Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

AALBORG UNIVERSITY
DENMARK

Discussion: 02. Feb 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 1
WITH SOLUTIONS

Exercise 1: UPPAAL introduction

In this exercise we show how to encode of the Switch extended state machine (Chap 2, Slide 24) in the tool UPPAAL. Later we ask you to do some modifications and analysis.

(a) Watch the introductory video to UPPAAL which can be found in the MS-teams files.

(b) Download UPPAAL from `www.uppaal.org`, download the UPPAAL Switch model from the MS-teams files.

(c) Modify the UPPAAL model model such that the process `User` turns the light on and off every three rounds.

(d) In your modified version, what is the maximal reachable value for local variable `x`?

(e) Verify that your model is deadlock free.

.... Solution ....

Download the UPPAAL model solution from MS-Teams.

Exercise 2: Induction

Prove by induction on k that

$$1 + \sum_{i=1}^{k} (2i + 1) = (k + 1)^2$$

for all $k \geq 1$.

.... Solution ....

Base case $k = 1$: One readily verifies that $1 + (2 + 1) = (1 + 1)^2$. Induction step $k \to k + 1$: By observing that

$$1 + \sum_{i=1}^{k+1} (2i + 1) = 1 + \sum_{i=1}^{k} (2i + 1) + (2(k + 1) + 1),$$

an application of the induction hypothesis on $1 + \sum_{i=1}^{k}(2i+1)$ yields

$$1 + \sum_{i=1}^{k+1}(2i+1) = (k+1)^2 + (2(k+1)+1).$$

This, in turn, yields the claim because:

$$\begin{aligned}1 + \sum_{i=1}^{k+1}(2i+1) &= (k+1)^2 + (2(k+1)+1)\\ &= k^2 + 2k + 1 + 2k + 2 + 1\\ &= k^2 + 4k + 4\\ &= (k+2)^2\\ &= ((k+1)+1)^2\end{aligned}$$

Exercise 3: Predicate logic
Formalize the following informal statements by using predicate logic.

(a) There exists a real number x such that $x + x$ is greater than 8.

(b) Every real number x is the double of some real number y.

(c) All natural numbers are positive.

.... Solution ....

(a) $\exists x. x + x > 8$

(b) $\forall x. \exists y. x = 2y$

(c) $\forall x \in \mathbb{N}. x > 0$

Exercise 4: Sets
Let us assume that we are given the sets $A = \{1, 2, 3\}$ and $B = \{2, 3\}$.

(a) Does $A \subseteq B$ hold true?

(b) Does $B \subseteq A$ hold true?

(c) In what relation stands B with respect to A?

(d) Find the set $A \cup B$.

(e) How does one call $A \cup B$?

(f) Find the set $A \cap B$.

(g) How does one call $A \cap B$?

(h) Find the set $A \times B$.

(i) How does one call $A \times B$?

(j) Find $\mathcal{P}(B)$, the power set of B.

.... Solution ....

(a) No, since $1 \notin B$.

(b) Yes, since $2, 3 \in A$.

(c) B is a subset of A. Alternatively, A is a superset of B.

(d) A.

(e) The union of A and B.

(f) B.

(g) The intersection of A and B.

(h) $\{(a, b) \mid a \in A, b \in B\} = \{(1, 2), (1, 3), (2, 2), (2, 3), (3, 2), (3, 3)\}$.

(i) The product set of A and B.

(j) The powerset of B is given by all subsets of B, i.e., $\mathcal{P}(B) = \{A \mid A \subseteq B\}$. Hence, the answer is $\{\emptyset, \{2\}, \{3\}, \{2, 3\}\}$. Note that the elements of the powerset are sets.

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

AALBORG UNIVERSITY
DENMARK

Discussion: 9. Feb 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 2
WITH SOLUTIONS

Exercise 1: UPPAAL simplified cruise controller
Consider the cruise controller from the lecture, one sub-component is the `SetSpeed` controller (Chap 1, Slides 20-21). The `SetSpeed` component takes as input the current `speed`, and the inputs `cruise`, `in`, `dec`, `pause` from the driver. Solve the following where (*) indicates that the exercise is optional.

(a) Download the initial UPPAAL model (sheet2CruiseController.xml) which can be found in the MS-teams files.

(b) Verify if the desired speed is between 40 and 80 speed units. You can use the following Timed Computation Tree Logic (TCTL) formula φ

$$\forall\Box \texttt{desired\_speed} \geq 40 \land \texttt{desired\_speed} \leq 80.$$

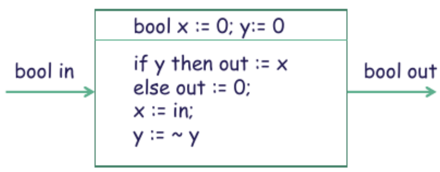
(c) Modify the component `SetSpeed` such that the above formula φ is satisfied.

(*) Create a component `DriverFast` that increases the variable `desired_speed` until it reaches the value 78. When the variable `desired_speed` is at value 78, the component enters a location `stable`. Verify the desired behavior by checking the following TCTL property.

$$\forall\Diamond \texttt{DriverFast.stable} \land \texttt{desired\_speed} == 78$$

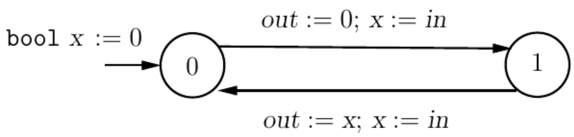
.... Solution ....
Download the solution file from MS-Teams

Exercise 2: Extended-state-machine
Describe the component `OddDelay` (Chapter 2, Slide 23) shown below as an extended-state machine with two modes. The mode of the state machine should capture the value of the state variable y, while the state variable x should be updated using assignments in the mode-switches.



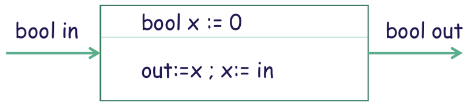
. . . . Solution . . . .

The extended-state-machine corresponding to the component `OddDelay` is shown below. The modes correspond to the values of the state variable y.



Exercise 3: Mealy machine

Consider the `Delay` component



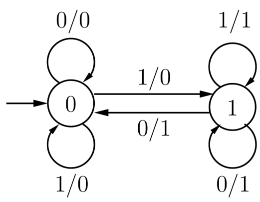
and suppose we replace the reaction description by

```
out := x; x := choose(in, x)
```

Describe in words the behavior of the modified component. Draw the Mealy machine corresponding to the component.

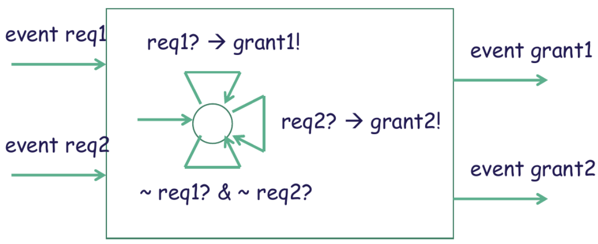
. . . . Solution . . . .

The component is nondeterministic. In state 0 (that is, state where the value of x equals 0), the component outputs 0, and if the input is 0, the state stays unchanged, while if the input is 1, the state either stays unchanged or is updated to 1. Symmetrically, in state 1, the component outputs 1, and if the input is 1, the state stays unchanged, while if the input is 0, the state either stays unchanged or is updated to 0. The two-state Mealy machine corresponding to the component is shown below:



Exercise 4: Reaction description

For the nondeterministic component Arbiter shown below, the reactions are expressed using the extended-state machine notation. Write an equivalent description using straight-line update code. You can use a local variable whose value is assigned nondeterministically using the choose construct.



. . . . Solution . . . .

The following code can be used as the reaction description of the component `Arbiter`. The value of the local variable x is chosen non-deterministically, and when both the input request events are present, its value is used to decide whether to issue the output event $grant_1$ or to issue the output event $grant_2$.

```
local bool x := choose(0, 1);
if req1? then
   if req2? then
      if (x == 0) then grant1! else grant2!
   else grant1!
else if req2? then grant2!.
```

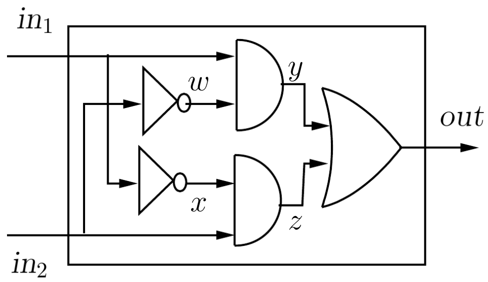
Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

Discussion: 16. Feb 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 3
WITH SOLUTIONS

Exercise 1: Uppaal Synchronous circuits

An Xor (Exclusive-Or) gate has two Boolean inputs in1 and in2 , and a Boolean output out. The output is 1 when exactly one of its two inputs are 1 and is 0 otherwise. The combinational component `SyncXor` below is defined by composing And, Or, and Not gates.

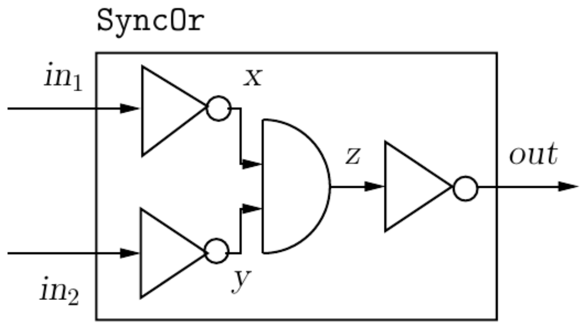


(a) Download the Uppaal model (SyncXOr.xml) which can be found in the MS-teams files. Get familiarized with the model by looking at the declarations and running some simulations. Note that the component `SyncXor` is defined as the parallel composition of the components `SyncNot1, SyncNot2, SyncAnd1, SyncAnd2, SyncOr1`. Observe that `SyncNot1=SyncNot(1,in2,w)` is an instance of the template `SyncNot(int id, bool &in, bool &out)` where `id` is a component identifier and `&in,&out` are reference values (C like references). Finally, note that the precedence constraints for these components is given by the matrix `precedence_rel`.

(b) Verify that the desired output is produced by checking following Timed Computation Tree Logic (TCTL) formula

$$\forall\Box \mathtt{t} > 0 \Rightarrow (\mathtt{out} == ((\mathtt{in}_1 \wedge \neg\mathtt{in}_2) \vee (\neg\mathtt{in}_1 \wedge \mathtt{in}_2))$$

(c) Copy the file SyncXor.xml to another file e.g. SyncOr.xml. Modify the Uppaal model to produce the component `SyncOr` described in the lecture (Chap 2, Slide 76) and depicted below.



(d) Verify that the desired output is produced by checking following Timed Computation Tree Logic (TCTL) formula

$$\forall\Box \texttt{t} > 0 \Rightarrow (\texttt{out} == (\texttt{in}_1 \vee \texttt{in}_2))$$

.... Solution ....
Download the solution file SyncOr.xml from MS-Teams

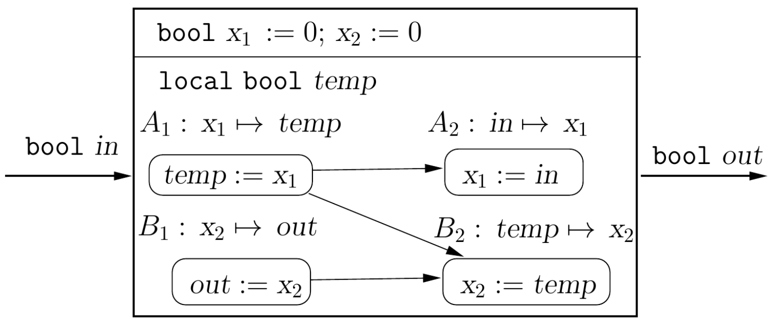
Exercise 2: Parallel Composition

Consider the component `DoubleSplitDelay` and its textual description

$$(\texttt{SplitDelay}[\texttt{out} \mapsto \texttt{temp}] \| \texttt{SplitDelay}[\texttt{in} \mapsto \texttt{temp}]) \setminus \{\texttt{temp}\}$$

This component is similar to the component `DoubleDelay` except we use instances of the component `SplitDelay` (Chap 2, Slide 50) instead of Delay. Show the block diagram version of `DoubleSplitDelay`, that is, list its state, input, output, and local variables, tasks, and precedence constraints. What are the await dependencies among output and input variables for `DoubleSplitDelay`?

.... Solution ....
The component DoubleSplitDelay has input variable `in`, output variable `out`, state variables `x1` and `x2` , and local variable `temp`, all of type bool. Its reaction description consists of 4 tasks as shown below.



The output variable `out` does not await the input variable `in`.

Exercise 3: Second to minute

Recall the event-triggered component `SecondToMinute` (Chap2, Slides 32,72) with the input event variable `second` and the output event variable `minute` such that `minute` is present every 60th time the event `second` is present. Now suppose we want to design an event-triggered component `SecondToHour` with an input event variable `second` and an output event variable `hour`, such that the output event hour is present every 3600th time the event `second` is present. Show how to construct (textual description) the desired component `SecondToHour` from the component `SecondToMinute` using the operations of parallel composition, instantiation, and output hiding.

.... Solution ....

$$(\texttt{SecondToMinute} \| \texttt{SecondToMinute}[\texttt{minute} \mapsto \texttt{hour}][\texttt{second} \mapsto \texttt{minute}]) \setminus \{\texttt{minute}\}.$$

Exercise 4: Induction, sum squared

Proof by induction on n

$$(1 + 2 + \cdots + n)^2 = (1^3 + 2^3 + \ldots n^3)$$

for all $n \geq 1$.

.... Solution ....

