Models and Tools for Cyber-Physical Systems

Digital Written Exam, June the 9th 2021

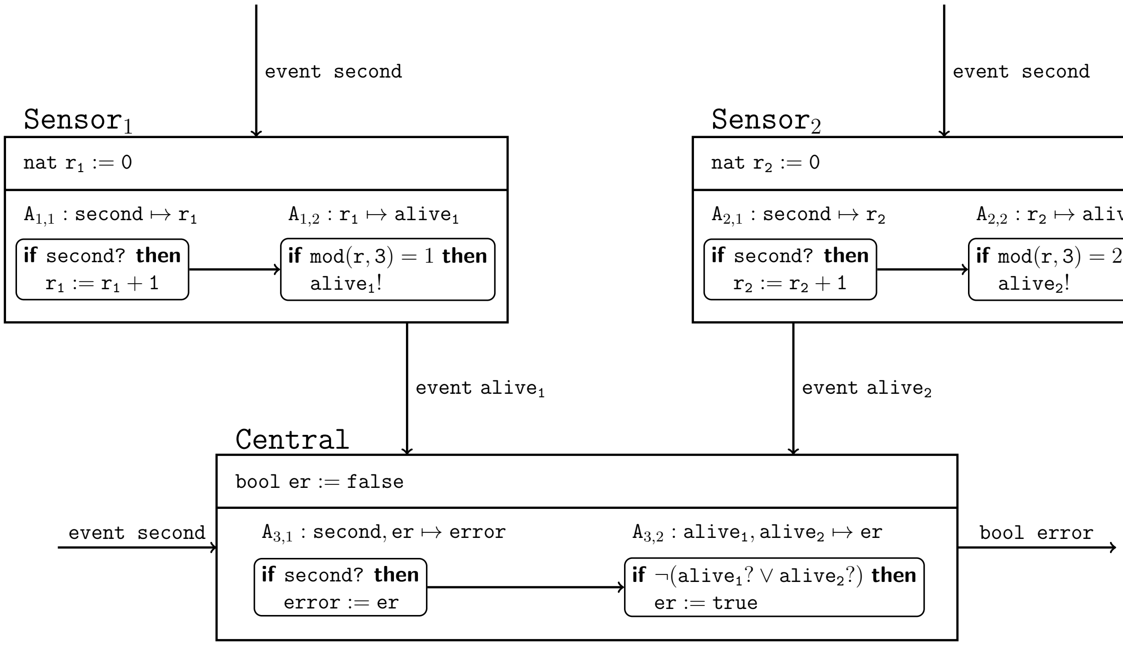
Please read the following before solving the exercises.

- 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`.


event second
event second
Sensor$_1$
nat r$_1$:= 0
A$_{1,1}$: second ↦ r$_1$
A$_{1,2}$: r$_1$ ↦ alive$_1$
if second? then r$_1$:= r$_1$ + 1
if mod(r, 3) = 1 then alive$_1$!
Sensor$_2$
nat r$_2$:= 0
A$_{2,1}$: second ↦ r$_2$
A$_{2,2}$: r$_2$ ↦ alive$_2$
if second? then r$_2$:= r$_2$ + 1
if mod(r, 3) = 2 then alive$_2$!
event alive$_1$
event alive$_2$
Central
bool er := false
event second
A$_{3,1}$: second, er ↦ error
A$_{3,2}$: alive$_1$, alive$_2$ ↦ er
if second? then error := er
if ¬(alive$_1$? ∨ alive$_2$?) then er := true
bool error

(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 ....

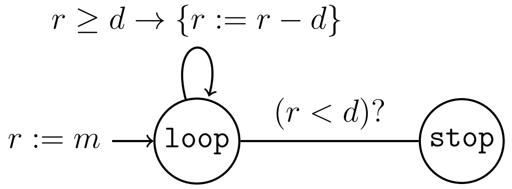
(a)
- $S = \{\texttt{er}, \texttt{r}_1, \texttt{r}_2, \}$
- $O = \{\texttt{error}, \texttt{alive}_1, \texttt{alive}_2\}$
- $I = \{\texttt{second}\}$
- $\prec = \{(A_{1,1}, A_{1,2}), (A_{2,1}, A_{2,2}), (A_{3,1}, A_{3,2}), (A_{1,2}, A_{3,2}), (A_{2,2}, A_{3,2})\}$

(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:

