Reactivity and Local Confluence

Termination and Local confluence

- In rewriting theory, local confluence is the following condition:

$$\frac{t \to t_1, \quad t \to t_2}{\exists\, s \;\; (t_1 \xrightarrow{*} s, \quad t_2 \xrightarrow{*} s)}$$

  Thus with local confluence we just have to close *one step* diagrams.

- Newman's lemma states that *local confluence* plus *termination* entails *confluence*.

- We present a suitable generalisation of this result to our framework.

- We begin by recalling the classic proof of Newman's lemma.

Well-founded sets and induction principle

Well-founded set A *well-founded* set $(W, >)$ is a set W equipped with a transitive relation $>$ which does not contain infinite descending sequences:

$$x_0 > x_1 > x_2 > \cdots$$

Hence $>$ must be *strict*. If $x \in W$ let $\downarrow(x) = \{y \in W \mid x > y\}$.

Induction principle Let $(W, >)$ be a well-founded set and let $A \subseteq W$.

$$\frac{\forall x \; (\downarrow(x) \subseteq A \;\supset\; x \in A \;)}{A = W}$$

Exercice (general culture, optional)

Let $(X, >)$ be a set X with a transitive relation $>$. Show that $(X, >)$ is well-founded if and only if the induction principle holds on $(X, >)$.

Local confluence and Newman lemma

Let $(X, \to)$ be a rewriting system, *i.e.*, a set X with a binary relation $\to$. $(X, \to)$ is:

confluent if for all $x \in X$

$$\frac{x \overset{*}{\to} x_1, \quad x \overset{*}{\to} x_2}{\exists\, y \;\; (x_1 \overset{*}{\to} y, \quad x_2 \overset{*}{\to} y)}$$

locally confluent if for all $x \in X$

$$\frac{x \to x_1, \quad x \to x_2}{\exists\, y \;\; (x_1 \overset{*}{\to} y, \quad x_2 \overset{*}{\to} y)}$$

terminating if all reduction sequences $x_0 \to x_1 \to \cdots$ are finite.

Newman's lemma

If a rewriting system is *locally confluent* and *terminates* then it is *confluent*.

Proof

- Consider a rewriting system $(X, \rightarrow)$.
- If $\overset{+}{\rightarrow}$ denotes the transitive closure of $\rightarrow$ then $(X, \rightarrow)$ terminates iff $(X, \overset{+}{\rightarrow})$ is well-founded.
- Let $A \subseteq X$ be the set of *elements* for which the confluence condition holds.

- We know that $(X, \xrightarrow{+})$ is well-founded. We show that $A = X$ by applying the induction principle.

$$\frac{\forall x \ (\downarrow (x) \subseteq A \ \supset \ x \in A \ )}{A = X}$$

- If x is a normal form then $x \in A$.

- Otherwise, suppose

$$x \longrightarrow x_1 \stackrel{*}{\longrightarrow} x_2, \quad x \longrightarrow x_3 \stackrel{*}{\longrightarrow} y_4$$

- By local confluence,

$$\exists\, x_5 \;\; (x_1 \stackrel{*}{\longrightarrow} x_5, \quad x_3 \stackrel{*}{\longrightarrow} x_5)$$

- By inductive hypothesis on x_1,

$$\exists\, x_6 \;\; (x_2 \stackrel{*}{\longrightarrow} x_6, \quad x_5 \stackrel{*}{\longrightarrow} x_6)$$

- Again by inductive hypothesis on x_3,

$$\exists\, x_7 \;\; (x_6 \stackrel{*}{\longrightarrow} x_7, \quad x_4 \stackrel{*}{\longrightarrow} x_7)$$

Back to processes...

- A process P is *terminating* (or strongly normalising) if there is no infinite sequence

$$P \xrightarrow{\tau} P_1 \xrightarrow{\tau} \cdots$$

- A process P is *reactive* (or fully terminating) if all its derivatives are terminating.

Exercise

Consider again the process

$$A(a,b) = a.\nu c\ (A(a,c) \mid \overline{b}.A(c,b))$$

Is the process $A(a,b)$ reactive? Consider the cases $a \neq b$ and $a = b$.

Exercise

Consider the process:

$$A = a.b + \tau.(a.c + \tau.A)$$

Check whether A is:

1. τ-inert,
2. locally confluent,
3. terminating.
4. reactive.
5. determinate.
6. confluent.

