Advances in Probabilistic Model Checking

Marta Kwiatkowska

Department of Computer Science, University of Oxford

Marktoberdorf, August 2011

Introduction

Probabilistic model checking

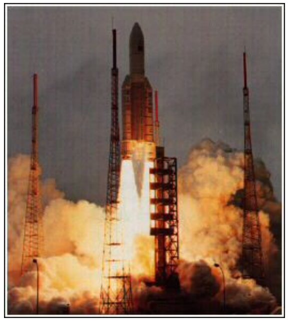
What is probabilistic model checking?

- Probabilistic model checking…
  - is a formal verification technique
    for modelling and analysing systems
    that exhibit probabilistic behaviour

- Formal verification…
  - is the application of rigorous,
    mathematics-based techniques
    to establish the correctness
    of computerised systems

Why formal verification?

- Errors in computerised systems can be costly…



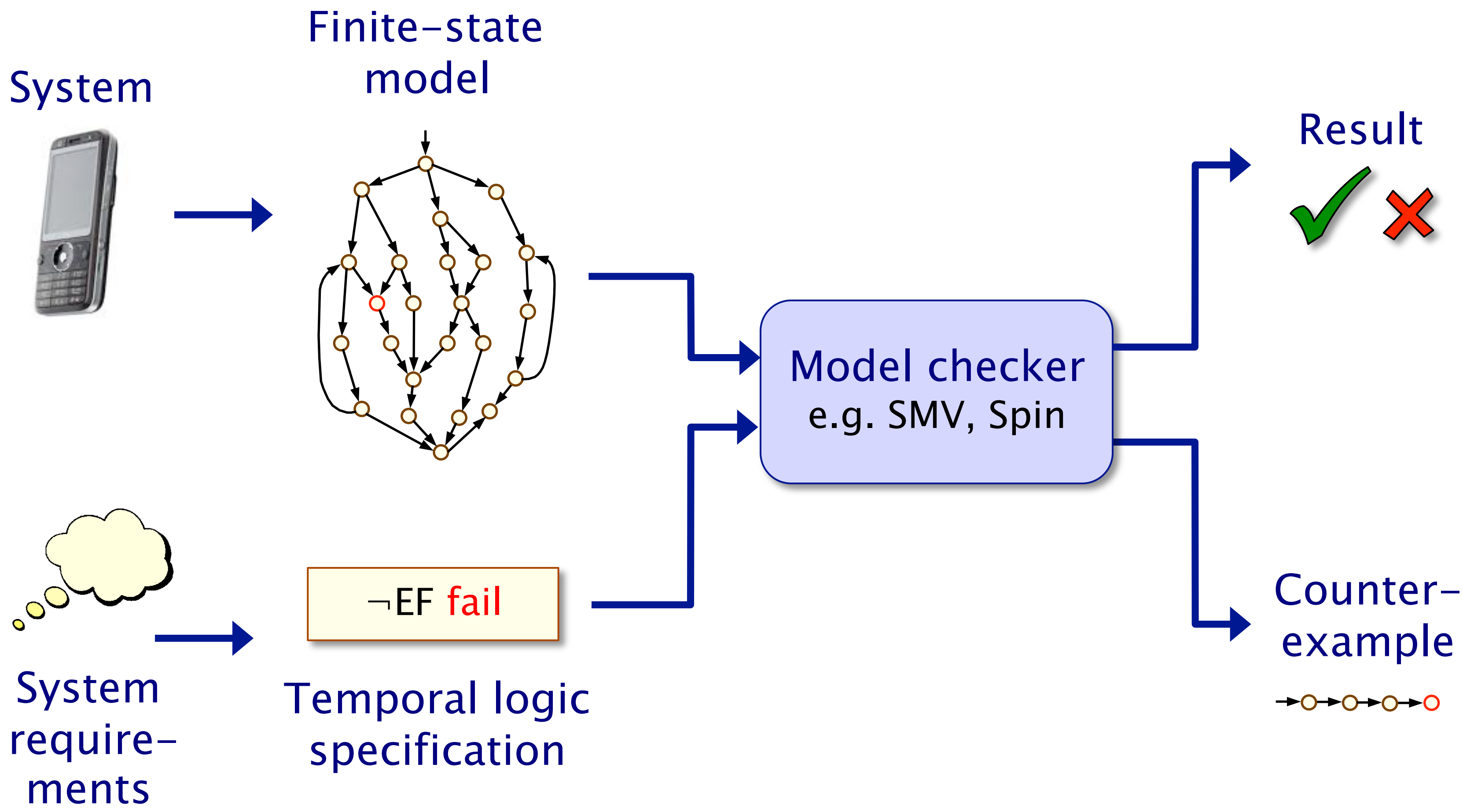
Pentium chip (1994)
Bug found in FPU.
Intel (eventually) offers to replace faulty chips.
Estimated loss: $475m

Ariane 5 (1996)
Self-destructs 37secs into maiden launch.
Cause: uncaught overflow exception.

Toyota Prius (2010)
Software "glitch" found in anti-lock braking system.
185,000 cars recalled.

- Why verify?
  - "Testing can only show the presence of errors, not their absence." [Edsger Dijstra]

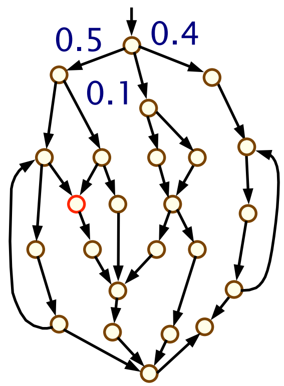
Model checking



Probabilistic model checking

System

Probabilistic model
e.g. Markov chain



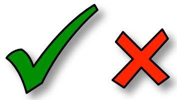
System require–ments

$P_{<0.1}$ [F fail]

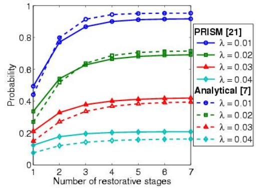
Probabilistic temporal logic specification
e.g. PCTL, CSL, LTL

Probabilistic model checker
e.g. PRISM

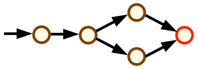
Result



Quantitative results



Counter–example



Why probability?

- Some systems are inherently probabilistic…

- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding

- Examples: real-world protocols featuring randomisation:
  - Randomised back-off schemes
    - CSMA protocol, 802.11 Wireless LAN
  - Random choice of waiting time
    - IEEE1394 Firewire (root contention), Bluetooth (device discovery)
  - Random choice over a set of possible addresses
    - IPv4 Zeroconf dynamic configuration (link-local addressing)
  - Randomised algorithms for anonymity, contract signing, …

Why probability?

- Some systems are inherently probabilistic…
- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding
- To model uncertainty and performance
  - to quantify rate of failures, express Quality of Service
- Examples:
  - computer networks, embedded systems
  - power management policies
  - nano-scale circuitry: reliability through defect-tolerance

Why probability?

- Some systems are inherently probabilistic…
- Randomisation, e.g. in distributed coordination algorithms
  - as a symmetry breaker, in gossip routing to reduce flooding
- To model uncertainty and performance
  - to quantify rate of failures, express Quality of Service
- To model biological processes
  - reactions occurring between large numbers of molecules are naturally modelled in a stochastic fashion

Verifying probabilistic systems

- We are not just interested in correctness

- We want to be able to quantify:
  - security, privacy, trust, anonymity, fairness
  - safety, reliability, performance, dependability
  - resource usage, e.g. battery life
  - and much more…

- Quantitative, as well as qualitative requirements:
  - how reliable is my car's Bluetooth network?
  - how efficient is my phone's power management policy?
  - is my bank's web-service secure?
  - what is the expected long-run percentage of protein X?

Probabilistic models

	Fully probabilistic	Nondeterministic
Discrete time	Discrete-time Markov chains (DTMCs)	Markov decision processes (MDPs) (probabilistic automata)
Continuous time	Continuous-time Markov chains (CTMCs)	CTMDPs/IMCs
		Probabilistic timed automata (PTAs)

Overview

- Lecture 1
  - Introduction
  - 1 – Discrete time Markov chains
  - 2 – Markov decision processes
  - 3 – Compositional probabilistic verification
  - 4 – Probabilistic timed automata
- Course materials available here:
  - http://www.prismmodelchecker.org/courses/marktoberdorf11/
  - lecture slides, reference list, exercises

Part 1

Discrete-time Markov chains

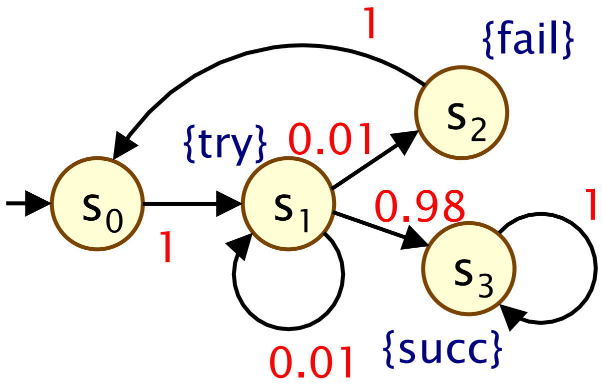
Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
- Case study: Bluetooth device discovery