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

(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

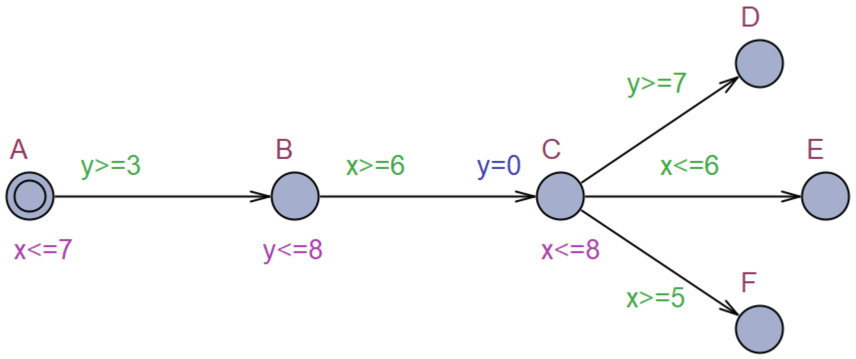
(a) $n = 2$

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

(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)

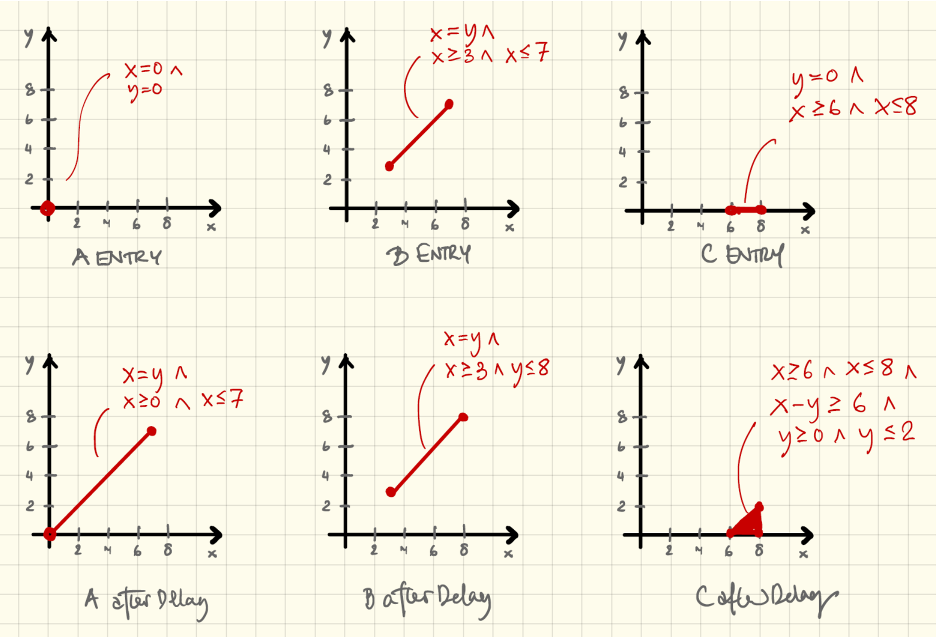
$$(\mathtt{A}, x = 0, y = 0)) \xrightarrow{6} (\mathtt{A}, x = 6, y = 6)) \to (\mathtt{B}, x = 6, y = 6)) \to (\mathtt{C}, x = 6, y = 0)) \to (\mathtt{F}, x = 6, y = 0))$$

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.

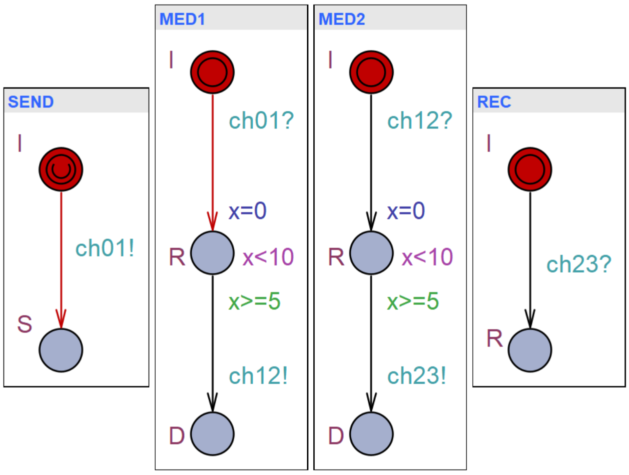


Figure 1: A small Communication System

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.

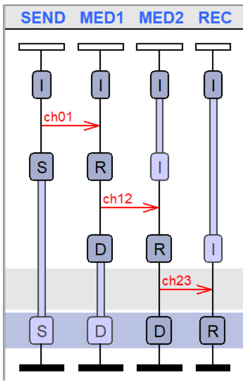
(b) Check in UPPAAL whether the properties ϕ_C and ϕ_M are satisfied by the modified model. If not use UPPAAL to generate a violating trace, and provide the corresponding MSC found in the simulator of UPPAAL.

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

. . . . Solution . . . .

