Probabilistic Model Checking

Marta Kwiatkowska
Gethin Norman
Dave Parker

University of Oxford

Part 2 - Discrete-Time Markov Chains

Overview

- Probability basics
- Discrete-time Markov chains (DTMCs)
  - definition, examples, probability measure
- Properties of DTMCs: The logic PCTL
  - syntax, semantics, equivalences, …
- PCTL model checking
  - algorithms, examples, …
- Costs and rewards

Probability basics

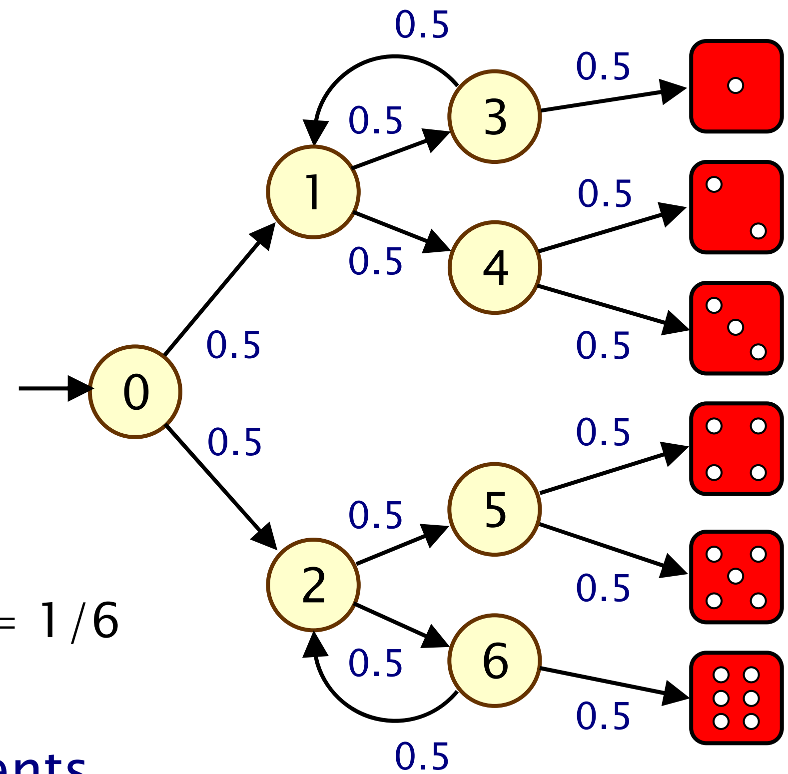
- First, need an experiment
  - The sample set Ω is the set of possible outcomes
  - An event is a subset of Ω, can form events $A \cap B$, $A \cup B$, $\Omega \setminus A$

- Examples:
  - toss a coin: $\Omega = \{H,T\}$, events: "H", "T"
  - toss two coins: $\Omega = \{(H,H),(H,T),(T,H),(T,T)\}$, event: "at least one H"
  - toss a coin ∞-often: Ω is set of infinite sequences event: "H in the first 3 throws"

- Probability is:
  - $P[\text{"H"}] = P[\text{"T"}] = 1/2$, $P[\text{"at least one H"}] = 3/4$
  - $P[\text{"H in the first 3 throws"}] = 1/2 + 1/4 + 1/8 = 7/8$

Probability example

- Modelling a 6-sided die using a fair coin
  - algorithm due to Knuth/Yao:
  - start at 0, toss a coin
  - upper branch when H
  - lower branch when T
  - repeat until value chosen

