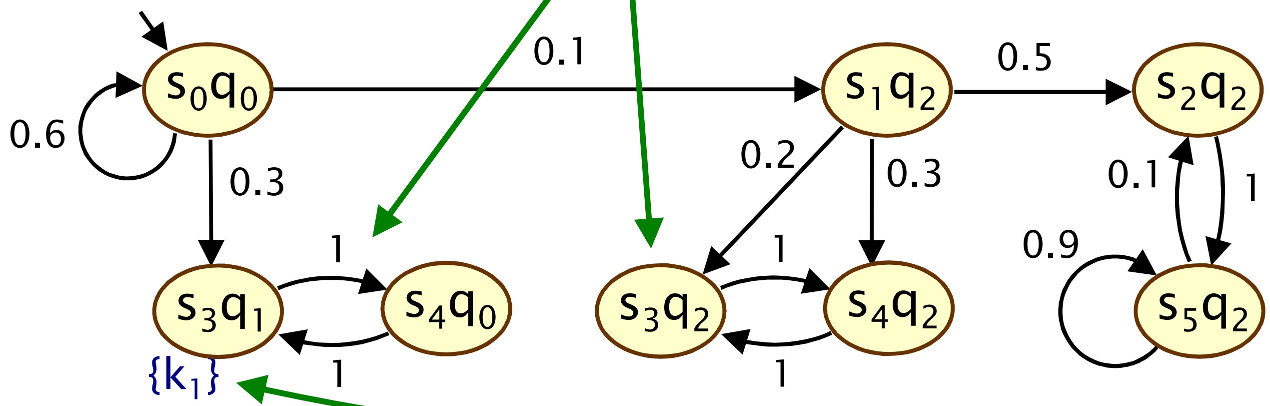
DRA A_ψ for $\psi = G\neg b \wedge GF\ a$



Acc ={ ({},{q_1}) }

Product DTMC $D \otimes A_\psi$

2 copies of s_3/s_4, one after seeing a b and one no b's



label states satisfying acceptance pair (L_1,K_1)

Product DTMC for a DRA

- For DTMC D and DRA A

$$\text{Prob}^{D}(s, A) = \text{Prob}^{D\otimes A}((s,q_s), \vee_{1\leq i\leq k} (FG\ \neg l_i \wedge GF\ k_i))$$

  - where $q_s = \delta(q_0,L(s))$

- Hence:

$$\text{Prob}^{D}(s, A) = \text{Prob}^{D\otimes A}((s,q_s), F\ T_{Acc})$$

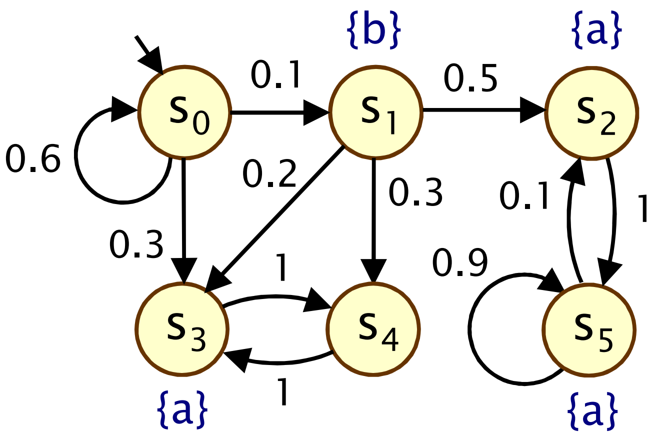
  - where T_{Acc} is the union of all accepting BSCCs in $D\otimes A$
  - an accepting BSCC T of $D\otimes A$ is such that, for some $1\leq i\leq k$, no states in T satisfy l_i and some state in T satisfies k_i

- Reduces to computing BSCCs and reachability probabilities

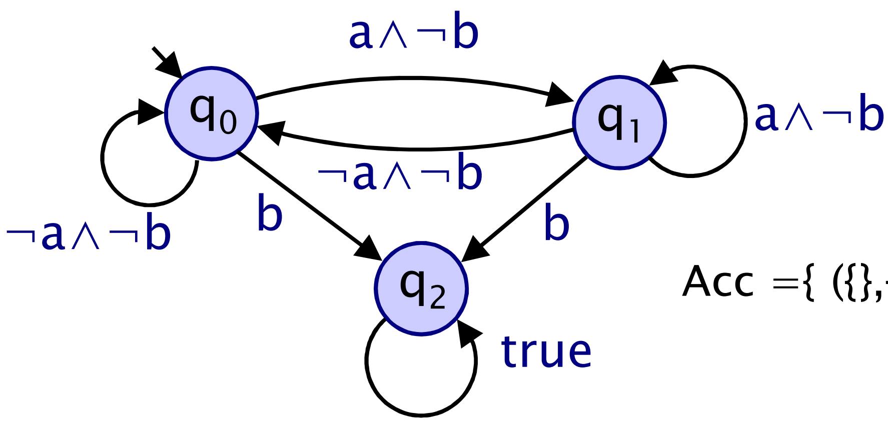
Example: LTL for DTMCs

- Compute Prob(s_0, G¬b ∧ GF a) for DTMC D:

