Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 1 - Solutions

Motivating Model Checking

We presented a factory robot, the task of which was to filter red bricks while letting bricks of other colours pass. The robot was programmed (Fig. 1) using the visual programming language which was supplied along with the LEGO Mindstorms EV3 software. The first task was to figure out whether its possible to fool the robot into letting a red brick pass.

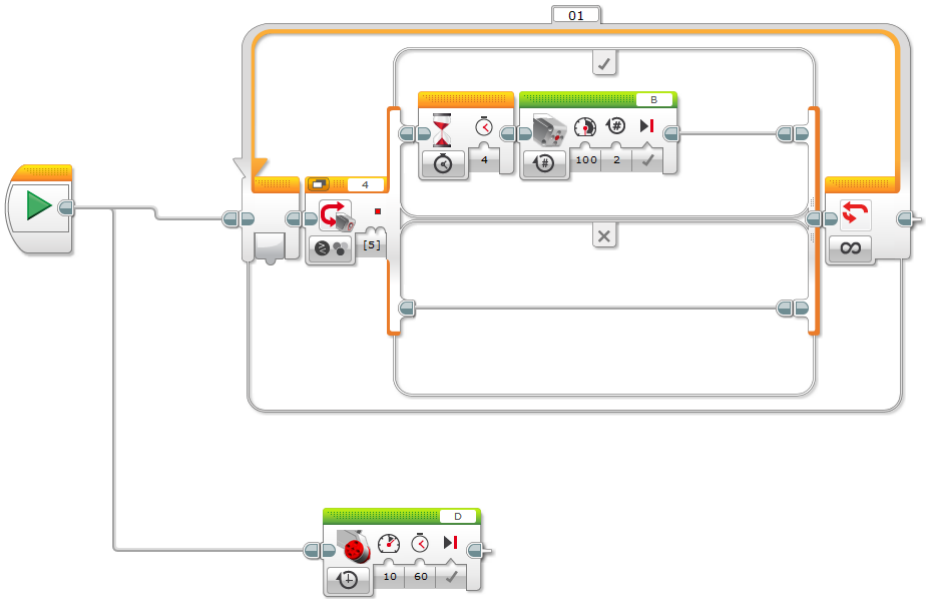


Figure 1: Program which was loaded onto the robot

We then modelled the robot compositionally using the tool `UPPAAL` (`http://www.uppaal.org`). Each of the logical components of the robot were modelled separately – the controller (Fig. 2), colour sensor (Fig. 3b), the kick actuator (Fig. 3a) and the blocks (Fig. 4).

Note: This is just one of the ways which could be used for modelling this system. The verbosity of the models are usually decided based on the property one wants to check. In this case, we wanted to check if it is the case that always a red block would be kicked out.

We then used `UPPAAL`'s verifier in order to check whether there is a case where a red block can pass the belt without being kicked out (`Red1.on_belt --> Red1.kicked`). The counterexample can be visualized in the simulator by setting `Options -> Diagnostic Trace -> Some` before checking the property.

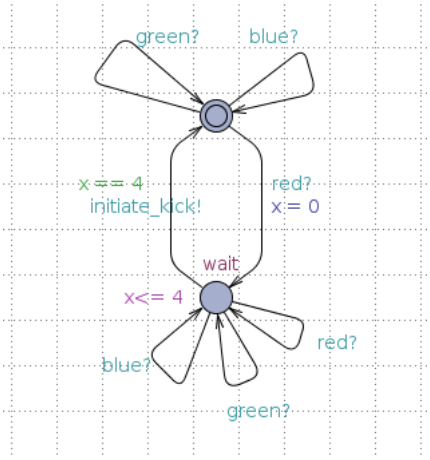
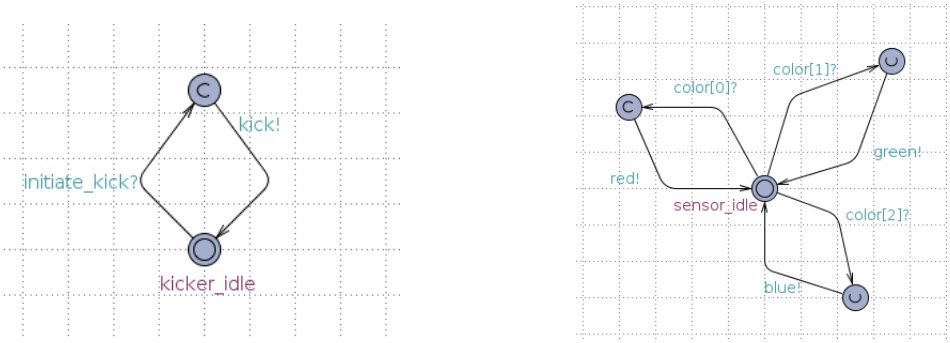