(a) ϕ_C = `E<> time<= 15 and REC.R`

(b) The property holds.



(c) 10

(d) ϕ_M = `A<> time>=30 and REC.R`

(e) The property holds

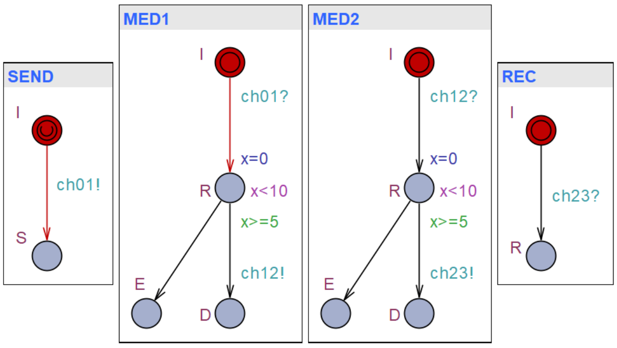
(f) 20

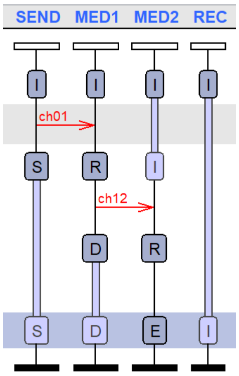
(g) Reachable combinations of locations: (I,I), (R,I), (D,R), (D,D).

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: 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:

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

(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 $\frac{d}{dt}x(t) = Ax(t)$, variable x denotes a vector of the form $x(t) = \binom{x_1}{x_2}$

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. By (a), in turn, 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.) From the course, we even know that the solution for such an initial condition will be given by $x(t) = (e^{-2t}2a, e^{-2t}a)$. One can cross-check that this is indeed the case by differentiating the expression:

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

Since $e^{-2t} \to 0$ as $t \to \infty$, we thus see that the solution of $\frac{d}{dt}x(t) = Ax(t)$ starting at $x(0) = (2a, a)^T$ indeed converges to zero as time goes by.

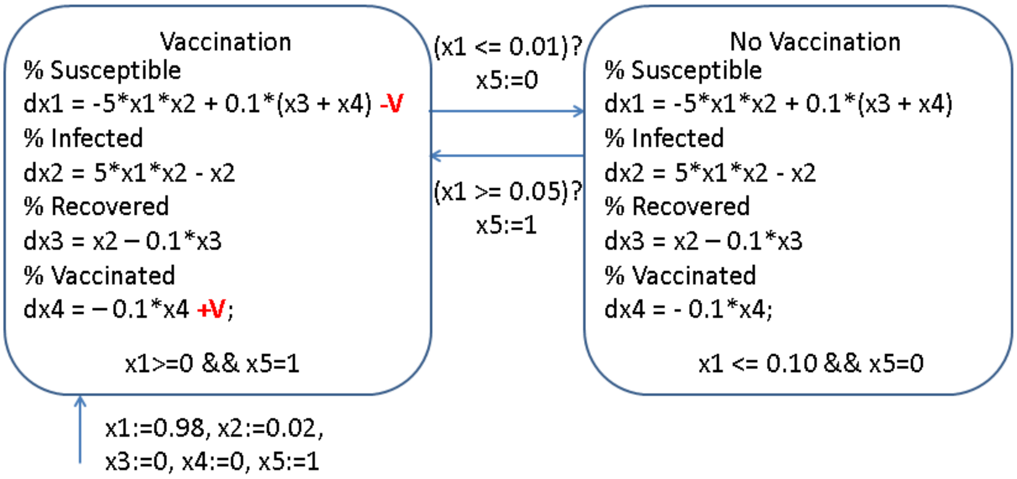


Figure 2: Hybrid model of a vaccination program.

Exercise 7: Hybrid Vaccination Model 10 Points

Consider the hybrid system of Figure 2 and the following incomplete MATLAB script below which can be found in Digital Exam:

Listing 5: Incomplete MATLAB script `vaccination.m`

```
function vaccination()
    T = 5.0;
    % Initial percentages of susceptible, infected, recovered, vaccinated and the initial mode
    x0 = [0.98, 0.02, 0.00, 0.00, 1]';
    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)
    dx = zeros(5,1);

    % ... add code ...

    if(x(5) == 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()
    T = 5.0;
    % Initial percentages of susceptible, infected, recovered, vaccinated and the initial mode
    x0 = [0.98, 0.02, 0.00, 0.00, 1]';
    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)
    dx = zeros(5,1);

    % added code
    if(x(5) == 1 && x(1) <= 0.01 )
        x(5) = 0;
    elseif(x(5) == 0 && x(1) >= 0.05)
        x(5) = 1;
    end

    if(x(5) == 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
```