Discrete-time Markov chains

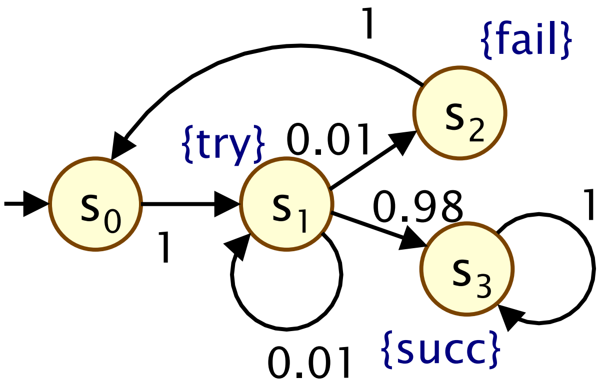
- Discrete-time Markov chains (DTMCs)
  - state-transition systems augmented with probabilities

- States
  - discrete set of states representing possible configurations of the system being modelled
- Transitions
  - transitions between states occur in discrete time-steps
- Probabilities
  - probability of making transitions between states is given by discrete probability distributions



Discrete-time Markov chains

- Formally, a DTMC D is a tuple $(S, s_{init}, \mathbf{P}, L)$ where:
  - S is a finite set of states ("state space")
  - $s_{init} \in S$ is the initial state
  - $\mathbf{P} : S \times S \to [0,1]$ is the transition probability matrix where $\Sigma_{s' \in S}\, \mathbf{P}(s,s') = 1$ for all $s \in S$
  - $L : S \to 2^{AP}$ is function labelling states with atomic propositions
- Note: no deadlock states
  - i.e. every state has at least one outgoing transition
  - can add self loops to represent final/terminating states



DTMCs: An alternative definition

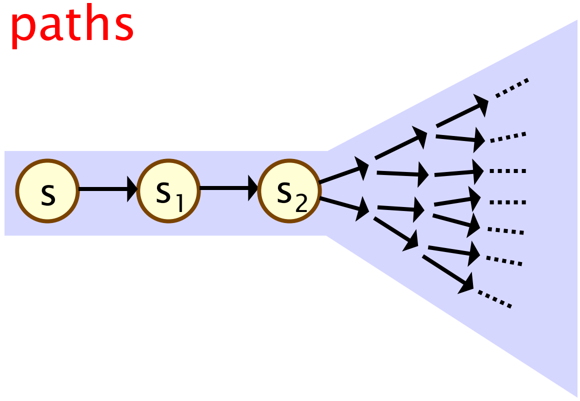
- Alternative definition: a DTMC is:
  - a family of random variables $\{ X(k) \mid k=0,1,2,\ldots \}$
  - $X(k)$ are observations at discrete time-steps
  - i.e. $X(k)$ is the state of the system at time-step k

- Memorylessness (Markov property)
  - $\Pr(X(k)=s_k \mid X(k-1)=s_{k-1}, \ldots , X(0)=s_0)$
    $= \Pr(X(k)=s_k \mid X(k-1)=s_{k-1})$

- We consider homogenous DTMCs
  - transition probabilities are independent of time
  - $P(s_{k-1},s_k) = \Pr(X(k)=s_k \mid X(k-1)=s_{k-1})$

Paths and probabilities

- A (finite or infinite) path through a DTMC
  - is a sequence of states $s_0 s_1 s_2 s_3 \ldots$ such that $\mathbf{P}(s_i, s_{i+1}) > 0 \ \forall i$
  - represents an execution (i.e. one possible behaviour) of the system which the DTMC is modelling
- To reason (quantitatively) about this system
  - need to define a probability space over paths
- Intuitively:
  - sample space: Path(s) = set of all infinite paths from a state s
  - events: sets of infinite paths from s
  - basic events: cylinder sets (or "cones")
  - cylinder set $C(\omega)$, for a finite path ω = set of infinite paths with the common finite prefix ω
  - for example: $C(ss_1s_2)$



Probability spaces

- Let Ω be an arbitrary non-empty set
- A σ-algebra (or σ-field) on Ω is a family Σ of subsets of Ω closed under complementation and countable union, i.e.:
  - if $A \in \Sigma$, the complement $\Omega \setminus A$ is in Σ
  - if $A_i \in \Sigma$ for $i \in \mathbb{N}$, the union $\cup_i A_i$ is in Σ
  - the empty set $\varnothing$ is in Σ
- Theorem: For any family F of subsets of Ω, there exists a unique smallest σ-algebra on Ω containing F
- Probability space (Ω, Σ, Pr)
  - Ω is the sample space
  - Σ is the set of events: σ-algebra on Ω
  - $Pr : \Sigma \to [0,1]$ is the probability measure:

    $Pr(\Omega) = 1$ and $Pr(\cup_i A_i) = \Sigma_i Pr(A_i)$ for countable disjoint A_i

Probability space over paths

- Sample space Ω = Path(s)

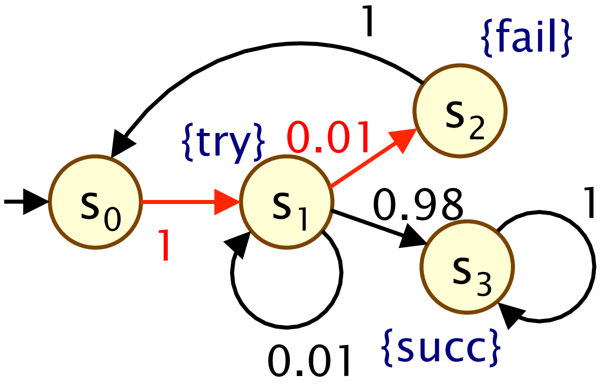
  set of infinite paths with initial state s
- Event set $\Sigma_{Path(s)}$
  - the cylinder set $C(\omega) = \{ \omega' \in Path(s) \mid \omega \text{ is prefix of } \omega' \}$
  - $\Sigma_{Path(s)}$ is the least σ-algebra on Path(s) containing C(ω) for all finite paths ω starting in s
- Probability measure Pr_s
  - define probability $\mathbf{P}_s(\omega)$ for finite path $\omega = ss_1 \ldots s_n$ as:
    - $\mathbf{P}_s(\omega) = 1$ if ω has length one (i.e. ω = s)
    - $\mathbf{P}_s(\omega) = \mathbf{P}(s,s_1) \cdot \ldots \cdot \mathbf{P}(s_{n-1},s_n)$ otherwise
    - define $Pr_s(C(\omega)) = \mathbf{P}_s(\omega)$ for all finite paths ω
  - Pr_s extends uniquely to a probability measure $Pr_s:\Sigma_{Path(s)} \rightarrow [0,1]$

- See [KSK76] for further details

Probability space – Example

- Paths where sending fails the first time
  - $\omega = s_0 s_1 s_2$
  - $C(\omega)$ = all paths starting $s_0 s_1 s_2 \ldots$
  - $\mathbf{P}_{s0}(\omega) = \mathbf{P}(s_0,s_1) \cdot \mathbf{P}(s_1,s_2)$

    $= 1 \cdot 0.01 = 0.01$
  - $Pr_{s0}(C(\omega)) = \mathbf{P}_{s0}(\omega) = 0.01$



- Paths which are eventually successful and with no failures
  - $C(s_0 s_1 s_3) \cup C(s_0 s_1 s_1 s_3) \cup C(s_0 s_1 s_1 s_1 s_3) \cup \ldots$
  - $Pr_{s0}(\, C(s_0 s_1 s_3) \cup C(s_0 s_1 s_1 s_3) \cup C(s_0 s_1 s_1 s_1 s_3) \cup \ldots \,)$

    $= \mathbf{P}_{s0}(s_0 s_1 s_3) + \mathbf{P}_{s0}(s_0 s_1 s_1 s_3) + \mathbf{P}_{s0}(s_0 s_1 s_1 s_1 s_3) + \ldots$

    $= 1 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.01 \cdot 0.98 + \ldots$

    $= 0.9898989898\ldots$

    $= 98/99$

Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
- Case study: Bluetooth device discovery

PCTL

- Temporal logic for describing properties of DTMCs
  - PCTL = Probabilistic Computation Tree Logic [HJ94]
  - essentially the same as the logic pCTL of [ASB+95]

- Extension of (non-probabilistic) temporal logic CTL
  - key addition is probabilistic operator P
  - quantitative extension of CTL's A and E operators

- Example
  - send $\rightarrow$ $P_{\geq 0.95}$ [true $U^{\leq 10}$ deliver]
  - "if a message is sent, then the probability of it being delivered within 10 steps is at least 0.95"

PCTL syntax

ψ is true with probability ~p

- PCTL syntax:

  – ϕ ::= true | a | $\phi \wedge \phi$ | $\neg\phi$ | $P_{\sim p}$ [ψ]　　(state formulas)

  – ψ ::= X ϕ | ϕ $U^{\leq k}$ ϕ | ϕ U ϕ　　(path formulas)

  "next"　"bounded until"　"until"

  – where a is an atomic proposition, used to identify states of interest, $p \in [0,1]$ is a probability, $\sim \in \{<,>,\leq,\geq\}$, $k \in \mathbb{N}$

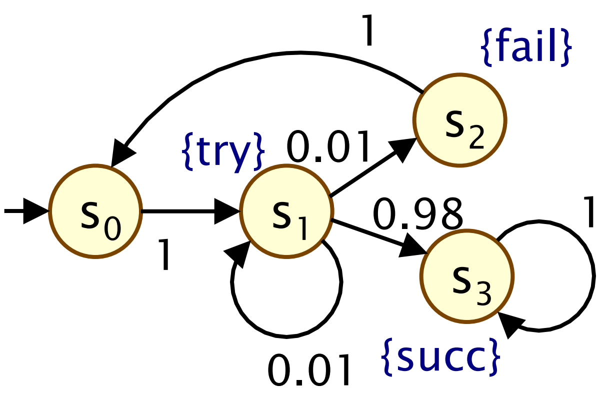
- A PCTL formula is always a state formula
  – path formulas only occur inside the P operator

