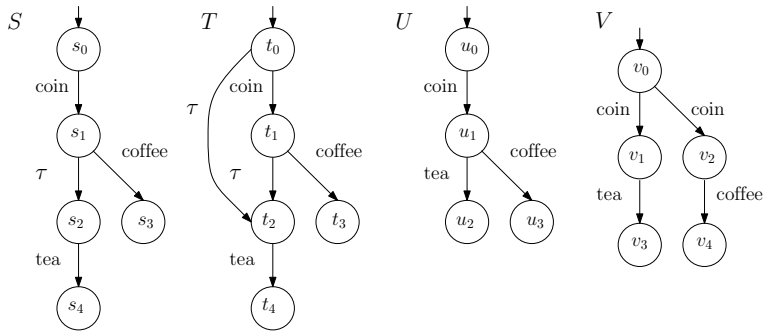
System Validation (IN4387) Resit Examination
February 1, 2012, 14:00-17:00

Important Notes. It is not allowed to use study material, computers, or calculators during the examination. The examination comprises 5 question and 4 pages. Please check beforehand whether your copy is properly printed. Give complete explanation and do not confine yourself to giving the final answer. The answers may be given in Dutch or in English. **Good luck!**

Exercise 1 (20 points) Consider the following labeled transitions systems and reason whether and why each of the following equalities hold.

1. S and T are strongly bisimilar,
2. S and T are branching bisimilar,
3. T and U are branching bisimilar,
4. U and V are language equivalent.
5. U and V are strongly bisimilar.



Exercise 2 (20 points) Assume that the sort $iNatural$ of natural numbers is defined as follows:

`sort`	iNatural;	
`cons`	zero: iNatural;	
	succ: iNatural → iNatural;	
`map`	plus: iNatural × iNatural → iNatural ;	
`var`	i, j: iNatural;	
`eqn`	plus(zero, i)= i;	(1)
	plus(succ(i), j)= succ(plus(i, j));	(2)

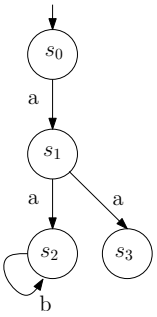
1. Prove that plus(i, zero) = i.
2. Prove that plus(i, succ(j)) = succ(plus(i, j)).
3. Prove that plus is commutative, i.e., plus(i, j) = plus(j, i).

Exercise 3 (20 points) Prove the following equations using the axioms provided in the appendix.

1. $(\alpha \mid \beta) \setminus (\beta \mid \gamma) = \alpha$
2. $x + y + ((c \vee d) \to x \diamond y) = y + x$,
3. $a.\delta \parallel (b + c) = a.(b + c).\delta + (b + c).a.\delta + (a \mid (b + c)).\delta$

Note that sequential composition binds stronger than nondeterministic choice.

Exercise 4 (20 points) Consider the following LTS.



In which states the formula $[a]\mu X.[b]X$ holds? Explain the steps towards the final answer.

Exercise 5 (20 points) Specify a track controller in mCRL2, with the following informal specification. The controller is supposed to control the entrance to a track which can allow for at most one train at a time. Thus, the trains, identified by a unique natural number, announce their arrival with an "arrive(i)" action, where i is the identifier of the train. If the track is not occupied, the train will be allowed to the track using the action "allow(i)". If the track is already occupied, the identifier of the train will be recorded in the list of waiting trains. Upon the departure of a train from the track, denoted by the action "depart", the first train in the waiting list, i.e., the one who has waited most, will be allowed into the track.

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 $d \not\approx e$

Table 1: Axioms for multi-actions

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	$\mathit{true} \to x \diamond y = x$
Cond2	$\mathit{false} \to x \diamond y = y$
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

Note that α and β range over (multi)actions and x, y and z ranger over processes.

Answer 1

1. S and T are not strongly bisimilar; assume, towards a contradiction, that they are bisimilar. Then, there exists a bisimulation relation relating s_0 and t_0. However, t_0 can perform a τ, which cannot be mimicked by s_0.

