Elevator in NuSMV

Curri Mattia

Temporal Logic

Temporal logic is a system for reasoning about propositions that change over time.

Temporal operators:

- G (Globally): A condition holds always.
- F (Finally): A condition will hold at some point in the future.
- X (Next): A condition holds at the next moment.
- U (Until): One condition holds until another becomes true.
- R (Release): One condition must hold as long as another does

Linear Temporal Logic (LTL) focuses on single timelines, while Computation Tree Logic (CTL) considers branching paths, allowing for multiple potential futures.

Model Checker

Given a desired property expressed as a temporal logic formula and a Kripke model, the goal is to determine whether the property holds in the model from a specific starting state. If the property is satisfied, the model checker outputs "yes." Otherwise, it provides a counterexample demonstrating where the property is violated.

To verify satisfaction, the model checker explores a portion of the model over a number of interactions, starting from zero. It attempts to find counterexamples within that segment of the Kripke structure. If no counterexample is found, it increases the number of interactions and continues the process. If a counterexample is found, execution stops, and the counterexample is displayed.

NuSMV

NuSMV is short for New Symbolic Model Verifier and it's a symbolic model checker that supports the analysis of specifications expressed in CTL and LTL. The aim of NuSMV is to have a robust, customizable and open-source model checker.

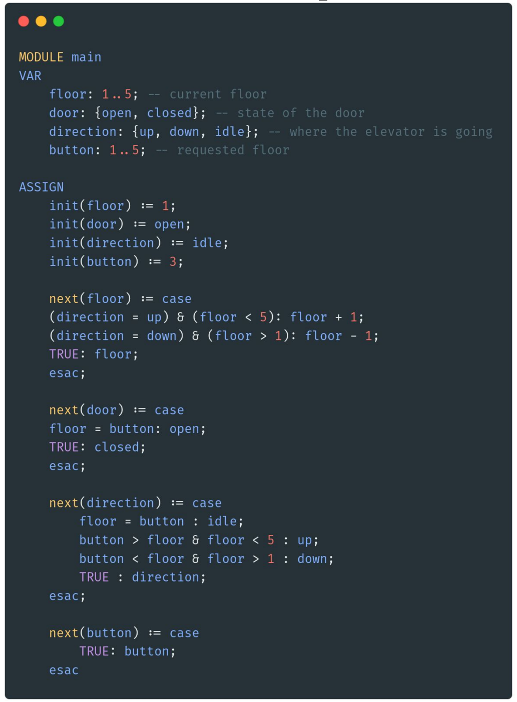
Exercise: Elevator in NuSMV

Model an elevator through LTL and verify certain properties using NuSMV.
The lift is located in a 4-storey building and it’s described by:
floor -> current floor
requested -> floor to be reached
door -> door status
direction -> direction of movement

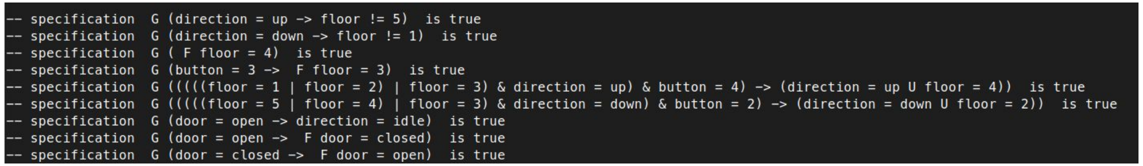
Exercise: Specifications

1. The elevator should note move if the door is open
2. Whenever the door is open, it will eventually be closed
3. Whenever the door is closed, it will eventually be open
4. The elevator con move upward only if the floor is not the highest
5. The elevator con move downward only if the floor is not the lowest
6. The elevator can visit any floor infinitely often
7. When a floor is requested, the elevator will eventually stop at the floor
8. When the elevator is traveling upward, id does not change its direction when there are passengers waiting to go to a higher floor
9. When the elevator is traveling downward, id does not change its direction when there are passengers waiting to go to a lower floor

Exercise: Specifications



```
MODULE main
VAR
    floor: 1..5; -- current floor
    door: {open, closed}; -- state of the door
    direction: {up, down, idle}; -- where the elevator is going
    button: 1..5; -- requested floor

ASSIGN
    init(floor) := 1;
    init(door) := open;
    init(direction) := idle;
    init(button) := 3;

    next(floor) := case
    (direction = up) & (floor < 5): floor + 1;
    (direction = down) & (floor > 1): floor - 1;
    TRUE: floor;
    esac;

    next(door) := case
    floor = button: open;
    TRUE: closed;
    esac;

    next(direction) := case
        floor = button : idle;
        button > floor & floor < 5 : up;
        button < floor & floor > 1 : down;
        TRUE : direction;
    esac;

    next(button) := case
        TRUE: button;
    esac
```



```
-- specification  G (direction = up -> floor != 5)  is true
-- specification  G (direction = down -> floor != 1)  is true
-- specification  G ( F floor = 4)  is true
-- specification  G (button = 3 ->  F floor = 3)  is true
-- specification  G (((((floor = 1 | floor = 2) | floor = 3) & direction = up) & button = 4) -> (direction = up U floor = 4))  is true
-- specification  G (((((floor = 5 | floor = 4) | floor = 3) & direction = down) & button = 2) -> (direction = down U floor = 2))  is true
-- specification  G (door = open -> direction = idle)  is true
-- specification  G (door = open ->  F door = closed)  is true
-- specification  G (door = closed ->  F door = open)  is true
```