Concurrency

Partiel, 24 november 2006

Instructions: you have three hours to solve all the exercises. Leave optional questions to the end. All documents 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. We may omit trailing dead processes.

CCS - deadlock Say that an agent can *deadlock* if it can perform a sequence of actions to become an agent strongly bisimilar to $\mathbf{0}$.

1. For any agent P, show that $P \sim \mathbf{0}$ if and only if P can do no action.

   Answer. Suppose $P \sim \mathbf{0}$. By contradiction. Suppose that $P \xrightarrow{\alpha} P'$ for some α and P'. Since $P \sim \mathbf{0}$, it must hold that $\mathbf{0} \xrightarrow{\alpha} Q$ for some Q. This is impossible by definition of the LTS. Then there is no α, P' such that $P \xrightarrow{\alpha} P'$. Conversely, suppose that P can do no action. Then $\{(P, \mathbf{0})\}$ is a bisimulation.

2. Consider the following example of a bidirectional 1-place communication channel in CCS:

$$Chan(i_1, i_2, o_1, o_2) = i_1.\overline{o_1}.Chan(i_1, i_2, o_1, o_2) + i_2.\overline{o_2}.Chan(i_1, i_2, o_1, o_2)$$

   graphically represented as: By means of only constant application, parallel composition, and restriction, use two copies of $Chan$ to construct a bidirectional 2-place channel.

   Answer. $(\boldsymbol{\nu} a, b)(Chan(i_1, b, a, o_2) \parallel Chan(a, i_2, o_1, b))$.

3. Is the resulting channel deadlock free? If it is not, revise the definition of $Chan$ to make it so.

   Answer. Let C be the process defined in answer 2. The resulting channel C is not deadlock free because $C \xrightarrow{i_1}\xrightarrow{i_2} (\boldsymbol{\nu} a, b)(a.Chan(i_1, b, a, o_2) \parallel \bar{b}.Chan(a, i_2, o_1, b))$, which is bisimilar to $\mathbf{0}$. If we revise the definition of $Chan$ as:

$$\begin{array}{rcll} Chan(i_1, i_2, o_1, o_2) & = & i_1.A(i_1, i_2, o_1, o_2) + i_2.B(i_1, i_2, o_1, o_2) & \text{no tokens received} \\ A(i_1, i_2, o_1, o_2) & = & \overline{o_1}.Chan(i_1, i_2, o_1, o_2) + i_2.D(i_1, i_2, o_1, o_2) & \text{token on } i_1 \\ B(i_1, i_2, o_1, o_2) & = & \overline{o_2}.Chan(i_1, i_2, o_1, o_2) + i_1.D(i_1, i_2, o_1, o_2) & \text{token on } i_2 \\ D(i_1, i_2, o_1, o_2) & = & \overline{o_1}.B(i_1, i_2, o_1, o_2) + \overline{o_2}.A(i_1, i_2, o_1, o_2) & \text{tokens on } i_1 \text{ and } i_2 \end{array}$$

   then the resulting bidirectional 2-place channel is deadlock free.

   Remark. Solution based on parallel composition, eg.

$$Chan(i_1, i_2, o_1, o_2) = i_1.\overline{o_1}.Chan(i_1, i_2, o_1, o_2) \parallel i_2.\overline{o_2}.Chan(i_1, i_2, o_1, o_2)$$

   are not correct because

$$Chan(i_1, i_2, o_1, o_2) \xrightarrow{i_1}\xrightarrow{\bar{o}_1} Chan(i_1, i_2, o_1, o_2) \parallel Chan(i_1, i_2, o_1, o_2)$$

   which is not a 1-place communication channel anymore.

4. Write down a Hennessy-Milner logic proposition Φ such that P satisfies Φ if and only if P can deadlock.

   Answer. $\Phi = \bigvee_{s \in Act^*} \langle s \rangle \bigwedge_{a \in Act} [a] F$.

CCS - equivalences

1. Let $X = \tau.X + \tau.a.\mathbf{0}$, and $Y = \tau.Y + a.\mathbf{0}$. For each combination of processes $P, Q \in \{a.\mathbf{0}, X, Y\}$ determine whether or not P and Q are (a) strongly bisimilar, and (b) weakly bisimilar. Justify your answer by either providing a suitable relation, or argue why no such relation can exist.

   Answer.

