Verification of Probabilistic Systems

Dave Parker

University of Oxford

MOVEP’08 – Orléans – June 2008
Motivation – Why probability?

• Some systems are inherently probabilistic…

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

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

• Some systems are inherently probabilistic…

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

• To model uncertainty and performance
  – to quantify rate of failures, express Quality of Service

• Examples:
  – computer networks, embedded systems
  – power management policies
  – nano-scale circuitry: reliability through defect-tolerance
Motivation – Why probability?

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

• We are not just interested in correctness

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

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

Finite-state model

Temporal logic specification

¬EF error

Model checker

Result

Error trace
Probabilistic model checking

Probabilistic model
e.g. Markov chain

Probabilistic model checker
e.g. PRISM

Result

\[ P_{<0.01} \ [ \ F \ error \ ] \]

Probabilistic temporal logic specification
e.g. PCTL

Quantitative results
Overview

- Discrete-time Markov chains (DTMCs)
- Properties of DTMCs: The logic PCTL
- PCTL model checking for DTMCs
- Beyond PCTL: Costs and rewards
- Tool support + A case study: contract signing
- Adding nondeterminism: Markov decision processes (MDPs)
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 \((S, s_{\text{init}}, P, L)\) where:
  - \(S\) is a finite set of states ("state space")
  - \(s_{\text{init}} \in S\) is the initial state
  - \(P : S \times S \rightarrow [0,1]\) is the transition probability matrix
    where \(\sum_{s' \in S} P(s,s') = 1\) for all \(s \in S\)
  - \(L : S \rightarrow 2^{\text{AP}}\) is function labelling states with atomic propositions

- Note: no deadlock states
  - i.e. every state has at least one outgoing transition
  - can add self loops to represent final/terminating states
DTMCs: An alternative definition

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

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

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

• A (finite or infinite) path through a DTMC
  – is a sequence of states \( s_0s_1s_2s_3 \ldots \) such that \( P(s_i,s_{i+1}) > 0 \ \forall i \)
  – 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: \( \text{Path}(s) = \) 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 \( \omega \)
    = set of infinite paths with the common finite prefix \( \omega \)
  – for example: \( C(ss_1s_2) \)
Probability spaces

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

- **Sample space** $\Omega = \text{Path}(s)$
  set of infinite paths with initial state $s$

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

- **Probability measure** $\Pr_s$
  - define probability $P_s(\omega)$ for finite path $\omega = ss_1...s_n$ as:
    - $P_s(\omega) = 1$ if $\omega$ has length one (i.e. $\omega = s$)
    - $P_s(\omega) = P(s,s_1) \cdot ... \cdot P(s_{n-1},s_n)$ otherwise
  - define $\Pr_s(C(\omega)) = P_s(\omega)$ for all finite paths $\omega$
  - $\Pr_s$ extends uniquely to a probability measure $\Pr_s : \Sigma_{\text{Path}(s)} \to [0,1]$  

- **See** [KSK76] for further details
Probability space – Example

• Paths where sending fails the first time
  - \( \omega = s_0s_1s_2 \)
  - \( C(\omega) = \) all paths starting \( s_0s_1s_2 \ldots \)
  - \( P_{s_0}(\omega) = P(s_0, s_1) \cdot P(s_1, s_2) \)
    \[ = 1 \cdot 0.01 = 0.01 \]
  - \( Pr_{s_0}(C(\omega)) = P_{s_0}(\omega) = 0.01 \)

• Paths which are eventually successful and with no failures
  - \( C(s_0s_1s_3) \cup C(s_0s_1s_1s_3) \cup C(s_0s_1s_1s_1s_3) \cup \ldots \)
  - \( Pr_{s_0}( C(s_0s_1s_3) \cup C(s_0s_1s_1s_3) \cup C(s_0s_1s_1s_1s_3) \cup \ldots ) \)
    \[ = P_{s_0}(s_0s_1s_3) + P_{s_0}(s_0s_1s_1s_3) + P_{s_0}(s_0s_1s_1s_1s_3) + \ldots \]
    \[ = 1 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.98 + 1 \cdot 0.01 \cdot 0.01 \cdot 0.98 + \ldots \]
    \[ = 0.9898989898\ldots \]
    \[ = 98/99 \]
Overview

- Discrete-time Markov chains (DTMCs)
- Properties of DTMCs: The logic PCTL
- PCTL model checking for DTMCs
- Beyond PCTL: Costs and rewards
- Tool support + A case study: contract signing
- Adding nondeterminism: Markov decision processes (MDPs)
PCTL

• **Temporal logic for describing properties of DTMCs**
  – $\text{PCTL} = \text{Probabilistic Computation Tree Logic}$ [HJ94]
  – essentially the same as the logic $\text{pCTL}$ 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**
  – $\text{send} \rightarrow P_{\geq 0.95} [\text{true U}^{\leq 10} \text{deliver}]$
  – “if a message is sent, then the probability of it being delivered within 10 steps is at least 0.95”
PCTL syntax

- **PCTL syntax:**

  \[ \phi ::= \text{true} | a | \phi \land \phi | \neg \phi | P_{\sim p} [ \psi ] \]  
  (state formulas)

  \[ \psi ::= X \phi | \phi U^{\leq k} \phi | \phi U \phi \]  
  (path formulas)

- 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**
  - \( s \models \phi \) denotes \( \phi \) is “true in state \( s \)” or “satisfied in state \( s \)”

- **Semantics of (non–probabilistic) state formulas:**
  - for a state \( s \) of the DTMC \((S, s_{\text{init}}, P, L)\):
    - \( s \models a \) \iff \( a \in L(s) \)
    - \( s \models \phi_1 \land \phi_2 \) \iff \( s \models \phi_1 \) and \( s \models \phi_2 \)
    - \( s \models \neg \phi \) \iff \( s \models \phi \) is false

- **Examples**
  - \( s_3 \models \text{succ} \)
  - \( s_1 \models \text{try} \land \neg \text{fail} \)
PCTL semantics for DTMCs

