

Homework 3 - maze

In this homework, we will solve mazes using a model checker. A maze is given as a 2D grid. The shape of the corridors is given by walls, denoted by w . Letter m stands for a mouse that is longing for a cheese c . In every step, the mouse can move up, down, left, or right (it cannot stand in the same position unless it is trapped; see below). The mouse cannot move onto a wall position. There could also be doors $d1, d2, \dots$ that are initially locked, and the mouse can pass each door only if the corresponding key $k1, k2, \dots$ has been already found (visited). Without the corresponding key, the mouse is not allowed to enter the position with the door. In the maze, there could be traps t also. If the mouse stands on a trap, the game is over, as it cannot move anymore. Moreover, there could be poisons p and antidotes a . Contrary to the keys and doors, poisons and antidotes are all of the same types. When the mouse visits a poison, the poison is eaten (it disappears), and the mouse is poisoned. The poisoned mouse can survive if an antidote position is visited in at most three subsequent steps. Contrary to keys, the mouse cannot take antidote with; hence, visiting antidote before a poison does not help. Contrary to poison, antidotes are never eaten. For simplicity, let us assume that in each position, there is at most one thing. For an example of a maze, see Figure 1.

w	w	w	w	w	w	w	w	w	w
w					k2	w		w	w
w		w	w	w		w		k1	w
w			w			d1		w	w
w		t	d2		w	w		w	w
w	d3	w	w	w	w	w		w	w
w	p		a	w				w	w
w		w	w	w	w		w	w	w
w		w		c	w		w	m	w
w		w			w		w		w
w					t	k3			w
w	w	w	w	w	w	w	w	w	w

(a) maze01.csv

w	w	w	w	w	w	w	w	w	w
w					k1	w		w	w
w		w	w	w		w		k2	w
w			w			d1		w	w
w		t	d2		w	w		w	w
w	d3	w	w	w	w	w		w	w
w	p		a	w				w	w
w		w	w	w	w		w	w	w
w		w		c	w		w	m	w
w		w			w		w		w
w					t	k3			w
w	w	w	w	w	w	w	w	w	w

(b) maze02.csv

w	w	w	w	w	w	w	w	w	w
w	m	k1					d1	c	w
w	w	w	w	w	w	w	w	w	w

(c) maze03.csv

w	w	w	w	w	w	w	w	w	w
w	m		d1		k1			c	w
w	w	w	w	w	w	w	w	w	w

(d) maze04.csv

Figure 1: Examples of mazes.

Each maze is specified in a csv file. Your task is to write a script or a short program that transforms the csv file onto a *nuXmv* model with CTL or LTL formulae checking the following properties. Keep in mind all the restrictions about obstacles on the path, like that the door must be unlocked with the

corresponding key, etc.

- P1 It may happen that the mouse won't run into any item (key, door, poison, cheese, antidote, trap).
- P2 Whenever the mouse eats a poison, it will survive.
- P3 Whenever the mouse eats a poison, it can survive.
- P4 It can happen that the mouse eats a poison and survives.
- P5 Whenever the mouse enters the position with door $d1$, it will eventually be trapped.
- P6 Whenever the mouse enters the position with door $d1$, then there is a free path to the cheese. A free path is a path without obstacles like doors (locked or unlocked), traps, or poisons.
- P7 The door $d1$ position is entered, only if key $k1$ has been already found.
- P8 The mouse can come at door $d1$ (i.e., to visit the neighboring position).
- P9 The mouse can enter the position of door $d1$.
- P10 There exists a path on which the mouse gets the cheese.

The formulae must be generic, which means that if a new maze is generated, the formulae stay unchanged and still may be used to decide whether the given property holds. There are some predefined mazes with solutions. You are welcome to write your own testing formulae as well as new mazes (and submit them with your solution). Test your solution on nymfeXY.fi.muni.cz before the submission. It is recommended to **start with manually built solutions first**.

When you are sure that it works well, start working on your smv-maze generator. Edit the **predefined Makefile** to work correctly with **your script** and specify the formulae corresponding to individual properties (read the comments in Makefile). Explain the basic ideas of your construction briefly in a separate **PDF report**. In case of ambiguity or opacity, feel free to ask by email (rehak@fi.muni.cz) or come to seminars/consultations.

Hint: Instead of checking whether the model satisfies a formula, you may also create an LTL formula for which the model checker generates a counter-example witnessing that there exists a specific path. E.g., write a counter-generating formula for $P9$ that is *false* if and only if the property $P9$ holds. In this case, mention explicitly that these formulae are counter-generating in the PDF and carefully assign indexes to Nx instead of Px in your Makefile.