A2: Analysing behaviour

Crossing the river – Part 2

David Pereira & José Proença

RAMDE – 2021/2022

To do: Produce a report as a PDF document including the answers to the exercises below.

To submit: The PDF report and a new `vending.mcrl2` file (for Exercise 3), placed in your group's git repository. ALL students should push commits.

Deadline: TBD

Auxiliary files: You will need the 3 files produced in your last assignment: `farmer1.mcrl2`, `farmer2.mcrl2` and `farmer3.mcrl`.

Verification of the farmer-fox-goose-beans problem

Recall the specifications in the `farmer1`, `farmer2`, and `farmer3` projects from the modelling exercises (`https://cister-labs.github.io/ramde2122/assignments/a2-modelling.pdf`) You will now verify properties of these systems. In `mcrl2ide`, a property can be written using `Tools>Add Property`. There are 2 types of properties: **Equivalence** and **Mu-Calculus**, covered by this assignment.

LTS Equivalence

Exercise 1. Create variations of the `Sys` processes in `farmer1` and `farmer2` and compare them to the originals as follows.

1.1. Create a new process `SysHide` in both `farmer1` and `farmer2` equal to `Sys` but hiding all allowed actions except `win` (using *`hide`*). **Show the resulting `SysHide` processes for each file.**

1.2. Combine both specifications of `farmer1` and `farmer2` in a single specification. Rename `Sys` from `farmer1` to `Sys1` and `SysHide1`, and similarly for `Sys` from `farmer2`. Redefine the function **`ok`** by setting it to true, i.e., define **`ok`**`(fm,f,g,b)=true;`.

Visualise the processes `SysHide1` and `SysHide2`. Compare them using strong bisimulation by adding a new **Equivalence** property that compares them. **What can you conclude?**

Verification of properties

Exercise 2. Answer the questions below on the use of mu-calculus for specifying properties in mCRL2.

2.1. What does the property "`[true*]<ready>true`" mean? Does it hold in any of these 2 LTSs?

2.2. Does the property "`[true*.foxr.win]false`" holds for `farmer1`? Does the equivalent property "`[true*.fox(right).win]false`" holds for `farmer2`? What can you conclude?

2.3. Consider now the extended system `farmer3`. In this example there is a an extra process called `Counter(n:Nat)`. **Define the following two properties** over actions of this counter:

1. It is possible to win after exactly 7 moves.
2. It is not possible to win in less than 7 moves.

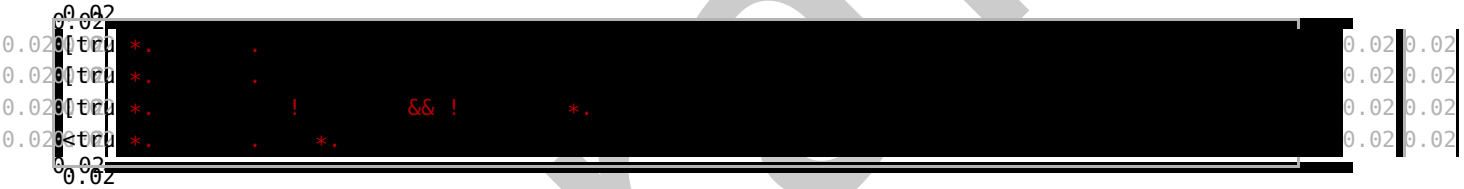
Modelling a vending machine

Exercise 3. Specify two interacting processes in mCRL2:

- **a vending machine** with 2 products, apples and bananas, costing 1eur and 2eur respectively; and
- **a user** who can insert 1eur or 2eur coins and request for products.

Provide two variations of this system and include them in files `vending1.mcrl2` and `vending2.mcrl2`, respectively, according to the requirements below. Try to keep the specifications simple. **Submit this file in your git repository.**

3.1. Specify in `vending1.mcrl2` a system such that the properties below hold.



Show your specification and **show a screenshot of its LTS**.

3.2. Specify another system in `vending2.mcrl2` such that the properties below hold.

Show your specification and **show a screenshot of its LTS**.

Self-peer-evaluation

Exercise 4. In a scale from 0-5, where 5 is better than 0, give a mark to you and each of your team groups for each of the following criteria:

- **Effort** (time spent)
- **Quality** (of the work produced)
- **Collaboration** (how easy it was to meet and interact)

Send this information individually by e-mail or via Teams to David Pereira and José Proença.