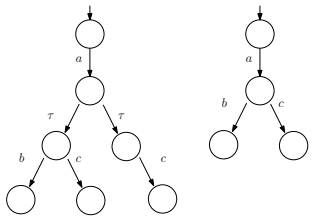
System Validation (IN4387)
November 2, 2012, 14:00-17:00

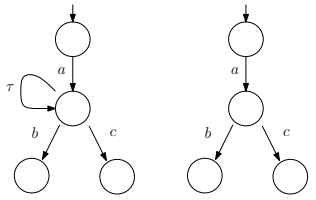
Important Notes. The examination comprises 5 question in 4 pages. Give complete explanation and do not confine yourself to giving the final answer. **Good luck!**

Exercise 1 (20 points) In each of the following item determine whether the specified notion of equivalence holds between the two given labeled transition systems. For each and every item provide a complete line of reasoning why a certain equivalence does or does not hold:

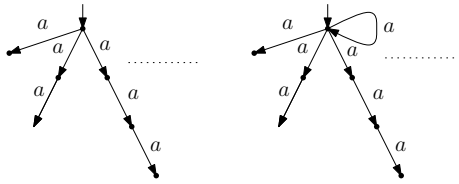
1. Strong bisimilarity:



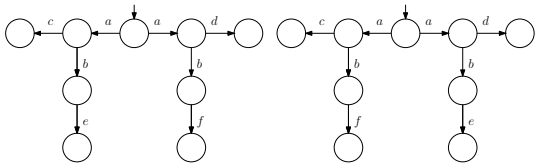
2. Branching bisimilarity:



3. Strong bisimilarity:



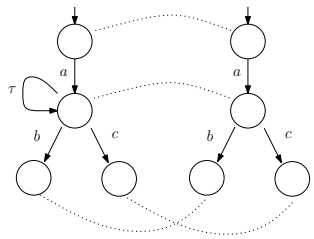
4. Branching bisimilarity:



Answer 1

1. No. They are not bisimilar. Assume that they were bimisimilar, then there would exist a bisimulation relation which relates the initial states. Then, the first 'a' transition on the left-hand-side LTS can only be mimicked by the only initial 'a' transition on the righ-hand side. Hence, the target of the two 'a' transition have to be related in the same relation. However, this cannot be the case since the state in the left-hand-side can do a τ transition, while the right-hand-side one cannot mimic it. Note that strong bisimilarity does not ignore τ transitions.

2. Yes, they are. A branching bisimulation relation relating the initial states is given below:



3. No, they are not bisimilar. Assume that they were bisimilar, then there would exist a bisimulation relation which relates the initial states. The 'a' loop in the initial state of the right-hand-side (rhs) LTS should be mimicked by an 'a' transition in the left-hand-side (lhs) one. Assume that the latter 'a' transition is the start of a trace of size n (for some arbitrary n); then it follows that the initial state of the rhs LTS should be bisimilar to the second state of this trace. The initial state of the rhs LTS can do an 'a'-loop, and this can only be mimicked by the second state in the trace of size n by performing an 'a' transition into the third state in the trace. Hence, the initial state of the rhs LTS should be related to the third state in the trace. Repeating this exercise on the second to the n-th state in the trace, will lead to the conclusion that the last state in the trace of size n, should be bisimilar to the initial state of the rhs LTS, which is clearly not true, because the last state of the trace deadlocks, while the initial state of the rhs LTS can still perform 'a' transitions.

4. No, they are not bisimilar. Assume that they were bisimilar, then there would exist a bisimulation relation which relates the initial states. The initial state of the lhs LTS can make an 'a' transition to the left. This can be mimicked by the initial state of the rhs LTS:

   - Either the initial state of the rhs LTS makes the 'a' transition to the left, then the states to the left of the initial states in the lhs and rhs LTSs should be related. Consider for example the latter state in the lhs LTS; it can perform a 'b' transition; this can be mimicked in the corresponding state in the rhs LTS by performing the only enabled 'b' transition. Hence, the targets of the two 'b' transitions should be related by the same bisimulation relation. However, the latter states cannot be bisimilar because the lhs state can perform an 'e' transition, which cannot be mimicked by the rhs state and likewise, the rhs can perform an 'f' transition, which cannot be mimicked by the lhs state.
   - Or the initial state of the rhs LTS makes the 'a' transition to the right, then the state to the left of the initial state of the lhs LTS should be related to the state to the right of the initial state of the rhs LTS. This cannot be true however, since the former state can perform a 'c' transition, which cannot be mimicked by the latter state (and likewise, the latter state performs a 'd' transition, which cannot be performed by the former state).

