Prof. Dr. Ir. Dr. h. c. Joost-Pieter Katoen

Christian Hensel, Matthias Volk

Introduction to Model Checking (Summer Term 2018)

— Solution 1 (due 30th April) —

General Remarks

- The exercises are to be solved in groups of *three* students. For sheet one, it is acceptable to form groups of two, but for the remaining sheets, we require you to form groups of three. You may use the L2P forum to search for group members.
- You may hand in your solutions for the exercises just before the exercise class starts at 12:15 or by dropping them into the “Introduction to Model Checking” box at our chair. Do *not* hand in your solutions via L2P or via e-mail.
- The solution for the first exercise sheet will be presented in the first exercise class on April 30th.
- Every sheet is worth 20 points. You need at least 40% of the exercise points to be admitted to the exam. If you gain at least 70% of the points *of the* ***marked**** *exercises* you get a 0.3 bonus on your grade for the exam. Note that this bonus cannot improve your grade if you failed the exam.

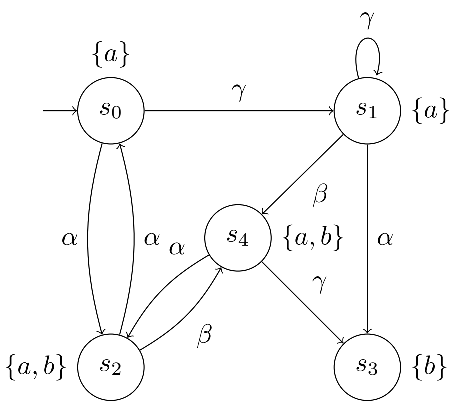
Exercise 1 (2 + 1 + 2 Points)

We call a transition system $\mathrm{TS} = (S, \mathrm{Act}, \rightarrow, S_0, \mathrm{AP}, L)$

- *action-deterministic* if $|\,S_0\,| \leq 1$ and $|\,\mathrm{Post}(s, \alpha)\,| \leq 1$ for all $s \in S$ and $\alpha \in \mathrm{Act}$, and
- *AP-deterministic* if $|\,S_0\,| \leq 1$ and $|\mathrm{Post}(s) \cap \{s' \in S \mid L(s') = A\}\,| \leq 1$ for all $s \in S$ and $A \in 2^{\mathrm{AP}}$,

where $\mathrm{Post}(s, \alpha) = \{s' \in S \mid \exists (s, \alpha, s') \in \rightarrow\}$ and $\mathrm{Post}(s) = \bigcup_{\alpha \in \mathrm{Act}} \mathrm{Post}(s, \alpha)$.

Let the transition system TS_1 be as follows.



(a) Give the formal definition of TS_1.

(b) Specify a finite and an infinite execution of TS_1.

(c) Decide whether TS_1 is (i) *AP*-deterministic, and/or (ii) action-deterministic. Justify your answer.

Solution:

(a) $\mathrm{TS}_1 = (S, Act, \rightarrow, S_0, AP, L)$ where

$$\begin{aligned} S &= \{s_0, \ldots, s_4\} & Act &= \{\alpha, \beta, \gamma\} \\ S_0 &= \{s_0\} & AP &= \{a, b\} \\ L(s_0) &= \{a\} & L(s_1) &= \{a\} \\ L(s_2) &= \{a, b\} & L(s_3) &= \{b\} \\ L(s_4) &= \{a, b\} \end{aligned}$$

Furthermore, $\rightarrow \subseteq S \times Act \times S$ is given as

$$\begin{aligned} \rightarrow = \{&(s_0, \gamma, s_1), (s_0, \alpha, s_2), (s_1, \gamma, s_1), (s_1, \alpha, s_3), (s_1, \beta, s_4) \\ &(s_2, \alpha, s_0), (s_2, \beta, s_4), (s_4, \alpha, s_2), (s_4, \gamma, s_3)\} \end{aligned}$$

(b) A finite execution is an initial, maximal path fragment that ends in a terminal state. For example:

$$s_0 \xrightarrow{\alpha} s_2 \xrightarrow{\beta} s_4 \xrightarrow{\gamma} s_3$$

