Concurrency Coursework 2015-16

Deadline: Sunday, April 3 2016 (week 10), 17:00

This coursework is to be done in pairs, **apart from Part A, for which you should provide your individual solution** *(which is expected to be **substantially** different from your partner's)*.

Instructor: c.kloukinas-at-city.ac.uk City University London, U.K.

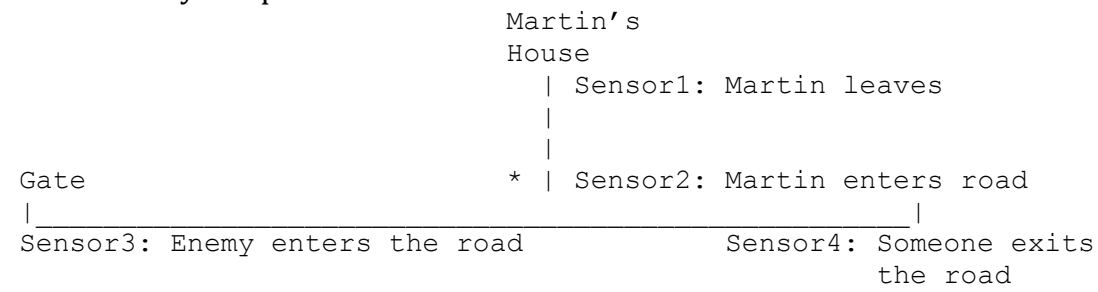
Whack-a-Mole? Enough is Enough!

Martin Mole has had enough with all the constant whacking and has decided to take action. So he's gone and installed four sensors outside his house to warn him of enemy cars coming and going. There are:

- One sensor that signals if an enemy has entered the road in from of his house,
- One sensor that signals if someone (enemy or Martin) has left the road,
- One sensor that signals if Martin has left his house, and
- One sensor that signals if Martin has entered the road.

At the same time Martin installed a gate at the entrance to his road and a warning indicator, which when on tells him that there are enemies on his road and when off tells him that there is no enemy currently on the road.

Let me draw you a picture:



The FSP processes for the sensors and gate are as below:

```
ENTERING_SENSOR = ( enter -> ENTERING_SENSOR ).

EXITING_SENSOR = ( exit -> EXITING_SENSOR ).

GATE = UP,
UP = ( lower -> DOWN
     | pass -> UP ),
DOWN = ( raise -> UP ).
```

When Martin exits the road he magically reappears inside his house. Enemies exiting the road also magically reappear in front of the gate.

Part A – Unsafe Model – INDIVIDUAL! (30%)

(a) Construct an FSP model of the unsafe model UNSAFE_MARTIN (i.e., where Martin can find himself on the road while enemies are also on it).
NOTE: A structural diagram will be ***extremely*** helpful before trying to do the model (and is required by the report – see below).

(b) Simulate your model to see that collisions with Martin are possible – give a trace (copy-paste it from the LTSA simulation window to ensure that it's correct).

(c) Define a safety property NO_CRASH, which checks for collisions with Martin.

(d) Verify with LTSA that collisions with Martin can occur using the safety property NO_CRASH.

Put everything inside file martin-your_login.lts (e.g., martin-abcd345.lts).

Part B – Safe Model – Pair (30%)

Now get together with your partner, compare models (you still need to submit your own one for Part A!), select one, modify/correct it if need be ***(keep versions!)***, and:

(e) Add a controller component to the new system (SAFE_MARTIN), so that collisions with Martin are impossible.

(f) Verify formally that SAFE_MARTIN satisfies NO_CRASH.

(g) Add a liveness property (LIVE_MARTIN), and formally verify whether Martin will always be able to eventually exit the road.

(h) Check whether the liveness property is satisfied in the SAFE_CONGESTED model, where enemies heavily congest the road.

Put everything inside file martin.lts.

Part C – Safe Model in Java – Pair (30%)

Implement your final model in Java – ***follow the model as closely as possible*** [1].

Feel free to reuse code from the book applets if that helps – but document what you've used.

Part D – Report – Pair (10%)

Document ***everything*** and submit a PDF (NOT WORD/TEXT/OTHER!).

1) Discuss any relevant problematic issues.
2) Explain your solutions and the motivation for them.
3) For step (a), give an explanation of the meaning of all processes' actions.
4) For step (a) and (e), discuss the relationship between your model and the specification on page 1.
5) For Part C, provide a UML diagram of your Java program and discuss the relationship between your model and your implementation.

Your report should be ***self-contained*** – it should be able for one to understand your solution and the motivation for it without having to read your model file or

[1] *Not many marks for unrelated implementations.*

Java code. So the report should include all necessary and relevant parts of the model and implementation that are needed for the discussion of them.

Be concise and precise – don't explain things to death, stay relevant.

Solutions that appear "out of the blue" don't earn any marks.

Grading

Grading is done based on whether you have demonstrably achieved the following learning objectives [2]:

- **Construct** unsafe and safe models from the specification;
- **Apply** standard solutions to common concurrency problems;
- **Relate** your models to the specification;
- **Simulate** your unsafe model;
- **Define** safety and liveness properties for your models;
- **Verify** the behaviour of your models with respect to your properties;
- **Implement** your model in Java; and
- **Relate** your implementation to your safe model.

Submission

All of you need to submit ***individually*** (Part A, remember?).

- Create a folder named after your City login, using lowercase letters: abcd345 (NOT ABCD345, NOT your name)
- Copy all files inside that folder.
- Produce a zip archive (a ZIP! Not a RAR/TAR/etc.!!!) of your folder and submit that (abcd345.zip).

Better check that when you unzip the archive you obtain a folder and all the files you wanted to submit.

[2] The coursework serves the purpose of helping you learn and therefore pass the final exam (and thus module) – not to give you easy marks. Cheating will put you in the bad position of failing the exam and the module (and a potential Academic Misconduct allegation).