• **Semantics of path formulas:**
  - for a path $\omega = s_0s_1s_2...$ in the DTMC:
    - $\omega \models X \phi \iff s_1 \models \phi$
    - $\omega \models \phi_1 U^{\leq k} \phi_2 \iff \exists i \leq k \text{ such that } s_i \models \phi_2 \text{ and } \forall j < i, s_j \models \phi_1$
    - $\omega \models \phi_1 U \phi_2 \iff \exists k \geq 0 \text{ such that } \omega \models \phi_1 U^{\leq k} \phi_2$

• **Some examples of satisfying paths:**
  - $X \text{ succ}$
    {try} {succ} {succ} {succ}
    \[ S_1 \rightarrow S_3 \rightarrow S_3 \rightarrow S_3 \rightarrow \ldots \]
  - $\neg \text{fail} U \text{ succ}$
    {try} {try} {succ} {succ}
    \[ S_0 \rightarrow S_1 \rightarrow S_1 \rightarrow S_3 \rightarrow S_3 \rightarrow \ldots \]
PCTL semantics for DTMCs

- **Semantics of the probabilistic operator $P$**
  - Informal definition: $s \models P_{\neg p} [\psi]$ means that “the probability, from state $s$, that $\psi$ is true for an outgoing path satisfies $\neg p$”
  - Example: $s \models P_{<0.25} [X \text{ fail }] \iff$ “the probability of atomic proposition fail being true in the next state of outgoing paths from $s$ is less than 0.25”
  - Formally: $s \models P_{\neg p} [\psi] \iff \text{Prob}(s, \psi) \sim p$
  - Where: $\text{Prob}(s, \psi) = \Pr_s \{ \omega \in \text{Path}(s) \mid \omega \models \psi \}$

![Diagram showing PCTL semantics for DTMCs]
More PCTL…

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

• **Negation and probabilities**

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

• All the sets of paths expressed by PCTL are measurable
  – i.e. are elements of the $\sigma$–algebra $\Sigma_{\text{Path}(s)}$
  – see for example [Var85] (for a stronger result in fact)

• Recall: probability space $(\text{Path}(s), \Sigma_{\text{Path}(s)}, \text{Pr}_s)$
  – $\Sigma_{\text{Path}(s)}$ contains cylinder sets $C(\omega)$ for all finite paths $\omega$ 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)

• A PCTL property $P_{\sim_p} [ \psi ]$ is...
  – qualitative when $p$ is either 0 or 1
  – quantitative when $p$ is in the range (0,1)

• $P_{>0} [ F \phi ]$ is identical to $EF \phi$
  – there exists a finite path to a $\phi$–state

• $P_{\geq1} [ F \phi ]$ is (similar to but) weaker than $AF \phi$
  – e.g. $AF$ “tails” (CTL) $\neq P_{\geq1} [ F “tails” ]$ (PCTL)
Quantitative properties

- Consider a PCTL formula $P_{\sim p} \ [ \psi \ ]$
  - 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 $P =? \ [ \psi \ ]$
  - “what is the probability that path formula $\psi$ 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
  – \( P_{=?} [ F \text{ err/total} > 0.1 ] \)
  – “what is the probability that 10% of the NAND gate outputs are erroneous?”

• Bluetooth wireless communication protocol
  – \( P_{=?} [ F \leq t \text{ reply\_count} = k ] \)
  – “what is the probability that the sender has received \( k \) acknowledgements within \( t \) clock-ticks?”

• Security: EGL contract signing protocol
  – \( P_{=?} [ F (\text{pairs\_a} = 0 & \text{pairs\_b} > 0) ] \)
  – “what is the probability that the party B gains an unfair advantage during the execution of the protocol?”
Overview

• Discrete–time Markov chains (DTMCs)
• Properties of DTMCs: The logic PCTL
• PCTL model checking for DTMCs
• Beyond PCTL: Costs and rewards
• Tool support + A case study: contract signing
• Adding nondeterminism: Markov decision processes (MDPs)
PCTL model checking for DTMCs

• **Algorithm for PCTL model checking** [CY88,HJ94,CY95]
  – inputs: DTMC $D=\langle S, s_{\text{init}}, P, L \rangle$, PCTL formula $\phi$
  – output: $\text{Sat}(\phi) = \{ s \in S \mid s \models \phi \} = \text{set of states satisfying } \phi$

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

• **Sometimes, focus on quantitative results**
  – e.g. compute result of $P=? [ F \text{ error } ]$
  – e.g. compute result of $P=? [ F \leq k \text{ error } ]$ for $0 \leq k \leq 100$
PCTL model checking for DTMCs

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

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

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

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

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

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

Example:
$P_{\sim_p} [ \neg a U b ]$
Precomputation – Prob0

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

Example:

$P_{\sim p} [\neg a U b ]$
Precomputation – Prob0

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

Example:

- $P_{\sim p} [ \neg a \cup b ]$

- $\text{Sat}(P_{> 0} [ \neg a \cup b ])$
Precomputation – Prob0

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

Example:

$P_{\sim p} [ \neg a U b ]$

\[ S^{no} = \text{Sat}(P_{\leq 0} [ \neg a U b ]) \]
Precomputation – Prob1

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

Example:

$P_{\sim p} \lbrack \neg a \cup b \rbrack$
Precomputation – Prob1

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

Example:

$P_{\sim p} [ \neg a \cup b ]$

$S_{\text{no}} = \text{Sat}(P_{\leq 0} [ \neg a \cup b ])$

![Diagram of a graph showing states and transitions](image)
Precomputation – Prob1

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

Example:
$P_{\sim p} [\neg a U b ]$

![Diagram](image-url)
Precomputation – Prob1

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

Example:
$P_{\sim p} [\neg a \cup b ]$

![Diagram](image.png)

$S^{yes} = \text{Sat}(P_{\geq 1} [\neg a \cup b ])$. 

