FIT3013 Assignment 2—A Lift Control System

Due Date: **23:59, Sunday 23 Oct, 2016**

1 Assignment Objectives

In assignment 1 you developed an Event-B specification for a multi-lift system, and worked with various proof obligations generated from the specification. In this assignment you will model the same problem (albeit a bit simplified) as a transition system and verify some properties, expressed in temporal logics, using the NuSMV model checker.[1]

Your tasks are to:

1. Develop a transition system-based model for the problem.
2. Express the above model as a NuSMV model.
3. Express, in LTL or CTL formulas, desirable properties of the model.
4. Verify the LTL/CTL formulas using NuSMV.

Objectives addressed by this assignment include:

- An appreciation of importance of consistent specifications,
- A good understanding of the fundamentals of modelling software systems using program graphs and transition systems,
- A good understanding of the fundamentals of the various language constructs of the NuSMV specification language,
- The ability to write simple NuSMV models,

[1]`http://nusmv.fbk.eu/NuSMV`

- The ability to write simple LTL/CTL specifications, and
- The ability to verify the correctness of LTL/CTL specifications and generate a counterexample when a specification doesn't hold.

2 A Multi-lift System

A multi-lift system consists of multiple lifts and a set of buttons on each floor and in each lift, among other things. A user can make requests in lifts and from floors, by pushing some buttons. These buttons include those for directions, floors, and possibly special requests (emergency stop, etc.), which we will ignore in this assignment.[2]

2.1 Floors

The system consists of 4 floors that the lifts stop by.

2.2 Buttons

The system consists of a number of buttons, inside the lifts and on the floors. On each floor there is only one set of up/down buttons. There is no down button on level 0 and up button on level 3.

Each button is `active` (pressed) or not, which is controlled by the parameter `reset` (supplied by the controller and/or a lift). When the button is `active` and the `reset` signal is set, the button is then set to inactive (`active = FALSE`). All buttons are initially inactive. Interestingly, an inactive button can nondeterministically become active or stay inactive, modelling the random event of a request (or the lack of).

2.3 Lifts

A lift can be `moving` or not, and it maintains a position (0, 1, 2 or 3). The movement and position of a lift is controlled by 2 variables: `go_up` and `go_dn`, which are set by the main controller module. The state transitions of a lift can be depicted as follows in Figure 1 (taken from `http://www.inf.ed.ac.uk/`

[2]The specification is inspired by `http://www.inf.ed.ac.uk/teaching/courses/ar/coursework/cw2.pdf`

`teaching/courses/ar/coursework/cw2.pdf`). positioned at, and whether the lift is moving or not. This module has two inputs, and

