ENS DE LYON

TD – Semantics and Verification

I– Modelling Concurrent Systems

Thursday 14th January 2021

TA: Ralph Sarkis
ralph.sarkis@ens-lyon.fr

In this first set of exercises, we will discuss the modelling of concurrent systems as circuits, transition systems and program graphs. In particular, we will discuss the interleaving of parallel systems.

Preliminaries: Interleaving Operators

In this preliminary section, we provide the definitions of the different interleaving operators from the book.

Definition 1 (Interleaving of Transition Systems)**.** Let $TS_i = (S_i, \mathrm{Act}_i, \to_i, I_i, \mathrm{AP}_i, L_i)$ be transition systems for $i = 1, 2$. The interleaved transition system is given by

$$TS_1 \,|||\, TS_2 = (S_1 \times S_2, \mathrm{Act}_1 \cup \mathrm{Act}_2, \to, I_1 \times I_2, \mathrm{AP}_1 \cup \mathrm{AP}_2, L),$$

where $L\langle s_1, s_2\rangle = L_1(s_1) \cup L_2(s_2)$ and the transition relation $\to$ is defined by the following two rules.

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle} \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle}$$

Definition 2 (Interleaving of Program Graphs)**.** Let $PG_i = (\mathrm{Loc}_i, \mathrm{Act}_i, \mathrm{Effect}_i, \hookrightarrow_i, \mathrm{Loc}_{0,i}, g_{0,i})$ be program graphs for $i = 1, 2$. The interleaved program graph is given by

$$PG_1 \,|||\, PG_2 = (\mathrm{Loc}_1 \times \mathrm{Loc}_2, \mathrm{Act}_1 \uplus \mathrm{Act}_2, \hookrightarrow, \mathrm{Loc}_{0,1} \times \mathrm{Loc}_{0,2}, g_{0,1} \wedge g_{0,2}),$$

where $\mathrm{Effect}(\alpha, \eta) = \mathrm{Effect}_i(\alpha, \eta)$ for $\alpha \in \mathrm{Act}_i$ and the transition relation $\hookrightarrow$ is defined by the following two rules.

$$\frac{\ell_1 \xhookrightarrow{g:\alpha}_1 \ell_1'}{\langle \ell_1, \ell_2\rangle \xhookrightarrow{g:\alpha} \langle \ell_1', \ell_2\rangle} \qquad \frac{\ell_2 \xhookrightarrow{g:\alpha}_2 \ell_2'}{\langle \ell_1, \ell_2\rangle \xhookrightarrow{g:\alpha} \langle \ell_1, \ell_2'\rangle}$$

Definition 3 (Handshaking)**.** Let $TS_i = (S_i, \mathrm{Act}_i, \to_i, I_i, \mathrm{AP}_i, L_i)$ be transition systems for $i = 1, 2$ and let H be a set of actions with $H \subseteq \mathrm{Act}_1 \cap \mathrm{Act}_2$ and $\tau \notin H$. The synchronised transition system is given by

$$TS_1 \,|||_H\, TS_2 = (S_1 \times S_2, \mathrm{Act}_1 \cup \mathrm{Act}_2, \to, I_1 \times I_2, \mathrm{AP}_1 \cup \mathrm{AP}_2, L),$$

where $L\langle s_1, s_2\rangle = L_1(s_1) \cup L_2(s_2)$ and the transition relation $\to$ is defined by the following three rules.

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2\rangle}\ \alpha \notin H \qquad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1, s_2'\rangle}\ \alpha \notin H \qquad \frac{s_1 \xrightarrow{\alpha}_1 s_1' \quad s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2\rangle \xrightarrow{\alpha} \langle s_1', s_2'\rangle}\ \alpha \in H$$

If $H = \mathrm{Act}_1 \cap \mathrm{Act}_2$, then we abbreviate $TS_1 \,|||_H\, TS_2$ by $TS_1 \parallel TS_2$.

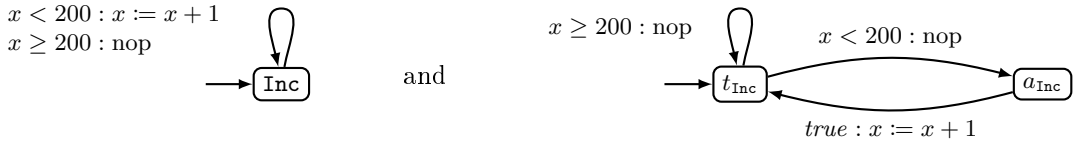
Note: We have that $||| = |||_\emptyset$.