Figure 2: Controller



(a) (b)

Figure 3: (a) Kick actuator and (b) color sensor

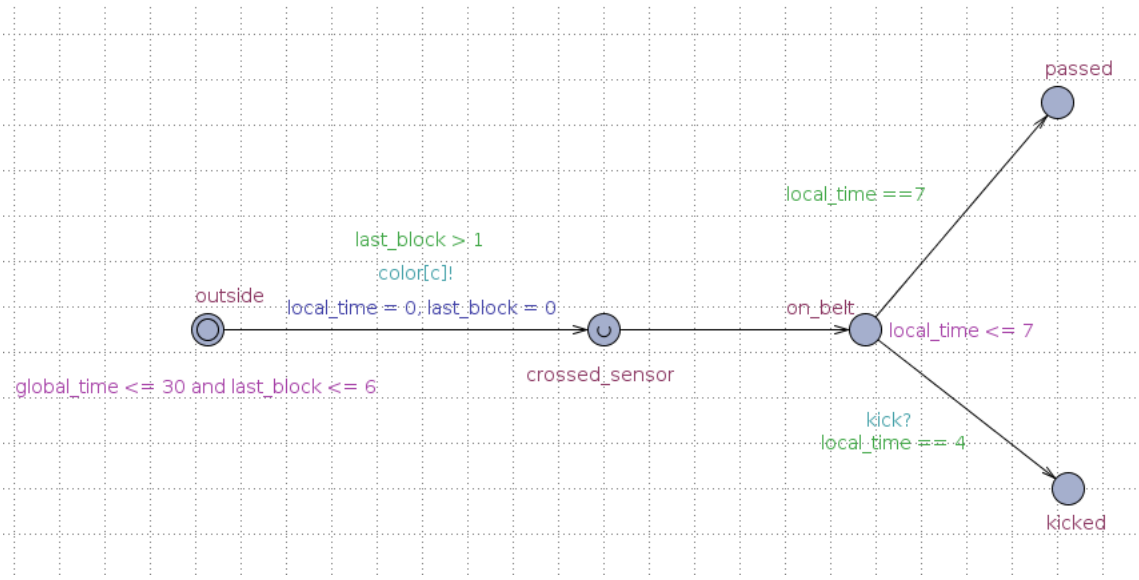


Figure 4: Block

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 2

Ex 1: Modelling[1]

Consider an autonomous elevator which operates between two floors. The requested behaviour of the elevator is as follows:

1. The elevator can stop either at the ground floor or the first floor. When the elevator arrives at a certain floor, its door automatically opens. It takes at least 2 seconds from its arrival before the door opens but the door must definitely open within 5 seconds.
2. Passengers can enter if and only if the elevators doors are open. They enter one by one and we (optimistically) assume that the elevator has a sufficient capacity to accommodate any number of passengers waiting outside.
3. The door can close only 4 seconds after the last passenger entered. After the door closes, the elevator waits at least 2 seconds and then travels up or down to the other floor.

Your task is to

1. model the elevator in `UPPAAL`,
2. think about which of the above properties you can formalize in `UPPAAL`, and
3. check your model by testing said properties.

How does your model change when additional floors are added? Can you model passengers on different floors or a limited capacity of the elevator? *Note: `UPPAAL` also supports non-clock variables.*

Ex 2: Modelling Continued

Algorithm 1 Fischer's Mutual Exclusion Protocol.

Require: id: Global, atomic variable, initialized to 0. *delay*: waiting time parameter
  while true do
    if $id \neq 0$ **then continue**
    $id \leftarrow i$
    pause(*delay*)
    if $id \neq i$ **then continue** L
    `// critical section`
    $id \leftarrow 0$

Model Fischer's mutual exclusion protocol (shown in Algorithm 1) in `UPPAAL`. For a system of 10 processes following this protocol, verify the listed properties.

1. Mutual exclusion (no two processes are in the critical section at the same time).
2. The system is deadlock free.
3. Whenever P(6) requests access to the critical section it will eventually enter the wait state.