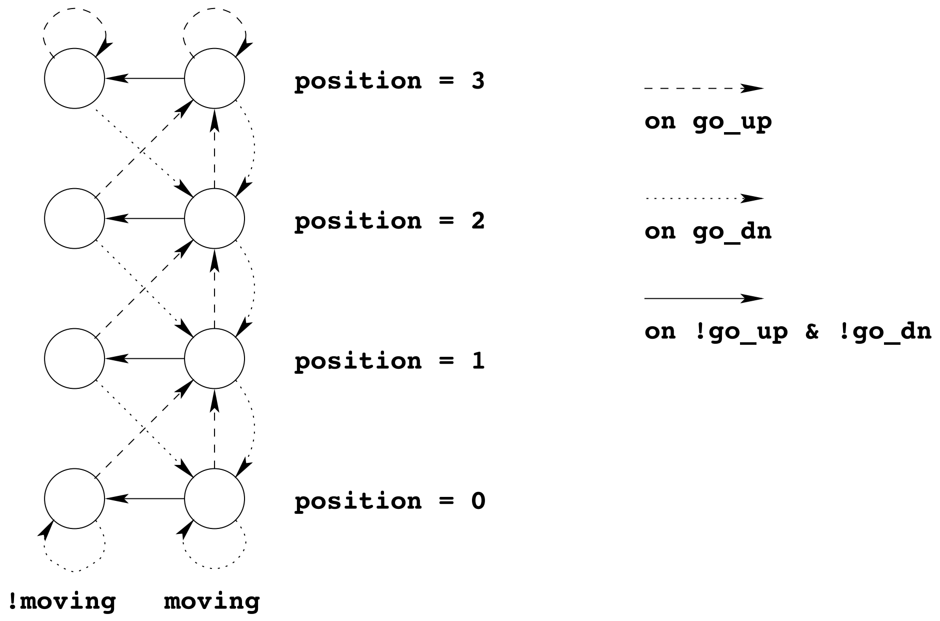

Figure 1: The state transitions of a lift.

2.4 The Controller

The controller initialises the lifts and buttons and controls whether and when a lift moves up or down. The controller has 3 modes: `going_u`, `going_d` and `idle`, and it changes its mode by observing the requests in the system and the positions of the lifts. The mode of the controller is used to, together with information about a lift, control the movement direction of the lift through the expressions `go_up` and `go_dn`. For example, when `mode = goingup`+, if there is a request above a lift's current position, it will continue to direct the lift to go up; otherwise, if there is no request above, the controller will set its mode to idle.

2.5 System Requirements

There are some basic requirements that the system must satisfy.

0. Multiple requests can be made from within a single lift.

   Hint: this requirement may need fairness conditions to be specified. Note that fairness conditions are not fully supported yet and may slow down the verification process. You can turn them off (commenting them out) when you're working on requirements that don't need them.

1. A request made from inside a lift for any other floor will eventually take the lift to that floor, and

2. A request made from a floor will eventually take a lift to that floor

   The above two requirements state that no request can be ignored, i.e., all requests have to be serviced eventually (the lift stops, not moving, at a floor).

3. (When there are no other requests from inside the lifts), only one lift is used to service a given floor request, but not both.

4. A lift will continue to travel in the same direction if it has requests in that direction.

5. When there are no pending requests in the direction that it has been travelling, a lift can change direction to service another request in the opposite direction.

6. The lift stops when there are no pending requests.

7. The controller never directs a lift to go up and down at the same time.

8. Whenever a lift is stopped at floor 0 and the floor buttons in the lift for floors 1 and 2 are active (pressed), the lift will stop at floor 1 before stooping at floor 2.

9. Whenever a lift is stopped at floor 0 and the down buttons on floors 1 and 2 are active (pressed), the lift will stop at floor 2 before stopping at floor 1.

10. If none of the up and down buttons on floor 2, and the floor 2 button in a lift is ever active, the lift will never stop at floor 2.

    This property will turn out to be false for the original specification provided. It is caused by a bug in the controller. Find and fix the bug, and document it in the report. Note: you may need to comment out the fairness conditions to find a counterexample for this requirement.

3 Questions to Ponder

We listed a number of questions in Assignment 1 for you to think about. Please consider the following questions and answer them in your report.

1. Is your scheduling algorithm synchronous or asynchronous?
2. In what order are requests received and recorded?
3. In what order are requests serviced?
4. How is it decided which lift is chosen to service a request?
5. While a lift is travelling up (or down) to service a request, what happens if a new request along the way and in the same direction is received by the system? For example, a lift is at level 1 and travelling to level 5. If a new up request on level 3 is received by the system, would this lift stop?
6. Is it guaranteed that a request will eventually be serviced?
7. How does a user know that she will eventually get to the desired floor?
8. What happens if one lift breaks down, in terms of the requests registered with that lift?
9. When a lift breaks down, will new requests still be scheduled for this lift? What if the lift is permanently decommissioned?
10. What if we allow requests to be cancelled? What part of the specification needs to be changed to accommodate it?
11. Do the lifts collaborate/compete with each other to service requests?
12. How do we make sure that a request is not to be serviced by 2 lifts?
13. Is your system *fair* for users? Is your system *fair* for lifts?

4 Tasks

In this assignment, you will, as a team, specify the multi-lift system as a transition system, and to *verify* its various properties using model checking techniques, through LTL/CTL specifications.

