Assignment 1: The rope bridge - in MCRL2

Luís Albuquerque (A79010) e Rafaela Pinho (A77293)

April 13, 2018

Contents

1 Introdução

2 Apresentação do Exercício

" In the middle of the night, four adventurers encounter a shabby rope bridge spanning a deep ravine.

For safety reasons, they decide that no more than 2 persons should cross the bridge at the same time and that a flashlight needs to be carried by one of them on every crossing. They have only one flashlight. The 4 adventurers are not all equally skilled: crossing the bridge takes them 1, 2, 5 and 10 minutes, respectively. A pair of adventurers cross the bridge in an amount of time equal to that of the slowest of the two adventurers.

One of the adventurers quickly proclaims that they cannot get all four of them across in less than 19 minutes. However, one of her companions disagrees and claims that it can be done in 17 minutes. We shall verify this claim and show that there is no faster strategy using mCRL2."

No ficheiro que o professor disponibilizou já era dado a seguinte informação:

1. Como que deve ser codificada a posição da lanterna e dos aventureiros;
2. É nos dado a dica que os aventureiros são identificados pela velocidade;
3. É nos dado o processo e as ações da lanterna.

```
sort Position = struct start | finish;
 act
  % Action declarations:
  % - 'forward'  means:  from start to finish
  % - 'back'     means:  from finish to start

  % The flashlight moves forward together with two adventurers,
  % identified by the action's parameters.
  forward_flashlight: Int # Int;

  % The flashlight moves back together with one adventurer, identified
  % by the action's parameter.
  back_flashlight: Int;

  % The Flashlight process models the flashlight:
  % 1. If it is at the 'start' side, it can move forward together with any
```

```
%     pair of adventurers.
% 2. If it is at the 'finish' side, it can move back together with any
%     adventurer.
proc Flashlight(pos:Position) =
    (pos == start) ->
     % Case 1.
         sum s,s':Int . forward_flashlight(s,s') . Flashlight(finish)
     <>
     % Case 2.
         sum s:Int . back_flashlight(s) . Flashlight(start);
```

3 Exercicio 1

3.1 Parâmetros do processo Aventureiro

O Aventureiro tem duas ações possíveis.

1. Atravessar a ponte no sentido start para finish: forward_adventure
   Uma vez que o máximo de aventureiros a atravessar a ponte é 2, é lógico que sempre que atravessem neste sentido vão aos pares. Como a identificação dos aventureiros é pela sua velocidade, forward_adventure tem como parâmetro dois inteiros, que corresponde ás suas velocidades.

   ```
   forward_flashlight: Int # Int;
   ```

2. Atravessar a ponte no sentido finish para start: back_adventure
   O objetivo do problema é todos atraveçem a ponte, por isso o mais lógico será apenas um aventureiro regressar para o "start". Daí os parmetro seja apenas um inteiro, a sua velocidade.

   ```
   back_flashlight: Int;
   ```

3.2 Processo do Aventureiro

O processo Aventureiro recebe a sua posição, ou "start" ou "finish" e a velocidade, que é o seu identificador. E caso a posição seja "start" o pretendido é avançar para o outro lado da ponte, para isso nós escrevemos as possibilidades, poderíamos usar o operador + e fazer

```
foward_adventure(0,speed ).Adventure(finish,speed) +
foward_adventure(1,speed ).Adventure(finish,speed) + ...
```

Mas o MCRL2 tem um comando, "sum", e tal como foi feito para o "Flashligth" pelo professor, permite fazermos um "+" para ações infinitas.
Depois foi só testarmos qual era o que tinha menor velocidade, ou o "s" gerado pelo "sum", ou o argumento do processo, para modelarmos apenas uma das possibilidades que sabemos que são iguais.
Já no caso de a posição ser "finish", não temos escolhas nem testes a fazer, basta o aventureiro chamado pelo processo regressar ao "start" para acompanhar outro aventureiro, uma vez que só poderão atravessar com a lanterna.

```
proc Adventurer (pos:Position , speed:Int) =
        (pos == start ) ->
        sum s: Int .(s > speed)->
                foward_adventure(speed,s ).Adventure(finish,speed)
                <>
                foward_adventure(s,speed).Adventure(finish,speed)
        <>
        back_adventure(speed).Adventure(start,speed);
```

4 Exercicio 2

4.1 Inicializar os 4 aventureiros e a lanterna

Os aventureiros começam todos do lado "start" e com a sua respetiva velocidade. Para os colocar em paralelo usamos "||".

```
init
  Adventure(start,1) || Adventure(start,2) || Adventure(start,5) ||
  Adventure(start,10) || Flashligth(start)
  Flashligth(start) ;
```

4.2 Comunicação de processos

Para que os proçessos comuniquem usamos o comando "comm".

```
init
 comm (
 {
  forward_adventurer | foward_adventurer | forward_flashligth -> forward,
  back_adventurer | back_flashligth -> back
  },

  Adventure(start,1) || Adventure(start,2) || Adventure(start,5) ||
```

```
  Adventure(start,10) || Flashligth(start)
);
```

4.3 Garantir que apenas as ações "forward" e "back" possam ocorrer.

Para que apenas os processos "forward" e "back" possam ocorrer usa-se o comando "allow"

```
init
allow({ forward, back },
 comm (
 {
  forward_adventurer | foward_adventurer | forward_flashligth ->forward,
  back_adventurer | back_flashligth -> back
  },
  Adventure(start,1) || Adventure(start,2) || Adventure(start,5) ||
  Adventure(start,10) || Flashligth(start)
  )
);
```

5 Exercicio 3

5.1 mcrl22lps

5.2 lpsxsim

6 Exercicio 4

6.1 lps2lts

6.2 ltsgraph

6.3 ltsview

7 Exercicio 5

8 Conclusão

*Bibliografia

AGUIRRE, L. A. Introdução ... Identificação de Sistemas, Técnicas Lineares e Não lineares Aplicadas a Sistemas Reais. Belo Horizonte, Brasil, EDUFMG. 2004.

A Código do Exercício

```
act   forward_adventurer,
      forward_flashlight,
      forward_referee,
      forward: Int # Int;

      back_adventurer,
      back_flashlight,
      back_referee,
      back: Int;

      report: Int;
```