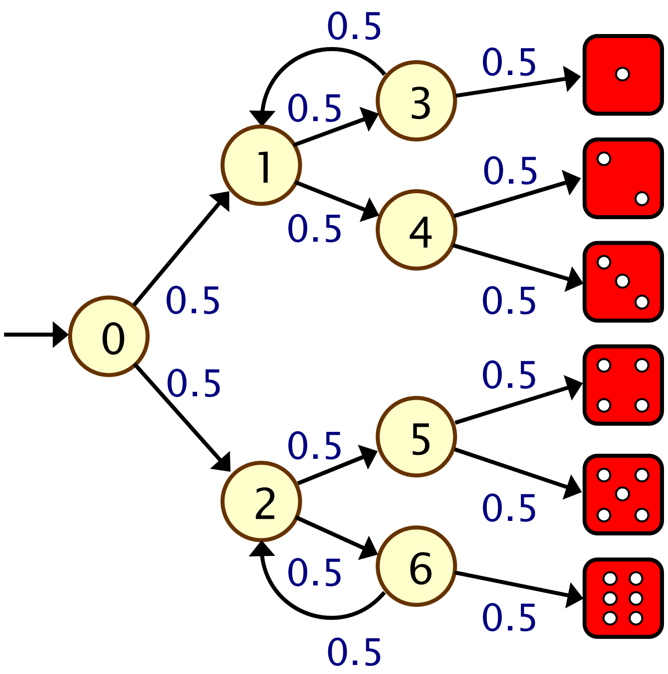


- Probability of obtaining a 6?
  - P[“eventually 6”]
  - $= (1/2)^3 + (1/2)^5 + (1/2)^7 + \ldots = 1/6$
- Obtain as disjoint union of events
  - TTH, TTTTH, TTTTTTH, …

Probability example

- Derive recursive linear equations for P["eventually 6"]
  - let x_i denote the probability for state i = 0,1,2,3,4,5,6
  - probability in state where die takes value 6 is 1
  - probability in all other final states is 0

$$x_6 = 1/2 \cdot x_2 + 1/2 \cdot 1$$
$$x_2 = 1/2 \cdot x_6$$
$$x_0 = 1/2 \cdot x_2$$



- Yields the unique solution:

$$x_0 = 1/6,\ \ x_2 = 1/3 \text{ and } x_6 = 2/3$$

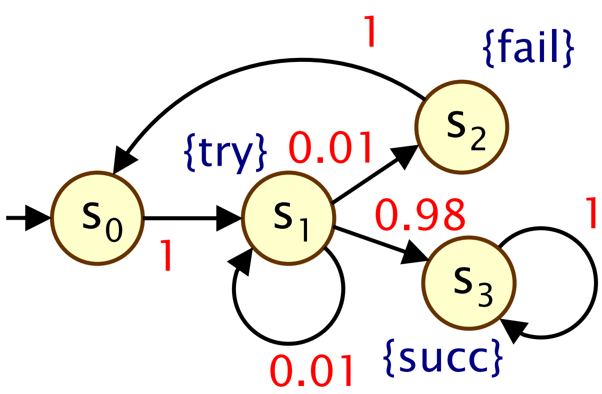
P["eventually 6"] $= x_0 = 1/6$

Overview

- Probability basics
- Discrete-time Markov chains (DTMCs)
  - definition, examples, probability measure
- Properties of DTMCs: The logic PCTL
  - syntax, semantics, equivalences, …
- PCTL model checking
  - algorithms, examples, …
- Costs and rewards

Discrete-time Markov chains

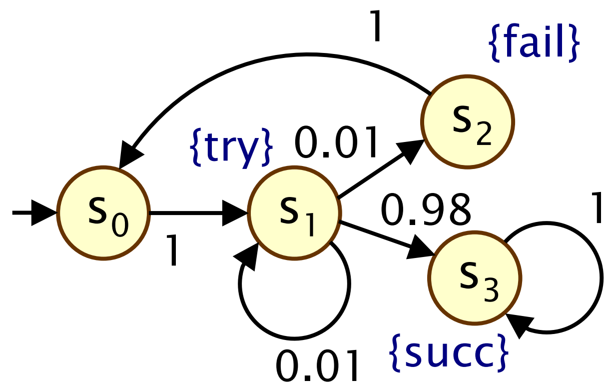
- Discrete-time Markov chains (DTMCs)
  - state-transition systems augmented with probabilities
- States
  - discrete set of states representing possible configurations of the system being modelled
- Transitions
  - transitions between states occur in discrete time-steps
- Probabilities
  - probability 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},P,L) where:
  - S is a finite set of states (“state space”)
  - $s_{init} \in S$ is the initial state
  - $P : S \times S \rightarrow [0,1]$ is the transition probability matrix where $\Sigma_{s' \in S}$ $P(s,s') = 1$ 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

