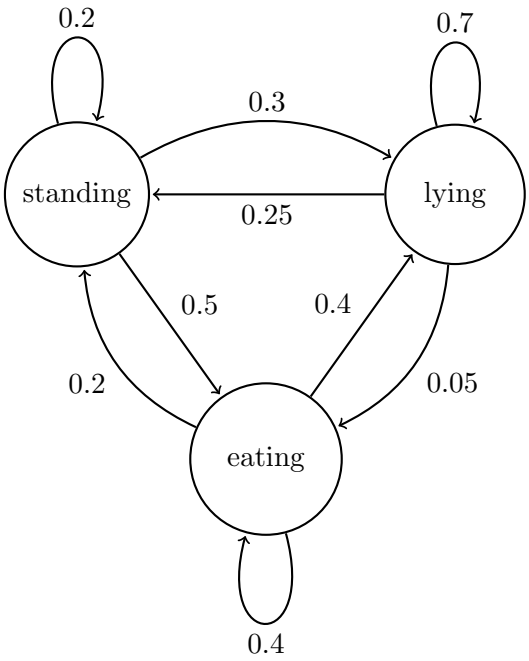
Exercise 4: Markov Processes, Cellular Automata and Fuzzy Logic

Formal Methods II, Fall Semester 2013

Solution Sheet

Markov Processes – Theoretical Exercise

1. (a) (1 point)



(b) (2 points) The transition matrix in the order (standing, lying, eating), using the notation from the script is as follows:

$$\mathbf{P} = \begin{pmatrix} 0.2 & 0.25 & 0.2 \\ 0.3 & 0.7 & 0.4 \\ 0.5 & 0.05 & 0.4 \end{pmatrix}$$

The stationary distribution vector $\mathbf{x}^*$ must fulfill the following requirement:

$$\mathbf{x}^* = \mathbf{P} \cdot \mathbf{x}^*$$

In other terms, the distribution shall not change from state X_{n-1} to state X_n. Using $\sum_i x_i = 1$ as constraint (the sum of all probabilities must equal 1), we obtain:

$$\mathbf{x}^* = \mathbf{P}\mathbf{x}^* = \mathbf{P}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \mathbf{P}\begin{pmatrix} x_1 \\ x_2 \\ 1 - x_1 - x_2 \end{pmatrix} = \begin{pmatrix} 0.2 & 0.25 & 0.2 \\ 0.3 & 0.7 & 0.4 \\ 0.5 & 0.05 & 0.4 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ 1 - x_1 - x_2 \end{pmatrix}$$

This equation can also be formulated in terms of the following linear equation system:

$$\begin{aligned} x_1 &= 0.2x_1 + 0.25x_2 + 0.2(1 - x_1 - x_2) \\ x_2 &= 0.3x_1 + 0.7x_2 + 0.4(1 - x_1 - x_2) \\ x_3 &= 0.5x_1 + 0.05x_2 + 0.4(1 - x_1 - x_2) \end{aligned}$$

which can easily be solved using standard techniques.

Let's have a look at an alternative way of solving the problem: If you compare the definition of the stationary distribution $\mathbf{x}^* = \mathbf{P} \cdot \mathbf{x}^*$ with the eigenvalue problem

$$\mathbf{P}\mathbf{v} = \lambda\mathbf{v}$$

you will notice some similarities. In particular, if you substitute $\mathbf{x}^*$ for $\mathbf{v}$ and set $\lambda = 1$, you will realize that the stationary distribution is in fact an eigenvector of $\mathbf{P}$ with a corresponding eigenvalue of $\lambda = 1$. Scaling the eigenvector to $|\mathbf{v}| = 1$ ($v_i > 0$), leads to the stationary distribution. The following octave[1] code shows how this could be done:

```
P=[0.2,0.25,0.2; 0.3,0.7,0.4; 0.5,0.05,0.4]
[Q,L] = eig(P)
x = Q(:,1)/sum(Q(:,1))
```

Using either one of the methods, you should arrive at a stationary distribution of:

$$\mathbf{x}^* \approx \begin{pmatrix} 0.227 \\ 0.539 \\ 0.234 \end{pmatrix} = \begin{pmatrix} 22.7\% \\ 53.9\% \\ 23.4\% \end{pmatrix}$$

Hidden Markov Models – Practical exercise

2. (5 points) The key to solving this question is to realize that the hidden states are the letters that are *meant* to be typed and the emission tokens are the letters that have been typed. Therefore, it is only necessary to introduce emission tokens with sensible emission probabilities for the letters. For example, 1 could be used for the correct letter and 0.3 for the surrounding letters. The algorithm itself does not require any changes.

[1] An open source clone of MATLAB. The above code also runs in MATLAB.