4.1 Informal Modelling as a Labelled Transition System

Recall that a labelled transition system is a 6-tuple $TS = (S, Act, \longrightarrow, I, AP, L)$. Model the multi-lift system as a labelled transition system, by specifying each component (S, *Act*, etc.) of the transition system.

If some component is too long or too tedious (e.g., $\longrightarrow$ or L) to fully describe, you can provide a few examples instead of a full model.

4.2 Modelling and Verification in NuSMV

A NuSMV specification, `lift.smv`, for the single-lift system is provided to help you get started. It contains basic definitions for buttons, a lift and the controller. **Do not** alter the specification in `lift.smv`.

Tasks of this component can be divided into the following parts.

Single-lift

1. Express as LTL, CTL or invariant formulas the desirable properties given in Section 2.5 above.

2. Run NuSMV on the specification to verify these properties.

3. When a property is found not to hold, use NuSMV to generate a counterexample.

4. Requirement 0 above doesn't hold, and it may be fixed by adding a fairness condition. Add the fairness condition to the specification and document your reasoning.

5. Requirement 10 above doesn't hold. Analyse the counterexample, and find and correct the bug in the original specification that causes it.

Include additional formulas if necessary.

Do not alter the specification in `lift.smv`. To fix the bugs, make a copy of `lift.smv`, name it `lift-fixed.smv` and write your corrected specification in this file. Verify that all properties hold (with fairness conditions turned on/off).

Multi-lift

1. Based on your transition system developed in Section 4.1, develop a NuSMV model for the system, in a file called `lift-multi.smv` by extending the single-lift model to **2 lifts**.
2. Express as LTL, CTL or invariant formulas the desirable properties given in Section 2.5 above.
3. Run NuSMV on the specification to verify these properties, and observe and discuss verification results.

Limit NuSMV to no more than 5 minutes for each run (LTL/CTL/invariant specification).

4.3 Report

Write a *maximum* 8-page report that contains the following contents.

- Include and explain your informal transition system formulation of the problem.
- Explain your NuSMV model: the roles of different variables, their initialisation, and the meaning of states and state transitions.
- Explain the meaning of your LTL specifications.
- Explain your fix for the bugs about requirements 0 and 10 for the single-lift part.
- Document your multi-lift extension of the specification.
- Answer the questions listed in Section 3 for both single-lift and multi-lift models.

5 Optional: Extended Functionality (Max 5 Extra Marks)

It turns out that some of the properties don't hold in the multi-lift part. Extend the specification to correct some of these problems. Document how the specification is modified/extended in your report.

Moreover, you may want to specify additional properties that the (either single-lift or multi-lift) system should hold. Document them in the report and show whether they hold or not. If not, explain why they don't.

Make a copy of `lift-multi.smv` and call it `lift-multi-fixed.smv`. Bug fixes and new property specifications should be made in this new file.

You may need to consider the incorporation of *fairness* in fixing the model.

Limit NuSMV to no more than 5 minutes for each run (LTL/CTL/invariant specification).

One *additional* page in the report is allowed for the bonus part.

6 Accessing NuSMV

NuSMV has been installed on a (virtual) server. You can log on to the server (`fit-app13-v02.infotech.monash.edu`), where NuSMV is installed, with your Authcate username/passwd. For example, if a student's Authcate ID is "`john`", he can login to the course server with the following command:

```
$ ssh john@fit-app13-v02.infotech.monash.edu
```

Note that if you would like to access the server off-campus, you will need to use the Monash VPN software. Instructions on how to install it can be found online.[3]

You can copy files to the server using the `scp` command. For example, if John wants to upload the file `puzzle.smv` onto the server, under his home directory, he can use the following (unfortunately a bit tedious) commands (assuming that file `puzzle.smv` is in the *current* directory on his desktop/laptop):

```
$ scp puzzle.smv john@fit-app13-v02.infotech.monash.edu:~/
```

[3]`http://www.its.monash.edu/staff/networks/vpn/install.html`

NuSMV is open source and can be installed on major operating systems. See NuSMV's web site[4] for instructions for download and installation.

