Quantitative Verification

Chapter 4: Markov decision processes

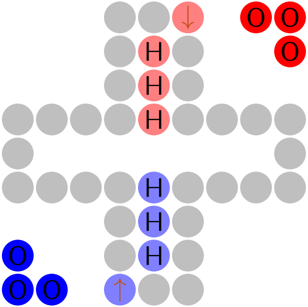
Jan Křetínský

Technical University of Munich

Winter 2021/22

Discrete-time
Markov Decision Processes
MDP

MDP



DTMC – purely probabilistic

Possible successor states are chosen based on probabilities but not on decisions.

We want decisions

to model both

- controllable setting (game theory, operations theory, control theory);
- uncontrollable setting (interleaving in concurrent systems, abstractions of models, open systems)

How to introduce decisions, i.e., non-determinism, to DTMC?

MDP: Definition

Definition:

A (labelled) Markov Decision Process (MDP) is a tuple

$$\mathcal{M} = (S, Act, \mathrm{P}, \pi_0, L)$$

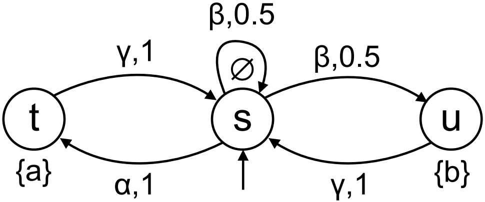
where

- S is a countable set of states,
- Act is a finite set of actions,
- $\mathrm{P} : S \times Act \times S \to [0,1]$ is the transition probability function, such that for each state s and action α,
  - $\sum_{s' \in S} \mathrm{P}(s, \alpha, s') = 1$, then we say that α is enabled in s; or
  - $\mathrm{P}(s, \alpha, s') = 0$ for all s', then we say that α is not enabled in s.
- π_0 is the initial distribution, and
- $L : S \to 2^{AP}$ is the labeling function.

The set of actions enabled in s is denoted by $Act(s)$. We assume that for each s, we have $Act(s) \neq \emptyset$.

MDP – Schedulers

Example:

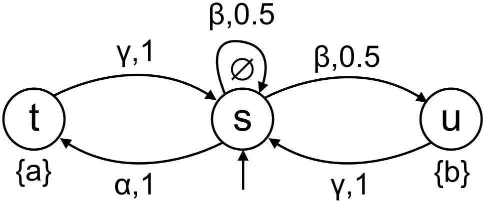


Problem:

How is the non-determinism resolved?

MDP – Schedulers

Example:



Problem:
How is the non-determinism resolved?
Allowing memory and randomness:

Definition (Scheduler):
A scheduler (also called strategy or policy) on an MDP $\mathcal{M} = (S, Act, \mathrm{P}, \pi_0, L)$ is a function Θ assigning to each history $s_0 \cdots s_n \in S^+$ a probability distribution over Act such that α is enabled in s_n whenever $\Theta(s_0 \cdots s_n)(\alpha) > 0$.

MDP – Schedulers

Definition (Induced DTMC):
Let $\mathcal{M} = (S, Act, \mathsf{P}, \pi_0, L)$ be a MDP and scheduler Θ on $\mathcal{M}$. The induced DTMC is given by

$$\mathcal{M}^{\Theta} = (S^{+}, \mathsf{P}^{\Theta}, \pi_0, L'),$$

where for any $h = s_0 s_1 \ldots s_n$, we define

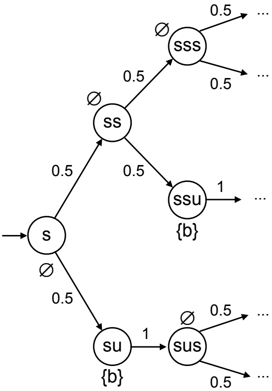
$$\mathsf{P}^{\Theta}(h, hs_{n+1}) = \sum_{\alpha \in Act} \Theta(h)(\alpha) \cdot \mathsf{P}(s_n, \alpha, s_{n+1})$$

and $L'(h) = L(s_n)$.

MDP – Schedulers

Example:

We choose a scheduler Θ that always takes action β in state s and action γ in state u. The induced DTMC $\mathcal{M}^{\Theta}$ for the previous example:



Notation

- P^{Θ} – the probability measure of $\mathcal{M}^{\Theta}$
- There is a bijection ξ mapping each sequence of states $s_0 s_1 s_2 \cdots$ to a sequence of histories $s_0 \;\; s_0 s_1 \;\; s_0 s_1 s_2 \;\; \cdots$ (a path of $\mathcal{M}^{\Theta}$).
- When using previous notation for sets of paths such as $\Diamond B$, we actually mean $\xi(\Diamond B)$

MDP – Schedulers

Classes of schedulers:

- A scheduler Θ is memoryless if for histories $s_0 s_1 \dots s_n \in S^+$ and $s'_0 s'_1 \dots s_n \in S^+$ with $s_n = s'_n$ it holds

$$\Theta(s_0 s_1 \dots s_n) = \Theta(s'_0 s'_1 \dots s'_n).$$

- A scheduler Θ is deterministic if for all histories $s_0 s_1 \dots s_n \in S^+$ it holds $\Theta(s_0 s_1 \dots s_n)(\alpha) = 1$ for some action α.

A memoryless deterministic (MD) Θ can be viewed as a function $\Theta : S \to Act$.

Example:

The scheduler of the previous example was memoryless and deterministic since the decision what action to take was fixed.

Note:

A scheduler has finite memory if representable by a finite automaton.

Analysis questions

For MC:

- Reachability: $\mathsf{x} = \mathsf{A}\mathsf{x} + \mathsf{b}$ (with $(x(s))_{s \in S_?}$)
- Probabilistic logics: combination of the techniques
- Transient analysis: $\pi_n = \pi_0 \mathsf{P}^n$
- Steady-state analysis: $\pi \mathsf{P} = \pi, \pi \vec{1} = 1$ (ergodic)
- Rewards: reduction to steady-state analysis

Analysis questions

For MC:

- Reachability: $\mathsf{x} = \mathsf{A}\mathsf{x} + \mathsf{b}$ (with $(x(s))_{s \in S_?}$)
- Probabilistic logics: combination of the techniques
- Transient analysis: $\pi_n = \pi_0 \mathsf{P}^n$
- Steady-state analysis: $\pi\mathsf{P} = \pi, \pi\vec{1} = 1$ (ergodic)
- Rewards: reduction to steady-state analysis

For MDP:

- Quantities not defined per se, but depend on the scheduler
- We can naturally consider the best case and the worst case among all schedulers
  (recall that non-determinism can model controllable or uncontrollable choice)

MDP – Reachability

MDP - Reachability

Min

When playing “Mensch Ärgere dich nicht” against a fixed opponent strategy, what is the minimal probability of having all pieces kicked out into the outside area again?

Max

What is the maximal probability of winning the game?

