Practical Class 8 — Liveness Specifications[1]

This practical class uses FDR to check specifications involving failure refinement.

Getting Started

1. Start FDR and an editor.

The Level Crossing

1. Copy the file `crossing.fdr2` to your directory. Do not overwrite the work you did on the level crossing in Practical Class 4 — you will need it later.
2. Define the processes $CARSPEC$ and $TRAINSPEC$ from the lecture, and enter the liveness specifications from the lecture as assertions involving failure refinement (written `[F=` for FDR).
3. Use FDR to check the liveness assertions. Both of them should be true.
4. Define an additional control process: $NEW\_CONTROL = gate.lower \rightarrow STOP$ and put it in parallel with $SAFE\_SYSTEM$ (synchronising on the *gate* events) to get NEW_SAFE_SYSTEM, in which the gate can be lowered once but never raised again. Check the $TRAINSPEC$ specification for NEW_SAFE_SYSTEM. Why is it not satisfied?
5. In Practical Class 4 you redefined $CONTROL$ so that after a train has crossed, the gate can stay down if another train is approaching. Change $CONTROL$ to that definition again now, and check the assertions again (still with $NEW\_CONTROL$ as well). You should find that $TRAINSPEC$ is satisfied.

[1] This practical sheet is provided as course material to accompany the book 'Concurrent and Real Time Systems'.

6. Double-clicking on a failed specification (or selecting "Debug" from the "Assert" menu) brings up a debug window which shows the failure pair of the process which is not allowed by the specification. You will see a trace and a set of events; by default this set is an *acceptance set*, which is the opposite of a refusal set, but this can be changed by using the buttons at the bottom of the window. If you select "Refusals", what you see is a *maximal refusal set*; the process is refusing all sets which are subsets of the maximal refusal set. Clicking on the "Allowed" button brings up a window which shows the set of events refused by the process, and the set of events refused by the specification. This identifies the events being refused by the process, which the specification requires to be accepted.

The Cyclic Scheduler

1. Copy the file `scheduler.fdr2` to your directory, or use the definitions which you produced in Practical Class 7.
2. Add the assertion

   ```
   CYCLE(0) [F= (SCHED \ {|finish|})
   ```

   and check it — it should be true.
3. Now check the assertion

   ```
   ALTSPEC [F= SCHED
   ```

   Why is it not true? Write a new assertion which expresses a liveness specification for alternation of $start.0$ and $finish.0$, and check it.
4. Check the assertions again for $NEWSCHED$, the alternative definition of the scheduler, either using the definitions from the supplied file, or your own definitions from Practical Class 7.
5. Check that each of $SCHED$ and $NEWSCHED$ is a failure refinement of the other; this means that they are failure equivalent (and therefore that they must satisfy exactly the same specifications as each other; why is this?).