- Base case $n = 1$. Then $1^2 = 1^3$ as required.
- Induction step $n \implies n + 1$

$$\begin{array}{rcll}
(1 + 2 + \cdots + n + (n+1))^2 & \stackrel{?}{=} & (1^3 + 2^3 + \ldots n^3 + (n+1)^3) & \\
(\frac{n(n+1)}{2} + (n+1))^2 & \stackrel{?}{=} & & \text{sum identity} \\
(\frac{n(n+1)}{2})^2 + 2(\frac{n(n+1)}{2})(n+1) + (n+1)^2 & \stackrel{?}{=} & & \\
(\frac{n(n+1)}{2})^2 + n(n+1)(n+1) + (n+1)^2 & \stackrel{?}{=} & & \\
(\frac{n(n+1)}{2})^2 + n^3 + 3n^2 + 3n + 1 & \stackrel{?}{=} & & \\
(\frac{n(n+1)}{2})^2 + n^3 + 3n^2 + 3n + 1 & \stackrel{?}{=} & (1 + 2 + \cdots + n)^2 + (n+1)^3 & \text{by I.H.} \\
(\frac{n(n+1)}{2})^2 + n^3 + 3n^2 + 3n + 1 & \stackrel{?}{=} & (\frac{n(n+1)}{2})^2 + (n+1)^3 & \text{sum identity} \\
(\frac{n(n+1)}{2})^2 + n^3 + 3n^2 + 3n + 1 & = & (\frac{n(n+1)}{2})^2 + n^3 + 3n^2 + 3n + 1 & \text{qed}
\end{array}$$

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

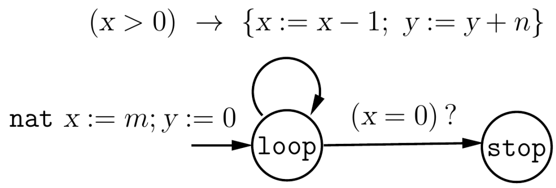
AALBORG UNIVERSITY
DENMARK

Discussion: 23. Feb 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 4
WITH SOLUTIONS

Exercise 1: UPPAAL: `GCD` **and** `Mult`

Consider the Euclid's algorithm to compute the Greatest Common Divisor of two natural numbers (Chap 3, Slide 7). Download the UPPAAL model "GCD.xml" from MS-Teams. Let $m = 260$ and $n = 10$. Verify that it computes the correct answer by checking the following invariant: $\forall\Box$`GCD.stop` $\Rightarrow$ `GCD.x` $== 10$. How many states are reachable? (Hint: You can run the following in a terminal: ../uppaal64-4.1.24/bin-Linux/verifyta -u GCD.xml). The following state machine is used to compute the multiplication of two natural numbers:



(a) In a similar way to the UPPAAL model `GCD` construct a UPPAAL model `Mult` for the above state machine.

(b) Let $m = 2000$ and $n = 2$. Verify that your model computes the correct answer by checking the following invariant: $\forall\Box$`Mult.stop` $\Rightarrow$ `Mult.y` $== m * n$

(c) How many states are reachable?

.... Solution ....

Download solution from MS Teams.

Exercise 2: Reachable states

The composed system

$$\texttt{RailRoadSystem1} = \texttt{Controller1} \| \texttt{Train}_{\texttt{W}} \| \texttt{Train}_{\texttt{E}}$$

(Chap 3, Slides 30-33) has four state variables, `east` and `west`, each of which can take two values, and $\texttt{mode}_W$ and $\texttt{mode}_E$, each of which can take three values. Thus, `RailRoadSystem1` has 36 states. How many of these 36 states are reachable?

.... Solution ....

Each state is denoted by listing the values of the variables `west`, `east`, $\texttt{mode}_W$, and $\texttt{mode}_E$, in that order. We use a, w, b, g, and r, as abbreviations for the values `away, wait, bridge, green, red`, respectively. Then, the initial state is $ggaa$, and has transitions to itself and to the states $rgaw$, $grwa$, and $rgww$. To compute the set of reachable states, we need to explore transitions from these three newly discovered states, and keep repeating till no new states are found to be reachable. It turns out that the following 13 states are reachable:

$$\{ggaa, rgaw, grwa, rgww, rgab, rrwb, grba, rrbw, rgwb, ggwa, ggba, rgbw, rgbb\}$$

Exercise 3: Inductive invariants

Consider a transition system T with two integer variables `x` and `y` and a Boolean variable `z`. All the variables are initially 0. The transitions of the system correspond to executing the conditional statement

```
if (z = 0) then {x := x + 1; z := 1} else {y := y + 1; z := 0}
```

Consider the property φ given by $(x = y) \vee (x = y+1)$. Is φ an invariant of the transition system T? Is φ an inductive invariant of the transition system T? Find a formula ψ such that ψ is stronger than φ and is an inductive invariant of the transition system T . Justify your answers.

.... Solution ....

The system has a single execution given by (a state is specified by listing the values of x, y, and z, in that order):

$$(0, 0, 0) \to (1, 0, 1) \to (1, 1, 0) \to (2, 1, 1) \to (2, 2, 0) \to \dots$$

The formula φ given by $(x = y \vee x = y+1)$ holds in every state of this execution, and is an invariant of the system. The formula φ, however, is not an inductive invariant. The state (1, 0, 0) satisfies the formula φ, and has a transition to the state (2, 0, 1), which does not satisfy the formula φ. Consider the formula ψ given by $(z = 0 \wedge x = y) \vee (z = 1 \wedge x = y + 1)$. Observe that if a state s satisfies ψ, it must satisfy one of the disjuncts in ψ, and thus, must satisfy either $(x = y)$ or $(x = y + 1)$, and thus, must satisfy φ. Thus, the property ψ is stronger than φ. The initial state (0, 0, 0) satisfies ψ. Consider a state s that satisfies ψ. Then s satisfies either $(z = 0 \wedge x = y)$ or $(z = 1 \wedge x = y + 1)$. In the former case, executing a transition from the state s increments x and sets z to 1, and thus, the resulting state satisfies $(z = 1 \wedge x = y + 1)$. By a similar reasoning if the state s satisfies $(z = 1 \wedge x = y + 1)$, then executing one transition from it leads to a state that satisfies $(z = 0 \wedge x = y)$. It follows that if there is a transition from the state s to state t, then the state t must satisfy ψ. Thus, the property ψ is an inductive invariant.

Exercise 4: Properties

(a) Consider transition system T with state variables a, b of type `bool`, and initial state s_0. The following set of executions

$$\{s_0 \to s_1 \to s_2 \to s_3 \cdots \mid \forall_i . s_i \models a \wedge b\}$$

describe the set of executions satisfying the property "Every state satisfies a and b". Using similar mathematical notation, describe the sets of executions satisfying the following properties:

(i) Every state satisfies a or b.

(ii) There is no state satisfying b before the first occurrence of a.

(iii) Every a will be eventually followed by an b.

(iv) Exactly three states satisfy a.

(v) If there are infinitely many a there are infinitely many b.

(vi) There are only finitely many a.

(b) Intuitively, violations (counter-examples) of safety properties are finite executions, whereas counter-examples of liveness properties are infinite executions. Which of the properties above are invariants, which are safety properties, and which are liveness properties?

Solution

(a) (i) $\{s_0 \to s_1 \to s_2 \to s_3 \cdots \mid \forall i.a \in s_i \models a \lor b\}$

(ii) $\{s_0 \to s_1 \to s_2 \to s_3 \cdots \mid \forall i.s_i \models b \Rightarrow \exists j \leq i.s_j \models a\}$

(iii) $\{s_0 \to s_1 \to s_2 \to s_3 \cdots \mid \forall i.s_i \models a \Rightarrow \exists j > i.s_j \models b\}$

(iv) $\{s_0 \to s_1 \to s_2 \to s_3 \cdots \mid \#\{i \mid s_i \models a\} = 3\}$

(v) $\{s_0 \to s_1 \to s_2 \to s_3 \cdots \mid (\forall i.\exists j > i.s_j \models a) \Rightarrow (\forall i.\exists j > i.s_j \models b)\}$

(vi) $\{s_0 \to s_1 \to s_2 \to s_3 \cdots \mid (\exists i.\forall j > i.s_j \not\models a)\}$

(b) Property (i) is an invariant and therefore a safety property. Property (ii) is a safety property. Property (iii,v,vi) are liveness properties. Property (iv) is neither safety nor liveness property. The property can be described as (3 or less states satisfy a) and (3 or more states satisfy a), the former is a safety property and the latter is a liveness property.

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

AALBORG UNIVERSITY
DENMARK

Discussion: 2. March 2022

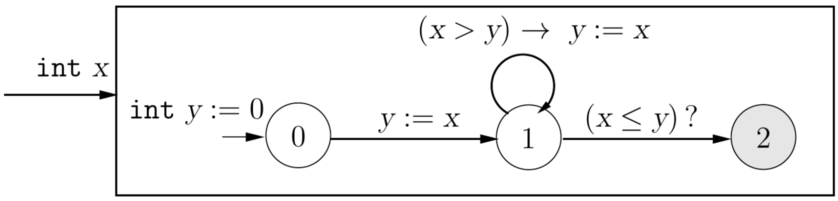
Models and Tools for Cyber-Physical Systems
Exercise sheet 5
WITH SOLUTIONS

Exercise 1: Monitor

Consider a component `C` with an output variable `x` of type int. Design a safety monitor to capture the requirement that the sequence of values output by the component `C` is strictly increasing (that is, the output in each round should be strictly greater than the output in the preceding round).

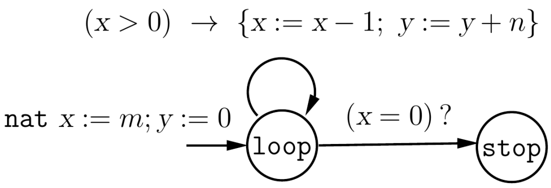
.... Solution ....

The monitor, shown below, maintains a state variable `y` to record the value of the input from the preceding round. The monitor enters the mode 2 exactly when the desired safety requirement gets violated.



Exercise 2: Symbolic transition system `Mult`

The following state machine is used to compute the multiplication of two natural numbers:



Describe this transition system symbolically using initialization and transition formulas.

.... Solution ....

The transition system `Mult(m,n)` has state variables x of type nat, y of type nat, and $mode$ of the enumerated type $\{loop, stop\}$. The initialization is given by the formula

$$(mode = loop) \wedge (x = m) \wedge (y = 0).$$

The transition formula is given as:

$$\begin{array}{ll} & [(mode = loop) \wedge (x > 0) \wedge (x' = x - 1) \wedge (y' = y + n) \wedge (mode' = loop)] \\ \vee & [(mode = loop) \wedge (x = 0) \wedge (x = x) \wedge (y = y) \wedge (mode = stop)] \end{array}$$

Exercise 3: Image computation

Consider the symbolic image computation for a transition system with two real-valued variables x and y and transition description given by the formula $x' = x + 1 \wedge y' = x$. Suppose the region A is described by the formula $0 \leq x \leq 4 \wedge y \leq 7$. Compute the formula describing the post-image of A.

.... Solution ....

Conjunction of the given region A and the transition formula gives

$$(x' = x + 1) \wedge (y' = x) \wedge (0 \leq x \leq 4) \wedge (y \leq 7)$$

The existential quantification of the unprimed variables leads to $(x' = y' + 1) \wedge (0 \leq y' \leq 4)$. Renaming the primed variables to their unprimed counterparts gives the desired post-image:

$$(x = y + 1) \wedge (0 \leq y \leq 4).$$

Exercise 4: Backward reachability

The symbolic breadth-first search algorithm (Chap 4, Slide 65) is a *forward search* algorithm that computes the set of states reachable from the initial states by repeatedly applying the image computation operator Post. The symbolic *pre-image* of A defined as:

$$\mathsf{Pre}(A, Trans) \stackrel{def}{=} \mathsf{Exists}(\mathsf{Conj}(\mathsf{Rename}(A, S, S'), Trans), S')$$

is the region over S that contains precisely those states s for which there is a transition (s, t) for some state t in A. Develop a *backward search* algorithm for the invariant verification problem that starts with the states that violate the desired invariant and computes the set of states that can reach the violating states by repeatedly applying the *pre-image* computation operator Pre.

.... Solution ....

The backward-search algorithm is symmetric to the algorithm in (Chap 4, Slide 65). The region Reach contains all the states from which a state satisfying the property φ has been discovered to be reachable. It initially contains the states that satisfy φ, and in each iteration, states from which there is a transition to a state already in Reach, are added using the pre-image computation. At any step, if the region Reach contains an initial state, the algorithm has discovered an execution from an initial state to a state satisfying φ, and can terminate.

Input: A transition system T given by a region $Init$ for initial states and a region $Trans$ for transitions, and a property φ.
Output: If φ is reachable in T, return 1, else return 0.

```
reg Reach := φ;
reg New := φ;
while IsEmpty(New) = 0 {
  if IsEmpty(Conj(New, Init)) = 0 then return 1;
  New := Diff(Pre(New, Trans), Reach);
  Reach := Disj(Reach, New);
}
return 0;
```

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

Discussion: 9. March 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 6
WITH SOLUTIONS

Exercise 1: Asynchronous process `Split`

We want to design an asynchronous process `Split` that is the dual of `Merge` (c.f. Figure 1 or Chapter 4, Slide 6). The process `Split` has one input channel `in` and two output channels `out1` and `out2` . The messages received on the input channel should be routed to one of the output channels in a nondeterministic manner so that all possible splittings of the input stream are feasible executions. Describe all the components of the desired process `Split`.

. . . . Solution . . . .