PCTL until – linear equations

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

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

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

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

• Example: $P_{\neg p} [\neg a \cup b ]$
• Let $x_s = \text{Prob}(s, \neg a \cup b)$

\[ S^{\text{no}} = \text{Sat}(P_{\leq 0} [\neg a \cup b ]) \]

\[
\begin{align*}
x_1 &= x_3 = 0 \\
x_4 &= x_5 = 1 \\
x_2 &= 0.1x_2 + 0.1x_3 + 0.3x_5 + 0.5x_4 = \frac{8}{9} \\
x_0 &= 0.1x_1 + 0.9x_2 = 0.8
\end{align*}
\]

$S^{\text{yes}} = \text{Sat}(P_{\geq 1} [\neg a \cup b ])$
PCTL model checking – Summary

• Computation of set $\text{Sat}(\Phi)$ for DTMC $D$ and PCTL formula $\Phi$
  – recursive descent of parse tree
  – combination of graph algorithms, numerical computation
  – complexity is linear in $|\Phi|$ and polynomial in $|S|$

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

• Discrete-time Markov chains (DTMCs)
• Properties of DTMCs: The logic PCTL
• PCTL model checking for DTMCs
• Beyond PCTL: Costs and rewards
• Tool support + A case study: contract signing
• Adding nondeterminism: Markov decision processes (MDPs)
Beyond PCTL

• PCTL, although useful in practice, has limited expressivity
  – essentially: probability of reaching states in \( X \), passing only through states in \( Y \), and (possibly) 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

• Both allow combinations of temporal operators
  – e.g. for liveness: \( P_{\sim_p} [ G F \phi ] \) – “...always eventually \( \phi \)”

• Model checking algorithms exist (but more expensive)
  – translate to Rabin automata, construct product DTMC, graph algorithms (BSCCs) + probabilistic reachability

• Another direction: extend DTMCs with 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 \((S, s_{\text{init}}, P, L)\), a reward structure is a pair \((\rho, \iota)\)
  - \(\rho : S \to \mathbb{R}_{\geq 0}\) is the state reward function (vector)
  - \(\iota : S \times S \to \mathbb{R}_{\geq 0}\) is the transition reward function (matrix)

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

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

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

\[ \phi ::= \ldots \mid P_{\sim p} [ \psi ] \mid R_{\sim r} [ I = k ] \mid R_{\sim r} [ C \leq k ] \mid R_{\sim r} [ F \phi ] \]

– where \( r \in \mathbb{R}_{\geq 0}, \sim \in \{<,>,\leq,\geq\}, k \in \mathbb{N} \)

• \( R_{\sim r} [ \cdot ] \) means “the expected value of \( \cdot \) satisfies \( \sim r \)”
Types of reward formulas

• Instantaneous: $R_{\sim r}[ I=k ]$
  - “the expected value of the state reward at time-step $k$ is $\sim r$”
  - 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 $\sim r$”
  - e.g. “the expected power consumption over one hour”

• Reachability: $R_{\sim r}[ F \phi ]$
  - “the expected reward cumulated before reaching a state satisfying $\phi$ is $\sim r$”
  - e.g. “the expected time for the algorithm to terminate”
Reward formula semantics

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

• **Recall:**
  – \( s \models P_p [ \psi ] \iff \Pr_s \{ \omega \in \text{Path}(s) \mid \omega \models \psi \} \sim p \)

• **For a state** \( s \) **in the DTMC:**
  – \( s \models R_r [ I=k ] \iff \text{Exp}(s, X_{I=k}) \sim r \)
  – \( s \models R_r [ C\leq k ] \iff \text{Exp}(s, X_{C\leq k}) \sim r \)
  – \( s \models R_r [ F \Phi ] \iff \text{Exp}(s, X_{F\Phi}) \sim r \)

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

• Definition of random variables:
  – for an infinite path \( \omega = s_0s_1s_2... \)

\[
X_{\omega_{\sim k}}(\omega) = \rho(s_k)
\]

\[
X_{\omega_{\sim k}}(\omega) = \begin{cases} 
0 & \text{if } k = 0 \\
\sum_{i=0}^{k-1} \rho(s_i) + \ell(s_i, s_{i+1}) & \text{otherwise}
\end{cases}
\]

\[
X_{\omega_{\sim k}}(\omega) = \begin{cases} 
0 & \text{if } s_0 \in \text{Sat}(\phi) \\
\infty & \text{if } s_i \not\in \text{Sat}(\phi) \text{ for all } i \geq 0 \\
\sum_{i=0}^{k_{\phi}-1} \rho(s_i) + \ell(s_i, s_{i+1}) & \text{otherwise}
\end{cases}
\]

– where \( k_{\phi} = \min\{ j \mid s_j \models \phi \} \)
Model checking reward properties

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

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

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

- Discrete-time Markov chains (DTMCs)
- Properties of DTMCs: The logic PCTL
- PCTL model checking for DTMCs
- Beyond PCTL: Costs and rewards
- Tool support + A case study: contract signing
- Adding nondeterminism: Markov decision processes (MDPs)
Tools – Probabilistic model checkers

- **PRISM (Probabilistic Symbolic Model Checker)**
  - DTMCs, MDPs, CTMCs + rewards, [Birmingham/Oxford]
- **MRMC (Markov Reward Model Checker)**
  - DTMCs, CTMCs + reward extensions, [Twente/Aachen]
- **LiQuor: LTL model checking for MDPs, Probmela language (probabilistic version of SPIN’s Promela), [Dresden]**

- Simulation-based probabilistic model checking:
  - APMC, Ymer (both based on PRISM language), VESTA
- Many other related tools/prototypes
  - RAPTURE, CADP, Möbius, APNN-Toolbox, SMART, GreatSPN, GRIP, CASPA, Premo, PASS, ...
The PRISM tool

• PRISM: Probabilistic symbolic model checker
  – developed at Birmingham/Oxford University, since 1999
  – free, open source (GPL), Linux/Unix/Mac/Windows/64-bit

• 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, simulator/debugger, graph plotting
  – efficient symbolic (BDD-based) implementation

