Stochastic Model Checking

Marta Kwiatkowska

University of Birmingham
www.cs.bham.ac.uk/~mzk

SFM-07:PE, Bertinoro, 31st May 2007

Overview

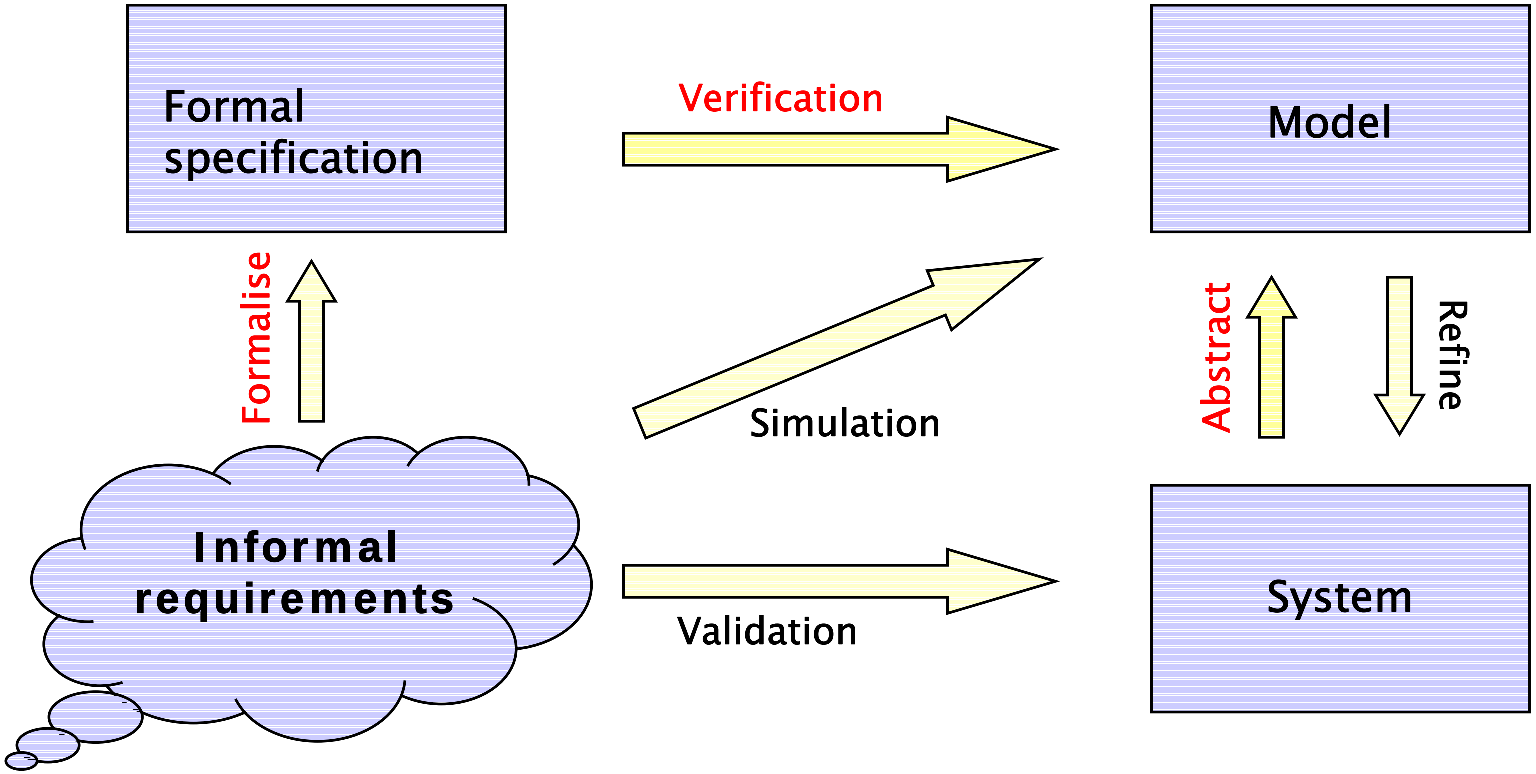
- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

Ubiquitous computing – The trends…

- Devices, ever smaller
  – Laptops, phones, PDAs, sensors…
- Networking, wireless, wired & global
  – Wireless & Internet everywhere
- Systems/software
  – Self-*
  – Mobile
  – Adaptive
  – Context-aware
- How to design & engineer
  – Adaptive systems and networks?
- How to ensure
  – Dependability and performance?

Modern trends in software engineering

- Verification and validation
  - Derive model, or extract from software
  - Verify correctness, validate if fit for purpose



Why must we verify?

"Testing can only show the presence of errors, not their absence."

"In their capacity as a tool, computers will be but a ripple on the surface of our culture. In their capacity as intellectual challenge, computers are without precedent in the cultural history of mankind."

Edsger Wybe Dijkstra

1930-2002

To rule out errors must consider all possible executions – often not feasible mechanically!

But my program works!

- True, there are many successful large-scale complex computer systems…
  - Online banking, electronic commerce
  - Information services, online libraries, business processes
  - Supply chain management
  - Mobile phone networks
- Yet many new potential application domains, far greater complexity, higher expectations
  - Automotive drive-by-wire
  - Medical sensors: heart rate & blood pressure monitors
  - Intelligent buildings and spaces: WiFi hotspots, environmental sensors
- Learning from mistakes costly…

Toyota Prius

Drive-by-wire, in car network

100s of embedded components used in modern cars

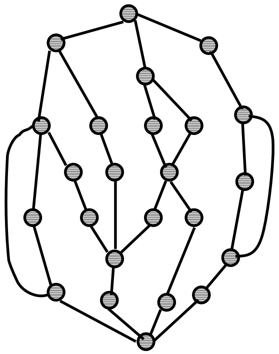
2005 Toyota Prius hybrid

In May 2005, Toyota recalls about 75,000 cars.
Some Prius drivers have reported sudden stalling or stopping at highway speeds.

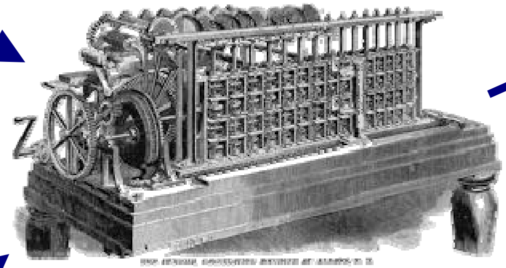
According to reports “the stalling problem is due to a software glitch in its sophisticated computer system.”

Such problems are becoming more common: BMW 7 series, …
Cost $?

Verification via model checking



Finite-state model



Model Checker

Error trace

Line 5: ...
Line 21: ...
Line 15: ...
...
Line 27: ...
Line 45: ...

init → F response

Temporal logic specification

Role of model checking

- Automated techniques for the assurance of
  - safety
  - security, privacy & trust
  - performance
  - dependability
- NB, 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?
- Focus on stochastic model checking
  - to capture probability and resource usage
  - range of quantitative analyses

Why probability?

- Randomisation used 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
- For quantitative analysis of software and systems
  - to quantify resource usage given a policy

    "the minimum battery capacity for a given scenario is .."
- In evidence-based, statistical analysis of behaviours
  - to quantify trust, anonymity, etc
- In modelling of biological processes
  - to quantify concentrations or numbers of molecules

    "the expected long-run percentage of Na molecules is … "

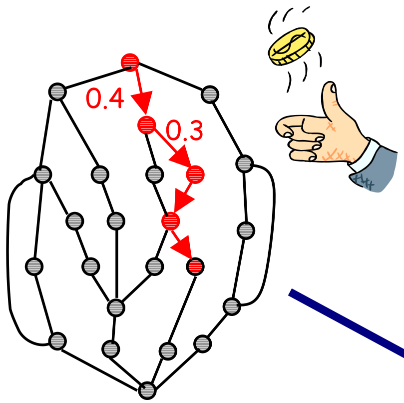
Real-world protocol examples

- Protocols featuring randomisation
  - Randomised back-off schemes
    - CSMA protocol
    - 802.11 Wireless LAN
  - Random choice of waiting time
    - IEEE 1394 Firewire root contention
    - Bluetooth, device discovery phase
  - Random choice over a set of possible addresses
    - IPv4 Zeroconf dynamic configuration (link-local addressing)
  - and more

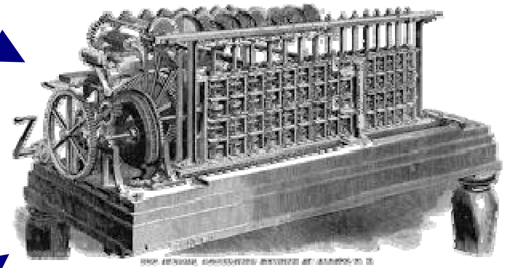
- Continuous probability distribution needed to model network traffic, node mobility, random delays…

Probabilistic model checking…

in a nutshell



Probabilistic model



Probabilistic
Model Checker

or

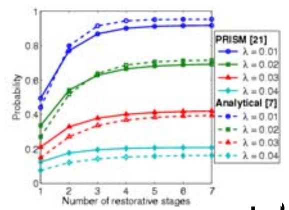
or

The probability

send → $P_{\geq p}$ [F deliver]

Probabilistic temporal
logic specification

State 5: 0.6789
State 6: 0.9789
State 7: 1.0
…
State 12: 0
State 13: 0.1245



Probabilistic model checking inputs

- Models: variants of Markov chains
  - Discrete-Time Markov Chains (DTMCs)
  - Markov Decision Processes (MDPs)
  - Continuous-Time Markov Chains (CTMCs)
  - Probabilistic Time Automata (PTAs)

- Specifications (informally)
  - "probability of delivery within time deadline is …"
  - "expected time to message delivery is …"
  - "expected power consumption is …"

- Specifications (formally)
  - Probabilistic temporal logics (PCTL, CSL, PTCTL)
  - Probability, time, cost/rewards

Probabilistic model checking involves…

- Construction of models
  - from a high-level modelling language
  - e.g. probabilistic process algebra

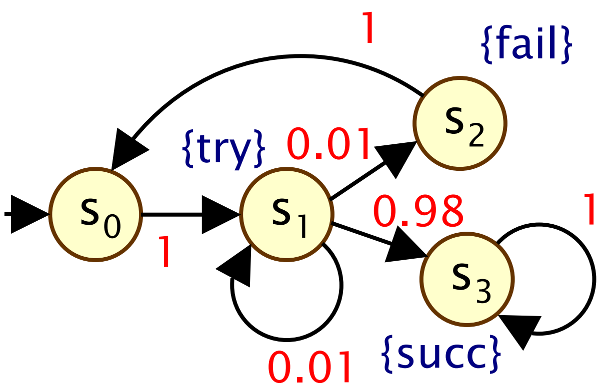
- Implementation of probabilistic model checking algorithms
  - graph-theoretical algorithms, combined with
    - (probabilistic) reachability
  - numerical computation – iterative methods
    - quantitative model checking (plot values for a range of parameters)
    - typically, linear equation or linear optimisation
    - exhaustive, unlike simulation
  - also sampling-based (statistical) for approximate analysis
    - e.g. hypothesis testing based on simulation runs

Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

Discrete-time Markov chains

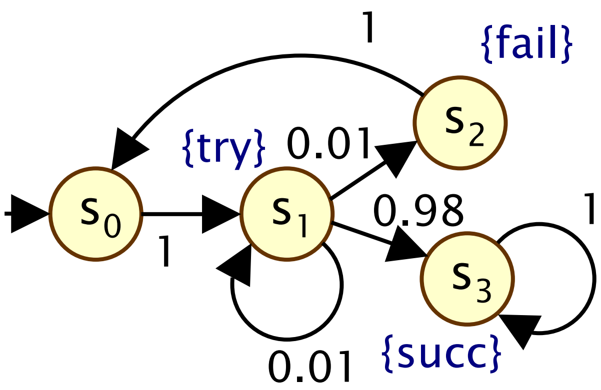
- Discrete-time Markov chains (DTMCs)
  - state-transition systems augmented with probabilities
- States
  - discrete set of states representing possible configurations of the system being modelled
- Transitions
  - transitions between states occur in discrete time-steps
- Probabilities
  - probability of making transitions between states is given by discrete probability distributions



Discrete-time Markov chains

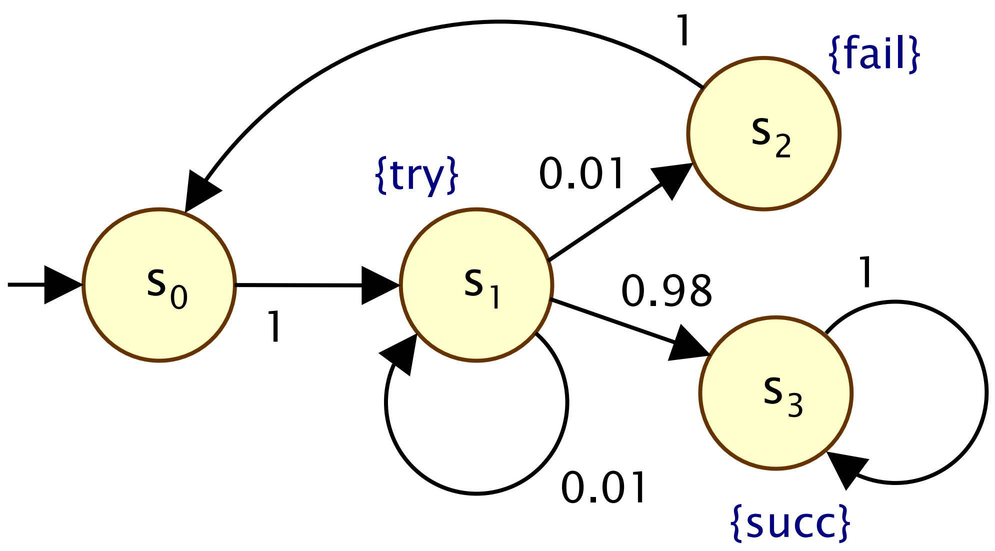
- Formally, a DTMC D is a tuple $(S, s_{init}, \mathbf{P}, L)$ where:
  - S is a finite set of states ("state space")
  - $s_{init} \in S$ is the initial state
  - $\mathbf{P} : S \times S \rightarrow [0,1]$ is the transition probability matrix where $\Sigma_{s' \in S} \mathbf{P}(s,s') = 1$ for all $s \in S$
  - $L : S \rightarrow 2^{AP}$ 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



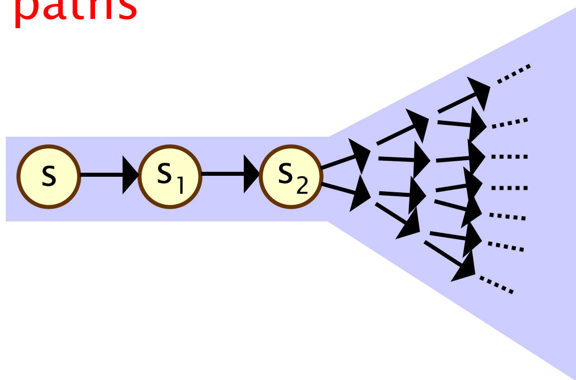
Simple DTMC example

- Modelling a very simple communication protocol
  - after one step, process starts trying to send a message
  - with probability 0.01, channel unready so wait a step
  - with probability 0.98, send message successfully and stop
  - with probability 0.01, message sending fails, restart



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

- Probability space (Ω, Σ, Pr)
  - Ω is the sample space
  - Σ is the set of events: σ-algebra on Ω
  - $Pr : \Sigma \rightarrow [0,1]$ 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) (infinite paths with initial state s)

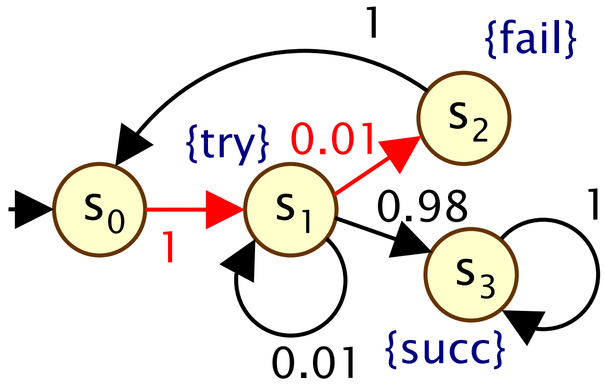
- Event set $\Sigma_{Path(s)}$ is the least σ-algebra on Path(s) containing
  - the cylinder sets $C(\omega) = \{ \omega' \in Path(s) \mid \omega \text{ is prefix of } \omega' \}$ for all finite paths ω starting in s

- 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. $\omega = s$)
    - $\mathbf{P}_s(\omega) = \mathbf{P}(s,s_1) \cdot \ldots \cdot \mathbf{P}(s_{n-1},s_n)$ otherwise
  - define $Pr_s(C(\omega)) = \mathbf{P}_s(\omega)$ for all finite paths ω
  - Pr_s extends uniquely to a probability measure $Pr_s:\Sigma_{Path(s)} \to [0,1]$

- 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
  - $C(s_0 s_1 s_3) \cup C(s_0 s_1 s_1 s_3) \cup C(s_0 s_1 s_1 s_1 s_3) \cup \ldots$
  - $Pr_{s0}(\, C(s_0 s_1 s_3) \cup C(s_0 s_1 s_1 s_3) \cup C(s_0 s_1 s_1 s_1 s_3) \cup \ldots \,)$
    $= \mathbf{P}_{s0}(s_0 s_1 s_3) + \mathbf{P}_{s0}(s_0 s_1 s_1 s_3) + \mathbf{P}_{s0}(s_0 s_1 s_1 s_1 s_3) + \ldots$
    $= 1 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.01 \cdot 0.98 + \ldots$
    $= 98/99$
    $= 0.9898989898\ldots$

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

- PCTL syntax:

ψ is true with probability $\sim p$

  – 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)

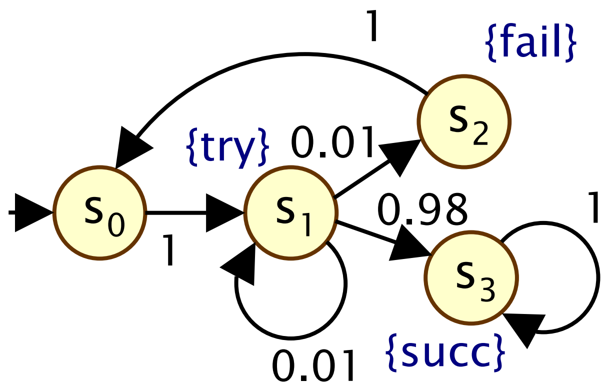
"next"   "bounded until"   "unbound until"

  – 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

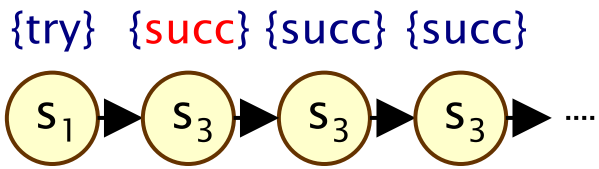
- PCTL formulas interpreted over states of a DTMC
  - $s \vDash \phi$ denotes ϕ is "true in state s" or "satisfied in state s"
- Semantics of (non-probabilistic) state formulas:
  - for a state s of the DTMC (S, s_{init}, P, L):
  - $s \vDash a \quad \Leftrightarrow \quad a \in L(s)$
  - $s \vDash \phi_1 \wedge \phi_2 \quad \Leftrightarrow \quad s \vDash \phi_1$ and $s \vDash \phi_2$
  - $s \vDash \neg\phi \quad \Leftrightarrow \quad s \vDash \phi$ is false
- Examples
  - $s_3 \vDash succ$
  - $s_1 \vDash try \wedge \neg fail$



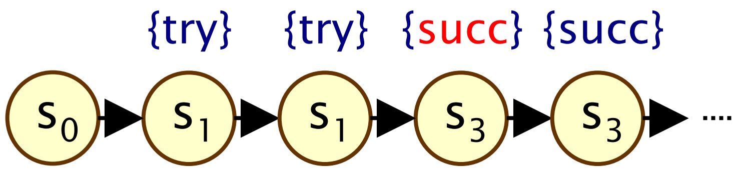
PCTL semantics for DTMCs

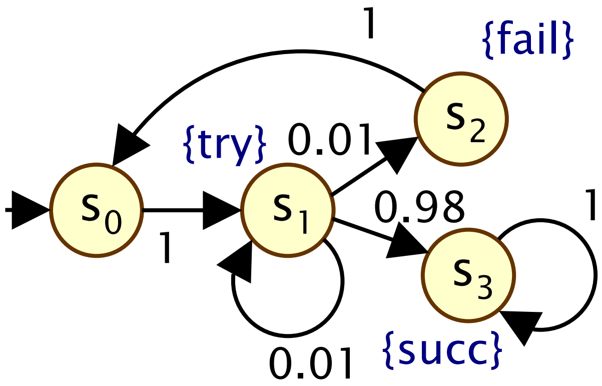
- Semantics of path formulas:
  - for a path $\omega = s_0 s_1 s_2 \ldots$ in the DTMC:
  - $\omega \vDash X\ \phi \quad \Leftrightarrow \quad s_1 \vDash \phi$
  - $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2 \quad \Leftrightarrow \quad \exists i \leq k$ such that $s_i \vDash \phi_2$ and $\forall j<i,\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

- Some examples of satisfying paths:
  - X succ



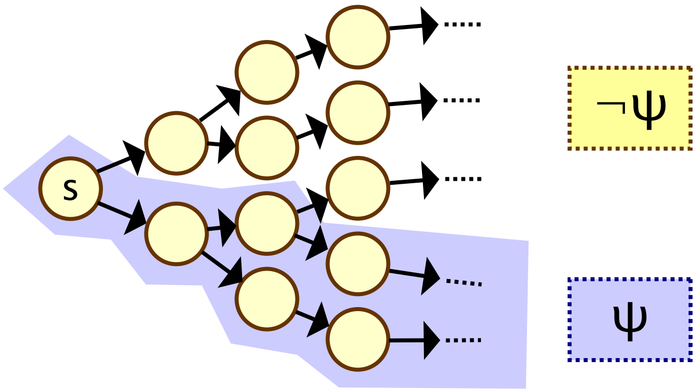
  - ¬fail U succ





PCTL semantics

- 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 $\sim p$"
  - example: $s \models P_{<0.25} [\ X\ \text{fail}\ ] \Leftrightarrow$ "the probability of atomic proposition fail being true in the next state of outgoing paths from s is less than 0.25"
  - formally: $s \models P_{\sim p} [\psi] \;\Leftrightarrow\; \text{Prob}(s, \psi) \sim p$
  - where: $\text{Prob}(s, \psi) = \text{Pr}_s \{\ \omega \in \text{Path}(s) \mid \omega \models \psi\ \}$



Prob(s, ψ) ~ p ?