The asynchronous process `Split` has a single input variable `in` of type `msg`. Its output variables are $\mathtt{out}_1$ and $\mathtt{out}_2$ of type `msg`. It maintains a single queue as its state variable with the declaration given by

$$\mathtt{queue(msg)x} := \mathtt{null}.$$

The input task A_i specified by

$$\neg\mathtt{Full(x)} \rightarrow \mathtt{Enqueue(in, x)}$$

stores the messages arriving on the input channel `in` the queue `x`. The output task A_o^1 is enabled when the queue `x` is nonempty and if so, removes a message from the queue and transmits it on the output channel $\mathtt{out}_1$:

$$\neg\mathtt{Empty(x)} \rightarrow \mathtt{out}_1 := \mathtt{Dequeue(x)}.$$

The output task A_o^2 is symmetric, and transmits messages on the output channel $\mathtt{out}_2$:

$$\neg\mathtt{Empty(x)} \rightarrow \mathtt{out}_2 := \mathtt{Dequeue(x)}.$$

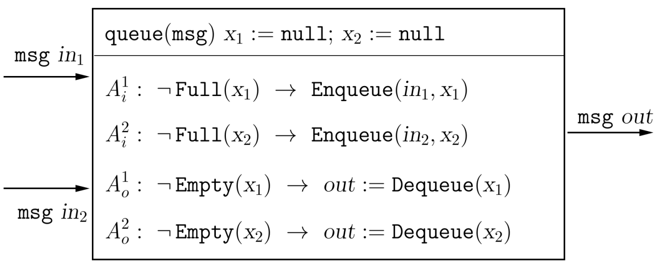


Fig. 1: Asynchronous process `Merge`

Note that a message stored in the queue x is transmitted on only one of the output channels, and the choice is nondeterministic.

Exercise 2: Asynchronous process composition
The asynchronous process `DoubleBuffer` is the result of the parallel composition of two `Buffer` components. In Chapter 4, Slide 17 we describe the "compiled" version the `DoubleBuffer` process. In the following, consider the asynchronous process

$$\mathtt{Merge}[\mathtt{out} \mapsto \mathtt{temp}] | \mathtt{Merge}[\mathtt{in}_1 \mapsto \mathtt{temp}][\mathtt{in}_2 \mapsto \mathtt{in}_3]$$

obtained by connecting two instances of the process `Merge` (c.f. Figure 1). Show the "compiled" version of this composite process similar to the description of `DoubleBuffer`. Explain the input/output behavior of this composite process.

.... Solution ....

The input channels are $\mathtt{in}_1$, $\mathtt{in}_2$, and $\mathtt{in}_3$, all of type msg. The output channel is out. When composing the two instances of Merge, we need to make sure that the state variables have distinct names. The state variables of the composite process and their initialization is specified by

$$\mathtt{queue}(\mathtt{msg})\mathtt{x}_1 := \mathtt{null}; \mathtt{x}_2 := \mathtt{null}; \mathtt{y}_1 := \mathtt{null}; \mathtt{y}_2 := \mathtt{null}.$$

The composite process has three input tasks corresponding to its three input channels specified by:

$$\begin{array}{lcl} A_i^1 & : & \neg\mathtt{Full}(\mathtt{x}_1) \to \mathtt{Enqueue}(\mathtt{in}_1, \mathtt{x}_1) \\ A_i^2 & : & \neg\mathtt{Full}(\mathtt{x}_2) \to \mathtt{Enqueue}(\mathtt{in}_2, \mathtt{x}_2) \\ A_i^3 & : & \neg\mathtt{Full}(\mathtt{y}_2) \to \mathtt{Enqueue}(\mathtt{in}_3, \mathtt{y}_2) \end{array}$$

The composition has two internal tasks, each of which is obtained by synchronizing an output of the first instance on the channel temp with a corresponding input processing by the second:

$$\begin{array}{lcl} A^1 & : & \neg\mathtt{Empty}(\mathtt{x}_1) \wedge \neg\mathtt{Full}(\mathtt{y}_1) \to \\ & & \{\mathtt{local\ msg\ temp} := \mathtt{Dequeue}(\mathtt{x}_1);\ \mathtt{Enqueue}(\mathtt{temp}, \mathtt{y}_1)\} \\ A^2 & : & \neg\mathtt{Empty}(\mathtt{x}_2) \wedge \neg\mathtt{Full}(\mathtt{y}_1) \to \\ & & \{\mathtt{local\ msg\ temp} := \mathtt{Dequeue}(\mathtt{x}_2);\ \mathtt{Enqueue}(\mathtt{temp}, \mathtt{y}_1)\} \end{array}$$

Finally, the composite process has two output tasks that remove messages from the queues $\mathtt{y}_1$ and $\mathtt{y}_2$ in order to transmit them on the output channel out:

$$\begin{array}{lcl} A_o^1 & : & \neg\mathtt{Empty}(\mathtt{y}_1) \to \mathtt{out} := \mathtt{Dequeue}(\mathtt{y}_1); \\ A_o^2 & : & \neg\mathtt{Empty}(\mathtt{y}_2) \to \mathtt{out} := \mathtt{Dequeue}(\mathtt{y}_2); \end{array}$$

The sequence of values output by the composite process represents a merge of the sequences of input values received on the three input channels. The relative order of values received on each of the input channels is preserved in the output sequence, but the three input sequences can be interleaved in any nondeterministic order.

Exercise 3: Peterson's mutual exclusion protocol (Uppaal)
Download the Peterson's mutual exclusion protocol Uppaal model (petersons.xml) from MS teams. The model includes the component `UnfairScheduler` which emulates the scheduler of an operating system.

- Verify if the protocol satisfy the mutual exclusion property:

$$\forall\Box\neg(\texttt{P1.Crit} \wedge \texttt{P2.Crit})$$

- Verify the non-starvation property: if a process wants to enter its critical section then it eventually does

$$\texttt{P1.Try1} -- > \texttt{P1.Crit}$$

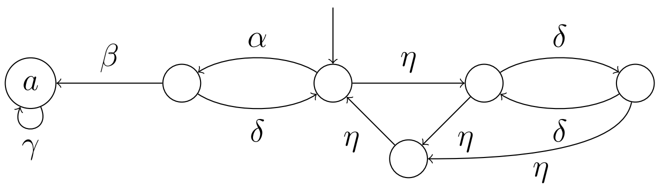
- Build a component `FairScheduler` such that the above properties are satisfied.

. . . . Solution . . . .

Download from MS teams.

Exercise 4: Fairness assumptions

Consider the following extended state machine with tasks $\{\alpha, \beta, \gamma, \delta, \eta\}$:



Under which fairness assumptions does the system satisfy the property "for all executions, eventually a"? Justify your answer.

(a) Infinitely often γ

(b) Infinitely often α and γ

(c) Infinitely often α or γ

(d) Strong-fairness for β

(e) Strong-fairness for α and β.

(f) Strong-fairness for α, β and η.

(g) Weak-fairness for α, β and η.

(h) Strong-fairness for α, β and Weak-fairness for η.

. . . . Solution . . . .

(a) Holds: demands that γ occurs infinitely often, so it occurs at least once. Therefore, the left state must be reached eventually.

(b) Holds: demands that γ occurs infinitely often. A closer inspection yields that there is no fair run. However, this means that the property is vacuously true.

(c) Does not hold: A fair run is the α-δ-loop, since only one of α and γ has to occur infinitely often.

(d) Does not hold: The η-loop is fair, since β is not infinitely often enabled.

(e) Does not hold: The right-most δ-loop is fair, since neither α nor β are infinitely often enabled.

(f) Holds: A fair run cannot stay in the right most loop, since η is infinitely often enabled. Therefore η must be infinitely often taken, so the starting state is visited infinitely often. Then α is infinitely often enabled; it is therefore infinitely often taken and the second state from the left is visited infinitely often. Therefore β has to be taken.

(g) Does not hold: The η-loop is fair. Since α is infinitely often not enabled, it does not have to be taken.

(h) Holds: The δ-loop is not fair, because η is always enabled in that loop. Therefore η has to be taken and α is infinitely often enabled. Thus as in an earlier case, α and therefore β have to be taken.

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

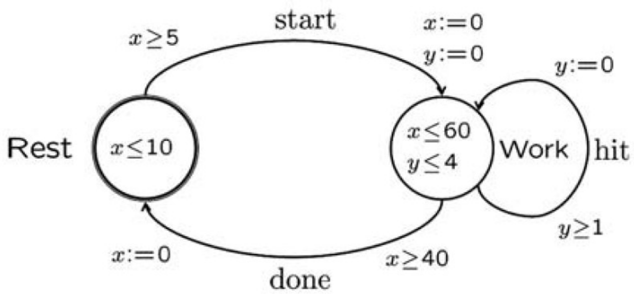
AALBORG UNIVERSITY
DENMARK

Discussion: 16. March 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 7
WITH SOLUTIONS

Exercise 1: Worker Process

Consider the timed automata below. Given that the Worker works for 70 minutes, what is the maximum and minimum number of hits if (s)he (a) starts with a rest(i.e. starts in the state (Rest, $x = 0, y = 0$) or (b) is immediately ready to work (i.e. starts in the state (Work, $x = 0, y = 0$). Provide valid transition sequences of the Worker that prove your claims.



Solution

a)

Starting in state $(Rest, x = 0, y = 0)$, we have:

min = 11 hits

$(Rest, x = 0, y = 0) \xrightarrow{10} (Rest, x = 10, y = 10) \xrightarrow{start}$
$(Work, x = 0, y = 0) \xrightarrow{4} (Work, x = 4, y = 4) \xrightarrow{\textsf{hit}}$
$(Work, x = 4, y = 0) \xrightarrow{4} (Work, x = 8, y = 4) \xrightarrow{\textsf{hit}}$
$\cdots \xrightarrow{\textsf{hit}} \times 6$
$(Work, x = 32, y = 0) \xrightarrow{4} (Work, x = 36, y = 4) \xrightarrow{\textsf{hit}}$
$(Work, x = 36, y = 0) \xrightarrow{4} (Work, x = 40, y = 4) \xrightarrow{done}$
$(Rest, x = 0, y = 4) \xrightarrow{10} (Rest, x = 10, y = 14) \xrightarrow{start}$
$(Work, x = 0, y = 0) \xrightarrow{4} (Work, x = 4, y = 4) \xrightarrow{\textsf{hit}}$
$(Work, x = 8, y = 0) \xrightarrow{4} (Work, x = 12, y = 4) \xrightarrow{\textsf{hit}}$

$(Work, x = 12, y = 0) \xrightarrow{2} (Work, x = 14, y = 2)$

max = 60 hits
$(Rest, x = 0, y = 0) \xrightarrow{5} (Rest, x = 5, y = 5) \xrightarrow{start}$
$(Work, x = 0, y = 0) \xrightarrow{1} (Work, x = 1, y = 1) \xrightarrow{\textsf{hit}}$
$(Work, x = 1, y = 0) \xrightarrow{1} (Work, x = 2, y = 1) \xrightarrow{\textsf{hit}}$
$\cdots \xrightarrow{\textsf{hit}} \times 57$
$(Work, x = 59, y = 0) \xrightarrow{1} (Work, x = 60, y = 1) \xrightarrow{\textsf{hit}}$
$(Work, x = 60, y = 0) \xrightarrow{done}$
$(Rest, x = 0, y = 1) \xrightarrow{5} (Rest, x = 5, y = 6)$

b)

Starting in state $(Work, x = 0, y = 0)$, we have:

min = 13 hits
$(Work, x = 0, y = 0) \xrightarrow{4} (Work, x = 4, y = 4) \xrightarrow{\textsf{hit}}$
$(Work, x = 4, y = 0) \xrightarrow{4} (Work, x = 8, y = 4) \xrightarrow{\textsf{hit}}$
$\cdots \xrightarrow{\textsf{hit}} \times 6$
$(Work, x = 32, y = 0) \xrightarrow{4} (Work, x = 36, y = 4) \xrightarrow{\textsf{hit}}$
$(Work, x = 36, y = 0) \xrightarrow{4} (Work, x = 40, y = 4) \xrightarrow{done}$
$(Rest, x = 0, y = 4) \xrightarrow{10} (Rest, x = 10, y = 14) \xrightarrow{start}$
$(Work, x = 0, y = 0) \xrightarrow{4} (Work, x = 4, y = 4) \xrightarrow{\textsf{hit}}$
$\cdots \xrightarrow{\textsf{hit}} \times 3$
$(Work, x = 20, y = 0) \xrightarrow{4} (Work, x = 24, y = 4)$

max = 65 hits
$(Work, x = 0, y = 0) \xrightarrow{1} (Work, x = 1, y = 1) \xrightarrow{\textsf{hit}}$
$(Work, x = 1, y = 0) \xrightarrow{1} (Work, x = 2, y = 1) \xrightarrow{\textsf{hit}}$
$\cdots \xrightarrow{\textsf{hit}} \times 57$
$(Work, x = 59, y = 0) \xrightarrow{1} (Work, x = 60, y = 1) \xrightarrow{\textsf{hit}}$
$(Work, x = 60, y = 0) \xrightarrow{done}$
$(Rest, x = 0, y = 1) \xrightarrow{5} (Rest, x = 5, y = 6) \xrightarrow{start}$
$(Work, x = 0, y = 0) \xrightarrow{1} (Work, x = 1, y = 1) \xrightarrow{\textsf{hit}}$
$\cdots \xrightarrow{\textsf{hit}} \times 4$
$(Work, x = 5, y = 0)$

Exercise 2: Alarm

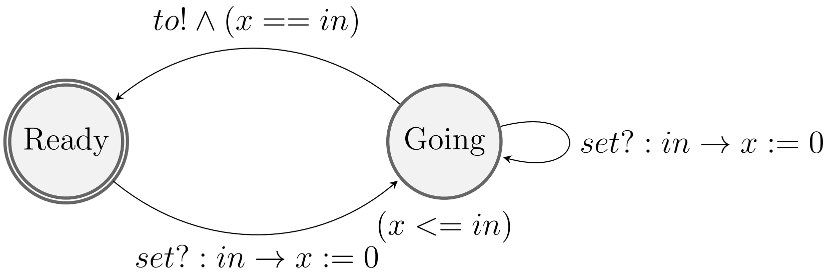
An *Alarm Timer* is a timed process which can be set to time-out after a prescribed time period has elapsed. You are asked to provide a timed automata model of an alarm timer T, which can be set to time-out after 5, 10 and 30 minutes by discrete input actions $set5, set10$ and $set30$. After the prescribed time period T signals the time-out by the the output action *to*. It is required that the alarm timer can be reset with a new time-out period at any given moment, in particular before the previously set time period has elapsed.