• See: www.prismmodelchecker.org
PRISM modelling language

- Simple, state-based language for DTMCs/MDPs/CTMCs
  - based on Reactive Modules [AH99]
- Modules (system components, composed in parallel)
- Variables (finite-valued, local or global)
- Guarded commands (labelled with probabilities/rates)
- Synchronisation (CSP-style) + process-algebraic operators (parallel composition, action hiding/renaming)

\[
\text{[send]} \ (s=2) \rightarrow \ p_{\text{loss}} : (s' = 3) \& (\text{lost}' = \text{lost} + 1) \ + \ (1 - p_{\text{loss}}) : (s' = 4);
\]
// Herman's 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 has a token
    [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
Case study: Contract signing

- Two parties want to agree on a contract
  - each will sign if the other will sign, but do not trust each other
  - there may be a trusted third party (judge)
  - but it should only be used if something goes wrong

- In real life: contract signing with pen and paper
  - sit down and write signatures simultaneously

- On the Internet...
  - how to exchange commitments on an asynchronous network?
  - “partial secret exchange protocol” [EGL85]
Contract signing – EGL protocol

• Partial secret exchange protocol for 2 parties (A and B)

• A (B) holds 2N secrets \(a_1, \ldots, a_{2N}, (b_1, \ldots, b_{2N})\)
  – a secret is a binary string of length \(L\)
  – secrets partitioned into pairs: e.g. \(\{ (a_i, a_{N+i}) \mid i=1, \ldots, N \}\)
  – A (B) committed if B (A) knows one of A’s (B’s) pairs

• Uses “1–out–of–2 oblivious transfer protocol” \(OT(S,R,x,y)\)
  – S sends \(x\) and \(y\) to R
  – R receives \(x\) with probability \(\frac{1}{2}\) otherwise receives \(y\)
  – S does not know which one R receives
  – if S cheats then R can detect this with probability \(\frac{1}{2}\)
Contract signing – EGL protocol

(step 1)

for ( i=1,…,N )
    OT( A, B, a_i, a_{N+i} )
    OT( B, A, b_i, b_{N+i} )

(step 2)

for ( i=1,…,L ) (where L is the bit length of the secrets)
    for ( j=1,…,2N )
        A transmits bit i of secret a_j to B
    for ( j=1,…,2N )
        B transmits bit i of secret b_j to A
EGL protocol – Step 1

Party A

1...L

1...N

N+1...2N

1...L

1...N

N+1...2N

OT(A,B,a_i,a_{N+i})

OT(B,A,b_i,b_{N+i})

(repeat for i=1...N)
EGL protocol – Step 2

Party A

A sends bit $i$ of $a_j$ to B for $j=1\ldots2N$

Then B does the same for $b_j$

(repeat for $i=1\ldots L$)

Party B

(repeat for $i=1\ldots L$)
Contract signing – Results

• Modelled in PRISM as a DTMC (no concurrency) [NS06]

• Highlights a weakness in the protocol
  – party B can act maliciously by quitting the protocol early
  – this behaviour not considered in the original analysis

• PRISM analysis shows
  – if B stops participating in the protocol as soon as he/she has obtained one of A’s pairs, then, with probability 1, at this point:
    • B possesses a pair of A’s secrets
    • A does not have complete knowledge of any pair of B’s secrets
  – protocol is not fair under this attack:
    – B has a distinct advantage over A
Contract signing – Results

• The protocol is unfair because in step 2:
  – A sends a bit for each of its secret before B does

• Can we make this protocol fair by changing the message sequence scheme?

• Since the protocol is asynchronous the best we can hope for is
  – B (or A) has this advantage with probability \( \frac{1}{2} \)

• We consider 3 possible alternative message sequence schemes (EGL2, EGL3, EGL4)
(step 1)
...
(step 2)
for ( i=1,...,L )
  for ( j=1,...,N ) A transmits bit i of secret $a_j$ to B
  for ( j=1,...,N ) B transmits bit i of secret $b_j$ to A
  for ( j=N+1,...,2N ) A transmits bit i of secret $a_j$ to B
  for ( j=N+1,...,2N ) B transmits bit i of secret $b_j$ to A
Modified step 2 for EGL2

Party A

1...L

1...N

N+1...2N

A sends bit $i$ of $a_j$ to B for $j=1...N$

Then B does the same for $b_j$

(after $j=1...N$, send $j=N+1...2N$)

(then repeat for $i=1...L$)

Party B

1...L

1...N

N+1...2N
(step 1)
...
(step 2)
for (i=1,...,L) for (j=1,...,N)
  A transmits bit i of secret $a_j$ to B
  B transmits bit i of secret $b_j$ to A
for (i=1,...,L) for (j=N+1,...,2N)
  A transmits bit i of secret $a_j$ to B
  B transmits bit i of secret $b_j$ to A
Modified step 2 for EGL3

Party A

A sends bit $i$ of $a_j$ to B for

Then B does the same for $b_j$

(repeat for $j=1\ldots N$ and for $i=1\ldots L$)

(then send $j=N+1\ldots 2N$ for $i=1\ldots L$)
(step 1)
...
(step 2)
\textbf{for} \ (i=1,\ldots,L) \ 
A transmits bit i of secret $a_1$ to B \\
\hspace{1em} \textbf{for} \ (j=1,\ldots,N) \ B \ transmits \ bit \ i \ of \ secret \ b_j \ to \ A \\
\hspace{2em} \textbf{for} \ (j=2,\ldots,N) \ A \ transmits \ bit \ i \ of \ secret \ a_j \ to \ B \\
\textbf{for} \ (i=1,\ldots,L) \ 
A \ transmits \ bit \ i \ of \ secret \ a_{N+1} \ to \ B \\
\hspace{1em} \textbf{for} \ (j=N+1,\ldots,2N) \ B \ transmits \ bit \ i \ of \ secret \ b_j \ to \ A \\
\hspace{2em} \textbf{for} \ (j=N+2,\ldots,2N) \ A \ transmits \ bit \ i \ of \ secret \ a_j \ to \ B
Modified step 2 for EGL4

Party A

A sends bit \( i \) of \( a_j \) to B for \( j = 2 \ldots N \)

Then A sends bit \( i \) of \( a_j \) to B for \( j = 2 \ldots N \)

(repeat for \( i = 1 \ldots L \))

(then send \( j = N+1 \ldots 2N \) in same fashion)