Program Graphs and Atomicity

The first part of the exercise is about different representations of programs as program graphs and the effect of separating tests and assignments. Suppose we are given the following program `Inc`.

`Inc`: **while** *true* **do if** $x < 200$ **then** $x := x + 1$

We may associate two different program graphs, the *atomic* $A_{\texttt{Inc}}$ and the *non-atomic* $N_{\texttt{Inc}}$, to this program:



The first program graph forces the atomic execution of the test and the assignment, while the second program graph separates these two. Since tests and the body of a while-loop are typically more complex, the second program graph is more realistic. These program graphs show by themselves the same behaviour. However, when combined with other processes that access the same variable, then the separation of test and assignment into the two states $t_{\texttt{Inc}}$ and $a_{\texttt{Inc}}$ matters, as we will see in the following exercise.

Exercise 1.

Let the two programs `Dec` and `Res` be given as follows.

`Dec`: **while** *true* **do if** $x > 0$ **then** $x := x - 1$

`Res`: **while** *true* **do if** $x = 200$ **then** $x := 0$

1. Give the atomic and non-atomic program graphs associated to the programs `Dec` and `Res`.
2. Compute $TS(A_{\texttt{Inc}} ||| A_{\texttt{Dec}})$ and $TS(A_{\texttt{Inc}}) ||| TS(A_{\texttt{Dec}})$. Which one better modelize shared memory?
3. Show that $0 \leq x \leq 200$ is an invariant in the interleaving $A_{\texttt{Inc}} ||| A_{\texttt{Dec}} ||| A_{\texttt{Res}}$.
4. Show that there is an execution trace in the interleaving $N_{\texttt{Inc}} ||| N_{\texttt{Dec}} ||| N_{\texttt{Res}}$ in which x is initially positive but becomes negative.

Mutual Exclusion

Exercise 2.

Consider the following mutual exclusion algorithm that was proposed 1966 as a simplification of Dijkstra's mutual exclusion algorithm in case there are just two processes:

boolean array $b = [0; 1]$;
integer $k = 1$, i, j;
`/* This is the program` P_i `for computer` i`, which may be either 0 or 1, computer` $j \neq i$ `is the other one, 1 or 0 */`
C0: $b(i) := \text{false}$;
C1: **if** $k \neq i$ **then**
C2: | **if** $\neg b(j)$ **then goto** C2;
| **else** $k := i$; **goto** C1;
else critical section;
$b(i) := \text{true}$;
remainder of program;
goto C0;

Here *C0*, *C1*, and *C2* are program labels, and the word "computer" should be interpreted as process.

1. Give the program graph representations for a single process. (A pictorial representation suffices.)

2. Give the reachable part of the transition system of $P_0 \,|||\, P_1$.
3. Check whether the algorithm indeed ensures mutual exclusion, that is, check whether that there is no reachable state in which both processes are in their critical section.

Properties of Interleaving

Exercise 3.

Give an example of program graphs PG_1 and PG_2, such that $TS(PG_1) \,|||\, TS(PG_2)$ has evaluations as states that are impossible in $TS(PG_1 \,|||\, PG_2)$ *Hint: In the interleaving* $TS(PG_1) \,|||\, TS(PG_2)$ *the variables of* PG_1 *and* PG_2 *are renamed and thus not shared.*

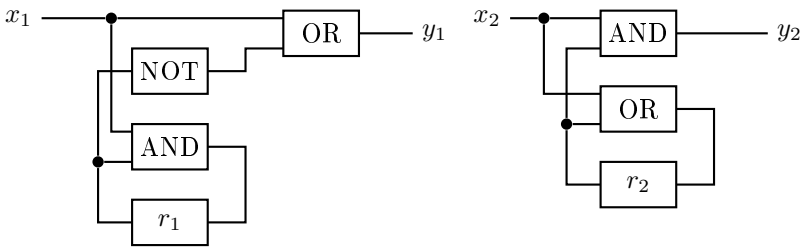
Exercise 4.

Show that the handshaking operator $|||$ is associative. That is, show for arbitrary transition systems TS_1, TS_2, TS_3 that $(TS_1 \,|||\, TS_2) \,|||\, TS_3$ and $TS_1 \,|||\, (TS_2 \,|||\, TS_3)$ are essentially the same transition system.

Sequential Hardware Circuits

Exercise 5.

Consider the following two sequential hardware circuits:



1. Assuming that the initial values of the registers are $r_1 = 0$ and $r_2 = 1$, give the transition systems of both hardware circuits.
2. Determine the reachable part of the interleaving of these transition systems.