[1] Based on `http://www.ru.is/kennarar/luca/GSSI/TUTORIALS/tutorial-ta1.pdf`

Hw 1: Modelling a Train

Note: This is a (voluntary) homework to further train modelling in `UPPAAL`*.*

Modelling

Construct a system modelling trains from multiple tracks crossing a bridge with a single track. The expected behaviour of the train is as follows.

- When the Train approaches a bridge, it sends a signal to the controller.
- If the bridge is occupied, the controller sends a stop signal to the train within 10 time units.
- Otherwise, if the train doesn't receive a stop signal within 10 time units, it starts to cross the bridge within 20 time units. It takes the train 3 to 5 time units to leave the bridge.
- If the train receives a stop signal within 10 time units, it comes to a stop. When it receives a go signal from the controller, it starts moving within 15 time units and it takes at least 7 time units to enter the bridge.

Design a controller which uses an FCFS strategy to process requests. If the bridge is free and a train requests to use it, add it to a queue. When the train leaves, remove it from the queue. If the bridge is being used, always add the train to the queue and ask it to wait until its turn.

Verification

For a system with three trains, verify the following properties.

- Train 1 can reach the other side.
- Train 0 can be crossing the bridge while Train 1 is waiting to cross.
- Train 0 can cross the bridge while other trains are waiting to cross.
- There is never more than one train crossing the bridge.
- There can never be three elements in the queue.
- Whenever a train approaches the bridge, it will eventually cross.
- The system is deadlock-free.

Note: You can also try to verify the timing constraints by equipping the model with additional clocks.

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 3

Ex 1: Region Construction

Draw the region automaton simulating the timed automaton in Fig. 1.

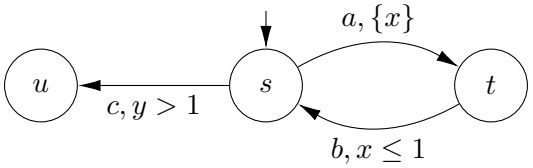
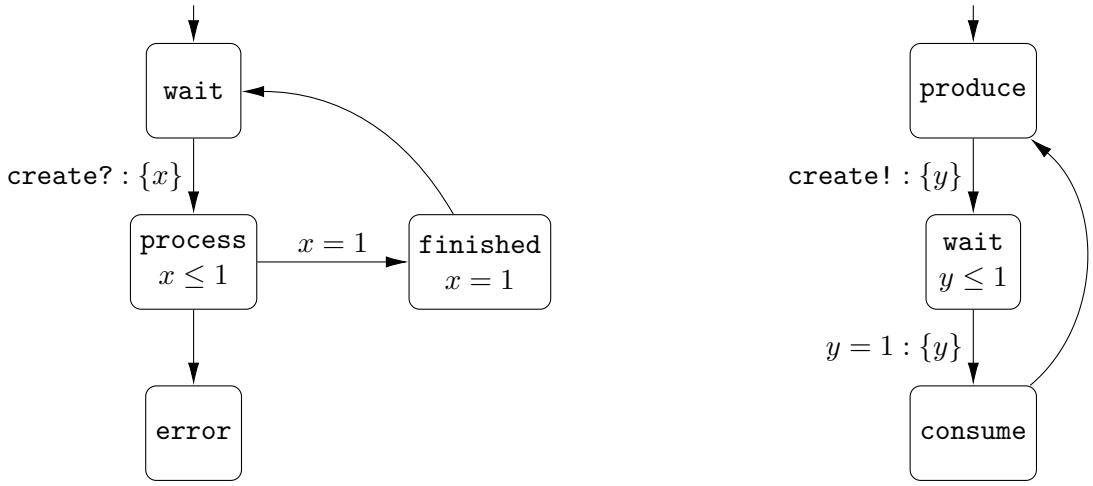


Figure 1: A Timed Automaton

Ex 2: TA Network

Consider the parallel composition of the following timed automata.



Which of the following sets of states is reachable in the composition? For each reachable set, write down a path to this set. For unreachable sets, (informally) argue why you think it is unreachable.

1. $(\texttt{error}, \cdot, \cdot)$
2. $(\texttt{finished}, \texttt{consume}, \{x \mapsto 1, y \mapsto 1\})$
3. $(\texttt{wait}, \texttt{produce}, \{x \mapsto 2, y \mapsto 2\})$
4. $(\texttt{process}, \texttt{produce}, \cdot)$
5. $(\texttt{wait}, \cdot, \{x \mapsto 3, y \mapsto 2\})$

Note: The semantics from the lecture are slightly different from those of `UPPAAL`.

