

Solutions to Exercises

Model-based Design of Distributed and Mobile Systems Lecture 2: Spring 2004

Mikkel Bundgaard
mikkelbu -at- itu.dk

Theory Department
IT University of Copenhagen

17th February 2005

1 General Comments

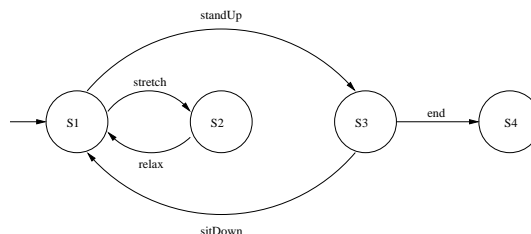
In the solution guide (dansk: vejledende løsninger) below one should remember that neither expressions nor LTSs are unique. So even though your solution does not match perfectly it can still be correct. These solutions are therefore intended primarily as a guide. Furthermore, I have tried to keep the expressions and LTSs as consistent and small as possible.

There is one section for each of the exercise slides followed by a solution (or a guide to a solution).

As with the exercises these solutions probably contain some errors, so if you discover one (or many) please contact me, so that I can correct the error ;-).

2 Exercises

- Write a CCS expression which denotes the following LTS



- It is possible to do the following traces:

- standUp, sitDown, stretch, relax, standup, end ?
- standUp, end, end ?
- standUp, sitDown, standUp, sitDown, ...

Answer to Exercise

As mentioned under General Comments there are several possible solutions to this exercise. I have chosen to represent each state with an equation. I use a Summation ($P1 + P2$) if a state have several outgoing transitions (see eg. S1).

$$S1 = \text{standUp}.S3 + \text{stretch}.S2$$

$$S2 = \text{relax}.S1$$

$$S3 = \text{end}.S4 + \text{sitDown}.S1$$

$$S4 = 0$$

Alternatively, we could have elided the state S4 and then just have written S3 as $S3 = \text{end}.0 + \text{sitDown}.S1$

- standUp, sitDown, stretch, relax, standup, end ?
This trace is possible, since we can from state S1 take the transition standUp to state S3, from here sitDown to state S1, ... until we end up in the state S4.
- standUp, end, end ?
This trace is *not* possible since after the transitions standUp followed by end we are in state S4 and from here there are no transitions with the label end.
- standUp, sitDown, standUp, sitDown, ... This trace is possible/impossible, depending on how one define the notion of trace. If one require that we end up in a terminal state, then this is not a trace, otherwise it is.

3 Exercises — Continued

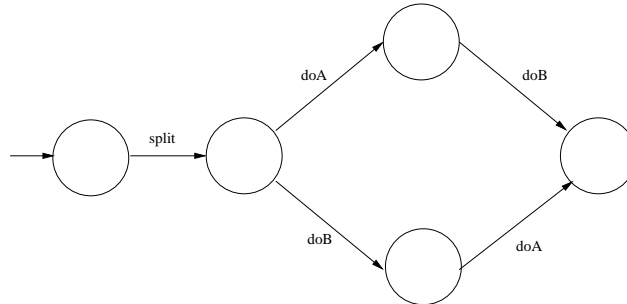
- Draw the labelled transition system which is the denotation of the following expression $\text{split}.(doA.0 \mid doB.0)$ (hint: it has 5 states)

Answer to Exercise

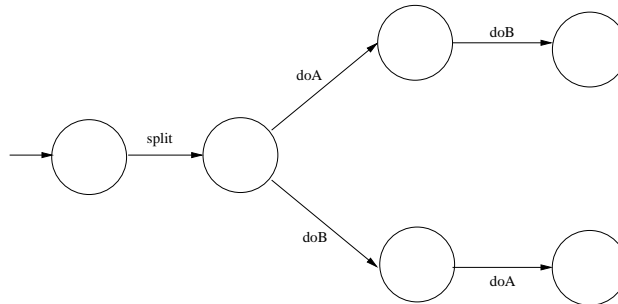
The following LTS is a possible solution to the exercise (the hint can be disregarded, since it is not all solutions which only have 5 states). From the expression we can see that the first transition must be split.

