Swansea University
Department of Computer Science

U Berger
November 16, 2022

CSM385/CSCM85 Modelling and Verification Techniques

Solutions and Feedback to the Coursework

Question 1

Consider two vending machines, VM_1 and VM_2 which accept a coin as payment and deliver tea. However, VM_1 requires payment upfront whereas VM_2 has the option of getting tea first and paying later. Here are the LTSs for these two machines

$$\{(\mathrm{VM}_1, \mathrm{coin}, C_1), (C_1, \mathrm{tea}, C_0), (C_1, \mathrm{coin}, C_2), (C_0, \mathrm{coin}, C_1), (C_2, \mathrm{tea}, C_1)\}$$

$$\{(\mathrm{VM}_2, \mathrm{coin}, P), (\mathrm{VM}_2, \mathrm{tea}, M), (P, \mathrm{tea}, \mathrm{VM}_2), (M, \mathrm{coin}, \mathrm{VM}_2)\}$$

(a) Draw both LTSs (you may use FDR but drawings by hand are fine as well).

(b) Give CSP definitions of both LTSs.

(c) Show that VM_1 and VM_2 are not trace equivalent.

(d) Find a state s in the LTS for VM_2 that is bisimilar to VM_1. Show bisimilarity by giving a bisimulation that contains the pair (VM_1, s).

[30 marks]

Solution.

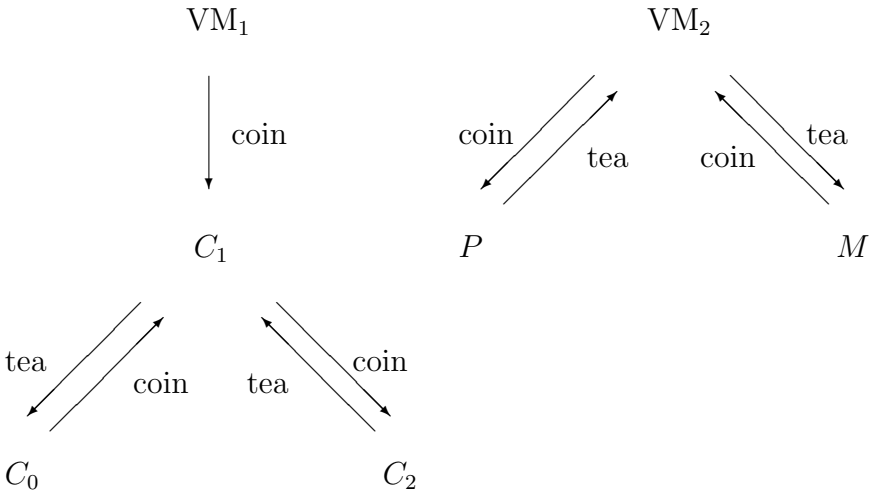
(a) The graphs are shown on the next page.

(b)

```
channel coin, tea

VM1 = coin -> C1
C1 = tea -> C0 [] coin -> C2
C0 = coin -> C1
C2 = tea -> C1

VM2 = coin -> P [] tea -> M
P = tea -> VM2
M = coin -> VM2
```



(c) VM_1 has the trace $\langle \mathrm{coin}, \mathrm{coin} \rangle$ which is not a trace of VM_2.

Alternative solution: VM_2 has the trace $\langle \mathrm{tea} \rangle$ which is not a trace of VM_1.

(d) We choose $s = M$. The relation

$$R = \{(\mathrm{VM}_1, M), (C_1, \mathrm{VM}_2), (C_0, M), (C_2, P)\}$$

is a bisimulation containing (VM_1, M).

To show that R is a bisimulation, we have to consider all pairs in R and demonstrate that every move of one side of the pair can be matched by the other side with a resulting pair that is again in R.

(VM_1, M)

$\mathrm{VM}_1 \stackrel{\mathrm{coin}}{\to} C_1$ is matched by $M \stackrel{\mathrm{coin}}{\to} \mathrm{VM}_2$ with $(C_1, \mathrm{VM}_2) \in R$.

$M \stackrel{\mathrm{coin}}{\to} \mathrm{VM}_2$ is matched by $\mathrm{VM}_1 \stackrel{\mathrm{coin}}{\to} C_1$ with $(C_1, \mathrm{VM}_2) \in R$.

(C_1, VM_2)

$C_1 \stackrel{\mathrm{tea}}{\to} C_0$ is matched by $\mathrm{VM}_2 \stackrel{\mathrm{tea}}{\to} M$ with $(C_1, \mathrm{VM}_2) \in R$.