PCTL semantics for DTMCs

- PCTL formulas interpreted over states of a DTMC
  - $s \vDash \phi$ denotes ϕ is "true in state s" or "satisfied in state s"

- Semantics of (non-probabilistic) state formulas:
  - for a state s of the DTMC $(S,s_{init},\mathbf{P},L)$:
  - $s \vDash a \quad \Leftrightarrow \quad a \in L(s)$
  - $s \vDash \phi_1 \wedge \phi_2 \quad \Leftrightarrow \quad s \vDash \phi_1$ and $s \vDash \phi_2$
  - $s \vDash \neg\phi \quad \Leftrightarrow \quad s \vDash \phi$ is false

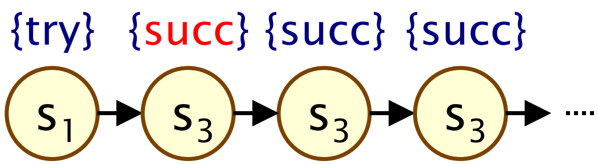
- Examples
  - $s_3 \vDash succ$
  - $s_1 \vDash try \wedge \neg fail$



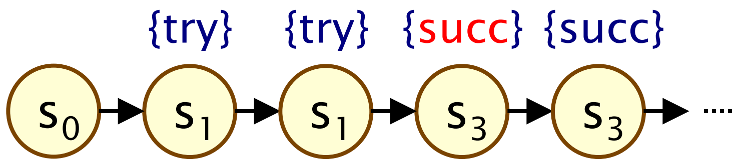
PCTL semantics for DTMCs

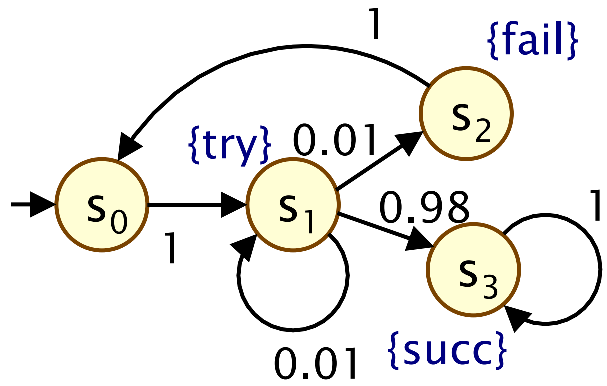
- Semantics of path formulas:
  - for a path $\omega = s_0 s_1 s_2 \ldots$ in the DTMC:
  - $\omega \vDash X\ \phi \quad \Leftrightarrow \quad s_1 \vDash \phi$
  - $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2 \quad \Leftrightarrow \quad \exists i \leq k$ such that $s_i \vDash \phi_2$ and $\forall j < i,\ s_j \vDash \phi_1$
  - $\omega \vDash \phi_1\ U\ \phi_2 \quad \Leftrightarrow \quad \exists k \geq 0$ such that $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2$

- Some examples of satisfying paths:
  - X succ



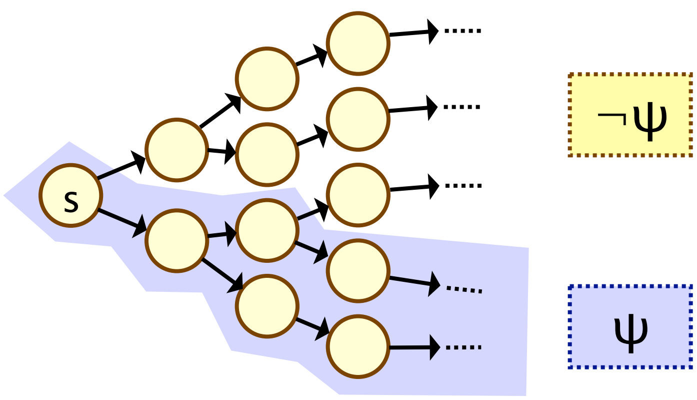
  - ¬fail U succ





PCTL semantics for DTMCs

- Semantics of the probabilistic operator P
  - informal definition: $s \models P_{\sim p} [\ \psi\ ]$ means that "the probability, from state s, that ψ is true for an outgoing path satisfies ~p"
  - example: $s \models P_{<0.25} [\ X\ fail\ ] \Leftrightarrow$ "the probability of atomic proposition fail being true in the next state of outgoing paths from s is less than 0.25"
  - formally: $s \models P_{\sim p} [\psi] \Leftrightarrow Prob(s, \psi) \sim p$
  - where: $Prob(s, \psi) = Pr_s \{ \omega \in Path(s) \mid \omega \models \psi \}$
  - (sets of paths satisfying ψ are always measurable [Var85])



$Prob(s, \psi) \sim p$?

More PCTL…

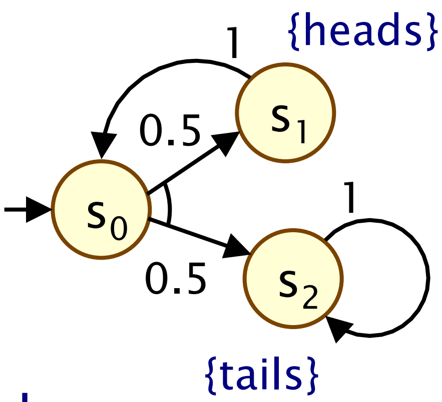
- Usual temporal logic equivalences:
  - false $\equiv$ $\neg$true (false)
  - $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$ (disjunction)
  - $\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$ (implication)
  - $F\ \phi \equiv \Diamond\ \phi \equiv \text{true}\ U\ \phi$ (eventually, “future”)
  - $G\ \phi \equiv \Box\ \phi \equiv \neg(F\ \neg\phi)$ (always, “globally”)
  - bounded variants: $F^{\leq k}\ \phi$, $G^{\leq k}\ \phi$

- Negation and probabilities
  - e.g. $\neg P_{>p}\ [\ \phi_1\ U\ \phi_2\ ] \equiv P_{\leq p}\ [\phi_1\ U\ \phi_2\ ]$
  - e.g. $P_{>p}\ [\ G\ \phi\ ] \equiv P_{<1-p}\ [\ F\ \neg\phi\ ]$

Qualitative vs. quantitative properties

- P operator of PCTL can be seen as a quantitative analogue of the CTL operators A (for all) and E (there exists)

- A PCTL property $P_{\sim p}$ [ψ] is…
  - qualitative when p is either 0 or 1
  - quantitative when p is in the range (0,1)



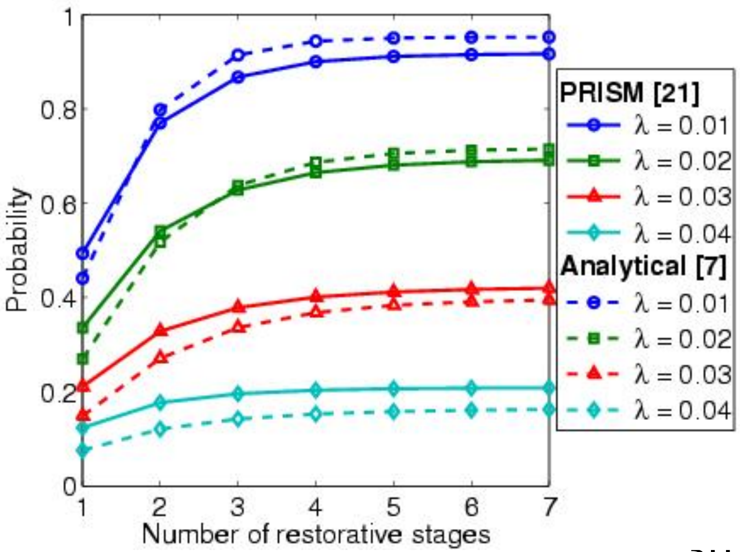
- $P_{>0}$ [F ϕ] is identical to EF ϕ
  - there exists a finite path to a ϕ-state

- $P_{\geq 1}$ [F ϕ] is (similar to but) weaker than AF ϕ
  - e.g. AF "tails" (CTL) $\neq$ $P_{\geq 1}$ [F "tails"] (PCTL)

Quantitative properties

- Consider a PCTL formula $P_{\sim p}$ [ψ]
  - if the probability is unknown, how to choose the bound p?
- When the outermost operator of a PTCL formula is P
  - we allow the form $P_{=?}$ [ψ]
  - "what is the probability that path formula ψ is true?"
- Model checking is no harder: compute the values anyway
- Useful to spot patterns, trends

- Example
  - $P_{=?}$ [F err/total>0.1]
  - "what is the probability that 10% of the NAND gate outputs are erroneous?"



Some real PCTL examples

reliability

- NAND multiplexing system
  - $P_{=?}$ [F err/total>0.1]
  - "what is the probability that 10% of the NAND gate outputs are erroneous?"

performance

- Bluetooth wireless communication protocol
  - $P_{=?}$ [$F^{\leq t}$ reply_count=k]
  - "what is the probability that the sender has received k acknowledgements within t clock-ticks?"

fairness

- Security: EGL contract signing protocol
  - $P_{=?}$ [F (pairs_a=0 & pairs_b>0)]
  - "what is the probability that the party B gains an unfair advantage during the execution of the protocol?"

Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
- Case study: Bluetooth device discovery

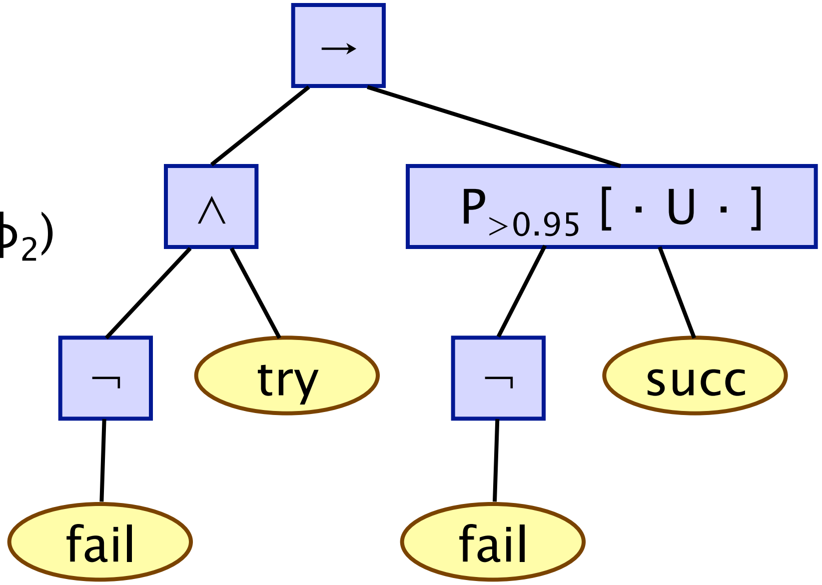
PCTL model checking for DTMCs

- Algorithm for PCTL model checking [CY88,HJ94,CY95]
  - inputs: DTMC $D=(S,s_{init},P,L)$, PCTL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \vDash \phi \}$ = set of states satisfying ϕ
- What does it mean for a DTMC D to satisfy a formula ϕ?
  - sometimes, want to check that $s \vDash \phi \ \forall \ s \in S$, i.e. $Sat(\phi) = S$
  - sometimes, just want to know if $s_{init} \vDash \phi$, i.e. if $s_{init} \in Sat(\phi)$
- Sometimes, focus on quantitative results
  - e.g. compute result of P=? [F error]
  - e.g. compute result of P=? [$F^{\leq k}$ error] for $0 \leq k \leq 100$

PCTL model checking for DTMCs

- Basic algorithm proceeds by induction on parse tree of ϕ
  - example: $\phi = (\neg \text{fail} \wedge \text{try}) \rightarrow P_{>0.95} [\, \neg \text{fail}\ U\ \text{succ}\,]$

- For the non-probabilistic operators:
  - $\text{Sat}(\text{true}) = S$
  - $\text{Sat}(a) = \{ s \in S \mid a \in L(s) \}$
  - $\text{Sat}(\neg\phi) = S \setminus \text{Sat}(\phi)$
  - $\text{Sat}(\phi_1 \wedge \phi_2) = \text{Sat}(\phi_1) \cap \text{Sat}(\phi_2)$



- For the $P_{\sim p} [\, \psi\,]$ operator
  - need to compute the probabilities Prob(s, ψ) for all states $s \in S$
  - focus here on "until" case: $\psi = \phi_1\ U\ \phi_2$

PCTL until for DTMCs

- Computation of probabilities $Prob(s, \phi_1 \text{ U } \phi_2)$ for all $s \in S$
- First, identify all states where the probability is 1 or 0
  - $S^{yes} = Sat(P_{\geq 1} [\ \phi_1 \text{ U } \phi_2\ ])$
  - $S^{no} = Sat(P_{\leq 0} [\ \phi_1 \text{ U } \phi_2\ ])$
- Then solve linear equation system for remaining states

- We refer to the first phase as "precomputation"
  - two algorithms: Prob0 (for S^{no}) and Prob1 (for S^{yes})
  - algorithms work on underlying graph (probabilities irrelevant)
- Important for several reasons
  - reduces the set of states for which probabilities must be computed numerically (which is more expensive)
  - gives exact results for the states in S^{yes} and S^{no} (no round-off)
  - for $P_{\sim p}[\cdot]$ where p is 0 or 1, no further computation required

PCTL until – Linear equations

- Probabilities $\mathrm{Prob}(s, \phi_1 \mathrm{U} \phi_2)$ can now be obtained as the unique solution of the following set of linear equations:

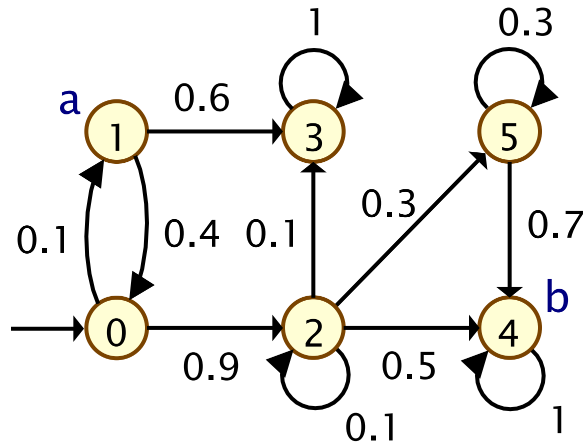
$$\mathrm{Prob}(s, \phi_1 \mathrm{U} \phi_2) = \begin{cases} 1 & \text{if } s \in S^{yes} \\ 0 & \text{if } s \in S^{no} \\ \sum_{s' \in S} P(s,s') \cdot \mathrm{Prob}(s', \phi_1 \mathrm{U} \phi_2) & \text{otherwise} \end{cases}$$

  - can be reduced to a system in $|S^?|$ unknowns instead of $|S|$ where $S^? = S \setminus (S^{yes} \cup S^{no})$

- This can be solved with (a variety of) standard techniques
  - direct methods, e.g. Gaussian elimination
  - iterative methods, e.g. Jacobi, Gauss-Seidel, …
    (preferred in practice due to scalability)

PCTL until – Example

- Example: $P_{>0.8}$ [¬a U b]

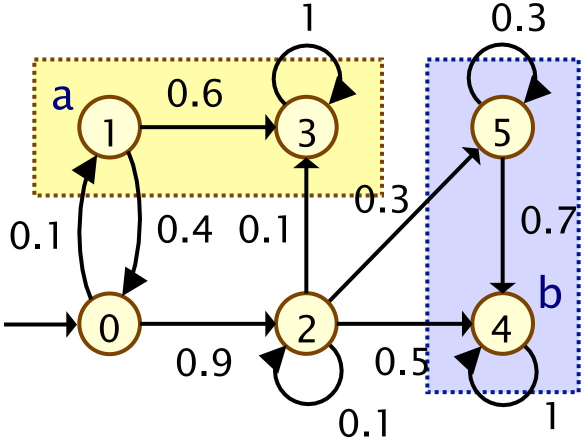


PCTL until – Example

- Example: $P_{>0.8}$ [¬a U b]

S^{no} =

Sat($P_{\leq 0}$ [¬a U b])



S^{yes} =

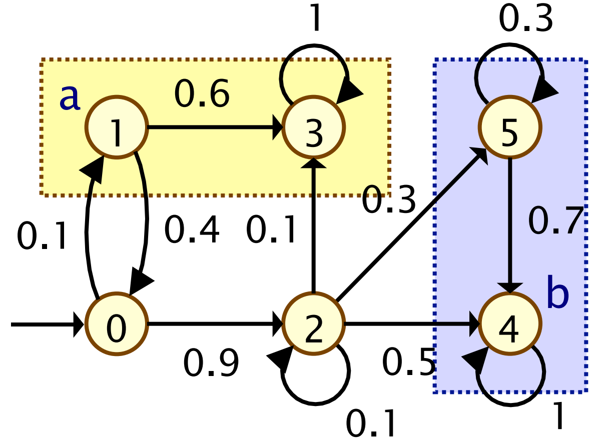
Sat($P_{\geq 1}$ [¬a U b])

PCTL until – Example

- Example: $P_{>0.8}$ [¬a U b]
- Let x_s = Prob(s, ¬a U b)
- Solve:

S^{no} =

Sat($P_{\leq 0}$ [¬a U b])



S^{yes} =

Sat($P_{\geq 1}$ [¬a U b])

$x_4 = x_5 = 1$

$x_1 = x_3 = 0$

$x_0 = 0.1x_1+0.9x_2 = 0.8$

$x_2 = 0.1x_2+0.1x_3+0.3x_5+0.5x_4 = 8/9$

<u>Prob</u>(¬a U b) = <u>x</u> = [0.8, 0, 8/9, 0, 1, 1]

Sat($P_{>0.8}$ [¬a U b]) = { s_2,s_4,s_5 }

PCTL model checking – Summary

- Computation of set Sat(Φ) for DTMC D and PCTL formula Φ
  - recursive descent of parse tree
  - combination of graph algorithms, numerical computation

- Probabilistic operator P:
  - $X\ \Phi$: one matrix-vector multiplication, $O(|S|^2)$
  - $\Phi_1\ U^{\leq k}\ \Phi_2$: k matrix-vector multiplications, $O(k|S|^2)$
  - $\Phi_1\ U\ \Phi_2$: linear equation system, at most |S| variables, $O(|S|^3)$

- Complexity:
  - linear in |Φ| and polynomial in |S|

Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
- Case study: Bluetooth device discovery

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)

- More expressive logics can be used, for example:
  - 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)