MDP - Reachability

Min

- Best case for reaching undesirable states when controlled
- Worst case for reaching desirable states when not controlled

The minimum probability to reach a set of states B from a state s (within n steps) is

$$\inf_{\Theta} P_s^{\Theta}(\Diamond B), \qquad \inf_{\Theta} P_s^{\Theta}(\Diamond^{\leq n} B)$$

Max

- Best case for reaching desirable states when controlled
- Worst case for reaching undesirable states when not controlled

The maximum probability to reach a set of states B from a state s (within n steps) is

$$\sup_{\Theta} P_s^{\Theta}(\Diamond B), \qquad \sup_{\Theta} P_s^{\Theta}(\Diamond^{\leq n} B)$$

Focus on maximum; minimum is similar

MDP - Reachability

Recall for DTMC

Let (S, P, π_0) be a finite DTMC and $B \subseteq S$. The vector x with $\mathrm{x}(s) = P_s(\Diamond B)$ is the unique solution of the equation system

$$\mathrm{x}(s) = \begin{cases} 1 & \text{if } s \in B, \\ 0 & \text{if } s \in S_0 = \{s \mid P_s(\Diamond B) = 0\}, \\ \sum_{u \in S} \mathrm{P}(s,u) \cdot \mathrm{x}(u) & \text{otherwise.} \end{cases}$$

MDP - Reachability

Recall for DTMC

Let (S, P, π_0) be a finite DTMC and $B \subseteq S$. The vector x with $\mathsf{x}(s) = P_s(\Diamond B)$ is the unique solution of the equation system

$$\mathsf{x}(s) = \begin{cases} 1 & \text{if } s \in B, \\ 0 & \text{if } s \in S_0 = \{s \mid P_s(\Diamond B) = 0\}, \\ \sum_{u \in S} \mathsf{P}(s,u) \cdot \mathsf{x}(u) & \text{otherwise.} \end{cases}$$

Theorem (Maximum Reachability Probability):

Let $(S, Act, \mathsf{P}, \pi_0, L)$ be a finite MDP and $B \subseteq S$. The vector x with $\mathsf{x}(s) = \sup_\Theta P_s^\Theta(\Diamond B)$ is the least solution of the equation system

$$\mathsf{x}(s) = \begin{cases} 1 & \text{if } s \in B, \\ 0 & \text{if } s \in S_0^{max} = \{s \mid \sup_\Theta P_s^\Theta(\Diamond B) = 0\}, \\ \max_{\alpha \in Act(s)} \sum_{u \in S} \mathsf{P}(s,\alpha,u) \cdot \mathsf{x}(u) & \text{otherwise.} \end{cases}$$

MDP - Reachability

Theorem (Optimal Memoryless Scheduler):

Let $\mathcal{M}$ be a finite MDP with state space S, and $B \subseteq S$. There exist memoryless deterministic schedulers $\Theta^{min}, \Theta^{max}$ such that for any $s \in S$ it holds

$$P_s^{\Theta^{min}}(\Diamond B) = \inf_{\Theta} P_s^{\Theta}(\Diamond B), \qquad P_s^{\Theta^{max}}(\Diamond B) = \sup_{\Theta} P_s^{\Theta}(\Diamond B)$$

Proof Sketch

MDP - Reachability

Theorem (Optimal Memoryless Scheduler):

Let $\mathcal{M}$ be a finite MDP with state space S, and $B \subseteq S$. There exist memoryless deterministic schedulers $\Theta^{min}, \Theta^{max}$ such that for any $s \in S$ it holds

$$P_s^{\Theta^{min}}(\Diamond B) = \inf_{\Theta} P_s^{\Theta}(\Diamond B), \qquad P_s^{\Theta^{max}}(\Diamond B) = \sup_{\Theta} P_s^{\Theta}(\Diamond B)$$

Proof Sketch

- For Θ^{min} it suffices to fix in each s an arbitrary action α that minimizes $\sum_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}_u$.

MDP - Reachability

Theorem (Optimal Memoryless Scheduler):

Let $\mathcal{M}$ be a finite MDP with state space S, and $B \subseteq S$. There exist memoryless deterministic schedulers $\Theta^{min}, \Theta^{max}$ such that for any $s \in S$ it holds

$$P_s^{\Theta^{min}}(\Diamond B) = \inf_{\Theta} P_s^{\Theta}(\Diamond B), \qquad P_s^{\Theta^{max}}(\Diamond B) = \sup_{\Theta} P_s^{\Theta}(\Diamond B)$$

Proof Sketch

- For Θ^{min} it suffices to fix in each s an arbitrary action α that minimizes $\sum_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}_u$.
- Does not work for Θ^{max}!

MDP - Reachability

Theorem (Optimal Memoryless Scheduler):

Let $\mathcal{M}$ be a finite MDP with state space S, and $B \subseteq S$. There exist memoryless deterministic schedulers $\Theta^{min}, \Theta^{max}$ such that for any $s \in S$ it holds

$$P_s^{\Theta^{min}}(\Diamond B) = \inf_{\Theta} P_s^{\Theta}(\Diamond B), \qquad P_s^{\Theta^{max}}(\Diamond B) = \sup_{\Theta} P_s^{\Theta}(\Diamond B)$$

Proof Sketch

- For Θ^{min} it suffices to fix in each s an arbitrary action α that minimizes $\sum_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}_u$.
- Does not work for Θ^{max}!
- For Θ^{max} we fix in each s among the actions that maximize $\sum_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}_u$ an arbitrary action α that minimizes the number of steps needed to reach B with positive probability.

MDP - Reachability

Theorem (Optimal Memoryless Scheduler):

Let $\mathcal{M}$ be a finite MDP with state space S, and $B \subseteq S$. There exist memoryless deterministic schedulers $\Theta^{min}, \Theta^{max}$ such that for any $s \in S$ it holds

$$P_s^{\Theta^{min}}(\Diamond B) = \inf_{\Theta} P_s^{\Theta}(\Diamond B), \qquad P_s^{\Theta^{max}}(\Diamond B) = \sup_{\Theta} P_s^{\Theta}(\Diamond B)$$

Proof Sketch

- For Θ^{min} it suffices to fix in each s an arbitrary action α that minimizes $\sum_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}_u$.
- Does not work for Θ^{max}!
- For Θ^{max} we fix in each s among the actions that maximize $\sum_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}_u$ an arbitrary action α that minimizes the number of steps needed to reach B with positive probability.

How can we compute the vectors of values?

- linear programming
- value iteration

MDP - Reachability - Linear Programming

MDP - Reachability - Linear Programming

Linear Program:

Let $(S, Act, \mathsf{P}, \pi_0, L)$ be a finite MDP and $B \subseteq S$. The vector x with $\mathsf{x}(s) = \sup_{\Theta} P_s^{\Theta}(\Diamond B)$ is the unique solution of the linear program

satisfying

$$\begin{aligned} \mathsf{x}(s) &= 1 && \forall s \in B, \\ \mathsf{x}(s) &= 0 && \forall s \in S_0^{\max}, \\ \mathsf{x}(s) &\geq \sum_{u \in S} \mathsf{P}(s, \alpha, u) \cdot \mathsf{x}(u) && \forall s \in S \setminus (B \cup S_0^{\max}), \forall \alpha \in Act. \end{aligned}$$

MDP - Reachability - Linear Programming

Linear Program:

Let $(S, Act, \mathrm{P}, \pi_0, L)$ be a finite MDP and $B \subseteq S$. The vector x with $\mathrm{x}(s) = \sup_\Theta P_s^\Theta(\Diamond B)$ is the unique solution of the linear program

$$\begin{aligned}
&\text{minimize} && \sum_{s \in S} \mathrm{x}(s) && \\
&\text{satisfying} && \mathrm{x}(s) = 1 && \forall s \in B, \\
& && \mathrm{x}(s) = 0 && \forall s \in S_0^{\max}, \\
& && \mathrm{x}(s) \geq \sum_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}(u) && \forall s \in S \setminus (B \cup S_0^{\max}), \forall \alpha \in Act.
\end{aligned}$$

MDP - Reachability - Value Iteration

Value Iteration Algorithm:

Let $\mathcal{M}$ be a finite MDP with state space S, and $B \subseteq S$.

