Prof. Dr. Ir. Joost-Pieter Katoen
apl. Prof. Dr. Thomas Noll
B. Kaminski, H. Wu, S. Chakraborty

Concurrency Theory WS 2013/2014

— Series 6 —

Hand in until December 3rd before the exercise class.

Exercise 1 (Value passing CCS) (3 Points)

1) Define a buffer which can store two integers in value passing CCS.

2) Assume you can define components based on this buffer with your own functionalities (e.g., value comparison, communication with other components). Using your own components to construct such a system using value passing CCS, which can take 3 integers each time through its input port and returns the sorted results through its output port.

Exercise 2 (Redesign the print system in π-calculus) (2 Points)

In the Example 9.9, we have introduced a print system modelled in π-calculus . In this exercise, we want to redesign the system in which client C will ask S to "tunnel" a specific channel for him to the printer P and then send the data via this channel rather than S just pass its channel a to C. Please model the system in π-calculus and convince yourself it is correct.

Exercise 3 (Structural congruence in π-calculus) (2 Points)

Show that

1) if $x \notin \text{fn}(Q)$ then $\mathsf{new}\ x\ Q \equiv Q$;

2) if $Q_1 \equiv Q_2$ then Q_1 and Q_2 have the same free names.

Exercise 4 (Polyadic π-calculus) (3 Points)

We wish to send messages consisting of more than one name. So we want to allow the forms

$$x(y_1 \dots y_n).P \text{ and } \overline{x}\langle z_1 \dots z_n \rangle.Q$$

(where all the y_i are distinct) for any $n \geq 0$. For a correct encoding, we have to ensure that there cannot be an inference on the channel along which a composite message is sent. To send a message $\langle z_1 \dots z_n \rangle$, we first send a *fresh* name w along x, then send the components z_i one by one along w. So we translate the multiple action prefixes as follows:

$$\begin{array}{lll} x(y_1 \dots y_n).P & \longmapsto & x(w).w(y_1).\cdots.w(y_n).P \\ \overline{x}\langle z_1 \dots z_n \rangle.Q & \longmapsto & \mathsf{new}\ w\ (\overline{x}\langle w \rangle.\overline{w}\langle z_1 \rangle.\cdots.\overline{w}\langle z_n \rangle.Q). \qquad \text{where } w \notin \mathsf{fn}(Q) \end{array}$$

Apply this encoding to

$$x(y_1 y_2).P \parallel \overline{x}\langle z_1 z_2 \rangle.Q \parallel \overline{x}\langle z_1' z_2' \rangle.Q' \quad .$$

Do at least two reduction sequences to convince yourself that only the right replacements occur!