Model Checking SS23
Assignment 11

Due: June 30th, 2023, 18:00

For this last assignment sheet we will further strengthen our skills using `storm` and `stormpy`.

1. [**10 Points**] Microgrid Demand Side Management.

   For this exercise we will model an abstract version of the algorithm proposed by Hildmann and Saffre. Please also refer to the Prism Case Study that evaluated the algorithm using *stochastic multiplayer games*.

   We are going to model the following using a Markov Decision Process: There are two actors the this scenario, a *manager* that controls the price for drawing energy from the grid and multiple *households* that can schedule a energy hungry task. Whether a household will start to execute its task is dependent on the current market price for energy:

   - If the price is below a fixed price limit the household will start its task in the next timestep.
   - If the price is above this price limit the household will only start its task dependent on the current price. This will be used to steer the households such that peaks in the power draw will be reduced.

   Your task is to implement the behaviour of the manager and the household. A day in our model is split into 16 time units and the probability to schedule the task is modelled after this curve using `P_init`.

   One time unit will follow a fixed process:

   (a) `checkIfShouldSchedule`: All households simultaneously check whether they want to schedule the task. They do so with a probability of `P_init`. This is only possible if they have not already scheduled the task. Household that have a task scheduled or are currently executing a task will continue to do so.

   (b) `updatePrice`: The households and the manager synchronize, such that the manager can update the price and 'broadcast' it to the households. Note that the household simply have to increment the `move` variable for this action.

   Apart from updating the price, the manager will also increment the time unit modulo `K`.

   (c) `checkIfShouldExecute`: The households check if they should execute the task given they have scheduled it. If the current price is below the fixed limit they will execute the task with probability one, otherwise they will execute it with a probability of $1 - \text{householdAbide}$.

   If a household executes a job, we set its `power_draw` to 4 and the `job_status` to 4. Also be sure to update the boolean flags accordingly.

   A household that did not schedule the task needs to increments its `mve` variable here.

(d) `updateJobStatus`: If a households `job_status` is greater 0 it will decrement this variable by one. If the status reaches 0 the household will stop executing the task and set its draw to 0.

Use the `move` variable to orchestrate the behaviour described above and introduce atleast a second household using module renaming

```
sudo docker run --rm -v $(pwd):/media -it movesrwth/stormpy /bin/sh
-c "python3 /media/mgm_simulator.py"
```

After you have successfully modelled the microgrid management you need to evaluate the algorithm. You are going to use the provided simulator script.

Evaluate the following scenarios:

(a) Report the maximum waiting time, average draw and average revenue per time step using the following properties:
   - `"Pmax=? [ G (totalDraw < 5) ];"`
   - `"R{"energyPriceAndSatisfaction"}max=? [ LRA ];"`
   - `"R{"keepStableDraw"}max=? [ LRA ];"` Introduce a new reward structure in the prism-file and an according model checking call in the simulator file to keep the total draw below 4 on the average.

(b) For all of the different schedulers, report the following by model checking properties on the induced Markov chain:
   - What is the probability to reject two requests at the same time.
   - Will the energy price ever reach 10?
   - What is the probability that a household will execute the task within 9 timesteps after it has been rejected in checkIfShouldExecute?

Have fun!