An Elevator Controller

Alessandro Artale
Faculty of Computer Science – Free Univeristy of Bolzano
artale@inf.unibz.it

1 Introduction

The NuSMV program (at the end of this document) describes the skeleton of an elevator system for a 4-floors building. The skeleton includes modules both for the physical system (reservation buttons, cabin, door) and for the controller. It also shows the connection between modules. The objective of the exercise is to complete the program, and check that the requirements described below are satisfied. In particular, you will have to:

1. Formalize the transition relation for the variables in the various modules;
2. Formalize with CTL/LTL the requirements for each module (specified in the following Sections), and make sure that the design satisfies them.

Note: Do not use fairness constraints.

1.1 Button Module

For each floor there is a button to request service, that can be pressed. A pressed button stays pressed unless reset by the controller. A button that is not pressed can become pressed nondeterministically.

REQ: No button can reach a state where it remains pressed forever.

1.2 Cabin Module

The cabin can be at any floor between 1 and 4. It is equipped with an engine that has a direction of motion, that can be either standing, up or down. The engine can receive one of the following commands: "nop", in which case it does not change status; "stop", in which case it becomes standing; "up" ("down"), in which case it goes up (down).

REQ: The cabin can move up only if the floor is not 4.

REQ: The cabin can move down only if the floor is not 1.

1.3 Door Module

The cabin is also equipped with a door that can be either open or closed. The door can receive either "open", "close" or "nop" commands from the controller, and it responds opening, closing, or preserving the current state.

1.4 Controller Module

The controller takes in input (as sensory signals) the floor and the direction of motion of the cabin, the status of the door, and the status of the four buttons. It decides the controls to the engine, to the door and to the buttons. The controller must also satisfy the following requirements.

REQ: The controller must not reset a button that is not pressed.

REQ: A button must be reset as soon as the cabin stops at the corresponding floor with the door open.

REQ: No pressed button can be reset until the cabin stops at the corresponding floor and opens the door.

REQ: The cabin can move only when the door is closed and the direction is standing.

REQ: If no button is pressed, the controller must issue no commands and the cabin must be standing.

REQ: The controller can issue a stop command only if the direction is up or down.

REQ: The controller can issue an open command to the door only if the door is closed.

REQ: The controller can issue a close command to the door only if the door is open.

The NuSMV Skeleton

```
-----------------------------------------------------------------------------
-- AN ELEVATOR CONTROLLER                                                  --
-----------------------------------------------------------------------------
-- This SMV program describes an elevator system for a 4-floors building.
-- It includes modules both for the physical system (reservation buttons,
-- cabin, door), and for the controller.

-----------------------------------------------------------------------------
-- BUTTON                                                                  --
-----------------------------------------------------------------------------
MODULE Button(reset)
  VAR
    pressed : boolean;
  ASSIGN
    init(pressed) := 0;

-----------------------------------------------------------------------------
-- CABIN                                                                   --
-----------------------------------------------------------------------------
MODULE Cabin(move_cmd)
  VAR
    floor     : { 1,2,3,4 };
    direction : { standing, moving_up, moving_down };
  ASSIGN
    init(direction) := standing;

-----------------------------------------------------------------------------
-- DOOR                                                                    --
-----------------------------------------------------------------------------
MODULE Door(door_cmd)
  VAR
    status : { open, closed };

-----------------------------------------------------------------------------
-- CONTROLLER                                                              --
-----------------------------------------------------------------------------
MODULE CTRL(floor, dir, door, pressed_1, pressed_2, pressed_3, pressed_4)
  VAR
```

```
  move_cmd : {move_up, move_down, stop, nop};
  door_cmd : {open, close, nop};
  reset_1 : boolean;
  reset_2 : boolean;
  reset_3 : boolean;
  reset_4 : boolean;
ASSIGN
-- Button N is reset only if it is pressed, we are at floor N, and
-- the door is open.
  reset_1 := (pressed_1 & floor = 1 & door = open);
  reset_2 := (pressed_2 & floor = 2 & door = open);
  reset_3 := (pressed_3 & floor = 3 & door = open);
  reset_4 := (pressed_4 & floor = 4 & door = open);
DEFINE
-- Check whether there are pending requests at the current floor,
-- at a higher floor, and at a lower floor.
  pending_here := (floor = 1 & pressed_1) | (floor = 2 & pressed_2) |
                  (floor = 3 & pressed_3) | (floor = 4 & pressed_4) ;
  pending_up   := (floor = 1 & ( pressed_2 | pressed_3 | pressed_4 )) |
                  (floor = 2 & (             pressed_3 | pressed_4 )) |
                  (floor = 3 & (                         pressed_4 )) ;
  pending_down := (floor = 4 & ( pressed_1 | pressed_2 | pressed_3 )) |
                  (floor = 3 & ( pressed_1 | pressed_2             )) |
                  (floor = 2 & ( pressed_1                        )) ;
ASSIGN
-- * RULES FOR THE DOOR COMMAND
-- * If the cabin is moving, do not send commands to the door.
-- * If there is a pending request at the current floor and
--   the door is closed, open it.
-- * If there are pending requests at different floors and the
--   door is open, close it.
-- * Otherwise, do not send commands to the door.
  door_cmd :=
    case
      ???
    esac;
VAR
-- Variable "last_dir" records the last movement direction of the cabin.
  last_dir : {moving_up,moving_down};
```

```
  ASSIGN
    next(last_dir) := case
        dir = standing : last_dir;
        1              : dir;
      esac;
  -- * RULES FOR THE MOVE COMMAND
  -- * If the door is open, do not send move commands to the cabin.
  -- * If there is a pending request at the current floor
  --   and the cabin is moving, stop it.
  -- * If there are pending requests both at higher and at lower floors,
  --   keep moving in "last_dir".
  -- * If there are pending requests at higher (lower) floors,
  --   move up (down).
  -- * Otherwise, do not send commands to the cabin.
    move_cmd := case
         ???
      esac;

-----------------------------------------------------------------------------
-- MAIN                                                                    --
-----------------------------------------------------------------------------

MODULE main
  VAR
    cabin : Cabin(ctrl.move_cmd);
    door  : Door(ctrl.door_cmd);
    button_1 : Button(ctrl.reset_1);
    button_2 : Button(ctrl.reset_2);
    button_3 : Button(ctrl.reset_3);
    button_4 : Button(ctrl.reset_4);
    ctrl : CTRL(cabin.floor, cabin.direction, door.status,
                button_1.pressed, button_2.pressed,
                button_3.pressed, button_4.pressed);
```