PCTL derived operators

- Basic logical equivalences:
  - $\text{false} \equiv \neg\text{true}$ (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)
- Negation and probabilities
  - e.g. $\neg P_{>p} [\ \phi_1\ U\ \phi_2\ ] \equiv P_{\leq p} [\phi_1\ U\ \phi_2\ ]$
- The "eventually" path operator
  - $F\ \phi \equiv \text{true}\ U\ \phi$ (F = "future")
  - sometimes written as $\Diamond\ \phi$ ("diamond")
  - "ϕ is eventually true"
  - bounded version: $F^{\leq k}\ \phi \equiv \text{true}\ U^{\leq k}$

More PCTL

- The "always" path operator
  - $G\ \phi \equiv \neg(F\ \neg\phi) \equiv \neg(\text{true}\ U\ \neg\phi)$ (G = "globally")
  - sometimes written as $\Box\ \phi$ ("box")
  - "ϕ is always true"
  - bounded version: $G^{\leq k}\ \phi \equiv \neg(F^{\leq k}\ \neg\phi)$
  - strictly speaking, $G\ \phi$ cannot be derived from the PCTL syntax in this way since there is no negation of path formulas)

- F and G represent two useful classes of properties:
  - reachability: the probability of reaching a state satisfying ϕ
  - i.e. $P_{\sim p}\ [\ F\ \phi\ ]$
  - invariance: the probability of ϕ always remaining true
  - i.e. $P_{\sim p}\ [\ G\ \phi\ ]$

PCTL and measurability

- All the sets of paths expressed by PCTL are measurable
  - i.e. are elements of the σ-algebra $\Sigma_{Path(s)}$
  - see for example [Var85] (for a stronger result in fact)
- Recall: probability space $(Path(s), \Sigma_{Path(s)}, Pr_s)$
  - $\Sigma_{Path(s)}$ contains cylinder sets $C(\omega)$ for all finite paths ω starting in s and is closed under complementation, countable union
- Next $(X\ \phi)$
  - cylinder sets constructed from paths of length one
- Bounded until $(\phi_1\ U^{\leq k}\ \phi_2)$
  - (finite number of) cylinder sets from paths of length at most k
- Until $(\phi_1\ U\ \phi_2)$
  - countable union of paths satisfying $\phi_1\ U^{\leq k}\ \phi_2$ for all $k \geq 0$

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)

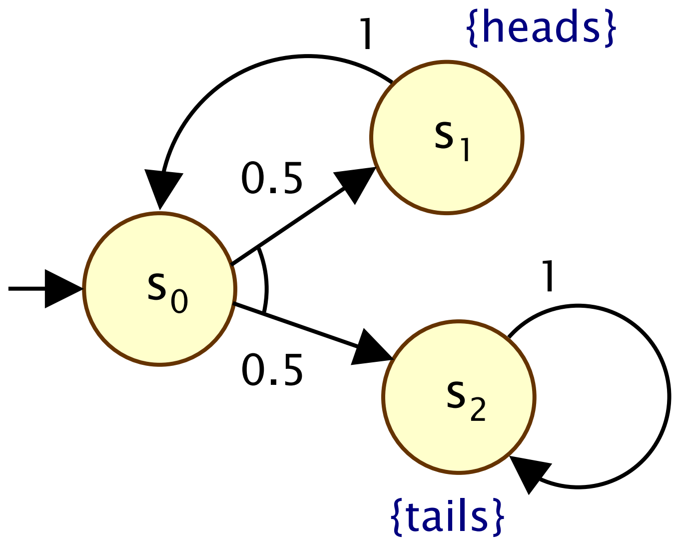
- Qualitative PCTL properties
  - $P_{\sim p}$ [ψ] where p is either 0 or 1
- Quantitative PCTL properties
  - $P_{\sim p}$ [ψ] where p is in the range (0,1)

- $P_{>0}$ [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
  - see next slide…

Example: Qualitative/quantitative

- Toss a coin repeatedly until “tails” is thrown

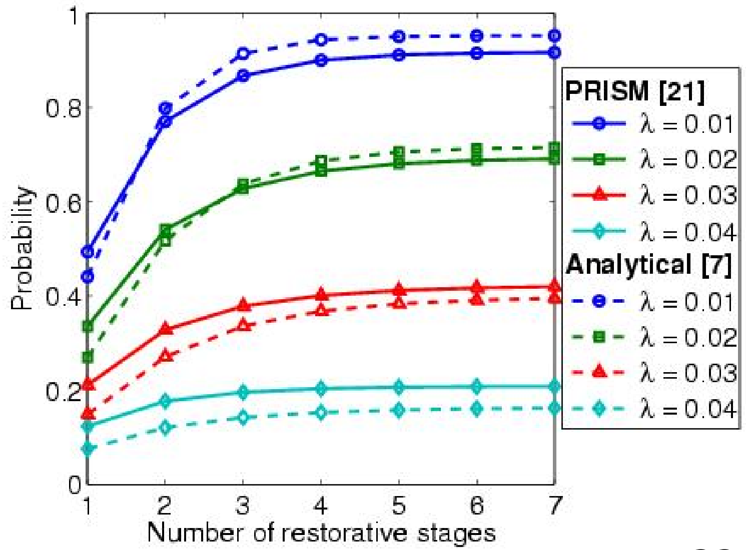
- Is “tails” always eventually thrown?
  - CTL: AF “tails”
  - Result: false
  - Counterexample: $s_0 s_1 s_0 s_1 s_0 s_1 \ldots$



- Does the probability of eventually throwing “tails” equal one?
  - PCTL: $P_{\geq 1}$ [F “tails”]
  - Result: true
  - Infinite path $s_0 s_1 s_0 s_1 s_0 s_1 \ldots$ has zero probability

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

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

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

PCTL model checking

- Algorithm for PCTL model checking [HJ94]
  - inputs: DTMC $D=(S,s_{init},\mathbf{P},L)$, PCTL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \models \phi \}$ = set of states satisfying ϕ

- What does it mean for a DTMC D to satisfy a formula ϕ?
  - sometimes, want to check that $s \models \phi\ \forall\ s \in S$, i.e. $Sat(\phi) = S$
  - sometimes, just want to know if $s_{init} \models \phi$, i.e. if $s_{init} \in Sat(\phi)$

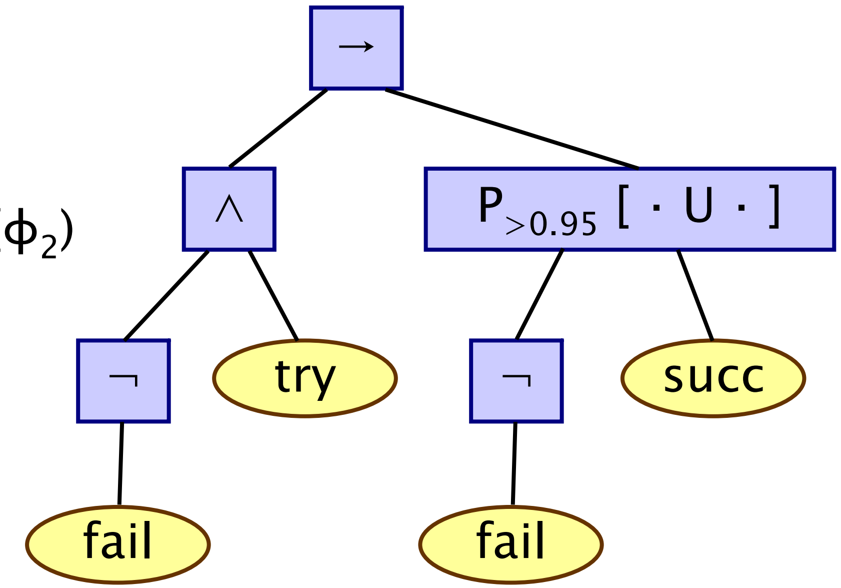
- Sometimes, focus on quantitative results
  - e.g. compute result of P=? [F error]
  - e.g. compute result of P=? [$F^{\leq k}$ error] for $0 \leq k \leq 100$

PCTL model checking

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

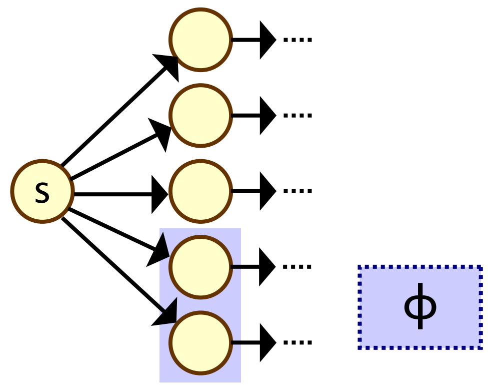
- For the $P_{\sim p}\ [\ \psi\ ]$ operator
  - need to compute the probabilities Prob(s, ψ) for all states $s \in S$



PCTL next

- Computation of probabilities for PCTL next operator
  - $\text{Sat}(P_{\sim p}[\ X\ \phi\ ]) = \{\ s \in S \mid \text{Prob}(s, X\ \phi) \sim p\ \}$
  - need to compute $\text{Prob}(s, X\ \phi)$ for all $s \in S$

- Sum outgoing probabilities for transitions to ϕ-states
  - $\text{Prob}(s, X\ \phi) = \Sigma_{s' \in \text{Sat}(\phi)}\ P(s,s')$

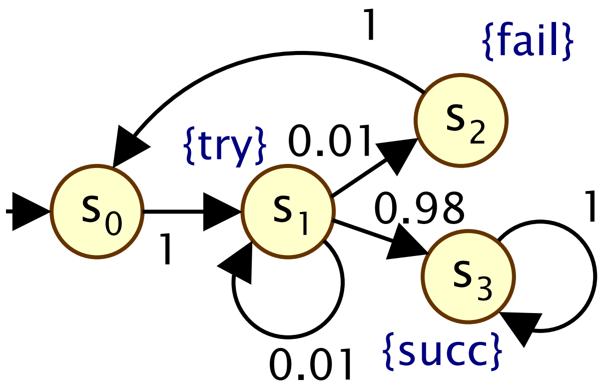


- Compute vector $\underline{\text{Prob}}(X\ \phi)$ of probabilities for all states s
  - $\underline{\text{Prob}}(X\ \phi) = \mathbf{P} \cdot \underline{\phi}$
  - where $\underline{\phi}$ is a 0-1 vector over S with $\underline{\phi}(s) = 1$ iff $s \vDash \phi$
  - computation requires a single matrix-vector multiplication

PCTL next – Example

- Model check: $P_{\geq 0.9}$ [X (¬try ∨ succ)]
  - Sat (¬try ∨ succ) = (S \ Sat(try)) ∪ Sat(succ)
    = $(\{s_0,s_1,s_2,s_3\} \setminus \{s_1\}) \cup \{s_3\} = \{s_0,s_2,s_3\}$
  - <u>Prob</u>(X (¬try ∨ succ)) = **P** · <u>(¬try ∨ succ)</u> = …

$$= \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.99 \\ 1 \\ 1 \end{bmatrix}$$



- Results:
  - <u>Prob</u>(X (¬try ∨ succ)) = [0, 0.99, 1, 1]
  - Sat($P_{\geq 0.9}$ [X (¬try ∨ succ)]) = $\{s_1, s_2, s_3\}$

PCTL bounded until for DTMCs

- Computation of probabilities for PCTL $U^{\leq k}$ operator
  - $Sat(P_{\sim p}[\ \phi_1\ U^{\leq k}\ \phi_2\ ]) = \{\ s \in S \mid Prob(s, \phi_1\ U^{\leq k}\ \phi_2) \sim p\ \}$
  - need to compute $Prob(s, \phi_1\ U^{\leq k}\ \phi_2)$ for all $s \in S$
- First identify states where probability is trivially 1 or 0
  - $S^{yes} = Sat(\phi_2)$
  - $S^{no} = S \setminus (Sat(\phi_1) \cup Sat(\phi_2))$
- Letting $S^? = S \setminus (S^{yes} \cup S^{no})$, compute solution of recursive equations:

$$Prob(s, \phi_1\ U^{\leq k}\ \phi_2) = \begin{cases} 1 & \text{if } s \in S^{yes} \\ 0 & \text{if } s \in S^{no} \\ 0 & \text{if } s \in S^? \text{ and } k = 0 \\ \sum_{s' \in S} P(s,s') \cdot Prob(s', \phi_1\ U^{\leq k-1}\ \phi_2) & \text{if } s \in S^? \text{ and } k > 0 \end{cases}$$

PCTL bounded until for DTMCs

- Simultaneous computation of vector $\underline{\text{Prob}}(\phi_1 \ U^{\leq k} \ \phi_2)$
  - i.e. probabilities $\text{Prob}(s, \phi_1 \ U^{\leq k} \ \phi_2)$ for all $s \in S$

- Iteratively define in terms of matrices and vectors
  - define matrix $\mathbf{P}'$ as follows: $\mathbf{P}'(s,s') = \mathbf{P}(s,s')$ if $s \in S^?$, $\mathbf{P}'(s,s') = 1$ if $s \in S^{yes}$ and $s=s'$, $\mathbf{P}'(s,s') = 0$ otherwise
  - $\underline{\text{Prob}}(\phi_1 \ U^{\leq 0} \ \phi_2) = \underline{\phi}_2$
  - $\underline{\text{Prob}}(\phi_1 \ U^{\leq k} \ \phi_2) = \mathbf{P}' \cdot \underline{\text{Prob}}(\phi_1 \ U^{\leq k-1} \ \phi_2)$
  - requires k matrix-vector multiplications

- Note that we could express this in terms of matrix powers
  - $\underline{\text{Prob}}(\phi_1 \ U^{\leq k} \ \phi_2) = (\mathbf{P}')^k \cdot \underline{\phi}_2$ and compute $(\mathbf{P}')^k$ in $\log_2 k$ steps
  - but this is actually inefficient: $(\mathbf{P}')^k$ is much less sparse than $\mathbf{P}'$

PCTL bounded until – Example

- Model check: $P_{>0.98}$ [$F^{\leq 2}$ succ] $\equiv$ $P_{>0.98}$ [true $U^{\leq 2}$ succ]
  - Sat (true) = S = $\{s_0,s_1,s_2,s_3\}$, Sat(succ) = $\{s_3\}$
  - $S^{yes} = \{s_3\}$, $S^{no} = \varnothing$, $S^? = \{s_0,s_1,s_2\}$, **P' = P**
  - $\underline{\text{Prob}}(\text{true } U^{\leq 0} \text{ succ}) = \underline{\text{succ}} = [0, 0, 0, 1]$

$$\underline{\text{Prob}}(\text{true } U^{\leq 1} \text{ succ}) = \mathbf{P'} \cdot \underline{\text{Prob}}(\text{true } U^{\leq 0} \text{ succ}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0.98 \\ 0 \\ 1 \end{bmatrix}$$

$$\underline{\text{Prob}}(\text{true } U^{\leq 2} \text{ succ}) = \mathbf{P'} \cdot \underline{\text{Prob}}(\text{true } U^{\leq 1} \text{ succ}) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0.98 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0.98 \\ 0.9898 \\ 0 \\ 1 \end{bmatrix}$$

  - Sat($P_{>0.98}$ [$F^{\leq 2}$ succ]) = $\{s_1, s_3\}$

PCTL unbounded until

- Computation of probabilities Prob(s, ϕ_1 U ϕ_2) for all s $\in$ S
- We first identify all states where the probability is 1 or 0
  - S^{yes} = Sat($P_{\geq 1}$ [ϕ_1 U ϕ_2])
  - S^{no} = Sat($P_{\leq 0}$ [ϕ_1 U ϕ_2])
- We refer to this as the "precomputation" phase
  - two precomputation algorithms: Prob0 and Prob1
- Important for several reasons
  - reduces the set of states for which probabilities must be computed numerically
  - for $P_{\sim p}[\cdot]$ where p is 0 or 1, no further computation required
  - gives exact results for the states in S^{yes} and S^{no} (no round-off)

Precomputation algorithms

- Prob0 algorithm to compute $S^{no} = Sat(P_{\leq 0} [\ \phi_1\ U\ \phi_2\ ])$:
  - first compute $Sat(P_{>0} [\ \phi_1\ U\ \phi_2\ ])$
  - i.e. find all states which can, with non-zero probability, reach a ϕ_2-state without leaving ϕ_1-states
  - i.e. find all states from which there is a finite path through ϕ_1-states to a ϕ_2-state: simple graph-based computation
  - subtract the resulting set from S

- Prob1 algorithm to compute $S^{yes} = Sat(P_{\geq 1} [\ \phi_1\ U\ \phi_2\ ])$:
  - first compute $Sat(P_{<1} [\ \phi_1\ U\ \phi_2\ ])$, reusing S^{no}
  - this is equivalent to the set of states which have a non-zero probability of reaching S^{no}, passing only through ϕ_1-states
  - again, this is a simple graph-based computation
  - subtract the resulting set from S

PCTL unbounded until

- Probabilities Prob(s, ϕ_1 U ϕ_2) can now be obtained as the unique solution of the following set of linear equations:

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

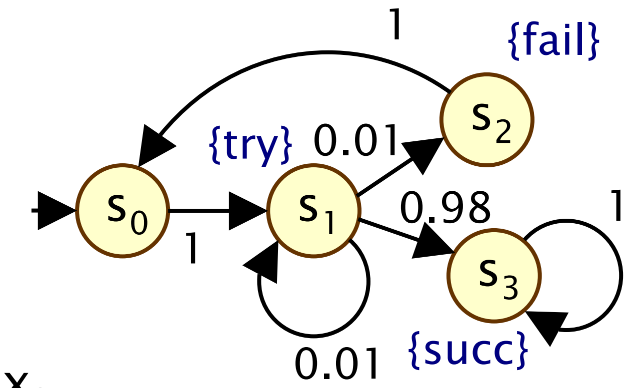
  - can be reduced to a system in $|S^?|$ unknowns instead of $|S|$

    $S^? = S \setminus (S^{yes} \cup S^{no})$

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

PCTL unbounded until – Example

- Model check: $P_{>0.99}$ [try U succ]
  - Sat(try) = $\{s_1\}$, Sat(succ) = $\{s_3\}$
  - S^{no} = Sat($P_{\leq 0}$ [try U succ]) = $\{s_0,s_2\}$
  - S^{yes} = Sat($P_{\geq 1}$ [try U succ]) = $\{s_3\}$
  - $S^{?} = \{s_1\}$
- Linear equation system:
  - $x_0 = 0$
  - $x_1 = 0.01 \cdot x_1 + 0.01 \cdot x_2 + 0.98 \cdot x_3$
  - $x_2 = 0$
  - $x_3 = 1$
- Which yields:
  - $\underline{\text{Prob}}$(try U succ) = $\underline{x}$ = [0, 98/99, 0, 1]
  - Sat($P_{>0.99}$ [try U succ]) = $\{s_3\}$