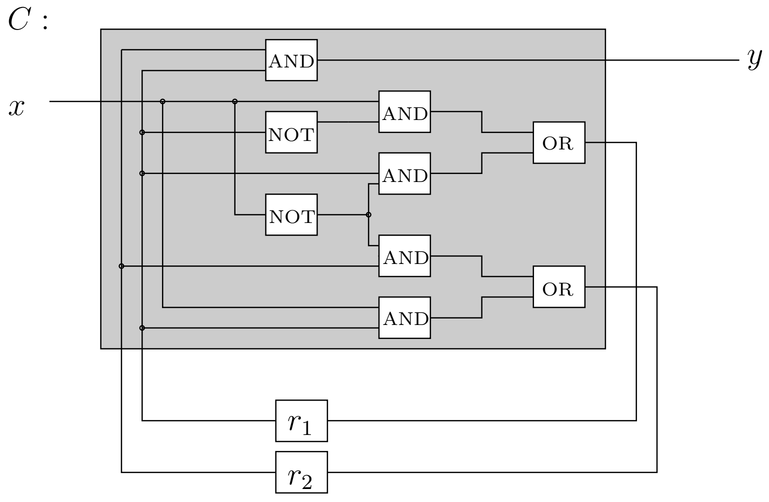
An example of an infinite execution is

$$s_0 \xrightarrow{\alpha} s_2 \left(\xrightarrow{\beta} s_4 \xrightarrow{\alpha} s_2 \right)^{\omega}$$

(c)
- TS_1 is AP-deterministic. There is no state $s \in S$ and action $\alpha \in \mathrm{Act}$ such that two different successors of $s', s'' \in \mathrm{Post}(s, \alpha), s' \neq s''$ have the same label $(L(s') = L(s''))$.
- TS_1 is action-deterministic. There is no state $s \in S$ that has multiple outgoing transitions labeled with the same $\alpha \in \mathrm{Act}$.

Exercise 2 (1 + 1 + 3 + 1 Points)

Consider the following sequential hardware circuit C.



(a) Give the functions λ_y, δ_{r_1} and δ_{r_2} that govern the changes to the output and register bits depending on the input bits.

(b) Formally specify the state space of the transition system TS for the circuit C. Justify your answer.

(c) Draw the transition system TS for the circuit C using the set of labels $\mathrm{AP} = \{x, r_1, r_2, y\}$ assuming that initially the values of the registers are 0. Make sure that ***all* formal components** of the transition system can be uniquely identified from your picture.

(d) Determine the set Reach(TS).

Solution:

(a)

$$\begin{aligned}\lambda_y =& r_1 \wedge r_2\\ \delta_{r_1} =& (x \wedge \neg r_1) \vee (\neg x \wedge r_1) = x \oplus r_1\\ \delta_{r_2} =& (\neg x \wedge r_2) \vee (x \wedge r_1)\end{aligned}$$

(b) The state space of the transition system TS is $\text{Eval}(\{x, r_1, r_2\}) \cong \mathbb{B}^3$ as it needs to store the values of the input and register bit(s). The values of the output bit are derived from these values and are not stored in the state space.

(c) The transition system looks as follows where states are denoted in the form $\langle x, r_1, r_2\rangle$ for readability.

$$\begin{tikzcd}
 & \emptyset & \{r_1\} & \{r_2\} & \{r_1, r_2, y\} \\
{} \arrow[r] & \langle 0,0,0\rangle \arrow[loop above, "\tau"] \arrow[d, "\tau"] & \langle 0,1,0\rangle \arrow[loop above, "\tau"] \arrow[d, "\tau"] & \langle 0,0,1\rangle \arrow[loop above, "\tau"] \arrow[d, "\tau"] & \langle 0,1,1\rangle \arrow[loop above, "\tau"] \arrow[d, "\tau"] \\
{} \arrow[r] & \langle 1,0,0\rangle \arrow[ur, "\tau"] \arrow[r, "\tau"] & \langle 1,1,0\rangle \arrow[ur, "\tau"] \arrow[r, bend left, "\tau"] & \langle 1,0,1\rangle \arrow[l, bend left, "\tau"] \arrow[ul, bend right, "\tau"'] & \langle 1,1,1\rangle \arrow[l, "\tau"] \arrow[ul, bend right, "\tau"'] \\
 & \{x\} & \{x, r_1\} & \{x, r_2\} & \{x, r_1, r_2, y\}
\end{tikzcd}$$

(d) The set of reachable states is

$$\text{Reach(TS)} = \{\langle 0,0,0\rangle, \langle 1,0,0\rangle, \langle 0,1,0\rangle, \langle 1,1,0\rangle, \langle 0,0,1\rangle, \langle 1,0,1\rangle\}\,.$$

Exercise 3★ (2 + 5 + 1 + 1 Points)

Consider the following mutual exclusion algorithm for two processes P_0 and P_1.

The pseudo code of the algorithm for process P_i is given as

```
     int k := 0;
     b := [true, true];
ℓ0 { while (true) do
   {   $b[i] := false$;
ℓ1 {   while (k != i) do
ℓ2 {     while (not b[1-i]) do
   {       k := i;
   {     end
       end
ℓ3 {   critical_section;
ℓ4 {   b[i] := true;
     end
```

where b is an array of two Boolean values which are initially true. k is a variable which is either 0 or 1, and initially 0.

(a) Give the program graph representation for a single process. Use the locations $\ell_0, \dots \ell_4$ corresponding to the indicated program fragments.