. . . . Solution . . . .

There are multiple ways to solve this problem, in this solution we try to minimize the amount of locations we need to model the Alarm as a Timed Automaton.

Observe that the three discrete actions exert the exact same behavior, meaning that we can use a single action to capture all three. We utilize one state variable $x : clock$, input values (5, 10 or 30) are assigned to *in* and received on channel *set*. Output is transmitted on channel *to*.

Model



Exercise 3: Time-Constrained Increments ***Optional – for the keen***

Consider the timed process `TimedInc` with parallel increments shown below. Argue that both the properties $(x_1 \leq 2x_2 + 2)$ and $(x_2 \leq 2x_1 + 2)$ are invariants of the system. Hint: you will need a stronger property to prove the invariance, e.g. including as one of 4 conjuncts $(y_1 = 0 \wedge y_2 \leq 1) \implies (x_1 \leq 2x_2 \wedge x_2 \leq 2x_1 + 2)$.

`nat` $\mathrm{x}_1 := 0;\ \mathrm{x}_2 := 0$
`clock` $y_1 := 0;\ y_2 := 0$
$CI:\ (y_1 \leq 2) \wedge (y_2 \leq 2)$
$A_1:\ (y_1 \geq 1) \rightarrow \{\mathrm{x}_1 := \mathrm{x}_1 + 1;\ y_1 := 0\}$
$A_2:\ (y_2 \geq 1) \rightarrow \{\mathrm{x}_2 := \mathrm{x}_2 + 1;\ y_2 := 0\}$

You may also try to model `TimedInc` in UPPAAL and verify the invariant directly (say within some bounds of x_1 and x_2).

. . . . Solution . . . .

You may find can find the UPPAAL model TimedIncSol.xml in MS teams.

Here is a sketch of a solution.

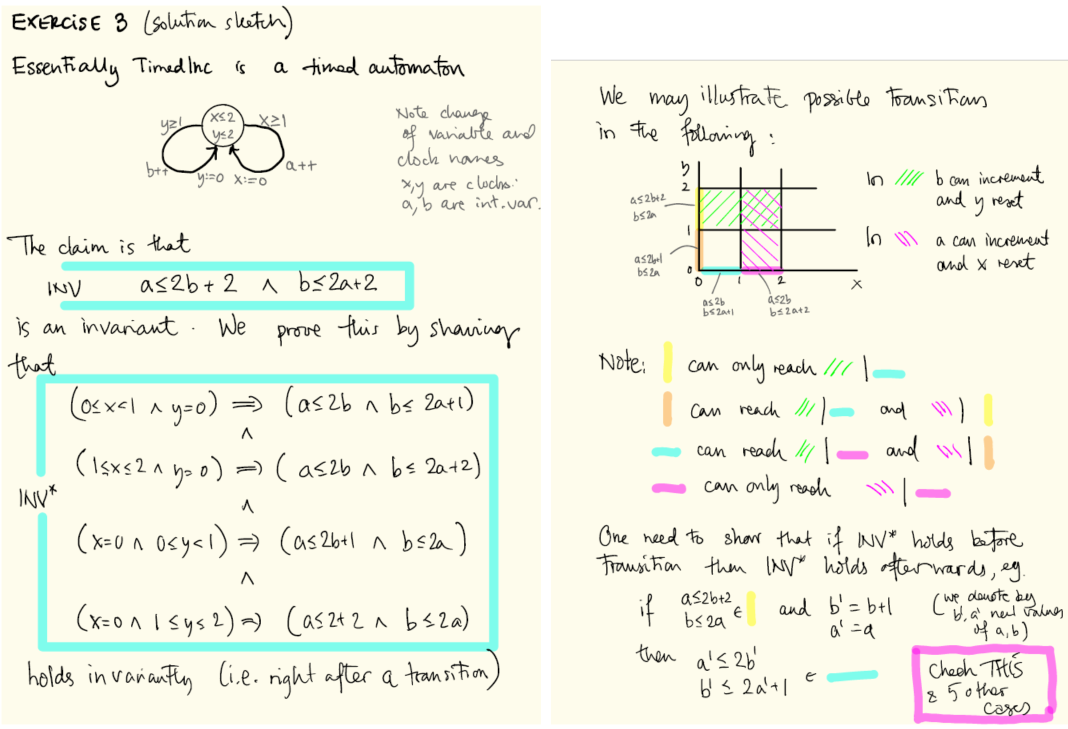


Abbildung 1: Solution to Ex 3

Exercise 4: Intelligent Light Control

Download the UPPAAL model of the Intelligent Light Controller (IntelligentLight.xml) from MS teams. Experiments using the simulator of UPPAAL with different settings of the clock-invariants in the locations `Rest` and `Busy` of the `User`. In particular set the clock-invariants so that the time-out transitions of the `Controller` can be taken. How will you verify that this is the case?

. . . . Solution . . . .

You obviously need to change the invariant of the `User` in location `Rest` so that one can delay more the 100 time-units there, e.g. change to `y<=150`.

Exercise 5: Fishers Protocol

Download the UPPAAL model of Fischers Protocol (FischersProtocol.xml) from MS teams and examine it using the simulator. In addition use verification to check whether the following properties are satisfied or not:

- Does the protocol satisfies the mutual exclusion property:

$$\forall\Box\neg(\texttt{Process}(1).\texttt{Crit} \wedge \texttt{Process}(2).\texttt{Crit})$$

- Does the protocol satisfies the non-starvation property: if a process wants to enter its critical section then it eventually does

$$\texttt{Process}(1).\texttt{Test} - - > \texttt{Process}(1).\texttt{Crit}$$

- Is the protocol deadlock-free, i.e.

$$\forall\Box\neg\text{deadlock}$$

Perform verification of the above properties for varying values of the constants `D1` and `D2`. In case the properties does not hold, try to obtain a counter-example. Also, perform the verifications for varying numbers of processes.

. . . . Solution . . . .

The first and the third property is satisfied. The second (liveness) property fails to hold for trivial reasons. In fact, it is possible to delay forever in the location-combination $(\texttt{Process}(1).\texttt{Test}, \texttt{Process}(2).\texttt{Test})$.

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

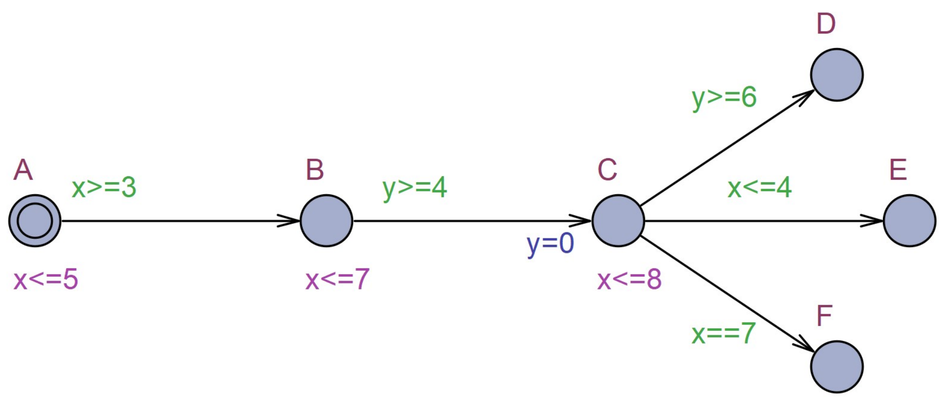
AALBORG UNIVERSITY
DENMARK

Discussion: 23. March 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 8
WITH SOLUTIONS

Exercise 1: Symbolic Exploration of Timed Automata

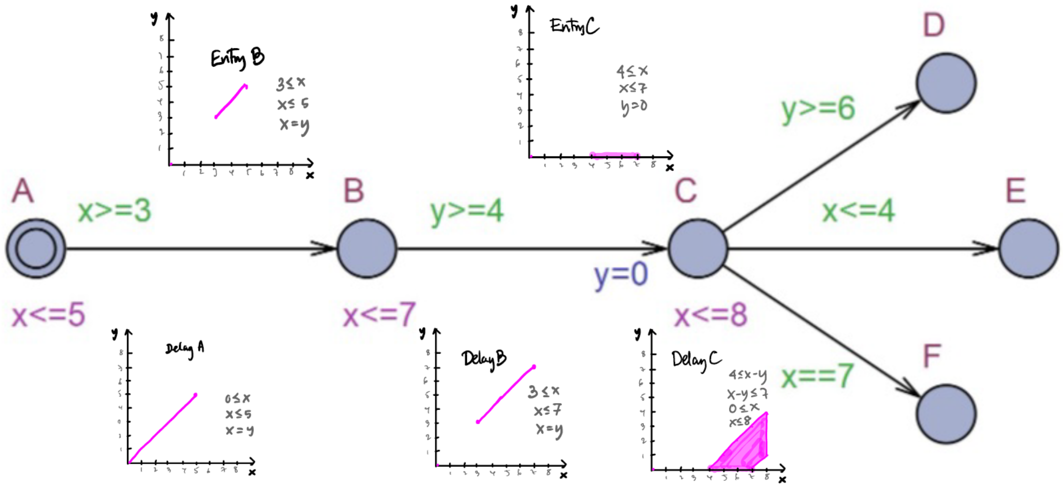
Consider the timed automaton $\mathcal{A}$ below with two clocks `x` and `y`. Perform a symbolic zone-base reachability analysis of $\mathcal{A}$ as in Lecture 6 (slides 37-44). Which of the locations `D`, `E` and `F` are reachable. Describe using difference constraints the reachable zones upon entry and after delay for the locations `A`, `B` and `C`.



You can find a Uppaal model of $\mathcal{A}$ in SimpleTA.xml in MS teams.

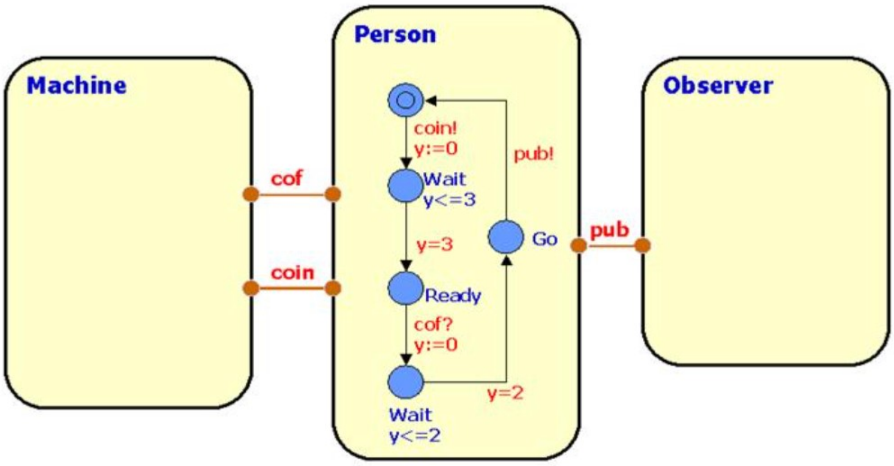
Solution

Locations E and F are reachable, whereas locations D is not reachable. The symbolic zone-based reachability of $\mathcal{A}$ is given below:



Exercise 2: Coffee Machine

In this exercise you are asked to design the control of a `Machine` (the control program) which will serve a coffee craving `Person` (the environment). As you can see below the person repeatedly (tries to) insert a `coin`, (tries to) extract `cof`fee after which (s)he will make a `pub`lication. Between each action the `Person` requires a suitable time-delay before being ready to participate in the next one.



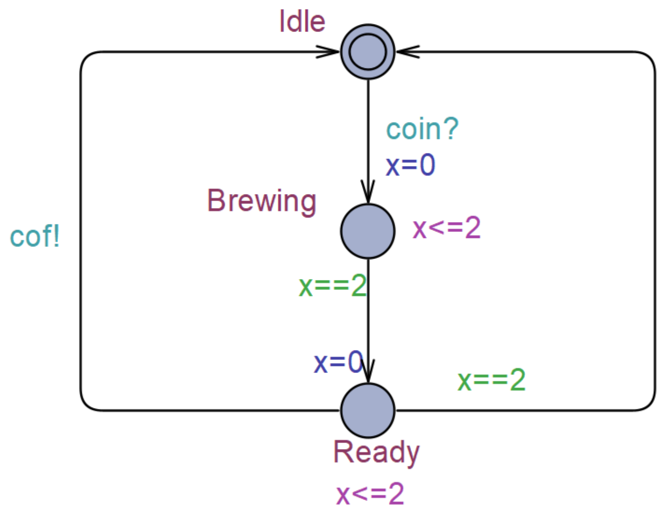
The `Machine` takes some time for brewing the coffee and will time-out if coffee has not been taken before a certain upper time-limit.

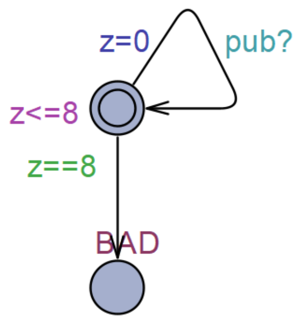
As a requirement we want the overall behaviour to ensure that the indicated Observer experiences a constant flow of publications from the system. In particular we want the `Observer` to complain if at any time more than 8 time-units elapses between two consecutive publications. Model the `Machine` and `Observer` in UPPAAL and analyse the behaviour of the system. Try to determine the maximum brewing time allowed by the Machine in order that the above requirement is met.