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, the non-probabilistic linear-time temporal logic
  - PCTL* [ASB+95,BdA95] which subsumes both PCTL and LTL

- These both allow combinations of temporal operators
  - e.g. for liveness: $P_{\sim p}$ [G F ϕ] – "always eventually ϕ"

- Model checking algorithms for DTMCs and PCTL* exist but are more expensive to implement (higher complexity)

Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

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 $(\underline{\rho},\iota)$
  - $\underline{\rho} : S \rightarrow \mathbb{R}_{\geq 0}$ is the state reward function (vector)
  - $\iota : S \times S \rightarrow \mathbb{R}_{\geq 0}$ 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, ι is not used
- Examples (for use with cumulative properties)
  - "time-steps": $\underline{\rho}$ returns 1 for all states and ι is zero (equivalently, $\underline{\rho}$ is zero and ι returns 1 for all transitions)
  - "number of messages lost": $\underline{\rho}$ is zero and ι 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 ι 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:
  - for a state s in the DTMC:
  - $s \vDash R_{\sim r} [\ I^{=k}\ ] \Leftrightarrow \mathrm{Exp}(s, X_{I=k}) \sim r$
  - $s \vDash R_{\sim r} [\ C^{\leq k}\ ] \Leftrightarrow \mathrm{Exp}(s, X_{C\leq k}) \sim r$
  - $s \vDash R_{\sim r} [\ F\ \Phi\ ] \Leftrightarrow \mathrm{Exp}(s, X_{F\Phi}) \sim r$

  where: Exp(s,X) denotes the expectation of the random variable $X : \mathrm{Path}(s) \rightarrow \mathbb{R}_{\geq 0}$ 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 $k_\phi = \min\{ j \mid s_j \vDash \phi \}$

Reward formula model checking

- Instantaneous: $R_{\sim r}$ [$I^{=k}$]
  - reduces to computation of bounded until probabilities
  - solution of recursive equations
- 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
  - reduces to solving a system of linear equation

Model checking PCTL summary

- Atomic propositions and logical connectives: trivial

- Probabilistic operator P:
  - $X\ \Phi$: one matrix-vector multiplications
  - $\Phi_1\ U^{\leq k}\ \Phi_2$: k matrix-vector multiplications
  - $\Phi_1\ U\ \Phi_2$: linear equation system in at most $|S|$ variables

- Expected reward operator R
  - $I^{=k}$: k matrix-vector multiplications
  - $C^{\leq k}$: k iterations of matrix-vector multiplication + summation
  - $F\ \Phi$: linear equation system in at most $|S|$ variables
  - details for the reward operators are in [KNP07a]

Model checking PCTL complexity

- Model checking of DTMC (S,s_{init},P,L) against PCTL formula Φ (including reward operators)
  - complexity is linear in $|\Phi|$ and polynomial in $|S|$

- Size $|\Phi|$ of Φ is defined as number of logical connectives and temporal operators plus sizes of temporal operators
  - model checking is performed for each operator
- Worst-case operators are $P_{\sim p}$ [Φ_1 U Φ_2] and $R_{\sim r}$ [F Φ]
  - main task: solution of linear equation system of size $|S|$
  - can be solved with Gaussian elimination: cubic in $|S|$
  - and also precomputation algorithms (max $|S|$ steps)
- Strictly speaking, $U^{\leq k}$ could be worse than U for large k
  - but in practice k is usually small

Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

Continuous-time Markov chains

- Continuous-time Markov chains (CTMCs)
  - labelled transition systems augmented with rates
  - discrete states and continuous time-steps

- Formally, a CTMC C is a tuple $(S, s_{init}, \mathbf{R}, L)$ where:
  - S is a finite set of states ("state space")
  - $s_{init} \in S$ is the initial state
  - $\mathbf{R} : S \times S \to \mathbb{R}_{\geq 0}$ is the transition rate matrix
  - $L : S \to 2^{AP}$ is a labelling with atomic propositions

- Transition rate matrix assigns rates to each pair of states
  - used as a parameter to the exponential distribution
  - transition between s and s' when $\mathbf{R}(s,s')>0$
  - probability triggered before t time units $1 - e^{-\mathbf{R}(s,s')\cdot t}$

Embedded DTMC

- Can determine the probability of each transition occurring
  - independent of the time at which it occurs
  - E(s) is the exit rate of state s $E(s) = \sum_{s' \in S} R(s,s')$
- Embedded DTMC: emb(C)=(S,s_{init},$\mathbf{P}^{emb(C)}$,L)
  - state space, initial state and labelling as the CTMC
  - for any s,s'∈S

$$\mathbf{P}^{emb(C)}(s,s') = \begin{cases} R(s,s')/E(s) & \text{if } E(s) > 0 \\ 1 & \text{if } E(s) = 0 \text{ and } s = s' \\ 0 & \text{otherwise} \end{cases}$$

- Alternative characterisation of the behaviour:
  - remain in s for delay exponentially distributed with rate E(s)
  - probability next state is s' is given by $\mathbf{P}^{emb(C)}(s,s')$

Continuous-time Markov chains

- Infinitesimal generator matrix

$$Q(s,s') = \begin{cases} -R(s,s') & s \neq s' \\ \sum_{s \neq s'} R(s,s') & \text{otherwise} \end{cases}$$

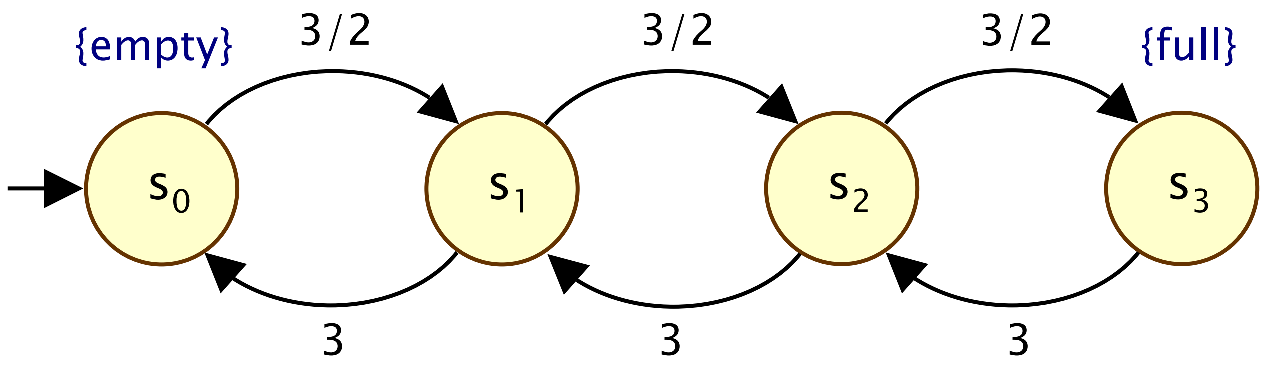
- Alternative definition: a CTMC is:
  - a family of random variables $\{ X(t) \mid t \in \mathbb{R}_{\geq 0} \}$
  - X(t) are observation made at time instant t
  - i.e. X(t) is the state of the system at time instant t

- Memoryless (Markov property)

  $P[X(t_k)=s_k \mid X(t_{k-1})=s_{k-1}, \ldots, X(t_0)=s_0] = P[X(t_k)=s_k \mid X(t_{k-1})=s_{k-1}]$

Simple CTMC example

- Modelling a queue of jobs
  - initially the queue is empty
  - jobs arrive with rate 3/2
  - jobs are served with rate 3
  - maximum size of the queue is 3

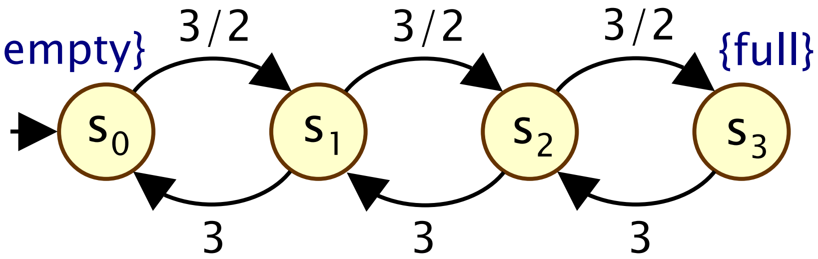


Simple CTMC example

$C = (S, s_{init}, R, L)$

$S = \{s_0, s_1, s_2, s_3\}$

$s_{init} = s_0$



$AP = \{empty, full\}$

$L(s_0)=\{empty\}$ $L(s_1)=L(s_2)=\varnothing$ and $L(s_3)=\{full\}$

$$\mathbf{R} = \begin{bmatrix} 0 & 3/2 & 0 & 0 \\ 3 & 0 & 3/2 & 0 \\ 0 & 3 & 0 & 3/2 \\ 0 & 0 & 3 & 0 \end{bmatrix} \quad \mathbf{P}^{emb(C)} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 2/3 & 0 & 1/3 & 0 \\ 0 & 2/3 & 0 & 1/3 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{Q} = \begin{bmatrix} -3/2 & 3/2 & 0 & 0 \\ 3 & -9/2 & 3/2 & 0 \\ 0 & 3 & -9/2 & 3/2 \\ 0 & 0 & 3 & -3 \end{bmatrix}$$

transition rate matrix

embedded DTMC

infinitesimal generator matrix

Paths of a CTMC

- Infinite path ω is a sequence $s_0t_0s_1t_1s_2t_2\ldots$ such that
  - $R(s_i,s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{>0}$ for all $i \in \mathbb{N}$
  - amount of time spent in the jth state: $time(\omega,j)=t_j$
  - state occupied at time t: $\omega@t=s_j$
    where j smallest index such that $\Sigma_{i\leq j}\, t_j \geq t$

- Finite path is a sequence $s_0t_0s_1t_1s_2t_2\ldots t_{k-1}s_k$ such that
  - $R(s_i,s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{>0}$ for all $i<k$
  - s_k is absorbing ($R(s,s') = 0$ for all $s' \in S$)
  - amount of time spent in the ith state only defined for $j\leq k$:
    $time(\omega,j)=t_j$ if $j<k$ and $time(\omega,j)=\infty$ if $j=k$
  - state occupied at time t: if $t\leq\Sigma_{i\leq k}\, t_j$ then $\omega@t$ as above
    otherwise $t>\Sigma_{i\leq k}\, t_j$ then $\omega@t=s_k$

Probability space

- Sample space: $Path_s$ (set of all paths from a state s)
- Events: sets of infinite paths
- Basic events: sets of paths with common finite prefix
  - probability of a single finite path is zero
  - include time intervals in cylinders

- Cylinder is a sequence $s_0,I_0,s_1,I_1,\ldots,I_{n-1},s_n$
  - $s_0,s_1,s_2,\ldots,s_n$ sequence of states where $\mathbf{R}(s_i,s_{i+1})>0$ for $i<n$
  - $I_0,I_1,I_2,\ldots,I_{n-1}$ sequence of of nonempty intervals of $\mathbb{R}_{\geq 0}$

- $C(s_0,I_0,s_1,I_1,\ldots,I_{n-1},s_n)$ set of (infinite and finite paths):
  - $\omega(i)=s_i$ for all $i \leq n$ and $time(\omega,i) \in I_i$ for all $i < n$

Probability space

- Define measure over cylinders by induction
  - $Pr_s(C(s))=1$
  - $Pr_s(C(s,I,s_1,I_1,\ldots,I_{n-1},s_n,I',s'))$ equals

$$Pr_s(C(s,I,s_1,I_1,\ldots,I_{n-1},s_n)) \cdot P^{emb(C)}(s_n,s') \cdot \left(e^{-E(s_n)\cdot \inf I'} - e^{-E(s_n)\cdot \sup I'}\right)$$

probability transition from s_n to s' (defined using embedded DTMC)

probability time spent in state s_n is within the interval I'

Probability space

- Probability space (Path(s), $\Sigma_{Path(s)}$, Pr_s)
- Sample space Ω = Path(s) (infinite and finite paths)
- Event set $\Sigma_{Path(s)}$
  - least σ-algebra on Path(s) containing all cylinders starting in s
- Probability measure Pr_s
  - Pr_s extends uniquely from probability defined over cylinders
- See [BHHK03] for further details

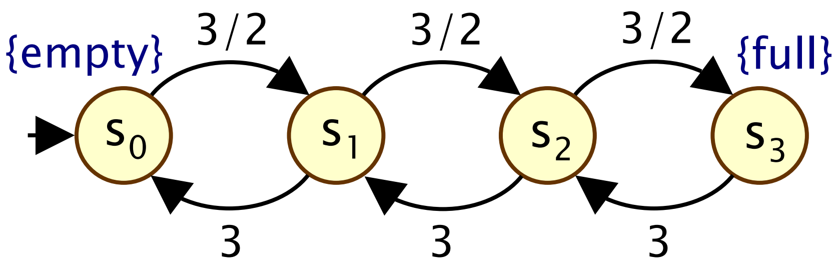
Probability space – Example

- Cylinder $C(s_0,[0,2],s_1)$

- $Pr(C(s_0,[0,2],s_1)) = Pr(C(s_0)) \cdot \mathbf{P}^{emb(C)}(s_0,s_1) \cdot (e^{-E(s0)\cdot 0} - e^{-E(s0)\cdot 2})$

$$= 1 \cdot 1 \cdot (e^{-3/2\cdot 0} - e^{-3/2\cdot 2})$$
$$= 1 - e^{-3}$$
$$\approx 0.95021$$

- Probability of leaving the initial state s_0 and moving to state s_1 within the first 2 time units of operation



Transient and steady-state behaviour

- Transient behaviour, C a CTMC
  - state of the model at a particular time instant
  - $\underline{\pi}^C_{s,t}(s')$ is probability of, having started in state s, being in state s' at time t
  - $\underline{\pi}^C_{s,t}(s') = Pr_s\{\ \omega \in Path^C(s) \mid \omega@t=s'\ \}$

- Steady-state behaviour
  - state of the model in the long-run
  - $\underline{\pi}^C_{s}(s')$ is probability of, having started in state s, being in state s' in the long run
  - $\underline{\pi}^C_{s}(s') = \lim_{t\to\infty} \underline{\pi}^C_{s,t}(s')$
  - the percentage of time, in long run, spent in each state

Computing transient probabilities

- Π_t – matrix of transient probabilities
  - $\Pi_t(s,s') = \underline{\pi}_{s,t}(s')$
- Π_t solution of the differential equation: $\Pi_t' = \Pi_t \cdot Q$
  - Q infinitesimal generator matrix
- Can be expressed as a matrix exponential and therefore evaluated as a power series

$$\Pi_t = e^{Q \cdot t} = \sum_{i=0}^{\infty} (Q \cdot t)^i / i!$$

  - computation potentially unstable
  - probabilities instead computed using the uniformised DTMC

Uniformisation

- Uniformised DTMC $\text{unif}(C)=(S,s_{init},\mathbf{P}^{\text{unif}(C)},L)$ of $C=(S,s_{init},\mathbf{R},L)$
  - set of states, initial state and labelling the same as C
  - $\mathbf{P}^{\text{unif}(C)} = \mathbf{I} + \mathbf{Q}/q$
  - $q \geq \max\{E(s) \mid s \in S\}$ is the uniformisation rate

- Each time step (epoch) of uniformised DTMC corresponds to one exponentially distributed delay with rate q
  - if $E(s)=q$ transitions the same as embedded DTMC (residence time has the same distribution as one epoch)
  - if $E(s)<q$ add self loop with probability $1-E(s)/q$ (residence time longer than $1/q$ so one epoch may not be 'long enough')

Uniformisation

- Using the uniformised DTMC the transient probabilities can be expressed by:

$$\Pi_t = e^{Q\cdot t} = e^{q\cdot(\mathbf{P}^{unif(C)} - \mathbf{I})\cdot t} = e^{(q\cdot t)\cdot \mathbf{P}^{unif(C)}} \cdot e^{-q\cdot t}$$

$$= e^{-q\cdot t} \cdot \left(\sum_{i=0}^{\infty} \frac{(q\cdot t)^i}{i!} \cdot \left(\mathbf{P}^{unif(C)} \right)^i \right)$$

$$= \sum_{i=0}^{\infty} \left(e^{-q\cdot t} \cdot \frac{(q\cdot t)^i}{i!} \right) \cdot \left(\mathbf{P}^{unif(C)} \right)^i$$

$$= \sum_{i=0}^{\infty} \gamma_{q\cdot t,i} \cdot \left(\mathbf{P}^{unif(C)} \right)^i$$

ith Poisson probability with parameter $q\cdot t$

$\mathbf{P}^{unif(C)}$ stochastic (all entries in [0,1] & rows sum to 1), therefore computations with **P** more numerically stable than **Q**.

Uniformisation

$$\Pi_t = \sum_{i=0}^{\infty} \gamma_{q\cdot t,i} \cdot \left(\mathbf{P}^{unif(C)}\right)^i$$

- $(\mathbf{P}^{unif(C)})^i$ is probability of jumping between each pair of states in i steps

- $\gamma_{q\cdot t,i}$ is the ith Poisson probability with parameter q·t
  - the probability of i steps occurring in time t, given each has delay exponentially distributed with rate q

- Can truncate the summation using the techniques of Fox and Glynn [FG88], which allow efficient computation of the Poisson probabilities

Uniformisation

- Computing $\underline{\pi}_{s,t}$ for a fixed state s and time t
  - can be computed efficiently using matrix-vector operations
  - pre-multiply the matrix Π_t by the initial distribution
  - in this $\underline{\pi}_{s,0}$ where $\underline{\pi}_{s,0}(s')$ equals 1 if s=s' and 0 otherwise

$$\underline{\pi}_{s,t} = \underline{\pi}_{s,0} \cdot \Pi_t = \underline{\pi}_{s,0} \cdot \sum_{i=0}^{\infty} \gamma_{q \cdot t,i} \cdot \left(\mathbf{P}^{unif(C)} \right)^i$$

$$= \sum_{i=0}^{\infty} \gamma_{q \cdot t,i} \cdot \underline{\pi}_{s,0} \cdot \left(\mathbf{P}^{unif(C)} \right)^i$$

  - compute iteratively to avoid the computation of matrix powers

$$\left(\underline{\pi}_{s,t} \cdot \mathbf{P}^{unif(C)} \right)^{i+1} = \left(\underline{\pi}_{s,t} \cdot \mathbf{P}^{unif(C)} \right)^{i} \cdot \mathbf{P}^{unif(C)}$$

Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

CSL

- Temporal logic for describing properties of CTMCs
  - CSL = Continuous Stochastic Logic [ASSB00,BHHK03]
  - extension of (non-probabilistic) temporal logic CTL
- Key additions:
  - probabilistic operator P (like PCTL)
  - steady state operator S
- Example: $\mathrm{down} \rightarrow \mathrm{P}_{>0.75}\ [\ \neg \mathrm{fail}\ \mathrm{U}^{\leq[1,2]}\ \mathrm{up}\ ]$
  - when a shutdown occurs, the probability of a system recovery being completed between 1 and 2 hours without further failure is greater than 0.75
- Example: $\mathrm{S}_{<0.1}[\mathrm{insufficient\_routers}]$
  - in the long run, the chance that an inadequate number of routers are operational is less than 0.1

CSL syntax

- CSL syntax:

ψ is true with probability ~p

  - ϕ ::= true | a | $\phi \wedge \phi$ | $\neg\phi$ | $P_{\sim p}$ [ψ] | $S_{\sim p}$ [ϕ]   (state formulas)
  - ψ ::= X ϕ | ϕ U^I ϕ (path formulas)

"next"

"time bounded until"

in the "long run" ϕ is true with probability ~p

  - where a is an atomic proposition, I interval of $\mathbb{R}_{\geq 0}$ and p ∈ [0,1], ~ ∈ {<,>,≤,≥}
- A CSL formula is always a state formula
  - path formulas only occur inside the P operator

CSL semantics for CTMCs

- CSL formulas interpreted over states of a CTMC
  - $s \models \phi$ denotes ϕ is "true in state s" or "satisfied in state s"

- Semantics of state formulas:
  - for a state s of the CTMC $(S, s_{init}, \mathbf{R}, L)$:

