

CS60110 Exercises 1

1. Work through the **Getting Started with Rodin** tutorial to familiarise yourself with the tool.
2. Refer to the **Controlling Cars on a Bridge** slides/notes. This exercise creates a variation on that case study.

Create a new project called say *MyCarProject*. Create a machine *Mch00*, just like machine *m1* of the notes, **but** without events *ML_in*, *IL_out*, and without any mention of machine *m0*'s variable *n* and constant *d*. So there is an island, and a bridge, variables *a*, *b*, *c*, with their intrinsic invariants, events *ML_out*, *IL_in*, and **no restriction** on how many cars can enter the island-bridge system. (There is also no abstract machine of *Mch00* and no machine variant.) Save *Mch00* and check that all POs are autoproved.

Create a context called say *Ctxd*. Add a constant *d* with axiom ' $d \in \text{NAT}$ '. Save *Ctxd*.

Select *Mch00* in the *Project Explorer* and using the menu item, *Refine* it, calling the refinement *Mch01*. Make *Mch01* **SEE** *Ctxd*. Give *Mch01* the same invariants as *Mch00*. Add the guard ' $a + b < d$ ' to event *ML_out*. Add a further invariant ' $a + b + c \leq d$ ' to *Mch01*. Save *Mch01*.

What goes wrong? Why?

Select *Mch00* in the *Project Explorer* and using the menu item, *Refine* it, calling the refinement *Mch10*. Add events *ML_in*, *IL_out* (previously omitted from *Mch00*) to *Mch10*, making them CONVERGENT events. Give *Mch10* the same invariants as *Mch00*. Add a variant ' $2 * b + c$ ' to *Mch10*. Save *Mch10*.

What goes wrong? Why?

Select *Mch10* the *Project Explorer* and using the menu item, *Refine* it, calling the refinement *Mch11*. Make it **SEE** *Ctxd*. Just as for *Mch01*, add the guard ' $a + b < d$ ' to event *ML_out*. Give *Mch11* the same invariants and the same variant as *Mch01*. Save *Mch11*.

What goes wrong? Why?

Go to the *Dependencies* tab of *Mch11* and change the *Abstract Machine* of *Mch11* from *Mch10* to *Mch01*. Go to the events *ML_in*, *IL_out*, and in each case remove the *Refined Event* attribute, and make them CONVERGENT events instead. Ensure the variant ' $2 * b + c$ ' is included in *Mch11*. Save again.

What goes wrong? Why?

CS60110 Exercise 2

Create a new project called *FileProject*. Duplicate the first part of the development of the **File Transfer Protocol** in the slides. That is, replicate the first three machines and the two refinements that connect them, *except that the data type being transmitted should be three bits*, rather than an unspecified set *D*.

[*Hints*. To make the terminology more transparent, introduce the constant *BIT*, and make it equal to the set $0..1$ in Rodin. To make access to the file contents easier later, model the file as three functions, each from $1..n$ to *BIT*.]

Develop one or more refinements of the previous machines that (a) introduce a parity check bit *pck*, set such that the total number of bits among "the three bits being sent and *pck*" that are 1 is always *even*, (b) introduce an event to flip some bit in transit, (c) check the parity of the bits on arrival and detects errors, (d) do not reduce the acknowledgement mechanism from a sequence number to a simple alternating bit, so that the acknowledgement that is returned distinguishes between success and failure, and contains enough information that the sender knows whether to retransmit the previous block or go on to the next one. [*Hint*. Consider using negative as well as positive numbers. They are in the type *INT*.]

Create a document that records your development called *FileProject.pdf*. Ensure that the identity of the author is clearly attributed. It should contain a discussion of the requirements that the project addresses in both informal and formal form (so the first part of this will largely duplicate what is in the **File Transfer Protocol** slides). It should also contain a discussion of how the design works, concentrating on the new component(s). It should contain the machines and contexts of the development. If your development discharges all proof obligations, say so in the document. If it generates undischarged proof obligations, comment on them in the document.

You must create the document using L^AT_EX, following the instructions and example in `Template.tex` and `Template.pdf` described elsewhere.

If you have time and enthusiasm, think of further enhancements to the above system, and develop and document them. There is one mark available for this.

MARKING SCHEME: Document structure/presentation: 1 mark. Recording the first three models: 3 marks. Development of new material: 5 marks. Any extras: 1 mark. **Total 10.**

SUBMISSION: A printout of your fully documented report should be *handed in* at the beginning of the next lecture session.

CS60110 Exercise 3

Create a new project called *PostalSystemProject*. In this exercise you are to develop a toy postal system from scratch. Given your previous experience, only relatively lightweight guidance is provided. You are expected to fill in requirements gaps using your own judgement.

First model. There are addresses and items. Items move from an origin address to a destination address.

Refinement: Introduce postboxes. Items are first posted into a postbox, then arrive at their destination address.

Refinement: Introduce post offices. Posted items are removed from postboxes, taken to their local office, transmitted to a destination office, and then delivered.

Refinement: Introduce categories and pricing. Items may be letters or parcels, and have various prices. You can introduce different sizes/weights/prices for different kinds of letters/parcels.

If you have time and enthusiasm, you may introduce further enhancements of the model.

Document your development as in Exercise 2; call the document *PostalSystemProject.pdf*.

MARKING SCHEME: Document structure/presentation: 1 mark. Recording the first four models/refinements: 2 marks each. Any extras: 1 mark. **Total 10.**

SUBMISSION: A printout of your fully documented report should be *handed in* at the beginning of the next lecture session.

CS60110 Overall Course Assessment

Exam: 50%

Project: 40%

Exercises 2 and 3: 10%