An initial sketch of a UPPAAL can be found in CoffeeMachine.xml in MS teams.

. . . . Solution . . . .

Below find proposal for `Machine` and `Observer`. In the suggested solution the brewing time of the `Machine` is 2 time-units.





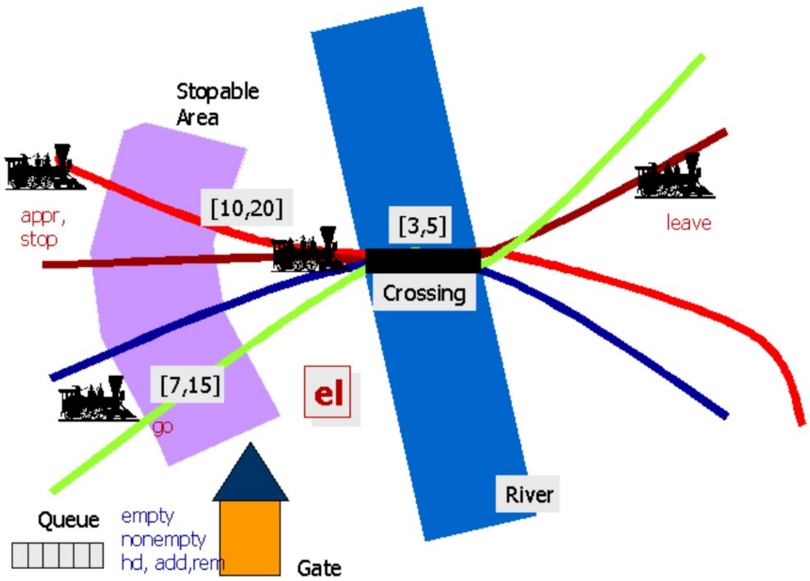
To verify that the `Observer` does not complain we check the following UPPAAL query

```
A[] ! Observer.BAD
```

Now reverifying the above query with changing brewing times pf the `Machine` we find that the maximum brewing where the `Observer` does not complain is 5 time-units.
You may find a complete solution in CoffeeMachineSol.xml in MS teams.

Exercise 3: Train-Gate Error Correction

Below is the overview of the `TrainGate` model. An erroneous UPPAAL model of this case may be found in train-gate-err.xml in MS teams and in train-gate-err.q you will find the properties.



Please do the following

(a) Use UPPAAL (simulation, verification, or whatever you want) to pin-point and explain the error(s).

(b) Having corrected the errors what is the minimum time one can guarantee between a train requesting access to the crossing and actually getting there?

.... Solution ....

(a) There are two errors both in the `Gate` component. 1) In the bottom-most location time can elapse arbitrarily – that is there is no time-bound between a train signals that it is approaching and it will be stopped. You should make the location committed. 2) The function for returning the front element has an index-error. The front element is `list[0]` and not `list[1]`.

(b) The minimum time is 80. You can find this by inserting a new clock `z` to the `Train` template and verifying queries of the type `Train(0).Appr --> (Train(0).Cross && Train(0).z<=B)` with different values of `B`. You can use binary search to find the smallest value of `B` that will make the query true.

Exercise 4: Job Shop Scheduling

Jobshop scheduling problems involves a number of jobs $J_0, \ldots, J_n$ to be treated by a number of machines $M_0, \ldots, M_m$. In this particular variant each job J_i needs to be treated exactly once by each machine M_j and in a particular order and for a particular time-duration. The problem is to find an time-assignment (a schedule) indicating at which time J_i should be treated by M_j in a way such that the preferred sequence of J_i is respected and so that a machine M_j is never used by more than one job at a time. The challenge is to identify the minimum Makespan, i.e. time-assignment which the minimal total duration.

	Sport	Economy	Local News	Comic Stip
Kim	2. 5 min	4. 1 min	3. 3 min	1. 10 min
Jüri	1. 10 min	2. 20 min	3. 1 min	4. 1 min
Jan	4. 1 min	1. 13 min	3. 11 min	2. 11 min
Wang	1. 1 min	2. 1 min	3. 1 min	4. 1 min

Consider the particular Jobshop scheduling problem above, where 4 persons (Kim, Juri, Wang and Jan = the Jobs) needs to read 4 sections (Sport, Economy, Local News, Comic Strip = the Machines) of a single news paper. The timing in minutes as well as the preferred number in the reading sequence for each person in indicated in the cells.
You may use the UPPAAL model Jobshop.xml in MS teams of this instance of the jobshop.

(a) Use the model checking engine of UPPAAL to determine the minimum make-span, i.e. the minimum time until all persons has completed reading all sections.

(b) Identify also the minimum time for Kim to finish reading his sections, and the minimum time for Wang to finish his sections.

(c) What is the minimum time to have both Kim and Jan finishing reading their sections. Try to illustrate using informative Gantt charts.

. . . . Solution . . . .

(a) The minimum make-span is 37. You can find this either by posing the UPPAAL query

```
E<> Kim.Done && Juri.Done && Jan.Done && Wang.Done
```

with diagnostic trace option "Fastest" activated (afterwards you find the time in the simulator). Alternatively you may search for the smallest value of `B` that will make the query `E<> Kim.Done && Juri.Done && Jan.Done && Wang.Done && time<=B` satisfied.

(b) Answer for `Kim` is 19 and for `Wang` is 4. Same technique to obtain this as above.

(c) Minimum time for both Kim and Jan to finish is 36 – so actually only 1 minute faster than the minimum time for having every one finished reading the newspaper.

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

AALBORG UNIVERSITY
DENMARK

Discussion: 13th April 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 9

Exercise 1: MATLAB Lecture
Download and install MATLAB by following the instructions on

`https://www.ekstranet.its.aau.dk/software/mathworks`

MATLAB is a simple language whose strength comes from its rich set of mathematical routines. It can be quickly picked up by someone custom to C or Java. To get started, watch the following series of tutorials. These comprise the preliminary/zero lecture of the third part of the course:

- `https://www.youtube.com/watch?v=upD2lwX2bS0`
- `https://www.youtube.com/watch?v=3HnyHcy7Dnc`
- `https://www.youtube.com/watch?v=8rdU49tBExg`
- `https://www.youtube.com/watch?v=Gf89qyP0BxY`
- `https://www.youtube.com/watch?v=7wa0ZM_cq2g`
  (As SW students, you should find this one particulary easy)

Exercise 2: MATLAB Warm-up

(a) Consider the following matrix and vector:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \qquad\qquad b = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

Solve the system of linear equations $Ax = b$ by hand using your favorite method.

(b) Create a new MATLAB function `ex9.m`. Afterwards, read the manual of `mldivide` (`doc mldivide`) and solve $Ax = b$ using it. Print the solution.

(c) Double check the obtained solution x by computing Ax by hand and in MATLAB.

(d) Add to `ex9.m` the function `myFun` that takes as input n, A, b and computes, using a for loop, the solutions of $Ax = \binom{5}{6+i}$ for all $0 \leq i \leq n$. Call `myFun` with $n = 3$ and make sure to print once again the solutions.

(e) Solve the foregoing exercise without using any loops. Instead, apply `mldivide` on A and a suitably chosen matrix B. To create B, use matrix commands seen in the tutorials. Recall in particular that:

- `3 .* B` multiplies all entries of a matrix by 3 (note that `.*` signifies elementwise multiplication, while `*` refers to matrix multiplication).
- `ones(n,m)` creates an $n \times m$ matrix filled with ones
- `zeros(n,m)` creates an $n \times m$ matrix filled with zeros
- `x:s:y` creates a row vector of numbers from x to y with a step size of s

Exercise 3: Matrices and Recursion

(a) Recall that the determinant of any 2×2 matrix

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{pmatrix}$$

can be computed via the formula $\det(B) = b_{1,1}b_{2,2} - b_{1,2}b_{2,1}$. Next, consider an arbitrary matrix $B \in \mathbb{R}^{n\times n}$:

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,n-1} & b_{1,n} \\ b_{2,1} & b_{2,2} & \dots & b_{2,n-1} & b_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{n,1} & b_{n,2} & \dots & b_{n,n-1} & b_{n,n} \end{pmatrix}$$

Extend `ex9.m` by the function `Lap(B,b)` that computes the determinant of B times b, that is, $b \cdot \det(B)$. To this end, use the following recursive formula for determinants:

$$\det(B) = \sum_{j=1}^{n} \texttt{Lap}(M_{1,j}, (-1)^{1+j} b_{1,j}),$$

where $M_{1,j}$ arises from B by removing the first row and the j-th column from B.

- Instead of computing $M_{1,j}$ in the naive way, make use of MATLAB's matrix operations:
  - `B([i:j],:)` gives the submatrix of B consisting of rows $i, \dots, j$
  - `B(:,[i:j])` gives the submatrix of B consisting of columns $i, \dots, j$ of B
  - The matrix operations can be used to read and write. For instance, the code `x = B(:,j)` copies the j-th column of B into x. Instead, the code `B(:,j) = []` deletes the j-th column in B.
- The above recursion formula is known as Laplace expansion, further details and examples can be found at:

https://en.wikipedia.org/wiki/Laplace_expansion

(b) Use `Lap` to compute the determinant of

$$B = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Cross check with the built-in command `det`.

(c) Test `Lap` on further matrices. Use to this end `randi` to generate random integer matrices. Limit yourself to small matrices with entries between 0 and 10. What can be said about the efficiency of `Lap` when the matrix dimension increases? Make a forecast before running the code.

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

AALBORG UNIVERSITY
DENMARK

Discussion: 20th April 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 10
WITH SOLUTIONS

Exercise 1: Car model

Consider the car model from the course

$$\frac{d}{dt}x(t) = v(t) \qquad\qquad \frac{d}{dt}v(t) = -\frac{k}{m}v(t) + \frac{1}{m}F,$$

where x is the position, v the velocity and F the applied force to a car of mass m with friction coefficient k.

(a) Assuming that a constant force F is applied, the velocity of the car will converge to steady-state velocity. Provide a formula for the steady-state velocity by assuming that the value of F is known.

(b) From now on, we assume that $F = 500$ (N), $m = 1000$ (kg), $k = 50$ $(\frac{m\,kg}{s^2})$, that we have no feedforward control (i.e., $D = 0$) and that we measure only the velocity of the car. Provide a control system using the (A, B, C, D) notation.

(c) Using the (A, B, C, D) notation and the initial condition $(x(0), v(0)) = (0, 0)^T$, compute and plot the numeric solution via the MATLAB commands `ss` and `step`.

(d) Assuming that a steady-state velocity of $v = 20$ $(\frac{m}{s})$ is desired, compute the corresponding constant force F. Adjust the (A, B, C, D) quadruple and plot the underlying numeric solution using the MATLAB commands `ss` and `step`.

.... Solution ....

(a) To obtain the velocity attained by a car after some time, we set $\dot{v} = 0$, which is equivalent to $0 = -\frac{k}{m}v + \frac{1}{m}F$. This, in turn, implies $v = \frac{F}{k}$. Note that $\dot{x} \neq 0$, i.e., while the velocity reached an equilibrium, the overall system did not because the car is moving at constant speed.

(b) It can be noted that

$$\begin{pmatrix} \frac{d}{dt}x \\ \frac{d}{dt}v \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & -\frac{1}{20} \end{pmatrix}}_{A} \cdot \begin{pmatrix} x \\ v \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{1000} \end{pmatrix}}_{B} \cdot F \qquad y = \underbrace{\begin{pmatrix} 0 & 1 \end{pmatrix}}_{C} \cdot \begin{pmatrix} x \\ v \end{pmatrix}$$

(c) A possible Matlab code is:

Listing 1: Matlab code

```
% (A,B,C,D) representation
A = [ [ 0 1]; [0 -0.05] ];
B = [0; 0.001];
C = [ 0 1 ];
D = 0;
% Create closed loop system. Multiply by 500 to account for
    the fact that we apply later a step input (so 1, rather
    than 500).
sys = ss(A,B*500,C,D)
% Solve numerically and plot
step(sys)
```

(d) Rearranging $v = \frac{F}{k}$ from (a) yields $F = vk$. With this, $v = 20$ ($\frac{m}{s}$) and $k = 50$ ($\frac{m\,kg}{s^2}$), we obtain $F = 1000$ (N). A possible Matlab code is thus

Listing 2: Matlab code

```
% (A,B,C,D) representation
A = [ [ 0 1]; [0 -0.05] ];
B = [0; 0.001];
C = [ 0 1 ];
D = 0;
% Create closed loop system. Multiply by 1000 to account for
    the fact that we apply later a step input (so 1, rather
    than 1000).
sys = ss(A,B*1000,C,D)
% Solve numerically and plot
step(sys)
```

Exercise 2: Lipschitz continuity

(a) Consider the differential equation $\frac{d}{dt}x(t) = g(x(t))$ with $g(x) = x^2 - x$. Using the MATLAB commands `ode45s` and `plot`, compute and plot the numeric solution of $\frac{d}{dt}x(t) = g(x(t))$ in the case of initial value $x(0) = 2.0$ on time intervals $[0; 0.67]$, $[0; 0.68]$, $[0; 0.69]$ and $[0; 0.70]$. Can you explain what happens?

(b) To understand better what happens in (a), consider a function $f : [a; b] \to \mathbb{R}$ that is differentiable and recall that the derivative (i.e., slope) of f at x, denoted by $\frac{d}{dx}f(x)$, is given by

$$\frac{d}{dx}f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

The mean-value theorem from calculus postulates that for any two $y_1, y_2 \in [a; b]$, there exists a x with $y_1 < x < y_2$ such that $f(y_2) - f(y_1) = \frac{d}{dx}f(x)(y_2 - y_1)$.

