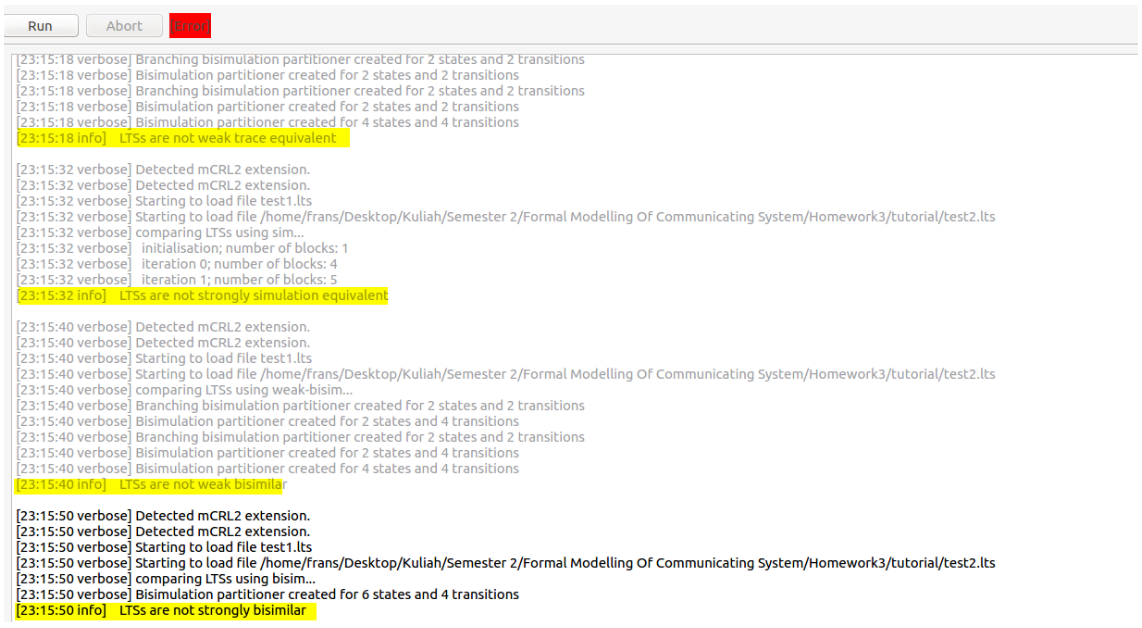
Homework 3 - Formal Modeling of Communicating System

Frans Juanda Simanjuntak - S3038971

April 4, 2017

1 Question 1

The process in the example 1 and example 2 **are not behaviorally equivalent**. It can be verified using *ltscompare* in mcrl2 and the results show that they are not weak trace equivalent nor strongly simulation equivalent nor weak bisimilar and nor strong bisimilar. Have a look at Figure 1 for the detail.



```
[23:15:18 verbose] Branching bisimulation partitioner created for 2 states and 2 transitions
[23:15:18 verbose] Bisimulation partitioner created for 2 states and 2 transitions
[23:15:18 verbose] Branching bisimulation partitioner created for 2 states and 2 transitions
[23:15:18 verbose] Bisimulation partitioner created for 2 states and 2 transitions
[23:15:18 verbose] Bisimulation partitioner created for 4 states and 4 transitions
[23:15:18 info]    LTSs are not weak trace equivalent

[23:15:32 verbose] Detected mCRL2 extension.
[23:15:32 verbose] Detected mCRL2 extension.
[23:15:32 verbose] Starting to load file test1.lts
[23:15:32 verbose] Starting to load file /home/frans/Desktop/Kuliah/Semester 2/Formal Modelling Of Communicating System/Homework3/tutorial/test2.lts
[23:15:32 verbose] comparing LTSs using sim...
[23:15:32 verbose]   initialisation; number of blocks: 1
[23:15:32 verbose]   iteration 0; number of blocks: 4
[23:15:32 verbose]   iteration 1; number of blocks: 5
[23:15:32 info]    LTSs are not strongly simulation equivalent

[23:15:40 verbose] Detected mCRL2 extension.
[23:15:40 verbose] Detected mCRL2 extension.
[23:15:40 verbose] Starting to load file test1.lts
[23:15:40 verbose] Starting to load file /home/frans/Desktop/Kuliah/Semester 2/Formal Modelling Of Communicating System/Homework3/tutorial/test2.lts
[23:15:40 verbose] comparing LTSs using weak-bisim...
[23:15:40 verbose] Branching bisimulation partitioner created for 2 states and 2 transitions
[23:15:40 verbose] Bisimulation partitioner created for 2 states and 4 transitions
[23:15:40 verbose] Branching bisimulation partitioner created for 2 states and 2 transitions
[23:15:40 verbose] Bisimulation partitioner created for 2 states and 4 transitions
[23:15:40 verbose] Bisimulation partitioner created for 4 states and 4 transitions
[23:15:40 info]    LTSs are not weak bisimilar

[23:15:50 verbose] Detected mCRL2 extension.
[23:15:50 verbose] Detected mCRL2 extension.
[23:15:50 verbose] Starting to load file test1.lts
[23:15:50 verbose] Starting to load file /home/frans/Desktop/Kuliah/Semester 2/Formal Modelling Of Communicating System/Homework3/tutorial/test2.lts
[23:15:50 verbose] comparing LTSs using bisim...
[23:15:50 verbose] Bisimulation partitioner created for 6 states and 4 transitions
[23:15:50 info]    LTSs are not strongly bisimilar
```

Figure 1: LTS Verification.

