Concurrence — Partiel

19 November 2006

Instructions: you have three hours to solve all the exercises. Only the slides of the lectures and your personal notes are authorised. You can admit the result of one question and move on. Whenever you must exhibit a bisimulation, you must specify the candidate relation and justify why it is closed under the relevant conditions. An extra bonus applies if you solve correctly all the questions of one exercise. Leave optional questions at the end.

Exercise 1. Consider CCS with non-deterministic guarded sums and definition of recursive equations. For each proposition below, prove it or exhibit a counterexample.

1. Let P and Q be two processes such that it holds $(\boldsymbol{\nu} a)P \sim (\boldsymbol{\nu} a)Q$ for a name a. Then $P \approx Q$.

   Answer. The statement is false. Let $P = a$ and $Q = \mathbf{0}$. The relation $\mathcal{R} = \{((\boldsymbol{\nu} a)a, (\boldsymbol{\nu} a)\mathbf{0})\}$ is a bisimulation. However, for $\phi = \langle a \rangle T$ it holds $P \Vdash \phi$ while $Q \not\Vdash \phi$. From the completness of Hennessy-Milner logic, it follows that $P \not\sim Q$.

2. Let P and Q be two processes such that there exists a process R such that $P + \tau.R \approx Q + \tau.R$. Then $P \approx Q$.

   Answer. The statement is false. Let $P = a + b$ and $Q = b$. Clearly $P \not\approx Q$. However, for $R = a$, it holds $P + \tau.R = a + b + \tau.a \approx b + \tau.a = Q + \tau.R$. To prove this, either check that the relation $\mathcal{R} = \{(a + b + \tau.a, b + \tau.a), (a, a), (\mathbf{0}, \mathbf{0})\}$ is a weak bisimulation, or simply apply the second τ-law of the axiomatization of weak bisimulation.

3. Let P and Q be two processes such that $!P \approx\, !Q$. Then $P \approx Q$.

   Answer. The statement is false. Let $P = a \parallel a$ and $Q = a$. Clearly $P \not\approx Q$. However $!P \approx\, !Q$, because it is easy to see that the relation

   $$\mathcal{R} = \{(\underbrace{a \parallel \ldots \parallel a}_{n \text{ times}} \parallel !(a \parallel a), !a) : n \geq 0\}$$

   is a bisimulation.

4. Let $a \triangleright b = !a.\bar{b}$ and let a, b, c be distinct names. It holds that $(\boldsymbol{\nu} b)(a \triangleright b \parallel b \triangleright c) \approx a \triangleright c$.

   Answer. It is not difficult to verify that the relation

   $$\mathcal{R} = \{((\boldsymbol{\nu} b)(\underbrace{b \parallel \ldots \parallel b}_{m_1 \text{ times}} \parallel \underbrace{c \parallel \ldots \parallel c}_{m_2 \text{ times}} \parallel a \triangleright b \parallel b \triangleright c), \underbrace{c \parallel \ldots \parallel c}_{n \text{ times}} \parallel a \triangleright c) : m_1, m_2, n \geq 0 \text{ and } m_1 + m_2 = n\}$$

   is a bisimulation up to $\equiv$.

Exercise 2. Let $\mathcal{L}$ be a set of names, ranged over by $x, y, \ldots$. Consider the language *source* defined by the grammar:

$$T \quad ::= \quad \texttt{newloc}\ x\ y.T \quad \big|\quad \texttt{read}\ x\ (y).T \quad \big|\quad \texttt{write}\ x\ y.T \quad \big|\quad \mathbf{0}$$

Intuitively, `newloc` $x\ y.T$ creates a new location called x containing y and resumes as T, `read` $x\ (y).T$ reads the content of the location x and binds it to y in the continuation T, `write` $x\ y.T$ updates the content of the location x with the value y and resumes as T.