A generalisation of Newman's lemma

- Suppose P is a *reactive* process and let W be the set of its derivatives.
- For $Q, Q' \in W$ write $Q > Q'$ if Q rewrites to Q' by a positive number of τ-actions.
- Then $(W, >)$ is a well founded order.

Proposition If a process is *reactive* and *locally confluent* then it is *confluent*.

Proof outline

Let B be the relation $\stackrel{\tau}{\rightarrow} \cup (\stackrel{\tau}{\rightarrow})^{-1} \cup \approx$ (restricted to W) and B^* its reflexive and transitive closure. Note B^* is symmetric too.

1. For every derivative Q of P it holds:

$$\frac{Q \stackrel{\tau}{\Rightarrow} Q_1, \quad Q \stackrel{\alpha}{\Rightarrow} Q_2}{\exists\, Q_3 \;\; (\; Q_1 \stackrel{\alpha}{\Rightarrow} Q_3, \quad Q_2 B^* Q_3 \;)}$$

2. The relation B^* is a weak-bisimulation.

3. The process P is τ-inert.

4. The process P is confluent.

NB B^* is a binary relation on W (the derivatives of P).

Step 1

For every derivative Q of P it holds:

$$\frac{Q \stackrel{\tau}{\Rightarrow} Q_1, \quad Q \stackrel{\alpha}{\Rightarrow} Q_2}{\exists Q_3 \ (\ Q_1 \stackrel{\alpha}{\Rightarrow} Q_3, \quad Q_2 B^* Q_3 \ )}$$

Proof By induction on the well founded order $(W, >)$.

- If $Q = Q_1$ then the statement holds trivially.
- So assume $Q \stackrel{\tau}{\rightarrow} Q_3 \stackrel{\tau}{\Rightarrow} Q_1$ and consider 2 cases.
  1. If $Q \stackrel{\tau}{\rightarrow} Q_4 \stackrel{\alpha}{\Rightarrow} Q_2$.
     - By local confluence, $Q_3 \stackrel{\tau}{\Rightarrow} Q_5$, $Q_4 \stackrel{\tau}{\Rightarrow} Q_6$, and $Q_5 \approx Q_6$.
     - By ind. hyp., $Q_6 \stackrel{\alpha}{\Rightarrow} Q_7$ and $Q_2 B^* Q_7$.
     - By def. of bis. $Q_5 \stackrel{\alpha}{\Rightarrow} Q_8$ and $Q_7 \approx Q_8$.
     - By ind. hyp., $Q_1 \stackrel{\alpha}{\Rightarrow} Q_9$ and $Q_8 B^* Q_9$.

     So $Q_2 B^* Q_7 \approx Q_8 B^* Q_9$ and by def. of B, $Q_2 B^* Q_9$.

2. If $Q \xrightarrow{\alpha} Q_4 \stackrel{\tau}{\Rightarrow} Q_2$ with $\alpha \neq \tau$.
   - By local confluence, $Q_3 \stackrel{\alpha}{\Rightarrow} Q_5$, $Q_4 \stackrel{\tau}{\Rightarrow} Q_6$, $Q_5 \approx Q_6$.
   - By ind. hyp., $Q_1 \stackrel{\alpha}{\Rightarrow} Q_7$ and $Q_5 B^* Q_7$.

   So $Q_2 \stackrel{\tau}{\Leftarrow} Q_4 \stackrel{\tau}{\Rightarrow} Q_6 \approx Q_5 B^* Q_7$. Hence $Q_2 B^* Q_7$.

Step 2

The relation B^* is a weak-bisimulation.

Proof Suppose $Q_0 B Q_1 \cdots B Q_n B Q_{n+1}$ and $Q_0 \stackrel{\alpha}{\Rightarrow} Q'_0$. Proceed by ind. on n and case analysis on $Q_n B Q_{n+1}$. By ind. hyp. we know that $Q_n \stackrel{\alpha}{\Rightarrow} Q'_n$ and $Q'_0 B^* Q'_n$.