These examples are not behaviorally equivalent because:

- The process in example 1 will always accept the coin and perform giveItem regardless the number of coins we inserted.
- The process in example 2 will accept the coin and perform giveItem if the input parameter is equal to or less than 10.

2 Question 2

The extended version of vending machine with data in order to provide different output corresponding to the number of coins as follows:

```
act
 giveItemA, giveItemB, giveItemC, takeItem;
 ins, acc, coin: Nat;
proc
 M = sum i:Nat.acc(i).((i == 10) -> giveItemA <>
                        (i == 20) -> giveItemB <>
                        (i == 5) -> giveItemC).M;
 U(n: Nat) = ins(n).takeItem.U(n);
init
 allow ({coin, giveItemA, giveItemB, giveItemC},
   comm (
        {ins|acc -> coin},
          U(10) || M
     )
);
```

The LTS graphs of the process is shown in Figure 2, Figure 3, and Figure 4 respectively.

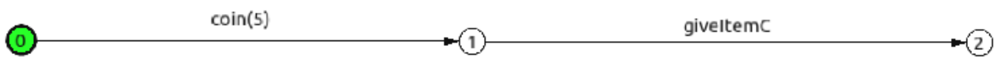


Figure 2: LTS graph with number of coin 5.

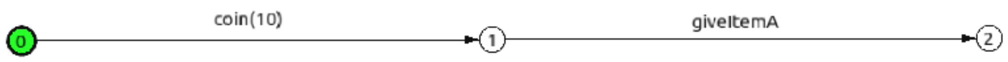


Figure 3: LTS graph with number of coin 10.

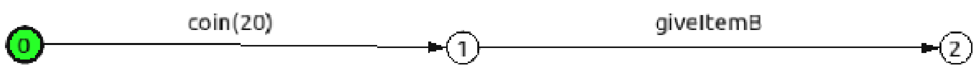


Figure 4: LTS graph with number of coin 20.

3 Question 3

- The representation of SMUni in mCRL2

```
act
  insertCoin ,acceptCoin ,takeCoffee ,giveCoffee;
  submitPub,coin ,coffee;
proc
 CM= acceptCoin.giveCoffee.CM;
 CS= submitPub.insertCoin.takeCoffee.CS;
 SMUni = CM || CS;
init
  hide({coin ,coffee},
  block({insertCoin ,acceptCoin ,takeCoffee ,giveCoffee},
 comm({insertCoin|acceptCoin -> coin ,
         giveCoffee|takeCoffee -> coffee},
     SMUni
  )));
```

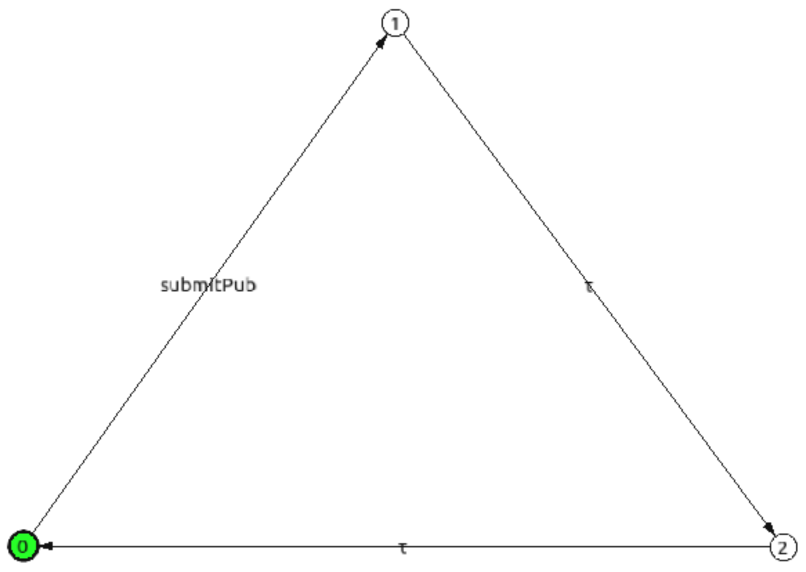


Figure 5: LTS graph of SMUni.

- The extended model of SMUni to represent the publications which have three or more computer scientists as authors as follows:

```
act
 insertCoin ,acceptCoin ,takeCoffee ,giveCoffee ,submitPub;
 acceptPub ,coin ,coffee ,paper;
proc
 CM= acceptCoin.giveCoffee.CM;
 CS1= submitPub.insertCoin.takeCoffee.CS1;
 CS2= submitPub.insertCoin.takeCoffee.CS2;
 CS3= submitPub.insertCoin.takeCoffee.CS3;
 PUB= acceptPub.PUB;
 SMUni = CM || (CS1 || CS2 || CS3 || PUB);
init
 hide({coin ,coffee},
 block({insertCoin ,acceptCoin ,takeCoffee ,giveCoffee},
 comm({insertCoin|acceptCoin -> coin ,
        giveCoffee|takeCoffee -> coffee ,
        submitPub |acceptPub -> paper},
     SMUni
 )));
```

New process namely **PUB** was created in order to handle the publication from different authors. CS and PUB will be synchronized through **paper**. CS will submit the paper and PUB will accept it.