2. S and T are not branching bisimilar; assume, towards a contradiction, that they are bisimilar. Then, there exists a branching bisimulation relation relating s_0 and t_0. However, t_0 can perform a τ, which cannot be mimicked by s_0.

3. S and T are not branching bisimilar, for the same reason as above.

4. U and V are language equivalent, because their sets of traces coincide and their sets of languages coincide.

5. S and T are not strongly bisimilar; assume, towards a contradiction, that they are bisimilar. Then, there exists a bisimulation relation relating u_0 and v_0. State v_0 can perform a coin transition to v_1; this transition can only be mimicked by u_0 by moving to u_1. Hence u_1 and v_1 should be related by the same bisimulation relation. However, u_1 can perform a coffee transition, which cannot be mimicked by v_1.

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

Table 3: Axioms for the parallel composition operators

Answer 2

1. By induction on the natural number n, we show that plus(n, zero) = n.

   For the base case, i.e., $n = zero$, we have.

plus(n, zero)	=	(assumption)
plus(zero, zero)	=	(1)
zero	=	(assumption)
n		

   For the induction step, assuming that $plus(n', zero) = n'$ (induction hypothesis), consider the case $n = \text{succ}(n')$

plus(n, zero)	=	(assumption)
plus(succ(n'), zero)	=	(2)
succ(plus(n', zero))	=	(induction hypothesis)
succ(n')	=	(assumption)
n		

2. By induction on the natural number n, we show that plus(n, succ(j)) = succ(plus(n,j)).

   For the base case, i.e., $n = zero$, we have.

plus(n, succ(j))	=	(assumption)
plus(zero, succ(j))	=	(1)
succ(j)	=	(1)
succ(plus(zero,j))	=	(assumption)
succ(plus(n,j))		

For the induction step, assuming that plus(n', succ(j)) = succ(plus(n',j)) (induction hypothesis), consider the case n = succ(n')

plus(n, succ(j))	=	(assumption)
plus(succ(n'), succ(j))	=	(2)
succ(plus(n', succ(j)))	=	(induction hypothesis)
succ(succ(plus(n', j)))	=	(2)
succ(plus(succ(n'), j))	=	(assumption)
succ(plus(n, j))		

3.

By induction on the natural number n, we show that plus(n, j) = plus(j, n).

We use the above-proven two items (Ex. 2.1 and 2.2) in the proof of this item.

For the base case, i.e., $n = zero$, we have.

plus(n, j)	=	(assumption)
plus(zero, j)	=	(1)
j	=	Ex. 2.1
plus(zero,j)	=	(assumption)
plus(n,j)		

For the induction step, assuming that plus(n', j) = plus(j, n') (induction hypothesis), consider the case n = succ(n')

plus(n, j)	=	(assumption)
plus(succ(n'), j)	=	(2)
succ(plus(n', j))	=	(induction hypothesis)
succ(plus(j, n'))	=	Ex. 2.2
plus(j, succ(n'))	=	(assumption)
plus(n, j)		

Answer 3

1.
$$\begin{array}{lcr}
(\alpha \mid \beta) \setminus (\beta \mid \gamma) & = & \text{MD3} \\
((\alpha \mid \beta) \setminus \beta) \setminus \gamma & = & \text{MA1, MD4} \\
\alpha \setminus \gamma & = & \text{MA3} \\
(\alpha \mid \tau) \setminus \gamma & = & \text{MD5} \\
\alpha \mid (\tau \setminus \gamma) & = & \text{MD1} \\
\alpha \mid \tau & = & \text{MA3} \\
\alpha &&
\end{array}$$

2. By induction (case distinction) on $c \vee d$; we only given the answer for the case $c \vee d = \mathit{true}$ below and the other case is similar.
$$\begin{array}{lcr}
x + y + ((c \vee d) \to x \diamond y) & = & \text{assumption} \\
x + y + (\mathit{true} \to x \diamond y) & = & \text{Cond1} \\
x + y + x & = & \text{A1, A2, A3} \\
x + y &&
\end{array}$$