$C_1 \stackrel{\mathrm{coin}}{\to} C_2$ is matched by $\mathrm{VM}_2 \stackrel{\mathrm{coin}}{\to} P$ with $(C_2, P) \in R$.

$\mathrm{VM}_2 \stackrel{\mathrm{coin}}{\to} P$ is matched by $C_1 \stackrel{\mathrm{coin}}{\to} C_2$ with $(C_2, P) \in R$.

$\mathrm{VM}_2 \stackrel{\mathrm{tea}}{\to} M$ is matched by $C_1 \stackrel{\mathrm{tea}}{\to} C_0$ with $(C_0, M) \in R$.

(C_0, M)

$C_0 \stackrel{\text{coin}}{\to} C_1$ is matched by $M \stackrel{\text{coin}}{\to} \text{VM}_2$ with $(C_1, \text{VM}_2) \in R$.

$M \stackrel{\text{coin}}{\to} \text{VM}_2$ is matched by $C_0 \stackrel{\text{coin}}{\to} C_1$ with $(C_1, \text{VM}_2) \in R$.

(C_2, P)

$C_2 \stackrel{\text{tea}}{\to} C_1$ is matched by $P \stackrel{\text{tea}}{\to} \text{VM}_2$ with $(C_1, \text{VM}_2) \in R$.

$P \stackrel{\text{tea}}{\to} \text{VM}_2$ is matched by $C_2 \stackrel{\text{tea}}{\to} C_1$ with $(C_1, \text{VM}_2) \in R$.

Note for marking: Roughly, (a)-(c) are worth 6 marks each, and (d) is worth 12 marks. In (d) it suffices to give the relation R.

Question 2 Decide for each of the following statements whether it is true for all processes A, B:

(a) If A and B are trace equivalent and A is deadlock free, then B is deadlock free.

(b) If A and B are bisimilar and A is deadlock free, then B is deadlock free.

In each case, either prove the statement, or give a counterexample.

You may use the following definition of a deadlock:

A process has S has a deadlock if there are a word w and a process S' such that $S \stackrel{w}{\to}^* S'$ and there is no transition from S'.

Hence, for example, part (b) can be equivalently reformulated as

(b) If A and B are bisimilar and B has a deadlock, then A has a deadlock.

[40 marks]

Solution.

(a) is false. For example, the following processes A and B are trace equivalent, however A is deadlock free, while B does have a deadlock:

```
channel e

A = e -> A
B = e -> B [] e -> STOP
```

(b) is true.

We show that for every word w: If A and B are bisimilar and $B \overset{w}{\to}^* B'$ is a deadlock, for some state B' (that is, there is no transition from B'), then there is a deadlock $A \overset{w}{\to}^* A'$ for some state A' (with the same word w).

We show this by induction on the length of w.

Induction Base: $w = \langle\rangle$.

Let A and B be bisimilar and assume $B \overset{w}{\to}^* B'$. Since w is the empty word, this mean that $B = B'$, that is B itself *is* a deadlock, that is, B has no transition. But then A must be a deadlock as well, since any transition from A would have be matched by a transition from B.

Induction Step: w is not the empty word.

Assume A and B are bisimilar and B has a deadlock $B \overset{w}{\to}^* B'$ for some state B'. Since w is not empty, $w = \langle a \rangle \frown w'$ for some event a and word w'. Therefore, $B \overset{a}{\to} B_1$ and $B_1 \overset{w'}{\to}^* B'$. Since A and B are bisimilar, this can be matched by a transition $A \overset{a}{\to} A_1$ such that A_1 and B_1 are again bisimilar.

Since w' is by one letter shorter than w, and $B_1 \overset{w'}{\to}^* B'$ is a deadlock, we can apply the induction hypothesis and conclude that A_1 has a deadlock $A_1 \overset{w'}{\to}^* A'$ for some state A'. But then A has the deadlock $A \overset{w}{\to}^* A'$, since $A \overset{a}{\to} A_1 \overset{w'}{\to}^* A'$.

Notes for marking: 20 marks are allocated to each part. In part (b), the proof needn't be as detailed as above.

Question 3

Consider the following definition of a robot that repeatedly reports its positions, then moves to the left or right (within a given finite range), or does some work.

```
min = 0
max = 5
Range = {min..max}

datatype Direction = L | R
```

```
channel move : Direction
channel position : Range
channel work

Robot(x) = position.x ->
           ( (if x > min then move.L -> Robot(x-1) else STOP) []
             (if x < max then move.R -> Robot(x+1) else STOP) []
             (work -> Robot(x)) )
```

Suppose doing work empties the robot's battery so that it needs at least two movements to recharge the battery (for example using a solar panel). We assume that, initially, the robot's battery is empty.

