Models and Tools for Cyber-Physical Systems

Digital Written Exam, June the 9th 2022, 10:00-14:00

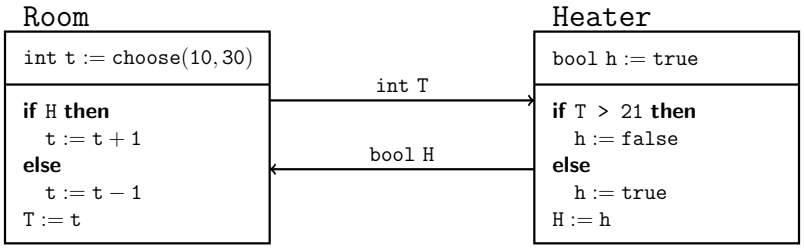
Please read the following before solving the exercises.

- This exam contains 7 exercises, worth 75 points in total. 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 Digitaleksamen in the form of UPPAAL, MATLAB, pdf or text files. For each of your solutions, ensure that it is clear which exercise is being solved.
- The exam is open book but asking/receiving help from others is not allowed.
- In case of emergencies: Students can contact the instructors during the exam by approaching the study secretary.

Last but not least, good luck!

Exercise 1: CPS – Discrete Room Heater 10 Points

Consider the block diagram below for a room equipped with a heater. The component `Room` has an integer state variable temperature `t`. If the heater is on then the temperature increases otherwise it decreases. The component `Heater` has a boolean state variable `h` indicating if the heater is on or off. If the room temperature is above 21 degrees, the heater is turned off, otherwise is turned on.



(a) Describe using words a *safety requirement* for the system

(b) Describe using words a *liveness requirement* for the system

(c) Is the underlying synchronous reactive component *deterministic*? Why?

(d) Is the underlying synchronous reactive component *finite-state*? Why?

(e) Is the property $\texttt{t} > 21 \implies \texttt{h} = \texttt{false}$ an *invariant*? If not, provide a counter-example.

Solution

(a)

(b)

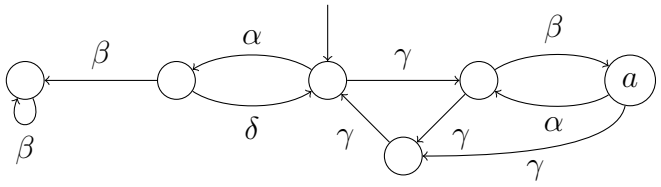
(c) No. It does not have a single initial state because of the statement `int t := choose(10, 30)`.

(d) No. The state and input variables `T, t` are not finite.

(e) No. The initial state with $(\texttt{t} = 30)$ and $(\texttt{h} = \texttt{true})$ is a counter example.

Exercise 2: Fairness Assumptions 10 Points

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



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

(a) Infinitely often α

(b) Infinitely often β

(c) Infinitely often α and β

(d) Infinitely often γ and strong-fairness for β

(e) Infinitely often γ and weak-fairness for β

Solution

(a) Does not hold. Runs induced by the $\alpha - \delta$-loop are fair.

(b) Does not hold. The runs induced by the β-self-loop are fair.

(c) Holds. Note that runs induced by the β-self-loop have been removed by the fairness assumption.

(d) Holds. The γ-loop is infinitely often visited and β is infinitely often enabled. Thus β has to be infinitely often taken.

(e) Does not hold. The γ-loop is infinitely often visited and β is infinitely often enabled. However, β is not continuously enabled, thus runs induced by the γ-loop are weakly-fair.

Exercise 3: Program Graphs UPPAAL 10 Points

The following process takes an integer `i` = 39. If `i` is even then it divides `i` by 2, otherwise it multiplies `i` by 3 and adds one. The program terminates when `i` is equal to 1. The program statements are annotated with labels.

```
initial:    int i = 39;
while:      while ( i != 1 ) {
if:           if (( i % 2) == 0)
ifcase:         i = i / 2;
              else
elsecase:       i = 3*i + 1;
done:       }
```

(a) Encode the process described above in a UPPAAL model. Use one UPPAAL location per code label. Upload your model with name ex3.xml to digital exam.