- Initialize $x_0(s)$ to 1 if $s \in B$ and to 0, otherwise.
- Iterate

$$x_{n+1}(s) = \begin{cases} 1 & \text{if } s \in B, \\ 0 & \text{if } s \in S_0^{\max}, \\ \max\limits_{\alpha \in Act(s)} \sum\limits_{u \in S} P(s, \alpha, u) \cdot x_n(u) & \text{otherwise} \end{cases}$$

until convergence, i.e., until $\max_{s \in S} |x_{n+1}(s) - x_n(s)| < \epsilon$ for a small $\epsilon > 0$

MDP - Reachability - Value Iteration

Value Iteration Algorithm:

Let $\mathcal{M}$ be a finite MDP with state space S, and $B \subseteq S$.

- Initialize $\mathrm{x}_0(s)$ to 1 if $s \in B$ and to 0, otherwise.
- Iterate

$$\mathrm{x}_{n+1}(s) = \begin{cases} 1 & \text{if } s \in B, \\ 0 & \text{if } s \in S_0^{\max}, \\ \max\limits_{\alpha \in Act(s)} \sum\limits_{u \in S} \mathrm{P}(s, \alpha, u) \cdot \mathrm{x}_n(u) & \text{otherwise} \end{cases}$$

until convergence, i.e., until $\max_{s \in S} |\mathrm{x}_{n+1}(s) - \mathrm{x}_n(s)| < \epsilon$ for a small $\epsilon > 0$

Theorem

- $\mathrm{x}_n(s) = \sup\limits_{\Theta} P_s^{\Theta}(\Diamond^{\leq n} B)$.
- $\lim\limits_{n \to \infty} \mathrm{x}_n(s) = \sup\limits_{\Theta} P_s^{\Theta}(\Diamond B)$.

MDP - Step-Bounded Reachability

Is a memoryless deterministic scheduler enough for optimizing $\Diamond^{\leq n} B$?

MDP - Step-Bounded Reachability

Is a memoryless deterministic scheduler enough for optimizing $\Diamond^{\leq n} B$?

No! For step-bounded reachability we might need finite memory. (Intuition: Depending on the current step, different paths of different length might be optimal).

MDP - Reachability - Computing $S_0^{\max}$

We rather compute the set

$$S_{>0}^{\max} = \{s \mid \sup_{\Theta} P_s^{\Theta}(\Diamond B) > 0\}$$

and return

$$S_0^{\max} = S \setminus S_{>0}^{\max}$$

MDP - Reachability - Computing $S_0^{\max}$

We rather compute the set

$$S_{>0}^{\max} = \{s \mid \sup_{\Theta} P_s^{\Theta}(\Diamond B) > 0\}$$

and return

$$S_0^{\max} = S \setminus S_{>0}^{\max}$$

$S_{>0}^{\max}$:

Initialize the set to B and in every iteration add states that reach the set in one step with positive probability for some enabled action. Repeat until fix-point is reached.

MDP - Reachability - Computing S_0^{max}

We rather compute the set

$$S_{>0}^{max} = \{s \mid \sup_{\Theta} P_s^{\Theta}(\Diamond B) > 0\}$$

and return

$$S_0^{max} = S \setminus S_{>0}^{max}$$

$S_{>0}^{max}$:
Initialize the set to B and in every iteration add states that reach the set in one step with positive probability for some enabled action. Repeat until fix-point is reached.

(Similarly for $S_{>0}^{min}$:

MDP - Reachability - Computing $S_0^{\max}$

We rather compute the set

$$S_{>0}^{\max} = \{s \mid \sup_{\Theta} P_s^{\Theta}(\Diamond B) > 0\}$$

and return

$$S_0^{\max} = S \setminus S_{>0}^{\max}$$

$S_{>0}^{\max}$:
Initialize the set to B and in every iteration add states that reach the set in one step with positive probability for some enabled action. Repeat until fix-point is reached.

(Similarly for $S_{>0}^{\min}$: replace “some” by ”every”)

Analysis questions

- Reachability: LP or VI
- Probabilistic logics: combination of the techniques (in particular reachability and bounded reachability)
- Transient analysis
- Steady-state analysis
- Rewards

MDP – PCTL & LTL

Recall: MDP non-determinism

We consider two different sources of non-determinism:

Controllable If we can control the choice of actions: Is there possibly a scheduler guaranteeing the specified desirable behavior?

Uncontrollable If we cannot control the choice of actions: Do all schedulers necessarily guarantee the specified desirable behavior?

Note: If we have undesirable behaviour specified, we can apply negation to obtain the desirable behaviour.

MDP + Logics

pLTL

Example: the probability that eventually red player is kicked out and then immediately kicks out blue player is possibly / necessarily ≥ 0.8

$$\exists\Theta \;/\; \forall\Theta : \quad P^{\Theta}(\mathcal{F}\ (rkicked \wedge \mathcal{X}\ bkicked)) \geq 0.8$$

PCTL

Example: with prob. necessarily ≥ 0.5 the probability to return to initial state is always necessarily ≥ 0.1: $P_{\geq 0.5}\ \mathcal{G}\ \ P_{\geq 0.1}\ \mathcal{F}\ init$

PCTL Semantics

Recall: DTMC

For a state s:

- $s \models true$ (always),
- $s \models a$ iff $a \in L(s)$,
- $s \models \phi_1 \wedge \phi_2$ iff $s \models \phi_1$ and $s \models \phi_2$,
- $s \models \neg\phi$ iff $s \not\models \phi$,
- $s \models \mathcal{P}_J(\psi)$ iff $P_s(Paths(\psi)) \in J$

MDP

Stays the same except for $\mathcal{P}_J$ defined in one of the following ways:

- Possibility (controllable): $s \models \mathcal{P}_J(\psi)$ iff $\exists\Theta : P_s^{\Theta}(Paths(\psi)) \in J$;
- Necessity (uncontrollable): $s \models \mathcal{P}_J(\psi)$ iff $\forall\Theta : P_s^{\Theta}(Paths(\psi)) \in J$.

Note

PCTL path formulae semantics stays the same.

PCTL Verification (1) – Algorithm

Algorithm

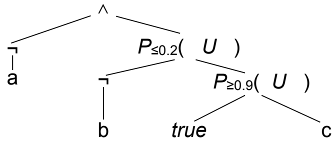
Input: MDP $\mathcal{M}$, state s, PCTL state formula Φ

Output: TRUE iff $s \models \Phi$.

The algorithm is conceptually the same as for DTMC:
Again, consider the bottom-up traversal of the parse tree of Φ:

- The leaves are $a \in AP$ or $true$ and
- the inner nodes are:
  - unary – labelled with the operator $\neg$ or $\mathcal{P}_J(\mathcal{X}\ )$;
  - binary – labelled with an operator $\wedge$, $\mathcal{P}_J(\ \mathcal{U}\ )$, or $\mathcal{P}_J(\ \mathcal{U}^{\leq n})$.

Example: $\neg a \wedge \mathcal{P}_{\leq 0.2}(\neg b\ \ \mathcal{U}\ \ \mathcal{P}_{\geq 0.9}(\Diamond\ c))$



Compute $Sat(\Psi) = \{s \in S \mid s \models \Psi\}$ for each node Ψ of the tree in a bottom-up fashion. Then $s \models \Phi$ iff $s \in Sat(\Phi)$.