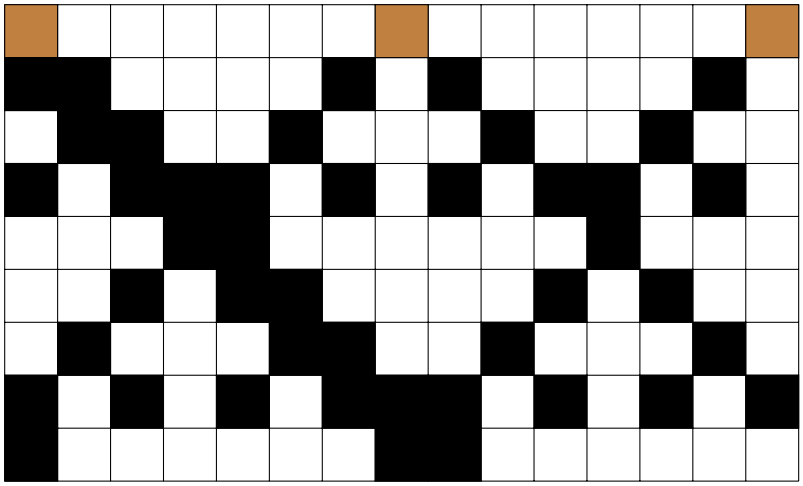
Cellular Automata

3. First, the decimal rule number 210_{10} has to be converted to binary, which leads to 11010010_2.

   (a) (1 point) This binary number is then filled into the rule table, maintaining the proper order (the least significant bit of the binary number always corresponds to $(0, 0, 0)$):

a_{n-1}^t	a_n^t	a_{n+1}^t	a_n^{t+1}
1	1	1	**1**
1	1	0	**1**
1	0	1	**0**
1	0	0	**1**
0	1	1	**0**
0	1	0	**0**
0	0	1	**1**
0	0	0	**0**

   (b) (2 points) The grid can then be drawn by checking the rule table. Note the boundary condition.



   (c) (2 bonus points) The inverted pattern is produced using rule 180. The rule number can intuitively be found by inverting the rule table found in part (a), and thus then reading the rule number in the reverse direction (note that this now is the "correct" order):

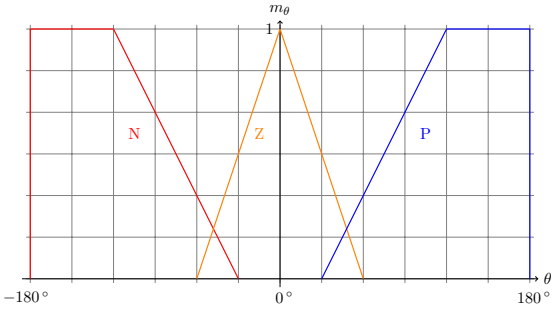
$$\begin{aligned} 210_{10} &= 11010010_2 && \text{invert} \\ &\rightarrow 00101101_2 && \text{reverse} \\ &\rightarrow 10110100_2 = 180_{10} \end{aligned}$$

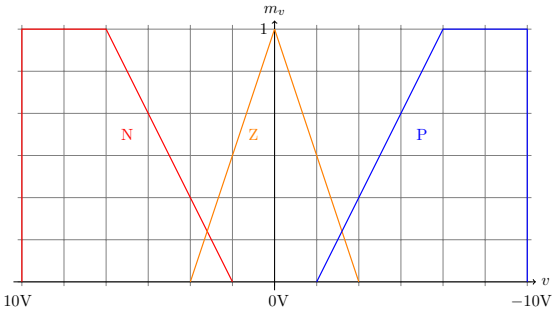
4. (4 points, 0.5 for each correct answer, no penalty)

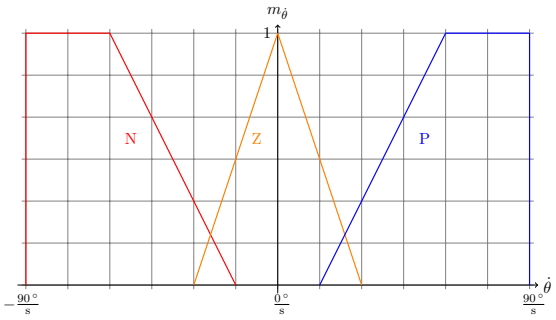
	t	**f**
Each cell of a CA can be looked upon as a Finite State Machine.	☑	☐
In 1D CAs, the complexity of the emerging patterns depends mostly on the neighborhood radius r.	☐	☑
Given an arbitrary elementary CA (with rule table) and the cell states at iteration i, the initial configuration can always be computed.	☐	☑
Given the cell states of a 1D CA at arbitrary iterations i and $i+1$ one can in some cases reproduce the rule table.	☑	☐
In 2D CAs, the Moore neighborhood consists of a central cell and 4 neighbors whereas the von Neumann neighborhood consists of the central cell and 8 neighbors.	☐	☑
The rule table of a 1D CA with fixed boundary conditions contains fewer rules than the rule table of a 1D CA with cyclic boundary conditions.	☐	☑
CAs can be used to simulate processes from e.g. physics and biology.	☑	☐
A majority of CA rules are capable of universal computation.	☐	☑

Fuzzy Logic

5. (a) (1 point) Membership functions for input variables θ and $\dot{\theta}$ as well as the output variable v:







There is no unique solution to this task. The particular functions are the designer's choice, that most probably have to be fine tuned by experimenting with the controller either in simulations or in the actual experimental setup.