(b) Verify if the variable `i` can be bigger than 200. Write down the query you used.

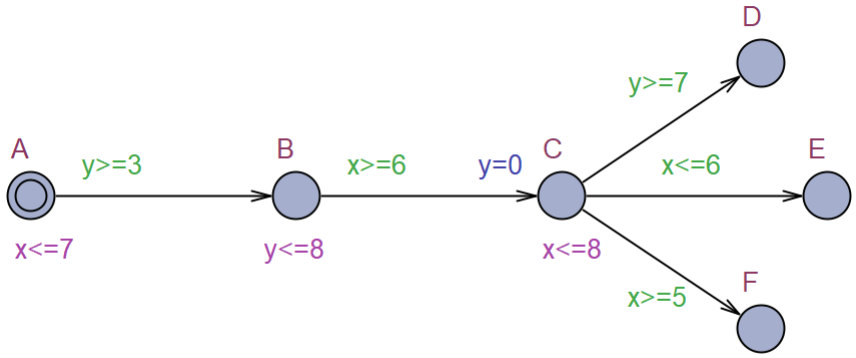
(c) How many states are reachable in your model? How did you compute the number of reachable states?

Solution

See the UPPAAL solution file.

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 $(\mathtt{A}, \mathtt{x} = 0, \mathtt{y} = 0)$?

(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, investigate whether it is possible to weaken the guard leading to the location, so that the location becomes reachable.
NOTE: x<=7 is weaker than x<=5. Similarly, x>=2 is weaker than x>=4.

Solution

(a) F and E

(b)

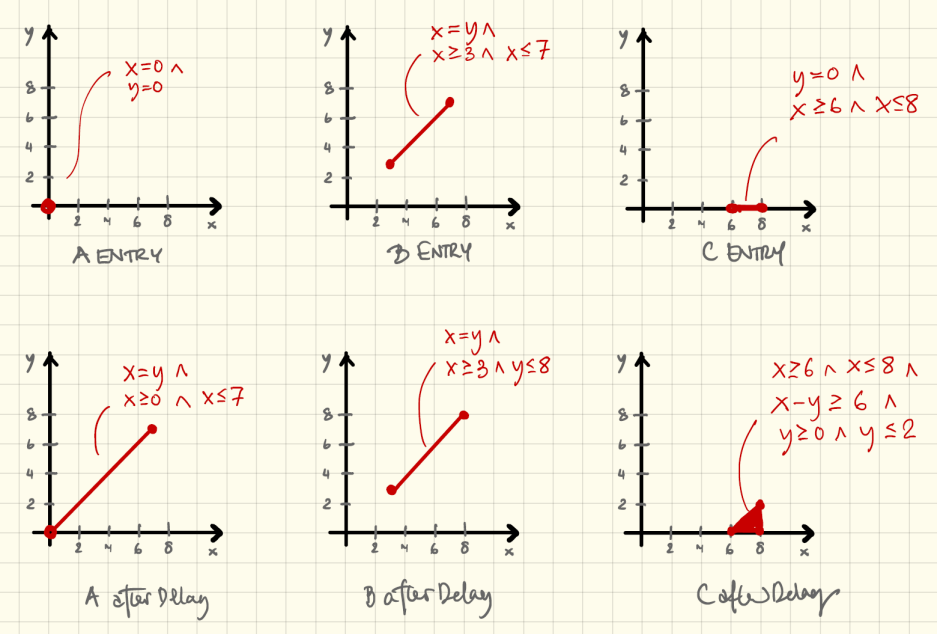
$$(\mathtt{A}, x = 0, y = 0)) \xrightarrow{6} (\mathtt{A}, x = 6, y = 6)) \to (\mathtt{B}, x = 6, y = 6)) \to (\mathtt{C}, x = 6, y = 0)) \to (\mathtt{F}, x = 6, y = 0))$$

and

$$(\mathtt{A}, x = 0, y = 0)) \xrightarrow{6} (\mathtt{A}, x = 6, y = 6)) \to (\mathtt{B}, x = 6, y = 6)) \to (\mathtt{C}, x = 6, y = 0)) \to (\mathtt{E}, x = 6, y = 0))$$

(c) 6 for both locations. The witnessing sequences are above.