PCTL Verification (2) – Algorithm

As before:

- $Sat(true) = S$,
- $Sat(a) = \{s \mid a \in L(s)\}$
- $Sat(\Phi_1 \wedge \Phi_2) = Sat(\Phi_1) \cap Sat(\Phi_2)$
- $Sat(\neg\Phi) = S \setminus Sat(\Phi)$

Path operator for "possibly"

We need to restrict to path operators of the form $\mathcal{P}_{\bowtie p}$ with $p \in [0, 1]$ and $\bowtie \in \{\leq, <, >, \geq\}$. We have

- for $\bowtie \in \{\leq, <\}$:
  $Sat(\mathcal{P}_{\bowtie p}(\Psi)) = \{s \in S \mid \min_\Theta P_s^\Theta(Paths(\Psi)) \bowtie p\}$
- for $\bowtie \in \{\geq, >\}$:
  $Sat(\mathcal{P}_{\bowtie p}(\Psi)) = \{s \in S \mid \max_\Theta P_s^\Theta(Paths(\Psi)) \bowtie p\}$

PCTL Verification (2) – Algorithm

As before:

- $Sat(true) = S$,
- $Sat(a) = \{s \mid a \in L(s)\}$
- $Sat(\Phi_1 \wedge \Phi_2) = Sat(\Phi_1) \cap Sat(\Phi_2)$
- $Sat(\neg\Phi) = S \setminus Sat(\Phi)$

Path operator for "possibly"

We need to restrict to path operators of the form $\mathcal{P}_{\bowtie p}$ with $p \in [0, 1]$ and $\bowtie \in \{\leq, <, >, \geq\}$. We have

- for $\bowtie \in \{\leq, <\}$:
  $Sat(\mathcal{P}_{\bowtie p}(\Psi)) = \{s \in S \mid \min_{\Theta} P_s^{\Theta}(Paths(\Psi)) \bowtie p\}$
- for $\bowtie \in \{\geq, >\}$:
  $Sat(\mathcal{P}_{\bowtie p}(\Psi)) = \{s \in S \mid \max_{\Theta} P_s^{\Theta}(Paths(\Psi)) \bowtie p\}$

"Necessarily"

can be done similarly by swapping max and min.

PCTL Verification – Algorithm (3)

Similar as before:

- Next:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\mathcal{X}\ \Phi)\Big) =$$

- Bounded Until:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\Phi_1\ \ \mathcal{U}^{\leq n}\ \Phi_2)\Big) =$$

- Unbounded Until:

$$\max_{\Theta} P_s\Big(Paths(\Phi_1\ \ \mathcal{U}\ \ \Phi_2)\Big) =$$

PCTL Verification – Algorithm (3)

Similar as before:

- Next:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\mathcal{X}\ \Phi)\Big) = \max_{\alpha \in Act(s)} \sum_{s' \in Sat(\Phi)} \mathrm{P}(s, s')$$

- Bounded Until:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\Phi_1\ \ \mathcal{U}^{\ \leq n}\ \Phi_2)\Big) = \max_{\Theta} P_s^{\Theta}\Big(Sat(\Phi_1)\ \ \mathcal{U}^{\ \leq n}\ Sat(\Phi_2)\Big)$$

- Unbounded Until:

$$\max_{\Theta} P_s\Big(Paths(\Phi_1\ \ \mathcal{U}\ \ \Phi_2)\Big) = \max_{\Theta} P_s\Big(Sat(\Phi_1)\ \ \mathcal{U}\ \ Sat(\Phi_2)\Big)$$

PCTL Verification – Algorithm (3)

Similar as before:

- Next:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\mathcal{X}\ \Phi)\Big) = \max_{\alpha \in Act(s)} \sum_{s' \in Sat(\Phi)} \mathrm{P}(s, s')$$

- Bounded Until:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\Phi_1\ \ \mathcal{U}^{\le n}\ \Phi_2)\Big) = \max_{\Theta} P_s^{\Theta}\Big(Sat(\Phi_1)\ \ \mathcal{U}^{\le n}\ Sat(\Phi_2)\Big)$$

- Unbounded Until:

$$\max_{\Theta} P_s\Big(Paths(\Phi_1\ \ \mathcal{U}\ \ \Phi_2)\Big) = \max_{\Theta} P_s\Big(Sat(\Phi_1)\ \ \mathcal{U}\ \ Sat(\Phi_2)\Big)$$

- similarly for $\min_{\Theta}$

PCTL Verification – Algorithm (3)

Similar as before:

- Next:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\mathcal{X}\ \Phi)\Big) = \max_{\alpha \in Act(s)} \sum_{s' \in Sat(\Phi)} \mathrm{P}(s, s')$$

- Bounded Until:

$$\max_{\Theta} P_s^{\Theta}\Big(Paths(\Phi_1\ \ \mathcal{U}^{\leq n}\ \Phi_2)\Big) = \max_{\Theta} P_s^{\Theta}\Big(Sat(\Phi_1)\ \ \mathcal{U}^{\leq n}\ Sat(\Phi_2)\Big)$$

- Unbounded Until:

$$\max_{\Theta} P_s\Big(Paths(\Phi_1\ \ \mathcal{U}\ \ \Phi_2)\Big) = \max_{\Theta} P_s\Big(Sat(\Phi_1)\ \ \mathcal{U}\ \ Sat(\Phi_2)\Big)$$

- similarly for $\min_{\Theta}$

As before:

can be reduced to step-bounded/unbounded max/min reachability.

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0,1]$
Output: TRUE iff $\exists\,\Theta : P_s^\Theta(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^\Theta(Paths(\Psi)) \leq p \iff$

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0,1]$
Output: TRUE iff $\exists\, \Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists \Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists \Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0, 1]$
Output: TRUE iff $\exists\,\Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \rightarrow \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff$.

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0,1]$
Output: TRUE iff $\exists\,\Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \rightarrow \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff \neg\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p$.

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0, 1]$
Output: TRUE iff $\exists\, \Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \to \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff \neg\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p$.

Algorithm

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0, 1]$
Output: TRUE iff $\exists\, \Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \rightarrow \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff \neg\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p$.

Algorithm

The algorithm is conceptually the same as for DTMC:

1. transform Ψ to a deterministic Rabin automaton R with $\mathrm{Lang}(R) = Paths(\Psi)$,

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0, 1]$
Output: TRUE iff $\exists\,\Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \to \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff \neg\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p$.

Algorithm

The algorithm is conceptually the same as for DTMC:

1. transform Ψ to a deterministic Rabin automaton R with $\mathrm{Lang}(R) = Paths(\Psi)$,
2. construct product MDP $\mathcal{M} \times R$,

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0, 1]$
Output: TRUE iff $\exists\, \Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \to \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff \neg\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p$.

Algorithm

The algorithm is conceptually the same as for DTMC:

1. transform Ψ to a deterministic Rabin automaton R with $\mathrm{Lang}(R) = Paths(\Psi)$,
2. construct product MDP $\mathcal{M} \times R$,
3. by graph algorithms, find in the product MDP all accepting end components,

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0, 1]$
Output: TRUE iff $\exists \Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \rightarrow \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff \neg\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p$.

Algorithm

The algorithm is conceptually the same as for DTMC:

1. transform Ψ to a deterministic Rabin automaton R with $\mathrm{Lang}(R) = Paths(\Psi)$,
2. construct product MDP $\mathcal{M} \times R$,
3. by graph algorithms, find in the product MDP all accepting end components,
4. their union is denoted by X
5. return TRUE iff $\max_{\Theta} P_s^{\Theta}(\Diamond X) \geq p$.

LTL Verification

Input: MDP $\mathcal{M}$, state s, LTL formula Ψ, threshold $p \in [0,1]$
Output: TRUE iff $\exists\,\Theta : P_s^{\Theta}(Paths(\Psi)) \geq p$.

Reducing subcases

We can reduce $\leq$ to $\geq$ by:
$\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p \iff \exists\Theta : P_s^{\Theta}(Paths(\neg\Psi)) \geq 1 - p$
and necessarily to possibly ($\forall \to \exists$) by:
$\forall\Theta : P_s^{\Theta}(Paths(\Psi)) > p \iff \neg\exists\Theta : P_s^{\Theta}(Paths(\Psi)) \leq p$.