- Another direction: extend DTMCs with costs and rewards…

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
  - $\omega \vDash a \;\Leftrightarrow\; a \in L(\omega(0))$
  - $\omega \vDash \psi_1 \wedge \psi_2 \;\Leftrightarrow\; \omega \vDash \psi_1$ and $\omega \vDash \psi_2$
  - $\omega \vDash \neg\psi \;\Leftrightarrow\; \omega \nvDash \psi$
  - $\omega \vDash X\,\psi \;\Leftrightarrow\; \omega[1\ldots] \vDash \psi$
  - $\omega \vDash \psi_1\,U\,\psi_2 \;\Leftrightarrow\; \exists k \geq 0$ 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

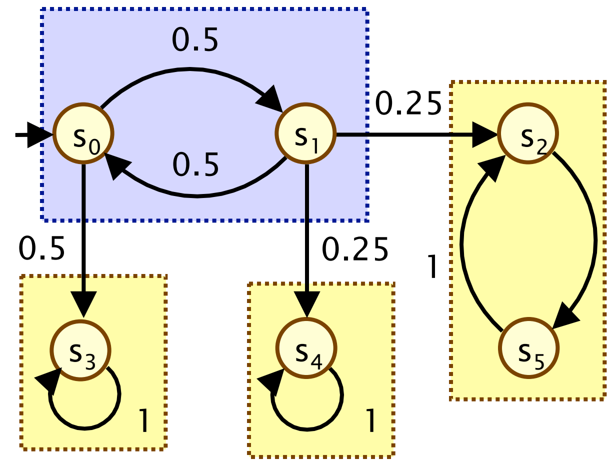
- $(\mathrm{F}\ tmp\_fail_1) \wedge (\mathrm{F}\ tmp\_fail_2)$
  - "both servers suffer temporary failures at some point"
- $\mathrm{GF}\ ready$
  - "the server always eventually returns to a ready-state"
- $\mathrm{FG}\ error$
  - "an irrecoverable error occurs"
- $\mathrm{G}\ (req \rightarrow \mathrm{X}\ ack)$
  - "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:
  - $\mathrm{Prob}(s, \psi) = \mathrm{Pr}_s \{ \omega \in \mathrm{Path}(s) \mid \omega \models \psi \}$
  - 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. $P_{<0.01}$ [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]

Fundamental property of DTMCs

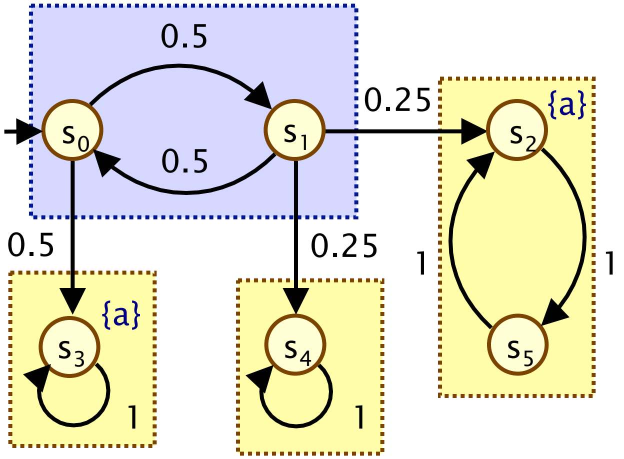
- Strongly connected component (SCC)
  - maximally strongly connected set of states
- Bottom strongly connected component (BSCC)
  - SCC T from which no state outside T is reachable from T
- Fundamental property of DTMCs:
  - "with probability 1,
    a BSCC will be reached
    and all of its states
    visited infinitely often"



- Formally:
  - $Pr_s\{\ \omega \in Path(s) \mid \exists\ i \geq 0, \exists$ BSCC T such that
    $\forall\ j \geq i\ \omega(i) \in T$ and
    $\forall\ s' \in T\ \omega(k) = s'$ for infinitely many k $\}\ =\ 1$

LTL model checking for DTMCs

- LTL model checking for DTMCs relies on:
  - computing probability of reaching a set of “accepting” BSCCs
  - e.g. for two simple LTL formulae: GF a (“always eventually a”), FG a (“eventually always a’) we have:



- 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

Deterministic Rabin automata

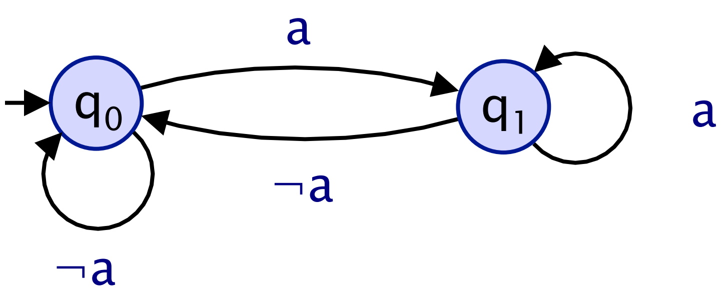
- ω-automata represent sets of infinite words
  - e.g. Buchi automata, Rabin automata, …
  - for probabilistic model checking, need deterministic automata
  - so we use deterministic Rabin automata (DRAs)

- 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, $\delta : Q \times \Sigma \rightarrow Q$ 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: $\bigvee_{1 \leq i \leq k} (FG\ \neg L_i \wedge GF\ K_i)$

LTL & DRAs

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



- Can convert any LTL formula ψ on atomic propositions AP
  - into an equivalent DRA A_ψ over alphabet 2^{AP}
  - i.e. $\omega \vDash \psi \Leftrightarrow trace(\omega) \in L(A_\psi)$ for any path ω
  - can potentially incur a double exponential blow-up (but, in practice, this does not occur and ψ is small anyway)

- LTL model checking for DTMCs – the basic idea
  - construct product of DTMC D and DRA A_ψ
  - compute $Prob^D(s, \psi)$ on product DTMC $D \otimes A$

Product DTMC for a DRA

- The product DTMC $D \otimes A$ for:
  - for DTMC $D = (S, s_{init}, \mathbf{P}, L)$ and
  - and (total) DRA $A = (Q, \Sigma, \delta, q_0, \{ (L_i, K_i) \}_{i=1..k})$
  - is the DTMC $(S \times Q, (s_{init}, q_{init}), \mathbf{P}', L')$ where:

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

$$\mathbf{P}'((s_1,q_1),(s_2,q_2)) = \begin{cases} \mathbf{P}(s_1,s_2) & \text{if } q_2 = \delta(q_1, L(s_2)) \\ 0 & \text{otherwise} \end{cases}$$

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

- Note:
  - $D \otimes A$ can be seen as unfolding of D where q for each state (s,q) records state of automata A for path fragment so far
  - since A is deterministic, $D \otimes A$ is a DTMC
  - each path in D has a corresponding (unique) path in $D \otimes A$
  - the probabilities of paths in D are preserved in $D \otimes A$

Product DTMC for a DRA

- For DTMC D and DRA A

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

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

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

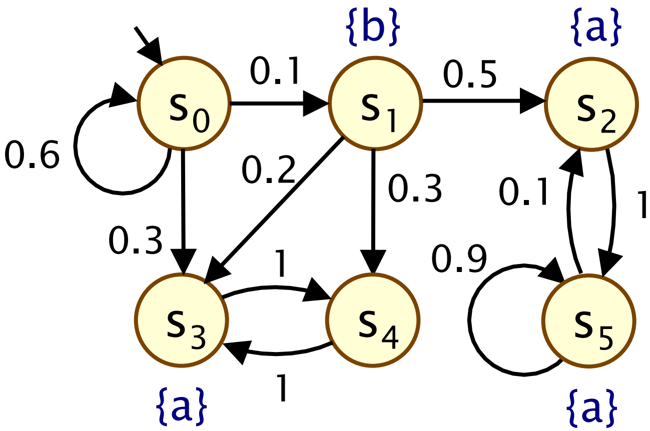
  – where T_{Acc} is the union of all accepting BSCCs in $D \otimes A$
  – an accepting BSCC T of $D \otimes A$ is such that, for some $1 \le i \le k$, no states in T satisfy l_i and some state in T satisfies k_i

- Reduces to computing BSCCs and reachability probabilities
  – so overall complexity for LTL is doubly exponential in $|\psi|$, polynomial in $|M|$; but can be reduced to singly exponential

