nuXmv: Exercises - Part A*

Patrick Trentin
patrick.trentin@unitn.it
http://disi.unitn.it/~trentin

Formal Methods Lab Class, May 13, 2016

Università degli Studi di Trento

*These slides are derived from those by Stefano Tonetta, Alberto Griggio, Silvia Tomasi, Thi Thieu Hoa Le, Alessandra Giordani, Patrick Trentin for FM lab 2005/15

Contents

1. Exercises
   - Elevator
   - Dining Philosophers

Exercise: Elevator [1/5]

Exercise:

- Given the `model` of an elevator system for a **4-floors** building, including the complete description of:
  - reservation buttons
  - cabin
  - door
  - controller
- Enrich the model with **properties** encoding the **requirements** that must be met by each component of the system, and **verify** that such requirements are satisfied.

Exercise: Elevator - Button [2/5]

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.

Requirements:

- The controller must not reset a button that is not pressed.

Exercise: Elevator - Cabin [3/5]

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).

Requirements:

- The cabin can receive a stop command only if the direction is up or down.
- The cabin can receive a move command only if the direction is standing.
- The cabin can move up only if the floor is not 4.
- The cabin can move down only if the floor is not 1.

Exercise: Elevator - Door [4/5]

The cabin is also equipped with a **door** (kept in a separate module in the SMV program), 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.

Requirements:

- The door can receive an open command only if the door is closed.
- The door can receive a close command only if the door is open.

Exercise: Elevator - Controller [5/5]

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.

Requirements:

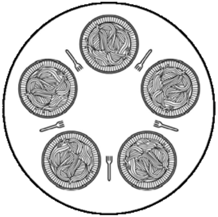
- no button can reach a state where it remains pressed forever.
- no pressed button can be reset until the cabin stops at the corresponding floor and opens the door.
- a button must be reset as soon as the cabin stops at the corresponding floor with the door open.
- the cabin can move only when the door is closed.
- if no button is pressed, the controller must issue no commands and the cabin must be standing.

Contents

1. Exercises
   - Elevator
   - Dining Philosophers

Exercise: Dining Philosophers [1/2]

Five philosophers sit around a circular table and spend their life alternatively thinking and eating. Each philosopher has a large plate of noodles and a fork on either side of the plate. The right fork of each philosopher is the left fork of his neighbor. Noodles are so slippery that **a philosopher needs two forks to eat it**. When a philosopher gets hungry, he tries to **pick up his left and right fork, one at a time**. If successful in acquiring two forks, he **eats for a while** (preventing both of his neighbors from eating), then **puts down the forks, and continues to think**.



Exercise: Dining Philosophers [2/2]

Exercise:

1. Implement in SMV a system that encodes the philosophers problem. Assume that when a philosopher gets hungry, he tries to pick up his left fork first and then the right one.
   Hint: you might consider an altruist philosopher, which can resign his fork in a deadlock situation.

2. Verify the correctness of the system, by specifiying and checking the following properties:
   - Never two neighboring philosophers eat at the same time.
   - No more than two philosophers can eat at the same time.
   - Somebody eats infinitely often.
   - If every philosopher holds his left fork, sooner or later somebody will get the opportunity to eat.

Exercises Solutions

- will be uploaded on course website within a couple of days
- send me an email if you need help or you just want to propose your own solution for a review

- learning programming languages requires practice: try to come up with your own solutions first!