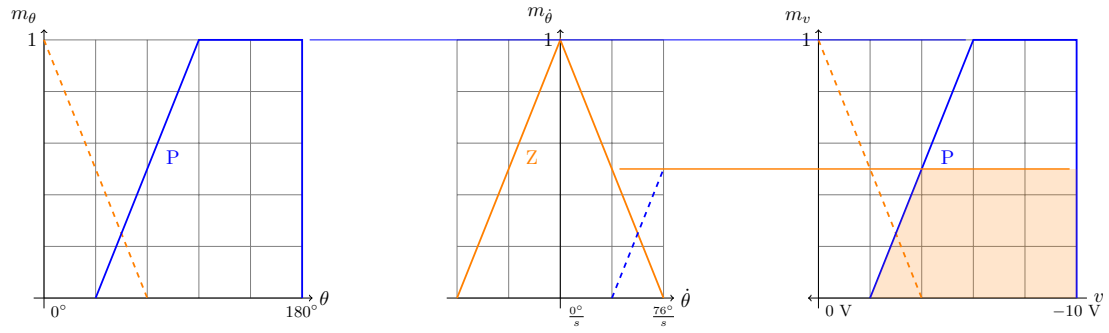
Note that the direction of the axis for the voltage is somewhat strange. This is to comply with the nomenclature in the additional lecture notes on fuzzy logic where the direction of the cart is counter- intuitive too.

(b) (2 points) One possible rule table:

$\dot{\theta}\backslash\theta$	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

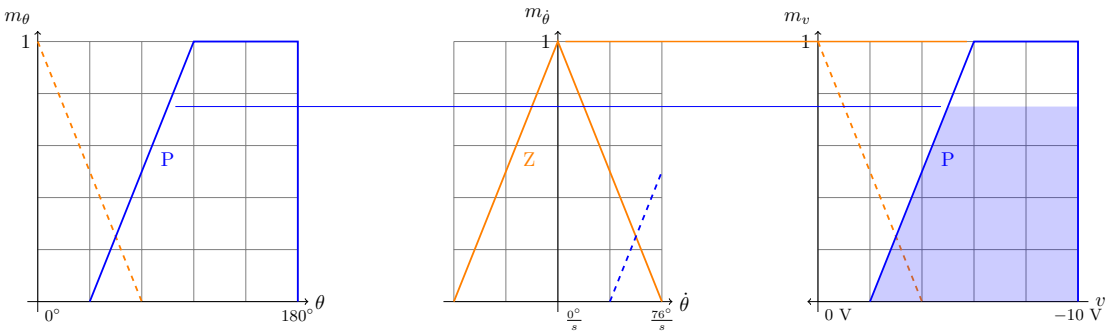
There might be other acceptable solutions depending on how they are justified.

(c) (2 points) In each cycle/period, the kinetic energy is maximal when the pendulum is passing the position at the "bottom" (stable equilibrium point), i.e. $\theta = 180\,^\circ$. We assume $\dot{\theta} = \frac{36\,^\circ}{s}$ for simplicity, although that value is supposed to increase during the swing-up process. The only sets to consider are P_θ and $\mathrm{Z}_{\dot{\theta}}$ because all other membership functions are 0, so the rule (P, Z; P) has to be activated. At this point, the geometrical output function is constructed as follows:



The last step is to compute the centroid of the polygon shaded in orange. One can easily see that this polygon can be split up into a triangle at the left and a rectangle at the right. The area of the rectangle is 3 and the v-coordinate of its centroid is 7. The triangle has an area of 0.5 and the v-coordinate of its centroid is roughly at -3.33. By geometric decomposition, the v-coordinate of the polygon is therefore approximately -6.56 V.

On the other hand, the potential energy is maximal in each cycle at the point at which the velocity changes sign, that is $\dot\theta = 0\,°$. Let us assume $\theta = 90\,°$. (Again, this value is supposed to increase from cycle to cycle. As in the previous example, the only rule that is to be considered is (P, Z; P):



The centroid of the polygon shaded in blue has a v-coordinate of roughly -6.