Take-home Examination IA164: Formal Methods for Software Engineering

Autumn 2008

Exercise 2 has to be made with the LTSA tool. Exercise 1 has to be made with a Z tool for typechecking, like Z-Eves (but any other Z tool the student can find on the web is also OK).

The examination has to be submitted electronically to `langerak@cs.utwente.nl`. Deadline: Monday 24 November 2008, 10.00h.

Points:
Exercise 1: 8 points, exercise 2: 8 points, exercise 3: 4 points (total 20 points).

---

1. You are asked to specify a system for project management support. Suppose there are projects, tasks, and employees (define basic types PROJ, TASK, and EMP). Each task belongs to a project. To each employee tasks can be assigned on a FIFO basis (so someone should finish a task he gets earlier before a task he gets later). In order to avoid stress, each employee can be assigned at most a number of *maxtask* tasks. A task can be assigned to at most one employee.

   (a) Specify the system and give the initial state.

   (b) Specify an operation that assigns a new task $t?$ from project $p?$ to employee $e?$. The operation should be robust. Use schema composition.

   (c) Employee $e?$ has finished the first task on his list. Specify an operation that deletes this task from the system. Give the precondition for this operation.

   (d) Specify an operation that gives for a project $p?$ all the employees that have been assigned tasks from this project.

2. In a sauna a customer can reserve a cabin at the entrance. After a cabin has been reserved, as soon as a cabin is free it is turned on and its number is given to the client. The client goes to the designated cabin; after each 5 minutes either the cabin stops, or the client pushes a "more" button for another 5 minutes of sauna. The maximum time a customer can get is 15 minutes. After the cabin stops it signals to the entrance the amount the customer has to pay (1 unit for every 5 minutes). Finally the customer pays at the entrance.

   (a) Specify the sauna for 1 customer and 1 cabin. Take the follwing process for the entrance:

   ```
   ENTRANCE = ( reserve -> turn_on -> get_number -> ENTRANCE
              | amount[m:M] -> pay[m] -> ENTRANCE).
   ```

   Assume a customer that stays for 15 minutes looks like this:

   ```
   CUSTOMER = (reserve -> get_number ->
                 more -> more -> end -> pay[m:M] -> CUSTOMER).
   ```

   (b) Now specify a sauna with 2 cabins and 3 customers (hint: carefully think about how to adapt the process ENTRANCE to avoid deadlocks). You may assume that a customer stays 10 minutes.

(c) Specify a safety property that states that the difference between the number of `reserve` actions and the number of `pay` actions is never bigger than 2. Does this property hold for you sauna?

(d) Specify a progress property stating that customer 3 will eventually get a cabin. Does this proprty hold? Now give priority to the `reserve` actions of customers 1 and 2. Does the property still hold?

3. We want to model a shop, two customers and a clerk. The customers may enter the shop with rate λ_1 resp. λ_2, perform action `enter`, in synchronization with the shop, action `serve`, in synchronization with the clerk, and start again. The clerk is called with action `call`, and then performs action `serve` with rate ν. After a customer has entered the shop, the shop calls the clerk via action `call` with rate μ. After synchronizing on action `call` the shop starts again.

   (a) Specify the shop, customers and the clerk, and their composition (do not yet hide any actions).

   (b) Give a formal derivation of a sequence of actions where a customer enters the shop, and the clerk is being called.

   (c) Now hide all actions, and give the resulting transition system.

   (d) Give a minimal CTMC that is bisimulation equivalent to the transition system in (c).