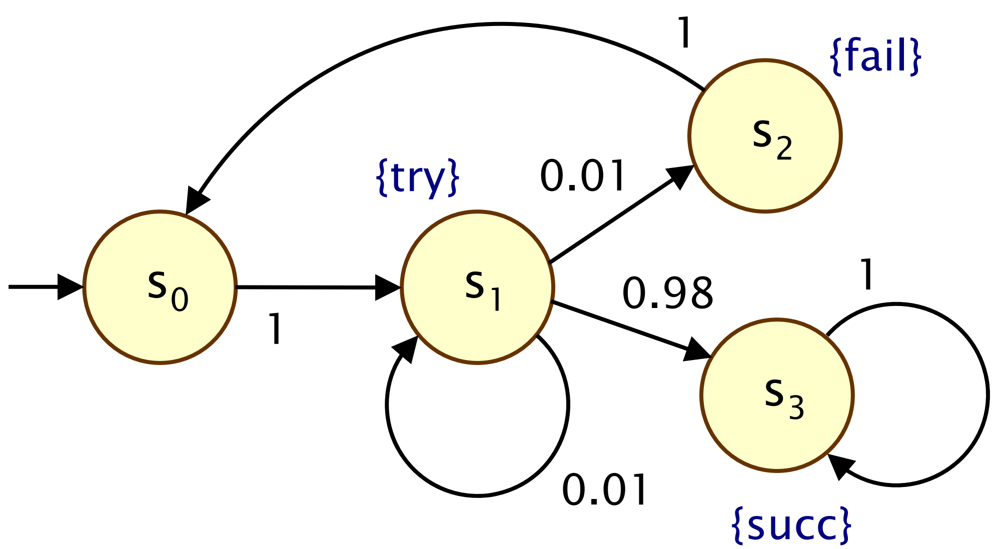


DTMCs: An alternative definition

- Alternative definition: a DTMC is:
  - a family of random variables $\{ X(k) \mid k=0,1,2,\ldots \}$
  - $X(k)$ are observations at discrete time-steps
  - i.e. $X(k)$ is the state of the system at time-step k
- Memorylessness (Markov property)
  - $\Pr(X(k)=s_k \mid X(k-1)=s_{k-1}, \ldots , X(0)=s_0)$
    $= \Pr(X(k)=s_k \mid X(k-1)=s_{k-1})$
- We consider homogenous DTMCs
  - transition probabilities are independent of time
  - $P(s_{k-1},s_k) = \Pr(X(k)=s_k \mid X(k-1)=s_{k-1})$