Exercise 2 (20 points) Consider the following two modal formulae:

$$[request]\langle true^* response\rangle true$$

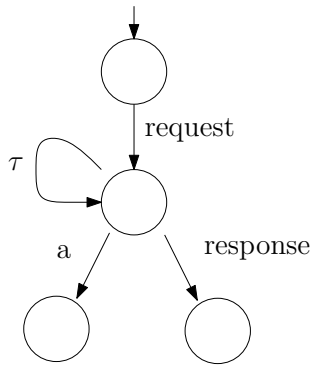
and

$$[request](\mu X.\langle true\rangle true \wedge [\overline{response}]X)$$

1. Explain in words what each of the two formulae means. **(10 points)**
2. Give a labeled transition system in which one of the two formulae holds and the other one doe not hold. (It does not matter which one you choose to hold.) **(10 points)**

Answer 2

1. The first formula states that after each 'request', there is at least one path leading to a 'response'. (There may be other paths not leading to response.) The second formula states that after each 'request', each path will eventually reach a 'response' action. (No path can avoid doing a response.)
2. The following LTS satisfies the first formula but not the second, since it can avoid the response by taking the τ transition infinitely many times. It also does not satisfy the second formula, because it has a path (starting with 'a') which can avoid the response altogether.



Exercise 3 (20 points) Define a sort (data type) ToDoList, where each element of this sort is either the empty list, or a non-empty list of prioritized tasks. A prioritized task is a pair (i, t) where i is a positive natural number determining the priority and t is an element of a sort $Task$, which contains a constant (constructor) $noTask$ and is not specified any further.

- Give the formal definition of $ToDoList$ and its constructors. **(5 points)**
- Define a function (map) $toDoNow$, which taks a $ToDoList$ as its parameter, and returns the task with the highest priority in the list, if it is non-empty, or $noTask$, otherwise. If needed, you may define and use other auxiliary functions used in the definition of $toDoNow$. **(15 points)**

Answer 3

-

```
sort   Task ;
sort   toDoList = List(Nat#Task);
cons   noTask: Task;
```

-

```
map   minPr: toDoList → Nat;
      toDoNow: toDoList → toDoList;
var   i, j: Nat;
      t : Task;
      l : toDoList;
eqn   minPr([])= 0;                                                    (1)
      minPr((i,t) ▷ l ) = min(i, minPr(l));
      toDoNow((i,t) ▷ l ) = if (minPr(l) >= i, t, toDoNow(l)) ;
      toDoNow( [] ) = noTask ;
```

Exercise 4 (20 points) Prove the following equations using the axioms provided in the appendix. Mention the name of the axiom used for each and every step.

1. $(a(1) \mid b(2)) \setminus (c(2) \mid b(3)) = a(1) \mid b(2)$ **(5 points)**,
2. $(a + b) \cdot c \parallel \delta = a \cdot c \cdot \delta + b \cdot c \cdot \delta$ **(5 points)**, and
3. $(c \wedge d) \rightarrow a \subseteq c \rightarrow a$ (Hint $x \subseteq y$ if and only if $x + y = y$) **(10 points)**.

Answer 4

1.

$$
\begin{array}{lcl}
(a(1) \mid b(2)) \setminus (c(2) \mid b(3)) & = & \text{(MD3)} \\
((a(1) \mid b(2)) \setminus c(2)) \setminus b(3) & = & \text{(MD5)} \\
(a(1) \mid (b(2) \setminus c(2))) \setminus b(3) & = & \text{(MA3)} \\
(a(1) \mid ((b(2) \mid \tau) \setminus c(2))) \setminus b(3) & = & \text{(MA3)} \\
(a(1) \mid (b(2) \mid (\tau \setminus c(2)))) \setminus b(3) & = & \text{(MD1)} \\
(a(1) \mid (b(2) \mid \tau)) \setminus b(3) & = & \text{(MA3)} \\
(a(1) \mid b(2)) \setminus b(3) & = & \text{(MD5)} \\
a(1) \mid (b(2) \setminus b(3)) & = & \text{(MA3)} \\
a(1) \mid ((b(2) \mid \tau) \setminus b(3)) & = & \text{(MD5)} \\
a(1) \mid (b(2)) \mid (\tau \setminus b(3))) & = & \text{(MD1)} \\
a(1) \mid (b(2)) \mid \tau) & = & \text{(MA3)} \\
a(1) \mid b(2) &&
\end{array}
$$

2.