Using the above observation, compute the Lipschitz constant L_c of $g(x) = x^2 - x$ when $x \in [-c; c]$. That is, provide an $L_c < \infty$ that satisfies $|g(x_1) - g(x_2)| \leq L_c|x_1 - x_2|$ for all $x_1, x_2 \in [-c; c]$. What can be said about L_c if $c \to \infty$?

. . . . Solution . . . .

(a) Create a Matlab file called `odeExample.m` with the content below (file names are important because the file name identifies the main function of a file). Once done, run the file.

Listing 3: Matlab code

```
% main function
function odeExample()
    % finite time horizon
    T = 0.67; % try also 0.68, 0.69 and 0.70
    % initial condition
    x0 = [2.0];
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    % invocation of numeric ODE solver
    [I,x]=ode45(@odeDrift,I,x0);
    % plot the vector/matrix
    plot(I(:),x(:,1));
end

% auxiliary function describing the ODE
function dx = odeDrift(t,x)
  dx=zeros(1,1);
  dx(1) = x(1)*x(1) - x(1);
end
```

(b) By relying on the mean-value theorem, it suffices to consider the derivative of $g(x) = x^2 - x$ which is $\frac{d}{dx}g(x) = 2x - 1$. Specifically, on $[-c; c]$, the Lipschitz constant L_c is

$$\max_{x \in [-c;c]} |2x - 1| = 2c + 1$$

This shows that the slopes of solutions of $\frac{d}{dt}x(t) = g(x(t))$ are not bounded on the entire $\mathbb{R}$. Hence, pathological behavior as observed in (a) can, at least in principle, happen.

(For those who are curios: One says that $\frac{d}{dx}x(t) = g(x(t))$ has a finite explosion time for $x(0) = 2$. Note however that there is no finite explosion when $x(0) = 0.5$. That is, the fact that slopes cannot be bounded does not necessarily imply a finite explosion time.)

Exercise 3: Eigenvalues and -vectors

(a) Calculate the eigenvalues and eigenvectors of

$$A = \begin{pmatrix} 4 & 6 \\ 1 & 3 \end{pmatrix}$$

(b) Are the eigenvectors of A linearly independent?

(c) Double-check your results by using the MATLAB commands `eig` and `rank`.

.... Solution ....

(a) Recalling that the eigenvalues of A are the roots of the characteristic polynomial of A, we first need to compute the characteristic polynomial of A. This is given by the matrix determinant

$$\begin{aligned} |A - \lambda I| &= \begin{vmatrix} 4-\lambda & 6 \\ 1 & 3-\lambda \end{vmatrix} \\ &= (4-\lambda)(3-\lambda) - 1 \cdot 6 \\ &= 12 - 4\lambda - 3\lambda + \lambda^2 - 6 \\ &= \lambda^2 - 7\lambda + 6 \end{aligned}$$

Using the quadratic formula from high school (or the Matlab command `roots([1,-7,6])`), we obtain the roots $\lambda^{(1)} = 1$ and $\lambda^{(2)} = 6$. To obtain the eigenvector of $\lambda^{(i)}$, we have to solve the linear system of equations $Ax^{(i)} = \lambda^{(i)} x^{(i)}$, that is

$$\begin{aligned} 4x_1^{(i)} + 6x_2^{(i)} &= \lambda^{(i)} x_1^{(i)} \\ 1x_1^{(i)} + 3x_2^{(i)} &= \lambda^{(i)} x_2^{(i)} \end{aligned}$$

(b) Since the eigenvectors of different eigenvalues are always linearly independent, $x^{(1)}$ and $x^{(2)}$ must be linearly independent. This is readily checked by confirming that there exists no $\beta \in \mathbb{R}$ such that $\beta x^{(1)} = x^{(2)}$.

(c) A possible Matlab code is

Listing 4: Matlab code

```
A = [ [ 4 6]; [1 3] ];
[V,D] = eig(A)
rank(V)
```

Frederik Meyer Bønneland
Jonas Hansen
Kim G. Larsen
Marco Muñiz
Max Tschaikowski

AALBORG UNIVERSITY
DENMARK

Discussion: 27th April 2022

Models and Tools for Cyber-Physical Systems
Exercise sheet 11 / Second Mini Project
WITH SOLUTIONS

Exercise 1: Parking a Car in MATLAB
Consider the car model from the course

$$\frac{d}{dt}x(t) = v(t) \qquad\qquad \frac{d}{dt}v(t) = -\frac{k}{m}v(t) + \frac{1}{m}F(t),$$

where x is the position, v the velocity and F the force applied in case of mass m and friction coefficient k. Starting at position $x(0) = -100$, we wish to bring the car to the point 0. To this end, we construct a gain matrix K for $m = 1000$ (kg) and $k = 50$.

(a) (Optional) Watch the following great video of Brian Douglas:

https://www.youtube.com/watch?v=FXSpHy8LvmY

(b) Assuming that we have no feedforward control (i.e., $D = 0$) and measure only the position of the car, provide a control system for the parking model using the (A, B, C, D) notation.

(c) Compute by hand the gain matrix K that ensures that $A - BK$ has poles/eigenvalues -1 and -2.

(d) Cross-check the gain matrix by computing it using the MATLAB command `place`. Compute afterwards a numeric solution using the commands `ss` and `initial`. Use as initial condition $(x(0), v(0)) = (-100, 0)^T$.

(e) Derive a formula for the Proportional-Derivative (PD) controller that brings the car to the state $(x, v) = (0, 0)^T$. Here a PD controller is a PID controller whose integral component has coefficient zero, i.e., $K_i = 0$. Moreover, implement the PD car controller using `ode45s` and plot it using `plot`.

(f) Devise now a PD controller that steers the car to $x = 100$ rather than $x = 0$.

(g) Using similarity transformation, compute the symbolic solution of the closed loop system $\frac{d}{dt}x = (A - BK)x$ for initial condition $x(0) = (-100, 0)^T$. To this end, compute first the diagonalization of $A - BK$ by invoking `[V,D] = eig(sym(A - BK))`. Afterwards, use further commands such as `inv` and `syms`. As indicated, use symbolic rather than numeric MATLAB routines.

Solution

(b) It can be noted that

$$\begin{pmatrix} \frac{d}{dt}x \\ \frac{d}{dt}v \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & 1 \\ 0 & -\frac{1}{20} \end{pmatrix}}_{A} \cdot \begin{pmatrix} x \\ v \end{pmatrix} + \underbrace{\begin{pmatrix} 0 \\ \frac{1}{1000} \end{pmatrix}}_{B} \cdot F \qquad y = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{C} \cdot \begin{pmatrix} x \\ v \end{pmatrix}$$

(c) Noting that

$$A - BK = A - \begin{pmatrix} 0 \\ \frac{1}{1000} \end{pmatrix} (k_1, k_2) = \begin{pmatrix} 0 & 1 \\ -\frac{k_1}{1000} & -\frac{1}{20} - \frac{k_2}{1000} \end{pmatrix}$$

we observe that

$$\begin{aligned} |(A - BK) - \lambda I| &= \begin{vmatrix} -\lambda & 1 \\ -\frac{k_1}{1000} & -\lambda - \frac{1}{20} - \frac{k_2}{1000} \end{vmatrix} \\ &= \lambda(\lambda + \tfrac{1}{20} + \tfrac{k_2}{1000}) + \tfrac{k_1}{1000} \\ &= \lambda^2 + \lambda(\tfrac{1}{20} + \tfrac{k_2}{1000}) + \tfrac{k_1}{1000} \end{aligned}$$

Since

$$(\lambda - (-1))(\lambda - (-2)) = \lambda^2 + 2\lambda + \lambda + 2 = \lambda^2 + 3\lambda + 2,$$

matching coefficients yields $k_2 = 2950$ and $k_1 = 2000$.

(d) A possible Matlab code is:

Listing 1: Matlab code

```
% (A,B,C,D) representation
A = [ [ 0 1]; [0 -0.05] ];
B = [0; 0.001];
C = [ 1 0 ];
D = 0;
% Place the poles
K = place(A,B,[-1,-2])
% Create closed loop system
syscl = ss(A-B*K, zeros(2,1),C,D);
% Solve numerically and plot
initial(syscl,[-100,0])
```

(e) We first note that the error is $e = (0 - C\binom{x}{v}) = (0 - x)$. Hence, $u_p = K_p(0 - x)$ and $u_d = K_d \frac{d}{dt}(0 - x) = K_d \cdot 0.05 \cdot v$. With this, the overall control input is thus:

$$u = u_p + u_d = (-K_p, 0.05 \cdot K_d)\begin{pmatrix} x \\ v \end{pmatrix}$$

Crucially, a comparison with (c) shows that picking PD gains is equivalent to picking a gain matrix K. That is, for any PD gains, there exist a gain matrix giving rise to the same control and vice versa. This correspondence holds true in general for PID controls (of single input single output systems). However, while both approaches are mathematically equivalent, finding good PID gains is easier than finding good gain matrices, see lecture slides.

A possible MATLAB code for the PD controller is given below.

Listing 2: Matlab code

```
function ex10()
    % finite time horizon
    T = 5;
    % initial condition
    s0 = [10,0]';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    % invocation of numeric ODE solver with PD gains
    Kp = 2; % since 2 = k1/1000
    Kd = -59; % since 59*0.05 = k2/1000
    [I,s]=ode45(@(t,s)odeDrift(t,s,Kp,Kd),I,s0);
    % plot
    plot(I(:),s(:,1),I(:),s(:,2));
end

function ds = odeDrift(t,s,Kp,Kd,ref)
  ds=zeros(2,1);
  ds(1) = s(2);
  ds(2) = -0.05*s(2)-Kp*s(1)+Kd*0.05*s(2);
end
```

(f) We change our error to $e = (ref - C\binom{x}{v}) = (ref - x)$. With this, $u_p = K_p(ref - x)$ and $u_d = K_d\frac{d}{dt}(ref - x) = 0.05 \cdot K_d \cdot v$. Note that ref vanishes in u_d as the derivative of a constant is zero. With this, we can modify the MATLAB code from (e) to:

Listing 3: Matlab code

```
function ex10()
    % finite time horizon
    T = 5;
    % initial condition
    s0 = [10,0]';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    % invocation of numeric ODE solver with PD gains
    Kp = 2;
    Kd = -59;
    % reference value which may have to be re-scaled, see
        video of Brian Douglas (not needed here, why?)
    ref = 100;
    [I,s]=ode45(@(t,s)odeDrift(t,s,Kp,Kd,ref),I,s0);
    % plot
    plot(I(:),s(:,1),I(:),s(:,2));
end

function ds = odeDrift(t,s,Kp,Kd,ref)
  ds=zeros(2,1);
```

```
        ds(1) = s(2);
        ds(2) = -0.05*s(2)+Kp*(ref-s(1))+Kd*0.05*s(2);
    end
```

(g) A possible Matlab code is:

Listing 4: Matlab code

```
    % A, B and K
    A = [ [ 0 1]; [0 -0.05] ];
    B = [0; 0.001];
    K = [2000 2950] ;
    % Compute eigenvalues and eigenvectors of A-BK
    [Vcl,Dcl] = eig(sym(A-B*K))
    % Declare symbolic variable for time
    syms T
    % Invert Vcl symbolically
    invVcl = inv(Vcl)
    % Compute the transformation matrix via similarity
       transformation
    expAclT = Vcl*diag(exp(diag(Dcl*T)))*invVcl
    % Compute the symbolic solution starting at (-1,0)'
    solSym = expAclT * [-100,0]'
    % Cross-check
    solSymCheck = expm(sym((A-B*K)*T)) * [-100,0]'
```

Models and Tools for Cyber-Physical Systems

Digital Written Exam, June the 9th 2021, 10:00-16:00

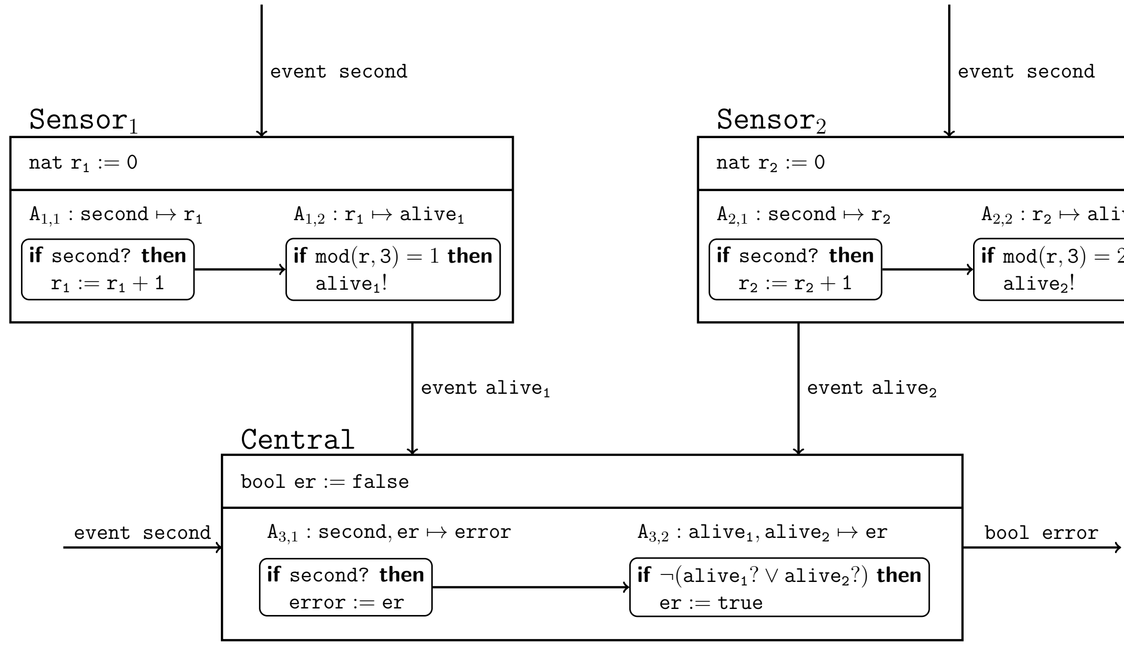
Please read the following before solving the exercises.

