Probabilistic Model Checking

Lecture 6

Costs & Rewards

Alessandro Abate

Department of Computer Science
University of Oxford
Overview

• Specifying costs and rewards
  – DTMCs
  – hints at syntax for PRISM language

• Properties: expected reward values
  – instantaneous
  – cumulative
  – reachability
  – temporal logic extensions

• Model checking
  – computing reward values

• Case study
  – randomised contract signing
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 range of quantitative measures of the system: notion of expected value of rewards
  – (alternative reward structures possible, e.g., based on var)
  – rewards as specifications in an extension of PCTL

• More precisely, we use two distinct property classes:
  • Instantaneous properties
    – e.g. the expected value of the reward at given time point
  • Cumulative properties
    – e.g. the expected cumulated reward over a period/horizon
DTMC reward structures

• For a DTMC \((S, s_{\text{init}}, P, L)\), a reward structure is a pair \((\rho, \iota)\)
  
  - \(\rho : S \rightarrow \mathbb{R}_{\geq 0}\) 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”: \(\rho\) maps each state to the number of jobs in the queue, \(\iota\) is not used (equal to zero everywhere)

• 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
**Expected reward properties**

- **Expected ("average") values of rewards…**
  - **Instantaneous**
    - “the expected value of the state reward at time-step k”
    - e.g. “the expected nr. of jobs at exactly 90 seconds after start”

- **Cumulative (time-bounded)**
  - “the expected reward cumulated up to time-step k”
  - e.g. “the expected power consumption accrued over one hour”

- **Reachability (also cumulative)**
  - “the expected reward cumulated before reaching states $T \subseteq S$”
  - e.g. “the expected time for the algorithm to terminate”
Expectation

• **Probability space** \( (\Omega, \Sigma, \Pr) \)
  – probability measure \( \Pr : \Sigma \rightarrow [0,1] \)

• **Random variable** \( X \)
  – a measurable function \( X : \Omega \rightarrow \Delta \)
  – usually real-valued, i.e.: \( X : \Omega \rightarrow \mathbb{R} \)

• **Expected (“average”) value of the random variable:** \( \text{Exp}(X) \)

\[
\text{Exp}(X) = \sum_{\omega \in \Omega} X(\omega) \cdot \Pr(\omega)
\]

\( \text{Exp}(X) = \int_{\omega \in \Omega} X(\omega) \, d\Pr \)

**discrete case**
Reachability + rewards

- Expected reward cumulated before reaching states \( T \subseteq S \)
- Define a random variable:
  - \( X_{\text{Reach}(T)} : \text{Path}(s) \rightarrow \mathbb{R}_{\geq 0} \)
  - where for an infinite path \( \omega = s_0s_1s_2... \)
    \[
    X_{\text{Reach}(T)}(\omega) = \begin{cases} 
      0 & \text{if } s_0 \in T \\
      \infty & \text{if } s_i \notin T \text{ for all } i \geq 0 \\
      \sum_{i=0}^{k_T-1} \rho(s_i) + \ell(s_i, s_{i+1}) & \text{otherwise}
    \end{cases}
    \]
  - where \( k_T = \min \{ j \mid s_j \in T \} \)
- Then define:
  - \( \text{ExpReach}(s, T) = \text{Exp}(s, X_{\text{Reach}(T)}) \)
  - denoting: expectation of the random variable \( X_{\text{Reach}(T)} \)
    with respect to the probability measure \( \text{Pr}_s \), i.e.:
    \[
    \int_{\omega \in \text{Path}(s)} X_{\text{Reach}(T)}(\omega) \, d\text{Pr}_s
    \]
Computing the rewards

• Determine states for which ProbReach(s, T) = 1

• Solve linear equation system:

\[
\text{ExpReach}(s, T) = \begin{cases} 
\infty & \text{if ProbReach}(s, T) < 1 \\
0 & \text{if } s \in T \\
\rho(s) + \sum_{s' \in S} P(s, s') \cdot (\nu(s, s') + \text{ExpReach}(s', T)) & \text{otherwise}
\end{cases}
\]
Example

- Let $\rho = [0, 1, 0, 0]$ and $\iota(s, s') = 0$ for all $s, s' \in S$
- Compute $\text{ExpReach}(s_0, \{s_3\})$
  - (“expected number of times pass through $s_1$ to get to $s_3$”)
- First check:
  - $\text{ProbReach}(\{s_3\}) = \{1, 1, 1, 1\}$
- Then solve linear equation system:
  - (letting $x_i = \text{ExpReach}(s_i, \{s_3\})$):
    - $x_0 = 0 + 1 \cdot (0 + x_1)$
    - $x_1 = 1 + 0.01 \cdot (0 + x_2) + 0.01 \cdot (0 + x_1) + 0.98 \cdot (0 + x_3)$
    - $x_2 = 0 + 1 \cdot (0 + x_0)$
    - $x_3 = 0$
  - Solution: $\text{ExpReach}(\{s_3\}) = [100/98, 100/98, 100/98, 0]$
- So: $\text{ExpReach}(s_0, \{s_3\}) = 100/98 \approx 1.020408$
Specifying reward properties in PRISM

