CS375 Practical Class 4 — Using FDR

This practical class introduces the tool FDR, and begins to use it to investigate specifications of processes.

Getting Started

1. Connect to `tartan` as usual, move to your CS375 directory, and start an editor.
2. Type `fdr2 &`. You should see two windows: one entitled "FDR 2.11", and a file chooser window.

Specifications

Recall that if P is a process, then $traces(P)$ is the set of finite sequences of events which P can do. An important relationship between processes is *trace refinement*. We say that P is *refined by* Q (written $P \sqsubseteq_t Q$) if $traces Q \subseteq traces(P)$. This means that every trace which Q can generate, can also be generated by P.

Refinement is used to write specifications. If we think of P as defining the maximum range of allowable behaviour, then the statement $P \sqsubseteq_t Q$ is a specification that Q should only do things which are allowed by the definition of P.

For example, if $P = a \rightarrow P \mid b \rightarrow P$, we can think of P as the specification "only do events a and b". The statement $P \sqsubseteq_t Q$, interpreted as a specification of Q, says that Q should only do events a and b.

If we define $Q = a \rightarrow b \rightarrow Stop$, it is true that $P \sqsubseteq_t Q$ because every trace of Q is also a trace of P (check this: what are the traces of Q? What are the traces of P?).

If we define $R = a \rightarrow c \rightarrow Stop$, it is *not* true that $P \sqsubseteq R$. The trace $\langle a, c \rangle$ is a trace of R but not a trace of P.

FDR can be used to check trace refinement, i.e. to check whether or not a process satisfies a specification.

1. Copy the file `/CS/ftp/pub/CS375/spec1.fdr2` to your directory. It contains the definitions of P, Q and R above. Notice the lines

   ```
   assert P [T= Q

   assert P [T= R
   ```

   which are machine-readable syntax for $P \sqsubseteq_t Q$ and $P \sqsubseteq_t R$.
2. Load `spec1.fdr2` into FDR by selecting it in the file chooser window. You should see the process names P, Q and R in the lower part of the FDR window, and the assertions `P [T= Q` and `P [T= R` above.
3. Double-click on the assertion `P [T= Q`. After a short pause, a tick should appear next to it. This indicates that the assertion is true, i.e. that Q satisfies the specification expressed by P.
4. Double-click on the assertion `P [T= R`. This time a cross appears, indicating that R does not satisfy the specification. Therefore there must be a trace of R which is not a trace of P.
5. To see what that trace is, select "Debug" from the "Assert" menu. You should get another window, which after a few seconds will display the trace $\langle a, c \rangle$ of process R. This is the trace (or one of them, if there are several) which is not allowed by the specification P.

Specifying Student Prizes

1. Copy the file `/CS/ftp/pub/CS375/spec2.fdr2` to your directory and load it into FDR. It contains the definitions of *STUDENT* and *COLLEGE* (the first definition of *COLLEGE*, which stops after *fail*). The process *SPECP* is a specification which can do three *pass* events followed by *prize*. Notice that after a *fail* event, *SPECP* allows any sequence of *pass* and *fail* events, but the only way *prize* can occur is if there are three *pass* events first, and no *fail* events.

2. Use FDR to check the assertion that $SPECP \sqsubseteq_t SYSTEM$. You should find that it is not true. Select "Debug" from the "Assert" menu to see the trace which is outside the specification. What is the problem?

3. Although we are only interested in the events *pass* and *prize*, *SYSTEM* also does other events (e.g. *year1*) and unless we put them into the specification too, *SYSTEM* cannot satisfy it. Define a process *EXTRA* which can do the events *year1*, *year2*, *year3*, *graduate* as much as it likes. Define a new process: $SPEC = SPECP\ {}_{SP}\|_{E}\ EXTRA$, where *SP* and *E* are suitable alphabets. Add these definitions to the file.

4. Change the assertion to $SPEC \sqsubseteq_t SYSTEM$ and check it again. Is the specification satisfied now?

5. Notice that we still have the incorrect definition of *COLLEGE*, which stops after *fail*, but the specification is satisfied. The specification says that a prize should not be awarded if *fail* has happened, and indeed *COLLEGE* satifies this requirement. This illustrates the point that trace refinement only specifies safety. However, we would like to check the correct definition of *COLLEGE* too. Change the definition, and check the assertion again.

Simplifying *EXTRA*

When writing trace specifications, it is quite common to have to define a process such as *EXTRA*. The definition can be simplified by using a menu choice, as follows.

$$\begin{aligned} E &= \{year1, year2, year3, graduate\} \\ EXTRA &= x : E \rightarrow EXTRA \end{aligned}$$

For reasons which will become clear later, the machine-readable syntax for this definition is:

```
EXTRA = [] x:E @ x -> EXTRA
```

1. Change the definition of *EXTRA* to this new form, and check the assertion again.