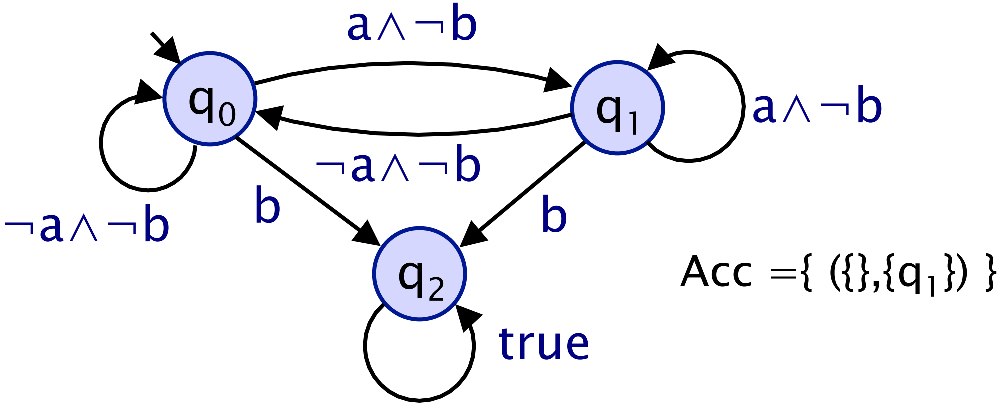
Example: LTL for DTMCs

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

DTMC D

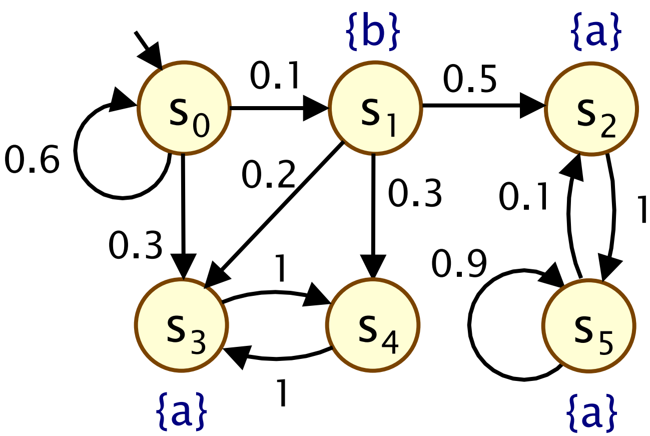


DRA A_ψ for ψ = G¬b ∧ GF a

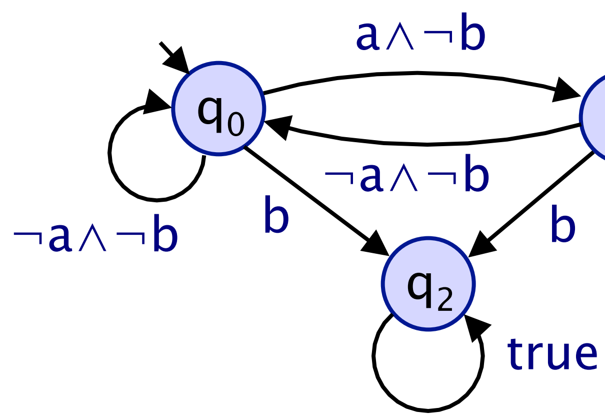


Example: LTL for DTMCs

DTMC D

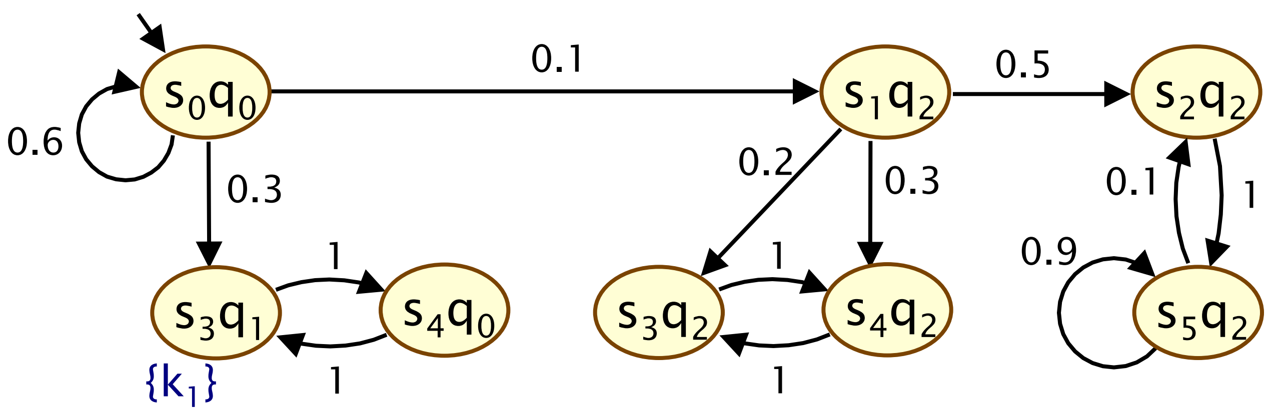


DRA A_ψ for $\psi = G\neg b \wedge GF\, a$



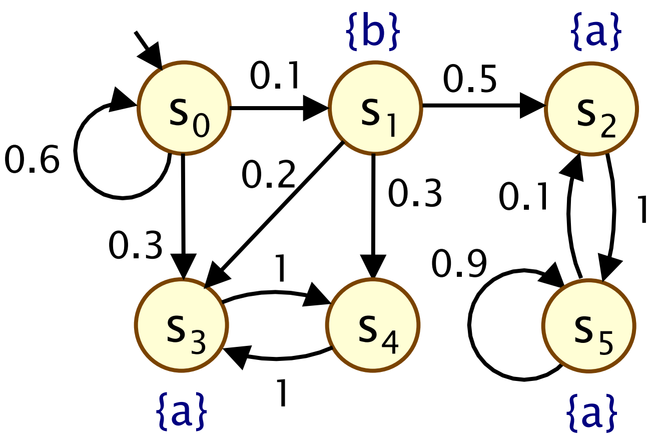
Acc ={ ({},{q_1}) }

Product DTMC $D \otimes A_\psi$

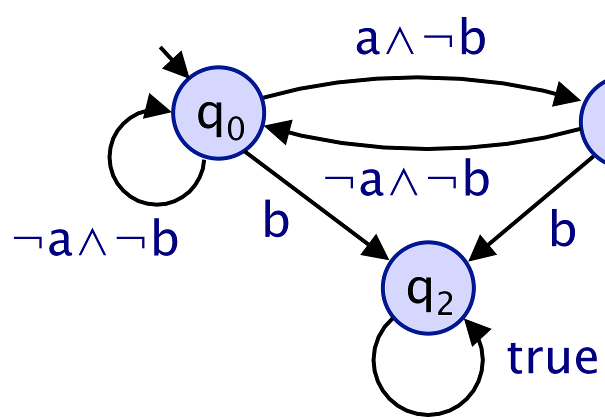


Example: LTL for DTMCs

DTMC D

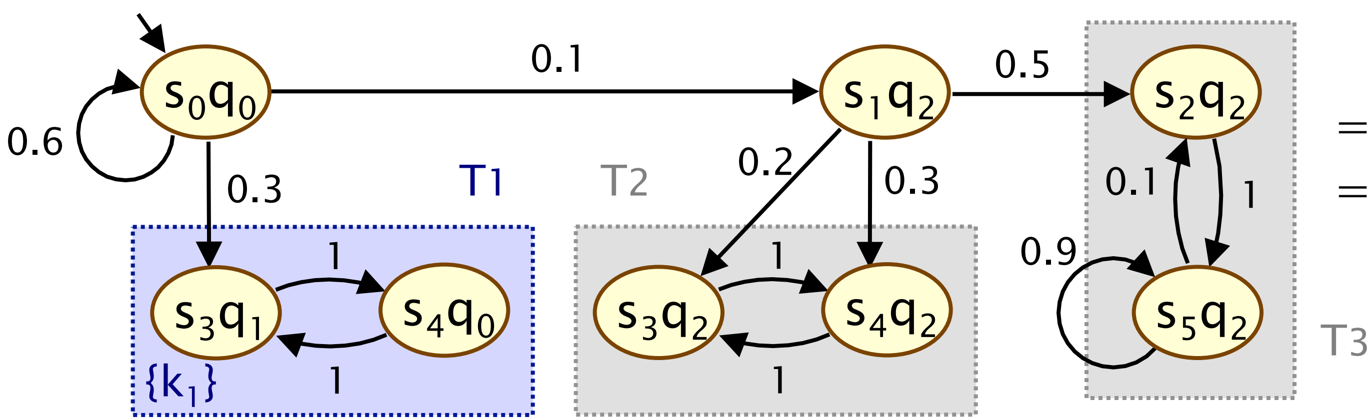


DRA A_ψ for $\psi = G\neg b \wedge GF\, a$



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

Product DTMC $D \otimes A_\psi$



$\mathrm{Prob}^D(s, \psi)$

$= \mathrm{Prob}^{D\otimes A\psi}(F\ T_1)$

$= 3/4.$

Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
- Case study: Bluetooth device discovery

Costs and rewards

- We augment DTMCs with rewards (or, conversely, costs)
  - real-valued quantities assigned to states and/or transitions
  - these can have a wide range of possible interpretations

- Some examples:
  - elapsed time, power consumption, size of message queue, number of messages successfully delivered, net profit, …

- Costs? or rewards?
  - mathematically, no distinction between rewards and costs
  - when interpreted, we assume that it is desirable to minimise costs and to maximise rewards
  - we will consistently use the terminology "rewards" regardless

Reward-based properties

- Properties of DTMCs augmented with rewards
  - allow a wide range of quantitative measures of the system
  - basic notion: expected value of rewards
  - formal property specifications will be in an extension of PCTL

- More precisely, we use two distinct classes of property…

- Instantaneous properties
  - the expected value of the reward at some time point

- Cumulative properties
  - the expected cumulated reward over some period

DTMC reward structures

- For a DTMC $(S, s_{init}, \mathbf{P}, L)$, a reward structure is a pair $(\underline{\rho}, \boldsymbol{\iota})$
  - $\underline{\rho} : S \rightarrow \mathbb{R}_{\geq 0}$ is the state reward function (vector)
  - $\boldsymbol{\iota} : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is the transition reward function (matrix)

- Example (for use with instantaneous properties)
  - “size of message queue”: $\underline{\rho}$ maps each state to the number of jobs in the queue in that state, $\boldsymbol{\iota}$ is not used