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



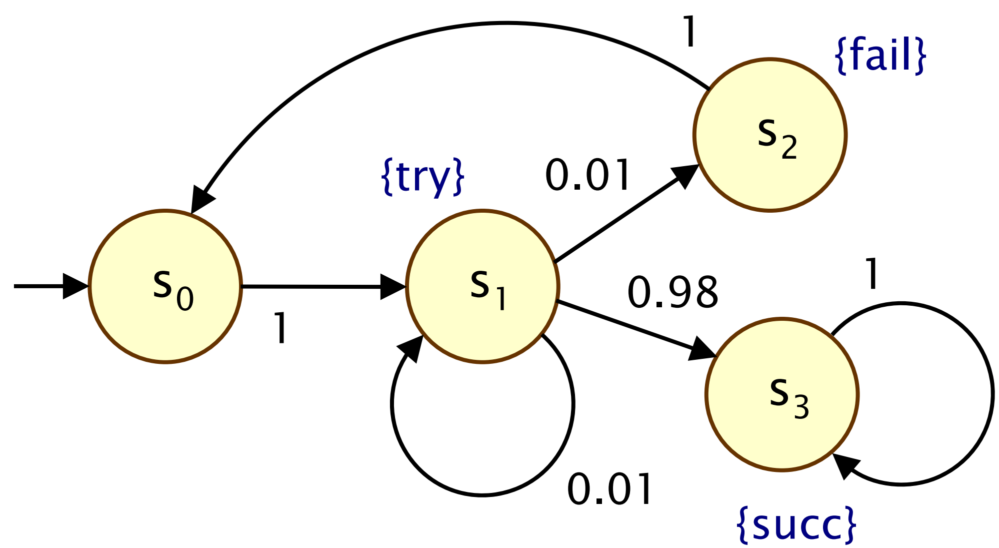
Simple DTMC example

$D = (S, s_{init}, P, L)$

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

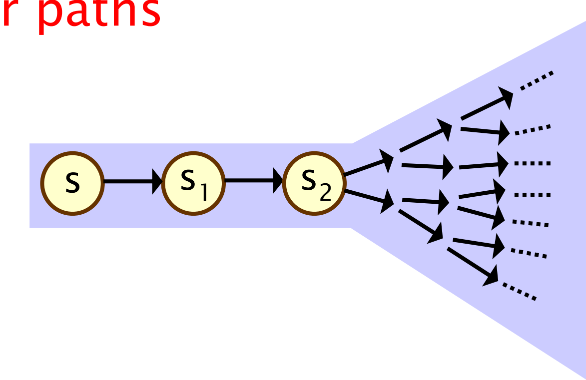
$AP = \{try, fail, succ\}$
$L(s_0) = \emptyset$,
$L(s_1) = \{try\}$,
$L(s_2) = \{fail\}$,
$L(s_3) = \{succ\}$

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0.01 & 0.01 & 0.98 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$



Paths and probabilities

- A (finite or infinite) path through a DTMC
  - is a sequence of states $s_0s_1s_2s_3\ldots$ 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(ω), for a finite path ω
    = set of infinite paths with the common finite prefix ω
  - for example: $C(ss_1s_2)$



Probability spaces

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

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

Probability space over paths

- Sample space Ω = Path(s)

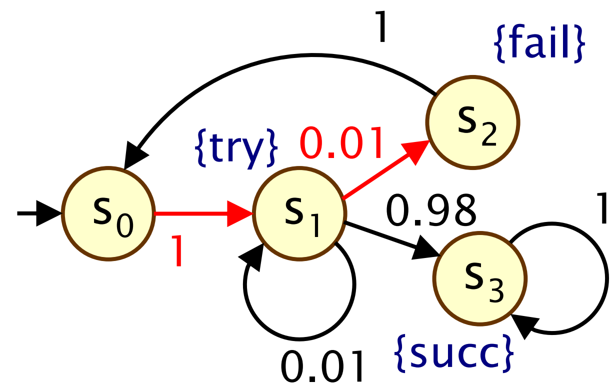
  set of infinite paths with initial state s
- Event set $\Sigma_{Path(s)}$
  - the cylinder set $C(\omega) = \{ \omega' \in Path(s) \mid \omega$ is prefix of $\omega' \}$
  - $\Sigma_{Path(s)}$ is the least σ-algebra on Path(s) containing $C(\omega)$ for all finite paths ω starting in s
- Probability 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 $s_0 s_1 s_2 \ldots$
  - $P_{s0}(\omega) = P(s_0,s_1) \cdot P(s_1,s_2)$

    $= 1 \cdot 0.01 = 0.01$
  - $Pr_{s0}(C(\omega)) = P_{s0}(\omega) = 0.01$



- Paths which are eventually successful and with no failures
  - $C(s_0s_1s_3) \cup C(s_0s_1s_1s_3) \cup C(s_0s_1s_1s_1s_3) \cup \ldots$
  - $Pr_{s0}(\ C(s_0s_1s_3) \cup C(s_0s_1s_1s_3) \cup C(s_0s_1s_1s_1s_3) \cup \ldots\ )$

    $= P_{s0}(s_0s_1s_3) + P_{s0}(s_0s_1s_1s_3) + P_{s0}(s_0s_1s_1s_1s_3) + \ldots$

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

    $= 98/99$

    $= 0.9898989898\ldots$

Overview

- Probability basics
- Discrete-time Markov chains (DTMCs)
  - definition, examples, probability measure
- Properties of DTMCs: The logic PCTL
  - syntax, semantics, equivalences, …
- PCTL model checking
  - algorithms, examples, …
- Costs and rewards

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

PCTL syntax

ψ is true with probability $\sim p$

- PCTL syntax:
  - $\phi \ ::= \ \text{true} \mid a \mid \phi \wedge \phi \mid \neg\phi \mid P_{\sim p}\,[\,\psi\,]$ (state formulas)
  - $\psi \ ::= \ X\,\phi \ \mid \ \phi\ U^{\leq k}\ \phi \ \mid \ \phi\ U\ \phi$ (path formulas)

"next" "bounded until" "until"

