AOR Priv.-Doz. Dr. Thomas Noll

Christina Jansen, Sabrina von Styp

Exercise 1 (CSP Semantics): (4 Points)

Consider the following CSP program c:

$c :=$
$y := 4;$ **if** $(y > 0) \rightarrow ((x := y) \,\|\, (x := 3))$ **fi**
do $(x == 3 \wedge \alpha?x \rightarrow \beta!x) \,\square\, (x == 3 \rightarrow \alpha!y)$ **od**

Provide all "meanings" of c using the formal semantics of CSP as given in the lecture.

Lösung:

Exercise 2 (LTS and Deadlocks): (2+1 Points)

The aim of this exercise is to develop a (simplified) model of a car's central locking system. Assume the following components:

- a door which is either open or closed
- a locker for the door which can be activated if the door is not open (otherwise an alarm should be issued), and
- a key which controls the whole mechanism.

a) Design a corresponding process definition and give its transition system!

b) Check if the car locking system you developed in part a.) has a deadlock. If this is the case, provide a deadlock free specification of the system.

Lösung:

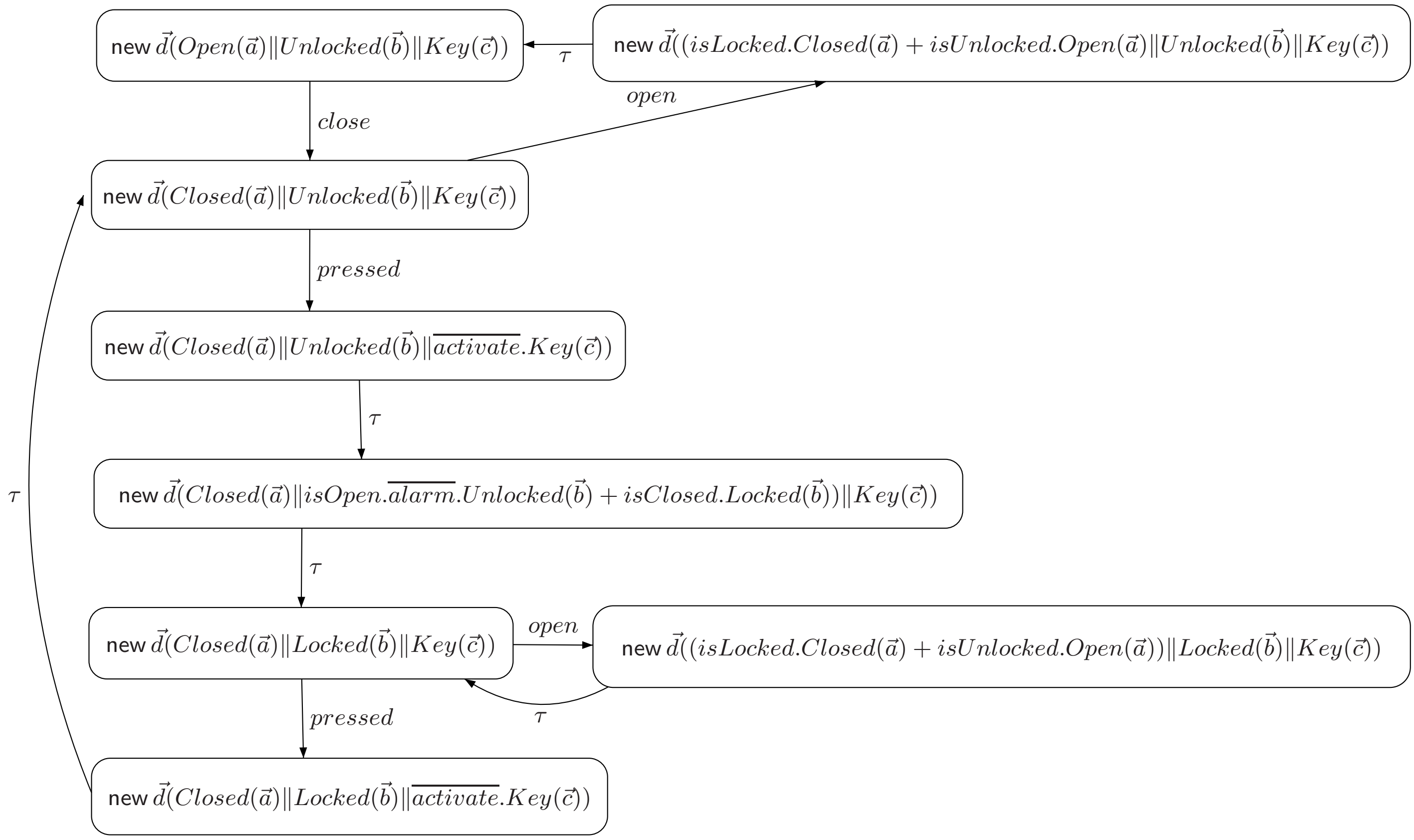
$$
\begin{aligned}
Door(\vec{a}) &= Open(\vec{a})\\
Open(\vec{a}) &= \overline{isOpen}.Open(\vec{a}) + close.Closed(\vec{a})\\
Closed(\vec{a}) &= \overline{isClosed}.Closed(\vec{a}) + open.(isLocked.Closed(\vec{a}) + isUnlocked.Open(\vec{a}))
\end{aligned}
$$