After this transition there are two possible transitions: doA or doB. If we for example take the transition doA then we end up in a state where the

only possible transition is doB and after this transition no other is possible (and vice versa).



As an alternative, we could have split the rightmost state into two states like the following LTS



And these solutions are actually “equivalent” (as will be presented later in the course).

4 Exercises — Continued

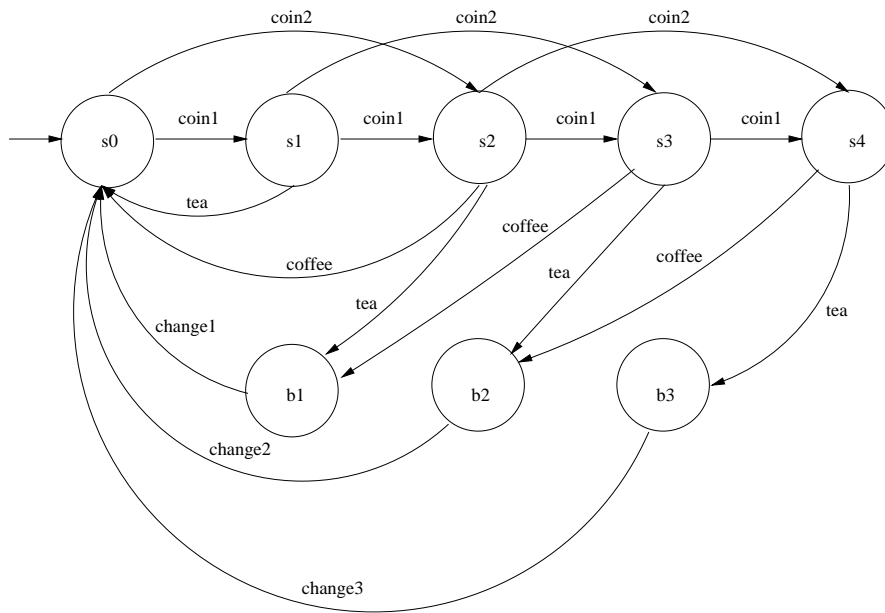
Beverage Machine

- Make first a LTS and then a CCS expression of the following beverage machine:
 - The machine can accept `coin1` (1 kr.) and `coin2` (2 kr.) up to a maximum of 4 kr.
 - When it has received enough money, it must be possible to get either `tea` (1 kr.) or `coffee` (2 kr), the machine must then give correct change back (`change1–change3`), and return to its initial state.
- How many states does the machine have ?
- Is it possible to do the following trace and end up in the initial state: `coin1, coin2, getTea, change2`

- Implement the CCS expression in MWB and try simulating different runs using `step`.

Answer to Exercise

In the following LTS the states s_0 - s_4 denotes that the person has paid 0-4 kr. to the beverage machine. The states b_1 - b_3 denotes that the machine is going to pay back 1-3 kr. in change. I have tried to keep the LTS minimal, meaning I have reused some of the states (eg. the states b_1 - b_3).



The corresponding CCS expression is

```

s0 = coin1.s1 + coin2.s2
s1 = coin1.s2 + coin2.s3 + tea.s0
s2 = coin1.s3 + coin2.s4 + tea.b1 + coffee.s0
s3 = coin1.s4 + tea.b2 + coffee.b1
s4 = tea.b3 + coffee.b2
b1 = change1.s0
b2 = change2.s0
b3 = change3.s0

```

With respect to the number of states of the machine it much depends on how minimal the LTS is. But mine LTS have 8 states.

It is possible to do the trace: `coin1, coin2, getTea, change2` and end up in the state s_0 .

5 Exercises — Continued

- Find all the possible initial transitions of the following process $trans1.0+(trans2.0|trans3.0)$ using the transition semantics
- Check the result in **MWB** with the command `transitions agent`, which prints out all transitions of *agent*
- Given that all the possible traces (to a terminal state) of a process is the following: x , y and x, z
- does this uniquely define the LTS (hint: non-determinism) ?
 - If yes, then draw the LTS
 - Otherwise, why not ? (give a counter-example with two LTSs)

Answer to Exercise

We are asked to find all the initial (or immediate) transitions using the transition semantics. From the structure of the syntax $trans1.0+(trans2.0|trans3.0)$ (made explicit by the parentheses) the last rule used must be SUM, since this is the only rule which have a conclusion, where the left-hand side contains a Summation of two terms (in the summation we let $I = \{0, 1\}$ and let $P_0 = trans1.0$ and $P_1 = (trans2.0|trans3.0)$).

