Model Checking SS25
Assignment 10
Due: June 3th, 2025, 09:00

For the last assignment sheets we will cover topics regarding *probabilistic model checking*. For this assignment sheet we will use the `prism` -modelling-language to describe two models and evaluate them with the probabilistic model checker `storm`. We recommend using one of the provided docker images, specifically the `debug` version:

```
docker pull movesrwth/storm:ci-debug
```

You can then either start a container and mount your local directory into it:

```
docker run -v $(pwd):/media -it movesrwth/storm:ci-debug
/opt/storm/build/bin/storm --prism <my_prism_file>
```

A different way would be to use the container in a *one-shot* way:

```
docker run -v $(pwd):/media -it movesrwth/storm:ci-debug\
/bin/sh -c '/opt/storm/build/bin/storm --prism /media/<my_prism_file>
```

Helpful and important command line flags for storm are:

- `--prism`: Pass a `prism` model file.
- `--prop`: Pass a property to check or a list of properties in a text file.
- `--debug`: Enable verbose debug output.
- `--explchecks`: Enable explanations for model building failures.
- `--build-all-labels`: Build all labels and reward structures.
- `--buildstateval`: Build all state valuations.
- `--exportresult`: Export the results into a json file.
- `--exportdot`: Export built model into dot file (follow-up with `dot -O -Tpdf <file>.dot`).
- `--help`: Print a list of all command line flags and exit.

Have fun!

1. **[50 Points]** Consider the communication protocol we have discussed in class. We will adapt the model of the protocol to drop messages after they have been lost. Whenever the model enters the `lost` state, with a chance of $\frac{1}{2}$ the protocol will drop the message and we enter the sink state `gone`.

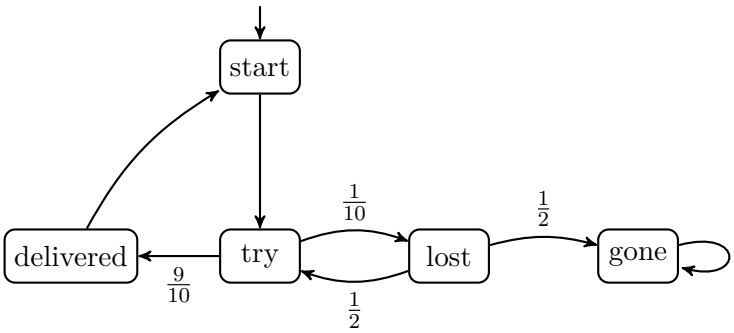

Figure 1: Communication protocol with a probability to drop messages.

This exercise is split in two parts:

a) Adapt the `.prism`-file that we have created in class to model the probability to drop messages.

b) Calculate the probability to eventually deliver the message using the method discussed in class. You can skip the graph-based analysis and let $S_{=0} = \{\text{gone}\}$ and $S_{=1} = \{\text{delivered}\}$.

Hand in your model in the **PRISM** language for subtask 1a and the linear equation system $(I - A_?) \cdot x = b$, as well as the results for x for subtask 1b.

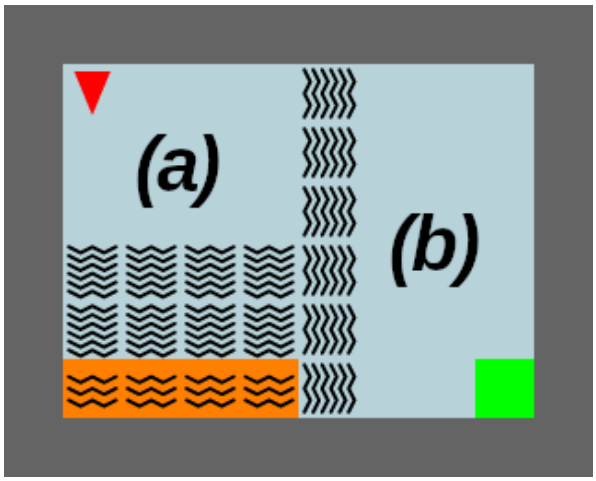

Figure 2: A MiniGrid environment.

2. **[50 Points]** Consider the *MiniGrid* environment depicted in Figure 2. As in the lecture, the different fields imply different behaviour:

   - An orange tile means lava. Stepping onto it, causes the robot to melt and be inoperable.
   - Squiggly lines are cliffs which cannot be climbed again.
   - Blue tiles imply slippery behaviour:
     (a) The robot slips to the right with a probability of 0.5 or up and down with a probability of your choice.
     (b) The robot slips uniformly random in any direction.

   Model this environment in the **PRISM**-language and hand in your text file containing the model. You may use the following snippet a starting point.

```
dtmc

// Use formulas and labels to help us understand your model

module MiniGrid
// Define variables and all the needed commands


endmodule
```

   After modelling the problem use `storm` to compute the probability of reaching the goal for *all states*. Please hand in your resulting `json`-file.

   Don't hesitate to discuss your ideas to model this problem with us and your colleagues via Discord!