

Exercise 1 (Transition Systems):

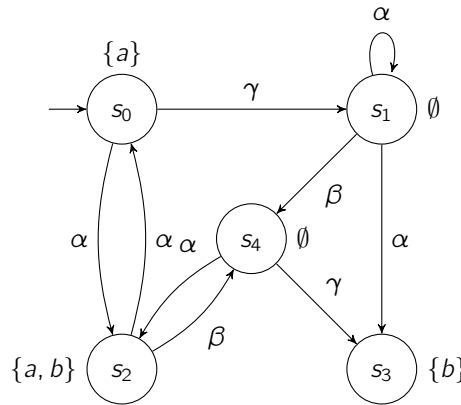
(2 points)

For this exercise we give the following definition:

Definition 1. *Deterministic Transition System* Let $TS = (S, Act, \rightarrow, I, AP, L)$ be a transition system.

1. TS is called *action-deterministic* if $|I| \leq 1$ and $|Post(s, \alpha)| \leq 1$ for all states s and actions α .
2. TS is called *AP-deterministic* if $|I| \leq 1$ and $|Post(s) \cap \{s' \in S \mid L(s') = A\}| \leq 1$ for all states s and $A \in 2^{AP}$.

Consider the following the transition system TS_1 .



- 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 an AP-deterministic or an action-deterministic transition system. Justify your answer.

Solution: _____

- a) $TS_1 = (S, Act, \rightarrow, I, AP, L)$ where

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

Furthermore, $\rightarrow \subseteq S \times Act \times S$ such that

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

- 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 not AP-deterministic: $\text{Post}(s_1) = \{s_1, s_3, s_4\}$ and $\{s' \in S \mid L(s') = \emptyset\} = \{s_1, s_4\}$. Hence s_1 has two successor states with identical labelling, i.e. $|\text{Post}(s_1) \cap \{s' \in S \mid L(s') = \emptyset\}| \not\leq 1$.
- TS_1 is not action-deterministic: $|\text{Post}(s_1, \alpha)| = |\{s_1, s_3\}| \not\leq 1$.

Exercise 2 (Reachability in Parallel Composition):

(1 point)

We are given three (primitive) processes P_1 , P_2 , and P_3 with shared integer variable x and local registers r_1 , r_2 and r_3 . The program of process P_i is as follows:

```
for k := 1 to 10 do {
   $r_i := x$ 
   $r_i := r_i + 1$ 
   $x := r_i$ 
}
```

That is, P_i executes ten times the assignment $x := x + 1$. Consider now the parallel program $P = P_1 \parallel P_2 \parallel P_3$ with x initially being 0. Does P have an execution that halts with the terminal value $x = 2$? Justify your answer.

Solution:

Yes, the program P can halt with output $x = 2$. Consider, for example, the following execution:

- all processes load $x = 0$ into their local registers one after the other;
- P_1 starts counting and is interrupted in its last iteration immediately before executing the loop body;
- P_2 is scheduled (starting with $r_2 = 0$) until it writes $x = 1$ into the shared memory;
- P_1 loads $x = 1$ into its register r_1 ;
- the rest of P_2 and P_3 are executed completely in some arbitrary order;
- P_1 increases its local copy of x by 1 and stores the final result $x = 2$ to the shared memory.

Exercise 3 (Mutual Exclusion):

(5 points)

The following program is a mutual exclusion protocol for two processes due to Pnueli. There is a single shared variable s which is either 0 or 1, and initially 1. Besides, each process has a local Boolean variable y that initially is 0. The program text for process P_i ($i = 0, 1$) is as follows:

```
while (true) do {
  // non-critical section
   $(y_i, s) := (1, i)$ ;
  wait until  $((y_{1-i} = 0) \vee (s \neq i))$ ;
  // critical section
   $y_i := 0$ 
}
```

