Model Checking 2025 - Projects

Linus Heck, Eline Bovy, and Nils Jansen
linus.heck@ru.nl, eline.bovy@ru.nl, n.jansen@science.ru.nl

General instructions

Select one project from the options given below. Each project should be made in groups of two. All reports should be in the NeurIPS 2024 format, see `https://www.overleaf.com/latex/templates/neurips-2024/tpsbbrdqcmsh` for a template.[1]

If you have any questions, do not hesitate to contact us!

Deadline for project choice: May 2 Register for one of the projects via Brightspace. Note that there are limited spots per project. The spots are divided on a first-come, first-served basis.

Deadline full project: June 13 Hand in the full project on Brightspace.

Short 10 **minute presentations: July 8**

Deliverables

- A report of around 6 pages, excluding figures and references, in the NeurIPS 2024 format. The report should answer the questions in detail and explain the decisions you made while solving the tasks.
- All code used for the project (Prism files, Python files, Jupyter notebooks).

[1] The tex file in the template contains a NeurIPS checklist which you can immediately remove. The rest of the content in the tex file explains the format guidelines and can also be removed. The format guidelines are also explained in the pdf.

1 Qwixx

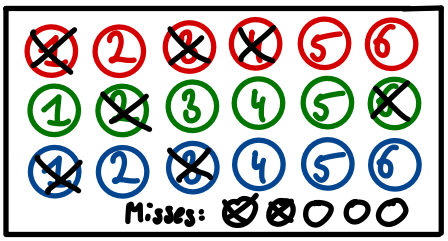

Figure 1: Example Qwixx game situation.

We consider a simplified version of the game Qwixx[2]. There is one player. They have a scoring sheet with three rows of six numbers. Each round, they throw six fair coins and count the resulting number of heads (we call this value x). They are now able to perform one of three actions:

- Cross off a number: If there is no cross on x and no cross on any number greater than x in some row, they can cross off x on that row.
- Cross off a miss: If a miss is available, they can cross off a miss. There are five misses the player can cross in one game.
- Give up: Instead of crossing off anything, the player can always give up and finish the game.

Example 1. *Consider the following game situation depicted in Figure 1. Suppose the player threw five heads. They cannot cross off the five in row two, because the six is already crossed. They can cross off the five in row one, cross off the five in row three, cross off another miss, or give up.*

Formulating and Checking the Model

Exercise 1 Model this game as the appropriate Markov model (e.g., using the PRISM language). Using a probabilistic model checker (e.g., Storm), check that the probability (with the best strategy) to get at least three crosses in row one is around 84.792%. What is the probability, with the best strategy, to get six crosses in row one? How does this relate to the probability to get six crosses in any row?

All games eventually end. The *points* are counted as follows:

- For each row, the player gets points for crossed off numbers. For one cross, they get one point, for two crosses, they get $1 + 2 = 3$ points, for three crosses, they get $1 + 2 + 3 = 6$ points, and so on.
- The player gets negative three points for each crossed-off miss.

[2] `https://de.wikipedia.org/wiki/Qwixx` (only available in German)

Exercise 2 Check that the minimal probability for the game to end in your model is one, i.e., the game always ends. Amend your model to be able to count points—note that you can calculate the points out of the number of crosses and misses *at the end of the game*. With the best strategy, what is the number of expected points?

Exercise 3 Export the scheduler that maximises the number of expected points. This is a large JSON file[3]. Visualize some aspect of this scheduler—e.g., the taken actions, the number of crosses at the end, etc.

Learning Probabilities

Assume that we are not sure about the fairness of one specific coin, let's say coin 1. We want to estimate the true distribution of coin 1.

Exercise 4 Use the Stormpy simulator to generate data points for the probability that throwing coin 1 results in heads at the first coin toss of the game.

1. Generate at least 50 data points. Compute the estimated probability distribution and distance bound of a coin toss with coin 1 according to step 1 of the UCRL2 algorithm. Use confidence parameter 0.6.
2. Generate another 50 data points. Update the estimated probability distribution and distance bound of a coin toss with coin 1 using the UCRL2 algorithm. Note that a random scheduler can be used to sample new data points as the coin toss probabilities are independent of the actions.

Qwixx Against Time

A normal Qwixx round would be played against other players, but we'll just play against time.

Exercise 5 Count the number of rounds (i.e., coin flips) in a game. What is the maximal expected number of rounds?

Exercise 6 Describe multi-objective model checking from the provided documentation.[4] Suppose we want to minimize our expected number of rounds while maximizing our expected score. Plot the Pareto curve that represents this query.