- Examples (for use with cumulative properties)
  - “time-steps”: $\underline{\rho}$ returns 1 for all states and $\boldsymbol{\iota}$ is zero (equivalently, $\underline{\rho}$ is zero and $\boldsymbol{\iota}$ returns 1 for all transitions)
  - “number of messages lost”: $\underline{\rho}$ is zero and $\boldsymbol{\iota}$ maps transitions corresponding to a message loss to 1
  - “power consumption”: $\underline{\rho}$ is defined as the per-time-step energy consumption in each state and $\boldsymbol{\iota}$ as the energy cost of each transition

PCTL and rewards

- Extend PCTL to incorporate reward-based properties
  - add an R operator, which is similar to the existing P operator

expected reward is ~r

  - ϕ ::= … | $P_{\sim p}$ [ψ] | $R_{\sim r}$ [$I^{=k}$] | $R_{\sim r}$ [$C^{\leq k}$] | $R_{\sim r}$ [F ϕ]

"instantaneous" "cumulative" "reachability"

  - where $r \in \mathbb{R}_{\geq 0}$, $\sim \in \{<,>,\leq,\geq\}$, $k \in \mathbb{N}$

- $R_{\sim r}$ [·] means "the expected value of · satisfies ~r"

Types of reward formulas

- Instantaneous: $R_{\sim r}$ [$I^{=k}$]
  - "the expected value of the state reward at time-step k is ~r"
  - e.g. "the expected queue size after exactly 90 seconds"
- Cumulative: $R_{\sim r}$ [$C^{\leq k}$]
  - "the expected reward cumulated up to time-step k is ~r"
  - e.g. "the expected power consumption over one hour"
- Reachability: $R_{\sim r}$ [F ϕ]
  - "the expected reward cumulated before reaching a state satisfying ϕ is ~r"
  - e.g. "the expected time for the algorithm to terminate"

Reward formula semantics

- Formal semantics of the three reward operators
  - based on random variables over (infinite) paths

- Recall:
  - $s \vDash P_{\sim p} [\ \psi\ ] \iff Pr_s \{ \omega \in Path(s) \mid \omega \vDash \psi \} \sim p$

- For a state s in the DTMC:
  - $s \vDash R_{\sim r} [\ I^{=k}\ ] \iff Exp(s, X_{I=k}) \sim r$
  - $s \vDash R_{\sim r} [\ C^{\leq k}\ ] \iff Exp(s, X_{C\leq k}) \sim r$
  - $s \vDash R_{\sim r} [\ F\ \Phi\ ] \iff Exp(s, X_{F\Phi}) \sim r$

  where: Exp(s, X) denotes the expectation of the random variable $X : Path(s) \rightarrow \mathbb{R}_{\geq 0}$ with respect to the probability measure Pr_s

Reward formula semantics

- Definition of random variables:
  - for an infinite path $\omega = s_0 s_1 s_2 \ldots$

$$X_{I=k}(\omega) = \underline{\rho}(s_k)$$

$$X_{C\leq k}(\omega) = \begin{cases} 0 & \text{if } k = 0 \\ \sum_{i=0}^{k-1} \underline{\rho}(s_i) + \iota(s_i, s_{i+1}) & \text{otherwise} \end{cases}$$

$$X_{F\phi}(\omega) = \begin{cases} 0 & \text{if } s_0 \in Sat(\phi) \\ \infty & \text{if } s_i \notin Sat(\phi) \text{ for all } i \geq 0 \\ \sum_{i=0}^{k_\phi - 1} \underline{\rho}(s_i) + \iota(s_i, s_{i+1}) & \text{otherwise} \end{cases}$$

  - where $k_\phi = \min\{ j \mid s_j \vDash \phi \}$

Model checking reward properties

- Instantaneous: $R_{\sim r}$ [$I^{=k}$]
- Cumulative: $R_{\sim r}$ [$C^{\leq t}$]
  - variant of the method for computing bounded until probabilities
  - solution of recursive equations

- Reachability: $R_{\sim r}$ [F ϕ]
  - similar to computing until probabilities
  - precomputation phase (identify infinite reward states)
  - then reduces to solving a system of linear equation

- For more details, see e.g. [KNP07a]

Overview (Part 1)

- Discrete-time Markov chains (DTMCs)
- PCTL: A temporal logic for DTMCs
- PCTL model checking
- LTL model checking
- Costs and rewards
- Case study: Bluetooth device discovery

The PRISM tool

- PRISM: Probabilistic symbolic model checker
  - developed at Birmingham/Oxford University, since 1999
  - free, open source (GPL), runs on all major OSs
- Support for:
  - discrete-/continuous-time Markov chains (D/CTMCs)
  - Markov decision processes (MDPs)
  - probabilistic timed automata (PTAs)
  - PCTL, CSL, LTL, PCTL*, costs/rewards, …
- Multiple efficient model checking engines
  - mostly symbolic (BDDs) (up to 10^{10} states, 10^7–10^8 on avg.)
- Successfully applied to a wide range of case studies
  - communication protocols, security protocols, dynamic power management, cell signalling pathways, …
  - http://www.prismmodelchecker.org/

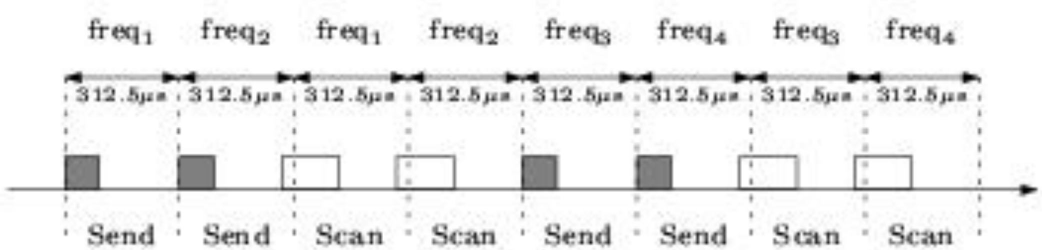
Bluetooth device discovery

- Bluetooth: short-range low-power wireless protocol
  - widely available in phones, PDAs, laptops, ...
  - open standard, specification freely available
- Uses frequency hopping scheme
  - to avoid interference (uses unregulated 2.4GHz band)
  - pseudo-random selection over 32 of 79 frequencies
- Formation of personal area networks (PANs)
  - piconets (1 master, up to 7 slaves)
  - self-configuring: devices discover themselves
- Device discovery
  - mandatory first step before any communication possible
  - relatively high power consumption so performance is crucial
  - master looks for devices, slaves listens for master

Master (sender) behaviour

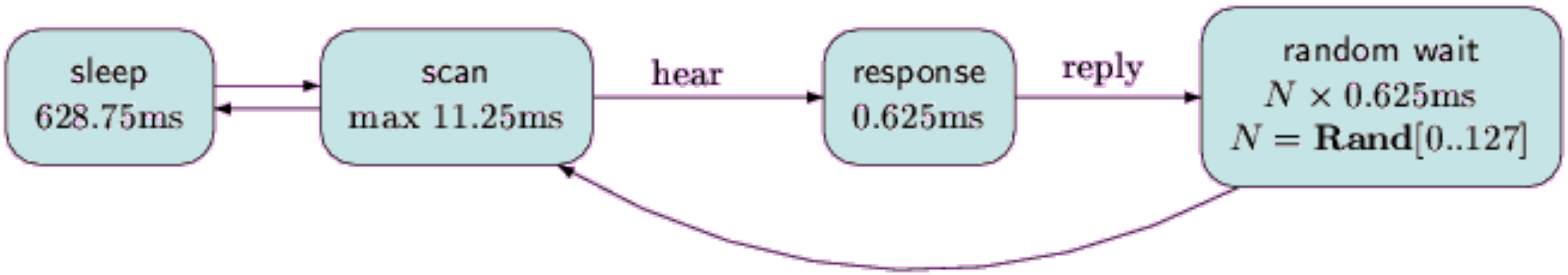
- 28 bit free-running clock CLK, ticks every 312.5µs
- Frequency hopping sequence determined by clock:
  - freq = $[CLK_{16-12}+k+(CLK_{4-2,0}-CLK_{16-12}) \bmod 16] \bmod 32$
  - 2 trains of 16 frequencies (determined by offset k), 128 times each, swap between every 2.56s
- Broadcasts "inquiry packets" on two consecutive frequencies, then listens on the same two

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	13	14	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
1	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
17	18	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	4	5	6	7	8	9	10	11	12	13	14	15	16
1	2	3	4	21	22	23	24	25	26	27	28	29	30	31	32
1	2	3	4	5	22	23	24	25	26	27	28	29	30	31	32
17	18	19	20	21	22	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	8	9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8	25	26	27	28	29	30	31	32
1	2	3	4	5	6	7	8	9	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24	25	26	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	12	13	14	15	16
1	2	3	4	5	6	7	8	9	10	11	12	29	30	31	32
1	2	3	4	5	6	7	8	9	10	11	12	13	30	31	32
17	18	19	20	21	22	23	24	25	26	27	28	29	30	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	16



Slave (receiver) behaviour

- Listens (scans) on frequencies for inquiry packets
  - must listen on right frequency at right time
  - cycles through frequency sequence at much slower speed (every 1.28s)



- On hearing packet, pause, send reply and then wait for a random delay before listening for subsequent packets
  - avoid repeated collisions with other slaves

Bluetooth – PRISM model

- Modelled/analysed using PRISM model checker [DKNP06]
  - model scenario with one sender and one receiver
  - synchronous (clock speed defined by Bluetooth spec)
  - model at lowest-level (one clock-tick = one transition)
  - randomised behaviour so model as a DTMC
  - use real values for delays, etc. from Bluetooth spec