   Strong bisimilarity: $X \not\sim Y$ because $Y \xrightarrow{a} \mathbf{0}$ and $X \not\xrightarrow{a}$; $X \not\sim a.\mathbf{0}$ because $X \xrightarrow{\tau} X$ and $a.\mathbf{0} \not\xrightarrow{\tau}$; $Y \not\sim a.\mathbf{0}$ because $Y \xrightarrow{\tau} Y$ and $a.\mathbf{0} \not\xrightarrow{\tau}$.

   Weak bisimilarity: the relation $\mathcal{R} = \{(X,Y), (X, a.\mathbf{0}), (Y, a.\mathbf{0}), (a.\mathbf{0}, a.\mathbf{0}), (\mathbf{0}, \mathbf{0})\}$ is a weak bisimulation. We deduce that $X \approx Y$, $X \approx a.\mathbf{0}$, and $Y \approx a.\mathbf{0}$.

2. Consider the CCS processes

$$S = \tau.b.S \qquad P = \overline{a}.b.P + b.P \qquad Q = a.Q\ .$$

   Using equational reasoning and the unique solution theorem prove $S = (\boldsymbol{\nu} a)(P \parallel Q)$.

   Answer. First, using equation (3) (expansion law), we get

$$\mathcal{A}_1 \vdash P \parallel Q = \tau.(b.P \parallel Q) + \overline{a}.(b.P \parallel Q) + b.(P \parallel Q) + a.(P \parallel Q)$$

   Then we use equation (4):

$$\mathcal{A}_1 \vdash (\boldsymbol{\nu} a)(P \parallel Q) = \tau.(\boldsymbol{\nu} a)(b.P \parallel Q)$$

   Similarly, using (3) and (4), we have:

$$\mathcal{A}_1 \vdash (\boldsymbol{\nu} a)(b.P \parallel Q) = b.(\boldsymbol{\nu} a)(P \parallel Q)$$

   It follows that $(\boldsymbol{\nu} a)(P \parallel Q)$ is solution of the same equation as the one defining S, hence $(\boldsymbol{\nu} a)(P \parallel Q) \sim S$ by the unique solution theorem (strong case).

pi-calculus - proxies Let

$$P = !p(x,r).\overline{r}\langle x\rangle \qquad\qquad C[-] = (\boldsymbol{\nu} p)(R\lfloor a\rfloor \parallel R\lfloor b\rfloor \parallel -)$$

where $R = (x).(\boldsymbol{\nu} n)\overline{p}\langle x, n\rangle.n(z)$. The process P can be thought of as a simple server (it computes the identity function on names) listening at p, and the instances of R as clients of the server.

1. Our aim is to build a proxy that intercepts and logs the invocations of P. Let p' be a fresh name, and let the context $F[-]$, defined as

$$F[-] = (\boldsymbol{\nu} p')(!p(x,r).\tau.\overline{p'}\langle x, r\rangle \parallel (\boldsymbol{\nu} p)(- \parallel !p'(x,r).\overline{p}\langle x, r\rangle))$$

   be such a proxy (the explicit action τ corresponds to the action of logging the communication). By comparing a trace of $C[F[P]]$ and $C[P]$ explain informally the behaviour of $F[-]$.

   Answer. (trace construction omitted.) In $C[P]$ the processes R send a value over the channel p. The process P receives it and sends a reply. In $C[F[P]]$, R and P cannot communicate because they do not share the channel p anymore. The output of R over p interacts with the input over p of $F[-]$. This information is sent again over p' after the τ action, and then sent to P over the channel p shared between $F[-]$ and P. The channel p' is used to carry x, r under the scope of the p shared between $F[-]$ and P.

2. The context $F[-]$ does not intercepts the replies of the server. Starting from the definition of $F[-]$, build a context $L[-]$ that intercepts also the replies of the server to the client, and that realises an explicit τ action if such interaction takes place.

   Answer. $(\boldsymbol{\nu} p')(!p(x,r).(\boldsymbol{\nu} r')(\overline{p'}\langle x, r'\rangle.r'(x).\tau.\overline{r}\langle x\rangle) \parallel (\boldsymbol{\nu} p)(- \parallel !p'(x,r).\overline{p}\langle x, r\rangle))$