• PRISM extends PCTL to include expected reward 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 \( \sim r \)”
Random variables for reward formulae

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

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

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

  \[ X_{F\phi}(\omega) = \begin{cases} 
  \sum_{i=0}^{k_{\phi}-1} \rho(s_i) + \iota(s_i, s_{i+1}) & \text{otherwise}
  \end{cases} \]

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

  $X_{F\phi}$ same as $X_{\text{Reach}(\text{Sat}(\phi))}$ from earlier
Reward formula semantics

- Formal semantics of the three reward operators:

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

  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 $\text{Pr}_s$

- We can also define $R_\approx [...]$ properties, as for the $P$ operator
  
  - e.g. $R_\approx [ F \Phi ]$ returns the value $\text{Exp}(s, X_{\Phi})$

\[\text{Exp}(s, X_{\Phi})\]
\text{same as}
\[\text{ExpReach}(s, \text{Sat}(\Phi))\]
seen earlier
Model checking reward operators

- As for model checking $P_{\neg p} [...]$, in order to check $R_{\neg r} [...]$
  - compute reward values for all states, compare with bound $r$

- Instantaneous: $R_{\neg r} [ I = k ]$ – compute $\text{Exp}(X_{I = k})$
  - solution of recursive equations
  - essentially: $k$ matrix–vector multiplications

- Cumulative: $R_{\neg r} [ C \leq k ]$ – compute $\text{Exp}(X_{C \leq k})$
  - solution of recursive equations
  - essentially: $k$ matrix–vector multiplications

- Reachability: $R_{\neg r} [ F \varphi ]$ – compute $\text{Exp}(X_{F \varphi})$
  - graph analysis + solution of linear system of equations
  - (see computation of $\text{ExpReach}(s, T)$ earlier)

Model checking $R$ operator has same complexity as $P$ operator
Model checking $R_{\sim r}[ I=k ]$

- Expected instantaneous reward at step $k$
  - can be defined in terms of transient probabilities for step $k$

\[
\text{Exp}(s, X_{I=k}) = \sum_{s' \in S} \pi_{s,k}(s') \cdot \rho(s')
\]

- \( \text{Exp}(X_{I=k}) = P^k \cdot \rho \)

- Yielding recursive definition:
  - \( \text{Exp}(X_{I=0}) = \rho \)
  - \( \text{Exp}(X_{I=k}) = P \cdot \text{Exp}(X_{I=(k-1)}) \)
  - i.e. $k$ matrix–vector multiplications
  - note: “backward” computation (like bounded–until prob) rather than “forward” computation (like transient probs)
Example

- Let $\rho = [0, 1, 0, 0]$ and $\iota(s, s') = 0$ for all $s, s' \in S$
- Compute $\text{Exp}(s_0, X_{I=2})$
  - ("probability of being in state $s_1$ at time 2")
  - $\text{Exp}(X_{I=0}) = [0, 1, 0, 0]$
  - $\text{Exp}(X_{I=1}) = P \cdot \text{Exp}(X_{I=0})$
    \[
    \begin{bmatrix}
      0 & 1 & 0 & 0 \\
      0 & 0.01 & 0.01 & 0.98 \\
      1 & 0 & 0 & 0 \\
      0 & 0 & 0 & 1
    \end{bmatrix}
    \begin{bmatrix}
      0 \\
      1 \\
      0 \\
      0
    \end{bmatrix}
    =
    \begin{bmatrix}
      1 \\
      0.01 \\
      0 \\
      0
    \end{bmatrix}
    \]
  - $\text{Exp}(X_{I=2}) = P \cdot \text{Exp}(X_{I=1})$
    \[
    \begin{bmatrix}
      0 & 1 & 0 & 0 \\
      0 & 0.01 & 0.01 & 0.98 \\
      1 & 0 & 0 & 0 \\
      0 & 0 & 0 & 1
    \end{bmatrix}
    \begin{bmatrix}
      1 \\
      0.01 \\
      0.01 \\
      0.0001
    \end{bmatrix}
    =
    \begin{bmatrix}
      0.01 \\
      0.0001 \\
      0.0001 \\
      1
    \end{bmatrix}
    \]
- Result: $\text{Exp}(s_0, X_{I=2}) = 0.01$
Model checking $R_{~r} [ C^{\leq k} ]$