Algorithm

The algorithm is conceptually the same as for DTMC:

1. transform Ψ to a deterministic Rabin automaton R with $\mathrm{Lang}(R) = Paths(\Psi)$,
2. construct product MDP $\mathcal{M} \times R$,
3. by graph algorithms, find in the product MDP all accepting end components, $\longleftarrow$ How to do this?!?
4. their union is denoted by X
5. return TRUE iff $\max_{\Theta} P_s^{\Theta}(\Diamond X) \geq p$.

End Components

- An end component is a subset of states S' and actions A' such that
  - $\sum_{s' \in S'} P(s, \alpha, s') = 1$ for each $s \in S'$ and $\alpha \in A'(s')$ and
  - it is strongly connected (when considering edges of all actions).
- With probability 1, infinitely often visited states on a run form an end component.

End Components

- An end component is a subset of states S' and actions A' such that
  - $\sum_{s' \in S'} P(s, \alpha, s') = 1$ for each $s \in S'$ and $\alpha \in A'(s')$ and
  - it is strongly connected (when considering edges of all actions).
- With probability 1, infinitely often visited states on a run form an end component.
- It is accepting if for some Rabin pair (E_i, F_i) it contains no state of E_i and some state of F_i.
- But: there are exponentially many end components.

End Components

- An end component is a subset of states S' and actions A' such that
  - $\sum_{s' \in S'} P(s, \alpha, s') = 1$ for each $s \in S'$ and $\alpha \in A'(s')$ and
  - it is strongly connected (when considering edges of all actions).
- With probability 1, infinitely often visited states on a run form an end component.
- It is accepting if for some Rabin pair (E_i, F_i) it contains no state of E_i and some state of F_i.
- But: there are exponentially many end components.

The solution: Maximal end components

- Maximal exist as union of two non-disjoint end components is an end component.
- Thus, we can deal with partition, instead.

MDP: Maximal End Components – Algorithm

A partition-refinement algorithm

Start with partition $\{S\}$. In each iteration for each partition class T.

MDP: Maximal End Components – Algorithm

A partition-refinement algorithm

Start with partition $\{S\}$. In each iteration for each partition class T.

1. Find in the induced subgraph of T (when considering edges of all actions) all SCCs that have at least one edge.
2. Repetitively:
   (a) Remove all actions that leave with positive probability its SCC.
   (b) Remove from each SCC all states that have no actions.
3. Replace T by what is left of each SCC.
4. Newly added classes may be not strongly-connected, repeat.

Accepting Maximal End Components

Accepting MEC for Rabin condition $(E_i, F_i)_{i \in I}$

- For each $i \in I$, run the algorithm with initial "partitioning" $S \setminus E_i$

  i.e. construct an MDP M_i by removing states E_i and repetitively removing
  (a) actions that lead with positive probability to some removed state and
  (b) states with no actions,

  then run the algorithm
- Accepting MEC in each M_i are those containing some state of F_i.

Analysis questions

- Reachability: LP or VI
- Probabilistic logics: combination of the techniques
- Transient analysis: preference over S needed
- Steady-state analysis: preference over S needed
- Rewards: solves transient and steady-state analysis

For best/worst transient/steady-state distribution, a preference over S needed

- Step bounded reachability $\Diamond^{\leq n} B$ is one approach to distribution after n steps (preferred are exactly the states in B).
- A more fine tuned preference can be specified by rewards

MDP – Rewards

- expected instantaneous reward
- expected mean payoff

MDP – Rewards

Instantaneous rewards

What is the maximal expected number of my pieces in the play area after 50 rounds?

Step-bounded cumulative rewards

What is the maximal expected number of times I kick out a piece of the opponent within the first 100 steps?

Cumulative rewards to reach a target

What is the minimal expected number of steps before the game ends?

Mean payoff (long-run average reward)

What is the average number of pieces on board?

(restart after game end $\Rightarrow$ infinite run)

MDP - Rewards - Instantaneous

Definition

$\sup_{\Theta} \mathsf{E}^{\Theta}[I_r^{=k}]$ where $I_r^{=k}(\xi(s_0 s_1 \ldots)) = r(s_k)$

MDP - Rewards - Instantaneous

Definition

$\sup_\Theta \mathrm{E}^\Theta[I_r^{=k}]$ where $I_r^{=k}(\xi(s_0 s_1 \ldots)) = r(s_k)$

Theorem

For an MDP with reward r, the vector $\mathrm{x}(s) = \sup_\Theta \mathrm{E}_s^\Theta[I_r^{=k}]$ equals to $\mathrm{x}^k(s)$ where

$$\mathrm{x}^\ell(s) = \begin{cases} r(s) & \text{if } \ell = 0 \\ \max\limits_{\alpha \in Act(s)} \sum\limits_{s' \in S} P(s, \alpha, s') \cdot \mathrm{x}^{\ell-1}(s') & \text{otherwise} \end{cases}$$

MDP - Rewards - Instantaneous

Definition

$\sup_\Theta \mathsf{E}^\Theta[I_r^{=k}]$ where $I_r^{=k}(\xi(s_0 s_1 \ldots)) = r(s_k)$

Theorem

For an MDP with reward r, the vector $\mathrm{x}(s) = \sup_\Theta \mathsf{E}_s^\Theta[I_r^{=k}]$ equals to $\mathrm{x}^k(s)$ where

$$\mathrm{x}^\ell(s) = \begin{cases} r(s) & \text{if } \ell = 0 \\ \max_{\alpha \in Act(s)} \sum_{s' \in S} P(s, \alpha, s') \cdot \mathrm{x}^{\ell-1}(s') & \text{otherwise} \end{cases}$$

Corollary

There are optimal deterministic schedulers for max $\mathsf{E}_s^\Theta[I_r^{=k}]$ (and similarly min).

MDP - Rewards - Instantaneous

Definition

$\sup_\Theta \mathsf{E}^\Theta[I_r^{=k}]$ where $I_r^{=k}(\xi(s_0 s_1 \ldots)) = r(s_k)$

Theorem

For an MDP with reward r, the vector $\mathrm{x}(s) = \sup_\Theta \mathsf{E}_s^\Theta[I_r^{=k}]$ equals to $\mathrm{x}^k(s)$ where

$$\mathrm{x}^\ell(s) = \begin{cases} r(s) & \text{if } \ell = 0 \\ \max\limits_{\alpha \in Act(s)} \sum\limits_{s' \in S} P(s, \alpha, s') \cdot \mathrm{x}^{\ell-1}(s') & \text{otherwise} \end{cases}$$

Corollary

There are optimal deterministic schedulers for max $\mathsf{E}_s^\Theta[I_r^{=k}]$ (and similarly min).