$$
\begin{array}{lcr}
(a + b) \cdot c \parallel \delta & = & \text{(M)} \\
((a + b) \cdot c \between \delta) + (\delta \between (a + b) \cdot c) + ((a + b) \cdot c | \delta) & = & \text{(A4)} \\
((a \cdot c + b \cdot c) \between \delta) + (\delta \between (a + b) \cdot c) + ((a + b) \cdot c | \delta) & = & \text{(LM4)} \\
(a \cdot c \between \delta) + (b \cdot c \between \delta) + (\delta \between (a + b) \cdot c) + ((a + b) \cdot c | \delta) & = & \text{(LM2)} \times 2 \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + (\delta \between (a + b) \cdot c) + ((a + b) \cdot c | \delta) & = & \text{(LM2)} \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + \delta + ((a + b) \cdot c | \delta) & = & \text{(LM2)} \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + ((a + b) \cdot c | \delta) & = & \text{(LM2)} \times 2 \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + ((a \cdot c + b \cdot c) | \delta) & = & \text{(LM2)} \times 2 \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + (a \cdot c) | \delta + (b \cdot c) | \delta & = & \text{(LM2)} \times 2 \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + (a \mid \delta) \cdot c + (b \mid \delta) \cdot c & = & \text{(LM2)} \times 2 \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + \delta \cdot c + \delta \cdot c & = & \text{(LM2)} \times 2 \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) + \delta + \delta & = & \text{(LM2)} \times 2 \\
a \cdot (c \parallel \delta) + b \cdot (c \parallel \delta) & = & \text{(M)} \times 2 \\
a \cdot (c \between \delta + \delta \between c + c | \delta) + b \cdot (c \between \delta + \delta \between c + c | \delta) & = & \text{(M)} \times 2 \\
a \cdot (c \cdot \delta + \delta \between c + c | \delta) + b \cdot (c \cdot \delta + \delta \between c + c | \delta) & = & \text{(LM1)} \times 2 \\
a \cdot (c \cdot \delta + \delta \between c + c | \delta) + b \cdot (c \cdot \delta + \delta \between c + c | \delta) & = & \text{(LM2)} \times 2 \\
a \cdot (c \cdot \delta + \delta + c | \delta) + b \cdot (c \cdot \delta + \delta + c | \delta) & = & \text{(A6)} \times 2 \\
a \cdot (c \cdot \delta + c | \delta) + b \cdot (c \cdot \delta + c | \delta) & = & \text{(S4)} \times 2 \\
a \cdot (c \cdot \delta + \delta) + b \cdot (c \cdot \delta + \delta) & = & \text{(A6)} \times 2 \\
a \cdot (c \cdot \delta) + b \cdot (c \cdot \delta) & = & \text{(A6)} \times 2
\end{array}
$$

3. $(c \wedge d) \rightarrow a \subseteq c \rightarrow a$, which means that we have to prove $c \rightarrow a = c \rightarrow a + (c \wedge d) \rightarrow a$. We do this by induction (case analysis) on the boolean variable d:

- Either $d = true$, then we have:

$$\begin{array}{lcr} c{\rightarrow}a + (c \wedge d){\rightarrow}a & = & d = true \\ c{\rightarrow}a + (c \wedge true){\rightarrow}a & = & \text{logic} \\ c{\rightarrow}a + c{\rightarrow}a & = & \text{(A3)} \\ c{\rightarrow}a + c{\rightarrow}a & = & \text{(A3)} \\ c{\rightarrow}a & & \end{array}$$

- Either $d = false$, then we have:

$$\begin{array}{lcr} c{\rightarrow}a + (c \wedge false){\rightarrow}a & = & d = false \\ c{\rightarrow}a + (c \wedge false){\rightarrow}a & = & \text{logic} \\ c{\rightarrow}a + false{\rightarrow}a & = & \text{(cond2)} \\ c{\rightarrow}a + \delta & = & \text{(A3)} \\ c{\rightarrow}a & & \end{array}$$

Exercise 5 (20 points) Specify the following system of two parallel processes:

The first process represents a temperature sensor, which can issue two types of actions: $snd\_temp(n)$ and snd_defect. The sensor can send any natural number between 0 and 200 as the parameter of *snd_temp* and may non-deterministically choose to send the *snd_defect* signal, after which it deadlocks.

The second process represents a thermostat, which receives temperature from the sensor and if the received value is in the range 0 and 50 it issues action *on* to the outside world; if the value is between 50 and 100 it sends action *off* to the outside world; if the received value is outside these ranges it ignores the value, but keeps on listening to the sensor at any case. Upon synchronizing with *snd_defect*, the thermostat will issue an *alarm* action and terminate.

The action names that are not specified in the above-given description can be chosen freely.