(d) The reachable zones upon entry and after delay:



(e) Changing the guard to $y \geq 2$ will make D reachable.

Exercise 5: Communication over 2 one-place buffers 15 Points

Figure 1 is a UPPAAL model of a small communication system. Here the component `SEND` that wants to send a message (or a signal) to the receiving component `REC`. The message is sent via a pipeline of two (identical) one-place buffer components `MED1` and `MED2` with timing constraints. The synchronization between components is made using three channels `ch01`, `ch12` and `ch23`. Once a buffer component has received the message - i.e. has taken the edge from `I` to `R` - the invariant `x<10` on `R` together with the guard `x>=5` on the edge from `R` to `D` ensures that the message will be passed on within a delay of 5 to 10 time-units.

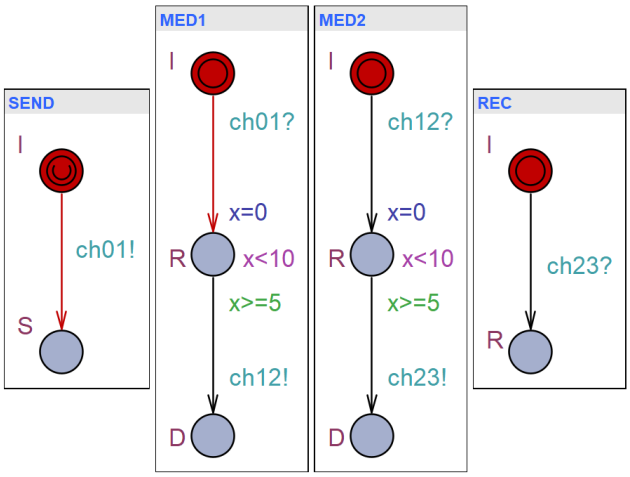


Figure 1: A small Communication System

You can find the complete UPPAAL model in Digital Exam in the file `Media.xml`.

(a) Formulate as a UPPAAL property ϕ_C (in TCTL) that the message *can be* be received by `REC` within 15 time-units from the initial state.

(b) Check in UPPAAL whether ϕ_C is satisfied, and if so, provide the corresponding Message Sequence Chart (MSC) found in the simulator of UPPAAL.

(c) What is the earliest time that the message *can be* received by `REC` from the initial state?

(d) Formulate as a UPPAAL property ϕ_M (in TCTL) that the message *must be* received by `REC` within 30 time-units from the initial state.

(e) Check in UPPAAL whether ϕ_M is satisfied or not.

(f) What is the latest time that the message *must be* received by `REC` from the initial state?

(g) Which combinations of the locations of `MED1` and `MED2` - i.e. {`MED1.I`, `MED1.R`, `MED1.D`} and {`MED2.I`, `MED2.R`, `MED2.D`} - are reachable from the initial state.

(h) Please upload to Digitaleksamen an extension of the UPPAAL model with the properties ϕ_C, ϕ_M in a file with name `MediaSol1.xml`.

In the following we want to change the models of the buffers `MED1` and `MED2` so that they can lose messages. You may do this by adding a new edge from the `R`-location to a new *error* location `E`.

(a) Modify the initial UPPAAL model of the communication systems so that buffers are loossy as described above.

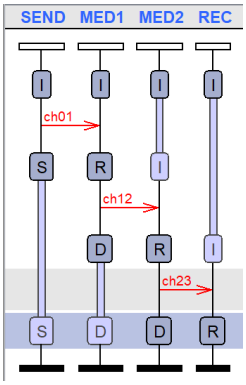
(b) Check in UPPAAL whether the properties ϕ_C and ϕ_M are satisfied by the modified model. If not use UPPAAL to generate a violating trace, and provide the corresponding MSC found in the simulator of UPPAAL.

