Concurrency Theory (WS 2011/12) Out: Tue, Jan 17 Due: Mon, Jan 23

Exercise Sheet 12

Jun.-Prof. Roland Meyer, Georgel Călin Technische Universität Kaiserslautern

Problem 1: Structural Congruence of Restricted Form

Argument the following results on the restricted form of π-calculus processes:

(a) Let $P \in \mathcal{P}$ be a process and $\mathtt{rf}(P) \in \mathcal{P}_{\mathtt{rf}}$ its restricted form. Prove that $P \equiv \mathtt{rf}(P)$.

(b) Let $P, Q \in \mathcal{P}$. Prove that $\mathtt{rf}(P) \equiv_{\mathtt{rf}} \mathtt{rf}(Q)$ if and only if $dec(\mathtt{rf}(P)) = dec(\mathtt{rf}(Q))$.

Problem 2: Structural Congruence & Normalization

(a) Show that $\nu a.P \equiv P$ if $a \notin fn(P)$. *Hint:* 0 *is useful.*

(b) Prove that the following two processes are structurally congruent:

$$P = \nu x \left(\nu s \left(\overline{x}\langle s\rangle.\overline{s}\langle a\rangle.\overline{s}\langle b\rangle \middle| x(u).u(y).u(z).\overline{y}\langle z\rangle\right) \mid x(t).t(w).t(v).\overline{v}\langle w\rangle\right)$$
$$Q = \nu x \left(\nu s \left(\overline{x}\langle s\rangle.\overline{s}\langle a\rangle.\overline{s}\langle b\rangle \middle| x(t).t(w).t(v).\overline{v}\langle w\rangle\right) \mid x(u).u(y).u(z).\overline{y}\langle z\rangle\right)$$

(c) Prove that each π-calculus process is structurally congruent to a process of the form

$$\nu x_1 \ldots \nu x_m. (P_1 | \ldots | P_n)$$

where each P_i is a choice or a call to an identifier.

Problem 3: Interpretation of Polyadic π-Calculus

Polyadic π-calculus is a generalization of π-calculus which allows using tuples as inputs/outputs:

- ○ $c(x_1, \ldots, x_n)$ denotes binding the input n-tuple on c pointwise to $(x_1, \ldots, x_n)$
- • $\overline{c}\langle a_1, \ldots, a_n\rangle$ denotes sending the tuple $(a_1, \ldots, a_n)$ on channel c.

A naive way of understanding the above is by:

$$c(x_1, \ldots, x_n) := c(x_1).c(x_2) \ldots c(x_n) \text{ and } \overline{c}\langle a_1, \ldots, a_n\rangle := \overline{c}\langle a_1\rangle.\overline{c}\langle a_2\rangle \ldots \overline{c}\langle a_n\rangle.$$

Why is this interpretation not satisfying ○ and •? Enhance the naive encoding to make it correct. This shows that polyadic π-calculus can be encoded into standard (called *monadic*) π-calculus.

Hint: think of $\overline{c}\langle a, b\rangle | c(x, y) | c(x', y')$. *Restricted names are helpful.*

Problem 4: π-calculus interpretation of FSA and PN

This exercise is meant to familiarize you with the behaviour (and expressiveness) of π-calculus.

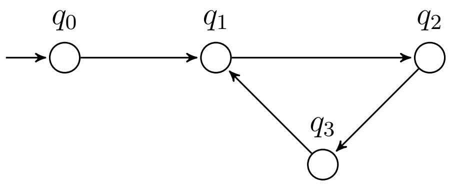
Let $\mathcal{A} = (Q, q_0, \to)$ be an arbitrary finite state automaton. By using a free name s for each state $s \in Q$, a configuration (state at runtime) of $\mathcal{A}$ is represented by

$$\overline{q}\langle q\rangle \Big| \prod_{q \to q'} K_{q\to q'}\lfloor q, q'\rfloor,$$

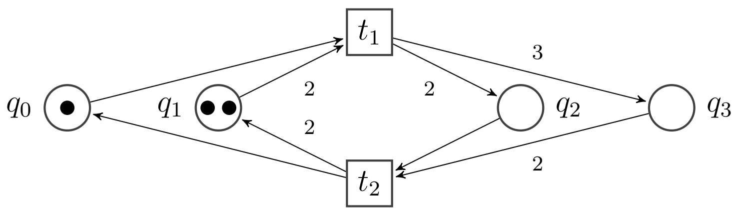
where $K_{q\to q'}(q, q') := q(x).\,\big(K_{q\to q'}\lfloor q, q'\rfloor \,\big|\, \overline{q'}\langle q'\rangle\big)$ describes the $q \to q'$ transition of $\mathcal{A}$.

The automaton is then described by $\overline{q_0}\langle q_0\rangle \Big| \prod_{q\to q'} K_{q\to q'}\lfloor q, q'\rfloor$.

(a) Represent the *FSA* below using the method above:



(b) Extend the described method to Petri nets and apply it to the following net:



Hints: – *synchronize the execution of transitions; you need deadlocks*

– *change the defining equation* $K_t(s, s') :=$ `here` $\big(K_t\lfloor s, s'\rfloor \,\big|\,$ `here` $\big)$

What are your observations on the process syntax? What is the size of the processes as compared to the automaton (Petri net) they represent?