The University of Melbourne
Department of Computing and Information Systems
SWEN90004 Modelling Complex Software Systems

Assignment 2b, 2016

Released: 11 May. Deadline: 30 May at 23:00

Objective

To use a higher-level modelling language to specify and reason about a concurrent system.

Background and context

There are two parts to Assignment 2. The first part, 2a, was worth 10% of your final mark; this final part, 2b, is worth 15%. In the first part of the assignment you designed and implemented a simulator for an automatic car park system where cars travel in a parking space and go through a lift in the process. Now the task is to model the system using FSP, use LTSA to check it, and to correct any problems discovered using this method.

The tasks

1. **Model:** The first task is to model your implementation in FSP. That is, reverse engineer a FSP model from your Java implementation. Your model should contain comments that explain the design and its components. As examples of what is expected for comments, see the published solutions to Workshops 9 and 10. Don't forget to include your name.

2. **Check:** Specify what you believe are the relevant safety and liveness properties for your FSP model. Use LTSA to check these properties.

3. **Update**: Modify both your model and your implementation to correct any problems found in Task 2. *Be sure to create a copy of the file containing your original FSP model. You are required to submit both.*

4. **Discussion:** What (if any) problems did you find in your original model as a result of using LTSA? Were these problems also in the implementation? If so, why do you think you picked these up now and not before submitting Part 2a?

   If you did not find problems, were you convinced when you submitted Part 2a that no problems existed? Why did you believe this? Do you still believe there are no problems?

   Keep your discussion to no more than 400 words.

Procedure and assessment

The assignment should be completed by students individually. The submission deadline is Monday 30 May at 23:00. A late submission will attract a penalty of 1 mark for every calendar day it is late. If you have a reason that you require an extension, email Harald *well before the due date* to discuss this.

To tackle the assignment, first work through (and *understand*) the examples from lectures, and do the workshop exercises. FSP is not difficult—it is simpler than most programming languages, and much simpler than languages like Java. However, as with other languages, the way to master it is to use it, and to learn by doing. Trying to do the assignment straight up means you may struggle. Work through some easier examples first.

Submit a single zip file via the LMS. The file should include

- A file called `lift.lts` with your initial FSP model, including the safety and liveness properties from Task 2.
- Your updated/corrected FSP model `lift_fixed.lts` (not required if there are no changes), including the safety and liveness properties from Task 2.
- A file called `discussion.txt` or similar, containing the discussion of issues.
- If relevant, a directory containing an updated (corrected) Java implementation.

We encourage the use of the LMS's discussion board for discussions about the project. However, all submitted work is to be your own individual work.

This project counts for 15 of the 50 marks allocated to project work in this subject. Marks will be awarded according to the following guidelines:

Criterion	**Description**	**Marks**
Clarity, abstraction	Descriptions of all actions and processes are clear and concise. FSP models are at a suitable level of abstraction. Only the behaviour relevant to interaction is specified, and there is sufficient detail to implement the concurrency objectives from the model.	4 marks
Correctness	Both models accurately reflect the original implementations. The "good" model behaves correctly, is free from deadlock, does not violate any safety properties, and demonstrates liveness properties.	4 marks
Completeness	The model is complete. All components have been modelled and all expected behaviour is present. All expected safety and liveness properties have been described.	3 marks
Formatting	The FSP source adheres to the code format rules from Part 2a where this makes sense, including the use of comments.	2 marks
Discussion	The discussion shows understanding of the subject material.	2 marks
Total		15 marks

Why backwards?

A valid question is: why are we modelling the system *after* implementing it? Should it not be done the other way? Well, yes and no. Many people use modelling to understand an existing code base (just look at the number of tools for reverse engineering UML models from code bases). Reverse engineering is a great way to understand problems with an existing system; for example, why a deadlock is occurring. It is true, however, that in most cases, it would be cheaper and easier to do the modelling first.

The other reason why the assignment is "backwards" is that trying to model a system using a new type of notation, such as FSP, will often end in disaster. We hope that, having gone through the Java programming stage, you feel familiar with the system to be modelled and thus can concentrate on the use of FSP. The exercise should be one of applying abstraction—a skill that is of utmost importance in any engineering discipline.

Harald Søndergaard and Lida Rashidi
11 May 2016