ETH zürich

P. Müller and D. Basin

Formal Methods and Functional Programming

Exercise Sheet 14: Modeling and LTL

Submission deadline: June 7, 2023, 23:59

Solutions should be submitted by email.

Installing and Running Spin

To run the Promela models, you will need to install the Spin model-checker as well as a C compiler. There are multiple ways to install Spin on your machine:

- **Windows:** You can download the archive from the following link and follow the readme: `https://polybox.ethz.ch/index.php/s/cQifMKXUW3G2iAI`
- **Ubuntu:** Run sudo `apt-get install spin` in a terminal to install the `spin` package.
- **Mac:** Use Homebrew (`https://brew.sh`) to install Spin by running `brew install spin`.
- **Executables:** Download pre-compiled executables from Spin's GitHub page: `https://github.com/nimble-code/Spin/tree/master/Bin`.
- **Compiling**: You can compile Spin from source from: `https://github.com/nimble-code/Spin`.

Short re-cap for running Spin:

- `spin filename.pml` will carry out a simulation of the model, yielding one random trace. This does *not* perform an exhaustive check the model.
- `spin -a filename.pml` will create a file `pan.c`, that must be compiled and run to exhaustively check a model. In case of failure, a corresponding trail file (`filename.pml.trail`) is typically generated, containing the information about the failing trace.
- `spin -t filename.pml` will replay the trace from the corresponding trail file.

Assignment 1 (Dining Philosophers)

The problem of the dining philosophers illustrates common issues for concurrent programs: We consider n philosophers sitting around a circular table. Each philosopher spends their life thinking and eating. In the center of the table is a large platter of spaghetti. Because the spaghetti are long and tangled a philosopher must use two forks to eat them. Since the philosophers can only afford n forks, a single fork is placed between each pair of philosophers, which they have to share. Each philosopher can only reach the forks to their immediate left and right with their left or right hand, respectively.

Note: For this exercise, you may *not* submit handwritten models. Only Promela files will be accepted. While model checking you may use the fairness flag `-f`, but, if you do, be sure to mention it in your solution. As explained in the exercise sessions, the fairness flag causes spin to only explore traces which schedule the individual processes in a *weakly-fair* manner; in particular, it cannot be the case that a process that is continually enabled (could make another step) is never scheduled. In the GUIs, this can be chosen by choosing "With Weak Fairness" in the Verification Options.

Task 1.1. You find a skeleton of a Promela model in `philosopher_skeleton.pml`. Complete the model such that each philosopher chooses non-deterministically to pick up their left or right fork first. Use Spin to find a deadlock and explain its source. How does the falsification time change when increasing the number n of philosophers?

Task 1.2. The deadlock can be avoided when each philosopher behaves more deterministically to pick up the forks. Note that not all philosophers have to behave the same. Model your solution in Promela and use Spin to check that it is indeed deadlock free.

Task 1.3. Check whether your solution is starvation free, i.e., whether there are no paths in which a philosopher never eats. You must modify the model accordingly, add an LTL formula, and model-check it with Spin.

Note: To model check a Promela file against an LTL formula f with Spin, you may express the LTL formula in an LTL block in the file (`ltl` $\{f\}$). The LTL formula is expressed using:

- `[] <> ! && || -> U` for operators $\Box$ $\Diamond$ $\neg$ $\wedge$ $\vee$ $\Rightarrow$ U resp.
- The equality operator as a primitive propositional formula. For example, to express `p`, one should write `(p==true)`.

Spin should be executed (if running from the command-line) with the analysis option -a. Furthermore, the final executable must also be executed with the -a option. For example:

```
spin -a file.pml
gcc -o ts.exe pan.c
./ts.exe -a
```

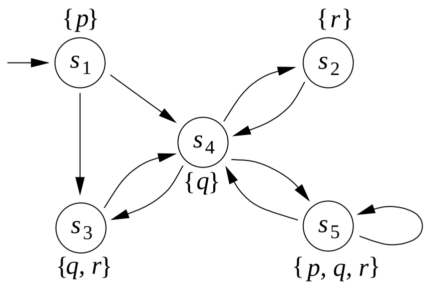
In the GUIs (e.g. xspin), you can achieve the same thing by choosing (in Verification Parameters) Liveness, acceptance cycles and “Apply Never Claim”/”use claim” (ignore the warnings in xspin - these are because the gui doesn’t understand the `ltl` block syntax (but the underlying version of spin does).

Task 1.4. Modify the solution to make it starvation free. Model-check the new model with your old LTL formula. You may introduce any restrictions you find appropriate to the behaviour of the philosophers.

Note: This exercise is a bit more involved.

Assignment 2 (Linear Temporal Logic)

Consider the transition system T over the set of atomic propositions $P = \{p, q, r\}$:



That is, T is the transition system $(\Gamma, s_1, \to, L)$ with

$$\Gamma = \{s_1, s_2, s_3, s_4, s_5\}$$

$$\to = \big\{(s_1, s_3), (s_1, s_4), (s_2, s_4), (s_3, s_4), (s_4, s_2), (s_4, s_3), (s_4, s_5), (s_5, s_4), (s_5, s_5)\big\}$$

$$L(s_i) = \begin{cases} \{p\} & \text{if } i = 1 \\ \{r\} & \text{if } i = 2 \\ \{q, r\} & \text{if } i = 3 \\ \{q\} & \text{if } i = 4 \\ \{p, q, r\} & \text{if } i = 5. \end{cases}$$

Task 2.1. Which of the following LTL formulas are satisfied in T, i.e., $T \models \varphi_i$?

$$\begin{aligned}
\varphi_1 &= \Diamond \Box r \\
\varphi_2 &= \Box \Diamond r \\
\varphi_3 &= \bigcirc \neg r \Rightarrow \bigcirc \bigcirc r \\
\varphi_4 &= \Box p \\
\varphi_5 &= p \, \mathsf{U} \, \Box (q \vee r) \\
\varphi_6 &= (\bigcirc \bigcirc q) \, \mathsf{U} \, (q \vee r) \\
\varphi_7 &= \Diamond \Box \bigcirc q \\
\varphi_8 &= (\Diamond \Box p) \Rightarrow (\Diamond \Box r)
\end{aligned}$$

Justify your answer. If $T \not\models \varphi_i$, provide a computation γ of T such that for the corresponding trace t, $t \not\models \varphi_i$.

Task 2.2. Formalize the following properties in LTL:

i. Eventually, it will not be possible for the system to go to state s_1.

ii. Whenever the system is in a state that satisfies r, then the next state satisfies q.

iii. p always implies r except perhaps in the initial state.

iv. Whenever the system is in state s_5, it will remain there until r becomes false.

v. q will be true at least twice.

vi. q will be true infinitely often.

vii. If p is true only at the initial state of a trace, then r is false infinitely many times in that trace.

viii. s_4 can never be repeated (there is no transition from s_4 to itself).

Task 2.3. A Promela model for the transition system above is given in `ts.pml`. Use Spin to model-check all the properties φ_i that *do not* contain the $\bigcirc$ operator. Inspect `ts.pml` and answer the following question: Why are `atomic` statements used?

Assignment 3 (Liveness and Safety Properties)

Task 3.1. Let A be a set of atomic propositions. Prove that a property is both a safety and liveness property if and only if it is equal to $\mathcal{P}(A)^\omega$.

Task 3.2. Is $\emptyset$ a safety property, a liveness property, both, or neither?

Task 3.3. Is $p\mathsf{U}q$ a safety property, a liveness property, both, or neither?