EECS3342 Lab 6:
Part2: Design of Control System for a Train Station

JSO/SH

February 10, 2019

Contents

In preparation for all labs, make sure that you have read and understood the Rodin User's Handbook. Make sure you understand the difference between a *Context* and a *Machine*. Pay careful attention to sections of the Handbook describing the Proving perspective, how to purge a proof and how to use the provers (e.g. see Section 3.4 on Proving). This Lab was written with Rodin Handbook v2.8 and Rodin 3.4 (you may be using later versions and there might thus be some differences).

Ensure that you have completed all previous Labs. In Lab5 you developed Part1 of the Train Station Model.

In Part1, we also asked you to answer 41 Questions. Ensure that you have answered and understood all the questions.

This Lab is Part2, which extends Part 1 further. So it is necessary to consult your development in Lab5 to do this Lab.

To Do

Read the rest of this Lab carefully, where it is explained in detail what you must do.

1. Complete the various models following up from Part1.
2. Ensure that you can answer all the questions.
3. Discharge all the Proof Obligations. Ensure that you understand them.
4. Ensure that you understand the notion of a witness.
5. Simulate your final model with Pro-B to validate it. Use Pro-B to check for deadlock freedom.
6. Learn how to import a project and export a project in a Zip file (e.g. to `trainstation.zip`). This is what you will have to do in Labtest/R.
7. You must also learn how to import a project from a zip file for the coming Labtest/R.
8. You must also document this Lab with Latex.

1 Part1 of Train Station Model

In Part1, you built a model of a software controller for a train station as illustrated by Fig. 1. Please ensure that Part1 is completed and all questions to Part1 answered before proceeding with Part2 below.

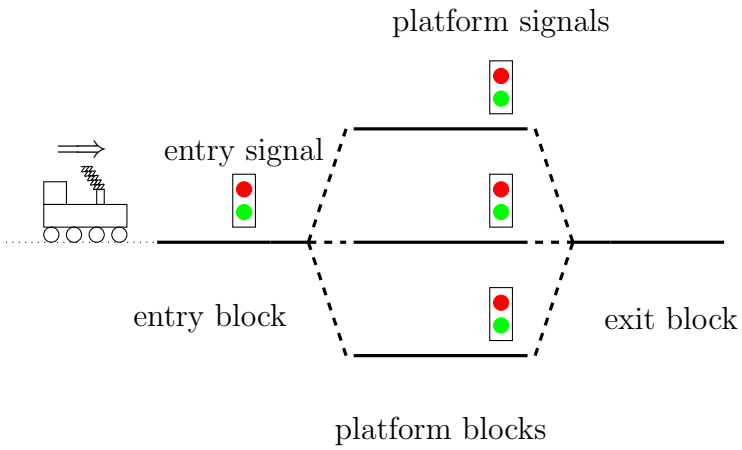


Figure 1: A signal control system

Part1 Modelling the Environment

Model 0

E0 Trains come in the station, run through it and leave.

Model 1

E1 The tracks inside the station are divided into blocks: the unique entrance block, the unique exit block and one block for every single platform.

E2 The tracks of each path through the station is made of the only entrance block, the block of one of the platforms and finally the only exit block.

E3 At any one time, any train inside the station is occupying exactly one of the block.

R4 (Safety) To avoid collisions, every block shall be occupied at most by one train.

In Part1, your last refinement was the machine `p1_m1_blocks`. Two new events were introduced in this refinement. These events must eventually be proved convergent.

- *MOVE_ONTO_PLATFORM*: with potential variant var1: $OCC \cap \{ent\}$.
- *MOVE_OFF_PLATFORM*: with potential variant var2: $OCC \cap PLATFORM$.

In this refinement, we can prove that variant var1 decreases and var2 does not increase. Hence we make *MOVE_ONTO_PLATFORM* **convergent** and *MOVE_OFF_PLATFORM* **anticipated**.

We applied these changes to the model and ensured that we understood and discharged the proof obligations.

In this Lab, we will now do further refinements.

2 Lab 6: Part2

2.1 Convergence of trains: Machine `p1_m2_blocks_variant`

Now refine machine `p1_m2_blocks` and call the refinement `p1_m2_blocks_variant`. Add the second variant and switch the anticipated event to convergent. Make sure all the proof obligations get discharged.

Question 1. The VAR Proof Obligation is expected and easily discharged (Why?). What is the FIN Proof Obligation? Why is it easy to discharge?