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

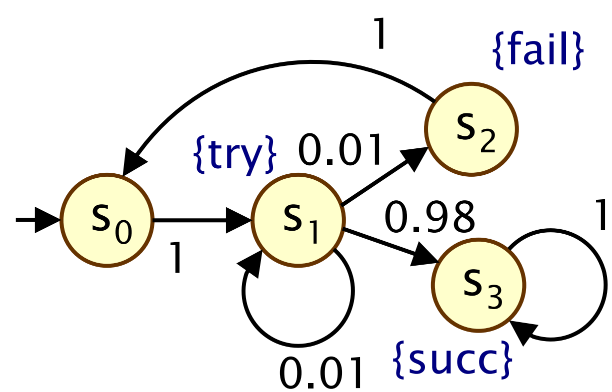
- A PCTL formula is always a state formula
  - path formulas only occur inside the P operator

PCTL semantics for DTMCs

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

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

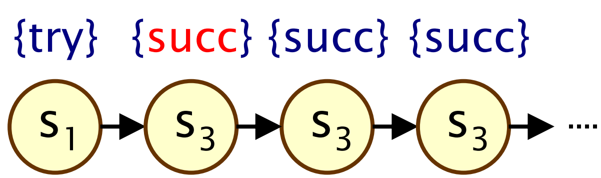
- Examples
  - $s_3 \vDash succ$
  - $s_1 \vDash try \wedge \neg fail$



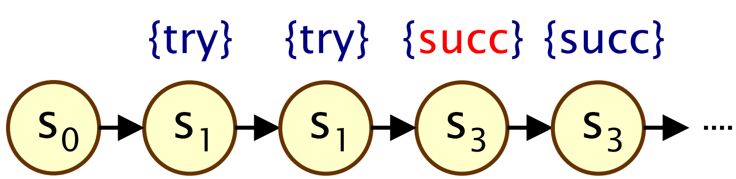
PCTL semantics for DTMCs

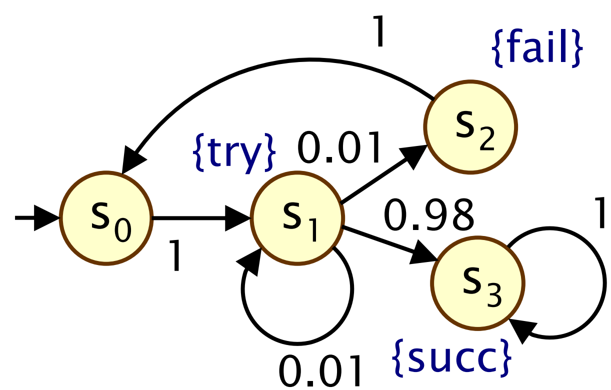
- Semantics of path formulas:
  - for a path $\omega = s_0 s_1 s_2 \ldots$ in the DTMC:
  - $\omega \vDash X\ \phi \quad \Leftrightarrow \quad s_1 \vDash \phi$
  - $\omega \vDash \phi_1\ U^{\leq k}\ \phi_2 \quad \Leftrightarrow \quad \exists i \leq k$ such that $s_i \vDash \phi_2$ and $\forall j<i,\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

- Some examples of satisfying paths:
  - X succ



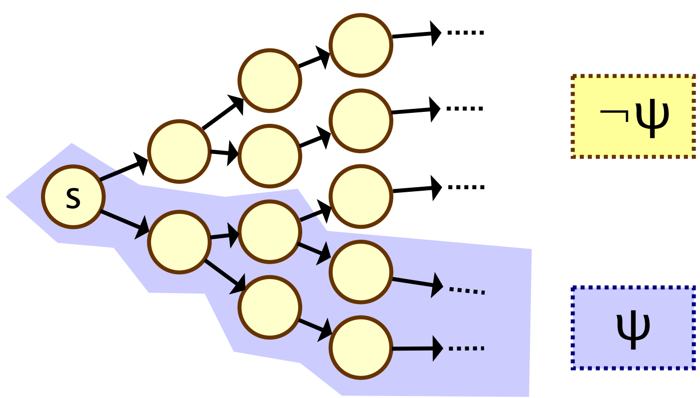
  - ¬fail U succ





PCTL semantics for DTMCs

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



Prob(s, ψ) ~ p ?

