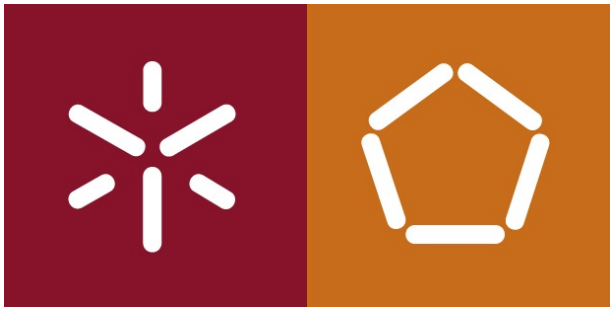


MEI - Cyber-Physical Programming

Jéssica Macedo Fernandes (a93318)

March 14, 2024

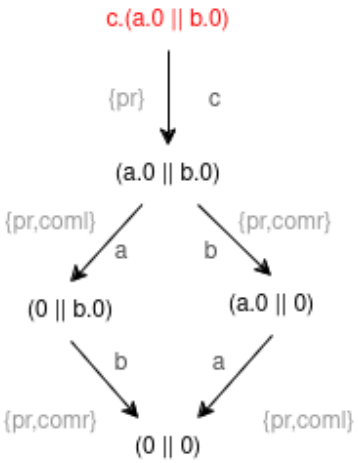
1 Exercise 1

Consider the CCS processes **c.**$(a.0 \parallel b.0)$ and **rec X. (a.X + a.b.X)**.

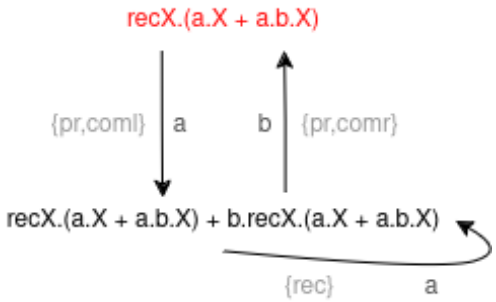
Part 1.1. Informally describe what they do.

The expression **c.**$(\boldsymbol{a.0 \parallel b.0})$ means that the process proceeds through channel 'c' initially, followed by concurrently traversing channel 'a' and terminating, and traversing channel 'b' and terminating subsequently. The expression **rec X. (a.X + a.b.X)** denotes that the process either traverses channel 'a' and then restarts, or traverses channel 'a' followed by channel 'b', and then restarts recursively.

Part 1.2. Present their transition systems using the semantic rules provided in the lectures.



(a) Transition system for **c.**$(a.0 \parallel b.0)$



(b) Transition system for **rec X. (a.X + a.b.X)**

2 Exercise 2

Consider the following scenario. There exist four processes P1,..., P4, each of them responsible for performing a certain task repetitively. For example P1 might read the current velocity, P2 the current altitude, P3 current radiation levels, etc... These processes (re)start their tasks in increasing order (P1 then P2 etc...) but can finish in any order. Note as well that process P1 can restart its task only when all processes P1,..., P4 finish their current tasks. Let us thus consider the process:

$$P = (I \parallel S \parallel P_1 \parallel \cdots \parallel P_4) \backslash \{\text{st1}, \ldots, \text{st4}, \text{end}\}$$

$$\begin{aligned} I &= \text{st1...st4.0} \\ S &= \text{rec } X.\ \text{end.end.end.end.}\overline{\text{st1}}\text{...}\overline{\text{st4}}\text{.X} \\ P_i &= \text{rec } Y_i.\ \text{st}_i\text{.a}_i\text{.b}_i\text{.}\overline{\text{end}}\text{.}Y_i \quad (1 \leq i \leq 4) \end{aligned}$$

Part 2.1. Explain why process P corresponds (or not) to the description above.

The description provided sets some constraints:

- Process P1 can only restart when all processes finish their tasks.
- Processes restart their tasks in asdescending order of process number.
- Processes can finish in any order.

If we look at the scheduler S = rec X. end.end.end.end.$\overline{\text{st1}}$...$\overline{\text{st4}}$.X, we can see that not only do the processes restart in order ($\overline{\text{st1}}$...$\overline{\text{st4}}$), but also that process P1 waits for the others to finish (end.end.end.end), and the order in which they finish is undefined since the 'end's are not specific to each process.
Therefore, we can consider that process P corresponds to the given description, meeting all the constraints.

Part 2.2. Process S acts a central scheduler that coordinates the processes P1,..., P4.
Rewrite P so that it does not rely on a central scheduler and explain the reasoning behind your refactoring.

The issue with losing the scheduler is losing the ability to know if processes have finished their execution, before allowing P1 to restart.

A solution would be to have each process perform its tasks only when the preceding process notifies it of completion, subsequently informing the next one in sequence.
This solution asserts that everyone execute their tasks in order and before P1 restarts. However, this would violate the constraint that processes can finish in any order, as it would necessitate sequential occurrence rather than parallel.

The following solution, P1 initiates its execution and notifies the remaining processes, assuring the order is followed (adhering to the constraint) that they can also start. Each process executes its tasks independently, but before signaling completion, it waits for the preceding one to do so. This ensures that by the time end4 is reached, all processes have terminated, and when this information reaches P1, it can start without concerns, knowing that P2, P3, and P4 have already finished.

Thus, the processes inform each other of their completion, ultimately passing the final message to P1, which restarts the execution of all processes. However, they don't need to wait for others to execute their tasks.

$$
\begin{aligned}
P &= (I \parallel P1 \parallel P2 \parallel P3 \parallel P4)\backslash\{\text{st1, st2,st3,st4,end2,end3}\} \\
I &= \overline{\text{st1}}.0 \\
P1 &= \text{rec } Y1.\text{st1}.a1.b1.\overline{\text{st2}}.\overline{\text{st3}}.\overline{\text{st4}}.Y1 \\
P2 &= \text{rec } Y2.\text{st2.a2.b2.}\overline{\text{end2}}.\text{Y2} \\
P3 &= \text{rec } Y3.\text{st3.a3.b3.end2.}\overline{\text{end3}}.\text{Y3} \\
P4 &= \text{rec } Y4.\text{st4.a4.b4.end3.}\overline{\text{st1}}.\text{Y4}
\end{aligned}
$$