3. In the following derivations, $\between$ and $\mid$ both bind more strongly than $+$ and less strongly than $\cdot$.
$$\begin{array}{lcr}
a.\delta \parallel (b + c) & = & \text{(M)} \\
(a.\delta \between (b + c)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(LM3)} \\
a.(\delta \parallel (b + c)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(M)} \\
a.((\delta \between (b + c)) + ((b + c) \between \delta) + ((b + c) \mid \delta)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(LM2)} \\
a.(\delta + ((b + c) \between \delta) + ((b + c) \mid \delta)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(A1, A6)} \\
a.(((b + c) \between \delta) + ((b + c) \mid \delta)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(LM4)} \\
a.((b \between \delta) + (c \between \delta) + ((b + c) \mid \delta)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(LM2)} \times 2 \\
a.(b.\delta + c.\delta + ((b + c) \mid \delta)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(A4)} \\
a.((b + c).\delta + ((b + c) \mid \delta)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(S7)} \times 2 \\
a.((b + c).\delta + (b \mid \delta) + (c \mid \delta)) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(S4)} \times 2 \\
a.((b + c).\delta + \delta + \delta) + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(A6)} \times 2 \\
a.(b + c).\delta + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(A6)} \times 2 \\
a.(b + c).\delta + ((b + c) \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(LM4)} \\
a.(b + c).\delta + (b \between a.\delta) + (c \between a.\delta) + (a.\delta \mid (b + c)) & = & \text{(LM2)} \times 2 \\
a.(b + c).\delta + (b.a.\delta) + (c.a.\delta) + (a.\delta \mid (b + c)) & = & \text{(A4)} \\
a.(b + c).\delta + (b + c).a.\delta + (a.\delta \mid (b + c)) & = & \text{(S1, S7)} \\
a.(b + c).\delta + (b + c).a.\delta + (b \mid a.\delta) + (b \mid a.\delta) & = & \text{(S1, S5)} \times 2 \\
a.(b + c).\delta + (b + c).a.\delta + (b \mid a).\delta + (c \mid a).\delta & = & \text{(A4)} \\
a.(b + c).\delta + (b + c).a.\delta + ((b \mid a) + (c \mid a)).\delta & = & \text{(S1,S7)} \\
a.(b + c).\delta + (b + c).a.\delta + (a \mid (b + c)).\delta &&
\end{array}$$

Answer 4 We start with the smallest subformula, namely $\mu X.[b]X$ and calculate the set of states in which this subformula holds.

Since this is a minimal fixed point, we start with replacing X by the empty set of states in $[b]X$ to obtain, the first approximate, namely $\{s_0, s_1, s_3\}$. For these states it holds that if a b action is possible, the target state is in the empty set, or in other words, no b action is possible at all.

For the second approximate, we put in $\{s_0, s_1, s_3\}$ instead of X in the formula $[b]X$ and obtain $\{s_0, s_1, s_3\}$ as the second approximate. For these states it holds that if a b action is possible, the target state is in $\{s_0, s_1, s_3\}$.

Since the first approximate and the second approximate coincide, we have reached a fixed point, i.e., $\{s_0, s_1, s_3\}$ is the set of states in which $\mu X.[b]X$ holds. Next, we replace $\mu X.[b]X$ with

its semantics $\{s_0, s_1, s_3\}$ and calculate the semantics of $[a]\mu X.[b]X$. The outcome is $\{s_0, s_2, s_3\}$, which are the states such that after each and every a transition, we end up in $\{s_0, s_1, s_3\}$.

Hence, the final result is $\{s_0, s_2, s_3\}$. Intuitively, this formula describes the set of states such that after each a no infinite sequence of b's is possible. Clearly all states but s_1 satisfy this formula: s_1 can make an a-transition to s_2 after which it can perform infinitely many b's.

Answer 5

```
act   arrive,allow : Nat;
      depart ;

proc  TrackCtrl (occ: Bool, waiting: List ( Nat ) )  =
          ∑i : Nat arrive(i) . (occ) → TrackCtrl (occ, i ▷ waiting ) ⋄ allow(i). TrackCtrl (true, waiting) +
          occ → depart. TrackCtrl (false, waiting) +
          (!occ && waiting ≠ []) → allow(rhead(waiting)). TrackCtrl (true, rtail(waiting));
init  TrackCtrl ([]) ;
```