Define a process `Empty` and a suitable synchronisation set `X` such that the process

```
Robot(0) [| X |] Empty
```

models this behaviour.

It will be convenient to define `Empty` simultaneously with a process `Full` that represents the fully charged battery.

[30 marks]

Solution.

```
Empty = move?x -> move?y -> Full
Full = work -> Empty [] move?x -> Full

System = Robot(0) [| {| move, work |} |] Empty
```

An alternative solution is

```
Battery(n) = if n < 2
             then move?x -> Battery(n+1)
             else work -> Battery(0) [] move?x -> Battery(2)

System = Robot(0) [| {| move, work |} |] Battery(0)
```

or similar.

Notes for marking: If Full is defined as `Full = work -> Empty` (that is, the alternative `move?x -> Full` is missing), 5 marks should be taken off.

Feedback

Overall, I am impressed by the amount of work you invested in this coursework. This paid off by many very good solutions and a high average mark.

There was an big variation in the quality of solutions. This indicates that there are a some key points of understanding which some students mastered very well while others are struggling. This specifically concerns the notions of trace equivalence and bisimilarity the meanings of which do not seem to be well understood by many of you (see in particular Questions 1 (d) and 2 (b)).

A general mistake was that some solutions didn't answer the question at hand, but answered something else. To prevent similar mistakes in the exam, my advice is to carefully read what a question asks and then answer it to the point (see in particular Questions 1 (c) and (d)).

Question 1

Parts (a) and (b) were answered flawlessly.

Part (c) asks for an argument why the processes VM_1 and VM_2 do not have the same set of traces. The question does *not* ask for a description of these set of traces. Since both sets are infinite, it is difficult to write them down precisely. Something like

$$\begin{array}{lcl} \mathrm{Traces}(\mathrm{VM}_1) & = & \{\langle\rangle, \langle \mathrm{coin}\rangle, \langle \mathrm{coin}, \mathrm{tea}\rangle, \langle \mathrm{coin}, \mathrm{coin}\rangle, \ldots\} \\ \mathrm{Traces}(\mathrm{VM}_2) & = & \{\langle\rangle, \langle \mathrm{coin}\rangle, \langle \mathrm{coin}, \mathrm{tea}\rangle, \langle \mathrm{tea}\rangle, \ldots\} \end{array}$$

is not suitable for a proof, since one does not know what is hidden in the ellipses, ... Also, many of such descriptions were incorrect, for example, the sets were described as if they contained only finitely many words.

A much stronger, and simpler, argument is to point out that a specific word is a trace of one process but not of the other.

Many answers provided such, stronger, arguments, but some consisted in just writing vague descriptions of the two trace sets. If these description were sufficiently informative, still full marks were awarded.

In part (d), the question consisted of (1) to find a state that is bisimilar to VM_1, (2) to prove bisimilarity by defining a bisimulation that contains the pair of these two states.

Some solutions answered (1) and (2) correctly, but many answers did not say what the state s, one is looking for, is, or they proved bisimilarity in a different way, for example by arguing that the LTSs are deterministic and therefore trace equivalence suffices. Other answers used the game characterisation of bisimilarity. In these cases, only partial marks could be awarded, since (2) specifically asks for the *bisimulation*. Moreover, many proofs that used the game characterisation were incorrect or incomplete.

In the model solution, a full proof that the relation R is indeed a bisimulation is given, to demonstrate how such a proof can look like. The proof was not required to receive full marks for part (d).

Question 2

General comment: Both parts of this question ask (1) for a decision whether the statement is true or false, (2) for a justification of the decision. A common mistake was to omit (1).

Part (a) was answered well, in general. Some counterexamples were not correct, though.

Part (b) was answered correctly only in a few cases. The main mistakes were:

- Some solutions contain statements such as "proof by induction on w" without saying *what* is proven.
- Frequently, it was argued that bisimilar states "behave in the same way", and therefore if one is deadlock free, the other must be deadlock free as well. This is not convincing since the question what it means to "behave in the same way" is not answered. Remember that it took top scientist in the world decades (with many incorrect attempts) to discover the notion of bisimilarity which answers this question satisfactorily. Therefore, it is important to refer to the exact definition of bisimilarity.

Question 3

The solutions were generally very good. The main purpose of this question was to test your understanding of the controlled interaction of two processes through the generalised parallel operator. I am pleased that you understood this concept.