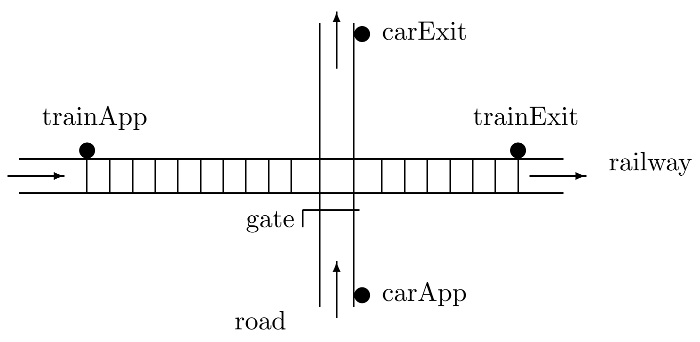
Project of T-79.179 in Spring 2005, μCRL Part: Railroad Crossing

The Railroad Crossing Problem



Consider a railroad crossing whose physical layout is depicted above. There is a road crossing a railway. Cars and trains cross the passageway area in turns. The crossing involves a gate, but unlike most crossings this one keeps the barrier down except when a car actually approaches and tries to cross. The system consist of three main components, the rail, the road, the controller, and its behavior can be described as follows.

There are two sensors along the rail (trainApp and trainExit) respectively telling the controller that a train is approaching and leaving the passageway. Whenever a train approaches and tries to enter the crossing, it triggers the trainApp sensor which sends a signal to the controller. If the train can safely enter the passageway area, i.e. the barrier is down and there are no cars in the passageway, the controller sends a green signal to the train. When the train has safely passed the crossing area, it triggers the trainExit sensor to signal the controller that the train left the passageway.

There are two sensors along the road (carApp and carExit) respectively telling the controller that a car is approaching and leaving the passageway. If a car approaches and tries to enter the crossing, it triggers the carApp sensor which sends a signal to the controller. If the car can safely enter the passageway area, i.e. there is no train in the passageway, the controller opens the gate for the car. When the car has passed the passageway area, it triggers the carExit sensor to signal the controller to close the gate.

The problem is to give a model of the crossing system such that it satisfies (at least) the following properties:

- No collision: it is never possible for a train and a car both to be able to cross at the same time.
- Barriers: whenever a train is in the crossing the gate has to be lowered.

The Assignments

Make the following assignments and send your solutions to `Firstname.Lastname@hut.fi` e-mail address of Misa Keinänen, not later than May 9, 2005. The solutions should be in Latin-1, Latin-9, ISO 8859-15, or 7bit ASCII format.

(a) Write down a recursive specification that models the railroad crossing system. Your specification should consist of recursive equations for components *rail*, *road*, *controller*, and the overall infinite behavior of the *crossing* should be defined as parallel composition of all these components. Use communication, encapsulation etc. to synchronize the components and to force the desired behavior. You should hand over the recursive specification and a detailed explanation of your model.

(b) Give the recursive specification from part (a) as a process declaration in μCRL language. Hand over a correct .mcrl declaration file.

(c) Use μCRL tool set to produce the process graph that belongs to the process declaration from part (b). How large is the state space of your model? Hand over the number of transitions and states.

(d) Use μCRL tool set to show that your model from part (b) does not contain any deadlocks. You should hand over the .dlk file generated from part (b) process declaration.

The μCRL tool set

The manual for the μCRL tool set can be obtained via `http://www.tcs.hut.fi/Studies/T-79.179/`. The μCRL software has been installed to some of the HUT Computing Centre workstations. The tools can be run in all Debian Linux machines of Computing Centre (thus in all those Unix machines that are considered by `http://www.hut.fi/cc/computers/`), in `vipunen.hut.fi`, and in `saha.hut.fi`. In these workstations, the command

use -q mcrl

allows you to use the needed tools.

If you prefer to use μCRL on your own PC, you can download the source files via `http://homepages.cwi.nl/~mcrl/index.html` and follow the installation instructions therein.