  - $s \models a$ $\Leftrightarrow$ $a \in L(s)$
  - $s \models \phi_1 \wedge \phi_2$ $\Leftrightarrow$ $s \models \phi_1$ and $s \models \phi_2$
  - $s \models \neg\phi$ $\Leftrightarrow$ $s \models \phi$ is false
  - $s \models P_{\sim p} [\psi]$ $\Leftrightarrow$ $\mathrm{Prob}(s, \psi) \sim p$
  - $s \models S_{\sim p} [\phi]$ $\Leftrightarrow$ $\Sigma_{s' \models \phi}\, \underline{\pi}_s(s') \sim p$

Probability of, starting in state s, satisfying the path formula ψ

Probability of, starting in state s, being in state s' in the long run

CSL semantics for CTMCs

- Prob(s, ψ) is the probability, starting in state s, of satisfying the path formula ψ
  - $\text{Prob}(s, \psi) = \text{Pr}_s \{\omega \in \text{Path}_s \mid \omega \vDash \psi \}$

if ω(0) is absorbing
ω(1) not defined

- Semantics of path formulas:
  - for a path ω of the CTMC:
  - $\omega \vDash X\ \phi \quad \Leftrightarrow \quad \omega(1)$ is defined and $\omega(1) \vDash \phi$
  - $\omega \vDash \phi_1\ U^I\ \phi_2 \quad \Leftrightarrow \quad \exists t \in I.\ (\ \omega@t \vDash \phi_2 \wedge \forall t'<t.\ \omega@t' \vDash \phi_1)$

there exists a time instant in the interval I where ϕ_2 is true and ϕ_1 is true at all preceding time instants

CSL derived operators

- (As for PCTL) can derive basic logical 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)

- The "eventually" operator (path formula)
  - F $\phi \equiv$ true U ϕ (F = "future")
  - sometimes written as $\lozenge\, \phi$ ("diamond")
  - "ϕ is eventually true"
  - timed version: $F^I\, \phi \equiv$ true $U^I\, \phi$
  - "ϕ becomes true in the interval I"

More on CSL

- Negation and probabilities
  - $\neg P_{>p} [\ \phi_1\ U^I\ \phi_2\ ] \equiv P_{\leq p} [\phi_1\ U^I\ \phi_2\ ]$
  - $\neg S_{>p} [\ \phi\ ] \equiv S_{\leq p} [\ \phi\ ]$

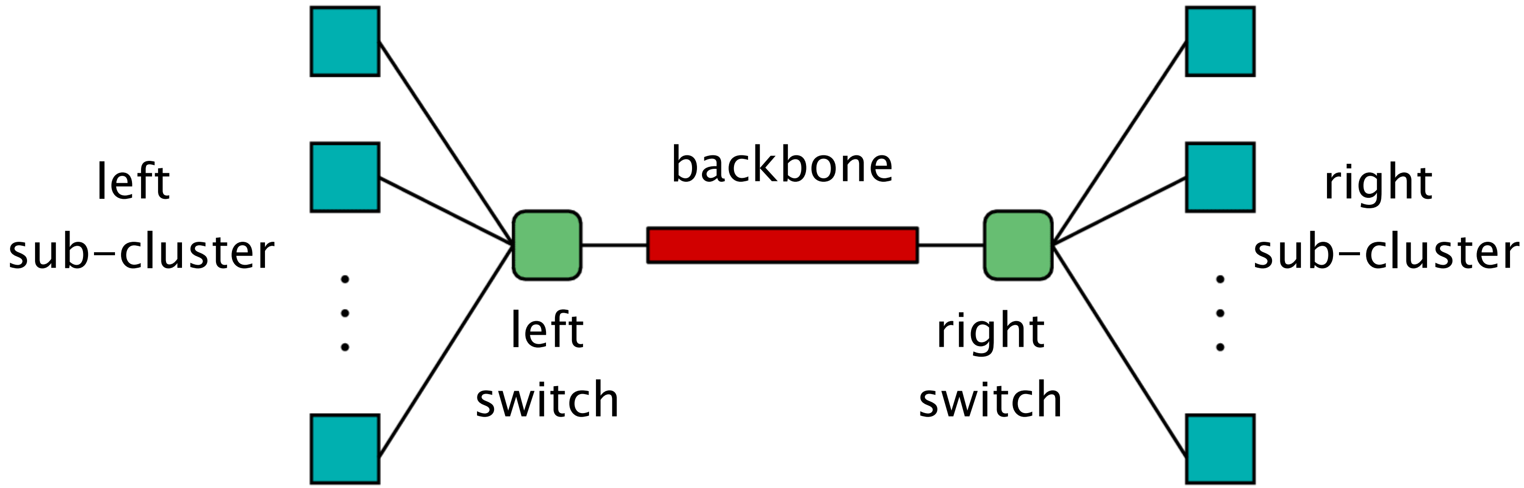
- The "always" operator (path formula)
  - $G\ \phi \equiv \neg(F\ \neg\phi) \equiv \neg(\text{true}\ U\ \neg\phi)$ (G = "globally")
  - sometimes written as $\Box\ \phi$ ("box")
  - "ϕ is always true"
  - bounded version: $G^I\ \phi \equiv \neg(F^I\ \neg\phi)$
  - "ϕ holds throughout the interval I"
  - strictly speaking, $G\ \phi$ cannot be derived from the CSL syntax in this way since there is no negation of path formulas
  - but, as for PCTL, we can derive $P_{\sim p}$ [$G\ \phi$] directly...

Quantitative properties

- Consider CSL formulae $P_{\sim p}$ [ψ] and $S_{\sim p}$ [ϕ]
  - if the probability is unknown, how to choose the bound p?
- When the outermost operator of a CSL formula is P or S
  - allow bounds of the form $P_{=?}$ [ψ] and $S_{=?}$ [ϕ]
  - what is the probability that path formula ψ is true?
  - what is the long-run probability that ϕ holds?
- Model checking is no harder: compute the values anyway

CSL example – Workstation cluster

- Case study: Cluster of workstations [HHK00]
  - two sub-clusters (N workstations in each cluster)
  - star topology with a central switch
  - components can break down, single repair unit
  - minimum QoS: at least ¾ of the workstations operational and connected via switches
  - premium QoS: all workstations operational and connected via switches



CSL example – Workstation cluster

- $P_{=?}[\text{true } U^{[0,t]} \neg\text{minimum}]$
  - the chance that the QoS drops below minimum within t hours
- $\neg\text{minimum} \rightarrow P_{<0.1}[F^{[0,t]} \neg\text{minimum}]$
  - when facing insufficient QoS, the probability of facing the same problem after t hours is less than 0.1
- $S_{=?}[\text{ minimum }]$
  - the probability in the long run of having minimum QoS
- $\text{minimum} \rightarrow P_{>0.8}[\text{minimum } U^{[0,t]} \text{ premium}]$
  - the probability of going from minimum to premium QoS within t hours without violating minimum QoS is at least 0.8
- $P_{=?}[\neg\text{minimum } U^{[t,\infty)} \text{ minimum}]$
  - the chance it takes more than t time units to recover from insufficient QoS

Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

CSL model checking

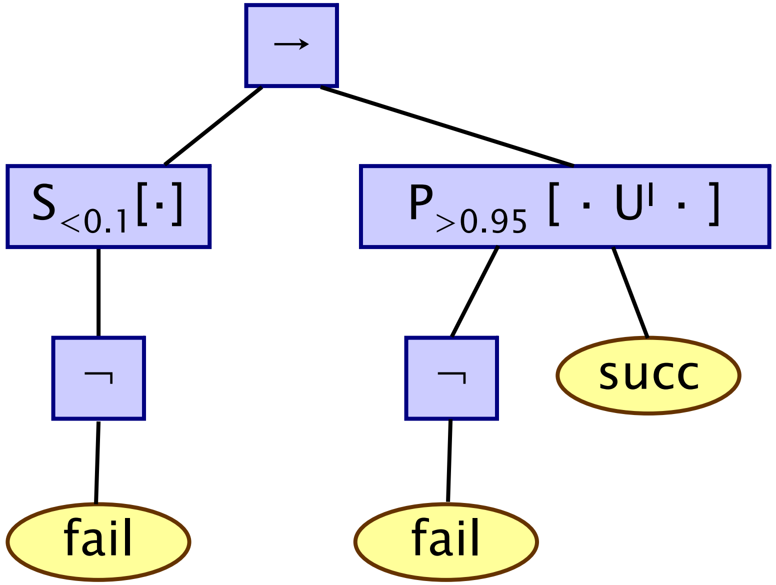
- Algorithm for CSL model checking [BHHK03]
  - inputs: CTMC $C=(S,s_{init},R,L)$, CSL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \vDash \phi \}$, the set of states satisfying ϕ

- What does it mean for a CTMC C to satisfy a formula ϕ?
  - check that $s \vDash \phi$ for all states $s \in S$, i.e. $Sat(\phi) = S$
  - know if $s_{init} \vDash \phi$, i.e. if $s_{init} \in Sat(\phi)$

- Sometimes, focus on quantitative results
  - e.g. compute result of $P=?$ [true $U^{[0,13.5]}$ minimum]
  - e.g. compute result of $P=?$ [true $U^{[0,t]}$ minimum] for $0 \leq t \leq 100$

CSL model checking

- Basic algorithm proceeds by induction on parse tree of ϕ
  - example: $\phi = S_{<0.9}[\neg fail\ ] \rightarrow P_{>0.95}\ [\ \neg fail\ U^I\ succ\ ]$



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

Untimed properties

- Untimed properties can be verified on the embedded DTMC
  - properties of the form: $P_{\sim p}$ [X ϕ] or $P_{\sim p}$ [ϕ_1 $U^{[0,\infty)}$ ϕ_2]
  - use algorithms for checking PCTL against DTMCs
- Certain qualitative time-bounded until formulae can also be verified on the embedded DTMC
  - for any (non-empty) interval I

    $s \models P_{\sim 0}$ [ϕ_1 U^{I} ϕ_2] if and only if $s \models P_{\sim 0}$ [ϕ_1 $U^{[0,\infty)}$ ϕ_2]

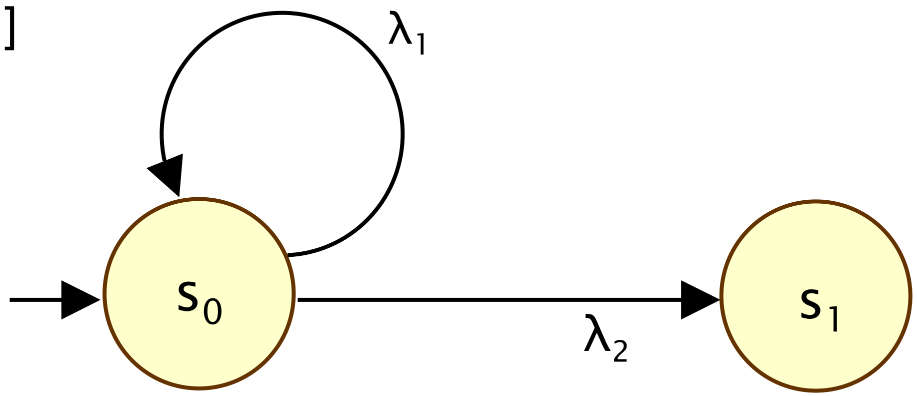
  - can use pre-computation algorithm Prob0

Untimed properties

- $s \vDash P_{\sim 1} [\phi_1 \; U^{[0,\infty)} \; \phi_2]$ does not imply $s \vDash P_{\sim 1} [\phi_1 \; U^{I} \; \phi_2]$

- Consider the following example
  - with probability 1 eventually reach state s_1

    $s_0 \vDash P_{\geq 1} [\phi_1 \; U^{[0,\infty)} \; \phi_2]$
  - probability of remaining in state s_0 until time-bound t is greater than zero for any t
  - $s_0 \vDash \neg P_{\geq 1} [\phi_1 \; U^{[0,t]} \; \phi_2]$



Model checking – Time-bounded until

- Compute Prob(s, $\phi_1 \; U^I \; \phi_2$) for all states where I is an arbitrary interval of the non-negative real numbers

  - Prob(s, $\phi_1 \; U^I \; \phi_2$) = Prob(s, $\phi_1 \; U^{cl(I)} \; \phi_2$)
    where cl(I) closure of the interval I

  - Prob(s, $\phi_1 \; U^{[0,\infty)} \; \phi_2$) = $Prob^{emb(C)}$(s, $\phi_1 \; U \; \phi_2$)
    where emb(C) is the embedded DTMC

- Therefore, remains to consider the cases when
  - I = [0,t] for some $t \in \mathbb{R}_{\geq 0}$
  - I = [t,t'] for some $t,t' \in \mathbb{R}_{\geq 0}$ such that $t \leq t'$
  - I = $[t,\infty)$ for some $t \in \mathbb{R}_{\geq 0}$

Model checking – $P_{\sim p}[\phi_1\ U^{[0,t]}\ \phi_2]$

- Computing the probabilities reduces to determining the least solution of the following set of integral equations:

- $Prob(s, \phi_1\ U^{[0,t]}\ \phi_2)$ equals
  - 1 if $s \in Sat(\phi_2)$,
  - 0 if $s \in Sat(\neg\phi_1 \wedge \neg\phi_2)$
  - and otherwise equals

probability in state s' of satisfying until before t–x time units elapse

$$\int_0^t \left(P^{emb(C)}(s,s') \cdot E(s) \cdot e^{-E(s)\cdot x} \right) \cdot Prob(s', \phi_1\ U^{[0,t-x]}\ \phi_2)\, dx$$

integrate over x between 0 and t

probability of moving from s to s' at time x

Model checking – $P_{\sim p}[\phi_1 \ U^{[0,t]} \ \phi_2]$

- Construct CTMC $C[\phi_2][\neg\phi_1 \wedge \neg\phi_2]$
  - where for CTMC $C=(S,s_{init},\mathbf{R},L)$, let $C[\theta]=(S,s_{init},\mathbf{R}[\theta],L)$ where $\mathbf{R}[\theta](s,s')=\mathbf{R}(s,s')$ if $s \notin Sat(\theta)$ and 0 otherwise
- Make all ϕ_2 states absorbing
  - in such a state $\phi_1 \ U^{[0,x]} \ \phi_2$ holds with probability 1
- Make all $\neg\phi_1 \wedge \neg\phi_2$ states absorbing
  - in such a state $\phi_1 \ U^{[0,x]} \ \phi_2$ holds with probability 0
- Problem then reduces to calculating transient probabilities of the CTMC $C[\phi_2][\neg\phi_1 \wedge \neg\phi_2]$:

$$Prob(s,\phi_1 \ U^{[0,t]} \ \phi_2) = \sum_{s' \in Sat(\phi_2)} \underline{\pi}_{s,t}^{C[\phi_2][\neg\phi_1 \wedge \neg\phi_2]}(s')$$

transient probability: starting in state the probability of being in state s' at time t

Model checking – $P_{\sim p}[\phi_1\ U^{[0,t]}\ \phi_2]$

- Can now adapt uniformisation to computing the vector of probabilities $\underline{\text{Prob}}(\ \phi_1\ U^{[0,t]}\ \phi_2)$
  - recall Π_t is matrix of transient probabilities $\Pi_t(s,s') = \underline{\pi}_{s,t}(s')$
  - computed via uniformisation: $\Pi_t = \sum_{i=0}^{\infty} \gamma_{q \cdot t,i} \cdot \left(P^{unif(C)} \right)^i$
- Combining with: $\text{Prob}(s, \phi_1\ U^{[0,t]}\ \phi_2) = \sum_{s' \in Sat(\phi_2)} \underline{\pi}_{s,t}^{C[\phi_2][\neg\phi_1 \wedge \neg\phi_2]}(s')$

$$
\begin{aligned}
\underline{\text{Prob}}(\phi_1\ U^{[0,t]}\ \phi_2) &= \Pi_t^{C[\phi_2][\neg\phi_1 \wedge \neg\phi_2]} \cdot \underline{\phi_2} \\
&= \left(\sum_{i=0}^{\infty} \gamma_{q \cdot t,i} \cdot \left(P^{unif(C[\phi_2][\neg\phi_1 \wedge \neg\phi_2])} \right)^i \right) \cdot \underline{\phi_2} \\
&= \sum_{i=0}^{\infty} \left(\gamma_{q \cdot t,i} \cdot \left(P^{unif(C[\phi_2][\neg\phi_1 \wedge \neg\phi_2])} \right)^i \cdot \underline{\phi_2} \right)
\end{aligned}
$$

Model checking – $P_{\sim p}[\phi_1 \ U^{[0,t]} \ \phi_2]$

- Have shown that we can calculate the probabilites as:

$$\underline{Prob}(\phi_1 \ U^{[0,t]} \ \phi_2) = \sum_{i=0}^{\infty} \left(\gamma_{q \cdot t, i} \cdot \left(\mathbf{P}^{unif(C[\phi_2][\neg\phi_1 \wedge \neg\phi_2])} \right)^i \cdot \underline{\phi_2} \right)$$

- Infinite summation can be truncated using the techniques of Fox and Glynn [FG88]

- Can compute iteratively to avoid matrix powers:

$$\left(\mathbf{P}^{unif(C)} \right)^0 \cdot \underline{\phi_2} = \underline{\phi_2}$$

$$\left(\mathbf{P}^{unif(C)} \right)^{i+1} \cdot \underline{\phi_2} = \mathbf{P}^{unif(C)} \cdot \left(\left(\mathbf{P}^{unif(C)} \right)^i \cdot \underline{\phi_2} \right)$$

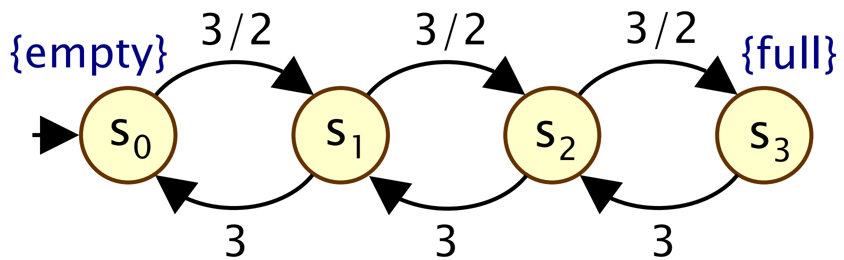
$P_{\sim p}[\phi_1 \ U^{[0,t]} \ \phi_2]$ – Example

- $P_{>0.65}$[true $U^{[0,7.5]}$ full]
  - "probability of the queue becoming full within 7.5 time units"
- State s_3 satisfies full and no states satisfy ¬true
  - in C[full][¬true ∧¬ full] only state s_3 made absorbing

$$\begin{bmatrix} 2/3 & 1/3 & 0 & 0 \\ 2/3 & 0 & 1/3 & 0 \\ 0 & 2/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

matrix of unif(C[full][¬true ∧¬full]) with uniformisation rate $\max_{s \in S} E(s) = 4.5$

s_3 made absorbing



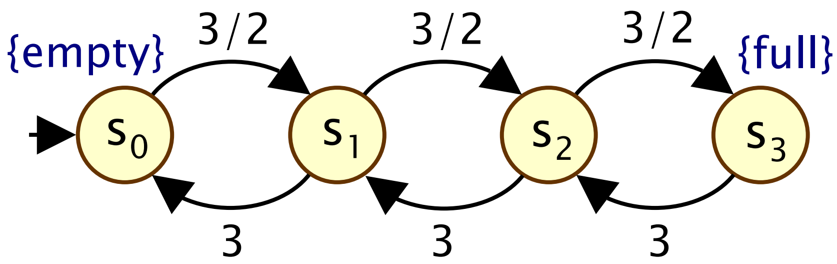
$P_{\sim p}[\phi_1 \ U^{[0,t]} \ \phi_2]$ – Example

- Computing the summation of matrix-vector multiplications

$$\underline{Prob}(\phi_1 \ U^{[0,t]} \ \phi_2) = \sum_{i=0}^{\infty} \left(\gamma_{q \cdot t,i} \cdot \left(P^{unif(C[\phi_2][\neg\phi_1 \wedge \neg\phi_2])} \right)^i \cdot \underline{\phi_2} \right)$$

  – yields $\underline{Prob}$(true $U^{[0,7.5]}$full) ≈ (0.6482,0.6823,0.7811,1)