Environments ρ are terms defined by the grammar $\rho \ ::= \ \emptyset \ \mid \ (x, y) \cdot \rho$. We call the pair $\langle \rho; T \rangle$ a configuration. The small step semantics of the source language is defined as

$$\frac{}{\langle \rho\, ;\, \texttt{newloc}\ x\ y.T \rangle \rightharpoonup \langle (x, y) \cdot \rho\, ;\, T \rangle} \qquad \frac{\texttt{update}\ \rho\ (x, y) = \rho'}{\langle \rho\, ;\, \texttt{write}\ x\ y.T \rangle \rightharpoonup \langle \rho'\, ;\, T \rangle}$$

$$\frac{\texttt{lookup}\ \rho\ x = z}{\langle \rho\,;\ \texttt{read}\ x\ (y).T\rangle \twoheadrightarrow \langle \rho\,;\ T\{^z/_y\}\rangle}$$

where the functions `lookup` and `update` are defined as:

$$\begin{array}{ll} \texttt{lookup}\ ((x,y)\cdot\rho)\ x = y & \texttt{lookup}\ ((x,y)\cdot\rho)\ z = \texttt{lookup}\ \rho\ z \quad \text{if } x \neq z \\ \texttt{update}\ ((x,y)\cdot\rho)\ x\ v = (x,v)\cdot\rho & \texttt{update}\ ((x,y)\cdot\rho)\ z\ v = (x,y)\cdot(\texttt{update}\ \rho\ z\ v) \quad \text{if } x \neq z \end{array}$$

and undefined otherwise (eg. when updating a location x not defined in ρ).

1. Reduce the term $\langle \emptyset\,;\ \texttt{newloc}\ x\ y.\texttt{newloc}\ x\ z.\texttt{write}\ x\ v.\mathbf{0}\rangle$.

   Answer. $\langle \emptyset\,;\ \texttt{newloc}\ x\ y.\texttt{newloc}\ x\ z.\texttt{write}\ x\ v.\mathbf{0}\rangle \twoheadrightarrow\twoheadrightarrow\twoheadrightarrow \langle (x,v)\cdot(x,y)\cdot\mathbf{0}\,;\ \mathbf{0}\rangle$

(Remark: Answers of 2. and 3. should validate the property stated in question 5.)

2. Define a function $[[-]]$ that encodes the terms T of the source language into terms of the pi-calculus. For that, *encode a location x containing y as a process that sends y over the channel x.*

   Answer.
   $$\begin{array}{lcll} [[\mathbf{0}]] & = & \mathbf{0} & \\ [[\texttt{newloc}\ x\ y.T]] & = & (\boldsymbol{\nu} x)(\overline{x}\langle y\rangle \parallel [[T]]) & \\ [[\texttt{read}\ x\ (y).T]] & = & x(y).(\overline{x}\langle y\rangle \parallel [[T]]) & \\ [[\texttt{write}\ x\ y.T]] & = & x(v).(\overline{x}\langle y\rangle \parallel [[T]]) & v \text{ fresh for } \texttt{write}\ x\ y.T \end{array}$$

3. Reusing the previous function, define an encoding of configurations $\langle \rho\,;\ T\rangle$ into pi-calculus. Be careful because the environment ρ can contain several pairs for a location x, but only the last one is actually accessible.

   Answer. Let
   $$\begin{array}{lcl} [[\langle \mathbf{0}\,;\ T\rangle]] & = & [[T]] \\ [[\langle (x,y)\cdot\rho\,;\ T\rangle]] & = & (\boldsymbol{\nu} x)(\overline{x}\langle y\rangle \parallel [[\langle \rho\,;\ T\rangle]]) \end{array}$$

   The translation of a configuration $\langle \rho\,;\ T\rangle$ is defined as $[[\langle \rho'\,;\ T\rangle]]$ where $\rho' = \texttt{reverse}(\rho, \mathbf{0})$. The auxiliary function `reverse` reverses a list and is defined as $\texttt{reverse}(\mathbf{0}, \rho) = \rho$ and $\texttt{reverse}((x,y)\cdot\rho, \rho') = \texttt{reverse}(\rho, (x,y)\cdot\rho')$.

4. Apply your translation to the configuration of question 1. and reduce the term so obtained. Does it reduce to the translation of the term that answers question 1.?

   Answer. The translated term is $(\boldsymbol{\nu} x)(\overline{x}\langle y\rangle \parallel (\boldsymbol{\nu} x)(\overline{x}\langle z\rangle \parallel x(w).\overline{x}\langle v\rangle))$. It reduces to $(\boldsymbol{\nu} x)(\overline{x}\langle y\rangle \parallel (\boldsymbol{\nu} x)\overline{x}\langle v\rangle)$, which is the translation of the configuration $\langle (x,v)\cdot(x,y)\cdot\mathbf{0}\,;\ \mathbf{0}\rangle$.