So we use the SUM rule

$$\text{SUM} \frac{P_j \xrightarrow{\alpha} P'_j}{\sum_{i \in I} P_i \xrightarrow{\alpha} P'_j} \quad (j \in I)$$

where we instantiate the rule with the actual values. Since there are two choices for j (0 and 1) we have to try both choices. First we start with $j = 0$.

$$\text{SUM} \frac{trans1.0 \xrightarrow{\alpha} P'_j}{trans1.0 + (trans2.0|trans3.0) \xrightarrow{\alpha} P'_j}$$

Now we have to find a rule which has as conclusion $trans1.0 \xrightarrow{\alpha} P'_j$. There is only one rule with a left-hand side consisting of a prefix followed by a process; ACT.

$$\text{ACT} \frac{}{trans1.0 \xrightarrow{trans1} 0}$$

This gives us the following derivation (or derivation tree) of a transition. The rule ACT also tells us that $\alpha = trans1$ and $P'_j = 0$.

$$\text{SUM} \frac{\text{ACT} \frac{}{trans1.0 \xrightarrow{trans1} 0}}{trans1.0 + (trans2.0|trans3.0) \xrightarrow{trans1} 0}$$

So one possible transition is the following

$$trans1.0 + (trans2.0|trans3.0) \xrightarrow{trans1} 0$$

But this is not the only transition of the expression. Remember we chose $j = 0$ and j could either be 0 or 1. So now we try to infer transitions using $j = 1$. As mentioned before the only possible rule to use is SUM.

$$\text{SUM} \frac{trans2.0|trans3.0 \xrightarrow{\alpha} P'_j}{trans1.0 + (trans2.0|trans3.0) \xrightarrow{\alpha} P'_j}$$

We now have to use a rule which has as conclusion $trans2.0|trans3.0$. There are three such rules (where the left-hand side contains a parallel composition) COM1, COM2, and SYNC, but we only consider the first two, since we are sure that SYNC cannot be applied. First we try with the rule COM1 (taking $P = trans2.0$ and $Q = trans3.0$).

$$\text{COM1} \frac{trans2.0 \xrightarrow{\alpha} P'}{trans2.0|trans3.0 \xrightarrow{\alpha} P' | trans3.0}$$

Again, the only possible rule is ACT.

$$\text{ACT} \frac{}{trans2.0 \xrightarrow{trans2} 0}$$

which instantiates P' to 0 and α to $trans2$ (and we had that $P'_j = P' | trans3.0$). So the entire derivation looks like this.

$$\text{SUM} \frac{\text{COM1} \frac{\text{ACT} \frac{}{trans2.0 \xrightarrow{trans2} 0}}{trans2.0|trans3.0 \xrightarrow{trans2} 0 | trans3.0}}{trans1.0 + (trans2.0|trans3.0) \xrightarrow{trans2} 0 | trans3.0}$$

The last transition can be derived following the same pattern as the previous transition (we just the rule COM2 instead of COM1). We only present the derivation and leaves the reasoning to the reader.

$$\text{SUM} \frac{\text{COM2} \frac{\text{ACT} \frac{}{trans3.0 \xrightarrow{trans3} 0}}{trans2.0|trans3.0 \xrightarrow{trans3} trans2.0 | 0}}{trans1.0 + (trans2.0|trans3.0) \xrightarrow{trans3} trans2.0 | 0}$$

As can be checked in MWB this is exactly the transitions of the expression (with the correct remaining processes):

MWB>agent P(trans1,trans2,trans3) = trans1.0 +(trans2.0|trans3.0)

MWB>transitions P<trans1,trans2,trans3>

Commitments:

|>trans1.0

|>trans2.trans3.0

|>trans3.trans2.0

The traces do not uniquely define the LTS, there is at least 3 different LTS which allows for the same traces.