(c) Please upload to Digital Exam your modified UPPAAL model in a file with name `MediaSol2.xml`.

Solution

(a) $\phi_C =$ `E<> time<= 15 and REC.R`

(b) The property holds.



(c) 10

(d) $\phi_M =$ `A<> time>=30 and REC.R`

(e) The property holds

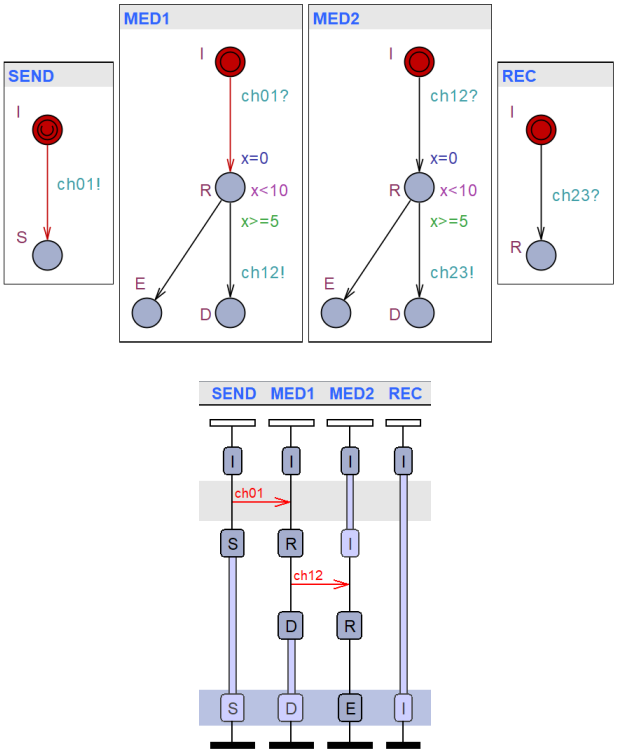
(f) 20

(g) Reachable combinations of locations: (I,I), (R,I), (D,R), (D,D).

Extension of the model to loossy buffers:

(a) The loosy models:

(b) The property ϕ_C holds, whereas ϕ_M does not. A violating trace is given by the MSC:



Exercise 6: DC Motor in MATLAB 15 Points

Consider the model of an electric DC motor

$$\frac{d}{dt}v(t) = 0.01 \cdot i - 0.10 \cdot v \qquad \frac{d}{dt}i(t) = -0.02 \cdot v - 2.00 \cdot i + 2 \cdot V(t),$$

where v is the rotational velocity, i the electric current and V the input voltage. The physical units are dropped for simplicity. [Intuitively, the higher the input voltage, the higher the current and the velocity. If no input voltage is applied, the velocity and the current decrease due to friction and resistance.]

(a) Assuming that we have no feedforward control (i.e., $D = 0$) and wish to measure the rotational velocity v only, describe the equations of the model using the (A, B, C, D) notation. That is, provide the state vector s and the matrices A, B and C.

(b) Compute the gain matrix K that ensures that $A - BK$ has eigenvalues -0.1 and -0.2 using the MATLAB command `place`. Cross-check that $A-BK$ has the desired eigenvalues using the MATLAB command `eig`. Document your use of MATLAB via *crisp* snippets.

(c) Derive a formula for the Proportional-Derivative (PD) control that steers the system to $v = 10$. Using the formula, complete the MATLAB code given below. Adjust the reference value by inspecting the plot.

You are allowed only to change the statements with the *complete comment*.

(d) Extend the MATLAB code given below so that it implements a PI controller, i.e., a PID controller whose derivative input is absent (i.e., $K_d = 0$). To this end, exploit that the integral input, u_i, satisfies $\frac{d}{dt}u_i(t) = K_i e(t)$, where e is the error.

You are allowed only to change the statements with the *complete comment*.