PCTL derived operators

- 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)

- 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
  - $F\ \phi \equiv$ true $U\ \phi$ (F = "future")
  - sometimes written as $\Diamond\ \phi$ ("diamond")
  - "ϕ is eventually true"
  - bounded version: $F^{\leq k}\ \phi \equiv$ 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 $\square\ \phi$ ("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. $P_{\sim p}\ [\ G\ \phi\ ]$

Derivation of $P_{\sim p}$ [$G\ \phi$]

- In fact, we can derive $P_{\sim p}$ [$G\ \phi$] directly in PCTL, e.g.
  - $s \models P_{>p}$ [$G\ \phi$] $\Leftrightarrow$ $\mathrm{Prob}(s, G\ \phi) > p$
    $\Leftrightarrow$ $\mathrm{Prob}(s, \neg(F\ \neg\phi)) > p$
    $\Leftrightarrow$ $1 - \mathrm{Prob}(s, F\ \neg\phi) > p$
    $\Leftrightarrow$ $\mathrm{Prob}(s, F\ \neg\phi) < 1 - p$
    $\Leftrightarrow$ $s \models P_{<1-p}$ [$F\ \neg\phi$]

- Other equivalences:
  - $P_{\geq p}$ [$G\ \phi$] $\equiv$ $P_{\leq 1-p}$ [$F\ \neg\phi$]
  - $P_{<p}$ [$G\ \phi$] $\equiv$ $P_{>1-p}$ [$F\ \neg\phi$]
  - $P_{>p}$ [$G^{\leq k}\ \phi$] $\equiv$ $P_{<1-p}$ [$F^{\leq k}\ \neg\phi$]
  - etc.

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(ω) for all finite paths ω starting in s and is closed under complementation, countable union

- Next (X ϕ)
  - cylinder sets constructed from paths of length one
- Bounded until (ϕ_1 $U^{\leq k}$ ϕ_2)
  - (finite number of) cylinder sets from paths of length at most k
- Until (ϕ_1 U ϕ_2)
  - countable union of paths satisfying ϕ_1 $U^{\leq k}$ ϕ_2 for all $k \geq 0$

Qualitative vs. quantitative properties

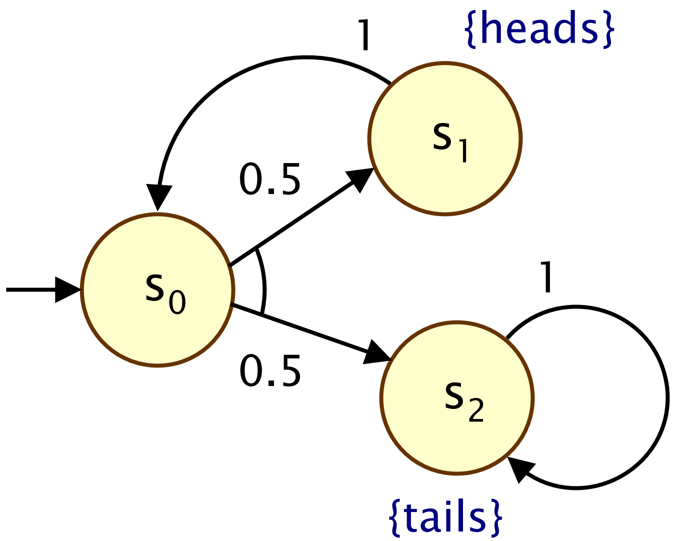
- P operator of PCTL can be seen as a quantitative analogue of the CTL operators A (for all) and E (there exists)