DTMC D



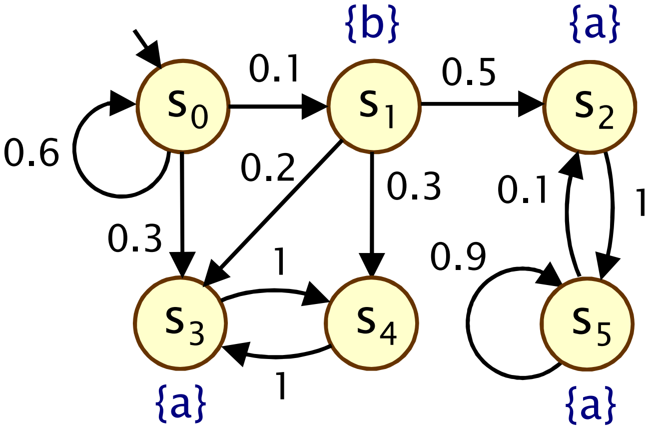
DRA A_ψ for ψ = G¬b ∧ GF a



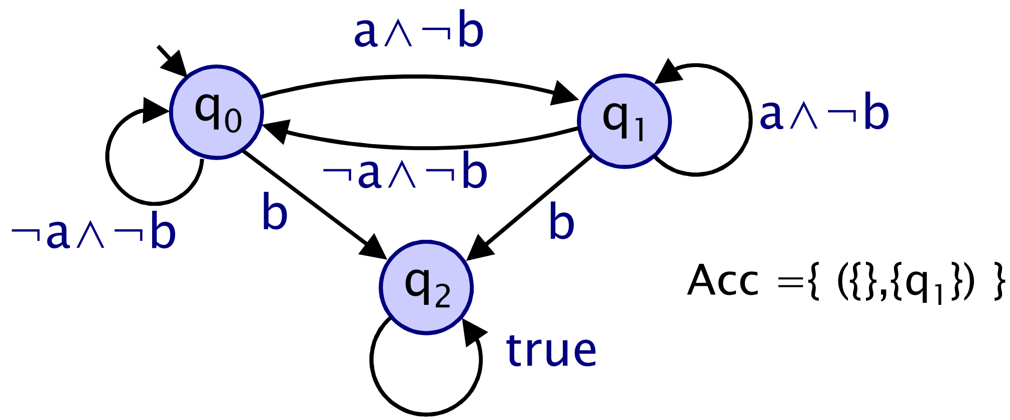
Acc ={ ({},{q_1}) }

Example: LTL for DTMCs

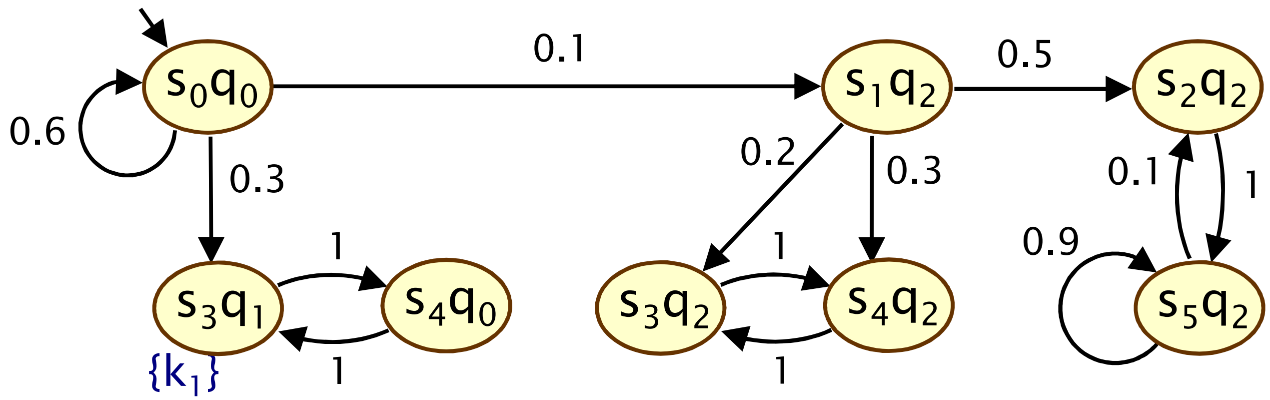
DTMC D



DRA A_ψ for $\psi = G\neg b \wedge GF\ a$

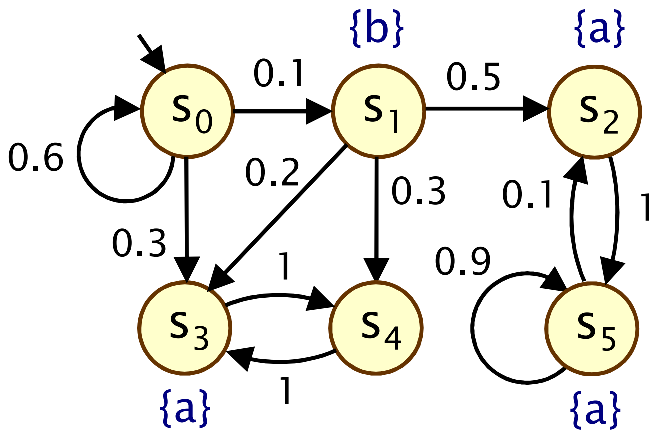


Product DTMC $D \otimes A_\psi$

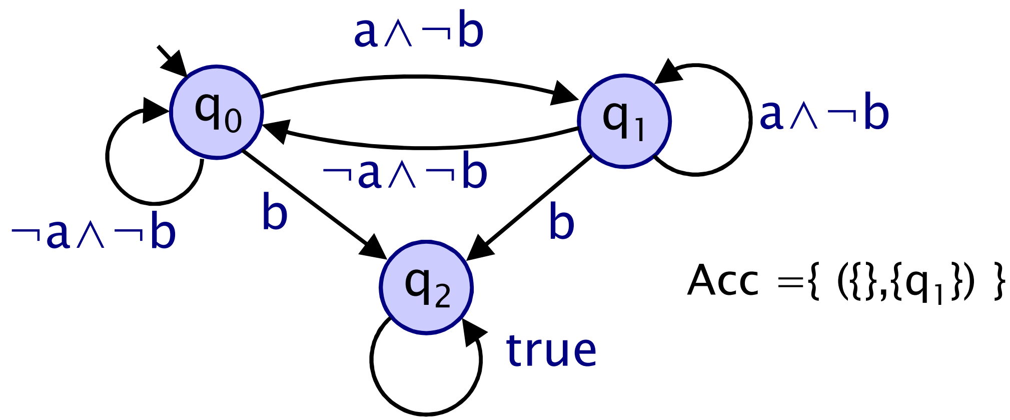


Example: LTL for DTMCs

DTMC D

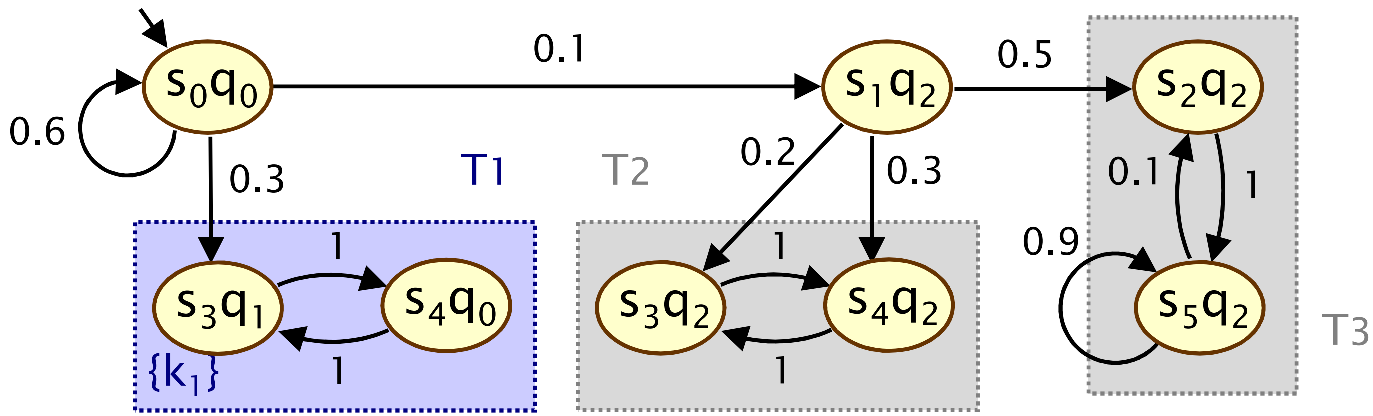


DRA A_ψ for $\psi = G\neg b \wedge GF\, a$



Product DTMC $D \otimes A_\psi$

$Prob^{D}(s_0, \psi) = Prob^{D \otimes A\psi}(s_0 q_0, F\ T_1) = 3/4$



Complexity of LTL model checking

- Complexity of model checking LTL formula ψ on DTMC D
  - is doubly exponential in $|\psi|$ and polynomial in $|D|$
  - (for the algorithm presented in these lectures)
- Double exponential blow-up comes from use of DRAs
  - size of NBA can be exponential in $|\psi|$
  - and DRA can be exponentially bigger than NBA
  - in practice, this does not occur and ψ is small anyway
- Polynomial-time operations required on product model
  - BSCC computation – linear in (product) model size
  - probabilistic reachability – cubic in (product) model size
- In total: $O(poly(|D|,|A_\psi|))$

- Complexity can be reduced to single exponential in $|\psi|$
  - see e.g. [CY88,CY95]

PCTL* model checking

- PCTL* syntax:
  - ϕ ::= true | a | $\phi \wedge \phi$ | $\neg\phi$ | $P_{\sim p}$ [ψ]
  - ψ ::= ϕ | $\psi \wedge \psi$ | $\neg\psi$ | X ψ | ψ U ψ
- Example:
  - $P_{>p}$ [GF (send → $P_{>0}$ [F ack])]

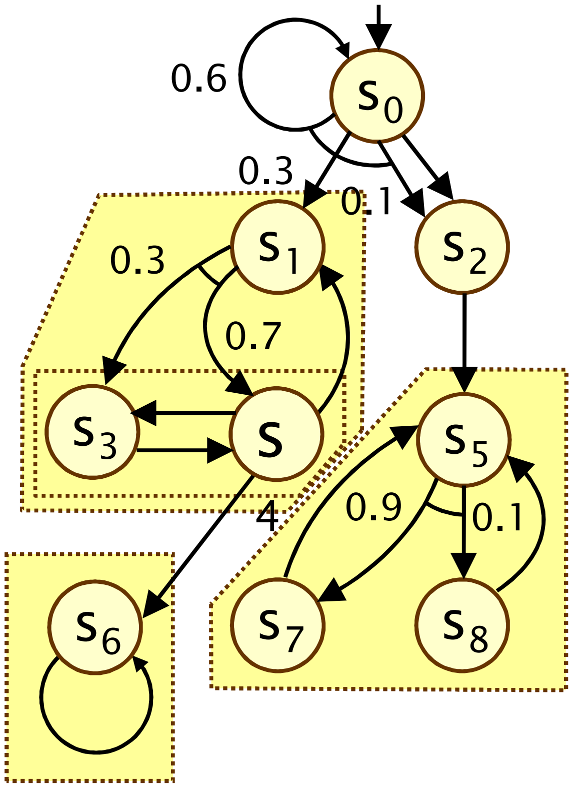
- PCTL* model checking algorithm
  - bottom-up traversal of parse tree for formula (like PCTL)
  - to model check $P_{\sim p}$ [ψ]:
    - replace maximal state subformulae with atomic propositions
    - (state subformulae already model checked recursively)
    - modified formula ψ is now an LTL formula
    - which can be model checked as for LTL

Overview (Part 4)