- $P_{>0.65}$[true $U^{[0,7.5]}$ full] satisfied in states s_1, s_2 and s_3



Model checking – $P_{\sim p}[\phi_1 \ U^{[t,t']} \ \phi_2]$

- In this case the computation can be split into two parts:
- Probability of remaining in ϕ_1 states until time t
  - can be computed as transient probabilities on the CTMC where are states satisfying $\neg\phi_1$ have been made absorbing
- Probability of reaching a ϕ_2 state, while remaining in states satisfying ϕ_1, within the time interval [0,t'–t]
  - i.e. computing Prob($\phi_1 \ U^{[0,t'-t]} \ \phi_2$)

$$\mathrm{Prob}(s, \phi_1 \ U^{[0,t]} \ \phi_2) = \sum_{s' \in Sat(\phi_1)} \underline{\pi}_{s,t}^{C[\neg\phi_1]}(s') \cdot \mathrm{Prob}(s', \phi_1 \ U^{[0,t'-t]} \ \phi_2)$$

sum over states satisfying ϕ_1

Probability of reaching state s' at time t and satisfying ϕ_1 up until this point

probability $\phi_1 \ U^{[t,t']} \ \phi_2$ holds in s'

Model checking – $P_{\sim p}[\phi_1 \ U^{[t,t']} \ \phi_2]$

- Letting $\mathrm{Prob}_{\phi}(s, \phi_1 U^{[0,t]} \phi_2) = \mathrm{Prob}(s, \phi_1 U^{[0,t]} \phi_2)$ if $s \in \mathrm{Sat}(\phi)$ and 0 otherwise, from the previous slide we have:

$$\underline{\mathrm{Prob}}(\phi_1 \ U^{[0,t]} \ \phi_2) = \Pi_t^{C[\neg\phi_1]}(s') \cdot \underline{\mathrm{Prob}}(\phi_1 \ U^{[0,t'-t]} \ \phi_2)$$

$$= \left(\sum_{i=0}^{\infty} \gamma_{q \cdot t, i} \cdot \left(P^{\mathrm{unif}(C[\neg\phi_1])} \right)^i \right) \cdot \underline{\mathrm{Prob}}_{\phi_1}(\phi_1 \ U^{[0,t'-t]} \ \phi_2)$$

$$= \sum_{i=0}^{\infty} \left(\gamma_{q \cdot t, i} \cdot \left(P^{\mathrm{unif}(C[\neg\phi_1])} \right)^i \cdot \underline{\mathrm{Prob}}_{\phi_1}(\phi_1 \ U^{[0,t'-t]} \ \phi_2) \right)$$

  – summation can be truncated using Fox and Glynn [FG88]
  – can compute iteratively (only scalar and matrix-vector operations)

Model checking – $P_{\sim p}[\phi_1\ U^{[t,\infty)}\ \phi_2]$

- Similar to the case for $\phi_1\ U^{[t,t']}\ \phi_2$ except second part is now unbounded, and hence the embedded DTMC can be used
- Probability of remaining in ϕ_1 states until time t
- Probability of reaching a ϕ_2 state, while remaining in states satisfying ϕ_1
  - i.e. computing Prob($\phi_1\ U^{[0,\infty)}\ \phi_2$)

$$\mathrm{Prob}(s, \phi_1\ U^{[0,t]}\ \phi_2) = \sum_{s' \in Sat(\phi_1)} \underline{\pi}^{C[\neg\phi_1]}_{s,t}(s') \cdot \mathrm{Prob}^{emb(C)}(s', \phi_1\ U\ \phi_2)$$

sum over states satisfying ϕ_1

Probability of reaching state s' at time t and satisfying ϕ_1 up until this point

probability $\phi_1\ U^{[0,\infty)}\ \phi_2$ holds in s'

Model checking – $P_{\sim p}[\phi_1\ U^{[t,\infty)}\ \phi_2]$

- Letting $Prob_\phi(s, \phi_1 U^{[0,t]}\phi_2) = Prob(s, \phi_1 U^{[0,t]}\phi_2)$ if $s \in Sat(\phi)$ and 0 otherwise, from the previous slide we have:

$$
\begin{aligned}
\underline{Prob}(\phi_1\ U^{[0,t]}\ \phi_2) &= \Pi_t^{C[\neg\phi_1]}(s') \cdot \underline{Prob}^{emb(C)}(\phi_1\ U\ \phi_2) \\
&= \left(\sum\nolimits_{i=0}^{\infty} \gamma_{q\cdot t,i} \cdot \left(P^{unif(C[\neg\phi_1])} \right)^i \right) \cdot \underline{Prob}^{emb(C)}(\phi_1\ U\ \phi_2) \\
&= \sum\nolimits_{i=0}^{\infty} \left(\gamma_{q\cdot t,i} \cdot \left(P^{unif(C[\neg\phi_1])} \right)^i \cdot \underline{Prob}^{emb(C)}(\phi_1\ U\ \phi_2) \right)
\end{aligned}
$$

  - summation can be truncated using Fox and Glynn [FG88]
  - can compute iteratively (only scalar and matrix-vector opertions

Model Checking – $S_{\sim p}[\ \phi\ ]$

- A state s satisfies the formula $S_{\sim p}[\phi]$ if $\sum_{s' \models \phi} \underline{\pi}^C_s(s') \sim p$
  - $\underline{\pi}^C_s(s')$ is probability, having started in state s, of being in state s' in the long run

- First, consider the simple case when C is irreducible
  - C is irreducible (strongly connected) if there exists a finite path from each state to every other state
  - the steady-state probabilities are independent of the starting state: denote the steady state probabilities by $\underline{\pi}^C(s')$
  - these probabilities can be computed as the unique solution of the linear equation system:

$$\underline{\pi}^C \cdot \mathbf{Q} = \underline{0} \quad \text{and} \quad \sum_{s \in S} \underline{\pi}^C(s) = 1$$

Q is the infinitesimal generator matrix of C

Model Checking – $S_{\sim p}[\ \phi\ ]$

- Equation system can be solved by any standard approach
  - Direct methods, such as Gaussian elimination
  - Iterative methods, such as Jacobi and Gauss-Seidel
- The satisfaction of the CSL formula
  - same for all states (steady state independent of starting state)
  - computed by summing steady state probabilities for all states satisfying ϕ

Model Checking – $S_{\sim p}[\ \phi\ ]$

- We now suppose that C is reducible

- First perform graph analysis to find set bssc(C) of bottom strongly connected components (BSCCs)
  - strongly connected components that cannot be left

- Treating each individual $B \in$ bscc(C) as an irreducible CTMC compute the steady state probabilities $\underline{\pi}^B$
  - employ the methods described above

- Calculate the probability of reaching each individual BSCC
  - can be computed in the embedded DTMC
  - if a_B is an atomic proposition true only in the states of B, this probability is given by $Prob^{emb(C)}(s, F\ a_B)$

Model Checking – $S_{\sim p}[\ \phi\ ]$

- For any states s and s' the steady state probability $\underline{\pi}^C_s(s')$ can then be computed as:

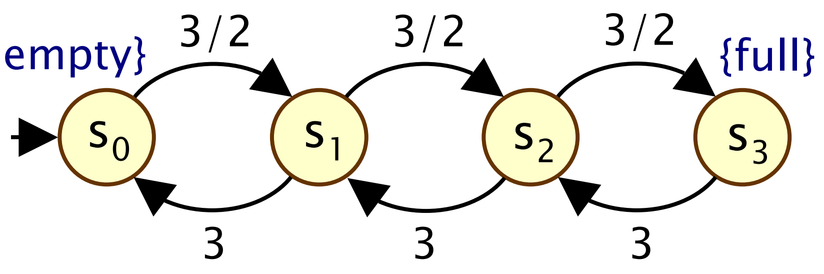
$$\pi^C_s(s') = \begin{cases} Prob^{emb(C)}(s, F\ a_B) \cdot \underline{\pi}^B(s') & \text{if } s' \in B \text{ for some } B \in bscc(C) \\ 0 & \text{otherwise} \end{cases}$$

- The total work required to compute $\underline{\pi}^C_s(s')$ for all s and s'
  - solve two linear equation systems for each BSCC B
    - one to obtain the vector $\underline{Prob}^{emb(C)}(F\ a_B)$
    - the other to compute the steady state probabilities $\underline{\pi}^B$
  - computation of the BSCCs requires only analysis of the underlying graph structure and can be performed using classical algorithms based on depth-first search

$S_{\sim p}[\ \phi\ ]$ – Example

- $S_{<0.1}$[full]
- CTMC is irreducible (comprises of a single BSCC)
  - steady state probabilities independent of starting state
  - can be computed by solving $\underline{\pi} \cdot \mathbf{Q} = 0$ and $\Sigma\ \underline{\pi}(s) = 1$

$$\mathbf{Q} = \begin{bmatrix} -3/2 & 3/2 & 0 & 0 \\ 3 & -9/2 & 3/2 & 0 \\ 0 & 3 & -9/2 & 3/2 \\ 0 & 0 & 3 & -3 \end{bmatrix}$$



$S_{\sim p}[\ \phi\ ]$ – Example

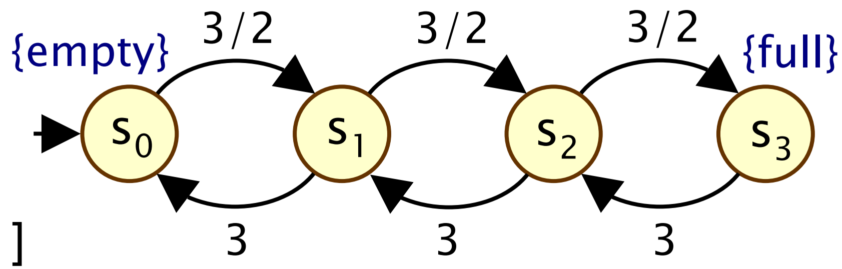
$$
\begin{array}{rcrcrcrcl}
-3/2\cdot\underline{\pi}(s_0) & + & 3\cdot\underline{\pi}(s_1) & & & & & = & 0\\
3/2\cdot\underline{\pi}(s_0) & - & 9/2\cdot\underline{\pi}(s_1) & + & 3\cdot\underline{\pi}(s_2) & & & = & 0\\
 & & 3/2\cdot\underline{\pi}(s_1) & - & 9/2\cdot\underline{\pi}(s_2) & + & 3\cdot\underline{\pi}(s_3) & = & 0\\
 & & & & 3/2\cdot\underline{\pi}(s_2) & - & 3\cdot\underline{\pi}(s_3) & = & 0
\end{array}
$$

$$
\underline{\pi}(s_0) + \underline{\pi}(s_1) + \underline{\pi}(s_2) + \underline{\pi}(s_3) = 1
$$

– solution: $\underline{\pi}=(8/15,4/15,2/15,1/15)$

– $\Sigma_{s' \models \text{full}}\ \underline{\pi}\,(s') = 1/15 < 0.1$

– so all states satisfy $S_{<0.1}[\text{ full }]$



Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

Costs and rewards

- We augment CTMCs with rewards
  - real-valued quantities assigned to states and/or transitions
  - these can have a wide range of possible interpretations
  - allows a wide range of quantitative measures of the system
  - basic notion: expected value of rewards
  - formal property specifications in an extension of CSL

- For a CTMC (S, s_{init}, R, L), a reward structure is a pair $(\underline{\rho}, \iota)$
  - $\underline{\rho} : S \rightarrow \mathbb{R}_{\geq 0}$ is a vector of state rewards
  - $\iota : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is a matrix of transition rewards

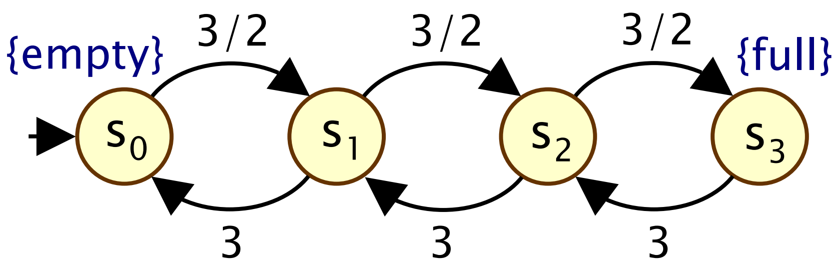
  - continuous time: reward $t \cdot \underline{\rho}(s)$ acquired if the CTMC remains in state s for $t \in \mathbb{R}_{\geq 0}$ time units

Reward structures – Example

- Example: “number of requests served”

$$\rho = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad \text{and} \quad \iota = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- Example: “size of message queue”
  - $\underline{\rho}(s_i)=i$ and $\iota(s_i,s_j)=0$ for all states s_i and s_j



CSL and rewards

- Extend CSL to incorporate reward-based properties
  - add R operator similar to the one in PCTL

expected reward is ~r

  - $\phi \ ::= \ \ldots \ | \ R_{\sim r} \, [\, I^{=t} \,] \ | \ R_{\sim r} \, [\, C^{\leq t} \,] \ | \ R_{\sim r} \, [\, F \ \phi \,] \ | \ R_{\sim r} \, [\, S \,]$

"instantaneous" "cumulative" "reachability" "steady-state"

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

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

Types of reward formulas

- Instantaneous: $R_{\sim r}$ [$I^{=t}$]
  - the expected value of the reward at time-instant t is ~r
  - "the expected queue size after 6.7 seconds is at most 2"
- Cumulative: $R_{\sim r}$ [$C^{\leq t}$]
  - the expected reward cumulated up to time-instant t is ~r
  - "the expected requests served within the first 4.5 seconds of operation is less than 10"
- Reachability: $R_{\sim r}$ [F ϕ]
  - the expected reward cumulated before reaching ϕ is ~r
  - "the expected requests served before the queue becomes full"
- Steady-state $R_{\sim r}$ [S]
  - the long-run average expected reward is ~r
  - "expected long-run queue size is at least 1.2"

Reward formula semantics

- Formal semantics of the four reward operators:
  - $s \models R_{\sim r}\ [\ I^{=t}\ ]$ $\Leftrightarrow$ $Exp(s, X_{I=t}) \sim r$
  - $s \models R_{\sim r}\ [\ C^{\leq t}\ ]$ $\Leftrightarrow$ $Exp(s, X_{C\leq t}) \sim r$
  - $s \models R_{\sim r}\ [\ F\ \Phi\ ]$ $\Leftrightarrow$ $Exp(s, X_{F\Phi}) \sim r$
  - $s \models R_{\sim r}\ [\ S\ ]$ $\Leftrightarrow$ $\lim_{t\to\infty}(\ 1/t \cdot Exp(s, X_{C\leq t})\ ) \sim r$
- where:
  - $Exp(s, X)$ denotes the expectation of the random variable $X : Path(s) \to \mathbb{R}_{\geq 0}$ with respect to the probability measure Pr_s

Reward formula semantics

- Definition of random variables:
  - path $\omega = s_0 t_0 s_1 t_1 s_2 \ldots$

state of ω at t

time spent in state s_{j_t} before t time units have elapsed

time spent in state s_i

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

$$X_{C\leq t}(\omega) = \sum_{i=0}^{j_t - 1} \left(t_i \cdot \underline{\rho}(s_i) + \iota(s_i, s_{i+1}) \right) + \left(t - \sum_{i=0}^{j_t - 1} t_i \right) \cdot \underline{\rho}(s_{j_t})$$

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

  - where $j_t = \min\{ j \mid \Sigma_{i \leq j} t_i \geq t \}$ and $k_\phi = \min\{ i \mid s_i \vDash \phi \}$

Model checking reward formulas

- Instantaneous: $R_{\sim r}$ [$I^{=t}$]
  - reduces to transient analysis (state of the CTMC at time t)
  - use uniformisation
- Cumulative: $R_{\sim r}$ [$C^{\leq t}$]
  - extends approach for time-bounded until [KNP06]
  - based on uniformisation
- Reachability: $R_{\sim r}$ [F ϕ]
  - can be computed on the embedded DTMC
  - reduces to solving a system of linear equation
- Steady-state: $R_{\sim r}$ [S]
  - similar to steady state formulae $S_{\sim r}$ [ϕ]
  - graph based analysis (compute BSCCs)
  - solve systems of linear equations (compute steady state probabilities of each BSCC)

Model checking complexity

- For model checking of a CTMC complexity:
  - linear in $|\Phi|$ and polynomial in $|S|$
  - linear in $q \cdot t_{max}$ (t_{max} is maximum finite bound in intervals)
- $P_{\sim p}[\Phi_1 \; U^{[0,\infty)} \; \Phi_2]$, $S_{\sim p}[\Phi]$, $R_{\sim r}$ [F Φ] and $R_{\sim r}$ [S]
  - require solution of linear equation system of size $|S|$
  - can be solved with Gaussian elimination: cubic in $|S|$
  - precomputation algorithms (max $|S|$ steps)
- $P_{\sim p}[\Phi_1 \; U^I \; \Phi_2]$, $R_{\sim r}$ $[C^{\leq t}]$ and $R_{\sim r}$ $[I^{=t}]$
  - at most two iterative sequences of matrix-vector product
  - operation is quadratic in the size of the matrix, i.e. $|S|$
  - total number of iterations bounded by Fox and Glynn
  - the bound is linear in the size of $q \cdot t$ (q uniformisation rate)

Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

The PRISM tool

- PRISM: Probabilistic symbolic model checker
  - developed at the University of Birmingham, since 1999
  - free, open source (GPL)
  - versions for Linux, Unix, Mac OS X, Windows, 64-bit OSs

- Modelling of:
  - DTMCs, MDPs, CTMCs + costs/rewards
- Verification of:
  - PCTL, CSL + extensions + costs/rewards
- Features:
  - high-level modelling language, wide range of model analysis methods, graphical user interface, efficient implementation

Getting PRISM + Other Resources

- PRISM website: www.cs.bham.ac.uk/~dxp/prism
  - tool download: binaries, source code (GPL)
  - on-line example repository (40+ case studies)
  - on-line documentation:
    - PRISM manual
    - PRISM tutorial
  - support: help forum, bug tracking, feature requests
    - hosted on Sourceforge
  - related publications, talks, tutorials, links

PRISM – Model building

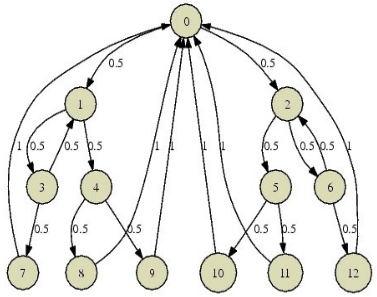
- First step of verification = construct full probabilistic model (not always necessary in non-probabilistic model checking)

High-level model

```
// Hermans self-stabilisation algorithm [Her90]

dtmc // Algorithm is fully synchronous

module process1 // first of N=5 symmetric processes

    x1 : [0..1]; // one bit per process; xi=x(i-1) means proc i
    [step] (x1=x5) -> 0.5 : (x1'=0) + 0.5 : (x1'=1);
    [step] !x1=x5 -> (x1'=x5);

endmodule

// Add further processes through renaming
module process2 = process1 [ x1=x2, x5=x1 ] endmodule
module process3 = process1 [ x1=x3, x5=x2 ] endmodule
module process4 = process1 [ x1=x4, x5=x3 ] endmodule
module process5 = process1 [ x1=x5, x5=x4 ] endmodule

// Can start in any possible configuration
init true endinit
```

(PRISM language)

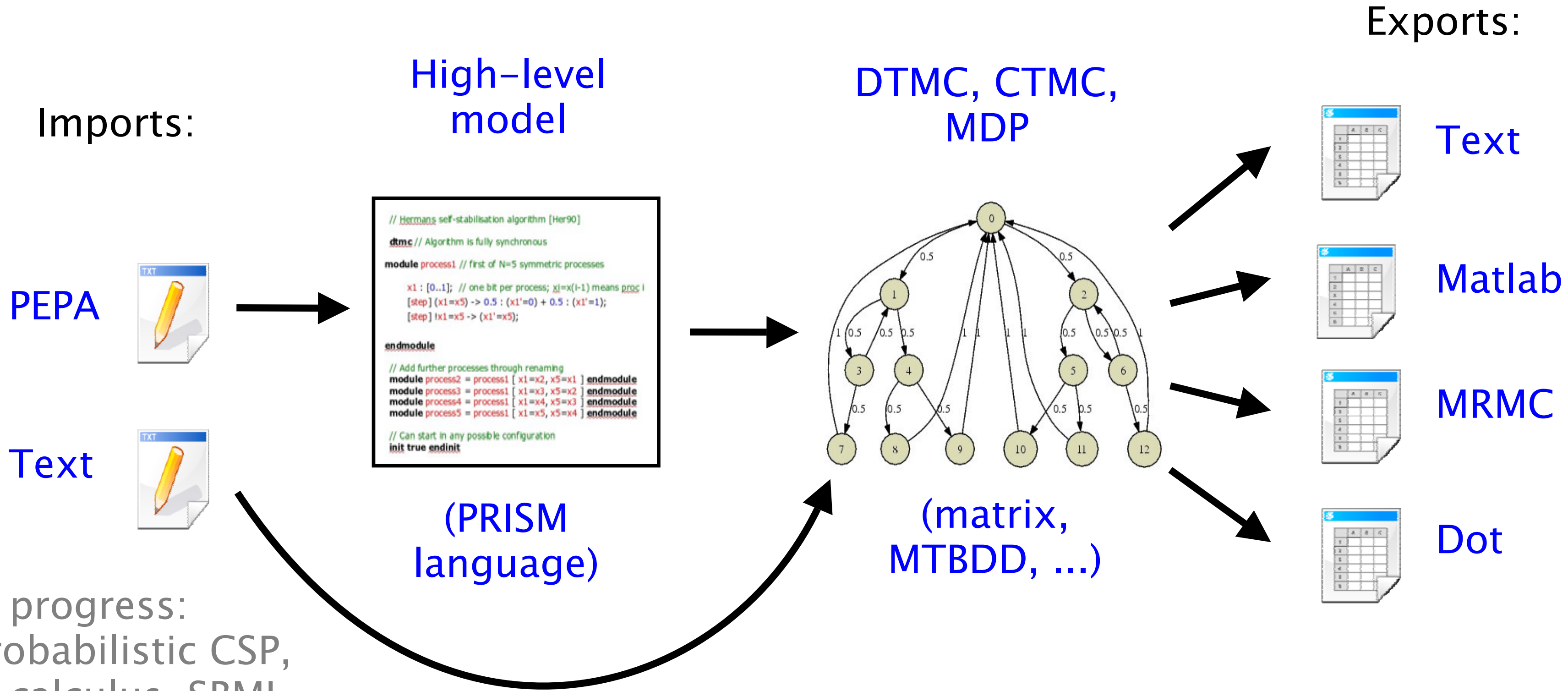
DTMC, CTMC, MDP



(matrix, MTBDD, ...)