[3] If you are using the Storm CLI, you can export the result vector for all states instead of just the initial state by using the property `filter(values, [your property] , true)` and the CLI flags `--buildstateval --buildfull --build-all-labels --buildchoicelab --exportscheduler sched.json`.

[4] `https://moves.rwth-aachen.de/wp-content/uploads/WS1819/mvps/lec23.pdf`

2 Airport Taxi

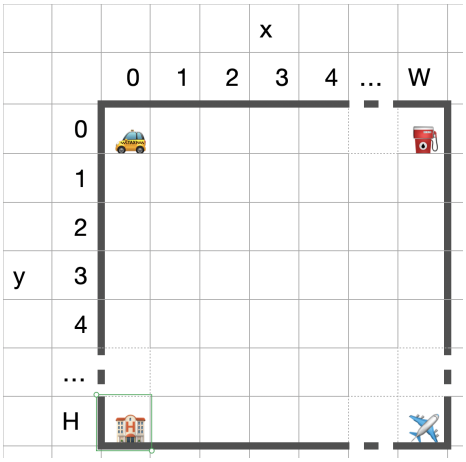

Figure 2: Taxi environment.

You are operating a taxi in the environment depicted in Figure 2. A grid of dimensionality N has two-dimensional coordinates that each range from 0 to $N-1$. The environment is set up as follows:

- The taxi starts at $(0, 0)$.
- There is a hotel at $(0, N-1)$.
- There is a gas station at $(N-1, 0)$.
- The airport is at $(N-1, N-1)$.

The taxi has the following properties:

- The positions x, y of the taxi, which start at $(0, 0)$.
- The fuel of the taxi, which starts at its fuel capacity F.
- Whether a customer has called from the hotel. Only one customer can call at a time.
- Whether a customer is currently sitting in the taxi. Only one customer can sit in the taxi at a time.

Each time step has three phases:

Phase 1. The taxi moves in a direction it chooses (but it cannot fall off the grid). Its fuel decreases by one.

Phase 2.

- If the taxi is at the fuel pump, its fuel is set to the fuel capacity.
- If the taxi is at the hotel, a customer has called, and no customer is currently in the taxi, that customer now sits in the taxi.
- If the taxi is at the airport and there is a customer in the taxi, the customer is successfully delivered.
- Otherwise, nothing happens here.

Phase 3.

- If the taxi's fuel tank is empty, it can no longer move.
- Otherwise, a customer calls with probability p. Go back to Phase 1.

Formulating and Checking the Model

Exercise 1 Model this environment as the appropriate Markov model (e.g., using the PRISM language). Suppose the grid size is seven, p is 0.1, and the fuel capacity is 30. Using a probabilistic model checker (e.g., Storm), verify that the expected number of customers delivered to the hotel in 200 steps with the best strategy is about eight.[5]

Exercise 2 Plot the maximal expected number of customers delivered for all fuel capacities between one and 100 with the constants above. What do you see in the plot? How can you explain it?

Exercise 3. Export the scheduler that maximises the number of expected points. This is a large JSON file [6]. Visualize some aspect of this scheduler—e.g., the taken actions, the number of crosses at the end, etc.

Learning Probabilities

We want to estimate the probability that a customer calls the taxi.

Exercise 4 Use the Stormpy simulator to generate data points for the probability that a customer calls the taxi. Use a fuel capacity of 30.

1. Generate at least 100 data points. Use PAC learning to compute an interval for the requested probability. Make sure that the true probability has

[5] Hint: You need to additionally formulate that the taxi can only take 200 steps.

[6] If you are using the Storm CLI, you can export the result vector for all states instead of just the initial state by using the property `filter(values, [your property] , true)` and the CLI flags `--buildstateval --buildfull --build-all-labels --buildchoicelab --exportscheduler sched.json`.

95% chance of being contained in the interval. Use $\sum_{(s,a)} |Post(s,a)_{>1}| \approx 600000$.

2. How many data points would you need to get an interval for the requested probability of length at most 0.1?

Parametric Taxi

Exercise 5 Set the fuel capacity to 30 again. Now parametrize the probability that a customer will call. Now plot the maximal expected number of customers delivered as a function of p.

Exercise 6 Describe the Parameter Lifting Algorithm. Using this algorithm, prove that the expected reward is at most nine for all $p \in [0.01, 0.99]$. Why is the Parameter Lifting Algorithm able to prove this relatively quickly (within one refinement step)?

3 Teddy Bear Differential Diagnosis

Figure 3: Teddy bear is sick.