5. We say that two environments ρ_1 and ρ_2 are equivalent, denoted $\rho_1 \simeq \rho_2$, if for all x it holds that $\texttt{lookup}\ \rho_1\ x = \texttt{lookup}\ \rho_2\ x$. Sketch the proof of the correctness of your encoding: if $\langle \rho\,;\ T\rangle \twoheadrightarrow \langle \rho'\,;\ T'\rangle$ then $[[\langle \rho\,;\ T\rangle]] \twoheadrightarrow [[\langle \rho''\,;\ T'\rangle]]$, where $\rho'' \simeq \rho'$. Proceed by case analysis on the transitions of the source language.

   Answer.

   - Case $\langle \rho\,;\ \texttt{newloc}\ x\ y.P\rangle \twoheadrightarrow \langle (x,y)\cdot\rho\,;\ P\rangle$. The encoding of $\langle \rho\,;\ \texttt{newloc}\ x\ y.P\rangle$ is $C[(\boldsymbol{\nu} x)(\overline{x}\langle y\rangle.[[P]]]$ where the context $C[-]$ is obtained by translating the configuration $\langle \rho\,;\ -\rangle$. The term $C[(\boldsymbol{\nu} x)(\overline{x}\langle y\rangle.[[P]]]$ which is equal to the translation of $\langle (x,y)\cdot\rho\,;\ P\rangle$.
   - Case $\langle \rho\,;\ \texttt{write}\ x\ y.T\rangle \twoheadrightarrow \langle \rho'\,;\ P\rangle$ because $\texttt{update}\ \rho\ (x,y) = \rho'$. The encoding of $\langle \rho\,;\ \texttt{write}\ x\ y.P\rangle$ is $C[x(v).(\overline{x}\langle v\rangle \parallel [[P]])]$ where $C[-]$ contains $\overline{x}\langle y\rangle$. The translated term reduces to $C'[\overline{x}\langle v\rangle \parallel [[P]]]$ where $C'[-]$ is obtained from $C[-]$ by removing the output $\overline{x}\langle y\rangle$. This outcome corresponds to the encoding of $\langle \rho'\,;\ T\rangle$ except that the location (x,v) is defined on top of the candidate environment, and not in the position where it was before. However, the two environments are equivalent.
   - The case for `read` is similar to the case `write`.

Exercise 3. The *private pi-calculus* is a symmetric variant of the pi-calculus in which only fresh names are exchanged: the output $\overline{x}(y)$ can be thought of as $(\boldsymbol{\nu} y)\overline{x}\langle y\rangle$. It is defined by the grammar

$$\begin{array}{rcl} P & ::= & \mathbf{0} \;\big|\; \pi.P \;\big|\; (\boldsymbol{\nu} z)P \;\big|\; \Sigma_i \pi_i.P_i \;\big|\; P \parallel P \\ \pi & ::= & x(z) \;\big|\; \overline{x}(z) \end{array}$$

In the terms $x(z).P$, $\overline{x}(z).P$, and $(\boldsymbol{\nu} z)P$, the z is bound in the continuation P (observe that, contrarily to standard pi-calculus, the output prefix is a binder). Its LTS is defined as

$$x(v).P \xrightarrow{x(v)} P \qquad \overline{x}(v).P \xrightarrow{\overline{x}(v)} P \qquad \frac{z \notin \mathsf{n}(\ell)}{(\boldsymbol{\nu} z)P \xrightarrow{\ell} (\boldsymbol{\nu} z)P'}$$

$$\frac{}{\Sigma_i \pi_i.P_i \xrightarrow{\pi_i} P_i} \qquad \frac{P \xrightarrow{\overline{x}(v)} P' \quad Q \xrightarrow{x(v)} Q'}{P \parallel Q \xrightarrow{\tau} (\boldsymbol{\nu} v)(P' \parallel Q')} \qquad \frac{P \xrightarrow{\ell} P' \quad \mathsf{bn}(\ell) \cap \mathrm{fn}(Q) = \emptyset}{P \parallel Q \xrightarrow{\ell} P' \parallel Q}$$