- Expected reward cumulated up to time step $k$

- Again, a recursive definition:

$$\text{Exp}(s, X_{C^{\leq k}}) = \begin{cases} 
\rho(s) + \sum_{s' \in S} P(s, s') \cdot (t(s, s') + \text{Exp}(s', X_{C^{\leq k-1}})) & \text{if } k > 0 \\
0 & \text{if } k = 0 
\end{cases}$$

- And in matrix/vector notation:

$$\overset{\text{}\text{Exp}}{\text{Exp}(X_{C^{\leq k}})} = \begin{cases} 
\rho + (P \circ \mathbf{1}) \cdot \mathbf{1} + P \cdot \overset{\text{}\text{Exp}}{\text{Exp}(X_{C^{\leq k-1}})} & \text{if } k > 0 \\
0 & \text{if } k = 0 
\end{cases}$$

- where $\circ$ denotes Schur (pointwise) matrix multiplication
- and $\mathbf{1}$ is a unit vector (of all 1s)
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” $\text{OT}(S,R,x,y)$
  - Sender $S$ sends $x$ and $y$ to receiver $R$
  - $R$ 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 $\frac{1}{2}$
EGL protocol – Step 1

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

Party A

1…L

1…N

N+1…2N

A sends bit $i$ of $a_j$ to B for $j=1…2N$

Then B does the same for $b_j$

(repeat over $i=1…L$)

Party B

1…L

1…N

N+1…2N
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 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 secrets 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

A sends bit \( i \) of \( a_j \) to B for \( j = 1 \ldots N \)

Then B does the same for \( b_j \)

(after \( j = 1 \ldots N \), send \( j = N+1 \ldots 2N \))

(then repeat over \( i = 1 \ldots L \))
Contract signing – EGL3

(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

\(a_j \) to B

A sends bit \( i \)

\(1 \ldots L \)

\(1 \ldots N \)

\(N+1 \ldots 2N \)

Then B does the same for \( b_j \)

\(1 \ldots L \)

\(1 \ldots N \)

\(N+1 \ldots 2N \)

(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)
for (i=1,…,L)
    A transmits bit i of secret $a_1$ to B
for (j=1,…,N) B transmits bit i of secret $b_j$ to A
for (j=2,…,N) A transmits bit i of secret $a_j$ to B
for (i=1,…,L)
    A transmits bit i of secret $a_{N+1}$ to B
for (j=N+1,…,2N) B transmits bit i of secret $b_j$ to A
for (j=N+2,…,2N) A transmits bit i of secret $a_j$ to B
Modified step 2 for EGL4

Party A

1...L

1...N

N+1...2N

A sends bit \( i \) of \( a_1 \) to B

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

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

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

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

Party B

1...L

1...N

N+1...2N
Contract signing – Results

• The chance that the protocol is unfair ($N = \text{secrets}$)
  – probability that one party gains knowledge first
  – $P = \mathbb{P}[F(\text{know}_B \land \neg\text{know}_A)]$ and $P = \mathbb{P}[F(\text{know}_A \land \neg\text{know}_B)]$
Contract signing – Results

- 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

\[ R = ? \ [ F \text{ know}_A ] \]

Reward structure:

Assign 1 to transitions corresponding to messages being sent from B to A after B knows a pair

(and 0 to all other transitions)
Contract signing – Results

- 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

\[ R = ? \left[ F \text{ know}_A \right] \]

Reward structure:

Assign 1 to transitions corresponding to any message being sent between A and B after B knows a pair

(and 0 to all other transitions)
Contract signing – Results

• Results show EGL4 is the ‘fairest’ protocol

• Except for measure of “duration of unfairness”
  – expected messages that need to be sent for a party to know a pair once the other party knows a pair
  – this value is larger for B than for A
  – and, in fact, as N increases, this measure:
    - increases for B
    - decreases for A

• Solution:
  – if a party sends a sequence of bits in a row (without the other party sending messages in between), require that the party send these bits as a single message
Contract signing – Results

• The duration of unfairness of the protocol
  – (with the solution on the previous slide applied to all variants)
Summing up…

• Costs and rewards
  – real-valued assigned to states/transitions of a DTMC

• Properties
  – expected instantaneous/cumulative reward values
  – PRISM property specifications: adds R operator to PCTL

• Model checking
  – instantaneous: matrix–vector multiplications
  – cumulative: matrix–vector multiplications
  – reachability: graph analysis + linear equation systems

• Case study
  – randomised contract signing