Answer 5

```
act   snd_temp, rcv_temp, sync_temp : Nat;
      snd_defect, rcv_defect, sync_defect, , on, off ;

proc  Sensor      =
                      snd_defect · delta +
                      sum n : Nat . (n ≤ 200) → snd_temp(n) · Sensor ;
      Thermostat  =
                      rcv_defect · alarm +
                      sum n : Nat . rcv_temp(n) .
                         (n ≤ 50) → on . Thermostat
                            ◊ ( (n ≤ 100) → off . Thermostat
                                  ◊ Thermostat ) ;
init                  allow ( { sync_temp, sync_defect, on, off, alarm },
                        ( comm { snd_temp | rcv_temp → sync_temp, snd_defect | rcv_defect → sync_defect },
                            Sensor || Thermostat )) ;
```

MA1	$\alpha\|\beta = \beta\|\alpha$
MA2	$(\alpha\|\beta)\|\gamma = \alpha\|(\beta\|\gamma)$
MA3	$\alpha\|\tau = \alpha$
MD1	$\tau \setminus \alpha = \tau$
MD2	$\alpha \setminus \tau = \alpha$
MD3	$\alpha \setminus (\beta\|\gamma) = (\alpha \setminus \beta) \setminus \gamma$
MD4	$(a(d)\|\alpha) \setminus a(d) = \alpha$
MD5	$(a(d)\|\alpha) \setminus b(e) = a(d)\|(\alpha \setminus b(e))$ if $a \not\equiv b$ or $d \not\approx e$
MS1	$\tau \sqsubseteq \alpha = \mathit{true}$
MS2	$a \sqsubseteq \tau = \mathit{false}$
MS3	$a(d)\|\alpha \sqsubseteq a(d)\|\beta = \alpha \sqsubseteq \beta$
MS4	$a(d)\|\alpha \sqsubseteq b(e)\|\beta = a(d)\|(\alpha \setminus b(e)) \sqsubseteq \beta$ if $a \not\equiv b$ or $d \not\approx e$
MAN1	$\underline{\tau} = \tau$
MAN2	$\underline{a(d)} = a$
MAN3	$\underline{\alpha\|\beta} = \underline{\alpha}\|\underline{\beta}$

Table 1: Axioms for multi-actions

Note that $a(d)$ and $b(e)$ range over (parameterized) actions, α and β range over (multi)actions and x, y and z range over processes.

A1	$x + y = y + x$
A2	$x + (y + z) = (x + y) + z$
A3	$x + x = x$
A4	$(x + y)\cdot z = x\cdot z + y\cdot z$
A5	$(x\cdot y)\cdot z = x\cdot(y\cdot z)$
A6	$x + \delta = x$
A7	$\delta\cdot x = \delta$
Cond1	$\mathit{true}\rightarrow x \diamond y = x$
Cond2	$\mathit{false}\rightarrow x \diamond y = y$
SUM1	$\sum_{d:D} x = x$
SUM3	$\sum_{d:D} X(d) = X(e) + \sum_{d:D} X(d)$
SUM4	$\sum_{d:D}(X(d) + Y(d)) = \sum_{d:D} X(d) + \sum_{d:D} Y(d)$
SUM5	$(\sum_{d:D} X(d))\cdot y = \sum_{d:D} X(d)\cdot y$

Table 2: Axioms for the basic operators

M	$x \parallel y = x \between y + y \between x + x\vert y$
LM1	$\alpha \between x = \alpha\cdot x$
LM2	$\delta \between x = \delta$
LM3	$\alpha\cdot x \between y = \alpha\cdot(x \parallel y)$
LM4	$(x + y) \between z = x \between z + y \between z$
LM5	$(\sum_{d:D} X(d)) \between y = \sum_{d:D} X(d) \between y$
S1	$x\vert y = y\vert x$
S2	$(x\vert y)\vert z = x\vert(y\vert z)$
S3	$x\vert\tau = x$
S4	$\alpha\vert\delta = \delta$
S5	$(\alpha\cdot x)\vert\beta = \alpha\vert\beta\cdot x$
S6	$(\alpha\cdot x)\vert(\beta\cdot y) = \alpha\vert\beta\cdot(x \parallel y)$
S7	$(x + y)\vert z = x\vert z + y\vert z$
S8	$(\sum_{d:D} X(d))\vert y = \sum_{d:D} X(d)\vert y$
TC1	$(x \between y) \between z = x \between (y \parallel z)$
TC2	$x \between \delta = x\cdot\delta$
TC3	$(x\vert y) \between z = x\vert(y \between z)$

Table 3: Axioms for the parallel composition operators