PRISM – Imports and exports

- Support for connections to other formats/tools:



In progress:
probabilistic CSP,
pi calculus, SBML,
Probmela, ...

Costs and rewards

- Real-valued quantities assigned to model states/transitions
  - many possible uses, e.g. time, power consumption, current queue size, number of messages lost, ...
- No distinction between costs (“bad”) and rewards (“good”)
  - PRISM terminology is rewards
- The meaning of these rewards varies depending on:
  - the type of property used to analyse the model: instantaneous or cumulative

PRISM property specifications

- Based on (probabilistic extensions of) temporal logic
  - incorporates PCTL for DTMCs/MDPs, CSL for CTMCs
  - also includes: quantitative extensions, costs/rewards
- Simple PCTL/CSL example:
  - P<0.001 [true U shutdown] – "the system eventually shuts down with probability at most 0.001"
- Usually focus on quantitative properties:
  - P=? [true U shutdown] – "what is the probability that the system eventually shuts down?"
  - nested probabilistic operators must be probability-bounded

Basic types of property specifications

- (Unbounded) reachability:
  - P=? [true U shutdown] – "probability of eventual shutdown"
- Transient/time-bounded properties:
  - P=? [true U[t,t] (deliv_rate < min)] – "probability that the packet delivery rate has dropped below minimum at time t"
  - P=? [!repair U≤200 done] – "probability of the process completing within 200 hours and without requiring repairs"
- Steady-state properties:
  - S=? [num_sensors ≥ min] – "long-run probability that an adequate number of sensors are operational"

Cost- and reward-based properties

- Two different interpretations of model rewards
  - instantaneous and cumulative properties
  - reason about expected values of rewards

- Instantaneous reward properties
  - state rewards only
  - state-based measures: “queue size”, “number of operational channels”, “concentration of reactant X”, ...

- R=? [I=t]
  - e.g. “expected size of the message queue at time t?”
- R=? [S]
  - e.g. “long-run expected size of the queue?”

Cost- and reward-based properties

- Cumulative reward properties
  - both state and transition rewards
  - CTMC state rewards interpreted as reward rates
  - e.g. "time", "power consumption", "number of messages lost"

- R=? [F end]
  - e.g. "expected time taken for the protocol to terminate?"
- R=? [C≤2]
  - e.g. "expected power consumption during the first 2 hours that the system is in operation?"
  - e.g. "expected number of messages lost during..."

Best/worst-case scenarios

- Combining “quantitative” and “exhaustive” aspects

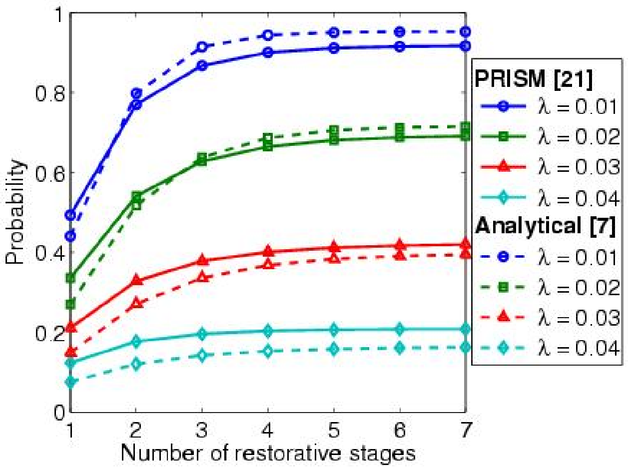
- Computing values for a range of states
  - R=? [F end {“init”}{max}] – “maximum expected run-time over all possible initial configurations”
  - P=? [true U≤t elected {tokens≤k}{min}] – “minimum probability of the leader election algorithm completing within t steps from any state where there are at most k tokens”

- All possible resolutions of nondeterminism (MDPs)
  - Pmin=? [!end2 U end1] – “minimum probability of process 1 finishing before process 2, for any scheduling of processes?”
  - Rmax=? [F message_delivered] – “maximum expected number of bits revealed under any eavesdropping strategy?”

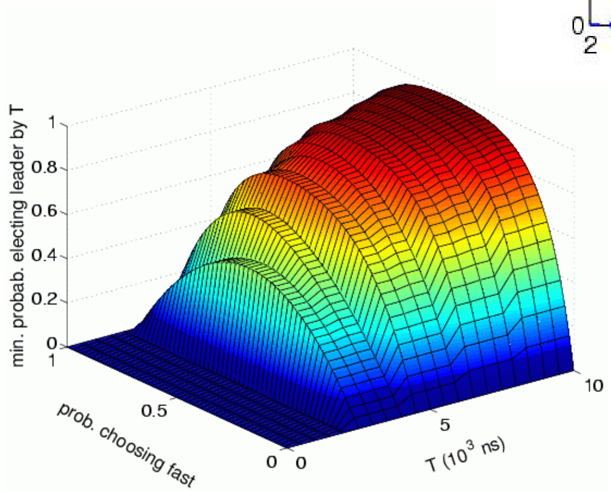
Identifying trends and anomalies

- Counterexamples (error traces)
  - widely used in non-probabilistic model checking
  - situation much less clear in probabilistic model checking
  - counterexample for $P<p$ [true U error] ? and for $P=?$ [...] ?
  - work in progress...

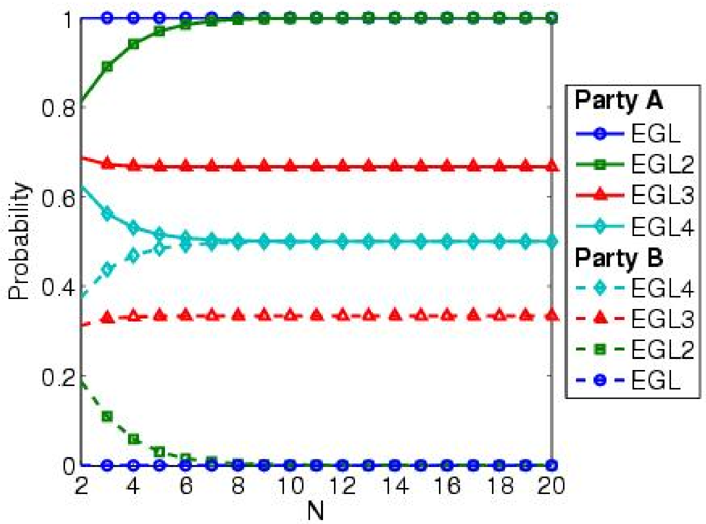
- Experiments: ranges of model/property parameters
  - e.g. $P=?$ [true $U^{\leq T}$ error] for $N=1..5$, $T=1..100$
    where N is some model parameter and T a time bound
  - identify patterns, trends, anomalies in quantitative results



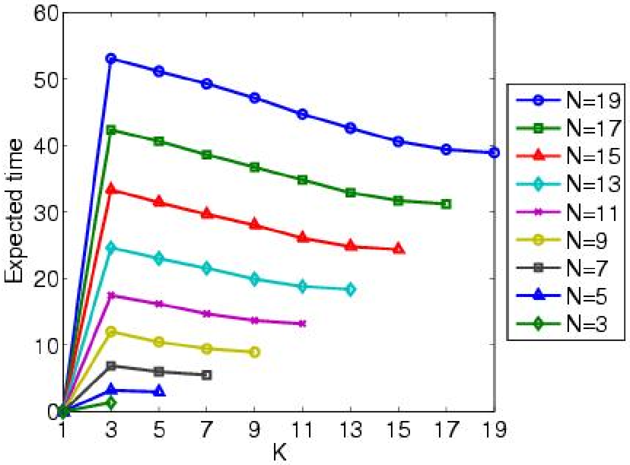
Probability that 10% of gate outputs are erroneous for varying gate failure rates and numbers of stages

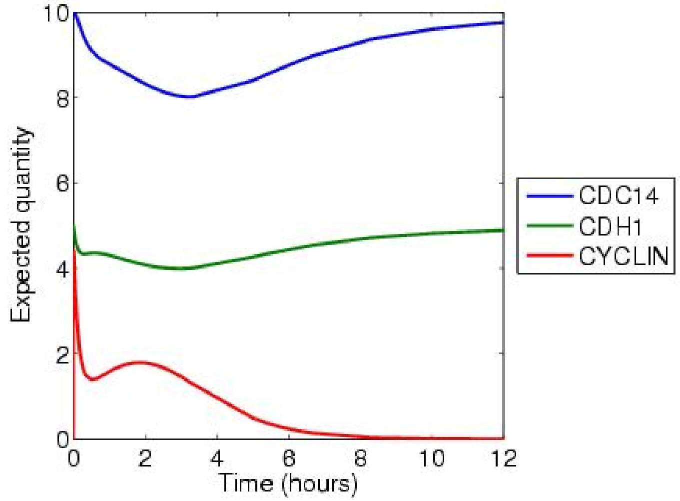


Optimum probability of leader election by time T for various coin biases

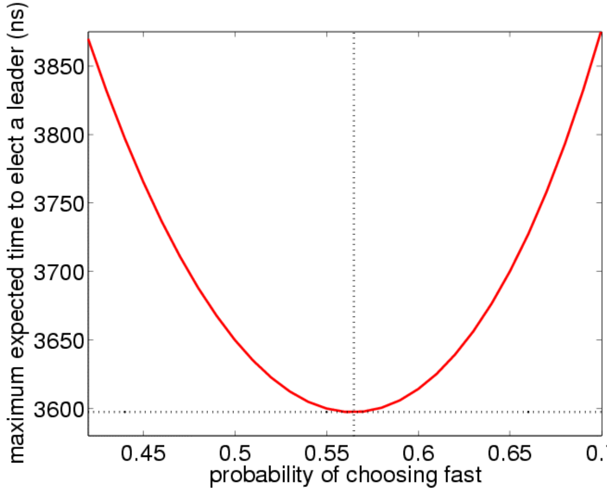


Probability that parties gain unfair advantage for varying numbers of secret packets sent





Worst-case expected number of steps to stabilise for initial configurations with K tokens amongst N processes



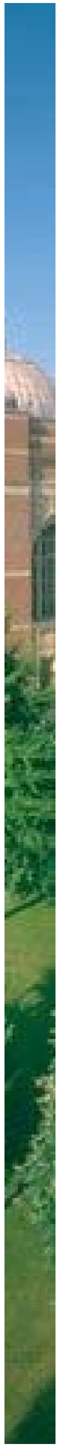
Expected reactant concentrations over the first 12 hours

Maximum expected time for leader election for various coin biases

PRISM functionality

- Graphical user interface
  - model/property editor
  - discrete-event simulator - model traces for debugging, etc.
  - verification of PCTL, CSL + costs/rewards, etc.
  - approximate verification using simulation + sampling
  - easy automation of verification experiments
  - graphical visualisation of results

- Command-line version
  - same underlying verification engines
  - useful for scripting, batch jobs



Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

Power management

- Power management
  - controls power consumption in battery-operated devices
  - savings in power usage translate to extended battery life
  - important for portable, mobile and handheld electronic devices

- System level power management
  - manages various system devices for power optimisation
  - system components manufactured with several power modes
  - e.g. disk drive has: active, idle, standby, sleep, …
  - modes can be changed by the operating system through APIs
  - exploits application characteristics
  - needs to be implemented at the O/S level

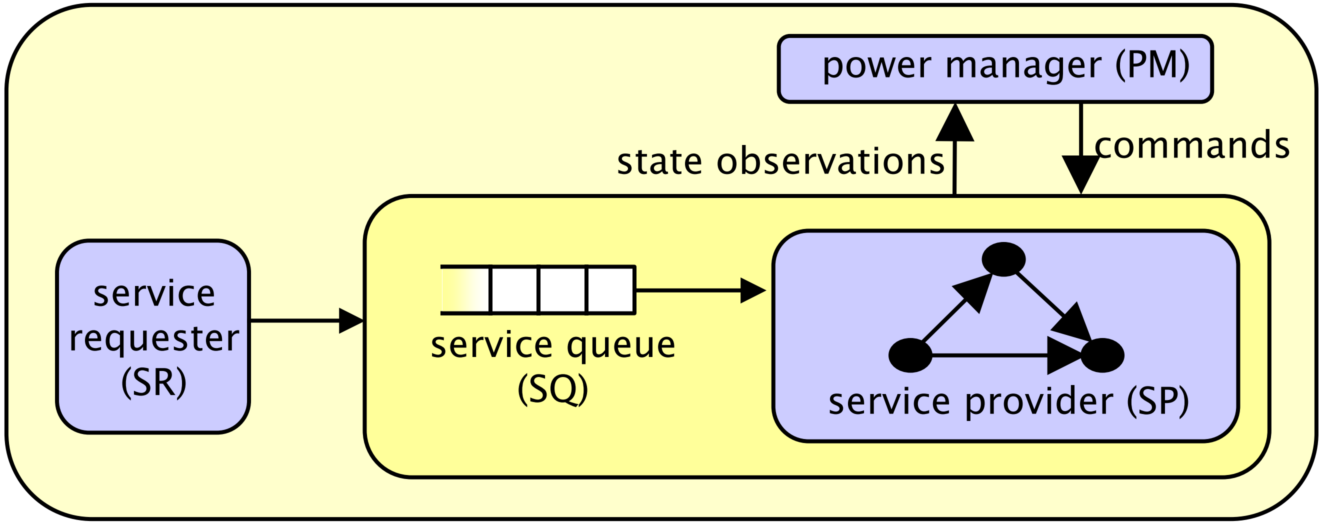
Dynamic Power Management (DPM)

- DPM make optimal decisions at runtime based on:
  - dynamically changing system state
  - workload
  - performance constraints

- Stochastic optimal control strategies for DPM
  - construct a mathematical model of the system in PRISM
  - transition times modelled with exponential distributions
  - formulate stochastic optimisation problems
    e.g. “optimise av. energy usage while av. delay below k”
  - create stochastic strategies by solving optimisation problem
    (exported to Maple for solution externally)
  - analyse strategies in PRISM

DPM – The system model

- Service requester (generates the service requests)
- Service provider (provides service to the requests)
- Service queue (buffers the requests)
- Power manager (monitors the states of the SP and SQ and issues state-transition commands to the SP)



Fujitsu disk drive – The PRISM model

- 4 state Fujitsu disk drive: busy, idle, standby and sleep
- Policies:
  - minimize the average power consumption
  - constraint on the average queue size

- Reward structure "power" (power consumption)
  - state rewards: the av. power consumption of SP in the state
  - transition rewards: energy consumed when SP changes state
- Reward structure "queue" (queue size)
  - state rewards: current size of the queue
- Reward structure "lost" (lost requests)
  - transition rewards: assign 1 to transitions representing the arrival of a request in a state where the queue is full

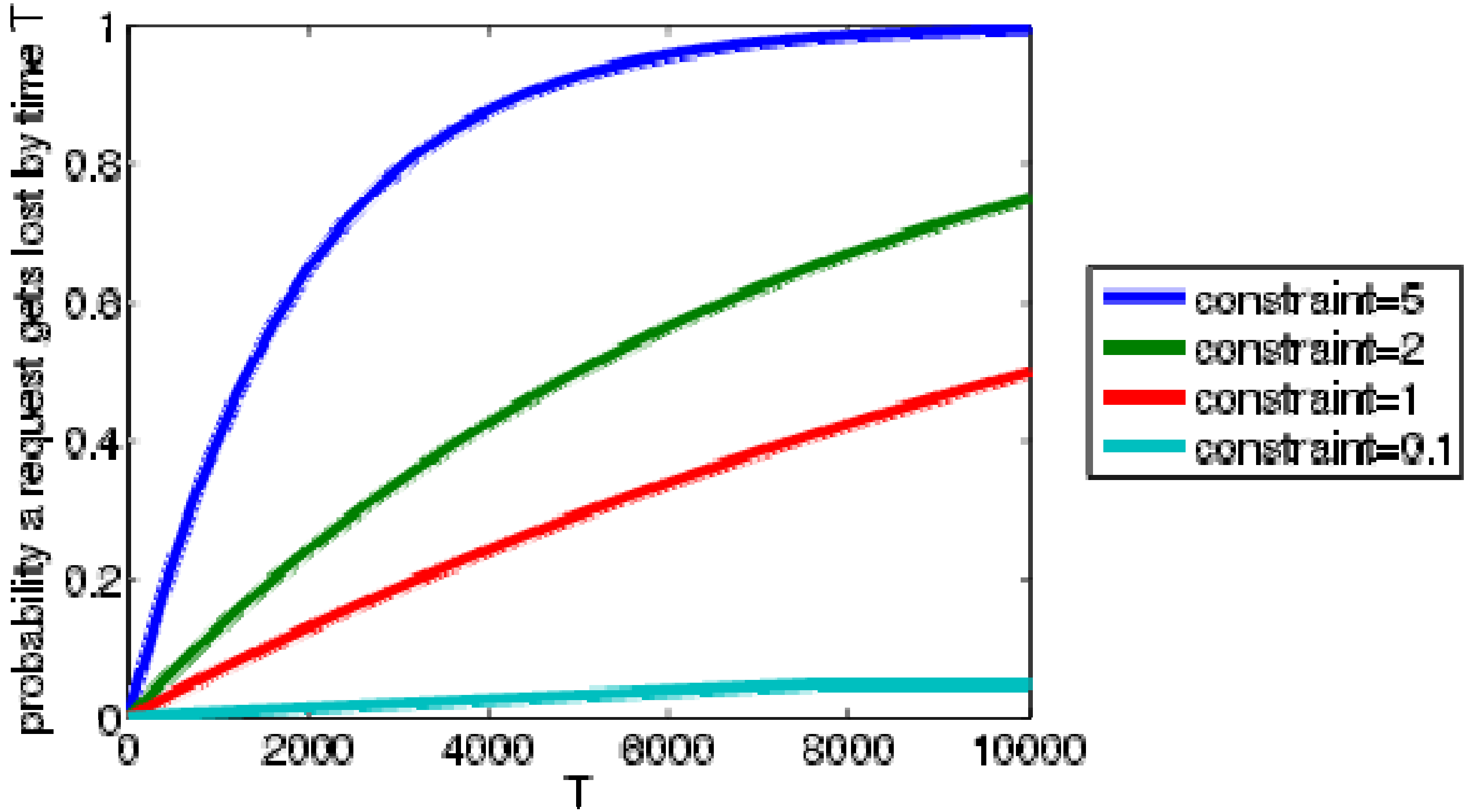
Fujitsu disk drive - Properties

- Selection of properties checked with PRISM

- Probability that queue size becomes ≥ M by time t
  - $P_{=?}[F^{\leq t}\ (q \geq M)]$
- Probability that at least M requests get lost by time t
  - $P_{=?}[F^{\leq t}\ (lost \geq M)]$
- Expected queue size at time t
  - $R_{\{\text{“queue”}\}=?}[I^{=t}]$
- Expected power consumption by time t
  - $R_{\{\text{“power”}\}=?}[C^{\leq t}]$
- Long run average number of requests lost
  - $R_{\{\text{“lost”}\}=?}[S]$

