Exercises

Concurrency and Distributed Systems

January 2022

Contents

FDR

If you do not already have a copy, you may download the FDR tool from

```
https://cocotec.io/fdr/index.html
```

Once you have installed it, you are ready to load your first script, and explore or check your first process.

The command `:help` will show you the available commands.

If you are working not in your home directory, but in a subdirectory `path`, then `:cd <path>` may save time.

The command `:load myfile` loads the script `myfile.csp`.

Vending machines

A simple vending machine may be defined in terms of the following transactions:

- `coin`: the user inserts a coin into the machine (and the coin is accepted);
- `tea`: the machine provides the user with a cup of tea (and that cup is accepted);
- `coffee`: the machine provides the user with a cup of coffee (again, the cup is accepted).

Following insertion of a `coin`, one version of the vending machine, `VM1`, is ready to engage in `coffee` and also ready to engage in `tea`. As far as the machine is concerned, either of these could be the next transaction.

Another version of the machine, `VM2`, will be ready to engage in exactly one of these transactions. That is, following the insertion of a `coin`, it will make an internal choice, and offer either `coffee` or `tea`.

A third version of the machine, `VM3`, will be ready to engage in at most one of these transactions. That is, following the insertion of a coin, it may offer `coffee`, or `tea`, or nothing at all.

None of these machines will accept another `coin` unless they have dispensed a drink.

Create a file `exercises02.csp` starting with the channel declaration

```
channel coin, tea, coffee
```

Check that your script loads successfully into the tool. If it doesn't, fix the problem(s) and try again.

The command `:reload` reloads the current script.

The `up arrow` key allows you to scroll through the list of previous commands.

Now write definitions for the three processes `VM1`, `VM2`, and `VM3`.

Draw graphs of your processes with the `:graph` command.

Explore their traces with the `:probe` command.

Deadlock

It is often useful to check whether a process can deadlock: that is, reach a state in which it is unable to engage in any future transactions.

We may determine whether or not a process `P` can deadlock by running a check in the tool:

```
assert P :[deadlock free]
```

Try this for each of your vending machine processes.