(b) Give the reachable part of the transition system of $P_0|||P_1$, i.e., $\text{TS}(P_0|||P_1)$, over the state space $\langle \ell_i, \ell_j, k, b[0], b[1] \rangle$.
Hint: If you wish, you may treat the states $\langle \ell_i, \ell_j, 0, b[0], b[1] \rangle$ and $\langle \ell_j, \ell_i, 1, b[1], b[0] \rangle$ to be equivalent. In other words, whenever you encounter a transition to a state with $k = 1$, you may swap the locations of the processes P_0 and P_1, swap the values of $b[0]$ and $b[1]$, set k to 0, and make the transition target this "new" state instead. This operation considerably reduces the size of the resulting state space.

(c) Check whether the algorithm ensures mutual exclusion, i.e. both processes are never in their critical section at the same time. Justify your answer.

(d) Does the algorithm guarantee that every process eventually enters its critical section? Justify your answer.

Solution:

(a) The program graph for process P_0 is given as $\mathcal{P} = (\text{Loc}, \text{Act}, \text{Effect}, \hookrightarrow, \text{Loc}_0, g_0)$ with:

- $\text{Loc} := \{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4\}$
- $\text{Loc}_0 := \{\ell_0\}$
- $g_0 := \{k = 0, b = [true, true]\}$
- $\text{Act}, \text{Effect}$ and $\hookrightarrow$ as depicted below

$$\begin{tikzcd}
{} \arrow[r] & \ell_0 \arrow[r, "{b[0] := false}"] & \ell_1 \arrow[r, "{\text{if } k = 0}"] \arrow[d, bend left, "{\text{if } k \neq 0}"] & \ell_3 \arrow[r, "\text{critial\_section}"] & \ell_4 \arrow[lll, bend right, "{b[0] := true}"'] \\
 & & \ell_2 \arrow[u, bend left, "{\text{if } b[1]}"] \arrow[loop right, "{\text{if } \neg b[1] \text{ do } k := 0}"] & &
\end{tikzcd}$$

(b) The transition system is given as $\text{TS} = (S, \text{Act}, \rightarrow, S_0, \text{AP}, L)$ and depicted below.

$$\begin{tikzcd}[column sep=small]
\arrow[r] & \langle \ell_0, \ell_0, 0, true, true \rangle \arrow[r] \arrow[d] & \langle \ell_0, \ell_1, 0, true, false \rangle \arrow[r, bend left] \arrow[d] & \langle \ell_0, \ell_2, 0, true, false \rangle \arrow[l, bend left] \arrow[d] & \langle l_2, l_0, 0, false, true \rangle \arrow[d] \arrow[dlll] \\
 & \langle l_1, l_0, 0, false, true \rangle \arrow[r] \arrow[dd] & \langle l_1, l_1, 0, false, false \rangle \arrow[r] \arrow[dd] & \langle l_1, l_2, 0, false, false \rangle \arrow[r] \arrow[dd] & \langle l_2, l_1, 0, false, false \rangle \arrow[loop right] \arrow[d] \\
 & & & & \langle l_2, l_2, 0, false, false \rangle \arrow[loop right] \\
 & \langle l_3, l_0, 0, false, true \rangle \arrow[r] \arrow[d] & \langle l_3, l_1, 0, false, false \rangle \arrow[r] \arrow[d] & \langle l_3, l_2, 0, false, false \rangle \arrow[r] \arrow[d] & \langle \ell_2, \ell_3, 0, false, false \rangle \arrow[loop right] \arrow[d] \\
 & \langle l_4, l_0, 0, false, true \rangle \arrow[r] \arrow[uuuu, bend left] & \langle l_4, l_1, 0, false, false \rangle \arrow[r] \arrow[uuuu, bend left] & \langle l_4, l_2, 0, false, false \rangle \arrow[r] \arrow[uuuu, bend left] & \langle \ell_2, \ell_4, 0, false, false \rangle \arrow[loop right] \arrow[uuuu, bend left]
\end{tikzcd}$$

(c) The algorithm ensures mutual exclusion. This can be easily seen in the transition system where no state $\langle \ell_3, \ell_3, ... \rangle$ exists. That means there is no state where both processes are in the critical section at the same time.

(d) The algorithm does not ensure that every process eventually enters its critical section. Consider the following path:

$$\langle \ell_0, \ell_0, 0, true, true\rangle \rightarrow \langle \ell_1, \ell_0, 0, false, true\rangle \rightarrow \langle \ell_1, \ell_1, 0, false, false\rangle$$
$$\rightarrow \langle \ell_1, \ell_2, 0, false, false\rangle \rightarrow \langle \ell_2, \ell_1, 0, false, false\rangle \rightarrow \langle \ell_2, \ell_2, 0, false, false\rangle$$

After encountering the state $\langle \ell_2, \ell_2, 0, false, false\rangle$ both processes are forever executing the lines `while (not b[1-i]) do k:= i; end` and the critical section can never be reached again by a process.