Oh no! The teddy bear is sick! They have exactly one of the following sicknesses:

1. Fever (probability 19%)
2. Homesickness (probability 18%)
3. Broken arm (probability 18%)
4. Afraid of the dark (probability 27%)
5. Has a cold (probability 8%)
6. Teddy bear is not sick (probability 10%)

The following tests exist with the following conclusions from positive outcomes:

1. Thermometer test: Teddy bear has fever or has a cold
2. Teddy bear feels scared: Teddy bear is afraid of the dark or is homesick
3. Teddy bear wants to drink some tea: Teddy bear is homesick or has a cold
4. X-Ray: Teddy bear has broken arm

All of the tests are always performed. Each test has a sensitivity (true positive rate) and specificity (true negative rate). If at the end, the only consistent possibility is the actual sickness that the teddy bear has, the teddy bear was correctly diagnosed. Otherwise, it is incorrectly diagnosed.

Formulating and Checking the Model

Exercise 1 For now, all tests have a sensitivity and specificity of 95%. Model the teddy bear as the appropriate Markov model (e.g., using the PRISM language). Using a probabilistic model checker (e.g., Storm), check the probability that the teddy bear is correctly diagnosed. In our model, this probability is around 85%.

Exercise 2: Teddy bear probabilities

- What is the probability that the teddy bear is not sick *and* is (incorrectly) diagnosed with some sickness?
- What is the possibility that the teddy bear has a fever but is (consistently) diagnosed with a cold?

Learning Probabilities

We want to estimate the probability of each sickness the teddy bear can have.

Exercise 3 Use the Stormpy simulator to generate data points for the distribution of the possible sicknesses.

1. Generate at least 50 data points. Assume the following prior intervals:
    1. Fever probability $\in [0.15, 0.35]$.
    2. Homesickness probability $\in [0.15, 0.3]$.
    3. Broken arm probability $\in [0.15, 0.2]$.
    4. Afraid of the dark probability $\in [0.22, 0.36]$.
    5. Has a cold probability $\in [0.5, 0.11]$.
    6. Teddy bear is not sick probability $\in [0.15, 0.35]$.

    Assume a strength interval of $[20, 30]$ for each sickness. Compute new probability intervals for the sickness distribution using LUI.
2. Generate another 50 data points. Update the probabilities computed in step 1 using LUI.

Parametric Teddy Bear

Exercise 4 Formulate the teddy bear model as a parametric model where you parametrize the test sensitivities and specificities.

Exercise 5 Describe the Parameter Lifting Algorithm. Using this algorithm, prove that for all sensitivity and specificity values between 85% and 99%, the probability that the teddy bear is diagnosed correctly is over 60%. Is it also over 70%?

Exercise 6 Now fix the sensitivities and specificity back to 95% and parametrize the probabilities that the teddy bear gets the sicknesses in the first place. Note that the probability that the teddy bear has no sickness is always the inverse probability of it having a sickness. Compute the symbolic expression that describes the probability that the teddy bear is correctly diagnosed as a function of these parameters. What can you see from this expression?

4 Network security

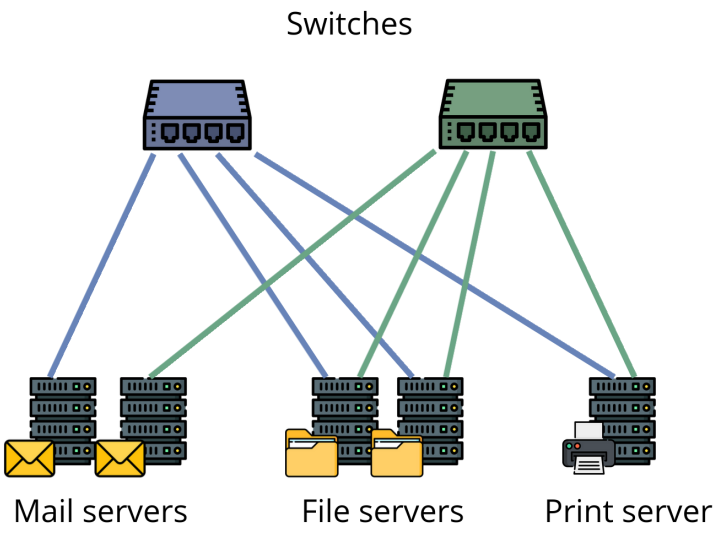

Figure 4: Network security

Consider the network in Figure 4, consisting of two network switches, two mail servers, two file servers, and a single print server. The network administrator must try to keep the system up and running. Unfortunately, the network is under attack. An adversary aims to compromise the network by sabotaging the network elements. If the adversary compromises a server, it immediately stops working and is considered `broken`. The switches are a bit more robust, and have two stages: `infected` and `broken`. Switches are still functional when infected.

