Concurrency (exam 2007-2008, second part)

You may consult the slides of the lectures. No other document or electronic device is allowed. Answers should be formulated in French or English, and preferably in a rigorous and sharp style. Write the solutions in a sheet different from the one used for the first part of the exam.

Exercise 1 (another definition of determinacy, 4 points) *Reminder. In the context of CCS, we have said that a process P is* determinate *if for any sequence s of observable actions, if $P \stackrel{s}{\Rightarrow} P_i$ for $i = 1, 2$ then $P_1 \approx P_2$. Also we denote with $sort(P)$ the collection of observable actions a derivative of P may perform.*

We introduce an alternative notion of 'R-determinacy'. We denote with R a process of the shape $\ell_1.\ell_2 \cdots \ell_n.0$ where ℓ_i are observable actions for $i = 1, \ldots, n$, and $n \geq 0$. We say that a process P is R-determinate *if for any process R of the shape above such that $sort(P) \cap sort(R) = \emptyset$, if $(P \mid R) \stackrel{\tau}{\Rightarrow} P_i$ for $i = 1, 2$ then $P_1 \approx P_2$. Prove or disprove the following assertions.*

(1) *If P is R-determinate and $P \approx Q$ then Q is R-determinate.*

SOL. Preliminary: $sort(a) \cap sort(\overline{a}) = \emptyset$.
First notice that $P \approx Q$ implies $sort(P) = sort(Q)$.
Hence $sort(Q) \cap sort(R) = \emptyset$ implies $sort(Q) \cap sort(R) = \emptyset$.
Suppose $(Q \mid R) \stackrel{\tau}{\Rightarrow} Q_i$ for $i = 1, 2$.
Because weak bisimulation is preserved by parallel composition we know that $(P \mid R) \approx (Q \mid R)$.
By definition of bisimulation, $\exists\, P_i \;\; (P \mid R) \stackrel{\tau}{\Rightarrow} P_i$ and $P_i \approx Q_i$ for $i = 1, 2$.
Because P is R-determinate, and by the previous remark, we know $P_1 \approx P_2$.
By transitivity of $\approx$, we conclude $Q_1 \approx Q_2$.

(2) *If P is R-determinate and $P \xrightarrow{\alpha} P'$ then P' is R-determinate, where α is any action.*

SOL. False. Consider $P = a.(b.c + b.d) + \overline{a}$.
If $sort(P) \cap sort(R) = \emptyset$ we have that $P \mid R \stackrel{\tau}{\not\rightarrow}$because P and R cannot synchronise and R alone cannot do τ actions.
Hence P is R-determinate. On the other hand, $P \xrightarrow{a} Q$ with $Q = (b.c + b.d)$ and taking $R = \overline{b}$ we see that $(Q \mid R) \xrightarrow{\tau} c$, $(Q \mid R) \xrightarrow{\tau} d$, and obviously $c \not\approx d$.

(3) *If P is determinate then it is R-determinate.*

SOL. False. Consider $P = a$. Then P is determinate.
On the other hand, taking $R = \overline{a}$ we see that $P \mid R \stackrel{\tau}{\Rightarrow} P \mid R$ and $P \mid R \stackrel{\tau}{\Rightarrow} 0$, and obviously $(P \mid R) \not\approx 0$.

(4) *If P is R-determinate then it is determinate.*

SOL. False, taking the same example as in (2). P is R-determinate but not determinate.

Exercise 2 (affinity and TCCS, 5 points) *In the course, we have defined an* affine *type system for a* monadic *π-calculus.*

(1) *In the* polyadic *π-calculus channels carry vectors of names and channel types have the shape $Ch(\tau_1, \ldots, \tau_n)$. Propose a generalisation to the* polyadic *π-calculus of the affine typing system. You will focus on the rules for input and output and notice that, as a special case, when the vector of names has length 0, you get the typing rules for CCS.*

Sol.

$$\frac{\Gamma \vdash a : Ch_u(\sigma_1, \ldots, \sigma_n) \quad \pi_2(u) = 1 \quad \Gamma, b_1 : \sigma_1, \ldots, b_n : \sigma_n \vdash P}{\Gamma \vdash a(b_1, \ldots, b_n).P}$$

$$\frac{\Gamma_0 \vdash a : Ch_u(\sigma_1, \ldots, \sigma_n) \quad \pi_1(u) = 1 \quad \Gamma_i \vdash b_i : \sigma_i \quad i = 1, \ldots, n \quad \Gamma_0 \vdash P}{(\Gamma_0 \oplus \Gamma_1 \oplus \cdots \oplus \Gamma_n) \vdash \overline{a}b.P}$$

(2) *Propose a typing rule for the operator else_next of* timed *CCS (TCCS). The resulting system should type the process P_1 while it should* not *type the process P_2 defined as follows:*

$$P_1 = (a.0 \triangleright b.a.0) \qquad P_2 = ((a.0) \triangleright (0 \triangleright a.0)) \mid (0 \triangleright a.0) \ .$$

Show the typing derivation of P_1 and explain why the typing of P_2 fails.

Sol.

$$\frac{\Gamma \vdash P \quad \Gamma \vdash Q}{\Gamma \vdash (P \triangleright Q)}$$

Let $\Gamma = a : Ch_{(0,1)}, b : Ch_{(0,1)}$. One checks $\Gamma \vdash a.0$ and $\Gamma \vdash b.a.0$. Hence $\Gamma \vdash P_1$.
On the other hand, the typing of P_2 fails for any context Γ since to type the two parallel components of P_2 we need the input usage and then the addition of two input usages is illegal.

(3) *Your typing system should have the property that if $\Gamma \vdash P$ and $P \stackrel{\mathsf{tick}}{\rightarrow} Q$ then $\Gamma \vdash Q$. Prove that.*

Sol. One proceeds by induction on $\Gamma \vdash P$.
For instance, suppose $\Gamma \vdash (P \triangleright Q)$ because $\Gamma \vdash P$ and $\Gamma \vdash Q$. Further suppose $(P \triangleright Q) \stackrel{\mathsf{tick}}{\rightarrow} Q$. Then $\Gamma \vdash Q$.

(4) *Your typing system should still have the* substitution *property stated in the course. Prove that, focusing just on the situations that do not arise in the system for the π-calculus.*

Sol. One proceeds by induction on $\Gamma, a : \sigma \vdash P$.
For instance, suppose $\Gamma, a : \sigma \vdash (P \triangleright Q)$, $\Gamma' \vdash b : \sigma$ and $(\Gamma \oplus \Gamma') \downarrow$.
By the typing rule for *else_next* we know that $\Gamma, a : \sigma \vdash P$ and $\Gamma, a : \sigma \vdash Q$.
By inductive hypothesis, $(\Gamma \oplus \Gamma') \vdash [b/a]P$ and $(\Gamma \oplus \Gamma') \vdash [b/a]P$.
Then by the typing rule for *else_next* we conclude that $(\Gamma \oplus \Gamma') \vdash ([b/a]P \triangleright [b/a]Q)$ and we notice that $[b/a](P \triangleright Q) = ([b/a]P \triangleright [b/a]Q)$.