HW 1: TCTL

Briefly read through the lecture notes regarding TCTL (slides 61-68). TCTL modelling will be practised next session.

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 4

Ex 1: Zone Construction

Draw the zone automaton simulating the timed automaton in Fig. 1.

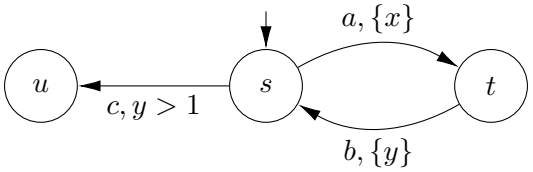


Figure 1: A Timed Automaton

Ex 2: TCTL

Let $\mathsf{AP} = \{\texttt{request}, \texttt{grant}, \texttt{error}, \texttt{idle}\}$ be atomic propositions and $C = \{x, y\}$ clocks. Express the following properties in TCTL:

- `error` is reachable within 100 time units.
- The system is `idle` in time units five to ten.
- For the first 1000 time units, whenever there is a `request`, a `grant` will follow in at most five time steps.
- The system is never `idle` on an `error` within the first 1000 time units.
- Whenever $y > 10$ within the first 1000 time units, the system is in `error` within 10 steps.
- At any state reachable within 1000 time units, `error` is reachable within 30 time units.
- Within the first 1000 time units, if a `request` can't be `grant`ed within three time units, then $x > 4$ and `error` will be reached within ten time units.

Important: In the tutorial, we discussed a different possible definition of TCTL than the one from the lecture.

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 5

Ex 1: TCTL Model Checking

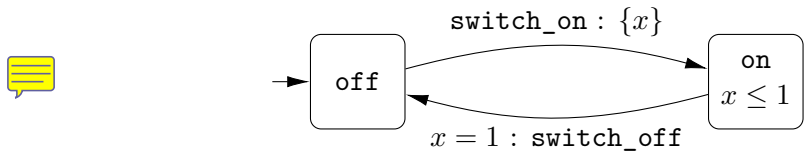
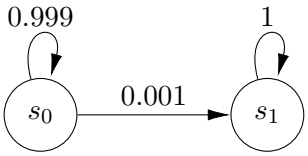


Figure 1: Timed automaton modelling a light

Consider the timed automaton shown in Fig. 1. Model check the TCTL properties "$\exists\Diamond^{\leq 1}\texttt{on}$" and "$\forall\Diamond^{\leq 1}\texttt{on}$". To this end, draw the region transition system, augmented with a new clock z.

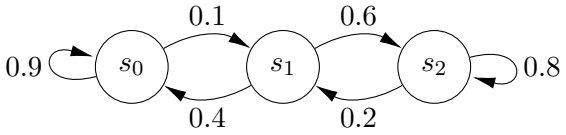
Ex 2: Reachability

Compute the probability of reaching s_1 from s_0 in the following simple Markov Chain. Additionally, compute the probability of reaching s_1 within 10 steps.



Ex 3: Matrix Representation

Write down the matrix representation of the following Markov Chain. Suppose the initial distribution is $\pi_0 = [1, 0, 0]$, i.e. the process starts in s_0. What is the transient distribution after three time steps?



Ex 4: Proof

Prove the following statements:

- Let P be a stochastic matrix, i.e. the matrix representation of some Markov Chain. Then, $\frac{1}{2}P + \frac{1}{2}I$ is aperiodic, where I is the unit matrix.
- There exists a finite state Markov Chain with an unique stationary distribution π^*, but for any $n \in \mathbb{N}$ we have that $\pi_n = P^n\pi_0 \neq \pi^*$.
- In the lecture, we saw that if all states are irreducible, aperiodic and recurrent non-null in a Markov Chain, there is an unique limiting distribution which does not depend on π_0. Show that each of these properties is required by finding a Markov Chain which does (i) not have a unique limiting distribution, and (ii) satisfies all but one of the properties.

HW 1: Get PRISM

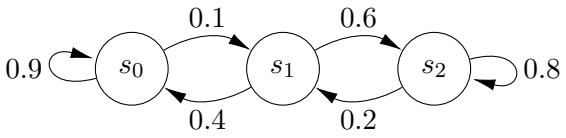
Download and install the `PRISM` Model Checker from `http://www.prismmodelchecker.org/download.php` or `https://github.com/prismmodelchecker/prism/releases`. Familiarize yourself with the GUI and the modelling language for DTMC.