Listing 1: Incomplete code of Exercise (c), **available** on Digitaleksamen

```
function motorPD()
    % finite time horizon
    T = 20;
    % initial condition
    s0 = [0,0]';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    % invocation of numeric ODE solver with PD gains
    Kp = 100; % proportional gain
    Kd = -1; % derivative gain
    ref = % complete (reference value)
    [I,s]=ode45(@(t,s)odeDrift(t,s,Kp,Kd,ref),I,s0);

    % plot
    figure()
    plot(I(:),s(:,1));
    legend('v');
    figure()
    plot(I(:),s(:,2));
    legend('i');
end

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

    % s(1) = v , s(2) = i
    ds(1) = % complete
    ds(2) = % complete
end
```

Listing 2: Incomplete MATLAB code of Exercise (d), **available** on Digitaleksamen

```
function motorPI()
    % finite time horizon
    T = 25;
    % initial condition
    s0 = [0,0,0]';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    % invocation of numeric ODE solver with PI gains
    Kp = 100; % proportional gain
    Ki = 5 % integral gain
    ref = 10; % reference value
    [I,s]=ode45(@(t,s)odeDrift(t,s,Kp,Ki,ref),I,s0);

    % plot
    figure()
    plot(I(:),s(:,1));
    legend('v');
    figure()
    plot(I(:),s(:,2));
    legend('i');
end

% ODE equations
function ds = odeDrift(t,s,Kp,Ki,ref)
    ds=zeros(3,1);

    % s(1) = v , s(2) = i , s(3) = u_i
    ds(1) = % complete
    ds(2) = % complete
    ds(3) = % complete
end
```

Solution

(a) It can be noted that

$$\begin{pmatrix} \frac{d}{dt}v \\ \frac{d}{dt}i \end{pmatrix} = \underbrace{\begin{pmatrix} -0.10 & 0.01 \\ -0.02 & -2.00 \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} v \\ i \end{pmatrix}}_{s} + \underbrace{\begin{pmatrix} 0 \\ 2 \end{pmatrix}}_{B} \cdot V \qquad y = \underbrace{\begin{pmatrix} 1 & 0 \end{pmatrix}}_{C} \cdot \begin{pmatrix} v \\ i \end{pmatrix}$$

(b) A possible MATLAB code is:

Listing 3: MATLAB code

```
% (A,B,C,D) representation
A = [ [ -0.10  0.01]; [-0.02  -2.00] ];
B = [0; 2];
C = [ 1 0 ];
D = 0;
% Place the eigenvalues
K = place(A,B,[-0.1,-0.2])
% Cross-check the eigenvalues
[V,D] = eig(A-B*K)
```

(c) We first note that the error is $e = (ref - C\binom{v}{i}) = (ref - v)$. Hence, $u_p = K_p(ref - v)$ and $u_d = K_d\frac{d}{dt}(ref - v) = -K_d(-0.10 \cdot v + 0.01 \cdot i)$. With this, the given MATLAB code can be completed as follows:

Listing 4: MATLAB code

```
function motor()
    % finite time horizon
    T = 20;
    % initial condition
    s0 = [0,0]';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    % invocation of numeric ODE solver with PD gains
    Kp = 100;
    Kd = -1;
    ref = 11; % adjustment via visual inspection
    %ref = 10*(10/s(size(s,1),1)); % precise adjustment
    [I,s]=ode45(@(t,s)odeDrift(t,s,Kp,Kd,ref),I,s0);
    %[I,s]=ode45(@(t,s)odeDrift(t,s,Kp,Kd,ref),I,s0);

    % plot
    figure()
    plot(I(:),s(:,1));
    legend('v');
    figure()
    plot(I(:),s(:,2));
    legend('i');
```

```
end

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

    ds(1) = -0.10*s(1) + 0.01*s(2);
    ds(2) = -0.02*s(1) -2*s(2) + 2*(Kp*(ref - s(1)) -Kd*(-0.10*s
        (1) + 0.01*s(2)));
end
```