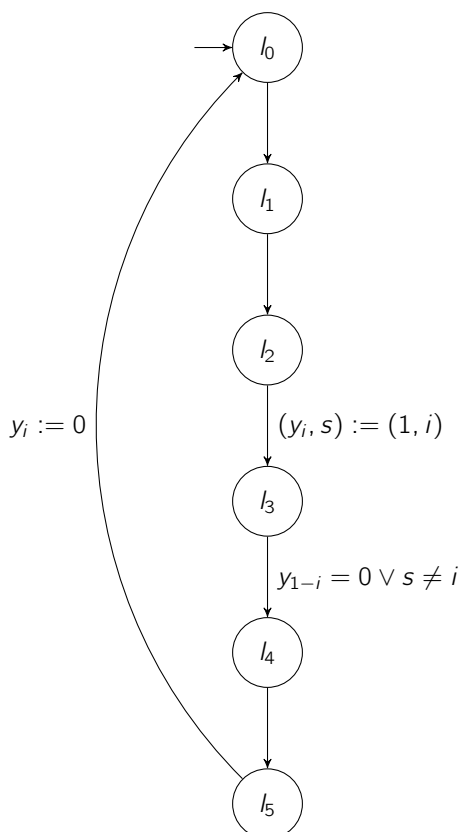
Here, the statement $(y_i, s) := (1, i)$; is a *multi-assignment*, i.e. the assignments $y_i := 1$ and $s := i$ are carried out in a single (atomic) step. Also, checking the boolean condition is done atomically.

- a) Define the program graph of a process in Pnueli's algorithm. Make sure that both the critical and the non-critical section correspond to distinguished locations.

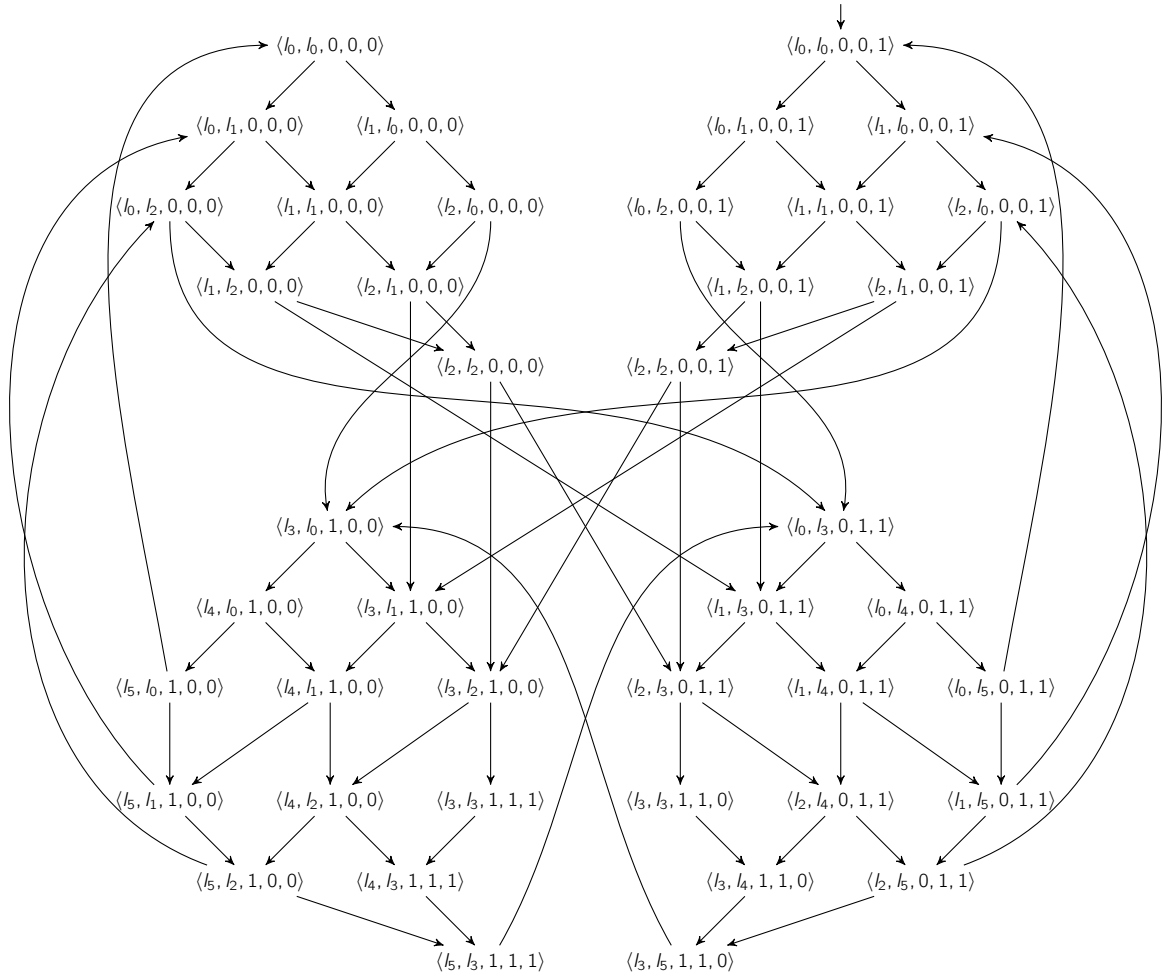
- b) Construct the transition system of their parallel composition $TS(P_0 ||| P_1)$ over the state space $\langle l_i, l_j, y_0, y_1, s \rangle$.
Hint: If you wish, you may treat the states $\langle l_i, l_j, y_0, y_1, 0 \rangle$ and $\langle l_j, l_i, y_1, y_0, 1 \rangle$ to be equivalent. In other words, whenever you encounter a transition to a state with $s = 0$, you may swap the locations of the processes P_0 and P_1 , swap the values of their private variables and set s to 1 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 time a process wants to enter its critical section it can eventually do so (starvation freedom)? Justify your answer.
- e) What happens, if we modify the condition for the waiting phase to $y_{1-i} = 0$ (that is, we drop the second disjunct)? In particular, does the algorithm then satisfy mutual exclusion and/or starvation freedom? Justify your answer.
Hint: If possible, you may use observations on the transition system as a justification.