3. Prove that $F[P] \approx P$, thus showing that the proxy does not influence the behaviour of the server.

   Answer. We will write $(\!|P|\!)^n_{x,y,(x_1,y_1),\ldots,(x_n,y_n)}$ for $\underbrace{P\{^{x_1,y_1}/_{x,y}\} \parallel \cdots \parallel P\{^{x_n,y_n}/_{x,y}\}}_{n \text{ times}}$. By abuse of notation, we will omit the substituted names, writing simply $(\!|P|\!)^n_{x,y}$. Let $F_1 = !p(x,r).\tau.\overline{p'}\langle x,r\rangle$ and let $F_2 = !p'(x,r).\overline{p}\langle x,r\rangle$. The relation

   $$\begin{aligned}\{\ (\ & (\!|\overline{x}\langle r\rangle|\!)^n_{x,r} \parallel P\ ,\\ & (\boldsymbol{\nu}p')((\!|\tau.\overline{p'}\langle x,r\rangle|\!)^{m_1}_{x,r} \parallel (\!|\overline{p'}\langle x,r\rangle|\!)^{m_2}_{x,r} \parallel F_1 \parallel (\boldsymbol{\nu}p)((\!|\overline{x}\langle r\rangle|\!)^{m_3}_{x,r} \parallel P \parallel (\!|\overline{p}\langle x,r\rangle|\!)^{m_4}_{x,r} \parallel F_2))\ ) :\\ & n = m_1 + m_2 + m_3 + m_4\\ & \quad \text{and each } (x_i, y_i) \text{ substituted on the LHS appears in one substitution on the RHS } \}^{=}\end{aligned}$$

   is a bisimulation. The intuition behind this relation is that each request received by P can appear at an arbitrary stage of processing in $F[P]$.

   Remark. Most of the solutions proposed did not consider the fact that $F[P]$ can receive a request for p at any stage of the treatment of a previous request, and ignored that arbitrary r and x could be received.

4. (Optional) Prove that $(\boldsymbol{\nu}p)(R\lfloor a\rfloor \parallel R\lfloor b\rfloor \parallel P) \approx (\boldsymbol{\nu}p)(R\lfloor a\rfloor \parallel P) \parallel (\boldsymbol{\nu}p)(R\lfloor b\rfloor \parallel P)$

   Answer. Let $A = (\boldsymbol{\nu}p)(R\lfloor a\rfloor \parallel R\lfloor b\rfloor \parallel P)$ and let $B = (\boldsymbol{\nu}p)(R\lfloor a\rfloor \parallel P) \parallel (\boldsymbol{\nu}p)(R\lfloor b\rfloor \parallel P)$. Since $\text{fv}(A) = \emptyset$, the relation $\{(A', \mathbf{0}) : A \rightarrow\!\!\!\triangleright^* A'\}^{=}$ is a bisimulation and $A \approx \mathbf{0}$. Similarly, $B \approx \mathbf{0}$. The result follows from transitivity of $\approx$.

pi-calculus - odd bisimilarity Pi-calculus bisimilarity tests that the continuations of bisimilar processes realising an input action are themselves bisimilar after receiving an *arbitrary* name. A well-known researcher proposed an alternative definition of bisimilarity, named odd bisimilarity, that tests the continuations upon receiving only names known to both the tested processes.

Formally, *odd bisimilarity*, denoted $\doteq$, is the largest symmetric relation between pi-calculus processes such that whenever $P \doteq Q$,

(a) if $z \in \text{fn}(P) \cap \text{fn}(Q)$ and $P \xrightarrow{x(z)} P'$, then there exists a process Q' such that $Q \stackrel{x(z)}{\Longrightarrow} Q'$ and $P' \doteq Q'$;

(b) if α is not an input action then $P \xrightarrow{\alpha} P'$ implies that there exists a process Q' such that $Q \stackrel{\hat{\alpha}}{\Longrightarrow} Q'$ and $P' \doteq Q'$.

We study the congruence properties of this relation.

1. Prove that if $P \doteq Q$ then for all z it holds that $(\boldsymbol{\nu}z)P \doteq (\boldsymbol{\nu}z)Q$.

   Answer. Let $\mathcal{R} = \{((\boldsymbol{\nu}z)P, (\boldsymbol{\nu}z)Q) : P \doteq Q\} \cup \doteq$. If we show that $\mathcal{R}$ is an odd bisimulation, then the result will follow by coinduction. If $P\ \mathcal{R}\ Q$ because $P \doteq Q$, then the result follows trivially. Suppose then that $(\boldsymbol{\nu}z)P \doteq (\boldsymbol{\nu}z)Q$ because $P \doteq Q$. We perform a case analysis on the actions realised by $(\boldsymbol{\nu}z)P$.