- Linear temporal logic (LTL)
- Strongly connected components
- ω-automata (Büchi, Rabin)
- LTL model checking for DTMCs
- LTL model checking for MDPs
- Beyond MDPs: stochastic multiplayer games

End components in MDPs

- End components of MDPs are the analogue of BSCCs in DTMCs
- An end component is a strongly connected sub-MDP
- A sub-MDP comprises a subset of states and a subset of the actions/distributions available in those states, which is closed under probabilistic branching

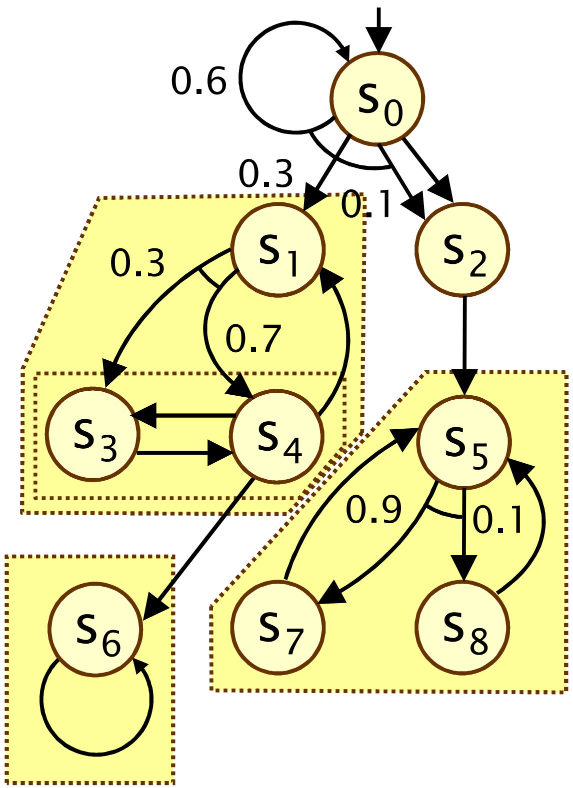


Note:

- action labels omitted
- probabilities omitted where =1

Recall – end components in MDPs

- End components of MDPs
  are the analogue of BSCCs in DTMCs
- For every end component, there
  is an adversary which, with
  probability 1, forces the MDP
  to remain in the end component,
  and visit all its states infinitely often
- Under every adversary σ, with
  probability 1 some end component
  will be reached and all of its
  states visited infinitely often
  (union of ECs reached with prob 1)



Long-run properties of MDPs

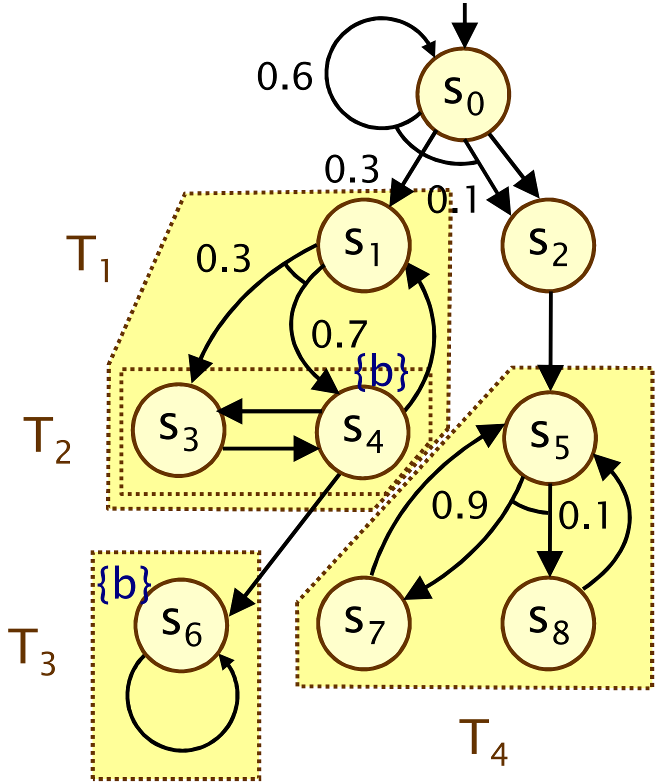
- Maximum probabilities
  - $p_{max}(s, GF\ a) = p_{max}(s, F\ T_{GFa})$
    - where T_{GFa} is the union of sets T for all end components (T,Steps’) with $T \cap Sat(a) \neq \varnothing$
  - $p_{max}(s, FG\ a) = p_{max}(s, F\ T_{FGa})$
    - where T_{FGa} is the union of sets T for all end components (T,Steps’) with $T \subseteq Sat(a)$
- Minimum probabilities
  - need to compute from maximum probabilities…
  - $p_{min}(s, GF\ a) = 1 - p_{max}(s, FG\neg a)$
  - $p_{min}(s, FG\ a) = 1 - p_{max}(s, GF\neg a)$

Example

- Model check: $P_{<0.8}$ [GF b] for s_0

- Compute p_{max}(GF b)
  - p_{max}(GF b) = p_{max}(s, F T_{GFb})
  - T_{GFb} is the union of sets T for all end components with T ∩ Sat(b) ≠ ∅
  - Sat(b) = { s_4, s_6 }
  - T_{GFb} = $T_1 \cup T_2 \cup T_3$ = { s_1, s_3 s_4, s_6 }
  - p_{max}(s, F T_{GFb}) = 0.75
  - p_{max}(GF b) = 0.75

- Result: $s_0 \vDash P_{<0.8}$ [GF b]



Automata-based properties for MDPs

- For an MDP M and automaton A over alphabet 2^{AP}
  - consider probability of "satisfying" language $L(A) \subseteq (2^{AP})^{\omega}$
  - $Prob^{M,adv}(s, P) = Pr_s^{M,adv} \{ \omega \in Path^{M,adv}(s) \mid trace(\omega) \in L(A) \}$
  - $p_{max}{}^{M}(s, A) = \sup_{adv \in Adv} Prob^{M,adv}(s, A)$
  - $p_{min}{}^{M}(s, A) = \inf_{adv \in Adv} Prob^{M,adv}(s, A)$

- Might need minimum or maximum probabilities
  - e.g. $s \models P_{\geq 0.99} [\ \psi_{good}\ ] \Leftrightarrow p_{min}{}^{M}(s, \psi_{good}) \geq 0.99$
  - e.g. $s \models P_{\leq 0.05} [\ \psi_{bad}\ ] \Leftrightarrow p_{max}{}^{M}(s, \psi_{bad}) \leq 0.05$
- But, ψ-regular properties are closed under negation
  - as are the automata that represent them
  - so can always consider maximum probabilities…
  - $p_{max}{}^{M}(s, \psi_{bad})$ or $1 - p_{max}{}^{M}(s, \neg\psi_{good})$

LTL model checking for MDPs

- Model check LTL specification $P_{\sim p}$ [ψ] against MDP M
- 1. Convert problem to one needing maximum probabilities
  - e.g. convert $P_{>p}$ [ψ] to $P_{<1-p}$ [¬ψ]
- 2. Generate a DRA for ψ (or ¬ψ)
  - build nondeterministic Büchi automaton (NBA) for ψ [VW94]
  - convert the NBA to a DRA [Saf88]
- 3. Construct product MDP M⊗A
- 4. Identify accepting end components (ECs) of M⊗A
- 5. Compute max. probability of reaching accepting ECs
  - from all states of the D⊗A
- 6. Compare probability for (s, q_s) against p for each s

Product MDP for a DRA

- For an MDP $M = (S, s_{init}, Steps, L)$
- and a (total) DRA $A = (Q, \Sigma, \delta, q_0, Acc)$
  - where $Acc = \{ (L_i, K_i) \mid 1 \leq i \leq k \}$

- The product MDP $M \otimes A$ is:
  - the MDP $(S \times Q, (s_{init}, q_{init}), Steps', L')$ where:

$$q_{init} = \delta(q_0, L(s_{init}))$$

$$Steps'(s,q) = \{ \mu^q \mid \mu \in Step(s) \}$$

$$\mu^q(s', q') = \begin{cases} \mu(s') & \text{if } q' = \delta(q, L(s)) \\ 0 & \text{otherwise} \end{cases}$$

$l_i \in L'(s,q)$ if $q \in L_i$ and $k_i \in L'(s,q)$ if $q \in K_i$

(i.e. state sets of acceptance condition used as labels)

Product MDP for a DRA

- For MDP M and DRA A

$$p_{max}^{M}(s, A) = p_{max}^{M \otimes A}((s,q_s), \vee_{1 \leq i \leq k} (FG \neg l_i \wedge GF k_i))$$