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 6

Ex 1: Matrix Representation

Write down the matrix representation of the following Markov Chain. Suppose the initial distribution is $\pi_0 = [1, 0, 0]$, i.e. the process starts in s_0. What is the transient distribution after three time steps?

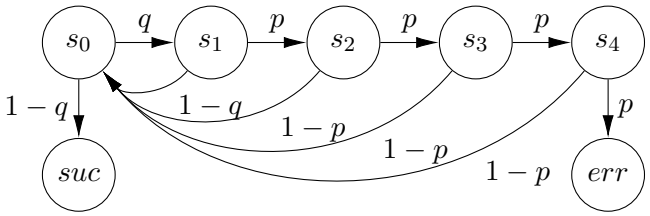


Ex 2: Modelling

Imagine jobs arriving at a server with an unbounded queue. The server works on a job at the head of the queue and, when finished, moves on to the next job. The server never drops a job, but just allows them to queue up. At every time step, with probability $p = \frac{1}{50}$ one job arrives, and independently, with probability $q = \frac{1}{30}$ one job departs, i.e. is finished by the server. Note that during a time step, we might have both an arrival and a transmission, or neither. Draw the Markov Chain modelling this server (assuming that you are interested in studying the number of jobs in the system).

Ex 3: PRISM

A simplified version of the IPv4 Zeroconf protocol is outlined below. Model the protocol in `PRISM` and compute the transient probabilities for the first few steps.



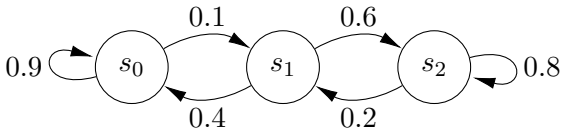
1. Randomly pick an address among the K (65024) addresses.
2. With m hosts in the network, collision probability is $q = \frac{M}{K}$
3. Send 4 ARP requests.
4. In case of collision, the probability of no answer to the ARP request is p (due to the lossy channel).

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 7

Ex 1: Steady State

Compute the steady state distribution of the following Markov Chain by solving the corresponding linear equation system.



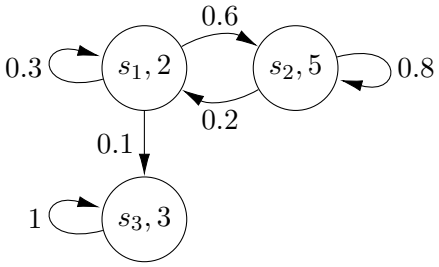
Ex 2: Proof

Prove the following statements:

- Let P be a stochastic matrix, i.e. the matrix representation of some Markov Chain. Then, $\frac{1}{2}P + \frac{1}{2}I$ is aperiodic, where I is the unit matrix.
- There exists a finite state Markov Chain with an unique stationary distribution π^*, but for any $n \in \mathbb{N}$ we have that $\pi_n = P^n\pi_0 \neq \pi^*$.
- In the lecture, we saw that if all states are irreducible, aperiodic and recurrent non-null in a Markov Chain, there is an unique limiting distribution which does not depend on π_0. Show that each of these properties is required by finding a Markov Chain which does (i) not have a unique limiting distribution, and (ii) satisfies all but one of the properties.

Ex 3: Average Reward

For each initial distribution π, compute the average reward obtained in the following Markov Chain.



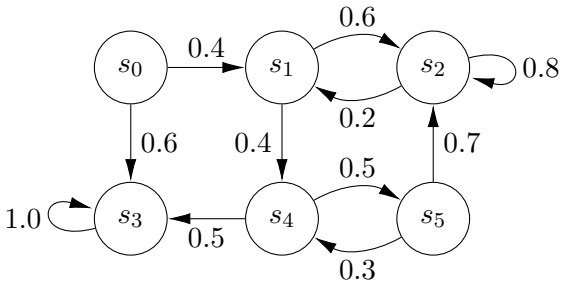
Given an initial distribution π and step bound n, how can you compute the average (accumulated) n-step reward? What is the 2-step average reward for $\pi = \{s_1 \mapsto 1\}$?

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 8

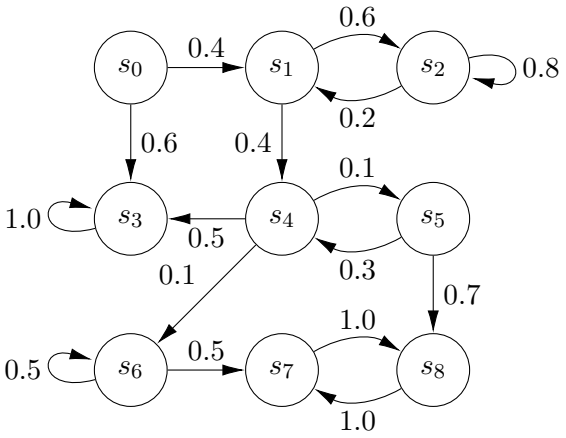
Ex 1: Reachability

Compute the probability of reaching the set $\{s_1, s_4\}$ in the following Markov Chain, using the method from the lecture.