The adversary attacks all network elements at once: each network element has an independent chance of being compromised at each time step.

- Switches have a 2% chance of going from `okay` to `infected`, but also a 1% chance of immediately going to `broken`. If a switch is already infected, there is a 2% chance of going to `broken`.
- Mail servers have a 3% chance of going from `okay` to `broken`.
- File servers have a 3% chance of going from `okay` to `broken`.
- The print server has a 2% chance of going from `okay` to `broken`.

If there is no server-switch pair left for a type of server, the network can no longer provide the service of that type to its users.

The network administrator has a few actions it can take:

- The administrator can repair a broken network element. The chance of

succeeding depends on the type of network element and the state of how broken it is.

  - A repair action on a switch has 50% chance of improving the state of the network element from `broken` to `infected` and a 45% chance of improving from `infected` to `okay`.
  - A repair action on a mail server has a 45% chance of succeeding.
  - A repair action on a file server has a 40% chance of succeeding.
  - A repair action on the print server has a 50% chance of succeeding.

  While the administrator is repairing a broken network element, attacks from the adversary have no effect.

- The administrator can inspect a broken network element to double the chance of a repair action. However, this only works as long as the inspected network element remains in the same condition, i.e., `infected` or `broken`.
- The administrator can also choose to do nothing.

Formulating and Checking the Model

Exercise 1 Model system as the appropriate Markov model (e.g., using the PRISM language). Using a probabilistic model checker (e.g., Storm), compute the probability that the network administrator can keep all the network's functionalities, i.e., mailing, printing, or accessing files, working for the first 24 time steps.

Exercise 2 Even with their best efforts, the administrator cannot prevent the network from losing functionality. To improve the system, the administrator can investigate which network elements cause the most problems.

1. For each type of functionality, compute the probability that this functionality stops working during the first 24 time steps despite the administrator's best efforts. Which functionality is hardest to keep up and running?
2. For each type of functionality, also compute the probability that this functionality stops working during the first 24 time steps if the administrator does nothing. Which network elements can the administrator best focus on? Explain why.

Exercise 3 Let's assume the network administrator has the budget to do two of the following upgrades:

1. Adding extra connections between the mail servers and switches, so that each switch is connected to each mail server.
2. Updating the switches so the probabilities of success for the attacker are halved.

3. Updating the file servers so the probability of success for the attacker is reduced to 2%.
4. Updating the mail servers so the probability of success for the attacker is reduced to 2%.
5. Updating the print servers so the probability of success for the attacker is reduced to 1%.

What is the best option with respect to minimizing the probability of losing any functionality within 24 time steps? With what probability can the network administrator now keep all the network's functionalities working for the first 24 time steps?

Learning Probabilities

We want to estimate the probability that the print server gets compromised during an attack.

Exercise 4 Use the Stormpy simulator to generate data points for the probability of the print server going from `okay` to `broken`. Use the probabilities of the original model. Generate at least 50 data points. Use frequentist estimation to compute a new probability for the print server going from `okay` to `broken`.

Partially Observable Network Maintenance

To make our problem a bit more realistic, we assume that the administrator cannot always observe whether a network element is sabotaged or not. More specifically, the administrator can only observe whether a functionality of the network still works or not. However, if the administrator inspects a network element, it can observe the state of the network element as well as double the repair probability for that network element.

Exercise 5 Model this problem as the appropriate Markov model (e.g., using the PRISM language). Discuss the changes you need to make to the model of Exercise 1.

Exercise 6 Assume that we start with an initial state where one of the switches and the mail server connected to the other switch are both `broken` and all other network components are `okay`. Draw part of the belief MDP for one step, only unfolding the wait action and the repair actions for the switches and the mail servers.

5 Wildlife tracking

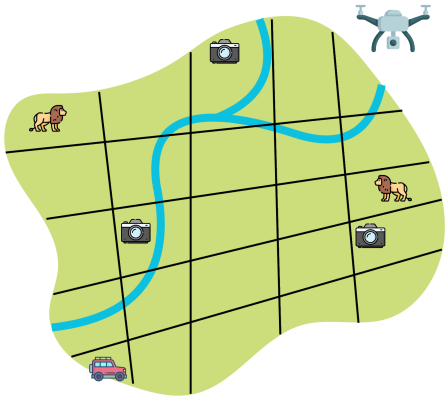

Figure 5: Wildlife tracking

We consider the problem of tracking the health of the lion population in a wildlife park. The park consists of 25 sectors, with the total lion population consisting of two separate tribes. The rangers have their home base in the bottom left sector of the park (see safari wagon in Figure 5). The rangers have various tools at their disposal to determine the health of the population.