  - where $q_s = \delta(q_0, L(s))$

- Hence:

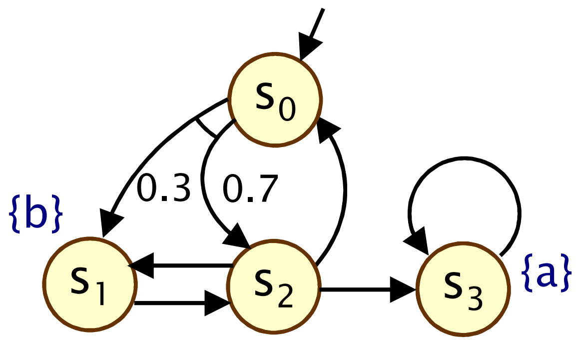
$$p_{max}^{M}(s, A) = p_{max}^{M \otimes A}((s,q_s), F\ T_{Acc})$$

  - where T_{Acc} is the union of all sets T for accepting end components (T,Steps') in D⊗A
  - an accepting end components is such that, for some $1 \leq i \leq k$:
    - $q \vDash \neg l_i$ for all $(s,q) \in T$ and $q \vDash k_i$ for some $(s,q) \in T$
    - i.e. $T \cap (S \times L_i) = \varnothing$ and $T \cap (S \times K_i) \neq \varnothing$

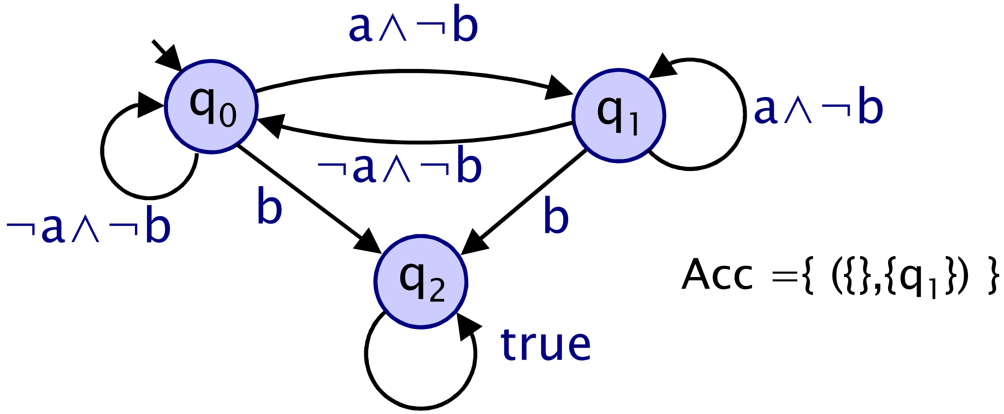
Example: LTL for MDPs

- Model check $P_{<0.8}$ [G ¬b ∧ GF a] for MDP M:
  - need to compute $p_{max}(s_0$, G ¬b ∧ GF a)

MDP M

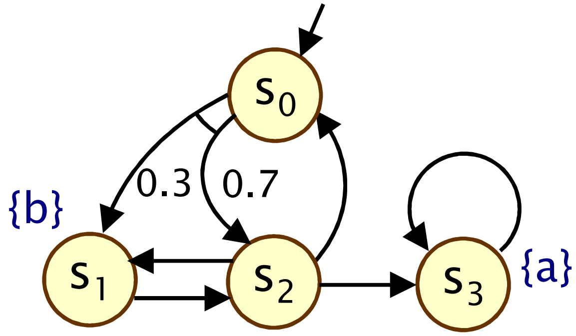


DRA A_ψ for ψ = G¬b ∧ GF a

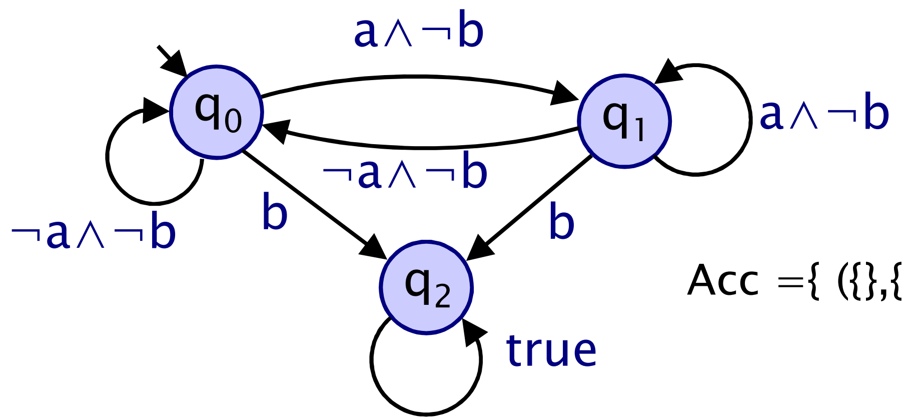


Example: LTL for MDPs

MDP M



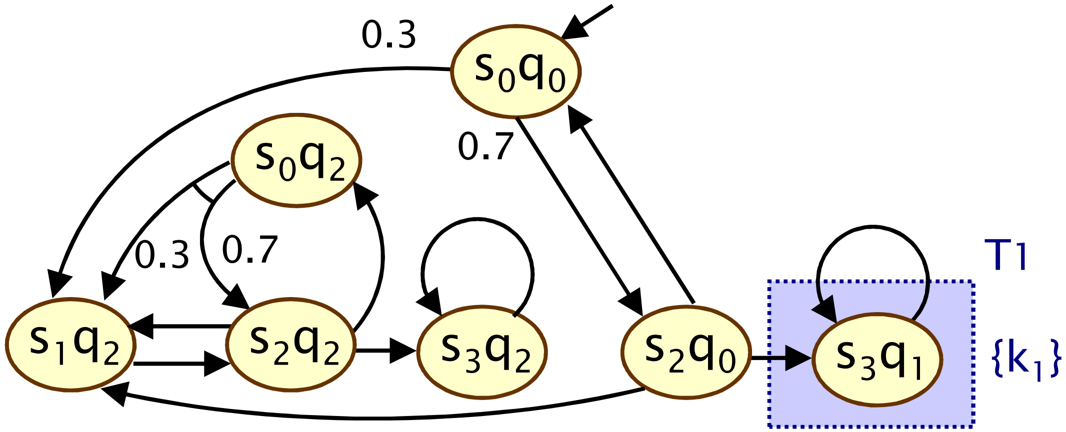
DRA A_ψ for $\psi = G\neg b \wedge GF\ a$



Acc ={ ({},{q_1}) }

Product MDP $M \otimes A_\psi$

$p_{max}^{M}(s_0, \psi) = p_{max}^{M \otimes A\psi}(s_0 q_0, F\ T_1) = 0.7$



LTL model checking for MDPs

- Complexity of model checking LTL formula ψ on MDP M
  - is doubly exponential in |ψ| and polynomial in |M|
  - unlike DTMCs, this cannot be improved upon
- PCTL* model checking
  - LTL model checking can be adapted to PCTL*, as for DTMCs
- Maximal end components
  - can optimise LTL model checking using maximal end components (there may be exponentially many ECs)
- Optimal adversaries for LTL formulae
  - e.g. memoryless adversary always exists for $p_{max}(s, GF\ a)$, but not for $p_{max}(s, FG\ a)$

Summary (LTL model checking)

- Linear temporal logic (LTL)
  - combines path operators; PCTL* subsumes LTL and PCTL
- ω-automata: represent ω-regular languages/properties
  - can translate any LTL formula into a Büchi automaton
  - for deterministic ω-automata, we use Rabin automata
- Long-run properties of DTMCs
  - need bottom strongly connected components (BSCCs)
- LTL model checking for DTMCs
  - construct product of DTMC and Rabin automaton
  - identify accepting BSCCs, compute reachability probability
- LTL model checking for MDPs
  - MDP-DRA product, reachability of accepting end components

PRISM: Recent & new developments

- New features:
  1. parametric model checking
  2. parameter synthesis
  3. strategy synthesis
  4. stochastic multi-player games
  5. real-time: probabilistic timed automata (PTAs)

- Further new additions:
  – enhanced statistical model checking (approximations + confidence intervals, acceptance sampling)
  – efficient CTMC model checking (fast adaptive uniformisation)
  – benchmark suite & testing functionality
  – www.prismmodelchecker.org