where $\mathsf{n}(\overline{x}(v)) = \mathsf{n}(x(v)) = \{x, v\}$, $\mathsf{n}(\tau) = \emptyset$, and $\mathrm{fn}(\overline{x}(v)) = \mathrm{fn}(x(v)) = \{x\}$, $\mathrm{fn}(\tau) = \emptyset$. Symmetric rules for parallel composition have been omitted. Observe that although substitutions are never explicitly mentioned, they intervene by α-conversion before each communication. For instance,

$$\overline{x}(z).P \;\Big|\Big|\; x(y).Q \;=_\alpha\; \overline{x}(z).P \;\Big|\Big|\; x(z).Q\{^z/_y\} \xrightarrow{\tau} (\boldsymbol{\nu} z)(P \;\Big|\Big|\; Q\{^z/_y\}) \;.$$

1. Define an appropriate structural congruence relation $\equiv$ and reduction semantics $\rightarrow$ for the private pi-calculus such that $P \rightarrow P'$ if and only if $P \xrightarrow{\tau}\equiv P'$. *Optional:* prove that.

   Answer.

$$P \;\Big|\Big|\; Q \equiv Q \;\Big|\Big|\; P \qquad P \;\Big|\Big|\; (Q \;\Big|\Big|\; R) \equiv (P \;\Big|\Big|\; Q) \;\Big|\Big|\; R \qquad P \;\Big|\Big|\; \mathbf{0} \equiv P$$

$$(\boldsymbol{\nu} v)P \;\Big|\Big|\; Q \equiv (\boldsymbol{\nu} v)(P \;\Big|\Big|\; Q) \text{ if } v \notin \mathrm{fn}(Q) \qquad (\boldsymbol{\nu} x)(\boldsymbol{\nu} y)P \equiv (\boldsymbol{\nu} y)(\boldsymbol{\nu} x)P$$

$$\Sigma_i \pi_i.P_i \equiv \Sigma_i \pi_{f(i)}.P_{f(i)} \text{ for } f \text{ permutation}$$

$$P_1 + \overline{x}(v).P \parallel Q_1 + x(v).Q \rightarrow (\boldsymbol{\nu} v)(P \parallel Q) \qquad \frac{P \rightarrow P'}{(\boldsymbol{\nu} x)P \rightarrow (\boldsymbol{\nu} x)P'} \qquad \frac{P \rightarrow P'}{P \parallel Q \rightarrow Q \parallel P}$$

$$\frac{P' \equiv P \rightarrow Q \equiv Q'}{P' \rightarrow Q'}$$

Let *strong ground bisimilarity*, denoted $\sim_g$, be the largest symmetric relation over private pi-calculus processes such that whenever $P \sim_g Q$, there is $z \notin \mathrm{fn}(P, Q)$ such that if $P \xrightarrow{\ell} P'$ where ℓ is $x(z)$, or $\overline{x}(z)$, or τ, then there exists Q' such that $Q \xrightarrow{\ell} Q'$ and $P' \sim_g Q'$.

2. Prove that

$$a(x).(\overline{x}(v) \;\Big|\Big|\; y(w)) \;\sim_g\; a(x).(\overline{x}(v).y(w) + y(w).\overline{x}(v)) \;.$$

   Does this equation hold in the pi-calculus equipped with strong bisimilarity?

   Answer. The relation

$$\begin{array}{rl} \mathcal{R} = \{ & (a(x).(\overline{x}(v) \parallel y(w)), a(x).(\overline{x}(v).y(w) + y(w).\overline{x}(v))), \\ & (\overline{x}(v) \parallel y(w), \overline{x}(v).y(w) + y(w).\overline{x}(v)), \\ & (y(w), y(w)), (\overline{x}(v), \overline{x}(v)), (\mathbf{0}, \mathbf{0})\} \end{array}$$

   is a strong ground bisimulation. This equation does not hold in the pi-calculus equipped with strong bisimilarity, because after $\xrightarrow{a(y)}$ the left hand side can perform a τ transition that cannot be matched by the right hand side.

3. Prove that strong ground bisimilarity on private pi-calculus is a congruence with respect to input prefix, that is, if $P \sim_g Q$ then for all x, z it holds $x(z).P \sim_g x(z).Q$.