- A drone keeps track of the tribes' movement through the sectors. At every time step, the drone reports the current placement of the tribes.
- A safari wagon that can travel to the various sections. At every time step, the rangers can use the car to travel to any edge-adjacent sector. Due to the poor quality of the roads, there is always a 20% chance of not reaching the next sector. If the rangers end up in the same sector as a tribe of lions, they can inspect the tribe's health with a 95% chance of succeeding.
- Movement cameras in three sectors (see cameras in Figure 5). The camera contains a picture of a tribe if the tribe has visited that sector on that day. The cameras have a limited memory and will only save pictures of the last tribe that visited their sector. A ranger can access a movement camera to check the pictures, which have a 60% chance of providing the rangers with the required information about the tribe's health. After checking a camera, the pictures are removed.

Each time step, there is a 12% chance that a tribe migrates to an edge-adjacent square. Each edge adjacent square has an equal chance of being migrated to. However, lions are territorial, so a tribe will never migrate to a sector the other tribe currently occupies or can move to.

Formulating and Checking the Model

Exercise 1 Model this problem as the appropriate Markov model (e.g., using the PRISM language). Use Figure 5 for the initial state. Using a probabilistic model checker (e.g., Storm), show that the two lion tribes will never be in the same sector.

Exercise 2 Let's assume each action takes the rangers an hour, and they have 14 hours available in the park in one day.

1. Compute the best effort probability that the health of the entire lion population is estimated in one day.
2. Compute which lion tribe is easiest to go after first (again with the time limit of one day).

Exercise 3 The rangers want to estimate the added value of each movement camera.

1. For each camera and lion tribe pair, the probability that the lion tribe will visit the camera during a day.
2. Which camera currently has the best placement? Which sector would be best to place the cameras according to the above metric, excluding the sectors where the lion tribes start?

Learning Probabilities

Consider the lion tribe starting top left corner of the wildlife park. We want to estimate the probability of that lion tribe migrating to the south from the top left corner if the other lion tribe is not nearby enough to influence the migration. Note that the rangers' position and action choice do not influence the tribe's migration behavior.

Exercise 4 Use the Stormpy simulator to generate data points for the above-described migration of the lion tribe.

1. Generate 50 data points. Given prior Dirichlet distribution $\alpha_{South} = 5, \alpha_{East} = 2$, and $\alpha_{Stay} = 43$, compute the MAP estimation of the probability for the south migration of a lion tribe from the top left corner without another lion tribe nearby.
2. Generate another 50 data points and use the posterior Dirichlet distribution of the previous step as the new prior Dirichlet distribution. What are the MAP estimates now?

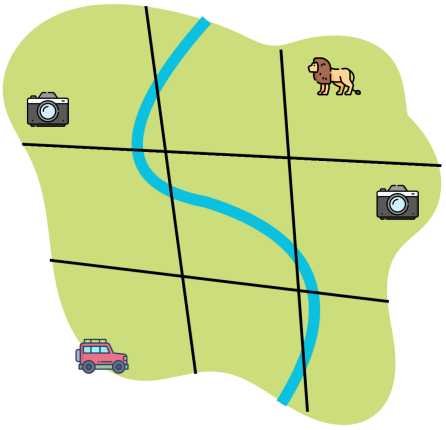

Figure 6: Partially observable wildlife tracking

Partial Observable Wildlife Tracking

Another smaller wildlife park has the same task of tracking the health of the lion population. This park consists of 9 sectors, with the total lion population consisting of a single tribe. The rangers again have their home base in the bottom left sector of the park (see safari wagon in Figure 6). The rangers have a safari wagon and two movement cameras to their disposal, which operate with the same probabilities as in the larger wildlife park of Exercise 1. However, this park does not have a drone, so the rangers cannot observe which sector the lions are in, unless they are in the same sector. The rangers only know their own position and the position of the cameras.

Exercise 5 Model this problem as the appropriate Markov model (e.g., using the PRISM language). Use Figure 6 for the initial state. Discuss the changes you need to make to the model of Exercise 1.

Exercise 6 Assuming the lion tribe starts in the top right corner, draw the belief MDP for two steps. This model should describe the belief for all combinations of two actions and two observations after the initial belief.