$$
\begin{aligned}
Locker(\vec{b}) &= Unlocked(\vec{b})\\
Unlocked(\vec{b}) &= \overline{isUnlocked}.Unlocked(\vec{b}) + activate.(isOpen.\overline{alarm}.Unlocked(\vec{b}) + isClosed.Locked(\vec{b}))\\
Locked(\vec{b}) &= \overline{isLocked}.Locked(\vec{b}) + activate.Unlocked(\vec{b})
\end{aligned}
$$

$$
Key(\vec{c}) = pressed.\overline{activate}.Key(\vec{c})
$$

$$
System(\vec{e}) = \mathsf{new}\, activate, isOpen, isClosed, isUnlocked, isLocked\, (Door(\vec{a}) \| Locker(\vec{b}) \| Key(\vec{c}))
$$

Here, we do not outline the entire labelled transition system but only a subset that shows the essential idea of the above process definition. To shorten notation, let $\vec{d} = (activate, isOpen, isClosed, isUnlocked, isLocked)$.



Deadlock:

(not complete, just one run!)

new $\vec{\alpha}$ (isL. Closed(a) + isUL. Open($\vec{a}$) || Locked($\vec{b}$) || activate. Key($\vec{c}$))

↓ τ (activate)

new $\vec{\alpha}$ (" || Unlocked($\vec{b}$) || Key($\vec{c}$))

↓ pressed

new $\vec{\alpha}$ (" || " || activate. Key($\vec{c}$))

↓ τ (activate)

new $\vec{\alpha}$ (" || isOpen. alarm. Unlocked($\vec{b}$) + isClosed. Locked($\vec{b}$)) || Key($\vec{c}$))

↓ pressed ↑ τ (activate)

new $\vec{\alpha}$ (" || " || activate. Key($\vec{c}$))

Run (without deadlock)

A || ack. Sync($\vec{\alpha}$))

↓ τ (activate) τ (ack)

(Closed($\vec{a}$) || Unlocked($\vec{b}$) || Key($\vec{c}$) || Sync($\vec{\alpha}$))

↓ OPEN τ(open) τ (isUnlocked) τ (ack)

(Open($\vec{a}$) || Unlocked($\vec{b}$) || Key($\vec{c}$) || Sync($\vec{\alpha}$))

↓ PRESS τ(pressed) τ(activate) τ(isOpen) τ(ack)

(Open($\vec{a}$) || Unlocked($\vec{b}$) || Key($\vec{c}$) || Sync($\vec{\alpha}$)) ALARM!!

⇒ "atomic" closed + locked + pressed Op. via Sync!