Ex 2: Connected Components

For the following Markov Chain, determine the set of its strongly connected components and identify which are "bottom".



Ex 3: Proof

Prove that a finite DTMC is irreducible iff its induced graph is strongly connected.

Ex 4: Cost-Bounded Reachability

Consider a finite Markov Chain with non-negative, integer (state)rewards. Given a state set B and threshold T, how can you (algorithmically) compute the probability of reaching B while not exceeding a total accumulated cost of T? What happens if we allow non-negative, rational rewards?

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 9

Remark

We use the following notation short-hands:

- $\mathsf{F}_{\sim p}\,\phi := \mathcal{P}_{\sim p}\,[\mathsf{F}\,\phi]$, and
- $\mathsf{G}_{\sim p}\,\phi := \mathcal{P}_{\sim p}\,[\mathsf{G}\,\phi]$

for $\sim \in \{=, <, \leq, >, \geq\}$, $p \in [0,1]$, and $\phi \in \mathsf{PCTL}$. We define analogous abbreviations for the step-bounded versions of F and G.

Ex 1: Logic Modelling

Translate the following formulae to English

1. $\mathtt{send} \implies \mathsf{F}^{\leq 10}_{\geq 0.95}\,\mathtt{deliver}$
2. $\mathcal{P}_{\leq 0.05}\left[\mathsf{F}\,\mathsf{F}_{\geq 0.9}\,\mathtt{error}\right]$
3. $\mathcal{P}\left[\mathsf{G}\,(\mathtt{send} \implies (\mathtt{empty}\,\mathsf{U}\,\mathtt{receive}))\right] \geq 0.5$
4. $\mathcal{P}_{\geq 0.8}\left[\mathtt{empty}\,\mathsf{U}\,(\mathtt{send} \wedge \mathsf{G}_{\leq 0.5}\,\neg\mathtt{receive})\right]$

Translate the following specifications into PCTL / PLTL formulae

1. The system with two processes satisfy mutual exclusion almost surely ($\mathtt{crit}_i$ holds if process i is in the critical section)
2. The probability that every `request` will eventually be `granted` with a probability greater than 0.95 is 0.99.
3. The probability that component B fails (`B_fail`) before component A (`A_fail`) is less than 0.4.
4. If the system is not operational ($\neg\mathtt{up}$), it almost surely reaches a state from which it has a greater than 0.99 chance of staying operational ($\mathtt{up}$) for 100 time units.

Ex 2: PCTL Satisfiability

For each of the following properties, draw a labelled Markov Chain which satisfies it or argue why the property is unsatisfiable.

1. $\mathsf{G}_{\leq 0.5}\,(a \wedge \neg b)$
2. $\mathsf{G}_{=1}\,(\neg a \wedge \mathsf{F}_{=1}\,a)$
3. $\neg a \wedge \mathcal{P}_{\geq 1}\,[b\,\mathsf{U}\,a]$
4. $\mathsf{F}_{=1}\,(\mathtt{a} \implies (\mathsf{G}_{=1}\,((b \implies c)\,\mathsf{U}\,\neg a \wedge (\mathsf{F}_{\geq 0.5}\,c \vee \neg b))))$
5. $\mathsf{G}_{>0}\,(\neg a \wedge \mathsf{F}_{>0}\,a)$ (*Note*: Tricky)

Ex 3: Automata

Draw a Rabin automaton for the following formulae:

