Model Checking SS24
Assignment 9

Due: June 17th, 2024, 09:00

For the last assignment sheets we will cover topics regarding *probabilistic model checking*. The tasks will mainly focus on modelling abstractions using the `prism` -modelling-language and evaluating your results and/or computing some results on pen and paper.

In order to evaluate your models we will use 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.
- `--buildfull`: Build all labels and reward structures.
- `--help`: Print a list of all command line flags and exit.

We will populate a git repository with the examples that we have and will code together curing class here:

https://git.pranger.xyz/sp/MC-ProbMC-PrismFiles

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`.

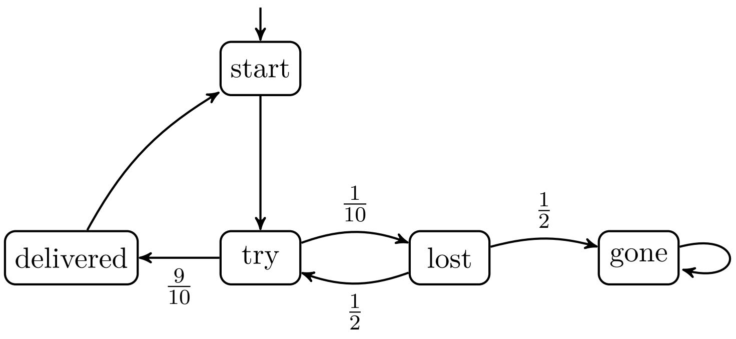

Figure 1: Communication protocol with a probability to drop messages.

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.

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.

2. [**50 Points**] We consider the following scenario. Three Cowboys: "The Good", "The Bad", and "The Ugly" meet each other in the desert for a famous duel.

   - The three may shoot as long as anyone else is still alive. Due to differences in (re)loading times, we assume that they shoot in turns. That is, The Good shoots first, then The Bad and finally The Ugly.
   - The Good has a chance of a half of hitting anyone. If he hits, he does so uniformly over the living contestants.
   - The Bad has a chance of 0.9 of hitting anyone. If The Ugly is alive, then he aims for him. If The Ugly already died, then he aims at The Good.
   - The Ugly hits either no one or one of the living contestants and he does so with a uniform probability over these events.

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

```
dtmc

module shootout
  # define needed variables

  # define commands that model the different events described in the text
endmodule
```

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