Party B

Then B sends bit \( i \) of \( b_j \) to B for \( j = 1 \ldots N \)

B sends bit \( i \) of \( b_j \) to B for \( j = 1 \ldots N \)
Contract signing – Results

- The chance that the protocol is unfair
  - probability that one party gains knowledge first
  - $P_{=?} [ F \text{ know}_B \land \lnot \text{ know}_A ]$ and $P_{=?} [ F \text{ know}_A \land \lnot \text{ know}_B ]$
Contract signing – More properties

- (1) How unfair the protocol is to each party
  - expected number of bits that a party needs to know a pair once the other party knows a pair

- (2) The influence that each party has on the fairness
  - once a party knows a pair, the expected number of messages from this party required before the other party knows a pair

- (3) The duration of unfairness of the protocol
  - once a party knows a pair, the expected total number of messages that need to be sent before the other knows a pair
Overview

- Discrete-time Markov chains (DTMCs)
- Properties of DTMCs: The logic PCTL
- PCTL model checking for DTMCs
- Beyond PCTL: Costs and rewards
- Tool support + A case study: contract signing

- Adding nondeterminism: Markov decision processes (MDPs)
Other models

• What’s missing from DTMCs?

• Nondeterminism
  – Markov decision processes (MDPs)…

• Real–time
  – continuous–time Markov chains (CTMCs)
    • exponentially distributed delays
  – probabilistic timed automata (PTAs)
    • real–valued clocks, discrete probabilistic choice, nondeterminism
Nondeterminism

• Some aspects of a system may not be probabilistic and should not be modelled probabilistically; for example:

  • **Concurrency** – scheduling of parallel components
    – e.g. randomised distributed algorithms – multiple probabilistic processes operating asynchronously

  • **Underspecification** – unknown model parameters
    – e.g. a probabilistic communication protocol designed for message propagation delays of between $d_{\text{min}}$ and $d_{\text{max}}$

  • **Unknown environments**
    – e.g. probabilistic security protocols – unknown adversary
Markov decision processes

• **Markov decision processes (MDPs)**
  – extension of DTMCs which allow **nondeterministic choice**

• **Like DTMCs:**
  – discrete set of states representing possible configurations of the system being modelled
  – transitions between states occur in discrete time-steps

• **Probabilities and nondeterminism**
  – in each state, a nondeterministic choice between several discrete probability distributions over successor states
Markov decision processes

- Formally, an MDP $M$ is a tuple $(S, s_{\text{init}}, \text{Steps}, L)$ where:
  - $S$ is a finite set of states ("state space")
  - $s_{\text{init}} \in S$ is the initial state
  - $\text{Steps} : S \rightarrow 2^{\text{Act} \times \text{Dist}(S)}$ is the transition probability function
    where $\text{Act}$ is a set of actions and $\text{Dist}(S)$ is the set of discrete probability distributions over the set $S$
  - $L : S \rightarrow 2^\text{AP}$ is a labelling with atomic propositions

- Notes:
  - $\text{Steps}(s)$ is always non-empty, i.e. no deadlocks
  - the use of actions to label distributions is optional
Simple MDP example

- Modification of the simple DTMC communication protocol
  - after one step, process starts trying to send a message
  - then, a nondeterministic choice between: (a) waiting a step because the channel is unready; (b) sending the message
  - if the latter, with probability 0.99 send successfully and stop
  - and with probability 0.01, message sending fails, restart
Example – Parallel composition

Asynchronous parallel composition of two 3-state DTMCs

Action labels omitted here
Paths and probabilities

• **A (finite or infinite) path through an MDP**
  
  - is a sequence of states and action/distribution pairs
  
  - e.g. \( s_0(a_0,\mu_0)s_1(a_1,\mu_1)s_2... \)
  
  - such that \((a_i,\mu_i) \in \text{Steps}(s_i)\) and \(\mu_i(s_{i+1}) > 0\) for all \(i \geq 0\)
  
  - represents an execution (i.e. one possible behaviour) of the system which the MDP is modelling
  
  - note that a path resolves both types of choices: nondeterministic and probabilistic

• **To consider the probability of some behaviour of the MDP**
  
  - first need to resolve the nondeterministic choices
  
  - ...which results in a **DTMC**
  
  - ...for which we can define a probability measure over paths
Adversaries

- An adversary resolves nondeterministic choice in an MDP
- adversaries are also known as “schedulers” or “policies”

- Formally:
  - an adversary $A$ of an MDP $M$ is a function mapping every finite path $\omega = s_0(a_1, \mu_1)s_1...s_n$ to an element of $\text{Steps}(s_n)$

- For each $A$ can define a probability measure $\Pr^A_s$ over paths
  - constructed through an infinite state DTMC $(\text{Path}^A_{\text{fin}}(s), s, P^A_s)$
  - states of the DTMC are the finite paths of $A$ starting in state $s$
  - initial state is $s$ (the path starting in $s$ of length 0)
  - $P^A_s(\omega, \omega') = \mu(s)$ if $\omega' = \omega(a, \mu)s$ and $A(\omega) = (a, \mu)$
  - $P^A_s(\omega, \omega') = 0$ otherwise
Adversaries – Examples

• Consider the simple MDP below
  – note that $s_1$ is the only state for which $|\text{Steps}(s)| > 1$
  – i.e. $s_1$ is the only state for which an adversary makes a choice
  – let $\mu_b$ and $\mu_c$ denote the probability distributions associated with actions $b$ and $c$ in state $s_1$

• Adversary $A_1$
  – picks action $c$ the first time
  – $A_1(s_0s_1) = (c, \mu_c)$

• Adversary $A_2$
  – picks action $b$ the first time, then $c$
  – $A_2(s_0s_1) = (b, \mu_b)$, $A_2(s_0s_1s_1) = (c, \mu_c)$, $A_2(s_0s_1s_0s_1) = (c, \mu_c)$
Adversaries – Examples

- Fragment of DTMC for adversary $A_1$
  - $A_1$ picks action $c$ the first time
Adversaries – Examples