What about step-bounded **cumulative** reward?

$$\mathrm{x}^\ell(s) = \begin{cases} 0 & \text{if } \ell = 0 \\ r(s) + \max\limits_{\alpha \in Act(s)} \sum\limits_{s' \in S} P(s, \alpha, s') \cdot \mathrm{x}^{\ell-1}(s') & \text{otherwise} \end{cases}$$

MDP - Rewards - Mean Payoff

Recall mean payoff (long-run average reward):

$$R_1 R_2 \cdots = 42\ 1\ 2\ 1\ 2\ 1\ 2 \cdots \qquad \lim_{n\to\infty} \frac{\sum_{i=1}^{n} R_i}{n} = 1.5$$

Example: Money investment

- > 0 earning, < 0 losing
- maximize expected mean payoff

MDP - Rewards - Mean Payoff

Recall mean payoff (long-run average reward):

$$R_1 R_2 \cdots = 42\ 1\ 2\ 1\ 2\ 1\ 2 \cdots \qquad \lim_{n\to\infty} \frac{\sum_{i=1}^n R_i}{n} = 1.5$$

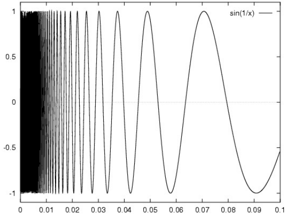
Example: Money investment

- > 0 earning, < 0 losing
- maximize expected mean payoff

Limit may not exist:

$$0\ (1)^{10}\ (0)^{1000}\ (1)^{1000000} \cdots$$

$$\liminf_{n\to\infty} \frac{\sum_{i=1}^n R_i}{n} = 0$$



Definition

$\sup_{\Theta} \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}^{\Theta}[r(A_i)]$ where A_i is (random variable for) ith action

MDP – Mean payoff – Value iteration

For ergodic systems, extensible to general but more complicated

Value vector $\vec{v}$ found by successive approximation
$\vec{w}^t$ is the optimal total reward in time t

1. Choose $\varepsilon > 0$, and take $\vec{w}^0 := \vec{0} \in \mathbb{R}^{|S|}$
2. Compute iteration:
   - $$\vec{w}_s^{t+1} := \max_{a \in Act(s)} r(a) + \sum_{s' \in S} \delta(a)(s')\vec{w}_{s'}, \text{ for } s \in S \tag{1}$$

MDP – Mean payoff – Value iteration

For ergodic systems, extensible to general but more complicated

Value vector $\vec{v}$ found by successive approximation
$\vec{w}^t$ is the optimal total reward in time t

1. Choose $\varepsilon > 0$, and take $\vec{w}^0 := \vec{0} \in \mathbb{R}^{|S|}$
2. Compute iteration:
   - $$\vec{w}_s^{t+1} := \max_{a \in Act(s)} r(a) + \sum_{s' \in S} \delta(a)(s')\vec{w}_{s'}, \text{ for } s \in S \qquad (1)$$
3. Compute error
   - $upper := \max_{s \in S}(\vec{w}_s^{t+1} - \vec{w}_s^t)$
   - $lower := \min_{s \in S}(\vec{w}_s^{t+1} - \vec{w}_s^t)$

   If $upper - lower > \varepsilon$: go to step 2. with t:=t+1;
   else $\frac{upper+lower}{2}$ is a $\frac{\varepsilon}{2}$-approximation of the value $\vec{v}$ (Stop)
   Optimal strategy: pick maximum in (1)

upper and *lower* approximate $\vec{v}$ from above and below, respectively
$\vec{w}^{t+1} - \vec{w}^t$, converges to $\vec{v}$

MDP – Mean payoff – Strategy iteration I

Sequence $f^0, f^1, \ldots$ of strategies such that $\vec{v}(f^{t+1}) \geq \vec{v}(f^t)$ and converging to an optimal strategy

Finitely many strategies $\Rightarrow$ termination

$$\begin{aligned}
\text{for all } s \in S: \quad & \vec{x}_s && = \textstyle\sum_{s' \in S} \delta(f(s), s')\vec{x}_{s'} \\
\text{for all } s \in S: \quad & \vec{x}_s + \vec{y}_s && = \textstyle\sum_{s' \in S} \delta(f(s), s')\vec{y}_{s'} + r(f(s)) \\
\text{for all } s \in S: \quad & \vec{y}_s + \vec{z}_s && = \textstyle\sum_{s' \in S} \delta(f(s), s')\vec{z}_{s'}
\end{aligned} \tag{2}$$

$\vec{x}$ is equal to $\mathbb{E}^f[MP]$
$\vec{y}$ is the difference between total and long-run rewards
$\vec{z}$ is used in the algorithm to prevent cycling

MDP – Mean payoff – Strategy iteration II

Using $(\vec{x}, \vec{y})$

$$B(s,f) = \left\{ a \in Act(s) \;\middle|\; \begin{array}{l} \sum_{s'} \delta(a)(s')\vec{x}_{s'} > \vec{x}_s \text{ or} \\ \sum_{s'} \delta(a)(s')\vec{x}_{s'} = \vec{x}_s \text{ and} \\ r(a) + \sum_{s'} \delta(a)(s')\vec{y}_{s'} > \vec{x}_s + \vec{y}_s \end{array} \right\} \quad (3)$$

1. Start with any $f \in F$.
2. Determine unique $(\vec{x},\vec{y})$-part in a solution of the linear system (2)
3. For every $s \in S$: determine $B(s,f)$ as defined in (3) using the values $\vec{x}$ and $\vec{y}$ from step 2
4. If $B(s,f) = \emptyset$ for every $s \in S$: go to step 6
   Otherwise: take any $g \neq f$ such that $g(s) \in B(s,f)$ if $g(s) \neq f(s)$
5. $f := g$ and go to step 2
6. f is an average optimal strategy

MDP – Mean payoff – Linear programming I

$\vec{v}$ the smallest solution of LP, strategy derived from its dual LP

Primary linear program:

$$\text{Minimize:} \quad \sum_{s \in S} \vec{\mu}_s \vec{x}_s$$

$$\text{Subject to:} \tag{4}$$

$$\text{for all } s \in S, a \in Act(s)\text{: } \vec{x}_s \geq \sum_{s' \in S} \delta(a)(s')\vec{x}_{s'}$$

$$\text{for all } s \in S, a \in Act(s)\text{: } \vec{x}_s \geq r(a) + \sum_{s' \in S} \delta(a)(s')\vec{y}_{s'} - \vec{y}_s$$

where $\vec{\mu}_s > 0$ arbitrarily chosen

MDP – Mean payoff – Linear programming II

Dual linear program:

Maximize:

$$\sum_{a \in A} r(a)\vec{x}_a$$

Subject to:

for all $s \in S$: (5)

$$\vec{\mu}_s + \sum_{a \in A} \delta(a)(s)\vec{y}_a = \sum_{a \in Act(s)} \vec{y}_a + \sum_{a \in Act(s)} \vec{x}_a$$

for all $s \in S$:

$$\sum_{a \in A} \delta(a)(s)\vec{x}_a = \sum_{a \in Act(s)} \vec{x}_a$$

$\vec{x}$: occupation measure in the limit
$\vec{y}_a$: expected number of taking action a during the transient phase

both flows subject to Kirchhof's law

MDP – Mean payoff – Linear programming III

Optimal strategy: f such that

- $\vec{x}_{f(s)} > 0$ if $s \in S_{\vec{x}}$
- $\vec{y}_{f(s)} > 0$ if $s \notin S_{\vec{x}}$

where $S_{\vec{x}} := \{s \in S \mid \sum_{a \in Act(s)} \vec{x}_a > 0\}$

Multiple mean payoff

Optimize multiple mean payoffs MP_i, $i \in \{1, \ldots, n\}$, in MDP:

- expectation

$$\bigwedge_i \mathbb{E}[MP_i] \geq \mathsf{exp}_i$$

- satisfaction (quantiles, percentiles)
  - conjunctive

$$\bigwedge_i \mathbb{P}[MP_i \geq \mathsf{sat}_i] \geq \mathsf{prob}_i$$

  - joint

$$\mathbb{P}[\bigwedge_i MP_i \geq \mathsf{sat}_i] \geq prob$$

- **conjunctions** thereof [CKK15,CR15]

Examples

Example 1: Money investment

- > 0 earning, < 0 losing
- maximize expected mean payoff $\mathbb{E}[MP]$

Examples

Example 1: Money investment

- > 0 earning, < 0 losing
- maximize expected mean payoff $\mathbb{E}[MP]$
- maximize probability $\mathbb{P}[MP \geq 0]$