– Beyond PRISM…

Beyond MDPs

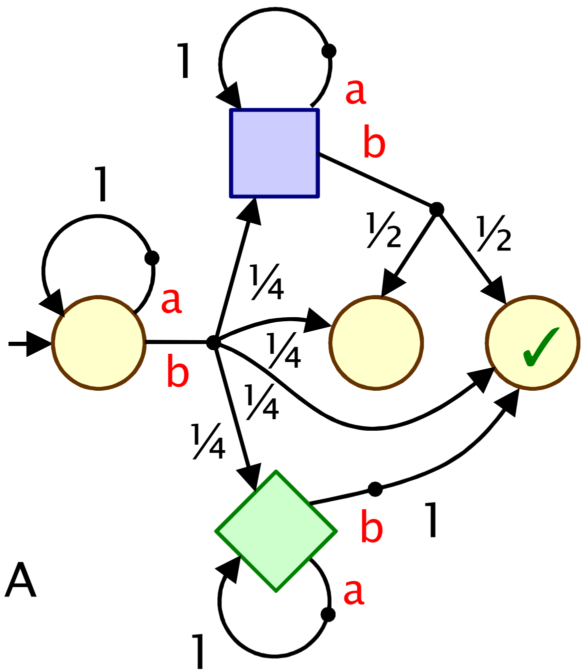
- Markov decision processes (1½ player games)
  - model control in presence of uncertainty
  - strategy/controller synthesis against environment
  - environment is passive
- Many situations where environment is active
  - multi-agent systems, …
- Stochastic multiplayer games
  - N players, each with own strategy, can cooperate or compete
  - stochasticity to model uncertainty
  - verification/synthesis expressed in terms of winning strategies

Stochastic multi-player games

- Stochastic multi-player game (SMGs)
  - probability + nondeterminism + multiple players

- A (turn-based) SMG is a tuple $(\Pi, S, \langle S_i \rangle_{i \in \Pi}, A, \Delta, L)$:
  - Π is a set of n players
  - S is a (finite) set of states
  - $\langle S_i \rangle_{i \in \Pi}$ is a partition of S
  - A is a set of action labels
  - $\Delta : S \times A \rightarrow Dist(S)$ is a (partial) transition probability function
  - $L : S \rightarrow 2^{AP}$ is a labelling with atomic propositions from AP

- Notation:
  - $A(s)$ denotes available actions in state A



Paths, strategies + probabilities

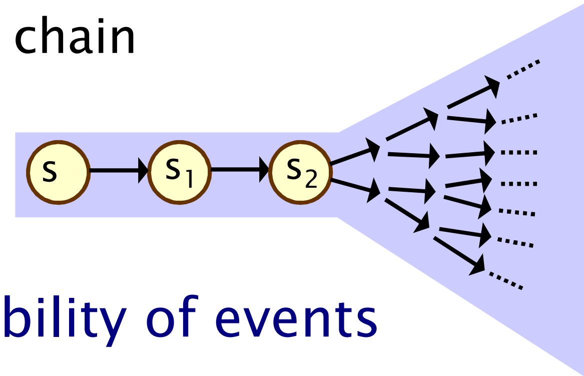
- A path is an (infinite) sequence of connected states in SMG
  - i.e. $s_0 a_0 s_1 a_1 \ldots$ such that $a_i \in A(s_i)$ and $\Delta(s_i,a_i)(s_{i+1})>0$ for all i
  - represents a system execution (i.e. one possible behaviour)
  - to reason formally, need a probability space over paths

- A strategy for player $i \in \Pi$ resolves choices in S_i states
  - based on history of execution so far
  - i.e. a function $\sigma_i : (SA)^*S_i \rightarrow Dist(A)$
  - Σ_i denotes the set of all strategies for player i

- A strategy profile is tuple $\sigma=(\sigma_1,\ldots,\sigma_n)$
  - combining strategies for all n players
  - deterministic if σ always gives a Dirac distribution
  - memoryless if $\sigma(s_0 a_0 \ldots s_k)$ depends only on s_k

Paths, strategies + probabilities…

- For a strategy profile σ:
  - the game's behaviour is fully probabilistic
  - essentially an (infinite-state) Markov chain
  - yields a probability measure Pr_s^σ over set of all paths $Path_s$ from s



- Allows us to reason about the probability of events
  - under a specific strategy profile σ
  - e.g. any (ω-)regular property over states/actions

- Also allows us to define expectation of random variables
  - i.e. measurable functions $X : Path_s \rightarrow \mathbb{R}_{\geq 0}$
  - $E_s^\sigma [X] = \int_{Path_s} X \, dPr_s^\sigma$
  - used to define expected costs/rewards…

Rewards

- Rewards (or costs)
  - real-valued quantities assigned to states (and/or transitions)
- Wide range of possible uses:
  - elapsed time, power consumption, size of message queue, number of messages successfully delivered, net profit, …
- We use:
  - state rewards: $r : S \rightarrow \mathbb{N}$ (but can generalise to $\mathbb{Q}_{\geq 0}$)
  - expected cumulative reward until a target set T is reached
- Allow for modelling e.g.
  - expected time for algorithm execution
  - expected resource usage (energy, messages sent, …)

Property specification: rPATL

- New temporal logic rPATL:
  - reward probabilistic alternating temporal logic

- CTL, extended with:
  - coalition operator $\langle\langle C \rangle\rangle$ of ATL
  - probabilistic operator P of PCTL
  - generalised version of reward operator R from PRISM

- Example:
  - $\langle\langle \{1,2\} \rangle\rangle \, P_{<0.01} \, [\, F^{\leq 10} \, \text{error} \,]$
  - “players 1 and 2 have a strategy to ensure that the probability of an error occurring within 10 steps is less than 0.1, regardless of the strategies of other players”

rPATL syntax

- Syntax:

$$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle C\rangle\rangle P_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle R^{r}_{\bowtie x}\ [F\phi]$$

$$\psi ::= X\ \phi \mid \phi\ U^{\leq k}\ \phi \mid F^{\leq k}\ \phi \mid G^{\leq k}\ \phi$$

- where:
  - $a \in AP$ is an atomic proposition, $C \subseteq \Pi$ is a coalition of players, $\bowtie \in \{\leq,<,>,\geq\}$, $q \in [0,1] \cap \mathbb{Q}$, $x \in \mathbb{Q}_{\geq 0}$, $k \in \mathbb{N} \cup \{\infty\}$
    r is a reward structure
- $\langle\langle C\rangle\rangle P_{\bowtie q}[\psi]$
  - “players in coalition C have a strategy to ensure that the probability of path formula ψ being true satisfies ⋈ q, regardless of the strategies of other players”
- $\langle\langle C\rangle\rangle R^{r}_{\bowtie x}\ [F\phi]$
  - “players in coalition C have a strategy to ensure that the expected reward r to reach a ϕ-state satisfies ⋈ x, regardless of the strategies of other players”

rPATL semantics

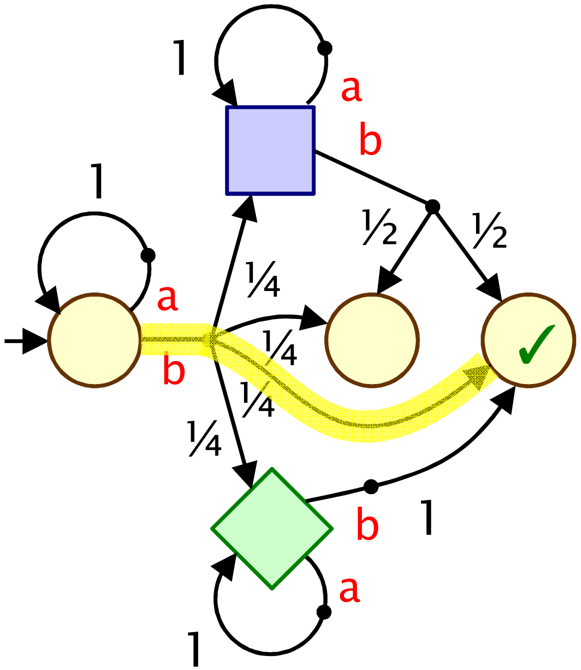
- Semantics for most operators is standard
- Just focus on P and R operators…
  - present using reduction to a stochastic 2-player game
  - (as for later model checking algorithms)

- Coalition game G_C for SMG G and coalition $C \subseteq \Pi$
  - 2-player SMG where C and $\Pi \backslash C$ collapse to players 1 and 2