1. If $Q_n \approx Q_{n+1}$ then $Q_{n+1} \stackrel{\alpha}{\Rightarrow} Q'_{n+1}$ and $Q'_n \approx Q'_{n+1}$. So $Q'_0 B^* Q'_n \approx Q'_{n+1}$ and we use $B^* \circ \approx \subseteq B^*$.
2. If $Q_n \stackrel{\tau}{\leftarrow} Q_{n+1}$ then $Q_{n+1} \stackrel{\alpha}{\Rightarrow} Q'_n$.
3. If $Q_n \stackrel{\tau}{\rightarrow} Q_{n+1}$ then by Step (1), $Q_{n+1} \stackrel{\alpha}{\Rightarrow} Q'_{n+1}$ and $Q'_n B^* Q'_{n+1}$.
   So $Q'_0 B^* Q'_n B^* Q'_{n+1}$ and we use $B^* \circ B^* \subseteq B^*$.

Step 3

The process P is τ-inert.

Proof By def., $\stackrel{\tau}{\rightarrow} \subseteq B^*$ and by Step (2), $B^* \subseteq \approx$.

Step 4

The processe P is confluent.

Proof By ind. on the well-founded order.

1. Suppose $Q \xrightarrow{\alpha} Q_3 \stackrel{\tau}{\Rightarrow} Q_1$ and $Q \xrightarrow{\beta} Q_4 \stackrel{\tau}{\Rightarrow} Q_2$, with $\alpha, \beta \neq \tau$.
   - By local confluence, $Q_3 \stackrel{\beta\backslash\alpha}{\Rightarrow} Q_5$, $Q_4 \stackrel{\alpha\backslash\beta}{\Rightarrow} Q_6$, and $Q_5 \approx Q_6$.
   - By Step (3), $Q_4 \approx Q_2$, and by weak bis., $Q_2 \stackrel{\alpha\backslash\beta}{\Rightarrow} Q_8$, $Q_6 \approx Q_8$.
   - By Step (3), $Q_3 \approx Q_1$, and by weak bis., $Q_1 \stackrel{\beta\backslash\alpha}{\Rightarrow} Q_7$, $Q_5 \approx Q_7$.

   So we have $Q_8 \approx Q_6 \approx Q_5 \approx Q_7$ as required.

2. Suppose $Q \xrightarrow{\tau} Q_3 \overset{\alpha}{\Rightarrow} Q_1$ and $Q \overset{\beta}{\Rightarrow} Q_2$.

   - By Step (3), $Q \approx Q_3$, and by weak bis., $Q_3 \overset{\beta}{\Rightarrow} Q_5$, $Q_2 \approx Q_5$.
   - By ind. hyp., $Q_1 \overset{\beta\backslash\alpha}{\Rightarrow} Q_6$, $Q_5 \overset{\alpha\backslash\beta}{\Rightarrow} Q_7$, and $Q_6 \approx Q_7$.
   - By weak bis., $Q_2 \overset{\alpha\backslash\beta}{\Rightarrow} Q_4$ and $Q_4 \approx Q_7$.

   So $Q_4 \approx Q_7 \approx Q_6$ as required.

Summary on confluence

1. We have 3 alternative characterisations of confluence.
2. A *confluent* process is always *τ-inert* and *determinate*.
3. *Restricted parallel composition* preserves confluence.
4. A *reactive* and *locally confluent* process is *confluent*.

A typing approach to determinacy: the case of the π-calculus

Confluence in CCS with value passing

Consider the process P

$$P = a(b).\overline{a}b$$

- It seems reasonable to regard P as *determinate*.
- However, according to a straightforward extension of the concept of confluence to CCS with values, P is *not confluent*.
- Possible relaxation: do not require confluence for distinct input actions with the same subject.

Confluence in the π-calculus

- Consider

$$P = \nu a\ (\overline{b}a \mid \overline{c}a)$$

- Again, a straightforward definition of confluence would lead us to conclude that P is *not* confluent.
- One has to take into account the fact that an output may free names bound in another output action.

This is a bit *ad hoc* and does not scale up very well.

Two approaches to determinacy

As a property of the lts

- Define *determinacy*/*confluence* at the level of the lts.
- Show that certain constructions preserve confluence of the lts.

Typing processes

- Define *well-typed processes*. Note that the typing rules provide a way to build processes.
- Restrict the attention to interactions with the environment that *respect the typing constraints*.
- Require a form of *typed equivalence* (as opposed to a type-free one).