The model and the LTL specifications can be verified by executing the following NuSMV command on the command line:

```
$ NuSMV puzzle.smv
```

When a specification violation is detected, NuSMV can find a short counterexample using its *bounded model checking* capability, with the following command:

```
$ NuSMV -bmc -bmc_length 10 -n 0 puzzle.smv
```

where `-bmc_length` tells NuSMV the maximum problem bound in search for a counterexample. The flag `-n` indicates the index (starting from 0) of the LTL specification for which the counterexample is to be generated. You may need to increase the value for parameter `-bmc_length`.

7 Marking Scheme

The assignment is worth 15 marks (plus maximum 5 bonus marks) and will be assessed for completeness and correctness.

4 marks: informal LTS. The transition system should define a appropriate and sufficient state space (S), correct initial states (I). It also should include appropriate atomic propositions (AP), the transition relation ($\longrightarrow$), as well as the labelling function (L).

5 marks: NuSMV model. The development of the NuSMV model for the systems, including variable definition, their initialisation and state transition. Note that the NuSMV model needs to be based on your transition system formulation. Specifically,

- Model should be syntactically correct
- Model should be reasonably understandable
- Model uses `DEFINE` & modules to reduce repetition

[4] `http://nusmv.fbk.eu/`

- Model doesn't contain redundant definitions
- Variable initialisation is correct
- Model correctly identifies boundary cases (lift changing direction, etc.)
- Model correctly updates variables (lift position, status, etc.)
- Model correctly syncs up variables (button turns off when a lift arrives, etc.)

3 marks: LTL/CTL/invariant specification. You should choose the appropriate type (LTL, CTL, or invariant) for a given property. The specification should be syntactically and logically correct.

3 marks: Report. Besides the informal transition system, the accompanying report should succinctly and correctly describe the NuSMV models, LTL/CTL/invariant specifications in a complementary way. The report should document modelling assumptions, decisions and rationales of these decisions. The report should also include a response to the questions listed in Section 3. Finally, the report should not exceed the page limit.

---

5 marks (bonus): Correction. The fix of additional bugs in the system and/or the specification of additional LTL properties.

Table 1: Sample summary of self assessment.

Member	**Percentage of work**	**Main contributions**
Marge Simpson	40%	Leadership, overseeing all work, research & writing
Homer Simpson	20%	A bit of research
Bart Simpson	20%	A bit of writing
Lisa Simpson	20%	Research & writing

7.1 Self Assessment

Even though the project assignments are group-based, assessment is individual-based. Hence, in your report, please include a brief statement of each member's contribution. It can be as simple as Table 1. Of course you are welcome to include more detailed information.

In addition, each member of a team will also need to complete a CATME peer evaluation survey, which can be found at `https://www.catme.org/`.

8 Submission

Submit your solution as a single compressed directory named after your group ID (i.e., `<fit3013_groupX>.zip`), where `X` is the group number. The URL of the submission page is: `http://moodle.vle.monash.edu/mod/assign/view.php?id=3331258`.

You **must** use the Moodle submission mechanism for this. Multiple submissions are allowed, but only the last one received will be marked.

The compressed directory should include the following parts:

- The pdf version of your report. LaTeX is encouraged for this purpose (because of its superiority of mathematical typesetting over other tools such as MS Word).
- The NuSMV specification files, including `lift.smv`, `lift-fixed.smv`, `lift-multi.smv`, and `lift-multi-fixed.smv` if you attempt the optional part.

Note that each group also needs to make a *separate submission* of the report for TurnItIn plagiarism checking, at `http://moodle.vle.monash.edu/mod/turnitintooltwo/view.php?id=3473009`.

For both submissions, only one member of each group will need to submit the report on Moodle. Multiple submissions are allowed, but only the last one will be marked.

Late submissions are allowed. However, a **daily penalty** of 10% will be applied. For example, if a submission is made on Tuesday 25 Oct 2016, $(1 - 2 * 10\%) = 80\%$ of the total, original marks will be the final marks.