- $\langle\langle C \rangle\rangle P_{\bowtie q}[\psi]$ is true in state s of G iff:
  - in coalition game G_C:
  - $\exists \sigma_1 \in \Sigma_1$ such that $\forall \sigma_2 \in \Sigma_2$. $Pr_s^{\sigma_1, \sigma_2}(\psi) \bowtie q$

- Semantics for R operator defined similarly…

Examples



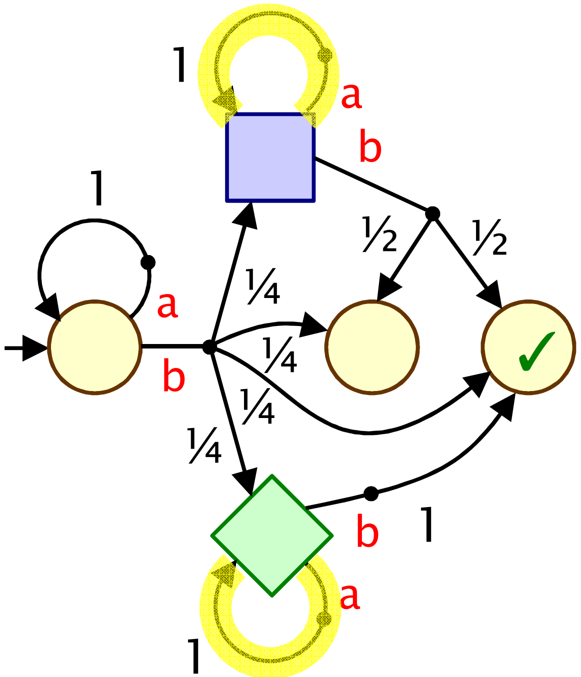
$\langle\langle \bigcirc \rangle\rangle P_{\geq 1/4}[\ F\ ✓\ ]$

true in initial state

$\langle\langle \bigcirc \rangle\rangle P_{\geq 1/3}[\ F\ ✓\ ]$

$\langle\langle \bigcirc, \square \rangle\rangle P_{\geq 1/3}[\ F\ ✓\ ]$

Examples



$\langle\langle$○$\rangle\rangle P_{\geq 1/4}[\ F$ ✓ $]$

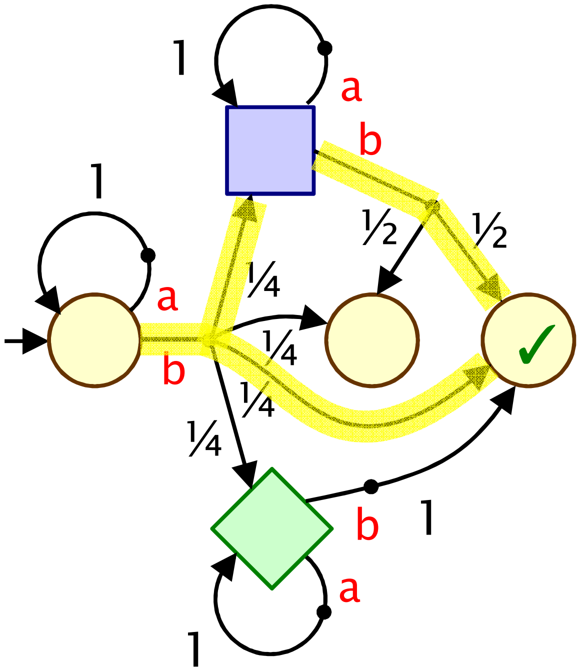
true in initial state

$\langle\langle$○$\rangle\rangle P_{\geq 1/3}[\ F$ ✓ $]$

false in initial state

$\langle\langle$○, □$\rangle\rangle P_{\geq 1/3}[\ F$ ✓ $]$

Examples



$\langle\langle$○$\rangle\rangle P_{\geq 1/4}$[F ✓]

true in initial state

$\langle\langle$○$\rangle\rangle P_{\geq 1/3}$[F ✓]

false in initial state

$\langle\langle$○, □$\rangle\rangle P_{\geq 1/3}$[F ✓]

true in initial state

Model checking rPATL

- Basic algorithm: as for any branching-time temporal logic
  - as for CTL, build and traverse the parse tree of the formula
  - compute $Sat(\phi) = \{ s \in S \mid s \vDash \phi \}$ for each subformula ϕ
- Main task: checking P and R operators
  - reduction to solution of stochastic 2-player game G_C
  - e.g. $\langle\langle C \rangle\rangle P_{\geq q}[\psi] \;\Leftrightarrow\; \sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(\psi) \geq q$
  - complexity: NP ∩ coNP (for subclass), o'wise NEXP ∩ coNEXP
  - compared to, e.g. P for Markov decision processes
- In practice though:
  - evaluation of numerical fixed points ("value iteration")
  - up to a desired level of convergence

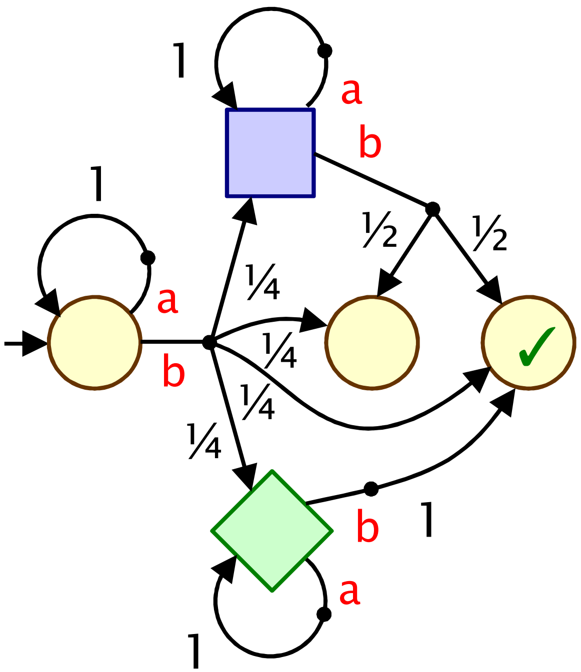
Probabilities for P operator

- E.g. $\langle\langle C\rangle\rangle P_{\geq q}[\ F\ \phi\ ]$: max/min reachability probabilities
  - compute $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}(F\ \phi)$ for all states s
  - deterministic memoryless strategies suffice
- Value is:
  - 1 if $s \in Sat(\phi)$, and otherwise least fixed point of:

$$f(s) = \begin{cases} \max_{a \in A(s)} \left(\sum_{s' \in S} \Delta(s,a)(s') \cdot f(s') \right) & \text{if } s \in S_1 \\ \min_{a \in A(s)} \left(\sum_{s' \in S} \Delta(s,a)(s') \cdot f(s') \right) & \text{if } s \in S_2 \end{cases}$$

- Computation:
  - start from zero, propagate probabilities backwards
  - guaranteed to converge

Example



rPATL: ⟨⟨○,□⟩⟩$P_{\geq 1/3}$ [F ✓]

Player 1: ○,□   Player 2: ◇

Compute: $\sup_{\sigma_1 \in \Sigma_1} \inf_{\sigma_2 \in \Sigma_2} Pr_s^{\sigma_1,\sigma_2}$ (F ✓)

Tool support: PRISM-games

- Prototype model checker for stochastic games
  - integrated into PRISM model checker
  - using new explicit-state model checking engine