(d) Since $\frac{d}{dt}u_i(t) = K_i(ref - v(t))$, we get:

Listing 5: MATLAB code

```
function motor()
    % finite time horizon
    T = 25;
    % initial condition
    s0 = [0,0,0]';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    % invocation of numeric ODE solver with PD gains
    Kp = 100; %proportional gain
    Ki = 5; %integral gain
    ref = 10; % reference value
    [I,s]=ode45(@(t,s)odeDrift(t,s,Kp,Ki,ref),I,s0);

    % plot
    figure()
    plot(I(:),s(:,1));
    legend('v');
    figure()
    plot(I(:),s(:,2));
    legend('i');
end

% ODE equations
function ds = odeDrift(t,s,Kp,Ki,ref)
    ds=zeros(3,1);

    ds(1) = -0.10*s(1) + 0.01*s(2);
    ds(2) = -0.02*s(1) -2*s(2) + 2*(Kp*(ref-s(1)) + s(3));
    ds(3) = Ki*(ref - s(1));
end
```

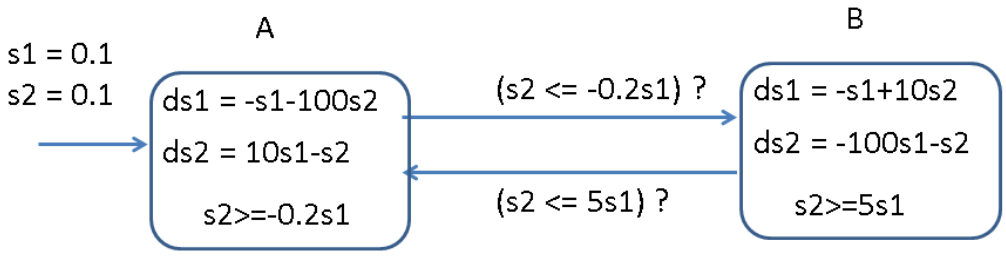


Figure 2: Hybrid System.

Exercise 7: Hybrid System 5 Points

Consider the hybrid system given in Figure 2. Complete the MATLAB code given below so that it prints a simulation of the system on time interval $[0; 0.2]$. You can only expand from the line containing the *complete comment*.

Listing 6: Incomplete MATLAB code of the hybrid system, **available** on Digitaleksamen

```
function hybrid()
    % finite time horizon
    T = 0.2;
    % initialization of states
    s0 = [0.1,0.1]';
    global disc;
    disc = 'A';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    [I,s]=ode45(@(t,s)odeDrift(t,s),I,s0);
    % plot
    plot(s(:,1),s(:,2));
end

function ds = odeDrift(t,s)
    global disc;
    ds=zeros(2,1);

    if % complete

    if(disc == 'A')
        ds(1) = -s(1) -100*s(2);
        ds(2) = 10*s(1) -s(2);
    else
        ds(1) = -s(1) +10*s(2);
        ds(2) = -100*s(1) -s(2);
    end
end
```

Solution

Listing 7: MATLAB code of the hybrid system.

```
function hybrid()
    % finite time horizon
    T = 0.2;
    % initialization of continuous state
    s0 = [0.1,0.1]';
    % initialization of discrete state
    global disc;
    disc = 'A';
    % time points at which solution should be approximated
    dt = T / 100;
    I = 0:dt:T;
    [I,s]=ode45(@(t,s)odeDrift(t,s),I,s0);
    % plot
    plot(s(:,1),s(:,2));
end


function ds = odeDrift(t,s)
    global disc;
    ds=zeros(2,1);

    if(disc == 'A' && s(2) <= -0.2*s(1))
        disc = 'B';
    elseif(disc == 'B' && s(2) <= 5*s(1))
        disc = 'A';
    end

    if(disc == 'A')
        ds(1) = -s(1) -100*s(2);
        ds(2) = 10*s(1) -s(2);
    else
        ds(1) = -s(1) +10*s(2);
        ds(2) = -100*s(1) -s(2);
    end
end
```