   Answer. Let $\mathcal{R} = \{(x(z).P, x(z).Q) : P \sim_g Q\}$. We show that $\mathcal{R}$ is a bisimulation.

   Suppose that $x(z).P \xrightarrow{x(z)} P$ for $z \notin \mathrm{fn}(x(z).P, x(z).Q)$. By definition of ground bisimilarity there exists Q' such that $x(z).Q \xrightarrow{x(z)} Q'$ and $P \sim_g Q'$. But the definition of the LTS guarantees that $Q' = Q$, and the result follows by the construction of $\mathcal{R}$.

4. *Optional.* Prove that strong ground bisimilarity is a congruence with respect to parallel composition, that is, if $P \sim_g Q$ then for all R it holds $P \parallel R \sim_g Q \parallel R$.

   Answer. Let $\mathcal{R} = \{((\boldsymbol{\nu}\tilde{v})(P \parallel R), (\boldsymbol{\nu}\tilde{v})(Q \parallel R)) : P \sim_g Q\}$. We show that $\mathcal{R}$ is a bisimulation.

   Suppose that $(\boldsymbol{\nu}\tilde{v})(P \parallel R) \xrightarrow{\ell} P_1$ because $P \xrightarrow{\ell} P'$ and ℓ is $x(z)$, or $\overline{x}\langle z\rangle$, or τ for $x \notin \tilde{v}$, $z \notin \mathrm{fn}((\boldsymbol{\nu}\tilde{v})(P \parallel R))$ and $z \notin \mathrm{fn}((\boldsymbol{\nu}\tilde{v})(Q \parallel R)$. Then $P_1 = (\boldsymbol{\nu}\tilde{v})(P' \parallel R)$. By definition of ground bisimilarity there exists Q' such that $Q \xrightarrow{\ell} Q'$ and $P' \sim_g Q'$. The LTS allows the transition $Q \parallel R \xrightarrow{\ell} (\boldsymbol{\nu}\{\}\tilde{v})(Q' \parallel R)$ and $(\boldsymbol{\nu}\tilde{v})(P' \parallel R) \mathrel{\mathcal{R}} (\boldsymbol{\nu}\tilde{v})(Q' \parallel R)$ follows from the definition of $\mathcal{R}$.

   The case $(\boldsymbol{\nu}\tilde{v})(P \parallel R) \xrightarrow{\ell} P_1$ because $R \xrightarrow{\ell} R'$ is similar (simpler).

   Suppose that $(\boldsymbol{\nu}\tilde{v})(P \parallel R) \xrightarrow{\ell} P_1$ because $P \xrightarrow{x(v)} P'$, $R \xrightarrow{\overline{x}(v)} R'$, and $P_1 = (\boldsymbol{\nu}v)(\boldsymbol{\nu}\tilde{v})(P' \parallel R')$. The variable $v \notin \mathrm{fn}(P)$, and we can choose it so that $v \notin \mathrm{fn}(P, Q)$. By definition of ground bisimilarity there exists Q' such that $Q \xrightarrow{x(v)} Q'$ and $P' \sim_g Q'$. The LTS allows the transition $(\boldsymbol{\nu}\tilde{v})(Q \parallel R) \xrightarrow{\tau} (\boldsymbol{\nu}v)(\boldsymbol{\nu}\tilde{v})(Q' \parallel R)$ and $(\boldsymbol{\nu}v)(\boldsymbol{\nu}\tilde{v})(P' \parallel R) \mathrel{\mathcal{R}} (\boldsymbol{\nu}v)(\boldsymbol{\nu}\tilde{v})(Q' \parallel R)$ follows from the definition of $\mathcal{R}$.

   The case $(\boldsymbol{\nu}\tilde{v})(P \parallel R) \xrightarrow{\ell} P_1$ because $P \xrightarrow{\overline{x}(v)} P'$, $R \xrightarrow{x(v)} R'$ is similar.

5. Can strong ground bisimilarity be defined over standard pi-calculus processes? If yes, describe the advantages and inconvenients with respect to strong bisimilarity.

   Answer. When applied on standard pi-calculus processes, strong ground bisimilarity is not preserved by parallel composition. As such, it is not a sound proof technique for any contextual equivalence.