- This exam contains 7 exercises. Each exercise is compulsory. The solution has to be composed in English. If you believe that the assignment wording is ambiguous or erroneous, then write down what additional assumption you are using and outline your reasons.
- In some exercises you are asked to extend incomplete UPPAAL and MATLAB files. Solutions have to be uploaded to Digital Exam in the form of UPPAAL, MATLAB, pdf or text files.
- Solutions submitted as digital pictures should be of sufficient quality (high resolution, enough light, not blurry, etc.).
- Allowed aids is the course material (lecture slides and videos, exercise sheets, the book of Alur, UPPAAL/MATLAB tutorials, ect.), the software tools UPPAAL and MATLAB, and your own notes. Anything else is rendered illegal, including, in particular, Googling or asking other persons for help.
- In case of emergencies: Students can contact the instructors during the exam by approaching the study secretary, as outlined in the guidelines for online exams. Keep an eye on your student mail for potential announcements during the exam.

Last but not least, good luck!

Exercise 1: Synchronous Model – Wireless Sensor Network 10 Points

Consider the block diagram for a wireless sensor network given below. The network consists of two sensors $\texttt{Sensor}_1$ and $\texttt{Sensor}_2$ and a central unit `Central`. At every second, the component `Central` monitors the well functioning of the sensors by checking if an event $\texttt{alive}_1$ or $\texttt{alive}_2$ is present. If no event is present, it sets the state variable `er` to `true`.



(a) Consider the synchronous parallel product $\texttt{Sensor}_1 \| \texttt{Sensor}_2 \| \texttt{Central}$. Give the resulting state variables S, output variables O, input variables I, and the precedence relation $\prec$ among tasks (e.g. $A_{1,1} \prec A_{1,2}$).

(b) Can the output variable `error` be set to `true`? If yes, provide a corresponding execution.

.... Solution ....

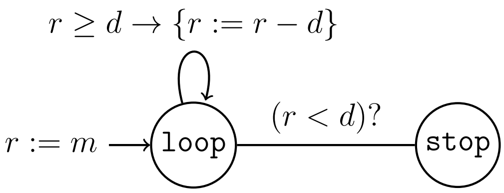
(a)
- $S = \{\texttt{er}, \texttt{r}_1, \texttt{r}_2, \}$
- $O = \{\texttt{error}, \texttt{alive}_1, \texttt{alive}_2\}$
- $I = \{\texttt{second}\}$
- $\prec = \{(A_{1,1}, A_{1,2}), (A_{2,1}, A_{2,2}), (A_{3,1}, A_{3,2}), (A_{1,2}, A_{3,2}), (A_{2,2}, A_{3,2})\}$

(b) Lets assume ordering on state variables $(\texttt{er}, \texttt{r}_1, \texttt{r}_2)$, and for outputs $, \texttt{error}, \texttt{alive}_1, \texttt{alive}_2$ then an execution

$$(false, 0, 0) \xrightarrow{\top/false, \top, \bot} (false, 1, 1) \xrightarrow{\top/false, \bot, \top} (false, 2, 2) \xrightarrow{\top/false, \bot, \bot}$$
$$(true, 3, 3) \xrightarrow{\top/true, \top, \bot} (true, 4, 4) \to \ldots$$

Exercise 2: Symbolic Transition System 10 Points

The following state machine finds the remainder $\texttt{REM}(m, d)$ resulting from the division of positive integers $m = 290$ and $d = 9$.



(a) Describe the underlying transition system symbolically giving, state variables, initialization formula, and transition formula φ.

(b) Given the region $A : (100 \leq r \leq 290)$, compute the image using the transition formula φ. Describe the required steps.

.... Solution ....

(a) The transition system REM(m,n) has state variable r and $mode$ of the enumerated type $\{loop, stop\}$. The initialization is given by the formula

$$(mode = loop) \wedge (r = m)$$

The transition formula φ is given as:

$$\begin{array}{ll} & [(mode = loop) \wedge (r \geq d) \wedge (r' = r - d) \wedge (mode' = loop)] \\ \vee & [(mode = loop) \wedge (r < d) \wedge (r' = r) \wedge (mode' = stop)] \end{array}$$

(b)
- Conjuction of A and φ, note $A \equiv (100 \leq r \wedge r \leq 290)$

$$(100 \leq r \leq 290) \wedge [(mode = loop) \wedge (r \geq 9) \wedge (r' = r - 9) \wedge (mode' = loop)]$$

- Existententially quantify $mode$

$$(100 \leq r \leq 290) \wedge (r \geq 9) \wedge (r' = r - 9) \wedge (mode' = loop)$$

- Existententially quantify r

$$(100 \leq r' + 9) \wedge (r' + 9 \leq 290) \wedge (mode' = loop)]$$

- Renaming

$$(91 \leq r \leq 281) \wedge (mode' = loop)$$

Exercise 3: Asynchronous Model - Shared Registers UPPAAL 10 Points

The following process increases a shared atomic register `n` by using a local register `r` and read-write operations.

```
global int k:=10; int n:=0;
process P_i
local int j:=0;
local int r:=0;
while ( j < k ) {
  r := n; r := r + 1; n := r;
  j := j + 1;
}
```

(a) Consider the product $(P_1 \| P_2)$, what is the minimal final value of global variable `n`? *Hint:* use the UPPAAL model `ex3.xml` with a suitable query to find the value.

(b) Explain how the minimal value for `n` is obtained.

(c) Consider the product $(P_1 \| P_2 \| P_3)$ what is the minimal final value of global variable `n`?

.... Solution ....

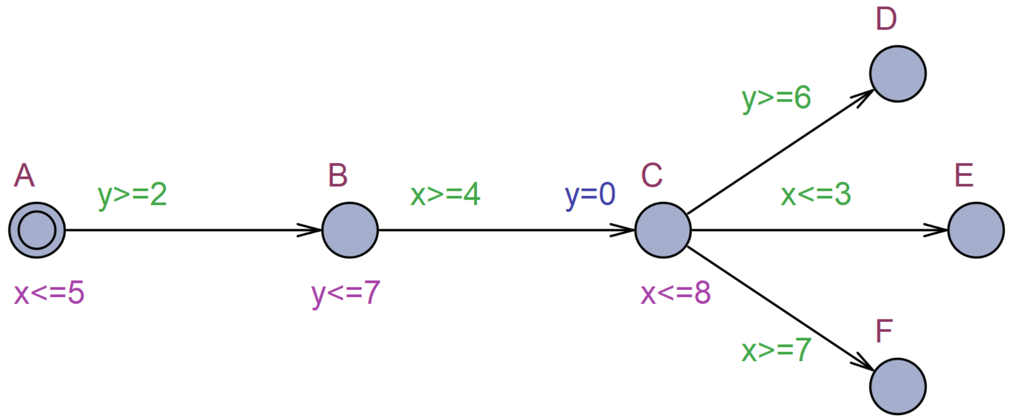
(a) $n = 2$

(b) One process e.g. P_2 is able to store in r the value 0, then P_1 executes the loop 9 times, then P_2 inteferes and sets $n = 1$. P_1 reads n, sets $r_1 = 1$ and increments to $r_1 = 2$. Then P_2 executes to completion. Finally P_1 sets $n = r_1 = 2$ and exits the loop.

(c) $n = 2$. For three processes and $k = 10$ the state space it too big and UPPAAL might not be able to explore it. To help your intuition you can set the value of $k = 5$ and observe that similar executions as for $(P_1 \| P_2)$ occur.

Exercise 4: Exploration of Timed Automata 10 Points

Consider the timed automaton $\mathcal{A}$ below with two clocks x and y.



(a) Which of the three locations D, E and F are reachable from the initial state $(\texttt{A}, \texttt{x} = 0, \texttt{y} = 0)$?

(b) For each of these three goal locations that is reachable, provide a timed transition sequence that leads to the location from the initial state.

(c) For each of the three goal locations that is reachable, what is the fastest time of reaching that location. Provide a witness timed transition sequence.

(d) Describe using difference constraints the reachable zones upon entry and after delay for the locations A, B and C.

(e) For each of the three goal locations that is NOT reachable, suggest a weakening of the guard leading to the location, so that the location becomes reachable.
NOTE: `x<=7` is weaker than `x<=5` and `x>=2` is weaker than `x>=4`.

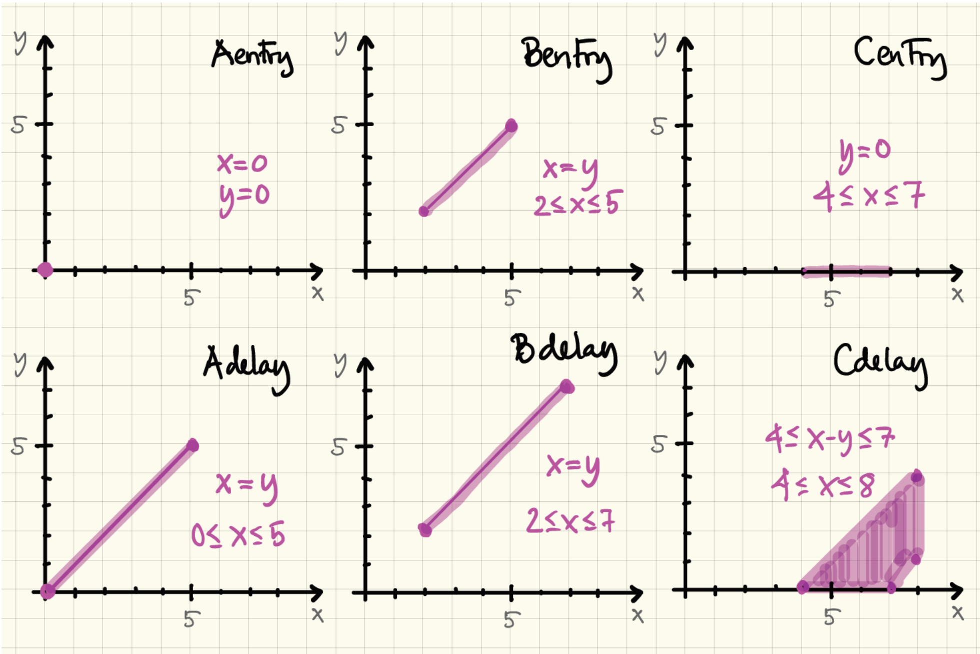
Solution

(a) F

(b)
$$\begin{array}{ll}
(\texttt{A}, \mathrm{x}=0, \mathrm{y}=0) & \xrightarrow{2} \\
(\texttt{A}, \mathrm{x}=2, \mathrm{y}=2) & \rightarrow \\
(\texttt{B}, \mathrm{x}=2, \mathrm{y}=2) & \xrightarrow{2} \\
(\texttt{B}, \mathrm{x}=4, \mathrm{y}=4) & \rightarrow \\
(\texttt{C}, \mathrm{x}=4, \mathrm{y}=0) & \xrightarrow{3} \\
(\texttt{C}, \mathrm{x}=7, \mathrm{y}=3) & \rightarrow \\
(\texttt{F}, \mathrm{x}=7, \mathrm{y}=3) & \rightarrow
\end{array}$$

(c) F is reachable in 7 time units. The above timed transition sequence is a witness.

(d) See figure below



(e) The guard to D should be weakened to `y>=4`. The guard to E should be weakend to `x<=4`.

Exercise 5: The Druzba Mutual Exclusion Problem 15 Points

The problem is based on a true story of one of the lectures of this course experienced during the conference CONCUR in 2002 in Brno. During this – otherwise extremely nice conference – accommodation was arranged in the local Druzba hostel. Rooms being nice, there was the unexpected surprise of sharing the shower with the neighbor (causing some screaming in at least one occasion), see Fig. 1 below.

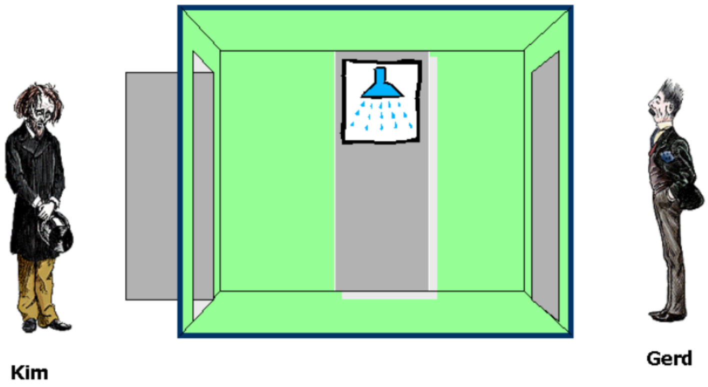


Figure 1: Sharing a Shower in Druzba

During the conference a lot of possible solutions for how to obtain mutual exclusion in the shower were discussed. Your job is to help find a good solution.