Answer. The set variant must be finite: $\mathit{finite}(OCC \cap PLATFORM)$. It follows from invariant: $\mathit{finite}(ST)$

2.2 Track Switches: Machine `p1_m3_switches`

In this refinement, we model a new aspect of the train tracks in the station: the switches. When a train is moving from the entrance block to one of the platforms, a track **switch** connects the entrance block to the block of a single platform and can select different platform and route trains appropriately.

First, in Rodin, refine `p1_m2_blocks_variant` into `p1_m3_switches`

To model the switches, we add two variables: IN_SW and OUT_SW. Variable IN_SW records the platform that a train would be routed to from the entrance when it moves on. Similarly, OUT_SW records the platform that is linked to the exit block. Any train attempting to move from a different platform would derail instead of reaching the exit block.

Question 2. What type should we give to IN_SW and OUT_SW to be consistent with this modelling idea?

- inv1: $IN\_SW \in \_\_$
- inv2: $OUT\_SW \in \_\_$

Answer.

- inv1: $IN\_SW \in PLATFORM$
- inv2: $OUT\_SW \in PLATFORM$

New Events To control the switches, we add two events: `TURN_IN_SWITCH` and `TURN_OUT_SWITCH`.

Question 3. Fill in the blanks:

TURN_IN_SWITCH
begin
$IN\_SW :\in \_\_$
end

TURN_OUT_SWITCH
begin
$OUT\_SW :\in \_\_$
end

Hint: At this point we are not yet at the final form of these events. Use non-determinism to arbitrarily select a platform. For example, TURN_IN_SWITCH must be taken before a train can move onto a platform.

Answer. Events:

TURN_IN_SWITCH
begin
$IN\_SW :\in PLATFORM$
end

TURN_OUT_SWITCH
begin
$OUT\_SW :\in PLATFORM$
end

2.3 Realism and a witness

Question 4. Beside `TURN_IN_SWITCH` and `TURN_OUT_SWITCH`, what are the two events in which IN_SW and OUT_SW are relevant?

Answer.
- IN_SW: $MOVE\_ONTO\_PLATFORM$
- OUT_SW: $MOVE\_OFF\_PLATFORM$

In this refinement, we wish to remove the parameter p in the abstract event `MOVE_ONTO_PLATFORM` and replace it with the concrete variable `IN_SW`. To do this, we need an appropriate **witness** for p as shown below:

Abstract Event

```
MOVE_ONTO_PLATFORM:  extended ordinary
REFINES
  MOVE_ONTO_PLATFORM
ANY
  t
  p
WHERE
  grd1:  t ∈ ST not theorem
  grd2:  LOC(t) = ent not theorem
  grd3:  p ∈ PLATFORM not theorem
  grd4:  p ∉ OCC not theorem
THEN
  act1:  LOC(t) ≔ p
  act2:  OCC ≔ (OCC \ {ent}) ∪ {p}
END
```

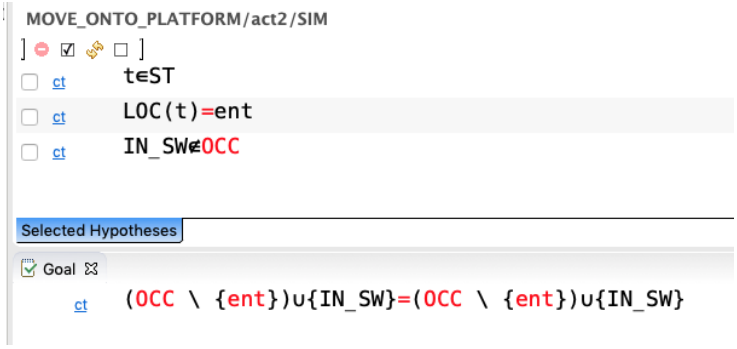
is refined by Concrete Event with Witness p

```
MOVE_ONTO_PLATFORM:  not extended ordinary
REFINES
  MOVE_ONTO_PLATFORM
ANY
  t
WHERE
  grd1:  t ∈ ST not theorem
  grd2:  LOC(t) = ent not theorem
  grd4:  IN_SW ∉ OCC not theorem
WITH
  p:  p = ???
THEN
  act1:  LOC(t) ≔ ???
  act2:  OCC ≔ ???
END
```

Question 5. What is the witness?