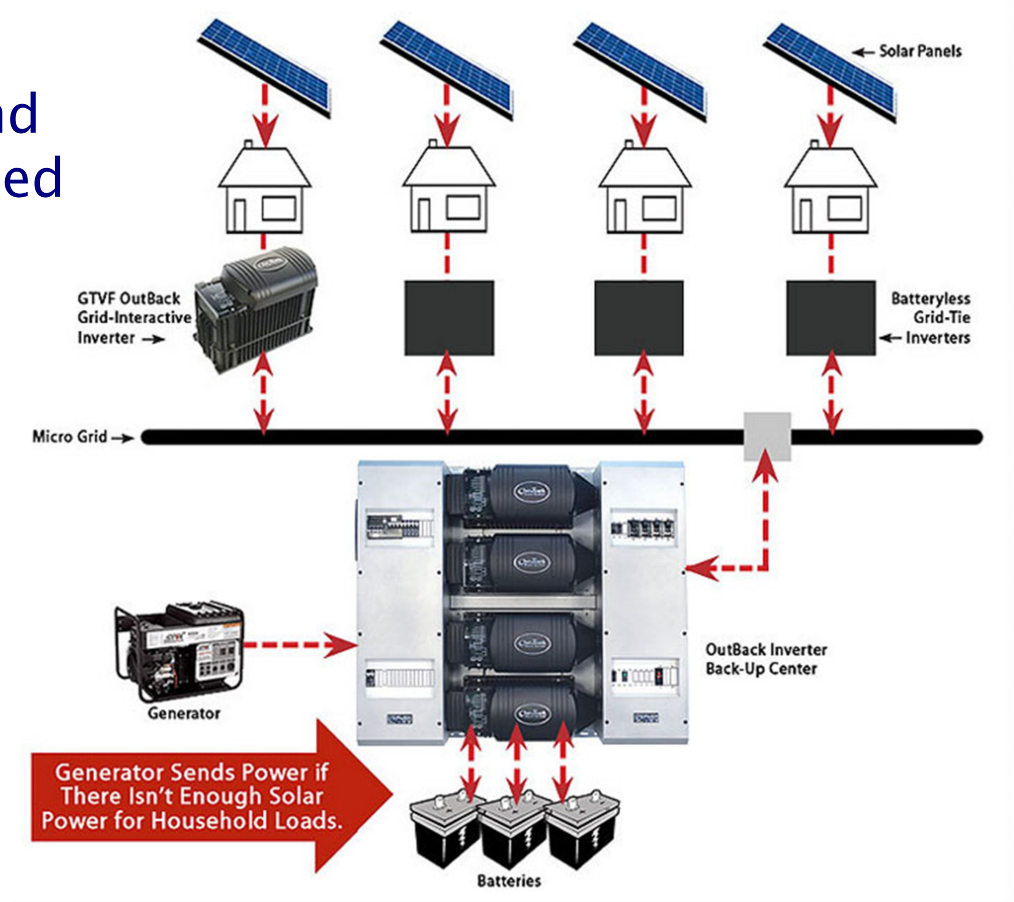
- SMGs added to PRISM modelling language
  - guarded command language, based on Reactive modules
  - finite data types, parallel composition, proc. algebra op.s, …

- rPATL added to PRISM property specification language
  - implemented value iteration based model checking

- Available now:
  - http://www.prismmodelchecker.org/games/

Case study: Smartgrid

- Microgrid: proposed model for future energy markets
  - localised energy management
- Neighbourhoods use and store electricity generated from local sources
  - wind, solar, …
- Needs: demand-side management
  - active management of demand by users
  - to avoid peaks

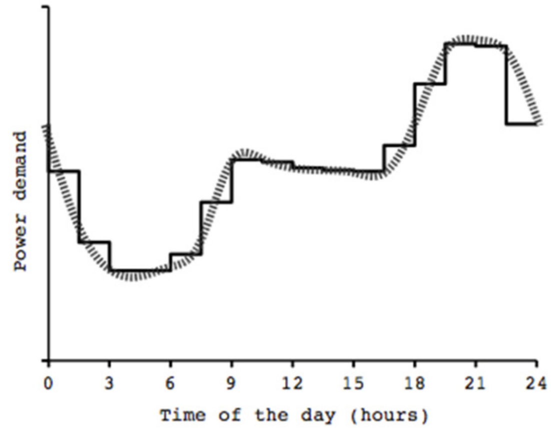


Microgrid demand-side management

- Demand-side management algorithm [Hildmann/Saffre'11]
  - N households, connected to a distribution manager
  - households submit loads for execution
  - load submission probability: daily demand curve
  - load duration: random, between 1 and D steps
  - execution cost/step = number of currently running loads
- Simple algorithm:
  - upon load generation, if cost is below an agreed limit c_{lim}, execute it, otherwise only execute with probability P_{start}
- Analysis of [Hildmann/Saffre'11]
  - define household value as V=loads_executing/execution_cost
  - simulation-based analysis shows reduction in peak demand and total energy cost reduced, with good expected value V
  - (if all households stick to algorithm)

Microgrid demand-side management

- The model
  - SMG with N players (one per household)
  - analyse 3-day period, using piecewise approximation of daily demand curve
  - fix parameters D=4, c_{lim}=1.5
  - add rewards structure for value V



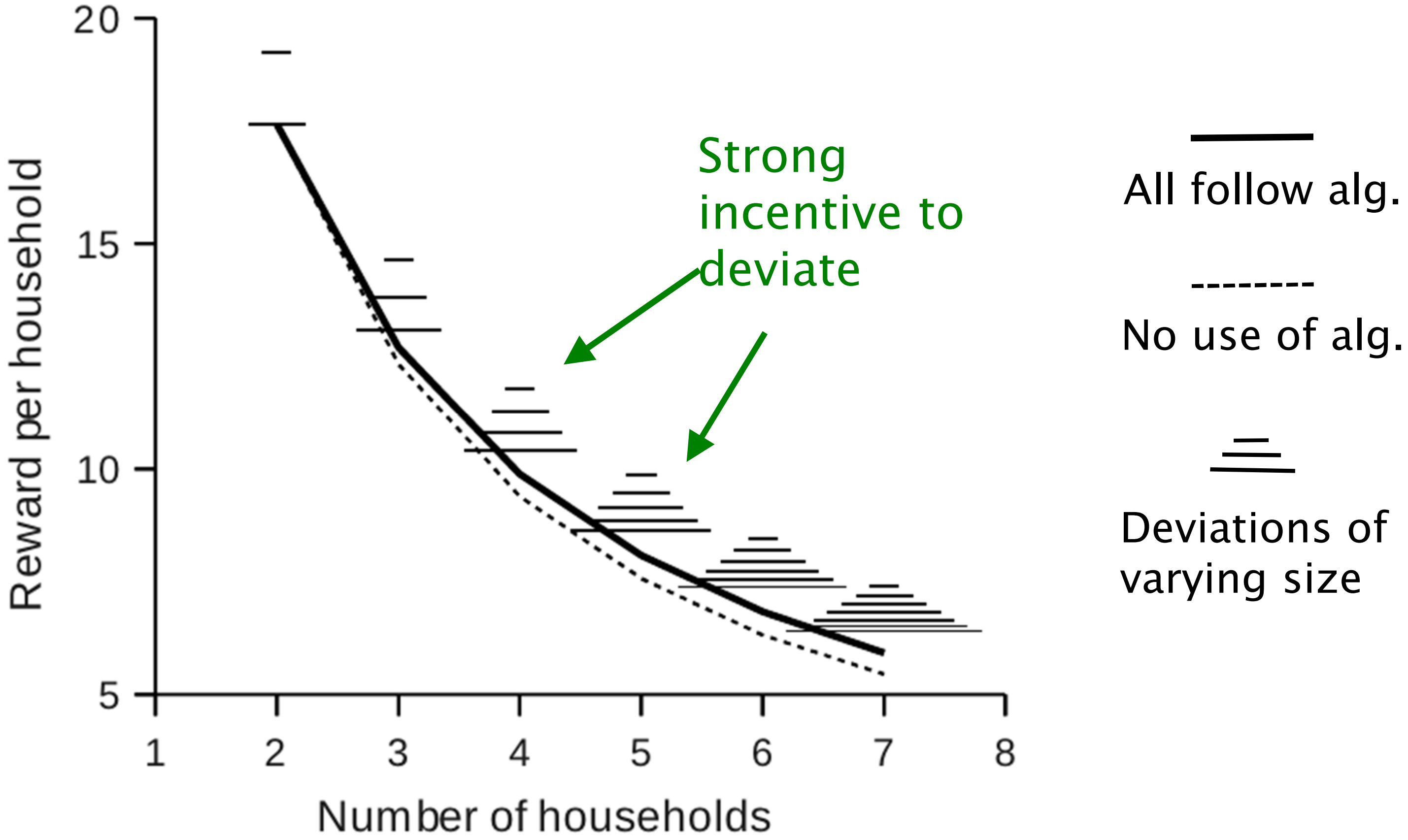
- Built/analysed models
  - for N=2,…,7 households

N	States	Transitions
5	743,904	2,145,120
6	2,384,369	7,260,756
7	6,241,312	19,678,246

- Step 1: assume all households follow algorithm of [HS'11] (MDP)
  - obtain optimal value for P_{start}
- Step 2: introduce competitive behaviour (SMG)
  - allow coalition C of households to deviate from algorithm