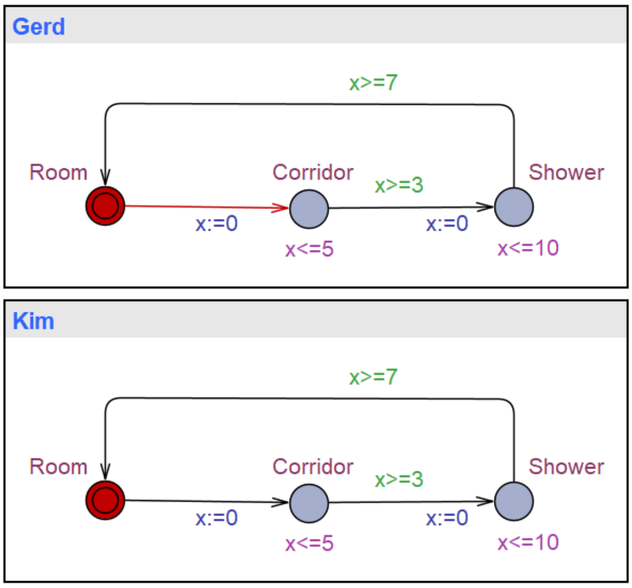


Figure 2: First model of the Druzba Mutex problem

In Fig. 2 you see an initial solution in UPPAAL to the problem. You can find the complete UPPAAL model in Digital Exam in the file `Druzba.xml`. Here the two users of the shower (Gerd and Kim) may at any moment in time make a go for the shower. This first requires waiting between 3 and 5 minutes in the `Corridor`. The actual use of the `Shower` will take between 7 and 10 minutes.

(a) Formulate as a UPPAAL property ϕ_M (in TCTL) the desired mutual exclusion on the `Shower` location.

(b) Check in UPPAAL whether the initial solution satisfies the mutual exclusion property ϕ_M. If not use UPPAAL to generate a violating trace. Please provide the corresponding Message Sequence Chart (MSC) found in the simulator of UPPAAL.

(c) Formulate as a UPPAAL property ϕ_G (in TCTL) the desired liveness property that whenever Gerd enters the `Corridor` he will eventually get to the `Shower`. Formulate a similar liveness property ϕ_K for Kim.

(d) Check in UPPAAL whether the initial solution satisfies the above liveness properties ϕ_G and ϕ_K and report the answer.

(e) Please upload to Digital Examn your extension of the initial solution with the properties ϕ_M, ϕ_G and ϕ_K in a file with name `DruzbaSol1.xml`.

Now assume that the bathroom has a Light which can be checked and switched on before entering the bathroom – and switched back off when leaving the bathroom.

(a) Extend the initial model with a Boolean variable `L` to represent whether the Light is on or off.

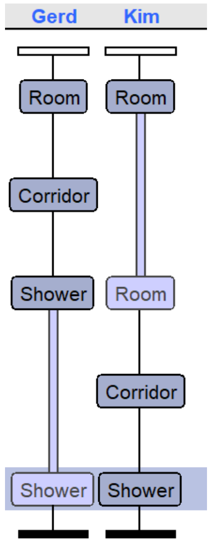
(b) As an improved solution, use `L` to "check and switch on" upon entering the `Cooridor`. Check in UPPAAL whether the properties ϕ_M, ϕ_G and ϕ_K are satisfied for the improved solution.

(c) Please upload to Digital Examn your proposal for the improved solution in a file with name `DruzbaSol2.xml`

.... Solution ....

(a) $\phi_M =$ `A[] not (Kim.Shower and Gerd.Shower)`

(b) The property does not hold. The following is a MSC witness.

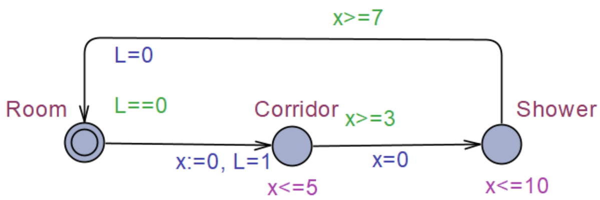


(c) $\phi_G =$ `Gerd.Corridor --> Gerd.Shower` and $\phi_K =$ `Kim.Corridor --> Kim.Shower`

(d) Both ϕ_G and ϕ_K holds for the initial model.

Extension of the model with a Boolean variable `L`:

(a) The extension results in the following model:;



(b) All properties ϕ_M, ϕ_G and ϕ_K holds.

Exercise 6: Continuous System 10 Points

In this exercise, you may (but are not obliged to) justify your answers using MATLAB. Any use of MATLAB must be however documented by crisp snippets of MATLAB's command window featuring the relevant inputs and outputs.

Let the following matrices be given:

$$A = \begin{pmatrix} -3 & 2 \\ -2 & 2 \end{pmatrix} \qquad\qquad B = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

(a) Prove that $x = 0$ is not a stable equilibrium of $\frac{d}{dt}x(t) = Ax(t)$.

(b) Compute a gain matrix K for which the feedback loop matrix $A - BK$ has eigenvalues -2 and -3.

(c) Is there an initial condition $x(0) \neq 0$ for which $\frac{d}{dt}x(t) = Ax(t)$ admits a solution that converges towards zero? If yes, provide such an initial condition, i.e., $x(0) \neq 0$ and $\lim_{t\to\infty} x(t) = 0$. If not, argue why such an initial condition does not exist.

.... Solution ....

(a) Since our system is $\frac{d}{dt}x(t) = Ax(t)$ and A is a 2×2 matrix, variable x is a vector $x(t) = \binom{x_1(t)}{x_2(t)}$.

With MATLAB: Input

Listing 1: Matlab input

```
A = [ [-3  2];  [-2  2] ];
eig(A)
```

yields output

Listing 2: Matlab output

```
ans =
-2
 1
```

Hence, $+1$ is an eigenvalue of A and the discussion from the course implies that $x = 0$ is not a stable equilibrium.

By hand: As discussed in the course, it suffices to prove that A has an eigenvalue with positive real part. The eigenvalues are the roots of the characteristic polynomial $p(\lambda) = |A - \lambda I|$, where $|\cdot|$ denotes the determinant. With this, we obtain

$$p(\lambda) = |A - \lambda I| = \begin{vmatrix} -3-\lambda & 2 \\ -2 & 2-\lambda \end{vmatrix} = (-3-\lambda)(2-\lambda) - (-2)2 = \lambda^2 + \lambda - 2$$

High school math then shows that $\lambda^2 + \lambda - 2 = (\lambda - 1)(\lambda + 2)$, implying that the eigenvalues of A are 1 and -2.

(b) *With MATLAB:* Input

Listing 3: Matlab input

```
A = [ [-3  2]; [-2  2] ];
B = [ 1; 0 ];
K = place(A,B,[-2 ,-3])
lambdas = eig(A - B*K)   % double check, not required
```

yields output

Listing 4: Matlab output

```
K = [4   -8]
lambdas =
-3
-2
```

By hand:

$$A - BK = A - \begin{pmatrix} 1 \\ 0 \end{pmatrix} (k_1, k_2) = \begin{pmatrix} -3-k_1 & 2-k_2 \\ -2 & 2 \end{pmatrix}$$

we observe that

$$\begin{aligned} |(A-BK) - \lambda I| &= \begin{vmatrix} -3-k_1-\lambda & 2-k_2 \\ -2 & 2-\lambda \end{vmatrix} \\ &= (-3-k_1-\lambda)(2-\lambda) - (-2)(2-k_2) \\ &= \lambda^2 + \lambda(-2+3+k_1) - 2 - 2k_1 - 2k_2 \end{aligned}$$

Since

$$(\lambda - (-2))(\lambda - (-3)) = \lambda^2 + 5\lambda + 6,$$

matching coefficients yields $k_1 = 4$ and $k_2 = -8$.

(c) From the course, we know that the eigenvectors underlying eigenvalues with negative real parts yield solutions converging to zero (instead, eigenvectors underlying eigenvalues with positive real part yield diverging solutions). From (a) we know that -2 is an eigenvalue of A. Because of this, solving the linear system of equations $Ax = -2x$ implies that $x = (2a, a)^T$ is, for any $a \neq 0$, an eigenvector for eigenvalue -2 (this can be computed by hand or for instance via `[V,D] = eig(A)` in MATLAB). Consequently, any $x(0) = (2a, a)^T$ with $a \neq 0$ constitutes an initial condition converging to zero.

(Add-on, not required.) Note that this does not contradict the fact that $x = 0$ is an unstable equilibrium. Indeed, for unstability, it suffices to have *at least one* eigenvalue with positive real part. This is here the case since $\lambda = 1$ is an eigenvalue of A. The corresponding eigenvectors solve the equation $Ax = x$ and are given by $x = (a, 2a)^T$ for any $a \neq 0$, while the corresponding diverging solutions are $x(t) = (e^t a, e^t 2a)$. On the other hand, the converging solutions are given by $x(t) = (e^{-2t}2a, e^{-2t}a)$. One can cross-check that this is indeed the case by differentiating the expression:

$$\begin{aligned} \frac{d}{dt}x_1(t) &= -2e^{-2t}2a = -3e^{-2t}2a + 2e^{-2t}a = -3x_1(t) + 2x_2(t) \\ \frac{d}{dt}x_2(t) &= -2e^{-2t}a = -2e^{-2t}2a + 2e^{-2t}a = -2x_1(t) + 2x_2(t) \end{aligned}$$

A similar check can be done by differentiating $x(t) = (e^t a, e^t 2a)$.

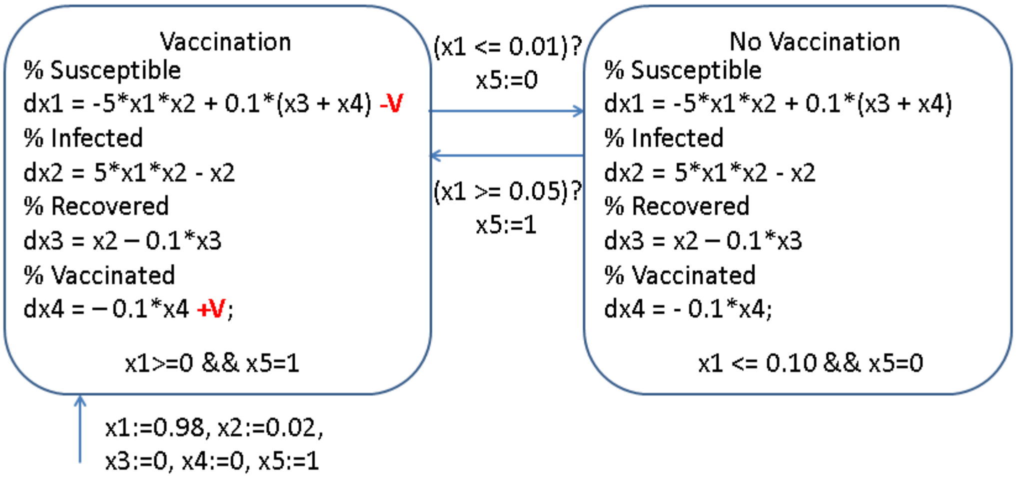


Figure 3: Hybrid model of a vaccination program.

Exercise 7: Hybrid Vaccination Model 10 Points

Consider the hybrid system of Figure 3 and the following incomplete MATLAB script below which can be found in Digital Exam:

Listing 5: Incomplete MATLAB script `vaccination.m`

```
function vaccination()
    % Initial percentages of susceptible, infected, recovered, vaccinated
    x0 = [0.98, 0.02, 0.00, 0.00]';
    % Initial mode
    global x5;
    x5 = 1;
    % Discretization of time interval
    T = 5.0;
    dt = T / 100;
    I = 0:dt:T;

    figure
    hold on
    % ... add code ...
    [I,x]=ode45(@(t,x)drift(t,x,V),I,x0);
    % ... add code ...
end

function dx = drift(t,x,V)
    global x5;
    dx = zeros(4,1);

    % ... add code ...

```

```
    if(x5 == 1)
        dx(1) = -5*x(1)*x(2) - V + 0.1*x(3) + 0.1*x(4);
        dx(2) = 5*x(1)*x(2) - x(2);
        dx(3) = x(2) - 0.1*x(3);
        dx(4) = V - 0.1*x(4);
    else
        dx(1) = -5*x(1)*x(2) + 0.1*x(3) + 0.1*x(4);
        dx(2) = 5*x(1)*x(2) - x(2);
        dx(3) = x(2) - 0.1*x(3);
        dx(4) = - 0.1*x(4);
    end
end
```

Extend the MATLAB script `vaccination.m` so that it

- Computes for each vaccination rate $V \in \{0.1, 0.2, \ldots, 1.0\}$ an execution of the hybrid system on the time interval $[0; 5]$ and;
- Plots the infection forecasts x2 of all ten executions in a common figure.

Note: Solutions defining new MATLAB functions or making use of MATLAB commands other than `plot` or `ode45` will be not considered.

. . . . Solution . . . .

Listing 6: Complete MATLAB solution

```
function vaccination()
    % Initial percentages of susceptible, infected, recovered,
        vaccinated
    x0 = [0.98, 0.02, 0.00, 0.00]';
    % Initial mode
    global x5;
    x5 = 1;
    % Discretization of time interval
    T = 5.0;
    dt = T / 100;
    I = 0:dt:T;

    figure
    hold on
    % added code
    for i = 1 : 10
        V = i*0.1;
        [I,x]=ode45(@(t,x)drift(t,x,V),I,x0);
        plot(I(:),x(:,2));
    end
end

function dx = drift(t,x,V)
    global x5;
```

```
    dx = zeros(4,1);

    % added code
    if(x5 == 1 && x(1) <= 0.01 )
        x5 = 0;
    elseif(x5 == 0 && x(1) >= 0.05)
        x5 = 1;
    end

    if(x5 == 1)
        dx(1) = -5*x(1)*x(2) - V + 0.1*x(3) + 0.1*x(4);
        dx(2) = 5*x(1)*x(2) - x(2);
        dx(3) = x(2) - 0.1*x(3);
        dx(4) = V - 0.1*x(4);
    else
        dx(1) = -5*x(1)*x(2) + 0.1*x(3) + 0.1*x(4);
        dx(2) = 5*x(1)*x(2) - x(2);
        dx(3) = x(2) - 0.1*x(3);
        dx(4) = - 0.1*x(4);
    end
end
```