- Fragment of DTMC for adversary $A_2$
  - $A_2$ picks action $b$, then $c$
Simple adversaries

- **Simple adversaries** always pick same choice in a given state
  - formally, for adversary A:
  - $A(s_0(a_1,u_1)s_1...s_n)$ depends only on $s_n$
  - resulting DTMC can be mapped to a $|S|$-state DTMC

- From previous example:
  - adversary $A_1$ (picks c in $s_1$) is simple, $A_2$ is not
PCTL for MDPs

- The temporal logic PCTL can also describe MDP properties
- Identical syntax to the DTMC case:

\[
\phi ::= \text{true} \mid a \mid \phi \land \phi \mid \neg \phi \mid P_{\sim p}[\psi]
\]

\[
\psi ::= X \phi \mid \phi U \leq k \phi \mid \phi U \phi
\]

- Semantics are also the same as DTMCs for:
  - atomic propositions, logical operators, path formulas

\[\psi \text{ is true with probability } \sim p\]
PCTL semantics for MDPs

- **Semantics of the probabilistic operator P**
  - can only define probabilities for a specific adversary $A$
  - $s \models P_{\neg p}[\psi]$ means “the probability, from state $s$, that $\psi$ is true for an outgoing path satisfies $\neg p$ for all adversaries $A$”
  - formally $s \models P_{\neg p}[\psi] \iff \text{Prob}^A(s, \psi) \sim p$ for all adversaries $A$
  - where $\text{Prob}^A(s, \psi) = \text{Pr}^A_s \{ \omega \in \text{Path}^A(s) \mid \omega \models \psi \}$

$s \models P_{\neg \psi} \iff \text{Prob}^A(s, \neg \psi) \sim p$
Minimum and maximum probabilities

• Letting:
  - \( p_{\text{max}}(s, \psi) = \sup_A \text{Prob}^A(s, \psi) \)
  - \( p_{\text{min}}(s, \psi) = \inf_A \text{Prob}^A(s, \psi) \)

• We have:
  - if \( \sim \in \{\geq,>\} \), then \( s \models P_{\sim p}[\psi] \iff p_{\text{min}}(s, \psi) \sim p \)
  - if \( \sim \in \{<,\leq\} \), then \( s \models P_{\sim p}[\psi] \iff p_{\text{max}}(s, \psi) \sim p \)

• Model checking \( P_{\sim p}[\psi] \) reduces to the computation over all adversaries of either:
  - the minimum probability of \( \psi \) holding
  - the maximum probability of \( \psi \) holding

• Crucial result for model checking PCTL on MDPs
  - simple adversaries suffice, i.e. there are always simple adversaries \( A_{\text{min}} \) and \( A_{\text{max}} \) for which:
    - \( \text{Prob}^{A_{\text{min}}}(s, \psi) = p_{\text{min}}(s, \psi) \) and \( \text{Prob}^{A_{\text{max}}}(s, \psi) = p_{\text{max}}(s, \psi) \)
Quantitative properties

• For PCTL properties with P as the outermost operator
  – quantitative form (two types): \( P_{\text{min}} =? [ \psi ] \) and \( P_{\text{max}} =? [ \psi ] \)
  – i.e. “what is the minimum/maximum probability (over all adversaries) that path formula \( \psi \) is true?”
  – corresponds to an analysis of best-case or worst-case behaviour of the system
  – model checking is no harder since compute the values of \( p_{\text{min}}(s, \psi) \) or \( p_{\text{max}}(s, \psi) \) anyway
  – useful to spot patterns/trends

• Example: CSMA/CD protocol
  – “min/max probability that a message is sent within the deadline”
Other classes of adversary

- A more general semantics for PCTL over MDPs
  - parameterise by a class of adversaries Adv

- Only change is:
  - \( s \models_{\text{Adv}} P_{\sim p} [\psi] \iff \operatorname{Prob}^{A}(s, \psi) \sim p \) for all adversaries \( A \in \text{Adv} \)

- Original semantics obtained by taking Adv to be the set of all adversaries for the MDP

- Alternatively, take Adv to be the set of all fair adversaries
  - path fairness: if a state is occurs on a path infinitely often, then each non-deterministic choice occurs infinite often
  - see e.g. [BK98]
Some real PCTL examples

- **Byzantine agreement protocol**
  - $P_{\min} \subseteq \tau [ F (\text{agreement} \land \text{rounds} \leq 2) ]$
  - “what is the minimum probability that agreement is reached within two rounds?”

- **CSMA/CD communication protocol**
  - $P_{\max} \subseteq \tau [ F \text{collisions}=\text{k} ]$
  - “what is the maximum probability of k collisions?”

- **Self-stabilisation protocols**
  - $P_{\min} \subseteq \tau [ F \leq t \text{ stable} ]$
  - “what is the minimum probability of reaching a stable state within k steps?”
PCTL model checking for MDPs

- **Algorithm for PCTL model checking** [BdA95]
  - inputs: MDP $M=(S,s_{init},Steps,L)$, PCTL formula $\phi$
  - output: $\text{Sat}(\phi) = \{ s \in S \mid s \models \phi \}$ = set of states satisfying $\phi$

- **As for PCTL model checking for DTMCs**
  - sometimes check: $s \models \phi$ for all $s \in S$, sometimes: $s_{init} \models \phi$
  - or compute quantitative results
    - e.g. compute result of $\text{P}_{\text{max}}=\? [ F_{\leq k} \text{ error } ]$ for $0 \leq k \leq 100$

- **Basic algorithm proceeds by induction on parse tree of the PCTL formula $\phi$**
PCTL model checking for MDPs

- Only need to consider $P_{\sim_p} [ \psi ]$ formulas
  - reduces to computation of $p_{\min}(s, \psi)$ or $p_{\max}(s, \psi)$ for all $s \in S$
  - dependent on whether $\sim \in \{\geq, >\}$ or $\sim \in \{<, \leq\}$

- Here, we cover the algorithm for computing $p_{\min}(s, \psi)$
  - i.e. the case where $\sim \in \{\geq, >\}$
  - computation of $p_{\max}(s, \psi)$ is very similar