- 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…

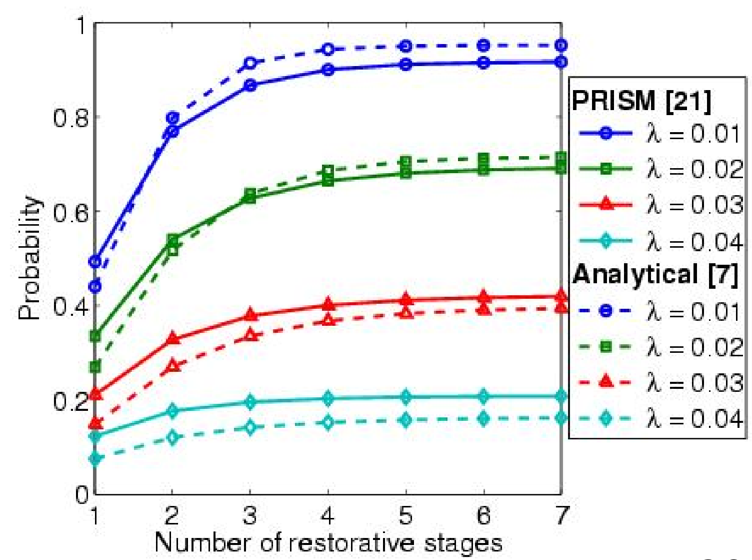
Example: Qualitative/quantitative

- 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

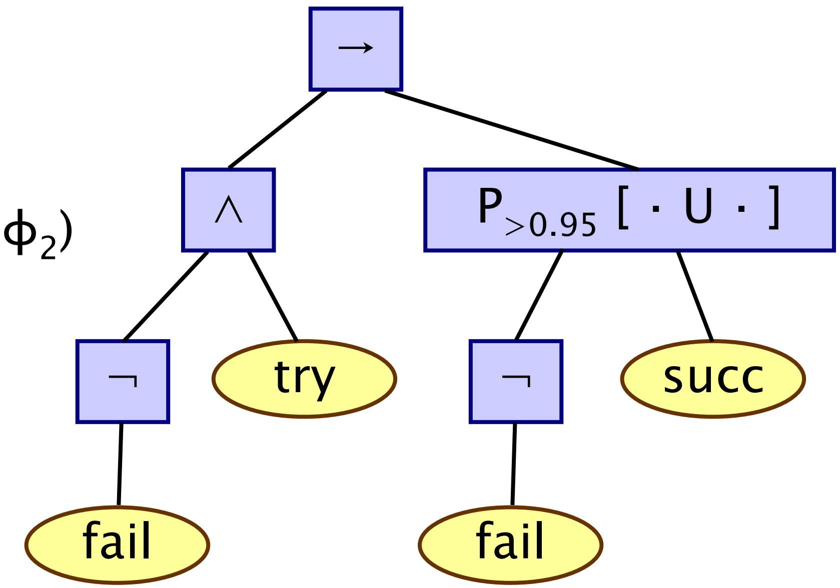
- Probability basics
- Discrete-time Markov chains (DTMCs)
  - definition, examples, probability measure
- Properties of DTMCs: The logic PCTL
  - syntax, semantics, equivalences, …
- PCTL model checking
  - algorithms, examples, …
- Costs and rewards

PCTL model checking for DTMCs

- Algorithm for PCTL model checking [CY88,HJ94,CY95]
  - inputs: DTMC $D=(S,s_{init},P,L)$, PCTL formula ϕ
  - output: $Sat(\phi) = \{ s \in S \mid s \models \phi \}$ = 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 for DTMCs

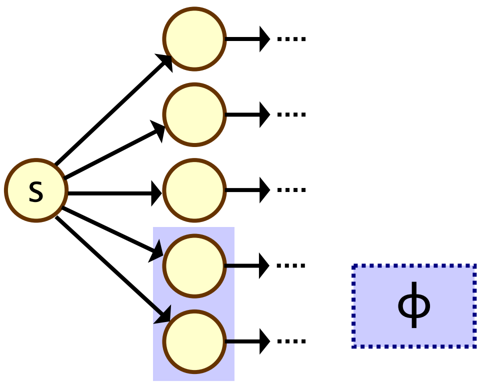
- Basic algorithm proceeds by induction on parse tree of ϕ
  - example: $\phi = (\neg fail \wedge try) \rightarrow P_{>0.95} [\ \neg fail\ U\ succ\ ]$
- For the non-probabilistic operators:
  - $Sat(true) = S$
  - $Sat(a) = \{ s \in S \mid a \in L(s) \}$
  - $Sat(\neg\phi) = S \setminus Sat(\phi)$
  - $Sat(\phi_1 \wedge \phi_2) = Sat(\phi_1) \cap Sat(\phi_2)$
- For the $P_{\sim p} [\ \psi\ ]$ operator
  - need to compute the probabilities $Prob(s, \psi)$ for all states $s \in S$