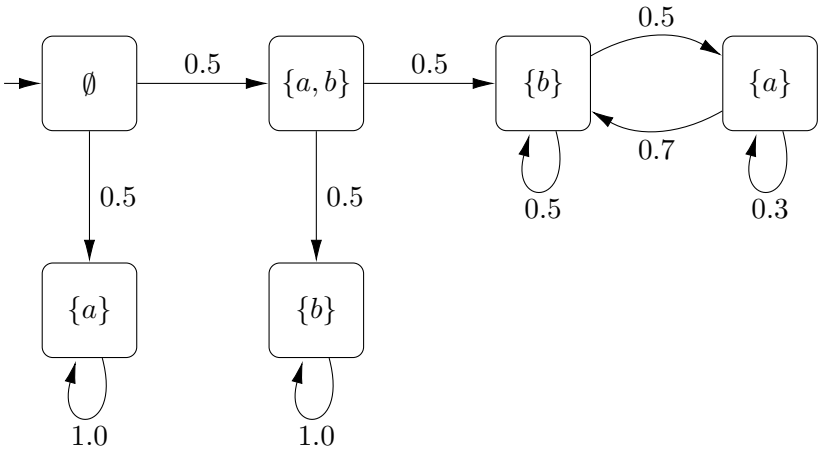
1. $\mathsf{G}\,\neg a \wedge \mathsf{G}\,\mathsf{F}\,b$
2. $\mathsf{G}\,(a \implies \mathsf{F}\,b)$

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 10

Ex 1: PLTL Model Checking

Consider the following Markov Chain.



Check whether $\mathcal{P}[\mathsf{F}\,\mathsf{G}\,a \vee \mathsf{F}\,\mathsf{G}\,\neg a] = 1$ holds on the given Markov Chain, using the methods from the lecture.

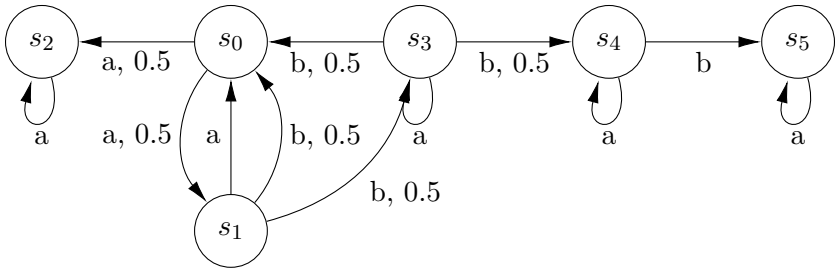
Ex 2: MDP Modelling

Model the following situation as an MDP.

> A robot is placed in a 3x2m arena at the south-west corner $(0, 0)$. Its goal is to reach a *repair station* at $(3, 1)$, but the arena is not without its challenges. At $(2, 0)$ and $(3, 2)$ are two rotor blades spinning at 20000 RPM which is certain death for the robot. A previous close encounter with another such hazard has left the robot with certain eccentricities. Firstly, it cannot move south or west. Secondly, if it tries to move north, it succeeds only 90% of the time. Finally, 10% of attempts to move east is accompanied with a major southwards deviation, which makes the robot end up 1m south of its intended target.

Ex 3: MDP Reachability

First, consider the following MDP.



What is the maximal probability of reaching s_2? What is the optimal strategy for maximizing the probability of reaching s_5? How many optimal strategies are there?

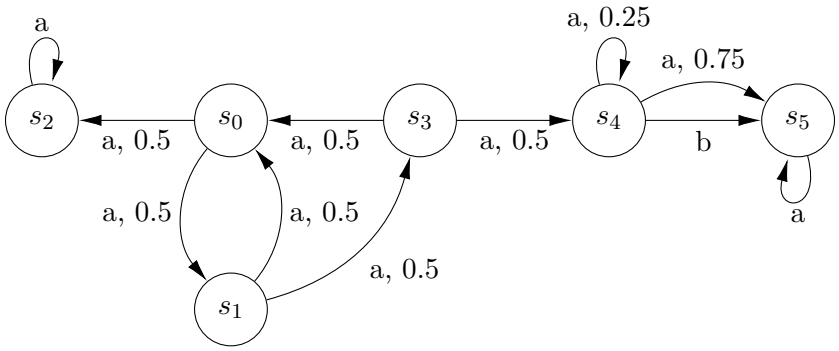
How would you try to solve this type of questions in general, e.g., for the MDP in the previous exercise?

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 11

Ex 1: Reachability LP

Consider the following MDP.



Write down the reachability LP for $B = \{s_5\}$.

Ex 2: Bounded Reachability

In the lecture, we learned that memoryless schedulers are sufficient for (unbounded) reachability, but not for its bounded counterpart. Try to come up with an MDP where some bounded reachability query can only be maximized by a scheduler with memory. Argue why finite memory is sufficient.

How would you solve a bounded reachability query in general?

