Submission deadline and procedure: not assessed.

Introduction

This is not an assessed exercise. You will be given assistance with the tools and with coding problems. Feedback and assistance will be provided over the course of the lab session. Sample solutions will be discussed, and further feedback will be provided, as appropriate.

Background

In the lectures, we have introduced the language of Communicating Sequential Processes (CSP), and some of the basic operators used to construct CSP processes. We looked at several examples (including the vending machine) of how CSP processes can be used to model real world entities. In this lab exercise we will familiarise ourselves with the tool used to analyse CSP processes (FDR) and the facilities it provides for us. The primary objective of this lab exercise is for you to become proficient in editing CSP-M scripts, and investigating those scripts in FDR.

Requirements

For these labs, you will need to be able to access the FDR tool. You will also need a text editor (as FDR is not an editor or an IDE). A useful editor is emacs, or alternatively Notepad. Microsoft Word is not a suitable text editor.

You will find the CSPM script vending.csp available on Course Resources – you should download a copy of this script.

Task 1

Getting started with FDR.

Load up the vending.csp script in your text editor. In this script you will see the vending machine models used in the lectures. The script includes the necessary channel (event) declarations, some sets of events (used for convenience in our models) and the definitions of each of the processes (models) presented in the lectures. Take some time to study this script so that you understand the difference between all of these, and how the process definitions have been constructed.

The "File" menu in FDR contains the option "Open". Use this to load the vending.csp script into FDR. If this script loads correctly, the main window should remain unchanged but the prompt will have changed to "vending.csp". On the right hand side you will see two other windows: Assertions, and Tasks. These should remain unchanged for now – more on these later.

Submission deadline and procedure: not assessed.

Task 2

Exploring FDR.

If you type the command ":help" into FDR, you will see a list of available commands, with a short explanation as to what they do. Many of these are not of interest to us at the moment (but will be later!). Commands that are of interest to us just now are:

- Graph
- Probe
- Processes

Type ":processes" into the tool. FDR will respond with a list of processes that it knows about. The result should look something like this:

```
vending.csp> :processes
BUFFER BadOptionVM BetterVM BrokenVM CHAOS
CompVM DIV OptionVM PeculiarVM RUN
SIGNAL_BUFFER SKIP STOP WEAK_BUFFER
```

Note that this list includes the proceses that we defined in our script, as well as some others that we will explore during this module.

Type in the command ":graph BrokenVM". FDR should respond with a picture of the graph of our BrokenVM process. This is a relatively simple graph: study how it relates to, and describes, the structure of our BrokenVM process. Repeat this exercise for the other processes we have defined and ensure you understand how the graphs describe the process structures.

Type in the command ":structure BrokenVM". FDR should respond with a description of the structure of this process. It will report that this is a low level machine (this will become clear later in the module when we explore concurrency). It will also report the initials of this process – the first events available, which in this case is the solitary event "coin". It will report the alphabet of the process. It will also report leaf machines – other processes that this process calls. In this case, the only other process called is the recursion so it will report only BrokenVM. Repeat this exercise for the other processes we have defined and ensure you understand the descriptions of process structures.

Type in the command ":probe BrokenVM". FDR should respond with a window that allows us to animate (or step through) the BrokenVM process. In this window you will see the name of the process (BrokenVM) and the initials (in this case the solitary initial coin). In our BrokenVM, after a coin is inserted, the machine breaks and the process behaves as STOP. Probe shows us this be indicating that after coin, is STOP.

A slightly more interesting example is BetterVM. Type the command ":probe BetterVM". Now we see the process name, followed by the initial coin. We can see that after this initial is the event choc. Clicking on coin has the effect of making this event happen – and we see that the available event now is choc. We can repeat tracing through this process until we reach STOP. Repeat this exercise for the other processes we have defined, and ensure you understand both the processes descriptions, and how probe allows you to step through and investigate them.

Submission deadline and procedure: not assessed.

Task 3

Writing our own CSPM!

Now you get the chance to try to write some of your own CSPM. With the original script loaded into your text editor, add a process called "MyVM" at the bottom. Using the definition of CompVM as a starting point, try to develop a vending machine that in addition to 100p and 50p coins, will also accept 20p coins. In doing this you will need to declare a new channel, and new behaviours. You will need to reload your script in FDR to see your new process. In doing so, it is likely that you will have some errors in your definition – FDR will report on syntactic errors when you load your amended script (in the same way a compiler does when you compile a program). Try to interpret the errors so that you can load it correctly. When it loads correctly, investigate your new process using the utilities we have practised above.

Submission

You are not required to submit your work for this exercise as it is not an assessed exercise. However you should make sure you have completed this exercise, especially practising and understanding use of FDR, as later laboratory work will build on this exercise.