PCTL next for DTMCs

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

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

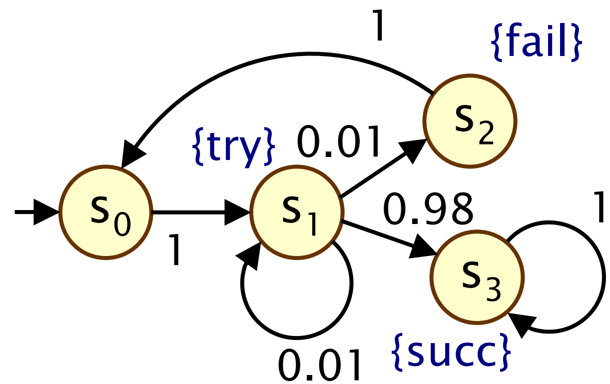


- Compute vector $\underline{Prob}(X\ \phi)$ of probabilities for all states s
  - $\underline{Prob}(X\ \phi) = \mathbf{P} \cdot \underline{\phi}$
  - where $\underline{\phi}$ is a 0-1 vector over S with $\underline{\phi}(s) = 1$ iff $s \vDash \phi$
  - computation requires a single matrix-vector multiplication

PCTL next – Example

- Model check: $P_{\geq 0.9}$ [X ($\neg$try $\vee$ succ)]
  - Sat ($\neg$try $\vee$ succ) = (S \ Sat(try)) $\cup$ Sat(succ)
    = ({s_0,s_1,s_2,s_3} \ {s_1}) $\cup$ {s_3} = {s_0,s_2,s_3}
  - $\underline{\text{Prob}}$(X ($\neg$try $\vee$ succ)) = **P** · $\underline{(\neg\text{try} \vee \text{succ})}$ = …

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



- Results:
  - $\underline{\text{Prob}}$(X ($\neg$try $\vee$ succ)) = [0, 0.99, 1, 1]
  - Sat($P_{\geq 0.9}$ [X ($\neg$try $\vee$ 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
  - $\underline{\text{Prob}}(\text{true } U^{\leq 0} \text{ succ}) = \underline{\text{succ}} = [0, 0, 0, 1]$

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

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

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

PCTL until for DTMCs

- Computation of probabilities Prob(s, ϕ_1 U ϕ_2) for all s $\in$ S
- Similar to the bounded until operator, we first identify all states where the probability is 1 or 0
  - $S^{yes} = Sat(P_{\geq 1}\,[\ \phi_1\ U\ \phi_2\ ])$
  - $S^{no} = Sat(P_{\leq 0}\,[\ \phi_1\ U\ \phi_2\ ])$
- 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($P_{\leq 0}$ [ϕ_1 U ϕ_2]) :
  - first compute Sat($P_{>0}$ [ϕ_1 U ϕ_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($P_{\geq 1}$ [ϕ_1 U ϕ_2]) :
  - first compute Sat($P_{<1}$ [ϕ_1 U ϕ_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 until for DTMCs

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

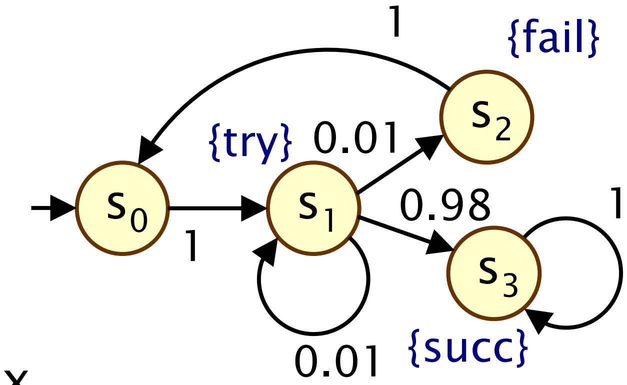
$$\text{Prob}(s, \phi_1 \,\text{U}\, \phi_2) = \begin{cases} 1 & \text{if } s \in S^{yes} \\ 0 & \text{if } s \in S^{no} \\ \sum_{s' \in S} P(s,s') \cdot \text{Prob}(s', \phi_1 \,\text{U}\, \phi_2) & \text{otherwise} \end{cases}$$

  - 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 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:
  - $\underline{Prob}$(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

- Probability basics
- Discrete-time Markov chains (DTMCs)
  - definition, examples, probability measure
- Properties of DTMCs: The logic PCTL
  - syntax, semantics, equivalences, …
- PCTL model checking
  - algorithms, examples, …
- Costs and rewards

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

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

Reward formula semantics

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

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

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

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

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

- 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 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 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

Summing up…

- Discrete-time Markov chains (DTMCs)
  - definition, examples, probability measure
- Properties of DTMCs: The logic PCTL
  - syntax, semantics, equivalences, …
- PCTL model checking
  - algorithms, examples, …
- Costs and rewards