Examples

Example 1: Money investment

- > 0 earning, < 0 losing
- maximize expected mean payoff $\mathbb{E}[MP]$
- maximize probability $\mathbb{P}[MP \geq 0]$
- maximize $\mathbb{E}[MP]$ while ensuring $\mathbb{P}[MP \geq 0] \geq 0.95$

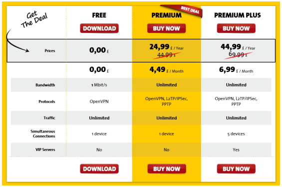
  "risk-averse" strategies

Examples

Example 1: Money investment

- > 0 earning, < 0 losing
- maximize expected mean payoff $\mathbb{E}[MP]$
- maximize probability $\mathbb{P}[MP \geq 0]$
- maximize $\mathbb{E}[MP]$ while ensuring $\mathbb{P}[MP \geq 0] \geq 0.95$

  "risk-averse" strategies

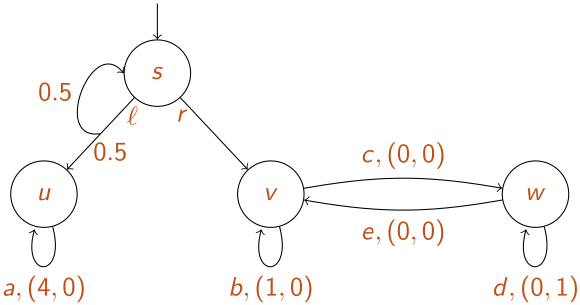


Get The Deal	FREE	PREMIUM (BEST DEAL)	PREMIUM PLUS
	DOWNLOAD	BUY NOW	BUY NOW
Prices	0,00 £	24,99 £ / Year ~~44,99 £~~	44,99 £ / Year ~~69,99 £~~
	0,00 £	4,49 £ / Month	6,99 £ / Month
Bandwidth	1 Mbit/s	Unlimited	Unlimited
Protocols	OpenVPN	OpenVPN, L2TP/IPSec, PPTP	OpenVPN, L2TP/IPSec, PPTP
Traffic	Unlimited	Unlimited	Unlimited
Simultaneous Connections	1 device	1 device	5 devices
VIP Servers	No	No	Yes
	DOWNLOAD	BUY NOW	BUY NOW

Example 2: Downloading service (multiple mean payoffs)

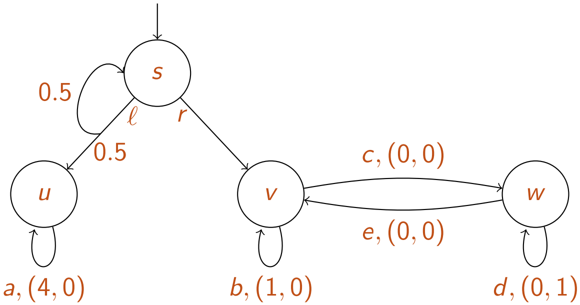
- gratis service: expected throughput $MP_1 \geq 1Mbps$
- premium service: $\mathbb{E}[MP_2] \geq 10Mbps$ and $\geq 95\%$ connections run on $\geq 5Mbps$; sold at p_2 per Mb
- need to hire MP_3 resources from a cloud each at price p_3
- while satisfying the guarantees, maximize $\mathbb{E}[p_2 \cdot MP_2 - p_3 \cdot MP_3]$

Example



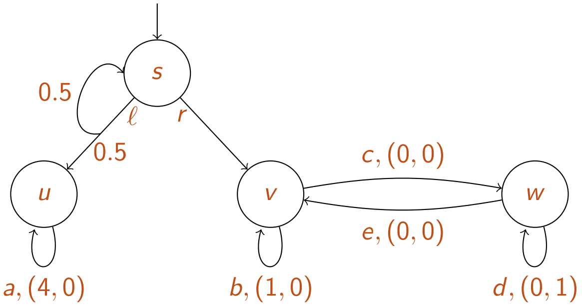
$\text{sat} = (0.5, 0.5), \text{prob} = (0.8, 0.8)$

Example



$\text{exp} = (1.1, 0.5)$, $\text{sat} = (0.5, 0.5)$, $\text{prob} = (0.8, 0.8)$

Example



$\text{exp} = (1.1, 0.5)$, $\text{sat} = (0.5, 0.5), \text{prob} = (0.8, 0.8)$

- linear programming
- feasible and practically useful

Model Construction Principles

Model Construction - Parallelism and Communication

The setting

- "Real" parallel system: $P = P_1 \parallel \ldots \parallel P_n$.

Model Construction – Parallelism and Communication

The setting

- "Real" parallel system: $P = P_1 \parallel \ldots \parallel P_n$.
- Transition system: $T = T_1 \parallel \ldots \parallel T_n$.

Model Construction - Parallelism and Communication

The setting

- "Real" parallel system: $P = P_1 \parallel \ldots \parallel P_n$.
- Transition system: $T = T_1 \parallel \ldots \parallel T_n$.

Our goal: Define semantic parallel operators on transition systems to model "real" parallel operators.

Model Construction - Parallelism and Communication

The setting

- "Real" parallel system: $P = P_1 \parallel \ldots \parallel P_n$.
- Transition system: $T = T_1 \parallel \ldots \parallel T_n$.

Our goal: Define semantic parallel operators on transition systems to model "real" parallel operators.

In the following we:

1. recall the notions without randomness
2. observe how to add the randomness

Model Construction - (non-random) Transition System

A transition system in a tuple

$$\mathcal{T} = (S, Act, \rightarrow, s_0, \mathbf{AP}, L)$$

- S is the state space, i.e., set of states,
- Act is a set of actions,
- $\rightarrow \subseteq S \times Act \times S$ is the transition relation of the form $s \xrightarrow{\alpha} s'$ where $s, s' \in S$ and $\alpha \in Act$.
- $s_0 \in S$ is the initial state,
- $\mathbf{AP}$ is a set of atomic propositions,
- $L : S \rightarrow 2^{\mathbf{AP}}$ is the labelling function.

Model Construction - Operators for parallelism (1)

1. Pure concurrency: Interleaving operator, no communication, no dependencies
2. Synchronous product: For hardware systems with a shared clock
3. Synchronous message passing
4. Communication via shared variables
5. Channel systems: Shared variables + communication via channels

Model Construction - 1. Interleaving Operator $|||$

$$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, s_{01}, \mathsf{AP}_1, L_1)$$

$$\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, s_{02}, \mathsf{AP}_2, L_2)$$

The composite transition system $\mathcal{T}_1 ||| \mathcal{T}_2$ is given by:

$$\mathcal{T}_1 ||| \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \langle s_{01}, s_{02} \rangle, \mathsf{AP}, L)$$

where $\rightarrow$ is given by:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

atomic propositions: $\mathsf{AP} = \mathsf{AP}_1 \uplus \mathsf{AP}_2$
labelling function: $L(\langle s_1, s_2 \rangle) = L(s_1) \cup L(s_2)$

Model Construction - 2. Synchronous Product $\otimes$

$$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, s_{01}, \mathsf{AP}_1, L_1)$$

$$\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, s_{02}, \mathsf{AP}_2, L_2)$$

The composite transition system $\mathcal{T}_1 \otimes \mathcal{T}_2$ is given by:

$$\mathcal{T}_1 \otimes \mathcal{T}_2 = (S_1 \times S_2, Act, \rightarrow, \langle s_{01}, s_{02} \rangle, \mathsf{AP}, L)$$

where $\rightarrow$ is given by:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \wedge s_2 \xrightarrow{\beta}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha * \beta} \langle s_1', s_2' \rangle}$$