   - If $(\boldsymbol{\nu}z)P \xrightarrow{\alpha} P'$ for $\alpha \in \{\tau, \overline{x}\langle y\rangle, (\boldsymbol{\nu}y)\overline{x}\langle y\rangle\}$, then $P \xrightarrow{\alpha} P''$ and $P' = (\boldsymbol{\nu}z)P''$. Since $P \doteq Q$, there exists Q'' such that $Q \xrightarrow{\alpha} Q''$ and $P'' \doteq Q''$. Then $(\boldsymbol{\nu}z)Q \xrightarrow{\alpha} (\boldsymbol{\nu}z)Q''$ and $(\boldsymbol{\nu}z)P''\ \mathcal{R}\ (\boldsymbol{\nu}z)Q''$ by construction of $\mathcal{R}$.
   - If $(\boldsymbol{\nu}z)P \xrightarrow{(\boldsymbol{\nu}z)\overline{x}\langle z\rangle} P'$ then $P \xrightarrow{\overline{x}\langle z\rangle} P'$. Since $P \doteq Q$, there exists Q' such that $Q \xrightarrow{\overline{x}\langle z\rangle} Q'$ and $P' \doteq Q'$. Then $(\boldsymbol{\nu}z)Q \xrightarrow{(\boldsymbol{\nu}z)\overline{x}\langle z\rangle} Q'$ and $P'\ \mathcal{R}\ Q'$ by construction of $\mathcal{R}$.
   - If $(\boldsymbol{\nu}z)P \xrightarrow{x(y)} P'$ with $y \in \text{fn}((\boldsymbol{\nu}z)P) \cap \text{fn}((\boldsymbol{\nu}z)Q)$, then $P \xrightarrow{x(y)} P'$ and $y \neq z$. Since $y \in \text{fn}((\boldsymbol{\nu}z)P) \cap \text{fn}((\boldsymbol{\nu}z)Q)$, it holds that $y \in \text{fn}(P) \cap \text{fn}(Q)$. Since $P \doteq Q$, there exists Q'' such that $Q \xrightarrow{x(y)} Q''$ and $P'' \doteq Q''$. Then $(\boldsymbol{\nu}z)Q \xrightarrow{x(y)} (\boldsymbol{\nu}z)Q''$, and $(\boldsymbol{\nu}z)P''\ \mathcal{R}\ (\boldsymbol{\nu}z)Q''$ by construction of $\mathcal{R}$.

2. By exhibiting a counterexample, show that it is not true that if $P \doteq Q$ then for all processes R it holds that $P \parallel R \doteq Q \parallel R$.

   Answer. Let $P = x(y).\mathbf{0}$ and $Q = z(w).\mathbf{0}$. Since $\mathrm{fv}(P) \cap \mathrm{fv}(Q) = \emptyset$ the relation {(P,Q)} is an odd-bisimulation and $P \doteq Q$. Let $R = \overline{x}\langle a\rangle.\overline{b}\langle b\rangle$. We have $P \parallel R \not\doteq Q \parallel R$ because $P \parallel R \xrightarrow{\tau} \overline{b}\langle b\rangle \xrightarrow{\overline{b}\langle b\rangle}$ and $Q \parallel R$ cannot mimick this sequence of actions.

3. Deduce that odd bisimilarity is not a sound proof method for reduction-closed barbed congruence.

   Answer. Consider the processes P, Q, and R defined in Answer 2. It holds $P \doteq Q$ but $P \not\cong Q$ because the context $R \parallel -$ tells them apart: $R \parallel P \twoheadrightarrow \overline{b}\langle b\rangle \downarrow b$ while $R \parallel Q \not\Downarrow b$. This shows that $\doteq \not\subseteq \cong$, that is odd bisimilarity is not a sound proof method for reduction-closed barbed congruence.

   Remark. Since odd bisimilarity is not a congruence with respect to parallel composition, the opposite inclusion $\cong \subseteq \doteq$ does not hold either.

4. (Optional) Is odd bisimilarity an equivalence relation?

   Answer. Odd bisimilarity is not transitive. Let $P = x(y), Q = z(y).\overline{y}\langle y\rangle$, and $R = x(y).\overline{y}\langle y\rangle$. It holds that $P \doteq Q$ and $Q \doteq R$, but $P \not\doteq R$ because $x \in \mathrm{fn}(P) \cap \mathrm{fn}(R)$ and $R \xrightarrow{x(x)}\xrightarrow{\overline{x}\langle x\rangle} \mathbf{0}$ and P cannot reply to the second action realised by R.