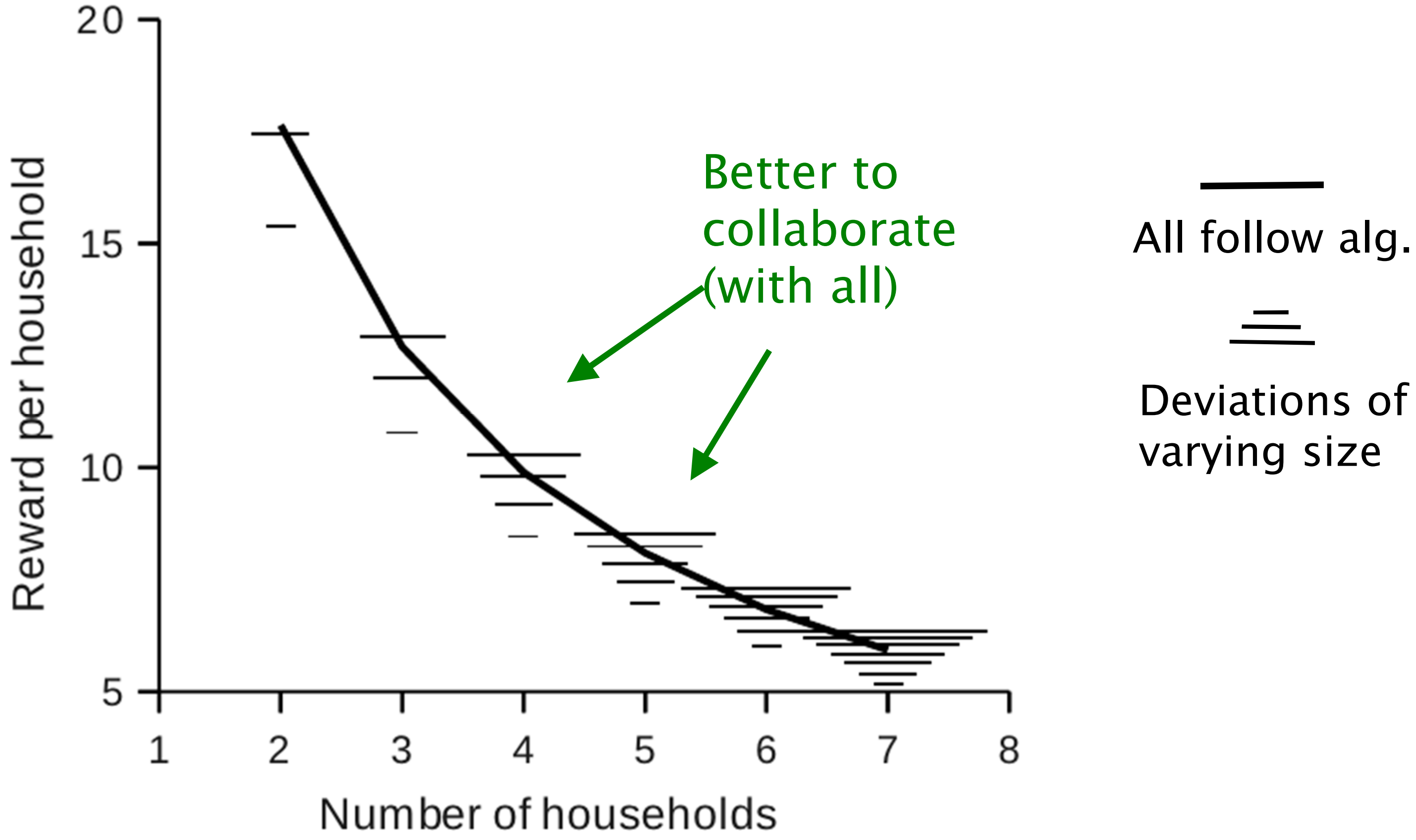
Results: Competitive behaviour

- Expected total value V per household
  - in rPATL: $\langle\langle C\rangle\rangle R^{r_C}_{max=?}$ [F time=max time] / |C|
  - where r_C is combined rewards for coalition C



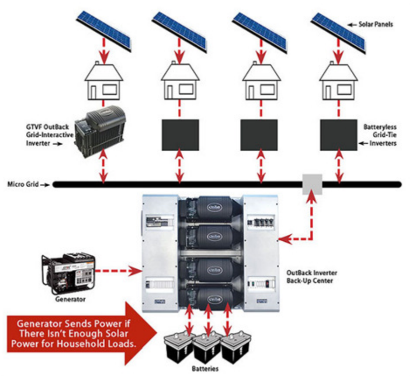
Results: Competitive behaviour

- Algorithm fix: simple punishment mechanism
  - distribution manager can cancel some loads exceeding c_{lim}

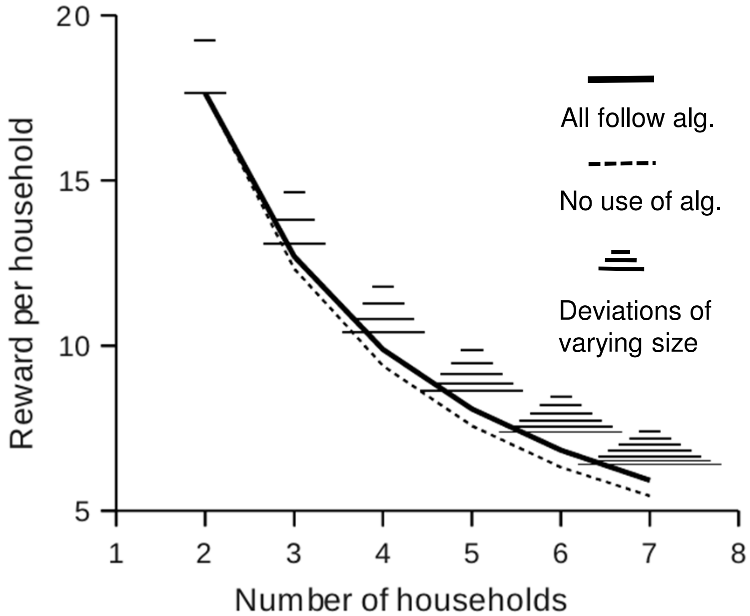


Case study: Energy management

- Energy management protocol for Microgrid
  - Microgrid: local energy management
  - randomised demand management protocol [Hildmann/Saffre'11]
  - probability: randomisation, demand model, …

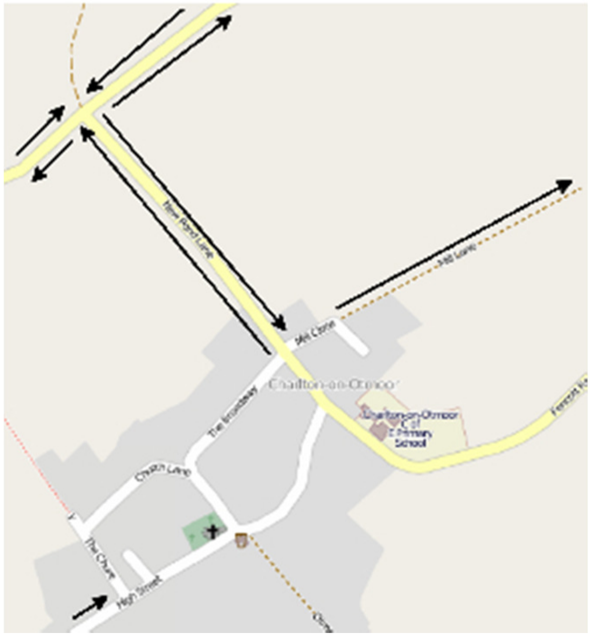


- Existing analysis
  - simulation-based
  - assumes all clients are unselfish
- Our analysis
  - stochastic multi-player game
  - clients can cheat (and cooperate)
  - exposes protocol weakness
  - propose/verify simple fix



Case study: Autonomous urban driving

- Inspired by DARPA challenge
  - represent map data as a stochastic game, with environment able to select hazards
  - express goals as conjunctions of probabilistic and reward properties
  - e.g. “maximise probability of avoiding hazards and minimise time to reach destination”



- Solution (PRISM-games)
  - synthesise a probabilistic strategy to achieve the multi-objective goal
  - enable the exploration of trade-offs between subgoals
- Applied to synthesise driving strategies for English villages
  - being developed in PRISM-games

Summary (Games)

- What has been achieved so far
  - extended probabilistic verification to stochastic multi-player games
  - compositional strategy synthesis from multiobjective specifications under development
  - new temporal logic rPATL for property specification
  - rPATL model checking algorithm based on num. fixed points
  - prototype model checker PRISM-games
  - case studies
- Future work
  - more realistic classes of strategy, e.g. partial information
  - new application areas, security, randomised algorithms, …
- Next: Probabilistic timed automata (PTAs)