Contrasting the two approaches (semi-formally)

Determinacy with respect to the lts A process P is *determinate* if:

$$\frac{P \stackrel{s}{\Rightarrow} P' \quad P \stackrel{s}{\Rightarrow} P''}{P' \approx P''}$$

Determinacy with respect to a type system A typable process P is *determinate* if

$$\frac{C \text{ static context} \quad C[P] \text{ typable} \quad C[P] \stackrel{\tau}{\Rightarrow} P' \quad C[P] \stackrel{\tau}{\Rightarrow} P''}{P' \approx_{\text{typed}} P''}$$

Example

- Suppose the typing system guarantees *point-to-point* interaction and consider:

$$P = a(x).a(y).\overline{b}x \mid \overline{a}3$$

- With respect to the first definition, P fails to be deterministic as:

$$P \stackrel{a2}{\Rightarrow} \overline{b}2 \quad P \stackrel{a2}{\Rightarrow} \overline{b}3$$

- However, if the environment plays by the typing rules it should *not* send a message on a.
- In other words $C[P]$, where $C = [\ ] \mid \overline{a}2$ fails to be typable.

Exercise

In the context of CCS, suppose we decide that a 'typable static context' C is defined as follows:

$$C ::= [\,] \parallel \ell.C$$

Say that a process P is *context-deterministic* (c-deterministic for short) if it satisfies:

$$\frac{C[P] \stackrel{\tau}{\Rightarrow} P_1 \quad C[P] \stackrel{\tau}{\Rightarrow} P_2}{P_1 \approx P_2}$$

1. Show that if P is c-deterministic and $P \stackrel{\tau}{\rightarrow} P'$ then $P \approx P'$.
2. Show the if P is c-deterministic and $P \stackrel{\alpha}{\rightarrow} P'$ or $P \approx P'$ then P' is c-deterministic.
3. Show that if P is c-deterministic then it is deterministic.
4. Is the converse true?

From sorting to affine resource usage

Initial goal Generalise the *sorting* system so that we keep the invariant that on every channel, at any time, at most one process can send and at most one process can receive.

Refining the goal: an example of process we want to type

$$
\begin{aligned}
P &= \nu b\ \overline{a}b.Prod(in, b) \\
Prod(in, b) &= in(c).\overline{b}c.Prod(in, b) \\
Q &= a(b).Cons(b, out) \\
Cons(b, out) &= b(c).\overline{out}c.Cons(b, out)
\end{aligned}
$$

Process P exports towards process Q a channel 'b' that will be used in output by $Prod$ and in input by $Cons$.

Affine channel usage

- Let $L = \{0, 1\}$ with a partial addition operation such that

$$x \oplus 0 = 0 \oplus x = x$$

  and $1 \oplus 1$ is *undefined*. Notation: in the following, $(X \oplus Y) \downarrow$ means that the sum is defined.

- A channel usage u is an element of L^2. At any time, a channel with usage (i, j) can be used by at most i processes to send and at most j processes to receive.

- The addition operation $\oplus$ is extended to L^2 componentwise.

- Substraction and inequlity are derived:

$$x \ominus y = z \quad \text{if } x = y \oplus z \ .$$
$$x \geq y \qquad \text{if } \exists\, z \ \ (x = y \oplus z) \ .$$

- We annotate every channel type constructor Ch with a usage. Thus the (monadic, simple) types are:

$$\sigma ::= o \parallel Ch_u(\sigma)$$

- The addition operation $\oplus$ is extended to types so that:

$$o \oplus o = o, \quad Ch_{u_1}(\sigma) \oplus Ch_{u_2}(\sigma) = Ch_{u_1 \oplus u_2}(\sigma)$$

  (the sum being undefined otherwise).

- Similarly, for subtraction and inequality.

Type judgments

- As usual, a *typing context* has the shape:

$$\Gamma = a_1 : \sigma_1, \ldots, a : \sigma_n$$

- The sum of contexts $\Gamma_1 \oplus \Gamma_2$ is defined if

$$a : \sigma_1 \in \Gamma_1, a : \sigma_2 \in \Gamma_2 \quad \Rightarrow \quad (\sigma_1 \oplus \sigma_2) \downarrow$$

- Type judment for channel names:

$$\frac{\sigma \geq \sigma'}{\Gamma, a : \sigma \vdash a : \sigma'}$$

Type judgments (continued)

- Consider the following processes:

$$P ::= 0 \parallel a(b).P \parallel \overline{a}b.P \parallel \nu a : \sigma \; P \parallel (P \mid P) \parallel A(\mathbf{a})$$

  We assume that generated names and process identifiers carry a type annotation ($\nu a : \sigma$ and $A : (\sigma_1, \ldots, \sigma_n)$, respectively).