Solution: _____

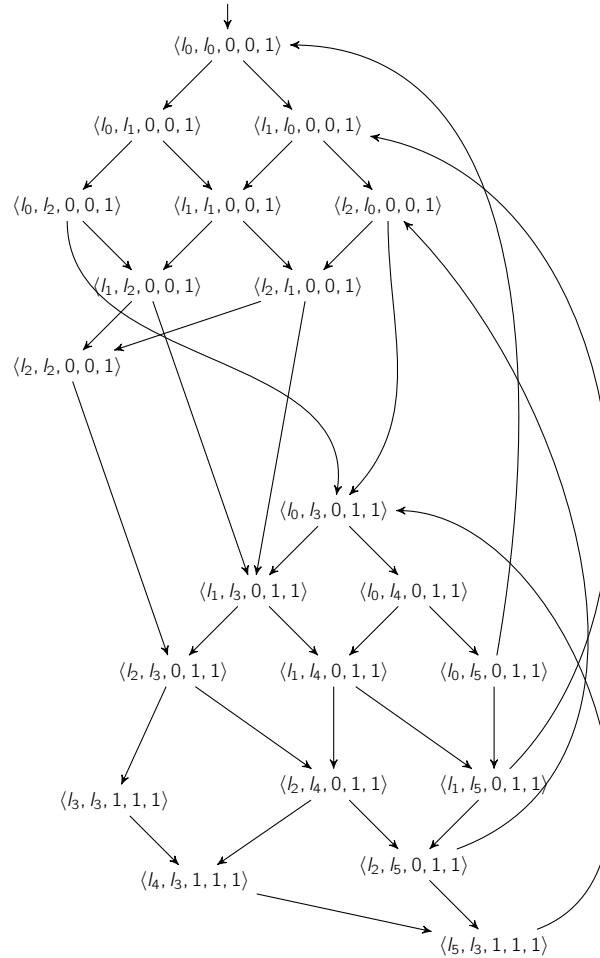
a)