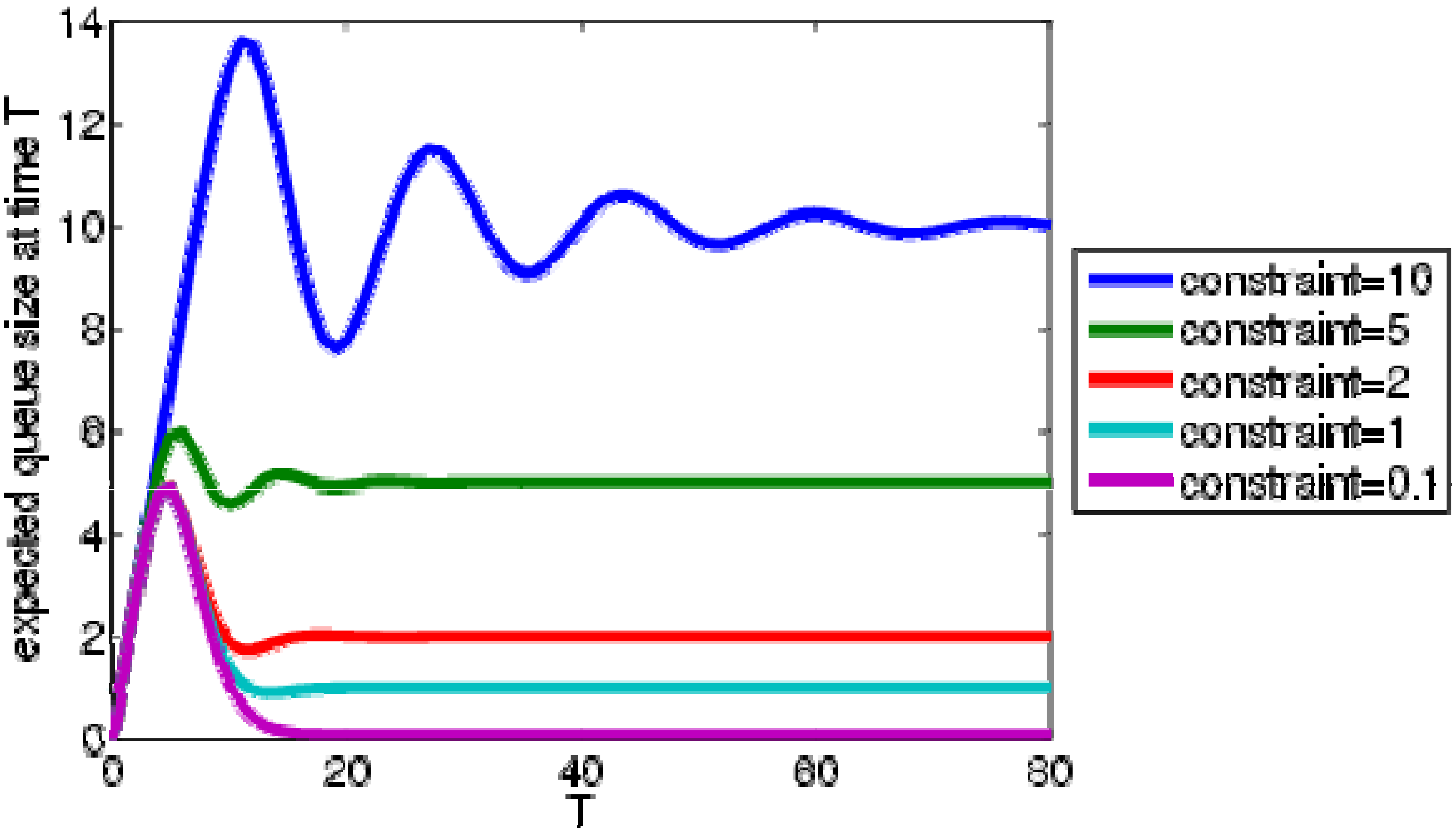
Fujitsu disk drive – PRISM results

- Probability M requests lost by time t $\quad P_{=?}[F^{\leq t}\ (lost \geq M)]$



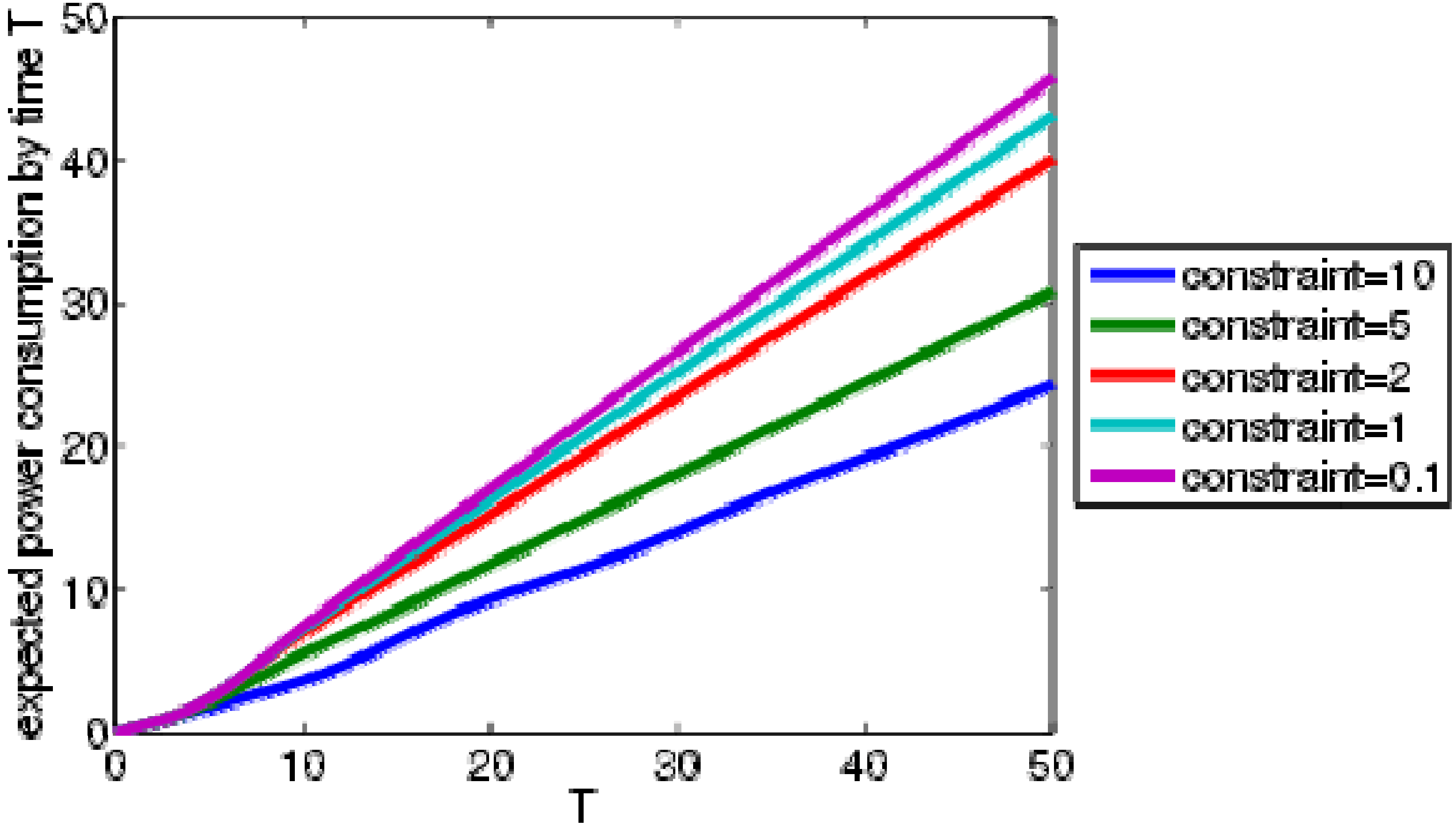
Fujitsu disk drive – PRISM results

- Expected queue size at time t $R_{\{\text{"queue"}\}=?}[I^{=t}]$



Fujitsu disk drive – PRISM results

- Expected power consumption by time t $R_{\{\text{"power"}\}=?}[C^{\leq t}]$



Overview

- Introduction to stochastic model checking
- Discrete-time Markov chains (DTMCs)
  - Properties of DTMCs: The logic PCTL
  - PCTL model checking
  - Costs and rewards
- Continuous-time Markov chains (CTMCs)
  - Properties of CTMCs: The logic CSL
  - CSL model checking
  - Costs and rewards
- Stochastic model checking in practice
  - PRISM software tool
  - Case study 1: Power Management
  - Case study 2: Biological Pathway

Biological systems

- Networks of subsystems
  - organisms, cells, molecules, …
- Interaction
  - governed by rules
  - causes transformations
- Evolution
  - continuous and discrete dynamics
- Mobility
  - motion in space and time, re-configurability, …
- Stochastic behaviour
  - unpredictability, noise, …

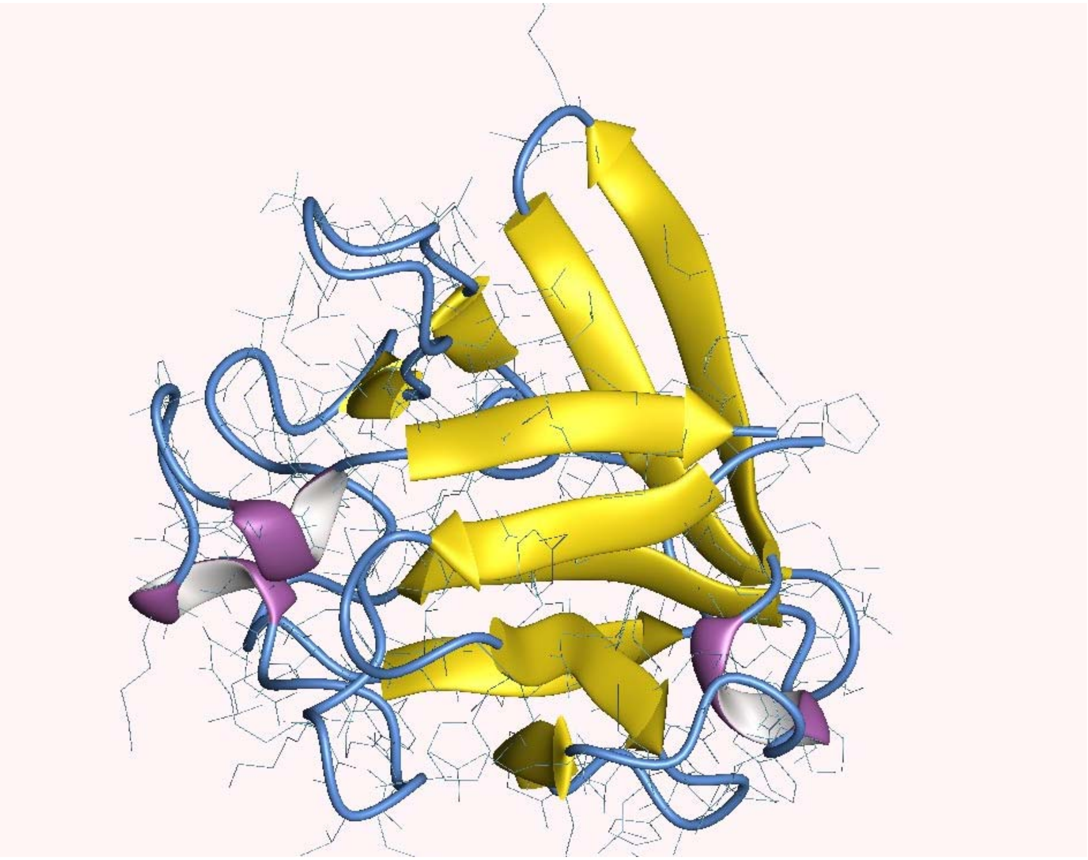
Not unlike computers, networks and the Internet…

Reuse methods for systems biology?

- Propose to use process calculi to model biological processes [Regev, Shapiro, Cardelli, …]

Modelling signalling pathways

- Focus on
  - networks of molecules
  - interaction
  - continuous & discrete dynamics
- Rather than
  - geometry
  - structure
  - sequence



Google images: Human FGF, http://160.114.99.91/astrojan/prot1t.htm

Modelling frameworks

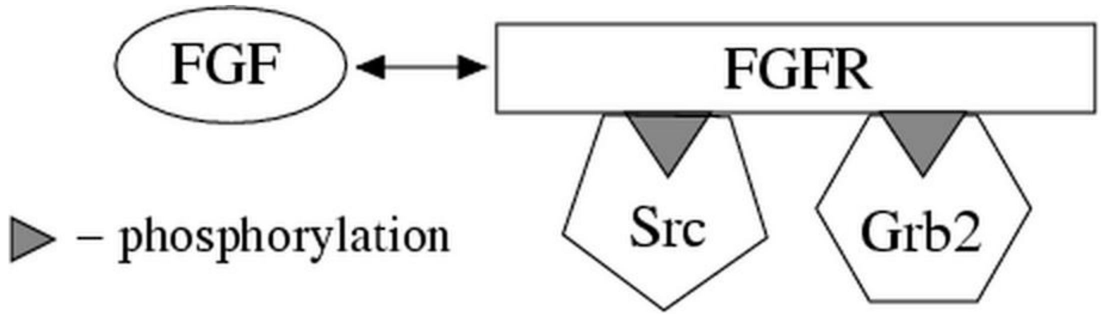
- Assume wish to model mixture of molecules
  - N different molecular species, interact through reactions
  - fixed volume V (spatially uniform), constant pressure and temperature
- Continuous deterministic approach
  - approximate the number of molecules in V at time t by a continuous function, if large numbers of molecules
  - obtain ODEs (ordinary differential equations)
  - not for individual runs, but average
- Discrete stochastic approach
  - discrete system evolution, via discrete events for reactions
  - obtain discrete-state stochastic process

Discrete stochastic approach

- Work with states as vectors $\underline{x}$ of molecule counts for each species
  - probability $P(\underline{x},t)$ that at time t there will be $\underline{x}_A$ of species A
- The good news!
  - if constant state-dependent rates, obtain CTMC
  - therefore, can use stochastic process algebras as model description languages
- The stochastic approach admits
  - discrete event simulation
  - numerical solution (probabilistic model checking)
  - and is realistic for a single time course evolution, not just average

Fragment of FGF pathway

- Fragment of Fibroblast Growth Factor (FGF) pathway
  - regulator of skeletal development, e.g. number of digits



- Biological challenges
  - unknown function of molecules, model different hypotheses
  - expensive experimental scenarios
- Aim to develop ODE and discrete stochastic models
  - ODE: use Cellarator & Mathematica
  - discrete: simulation (BioSPI, SPiM), verification (PRISM)

FGF fragment – The reactions

1: FGF binds/releases FGFR

FGF + FGFR $\rightarrow$ FGFR:FGF	k1=5e+8 M-1s-1
FGF + FGFR $\leftarrow$ FGFR:FGF	k2=0.002 s-1

2: Phosphorylation of FGFR (whilst FGFR:FGF)

FGFR1 $\rightarrow$ FGFR1P	k3=0.1 s-1
FGFR2 $\rightarrow$ FGFR2P	k4=0.1 s-1

3: Dephosphorylation of FGFR

FGFR1P $\rightarrow$ FGFR1	k5=0.1s-1
FGFR2P $\rightarrow$ FGFR2	k6=0.1s-1

4: Effectors bind phosphorylated FGFR

SRC + FGFR1P $\rightarrow$ SRC:FGFR	k7=1e+6 M-1s-1
SRC + FGFR1P $\leftarrow$ SRC:FGFR	k8=0.02 s-1
GRB2 + FGFR2P $\rightarrow$ GRB2:FGFR	k9=1e+6 M-1s-1
GRB2 + FGFR2P $\leftarrow$ GRB2:FGFR	k10=0.02 s-1

5: Relocation of FGFR (whilst SRC:FGFR)

SRC:FGFR $\rightarrow$ relocFGFR	k11=1.1e-3 s-1

FGF fragment – The modelling approach

- Consider a hypothesis about interaction between molecular species in the FGF pathway
  - obtain a set of ODEs from reactions, plot time trajectories for average concentrations (Cellerator)
  - model as a stochastic pi-calculus process, simulate to obtain individual time trajectories (BioSPI, SPiM)
  - model in reactive modules, analyse using probabilistic model checking (PRISM)

- Probabilistic model checking, as opposed to simulation
  - wide range of quantitative properties
  - compute for range of parameters: quantitative trends
  - can definitively establish causal relationships
  - able to identify best/worst case scenarios
  - but suffers from state explosion problems

Stochastic π−calculus code fragment

```
FGFR ::= FGFR_FGF_0 | FGFR_Ph1_0 | ...

FGFR_FGF_0 ::= reloc1?[], true ;                 % relocation
   bind_fgf!{ rel_fgf, reloc4 }, FGFR_FGF_1.      % binding FGF
FGFR_FGF_1 ::=  rel_fgf?[] , FGFR_FGF_0;          % releasing FGF
           ph1?[] , FGFR_FGF_1;                   % phosphorylation
           reloc1?[] , reloc4 ! [] , true;        % relocation …

FGFR_Ph1_0 ::= ph1![] , FGFR_Ph1_1 .              % phosphorylation
FGFR_Ph1_1 ::= dph1![] , FGFR_Ph1_1;              % dephosphorylation
   bind_src!{rel_src1, rel_src2 } , FGFR_SRC. % binding Src

FGFR_SRC ::= rel_src1?[], FGFR_Ph1_1 ;            % releasing Src
   dph1![], rel_src2![], FGFR_Ph1_0;              % dephos (& release Src)
   reloc![], reloc1![], reloc2![] , true.         % relocation
```

Simple PRISM Example

1. A+B ↔ A:B (binding/unbinding rates r_1/r_2)
2. A → (degradation rate r_3)

```
module A
  a : [0..1] init 1
  [bind] a=1 → r1 : (a'=0);
  [rel] a=0 →       r1 : (a'=1);
  [] a=1 → r1 : (a'=0);
endmodule
```

```
module B
  b : [0..1] init 1
  [bind] b=1 → (b'=0);
  [rel] b=0 → (b'=1);
endmodule
```

```
module AB
  ab : [0..1] init 0
  [bind] ab=0 → (ab'=1);
  [rel] ab=1 → (ab'=0);
endmodule
```

reward structure 1:
time A and B are bound

```
rewards "r1"
  ab=1 : 1;
endrewards
```

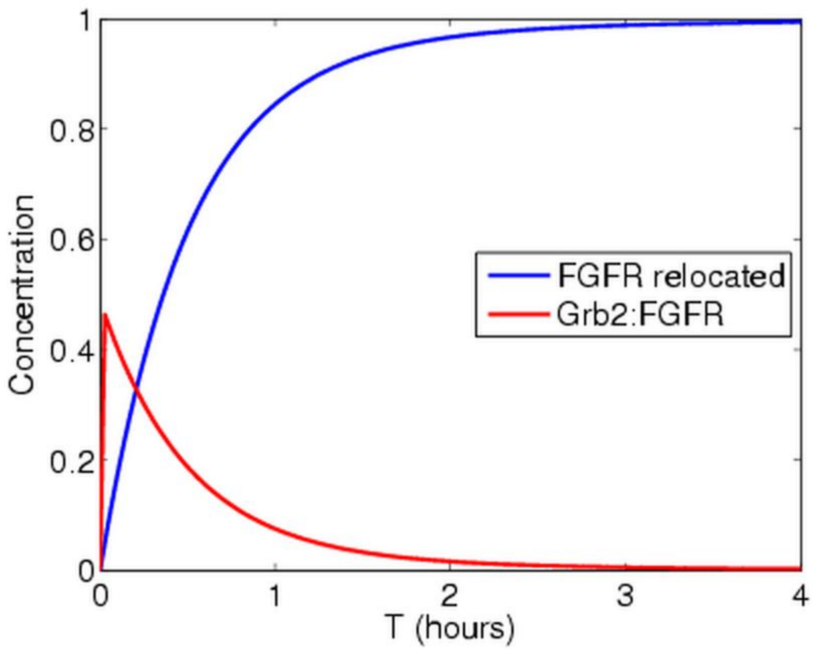
reward structure 2:
binding of A & B

```
rewards "r2"
  [bind] true : 1;
endrewards
```