- Modelling challenges
  - complex interaction between sender/receiver
  - combination of short/long time-scales – cannot scale down
  - sender/receiver not initially synchronised, so huge number of possible initial configurations (17,179,869,184)

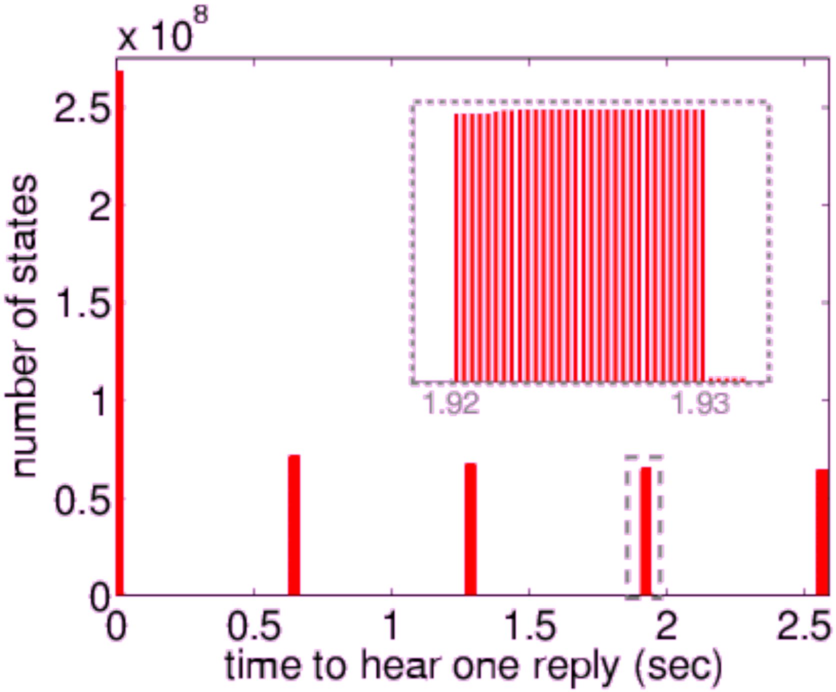
Bluetooth – Results

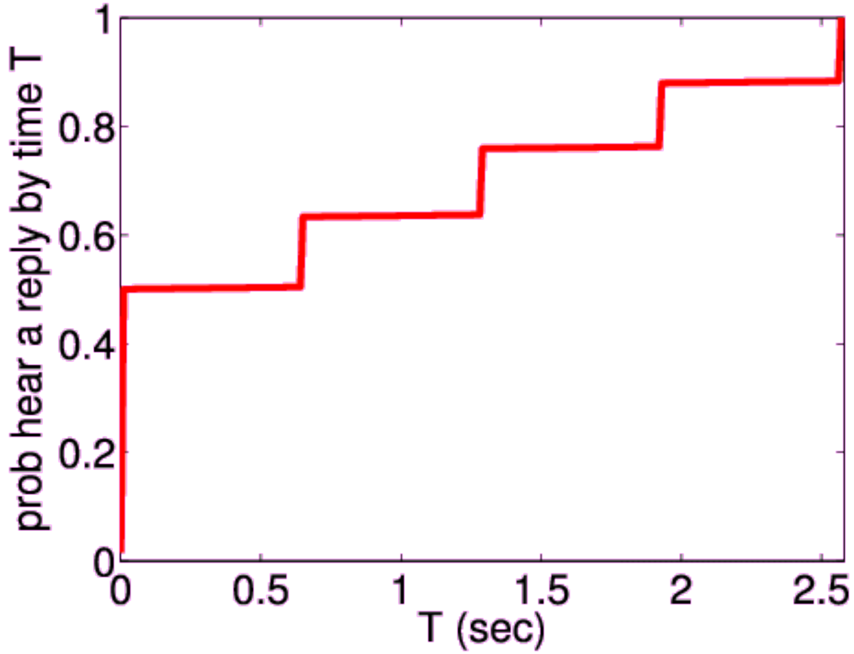
- Huge DTMC – initially, model checking infeasible
  - partition into 32 scenarios, i.e. 32 separate DTMCs
  - on average, approx. 3.4×10^9 states (536,870,912 initial)
  - can be built/analysed with PRISM's MTBDD engine

- We compute:
  - R=? [F replies=K {"init"}{max}]
  - "worst-case expected time to hear K replies over all possible initial configurations"

- Also look at:
  - how many initial states for each possible expected time
  - cumulative distribution function (CDF) for time, assuming equal probability for each initial state

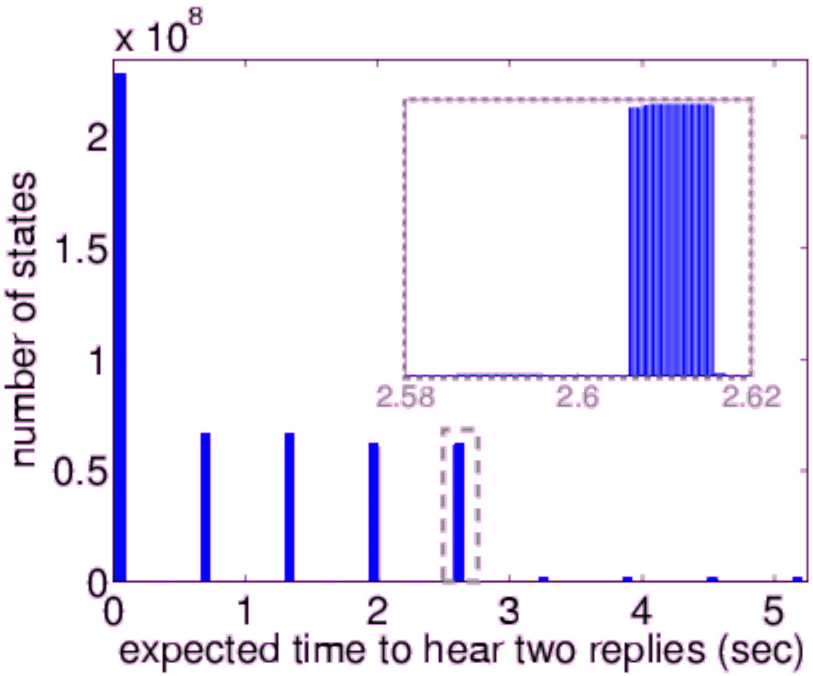
Bluetooth – Time to hear 1 reply

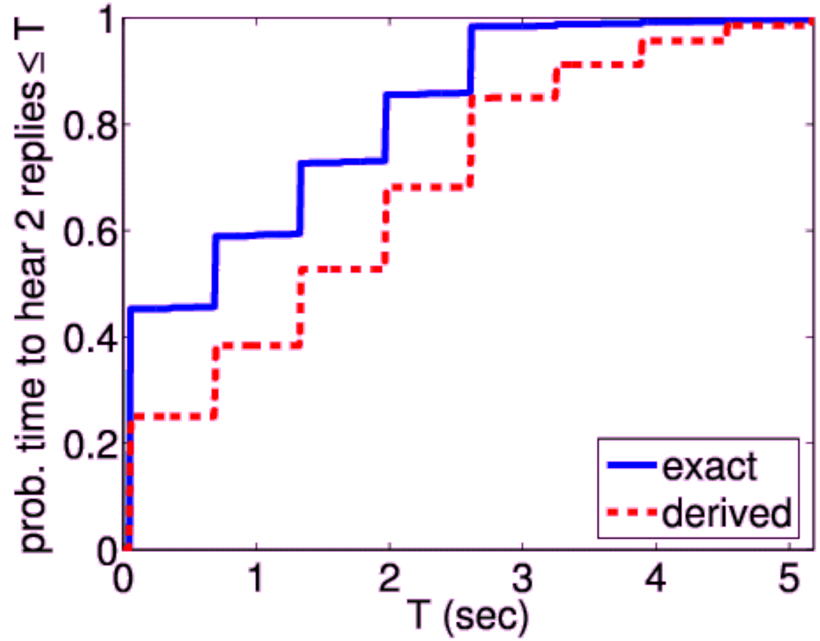




- Worst-case expected time = 2.5716 sec
  - in 921,600 possible initial states
  - best-case = 635 µs

Bluetooth – Time to hear 2 replies





- Worst-case expected time = 5.177 sec
  - in 444 possible initial states
  - compare actual CDF with derived version which assumes times to reply to first/second messages are independent

Bluetooth – Results

- Other results: (see [DKNP06])
  - compare versions 1.2 and 1.1 of Bluetooth, confirm 1.1 slower
  - power consumption analysis (using costs + rewards)

- Conclusions:
  - successful analysis of complex real-life model
  - detailed model, actual parameters used
  - exhaustive analysis: best/worst-case values
    - can pinpoint scenarios which give rise to them
    - not possible with simulation approaches
  - model still relatively simple
    - consider multiple receivers?
    - combine with simulation?

Summary

- Probabilistic model checking
  - automated quantitative verification of stochastic systems
  - to model randomisation, failures, …
- Discrete-time Markov chains (DTMCs)
  - state transition systems + discrete probabilistic choice
  - probability space over paths through a DTMC
- Property specifications
  - probabilistic extensions of temporal logic, e.g. PCTL, LTL
  - also: expected value of costs/rewards
- Model checking algorithms
  - combination of graph-based algorithms, numerical computation, automata constructions

- Next: Markov decision processes (MDPs)