Exercise 3 (Parallel Composition of CCS): (2+3 Points)

An engineer is charged with developing an elevator control for a building with five floors, starting with a CCS model. His subspecification for requesting the elevator and selecting the target floor looks as follows:

$$Elevator(req, fl_1, \ldots, fl_5) = req.fl_1.Elevator(req, fl_1, \ldots, fl_5) + \ldots + req.fl_5.Elevator(req, fl_1, \ldots, fl_5).$$

A computer scientist who was called for supporting the engineer suggests the following solution instead:

$$Elevator(req, fl_1, \ldots, fl_5) = req.(fl_1.Elevator(req, fl_1, \ldots, fl_5) + \ldots + fl_5.Elevator(req, fl_1, \ldots, fl_5)).$$

a) Are both systems trace equivalent?

b) Test the elevator subsystem together with the specification of a user who would like to reach the fourth floor:

$$User(req, fl_4) = \overline{req}.\overline{fl_4}.nil.$$

Do both specifications of the elevator guarantee that the user is satisfied?

Lösung:

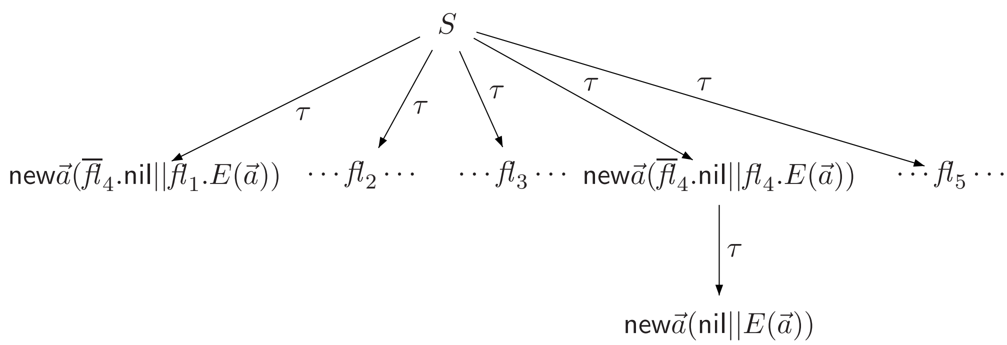
To shorten notation, let $\vec{a} = (req, fl_1, \ldots, fl_5)$.

a) Let $E(\vec{a}) = req.fl_1.E(\vec{a}) + \ldots + req.fl_5.E(\vec{a})$ and $E' = req.(fl_1.E'(\vec{a}) + \ldots + fl_5.E'(\vec{a}))$.
$\Rightarrow Tr(E) = [req.(fl_1 + \ldots + fl_5)]^*.(req + \varepsilon) = Tr(E')$
$\Rightarrow E$ and E' are trace equivalent.

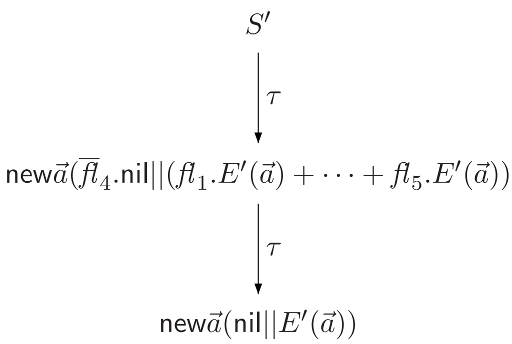
b) To test the elevator specification against a user who wants to go to the fourth floor, we compose the elevator specification and the user in parallel:

$$U(\vec{a}) = \overline{req}.\overline{fl_4}.\mathsf{nil}$$
$$S = \mathsf{new}\, req, fl_1, \ldots, fl_5(U \| E)$$
$$S' = \mathsf{new}\, req, fl_1, \ldots, fl_5(U \| E')$$

In its LTS, process S has five different outgoing τ-transitions:



Formally, $S \xrightarrow{\tau} \mathsf{new}\,\vec{a}[(\overline{fl_4}.\mathsf{nil})\|(fl_i.E(\vec{a})]$ for $1 \leq i \leq 5$. Four of these transitions (those where $i \neq 4$) exhibit τ deadlocks. The LTS of process S' has only one outgoing transition, which does not cause deadlocks. The choice is delayed such that the corresponding communication can take place:



Formally, $S' \xrightarrow{\tau} \mathsf{new}\,\vec{a}[\overline{fl_4}.\mathsf{nil}\|fl_1.E'(\vec{a}) + \ldots + fl_5.E'(\vec{a})] \xrightarrow{\tau} \mathsf{new}\,\vec{a}(\mathsf{nil}||E')$

Thus S' guarantees that the user will reach the fourth floor, whereas S does not.