- The type judgment for processes is:

$$\Gamma \vdash P$$

- The typing rules are:

$$\frac{}{\Gamma \vdash 0} \qquad \frac{\Gamma_i \vdash P_i \quad i = 1, 2}{(\Gamma_1 \oplus \Gamma_2) \vdash (P_1 \mid P_2)}$$

$$\frac{\Gamma, a : \sigma \vdash P}{\Gamma \vdash \nu a : \sigma \; P} \qquad \frac{\begin{array}{c} A : (\sigma_1, \ldots, \sigma_n) \\ \Gamma_i \vdash b_i : \sigma_i, i = 1, \ldots, n \end{array}}{(\Gamma_1 \oplus \cdots \oplus \Gamma_n) \vdash A(b_1, \ldots, b_n)}$$

$$\frac{\begin{array}{c} \Gamma \vdash a : Ch_u(\sigma) \\ \pi_2(u) = 1 \quad \Gamma, b : \sigma \vdash P \end{array}}{\Gamma \vdash a(b).P} \qquad \frac{\begin{array}{c} \Gamma_1 \vdash a : Ch_u(\sigma) \\ \pi_1(u) = 1 \quad \Gamma_2 \vdash b : \sigma \quad \Gamma_1 \vdash P \end{array}}{(\Gamma_1 \oplus \Gamma_2) \vdash \overline{a}b.P}$$

Typing the motivating example

$$
\begin{array}{ll}
P & = \nu b{:}\ Ch_{(1,1)}(o)\ \overline{a}b.Prod(in, b) \\
Prod(in, b) & = in(c).\overline{b}c.Prod(in, b) \\
Q & = a(b).Cons(b, out) \\
Cons(b, out) & = b(c).\overline{out}c.Cons(b, out)
\end{array}
$$

$$
\begin{array}{ll}
\Gamma_P & = a : Ch_{(1,0)}(Ch_{(0,1)}(o)), in : Ch_{(0,1)}(o) \\
\Gamma_Q & = a : Ch_{(0,1)}(Ch_{(0,1)}(o)), out : Ch_{(1,0)}(o) \\
Prod & : (Ch_{(0,1)}(o), Ch_{(1,0)}(o)) \\
Cons & : (Ch_{(0,1)}(o), Ch_{(1,0)}(o))
\end{array}
$$

Remarks

- We can split the usages of a channel (input/output).
- When we send, we consume the usage of the name we send.
- When we receive, we assume the usage of the name we receive.
- It is crucial that a name which is sent is received at most once, otherwise the linearity information is lost.

Exercise

Suppose we have a situation where we import a channel on which *Prod* and *Cons* will start interacting:

$$\nu b : Ch_{(1,1)}(o)\ \overline{a}b.0 \mid a(b).(Prod(in, b) \mid Cons(b, out))$$

Is there Γ such that $\Gamma \vdash P$?

Exercise

Suppose we emit *twice* a fresh name b as in the following process:

$$P = \nu b : Ch_{(1,1)}(o)\ \overline{a}b.0 \mid \overline{a'}b.0\ .$$

Is there a Γ such that $\Gamma \vdash P$?

Exercise

Suppose $\Gamma \vdash P$ and $P \xrightarrow{\tau} P_i$ for $i = 1, 2$ and suppose that the two τ transitions are generated by two distinct synchronisations.

- Explain why the transitions do not 'superpose' and conclude that either $P_1 = P_2$ or $\exists\, Q\ \ (P_1 \xrightarrow{\tau} Q, P_2 \xrightarrow{\tau} Q)$.
- Is this enough to conclude that typable processes are confluent with respect to τ transitions?

Formal properties (outline)

A list of definitions and properties to show that *typable processes* enjoy a *strong confluence* property with respect to *typable transitions*.

1. Weakening.
2. Substitution.
3. Actions compatible with a given context.
4. Typed transitions.
5. Subject reduction.
6. Typed equivalence and strong confluence of typed transitions.

Weakening

$$\frac{\Gamma \vdash P \quad (\Gamma \oplus \Gamma') \downarrow}{(\Gamma \oplus \Gamma') \vdash P}$$

Exercise Prove this by induction on $\Gamma \vdash P$. For instance, consider the case where $(\Gamma_1 \oplus \Gamma_2) \vdash \overline{a}b.P$.