Answer. $p = IN\_SW$

Consider the SIM proof obligation below:



Question 6. Rodin simplified the goal for us in the SIM proof obligation. What is the original goal that had to be simplified. Explain its significance.

Answer. When the concrete event is taken, there must be an abstract post-state OCC' that satisfies the abstract before-after predicate, i.e. the original goals is:

$$\exists p \cdot G(t, p, ..) \wedge (\exists OCC' \cdot OCC' = (OCC \setminus \{ent\}) \cup \{p\})$$

and we may use the concrete before after predicate involving OCC in the proof.

Question 7. In the concrete event `MOVE_ONTO_PLATFORM`, what are the actions act1 and act2?

Answer. The actions are as follows:
act1: $LOC(t) := IN\_SW$
act2: $OCC := (OCC \setminus \{ent\}) \cup \{IN\_SW\}$

For the abstract event `MOVE_OFF_PLATFORM`, we had a guard $LOC(t) \in PLATFORM$. This is no longer sufficient in the concrete event. So changes will need to be made consistent with the switches.

Question 8. In the events identified above, an arbitrary platform is involved. Change those events so that the selected platform will be selected consistently with the model of the switches. Start by making the two events "not extended".

Answer.
- *MOVE_ONTO_PLATFORM*
  - remove parameter p
  - add witness p: $p = IN\_SW$
  - remove grd3

  - replace grd4 with $\neg IN\_SW \in OCC$
  - replace act1 with $LOC(t) := IN\_SW$
  - replace act2 with $OCC := (OCC \setminus \{ent\}) \cup \{IN\_SW\}$
- $MOVE\_OFF\_PLATFORM$
  - replace grd2 with $LOC(t) = OUT\_SW$

Question 9. Explain once again informally, what is the purpose of the simulation proof obligations, labelled `SIM`, for the above events?

Answer. It ensures that, although the effect of the event is formulated differently in the refinement than in the abstract machine, all the possibilities of the refinement are also captured by the abstraction.

2.4 Safety

Question 10. What modelling element introduced in this refinement prevents trains from derailing?

Answer. grd2 of $MOVE\_OFF\_PLATFORM$, $LOC(t) = OUT\_SW$

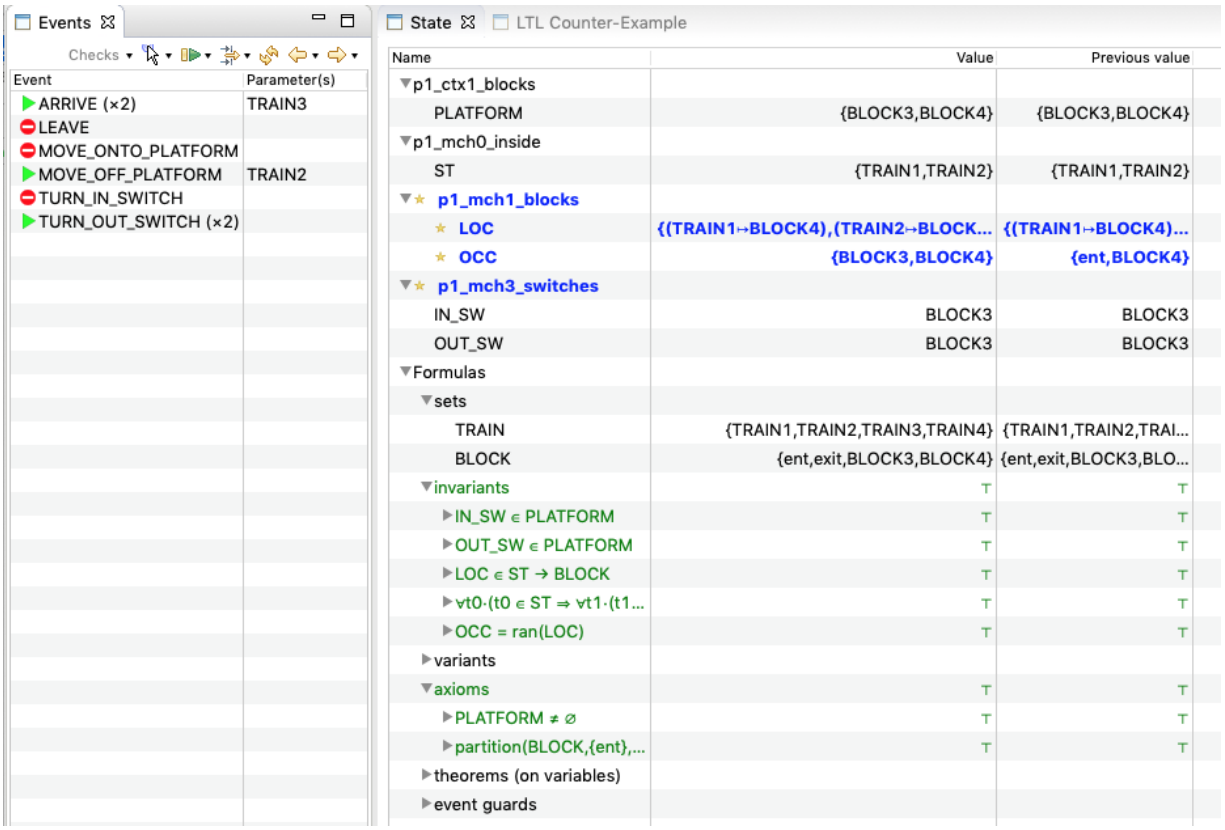
Question 11. Fix the new events so that they correctly specify platform choice. You do not need to deal with divergence.