Ex 3: Sound Value Iteration

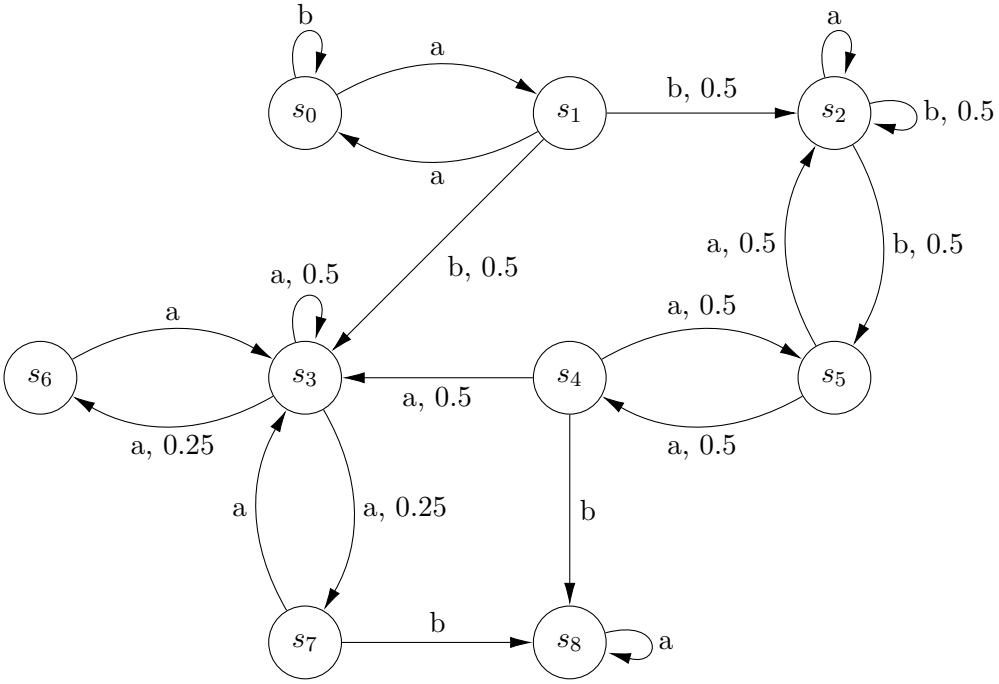
We defined the "stopping criterion" of value iteration as $\max_s |x_{n+1}(s) - x_n(s)| < \varepsilon$ for some small ε. This tends to work in practise, but is not sound in general, i.e. there are some MDP where this property is fulfilled, but the resulting strategy is not optimal. Can you think of such an MDP?

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 12

Ex 1: MEC Decomposition

Compute the MECs (both states and actions) of the following MDP.



Ex 2: Reachability as special case

In the lecture, we saw expected step-bounded reward and expected long-run average reward. How can you rephrase (bounded) reachability as an instance of these problems?

Ex 3: Discounted Reward

As a "trade-off" between step-bounded reward and long-run average, one can define discounted reward. Given a "discount factor" γ with $0 < \gamma < 1$, the reward obtained by a single run $s_0 s_1 \cdots$ is

$$\sum_{i=0}^{\infty} \gamma^i r(s_i).$$

One can easily define a value iteration algorithm by iterating

$$x_{n+1}(s) = r(s) + \gamma \max_{a \in \mathsf{Act}(s)} \sum_{s'} P(s, a, s') x_n(s')$$

Prove that this iteration converges in the limit by using the Banach fixed-point theorem (also known as contraction mapping theorem).

Hint: Use the $\mathcal{L}^\infty$-norm.

Technische Universität München
Institut für Informatik I07
Theoretical Computer Science

Quantitative Verification 13

Ex 1: Modelling a CTMC

Model the following scenario as a CTMC and uniformize it.

> We consider a printer in a local network. Four printing jobs arrive consecutively with a rate of 2 and are placed in the printer's queue. It has a queue of size two. The jobs are handled with a rate of 1. If the queue overflows, the (buggy) firmware of the printer crashes.

Ex 2: Two Views on CTMC

In the lecture, we switched between two views on CTMCs. First, waiting in a state for some time distributed exponentially with rate $E(s) = \sum_{s'} R(s, s')$ and then randomly moving to a successor proportional to the respective rates. Second, running an exponential process for each successor separately and picking the first one. In this exercise, we show that the second view implies the first.

To this end, consider the following. Let X and Y be two independent, exponentially distributed variables with rates λ_X and λ_Y, respectively. How is $Z := \min\{X, Y\}$ distributed?

Ex 3: Self loops

Show that adding or removing a self loop to a state in a CTMCs does not change the probability distribution over its successors.