Substitution

$$\frac{\Gamma, a : \sigma \vdash P \quad \Gamma' \vdash b : \sigma \quad (\Gamma \oplus \Gamma') \downarrow}{(\Gamma \oplus \Gamma') \vdash [b/a]P}$$

Exercise Prove this by induction on $\Gamma, a : \sigma \vdash P$. For instance, consider the case where $\Gamma, a : Ch_u(\sigma) \vdash a(c).P$.

Actions compatible with a given context

We define P_α as a 'minimal' environment that allows the action α to happen:

$$P_\alpha = \begin{cases} 0 & \text{if } \alpha = \tau \\ a(b).0 & \text{if } \alpha = \overline{a}b \text{ or } \alpha = \nu b \overline{a} b \\ \overline{a}b.0 & \text{if } \alpha = ab \end{cases}$$

Then we use this to define when an action is compatible with a given typing context:

$$(\Gamma, \alpha) \downarrow \quad \text{if } \exists\, \Gamma' \ \ (\ \Gamma' \vdash P_\alpha \text{ and } (\Gamma \oplus \Gamma') \downarrow\ )$$

Exercise

Suppose $\Gamma = a : Ch_{(1,0)}(Ch_{(0,1)}(o)), b : Ch_{(1,0)}(Ch_{(0,1)}(o))$. Prove or disprove the following:

1. $(\Gamma, \overline{a}c) \downarrow$
2. $(\Gamma, ac) \downarrow$.

Typed transitions

We introduce a notion of 'typed' transition.

$$P \overset{\Gamma,\alpha}{\to} P' \text{ if } \quad P \overset{\alpha}{\to} P', \quad \Gamma \vdash P, \text{ and } \quad (\Gamma, \alpha) \downarrow$$

Subject reduction

Suppose $(\Gamma, \alpha)\downarrow$. We define the residual $\Gamma(\alpha)$ of Γ after the action α:

$$\Gamma(\alpha) = \begin{cases} \Gamma & \text{if } \alpha = \tau \\ \Gamma \oplus (b:\sigma) & \text{if } \alpha = ab, \Gamma \vdash a : Ch_u(\sigma) \\ \Gamma \ominus (b:\sigma) & \text{if } \alpha = \overline{a}b, \Gamma \vdash a : Ch_u(\sigma) \\ (\Gamma, b:\sigma') \ominus (b:\sigma) & \text{if } \alpha = \nu b:\sigma'\overline{a}b, \Gamma \vdash a : Ch_u(\sigma) \end{cases}$$

Subject reduction (continued)

Typing is preserved by typed transitions.

$$\frac{P \overset{\Gamma,\alpha}{\to} P'}{\Gamma(\alpha) \vdash P'}$$

Exercise Prove this by induction on $P \overset{\alpha}{\to} P'$. For instance, consider the case where $P \overset{\Gamma,\tau}{\to} P'$ by a synchronisation action.

Typed bisimulation

- We can define a notion of *typed equivalence* as a *family* of symmetric relations $\{S_\Gamma \mid \Gamma \text{ context}\}$ such that:
  1. $(P, Q) \in S_\Gamma$ implies that P and Q are typable in the context Γ, and
  2. a *typed* transition of P with respect to the context Γ is matched by a typed transition of Q in the usual way.
- With respect to this notion of typed equivalence we would like to show that $P \overset{\Gamma,\tau}{\longrightarrow} Q$ implies $P \approx_\Gamma Q$.

NB We omit the formal development of this part because it is a bit technical and conceptually quite close to the one for the 'type-free' case.

Key commutation property

Typed transitions enjoy a *strong* confluence property.

$$\frac{P \overset{\Gamma,\tau}{\rightarrow} Q \quad P \overset{\Gamma,\alpha}{\rightarrow} P'}{\exists\, P' \;\; (\; P' \overset{\Gamma(\alpha),\tau}{\rightarrow\!\!=} P' \quad Q \overset{\Gamma,\alpha}{\rightarrow\!\!=} P' \;)}$$

NB Strong confluence of τ transitions is a special case.

References and historical remarks

- The notions of determinacy and confluence presented are based on chapter 11 of:

  Robin Milner. Communication and Concurrency, Prentice-Hall, 1989.

- Amazingly, this book does not refer to Kahn networks which were introduced in:

  Gilles Kahn. The semantics of a simple language for parallel programming, IFIP Conf. on Information Processing 74, North-Holland, 1974.