- Focus on until formulas, i.e. $p_{\min}(s, \phi_1 U \phi_2)$
  - next ($X \phi$) and bounded until ($\phi_1 U_{\leq k} \phi_2$) are simple extensions of DTMC case
  - see e.g. [BdA95], [KNP04a] for further details
PCTL until for MDPs

- Computation of probabilities $p_{min}(s, \phi_1 U \phi_2)$ for all $s \in S$
- First identify all states where the probability is 1 or 0
  - “precomputation” algorithms, yielding sets $S^{yes}$, $S^{no}$
- Then compute (min) probabilities for remaining states ($S^?$)
  - either: solve linear optimisation problem
  - or: approximate with an iterative solution method

Example:

$$P \geq p \left[ F \text{ goal } \right] \equiv P \geq p \left[ \text{true } U \text{ goal } \right]$$
PCTL until – Precomputation

- Identify all states where $p_{\text{min}}(s, \phi_1 U \phi_2)$ is 1 or 0
  - $S_{\text{yes}} = \text{Sat}(P \geq 1 [\phi_1 U \phi_2])$, $S_{\text{no}} = \text{Sat}(\neg P > 0 [\phi_1 U \phi_2])$

- Two graph-based precomputation algorithms:
  - algorithm Prob1A computes $S_{\text{yes}}$
    - for all adversaries the probability of satisfying $\phi_1 U \phi_2$ is 1
  - algorithm Prob0E computes $S_{\text{no}}$
    - there exists an adversary for which the probability is 0

Example:

$P \geq p [ F \text{ goal } ]$

$S_{\text{yes}} = \text{Sat}(P \geq 1 [ F \text{ goal } ])$

$S_{\text{no}} = \text{Sat}(\neg P > 0 [ F \text{ goal } ])$
Method 1 – Linear optimisation problem

• Probabilities $p_{\text{min}}(s, \phi_1 \cup \phi_2)$ for remaining states in the set $S' = S \setminus (S^{\text{yes}} \cup S^{\text{no}})$ can be obtained as the unique solution of the following linear optimisation problem:

$$\text{maximize } \sum_{s \in S'} x_s \text{ subject to the constraints :}$$

$$x_s \leq \sum_{s' \in S'} \mu(s') \cdot x_{s'} + \sum_{s' \in S^{\text{yes}}} \mu(s')$$

for all $s \in S'$ and for all $(a, \mu) \in \text{Steps}(s)$

• Simple case of a more general problem known as the stochastic shortest path problem [BT91]

• This can be solved with standard techniques
  – e.g. Simplex, ellipsoid method
PCTL until – Example

Let \( x_i = p_{\text{min}}(s_i, F \text{ goal}) \)

\( S^{\text{yes}}: x_2 = 1 \), \( S^{\text{no}}: x_3 = 0 \)

For \( S^? = \{x_0, x_1\} \):

Maximise \( x_0 + x_1 \) subject to constraints:

- \( x_0 \leq x_1 \)
- \( x_0 \leq 0.25 \cdot x_0 + 0.5 \)
- \( x_1 \leq 0.1 \cdot x_0 + 0.5 \cdot x_1 + 0.4 \)
PCTL until – Example

Let $x_i = \min(s_i, F \text{ goal})$

$S^\text{yes}$: $x_2=1$, $S^\text{no}$: $x_3=0$

For $S? = \{x_0, x_1\}$:

Maximise $x_0 + x_1$ subject to constraints:

- $x_0 \leq x_1$
- $x_0 \leq \frac{2}{3}$
- $x_1 \leq 0.2 \cdot x_0 + 0.8$
Let $x_i = p_{\text{min}}(s_i, F_{\text{goal}})$

$S^{\text{yes}}: x_2 = 1$, $S^{\text{no}}: x_3 = 0$

For $S = \{x_0, x_1\}$:

Maximise $x_0 + x_1$ subject to constraints:

- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2 \cdot x_0 + 0.8$

Solution: $(x_0, x_1) = (2/3, 14/15)$
Let $x_i = p_{\text{min}}(s_i, F \text{ goal})$

$S^{\text{yes}}$: $x_2 = 1$, $S^{\text{no}}$: $x_3 = 0$

For $S^? = \{x_0, x_1\}$:

Maximise $x_0 + x_1$ subject to constraints:

- $x_0 \leq x_1$
- $x_0 \leq 2/3$
- $x_1 \leq 0.2 \cdot x_0 + 0.8$

Two simple adversaries
Method 2 – Iterative solution

- For probabilities $p_{\text{min}}(s, \phi_1 \cup \phi_2)$ it can be shown that:

$$- p_{\text{min}}(s, \phi_1 \cup \phi_2) = \lim_{n \to \infty} x_s^{(n)}$$

where:

$$x_s^{(n)} = \begin{cases} 
1 & \text{if } s \in S^{\text{yes}} \\
0 & \text{if } s \in S^{\text{no}} \\
0 & \text{if } s \in S^? \text{ and } n = 0 \\
\min_{(a, \mu) \in \text{Steps}(s)} \left( \sum_{s' \in S} \mu(s') \cdot x_{s'}^{(n-1)} \right) & \text{if } s \in S^? \text{ and } n > 0 
\end{cases}$$

- This forms the basis for an (approximate) iterative solution
  - iterations terminated when solution converges sufficiently
  - equivalent to well–known “value iteration” method for MDPs
PCTL until – Example