- b) In this transition system, states are of the type $\langle l_i, l_j, y_0, y_1, s \rangle$, where P_0 is in location l_i and P_1 is in location l_j . The reachable part of the transition system is:



If we apply the suggested reduction, we obtain the following transition system:

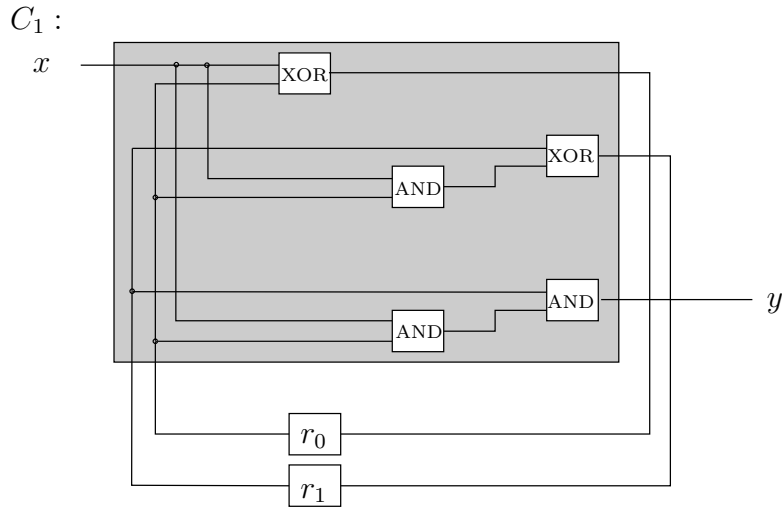


- c) In the reachable fragment, there is no state where both processes are in location l_4 , i.e. in the critical section. So, the algorithm ensures mutual exclusion.
- d) After performing the suggested symmetry reduction, there is no way of telling which process is in which state, since we only observe how many processes are in a given location. That means that we cannot determine whether the algorithm guarantees starvation freedom on the reduced transition system. In fact, the reduced transition system has a loop in which the second process is in location l_3 , i.e. waiting, but never enters the critical section. However, the full transition system shows that this loop was introduced by the abstraction and is not present in the actual behavior of the algorithm. Therefore, starvation freedom is guaranteed.
- e) If the disjunct $s \neq i$ is omitted from the waiting condition, it may happen that both processes indicate that they want to enter the critical section by setting y_i to 1, but then wait for each other forever. Therefore, the algorithm still guarantees mutual exclusion, but starvation freedom is lost. Seeing that in this scenario no further move is possible (deadlock), the situation is even worse.

Exercise 4 (Circuits):

(2 points)

The circuit C_1 describes the layout of a hardware adder that stores a 2-bit binary number represented by the registers r_0 and r_1 . In each cycle, the value of x is added to the currently stored value; y is used as the carry bit:



Give the transition system representation TS_1 of the circuit C_1 .

Solution: _____

The output and the circuit control functions are given as follows:

$$\lambda_y = r_1 \wedge (x \wedge r_0)$$

$$\delta_{r_0} = x \oplus r_0$$

$$\delta_{r_1} = (x \wedge r_0) \oplus r_1$$

The transition system is given by $TS_1 = (\text{Eval}(\{x, r_0, r_1\}), \text{Act}, \rightarrow, I, AP, L)$ where

- $\text{Act} = \{\tau\}$
- $I = \{\langle x, r_0, r_1 \rangle \mid x, r_0, r_1 \in \mathbb{B}\} = \text{Eval}(\{x, r_0, r_1\})$.
- $AP = \{x, r_0, r_1, y\}$
- $\rightarrow$ and $L : S \rightarrow 2^{AP}$ as given in the drawing below in which states are of the form $\langle x, r_0, r_1 \rangle$.