FGF fragment - Results

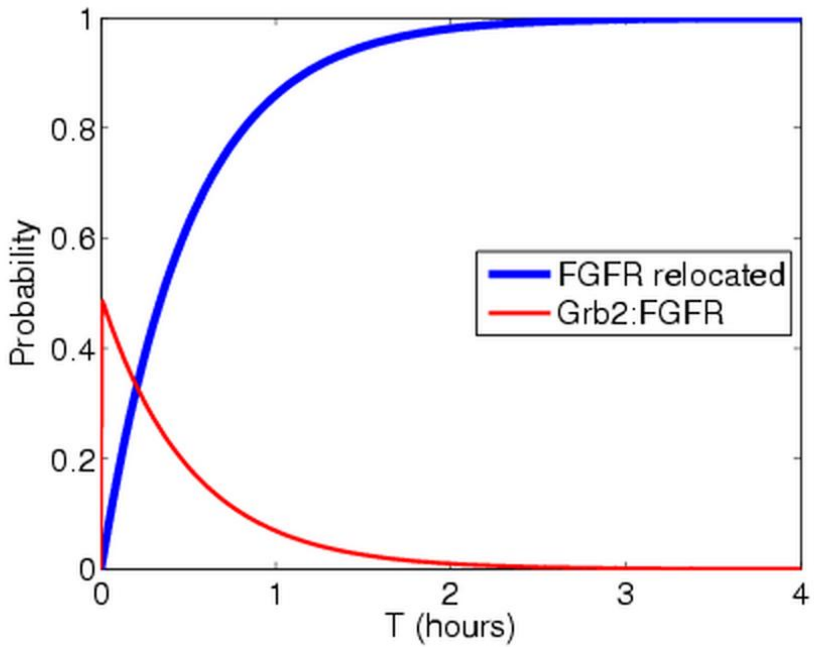
Concentration/quantity of two forms of FGFR over time

ODEs



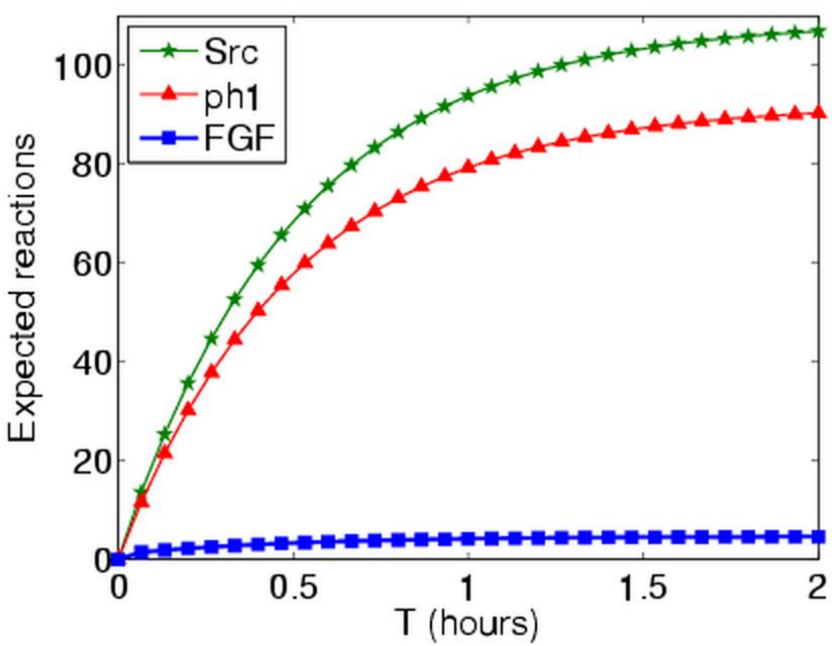
PRISM

BioSPI (10 runs)

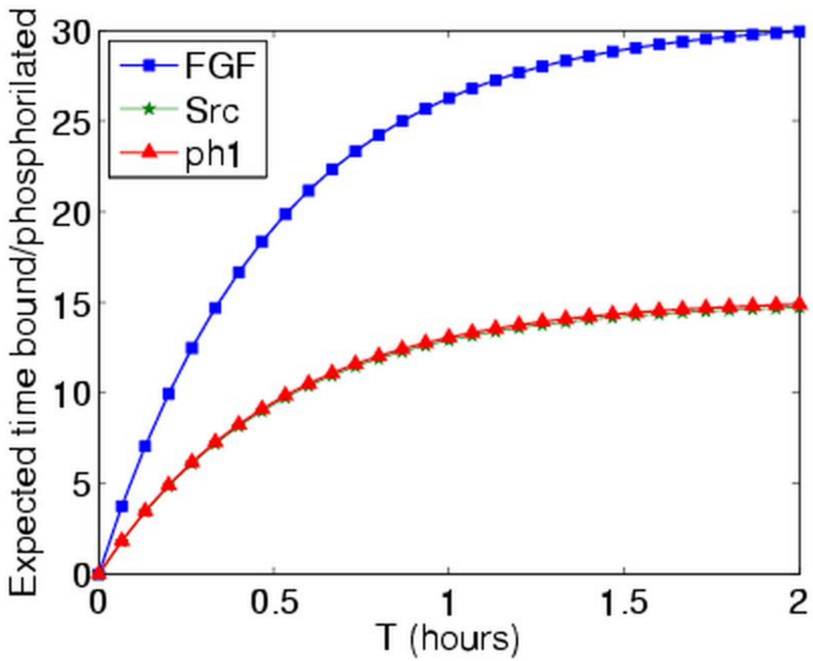


FGF fragment – PRISM results $R_{=?}[C \leq T]$

Expected number of reactions by time T
(assign reward 1 to transitions in which the reaction occurs)



Expected time complex spends bound up to time T
(assign reward 1 to states in which the complex is bound)



A variant of the FGF fragment

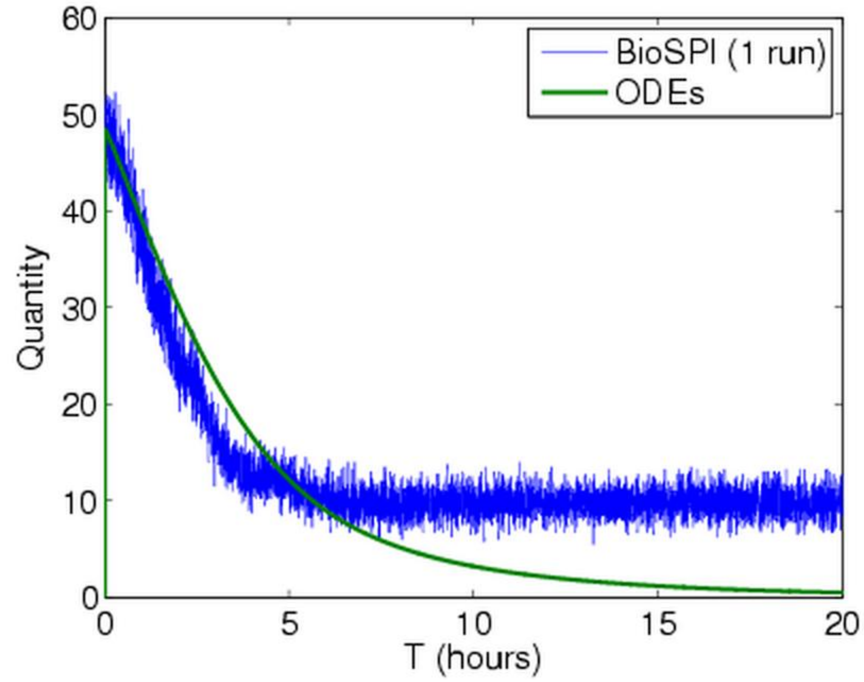
- Src positively regulates FGFR signalling by recruiting non-activated FGFR to the membrane, add reaction:

$$\text{FGFR:Src} \rightarrow \text{FGFR:Src} + \text{FGFR} + \text{Src}$$

Change initial amount of Src from 100 to 10 molecules, and similarly for ODEs

Difference between ODE and BioSPI caused by stochastic approach more accurate when number of molecules small

i.e. Src cannot be totally degraded in ODE



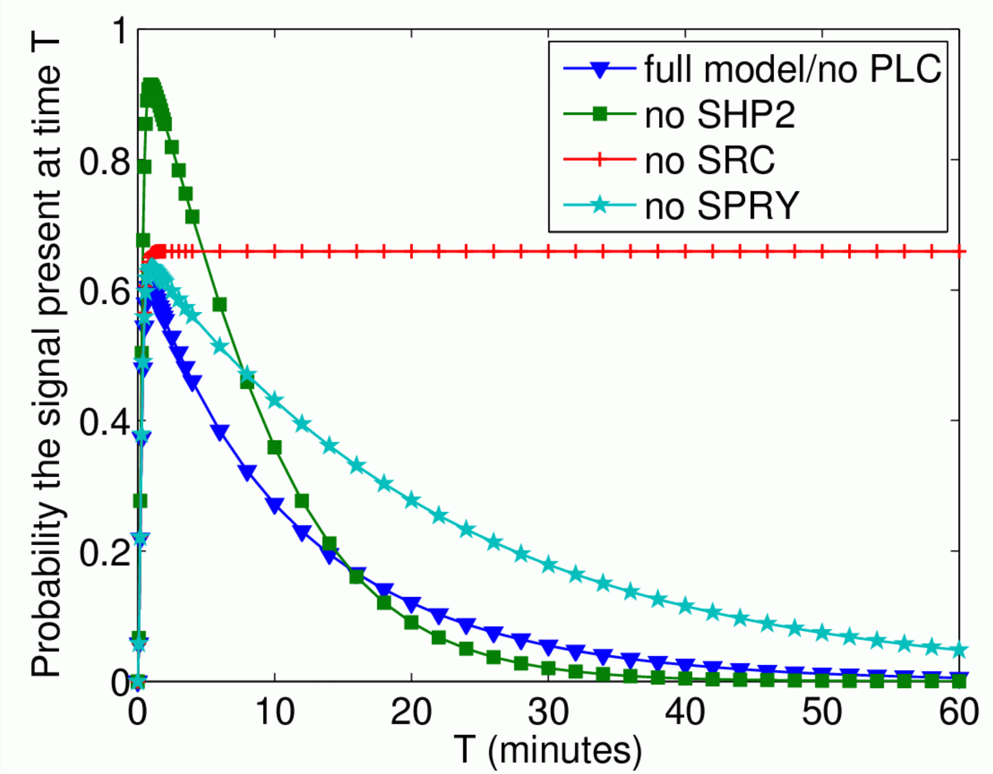
PRISM model of full FGF pathway

- Biological Model
  - 12 elements
  - 14 phosphorylation sites
  - 14 sets of reaction rules (38 rules)

- PRISM model
  - one element of each type (10 modules and 26 variables)
  - relatively small state space
    (80,616 states and 560,520 transitions)
  - however, highly complex: large number of interactions
  - ODE model > 300 equations

FGF pathway – Model checking results

- Probability Grb2 bound to FRS2 at time T
  - $P_{=?}$ [true $U^{[T,T]}$ a_{Grb2}]



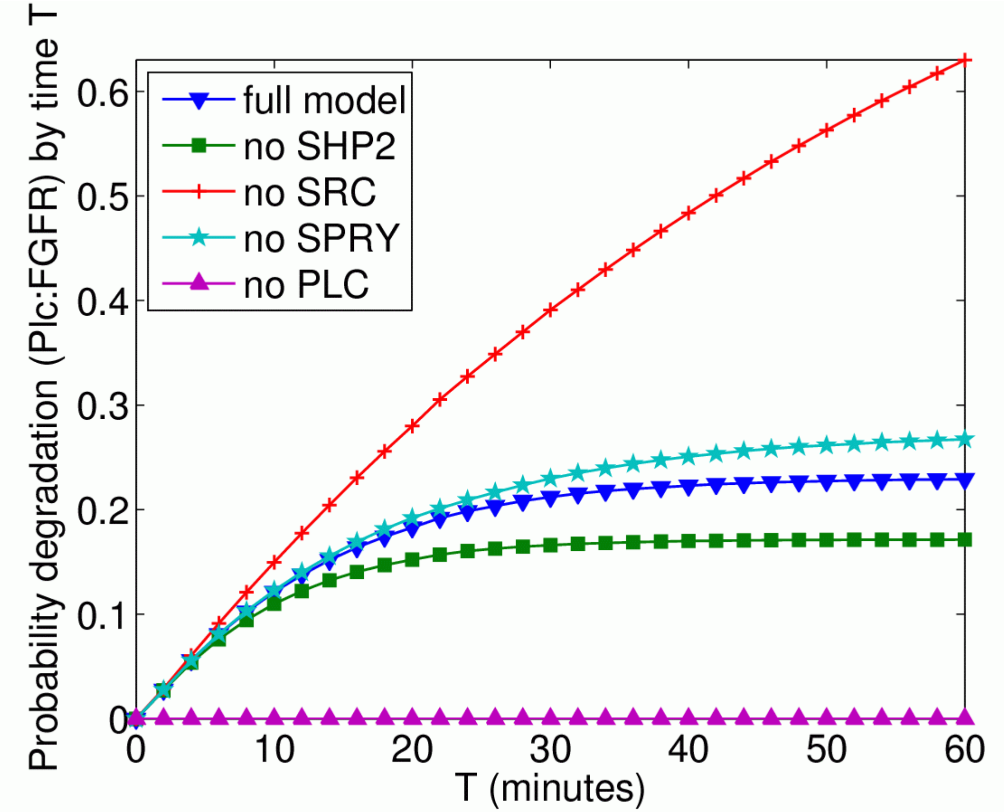
no SRC: no relocation of FRS2, and hence the signal can remain active

no SHP2: main cause of FRS2 dephosphorylation lost increasing the chance that:

- Grb2 bound to FRS
  faster increase in signal
- SRC bound to FRS2
  faster degradation in signal

FGF pathway - Model checking results

- Probability PLC causes degradation/relocation by T
  - $P_{=?} [\neg(a_{src} \vee a_{spry} \vee a_{plc}) \; U^{[0,T]} \; a_{plc}]$



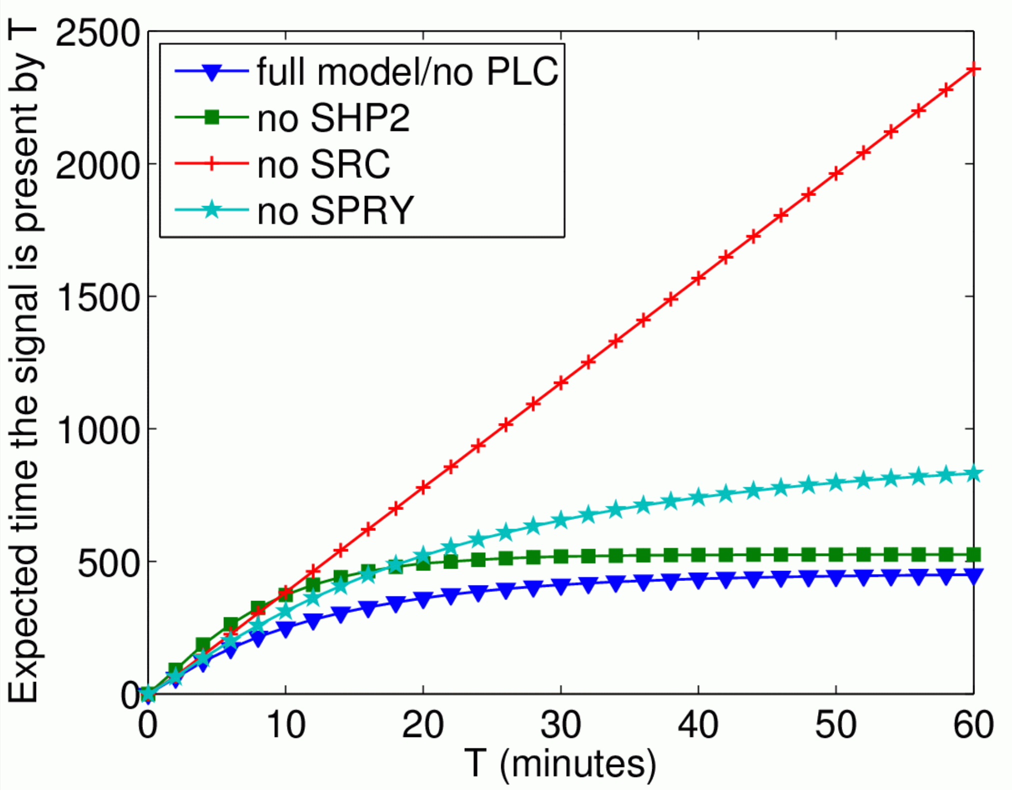
no PLC: PLC cannot cause degradation

no SRC: FRS2 not relocated, more chance of degradation by PLC

no SHP2: greater chance SRC bound to FRS2, increasing the possibility of FRS2 causing relocation

FGF pathway – Model checking results

- Expected time GRB2 bound to FRS2 within time T
  - $R_{=?}$ [C≤T] (assign reward 1 to states where Grb2:FRS2)

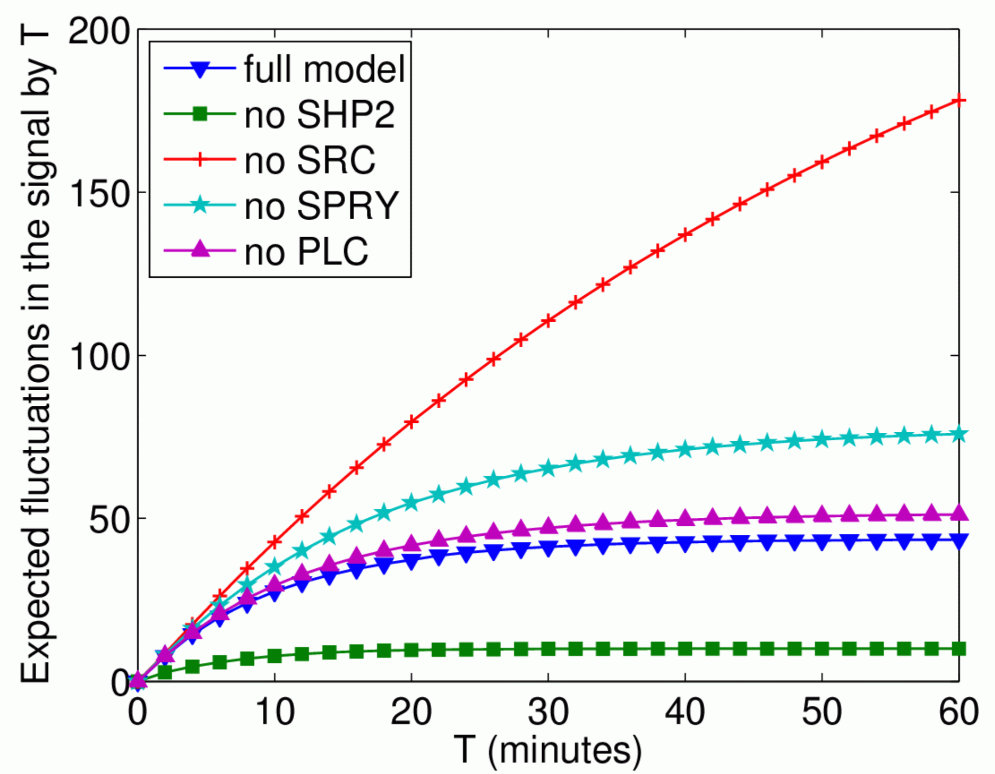


No SRC: no relocation of FRS2 and greater chance FRS2 remains active for longer, hence GRB2 and FRS2 spend more time bound

SPRY: no degradation of FRS2, again GRB2 and FRS2 spend more time bound (but SPRY has smaller influence than SRC)

FGF pathway - Model checking results

- Expected number of times GRB2 & FRS2 bind by T
  - $R_{=?}$ [C≤T] (assign reward 1 to transitions binding Grb2/FRS2)



Cases when SRC and SPRY removed: increased chance that FRS2 remains active, and hence GRB2 and FRS2 can bind more often

No SHP2: decrease in the chance that GRB2:FRS2 unbind, therefore the chance that GRB2 and FRS2 are in a position to (re)bind decreases

Conclusions

- We have given an overview of stochastic model checking
  - Two model types: discrete and continuous time Markov chains
  - Two property specification formalisms: PCTL and CSL with costs and rewards
  - Further models: Markov decision processes and probabilistic timed automata
- Introduced stochastic model checking software
  - Implementation of model checking algorithms within PRISM
  - Similar tools: ETMCC/MRMC, PROBMELA, Vesta, Rapture, Ymer, APMC, APNN-Toolbox, SMART, Mobius
- Demonstrated usefulness of the techniques
  - Examples from biology and performance
  - For further examples see

    www.cs.bham.ac.uk/~dxp/prism/