Compute: $p_{\min}(s_i, F \text{ goal})$

$S^{\text{yes}} = \{x_2\}$, $S^{\text{no}} = \{x_3\}$, $S^? = \{x_0, x_1\}$

$[x_0^{(n)}, x_1^{(n)}, x_2^{(n)}, x_3^{(n)}]$

$n=0$: $[0, 0, 1, 0]$

$n=1$: $[\min(0, 0.25 \cdot 0 + 0.5),$

$0.1 \cdot 0 + 0.5 \cdot 0 + 0.4, 1, 0]$

$= [0, 0.4, 1, 0]$

$n=2$: $[\min(0.4, 0.25 \cdot 0 + 0.5),$

$0.1 \cdot 0 + 0.5 \cdot 0.4 + 0.4, 1, 0]$

$= [0.4, 0.6, 1, 0]$

$n=3$: $...$
PCTL until – Example

\[
\begin{align*}
\text{n=0:} & \quad [0.000000, 0.000000, 1, 0] \\
\text{n=1:} & \quad [0.000000, 0.400000, 1, 0] \\
\text{n=2:} & \quad [0.400000, 0.600000, 1, 0] \\
\text{n=3:} & \quad [0.600000, 0.740000, 1, 0] \\
\text{n=4:} & \quad [0.650000, 0.830000, 1, 0] \\
\text{n=5:} & \quad [0.662500, 0.880000, 1, 0] \\
\text{n=6:} & \quad [0.665625, 0.906250, 1, 0] \\
\text{n=7:} & \quad [0.666406, 0.919688, 1, 0] \\
\text{n=8:} & \quad [0.666602, 0.926484, 1, 0] \\
\text{n=9:} & \quad [0.666650, 0.929902, 1, 0] \\
\cdots & \quad \text{...} \\
\text{n=20:} & \quad [0.666667, 0.933332, 1, 0] \\
\text{n=21:} & \quad [0.666667, 0.933332, 1, 0] \\
\end{align*}
\]

\[
\approx [2/3, 14/15, 1, 0]
\]
### PCTL until – Example

![Graph showing PCTL until Example](image)

<table>
<thead>
<tr>
<th>n</th>
<th>[x_0^{(n)}, x_1^{(n)}, x_2^{(n)}, x_3^{(n)}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>n=0</td>
<td>[0.000000, 0.000000, 1, 0]</td>
</tr>
<tr>
<td>n=1</td>
<td>[0.000000, 0.400000, 1, 0]</td>
</tr>
<tr>
<td>n=2</td>
<td>[0.400000, 0.600000, 1, 0]</td>
</tr>
<tr>
<td>n=3</td>
<td>[0.600000, 0.740000, 1, 0]</td>
</tr>
<tr>
<td>n=4</td>
<td>[0.650000, 0.830000, 1, 0]</td>
</tr>
<tr>
<td>n=5</td>
<td>[0.662500, 0.880000, 1, 0]</td>
</tr>
<tr>
<td>n=6</td>
<td>[0.665625, 0.906250, 1, 0]</td>
</tr>
<tr>
<td>n=7</td>
<td>[0.666406, 0.919688, 1, 0]</td>
</tr>
<tr>
<td>n=8</td>
<td>[0.666602, 0.926484, 1, 0]</td>
</tr>
<tr>
<td>n=9</td>
<td>[0.666650, 0.929902, 1, 0]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>n=20</td>
<td>[0.666667, 0.933332, 1, 0]</td>
</tr>
<tr>
<td>n=21</td>
<td>[0.666667, 0.933332, 1, 0]</td>
</tr>
</tbody>
</table>

\[\approx [2/3, 14/15, 1, 0]\]
Costs and rewards

- Can use costs and rewards in similar fashion to DTMCs:

- Augment MDPs with rewards (or costs)
  - (but often assign to states/actions, not states/transitions)

- Extend logic PCTL with R operator
  - semantics extended in same way as P operator
  - e.g. \( s \models R_{\sim r} [ F \phi ] \iff \text{Exp}^A(s, X_{F\phi}) \sim r \) for all adversaries \( A \)
  - quantitative properties: \( R_{\min} \subseteq \cdots \) and \( R_{\max} \subseteq \cdots \)

- Examples:
  - “the minimum expected queue size after exactly 90 seconds”
  - “the maximum expected power consumption over one hour”
  - the maximum expected time for the algorithm to terminate
Model checking MDP reward formulas

• **Instantaneous:** $R_{\sim r} [ I=k ]$
  – similar to the computation of bounded until probabilities
  – solution of **recursive equations**

• **Cumulative:** $R_{\sim r} [ C\leq k ]$
  – extension of bounded until computation
  – solution of **recursive equations**

• **Reachability:** $R_{\sim r} [ F \phi ]$
  – similar to the case for $P$ operator and until
  – graph–based precomputation (identify $\infty$–reward states)
  – then **linear optimization problem** (or iterative solution)
MDP model checking – Summary

• Computation of set Sat(Φ) for MDP M and PCTL formula Φ
  – recursive descent of parse tree
  – combination of graph algorithms, numerical computation
  – complexity is linear in |Φ| and polynomial in |S|
  – S is states in MDP, assume |Steps(s)| is constant

• Probabilistic operator P:
  – X Φ: one matrix–vector multiplication, \( O(|S|^2) \)
  – \( \Phi_1 \cup^k \Phi_2 \): k matrix–vector multiplications, \( O(k|S|^2) \)
  – \( \Phi_1 \cup \Phi_2 \): linear optimisation problem, polynomial in |S|

• Expected reward operator R
  – I=^k: k matrix–vector multiplications, \( O(k|S|^2) \)
  – C\leq^k: k iterations of matrix–vector multiplication + summation
  – F Φ: linear optimisation problem in at most |S| variables
Summary

• Probabilistic model checking
  – automated quantitative verification of stochastic systems
  – to model randomisation, failures, ...

• Probabilistic models
  – discrete-time Markov chains (DTMCs)
  – Markov decision processes (MDPs)

• Property specifications:
  – probabilistic extensions of temporal logic, e.g. PCTL
  – expected value of costs/rewards

• Model checking algorithms
  – combination of graph-based algorithms, numerical
    computation (linear equations, linear optimisation, …)

• Tool support, case studies
  – PRISM, randomised contract signing algorithm
Further information

• **Slides from full lecture course at:**
  – [www.prismmodelchecker.org/lectures/](http://www.prismmodelchecker.org/lectures/)
  – DTMCs/MDPs/CTMCs/PTAs
  – case studies, implementation, advanced topics

• **See also the PRISM web site:**
  – [www.prismmodelchecker.org](http://www.prismmodelchecker.org)
  – related publications
  – case study repository
  – tool download, tutorial, manual
  – and much more...