Exercises 5

Task 5.1. Assert It

Provide assert statements that can be used to ensure:

1. (Array sorting, ex05_sortMerge.pml) At the end of the program, both halves of the array are sorted.
2. (Rice eating, ex05_riceEatingDeadlock.pml) At the end of each process, two chopstick are free.

Task 5.2. Base Ball Games

Assertions can be (ab)used to solve “cryptarithm” puzzles. In cryptarithms, each letter represents a certain digit, and the task of the player is to reveal the code. For example, cryptarithm `MOSES + MEETS = SALOME` encodes the expression `93121 + 92271 = 185392` (`M` is 9, `O` is 3, `S` is 1, and so on). Normally, it is presumed that different letters encode different digits.

Your task is to solve the cryptarithm `BASE + BALL = GAMES` using SPIN. This can be done as follows:

1. Prepare array `A` containing digits 0 to 9 in a random order. Set global index variable `i` to 0, and run 10 independent processes. Each process assigns its `_pid` to `A[i]`, and increases the index. As a result, you’ll obtain `A`.
2. Use the first 7 elements of `A` to represent digits of the puzzle.
3. Assert that `BASE + BALL` is not equal to `GAMES`.
4. SPIN will find the case where your assertion fails, and thus `BASE + BALL` IS equal to `GAMES`!
5. Make sure to place one empty `printf()` call before your `assert` statement, otherwise you might not see all the variables in the trace output.

Task 5.3. Gentle Rice Eaters

The rice eating modeling code (shown in the **Detecting Livelocks** section of the lecture 5) assumes that both chopsticks are initially free. Modify it to represent the case where one of the eaters is already eating rice when the program starts.

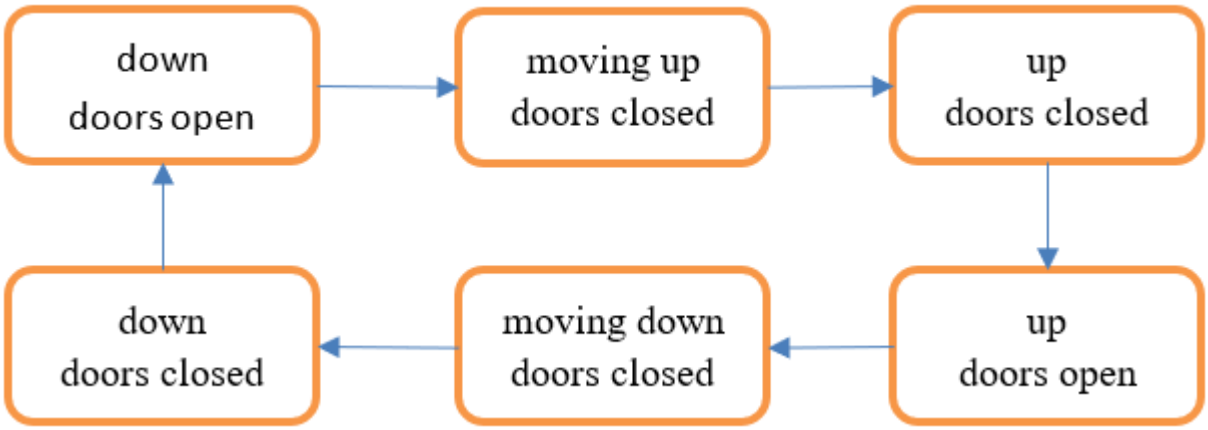
Provde LTL formulas to prove that:

1. At least one eater eventually starts eating.
2. Nobody is eating indefinitely often (in other words, there is no guarantee that the next meal will happen).

3. It is possible to reach a situation when nobody is eating.

Task 5.4. Elevator Logic v2

The code in ex05_elevator.pml models elevator behavior in a two-story building according to the following diagram:



–

Create LTL formulas to ensure that:

1. Elevator returns to the state “down with doors open” indefinitely often.
2. When the doors are open, the elevator is either in the “up” or “down” state.
3. Once the doors are closed, eventually the elevator will start moving.