Question 12. Did you ensure that all the proof obligations are discharged. Do you understand all the proof obligations?

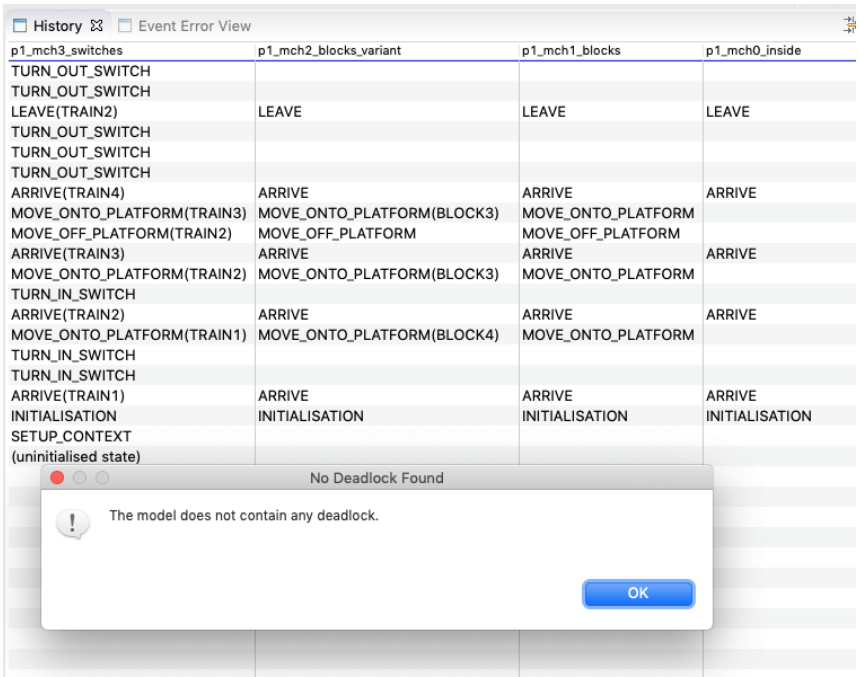
2.5 Pro-B Validation of the model

You must now validate your model with Pro-B. In your Pro-B settings, temporarily set size of unspecified sets to 4. Afterwards you can set it back to the smaller default.

Now run Pro-B on the last refinement and check that the events are enabled and executed correctly.



Ensure that you execute some random simulations. In addition, you can check for deadlock freedom.



This is done from the `Checks: Constraint Based Checking` Pro-B menu.

3 Latex Documentation of a Development

In this Lab, you must practice writing a Report in LaTeX. You will have to do this for your Project. You must generate a *Report.pdf* using Latex. Your report must contain the following:

- Title of your report: EECS3342 – Lab6 – Train System
- Your name and Prism Login used to submit the report.
- Image that all your proofs have been discharged for all the contexts and all the machines. You may use the PDF import, where PDF is generated using the Rodin/LatexPlugin).

In the explore window, if you right click on a context or machine component in the explorer view, you can generate Latex documentation for that component.

The Latex documents are in the workspace for that Project.

See `https://wiki.eecs.yorku.ca/project/sel-students/p:tutorials:latex:start` for how to use Latex on Prism workstations or the SEL-VM.

There is a report template in the SVN.