  Incidentally, synchronous data flow languages such as LUSTRE can be regarded as a refinement of this model.

- A rather complete study of the notion of confluence in the more general framework of the π-calculus is in:

  Anna Philippou, David Walker. On confluence in the pi-Calculus. ICALP 1997: 314-324. (See also Anna Philippou PhD thesis, University of Warwick 1996).

- This builds on the PhD thesis of Sanderson and Tofts (in Edinburgh in the early 90's) where notions of confluence for CCS with value passing were proposed.

- The presented generalisation of Newman's lemma is due to

  J. Groote, M. Sellink. Confluence for process verification. Theor. Comput. Sci. 170(1-2):47-81, 1996.

- The classical reference for Linear Logic is:

  J.-Y. Girard. Linear Logic. Theoretical Computer Science, 50(1), 1987.

  Several up-to-date tutorials on this topic are available (Curien, Danos-Di Cosmo,...)

- This work has influenced a number of works on the static analysis of programs which exploit the notion of linearity. An early example in the framework of the π-calculus is:

  Naoki Kobayashi, Benjamin C. Pierce, David N. Turner. Linearity and the pi-calculus. ACM Transactions on Programming Languages and Systems (TOPLAS), 21(5), 1999.

- The type system discussed here is a bit different in that:
  - the usage of resources in the typing is affine (at most once) rather than linear (exactly once).
  - we maintain the invariant:

    At any time at most one process can send (receive) on a channel.

    rather than:

    a channel is used at most once to send (receive).

    (which is more in line with the sorting system we have previously considered).

- Beware that going from logic to programming some properties are lost. For instance, intuititionistic logic inspires ML type systems. However in ML, all types are inhabited and programs do not always normalise!

Exercise (revision)

Suppose P is a CCS process that is reactive and such that for every derivative Q of P we have:

$$\frac{Q \xrightarrow{\tau} Q_1 \qquad Q \xrightarrow{\tau} Q_2}{Q_1 \approx Q_2}$$

Show that this implies that for every derivative Q of P we have:

$$\frac{Q \stackrel{\tau}{\Rightarrow} Q_1 \quad \text{and} \quad Q \stackrel{\tau}{\Rightarrow} Q_2}{\exists\, Q_1', Q_2' \;\; (\; Q_1 \stackrel{\tau}{\Rightarrow} Q_1', \quad Q_2 \stackrel{\tau}{\Rightarrow} Q_2', \quad \text{and} \quad Q_1' \approx Q_2' \;)}$$

Exercise (revision)

We consider a *linear* variant of the typing system.

A *type* is *neutral* if it is either o or $Ch_{(0,0)}(\sigma)$. Thus if σ is neutral and $\sigma \oplus \sigma'$ is defined then $\sigma \oplus \sigma' = \sigma'$.

A *context* is *neutral* if it contains only neutral types.

The rule for typing names is:

$$\frac{\Gamma \text{ neutral}}{\Gamma, a : \sigma \vdash a : \sigma}$$

The rules for (some of) the processes are:

$$\frac{\Gamma \text{ neutral}}{\Gamma \vdash 0} \qquad \frac{\Gamma_i \vdash P_i \quad i = 1, 2}{(\Gamma_1 \oplus \Gamma_2) \vdash (P_1 \mid P_2)}$$

$$\frac{\Gamma_1 \vdash a : Ch_{(0,1)}(\sigma) \quad \Gamma_2, b : \sigma \vdash P}{\Gamma_1 \oplus \Gamma_2 \vdash a(b).P} \qquad \frac{\Gamma_1 \vdash a : Ch_{(1,0)}(\sigma) \quad \Gamma_2 \vdash b : \sigma \quad \Gamma_3 \vdash P}{(\Gamma_1 \oplus \Gamma_2 \oplus \Gamma_3) \vdash \overline{a}b.P}$$

For this system, prove the following versions of weakening and substitution:

$$\frac{\Gamma \vdash P \quad \Gamma' \text{ neutral} \quad (\Gamma \oplus \Gamma') \downarrow}{(\Gamma \oplus \Gamma') \vdash P} \qquad \frac{\Gamma, a : \sigma \vdash P \quad \Gamma' \vdash b : \sigma \quad (\Gamma \oplus \Gamma') \downarrow}{(\Gamma \oplus \Gamma') \vdash [b/a]P}$$

What's wrong with the weakening rule for the affine system?