$* : Act_1 \times Act_2 \rightarrow Act$

Model Construction – 3. Synch. Message Passing $\|_{Syn}$

$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, s_{01}, \mathsf{AP}_1, L_1)$
$\mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, s_{02}, \mathsf{AP}_2, L_2)$
Concurrent execution with synchronization over all actions in
$Syn \subseteq Act_1 \cap Act_2$:

$$\mathcal{T}_1 \|_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \langle s_{01}, s_{02} \rangle, \mathsf{AP}, L)$$

- Interleaving for $\alpha \notin Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

- Handshaking for $\alpha \in Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \wedge s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle}$$

Model Construction - Operators for parallelism (2)

1. Pure concurrency: Interleaving operator, no communication, no dependencies
2. Synchronous product: For hardware systems with a shared clock
3. Synchronous message passing: Interleaving + synchronization
4. Communication via shared variables
   - Encode possible variable values as states
   - Transition system describes possible updates and lookups
   - Resort to synchronous message passing
5. Channel systems: Shared variables + communication via channels
   - communication over shared variables
   - synchronous message passing (channels of capacity 0)
   - asynchronous message passing (channels of capacity ≥ 1)

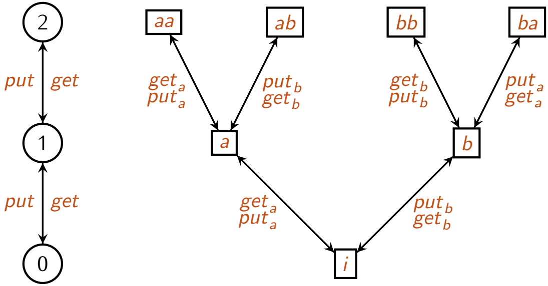
   can be encoded into
   - transition systems using only
   - synchronous message passing

Model Construction – 4. Shared Variables

- Given n different processes $i = 1, \ldots, n$
- To model variable x with values $V = \{v_1, \ldots, v_m\}$
- Introduce another process and new actions
- $\mathcal{T}_x = (S_x, Act_x, \rightarrow_x, \ldots)$
  - $S_x = \{v_1, \ldots, v_m\}$
  - $Act_x = \{get_{x,i,v}, set_{x,i,v} \mid i \in \{1, \ldots, n\}, v \in V\}$
  - $\rightarrow_x = \{(v, get_{x,i,v}, v), (v, set_{x,i,v'}, v') \mid i \in \{1, \ldots, n\}, v \in V, v' \in V\}$
  - Act of process i is extended by Act_x to get and set the variable x
  - Mathematical operations can be derived

Model Construction – 5. Asynchronous message pass.

- Extension similar to shared variables
- Use transition system to model channel
  - parallel composition
  - rename actions as needed



Model Construction - Operators for parallelism (3)

- Pure concurrency and Synchronous product are special cases of synchronous message passing
- Communication via shared variables and Channel systems can be encoded by synchronous message passing

Model Construction Principles
The Stochastic Case

Probabilistic automata - Pure concurrency $|||$

$$\mathcal{D}_1 = (S_1, Act_1, \rightarrow_1, \ldots)$$

$$\mathcal{D}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$$

The composite transition system $\mathcal{D}_1 \mathbin{|||} \mathcal{D}_2$ is given by:

$$\mathcal{D}_1 \mathbin{|||} \mathcal{D}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

where $\rightarrow$ is given by:

$$\frac{s_1 \xrightarrow{\alpha}_1 \mu_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle \mu_1, s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 \mu_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, \mu_2 \rangle}$$

where $\langle \mu_1, s_2 \rangle(\langle s_1', s_2' \rangle) = \mu_1(s_1')$ if $s_2' = s_2$ and 0 otherwise, and $\langle s_1, \mu_2 \rangle(\langle s_1', s_2' \rangle) = \mu_2(s_2')$ if $s_1' = s_1$ and 0 otherwise.

Probabilistic automata - Synch. Message Passing $\|_{Syn}$

Recall:

$$\mathcal{T}_1 = (S_1, Act_1, \rightarrow_1, \ldots) \qquad \mathcal{T}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$$

Concurrent execution with synchronization over all actions in $Syn \subseteq Act_1 \cap Act_2$:

$$\mathcal{T}_1 \|_{Syn} \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

- Interleaving for $\alpha \notin Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

- Handshaking for $\alpha \in Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1' \wedge s_2 \xrightarrow{\alpha}_1 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2' \rangle}$$

Probabilistic automata - Synch. Message Passing $\|_{Syn}$

$$\mathcal{D}_1 = (S_1, Act_1, \rightarrow_1, \ldots) \qquad \mathcal{D}_2 = (S_2, Act_2, \rightarrow_2, \ldots)$$

Concurrent execution with synchronization over all actions in $Syn \subseteq Act_1 \cap Act_2$:

$$\mathcal{D}_1 \|_{Syn} \mathcal{D}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \rightarrow, \ldots)$$

- Interleaving for $\alpha \notin Syn$:

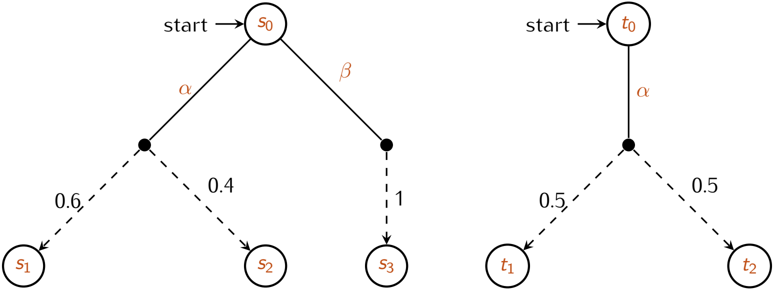
$$\frac{s_1 \xrightarrow{\alpha}_1 \mu_1}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle \mu_1, s_2 \rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 \mu_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, \mu_2 \rangle}$$

- Handshaking for $\alpha \in Syn$:

$$\frac{s_1 \xrightarrow{\alpha}_1 \mu_1 \wedge s_2 \xrightarrow{\alpha}_1 \mu_2}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle \mu_1, \mu_2 \rangle}$$

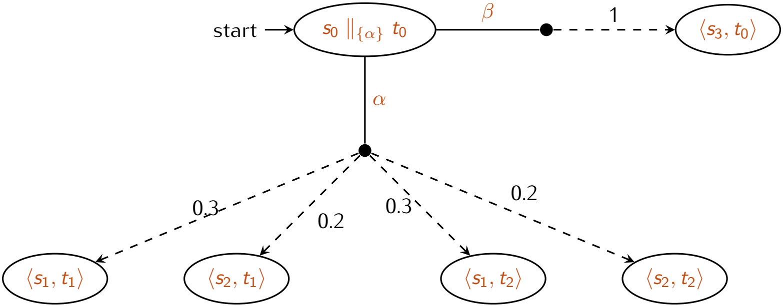
where $\langle \mu_1, \mu_2 \rangle(\langle s_1', s_2' \rangle) = \mu_1(s_1') \cdot \mu_2(s_2')$.

Probabilistic automata - Example



What is $s_0 \parallel_{\{\alpha\}} t_0$?

Probabilistic automata - Example



Probabilistic automata - Parallelism&Communication

- Pure concurrency
- Synchronous product
- Synchronous message passing
- Communication via shared variables
- Channel systems

What is the difference pf PA to MDPs, actually?

Probabilistic automata - Parallelism&Communication

- Pure concurrency
- Synchronous product
- Synchronous message passing
- Communication via shared variables
- Channel systems

What is the difference pf PA to MDPs, actually?

MDP: each state has at most one transition for a given action.
PA: each state can have several transitions for a given action.

Outlook

Further models

- PTA, Attack trees
- STA
- CTMC, CTMDP, fault trees (transient, steady-state, CSL)
- hybrid automata (reachability)
- corresponding games