SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

CONCURRENT SYSTEMS
LECTURE 12

Prof. Daniele Gorla

CCS

- Up to now, we have considered non-deterministic processes
- Two main features are missing for modeling a concurrent system:
  - Simultaneous execution of proc's
  - Interprocess interaction
- Solutions adopted:
  - Parallel composition, with interleaving semantics
  - Producer/consumer paradigm

Given a set of names N (that denote events)

- a ($\in N$) denotes consumption of event a
- $\bar{a}$ (for $a \in N$) denotes production of event a
- a and $\bar{a}$ are complementary actions: they let two parallel processes synchronize on the event a

When two processes synchronize, an external observer has no way of understanding what is happening in the system

→ synchronization is not observable from the outside; it produces a special 'silent' action, that we denote with τ

The set of actions we shall consider is: $\mathcal{A} = \mathcal{N} \cup \overline{\mathcal{N}} \cup \{\tau\}$

It is also useful to force some processes of the system to synchronize between them (without the possibility of showing to the outside some actions)

The restriction operator P\a restricts the scope of name a to process P
(a is visible only from within P)

This is similar to local variables in a procedure of an imperative program

The set of CCS processes is defined by the following grammar:

$$P \ ::= \ \sum_{i \in I} \alpha_i.P_i \mid A(a_1 \dots a_n) \mid P|Q \mid P\backslash a$$

where I is a finite index set and $\alpha_i \in \mathcal{A}$, for all $i \in I$.

Inference rules

$$\sum_{i\in I} \alpha_i.P_i \xrightarrow{\alpha_j} P_j \quad \textit{for all } j \in I$$

$$\frac{P\{a_1/x_n \ldots a_n/x_n\} \xrightarrow{\alpha} P'}{A(a_1 \ldots a_n) \xrightarrow{\alpha} P'} \; A(x_1 \ldots x_n) \triangleq P$$

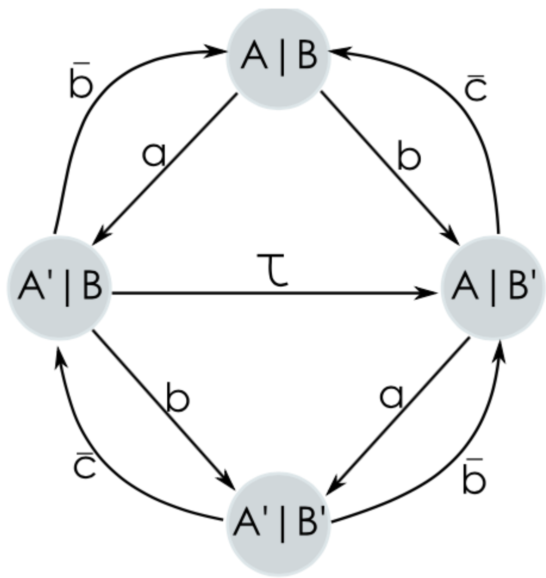
$$\frac{P \xrightarrow{\alpha} P'}{P\backslash a \xrightarrow{\alpha} P'\backslash a} \; \alpha \notin \{a, \bar{a}\}$$

$$\frac{P_1 \xrightarrow{\alpha} P_1'}{P_1|P_2 \xrightarrow{\alpha} P_1'|P_2} \qquad \frac{P_2 \xrightarrow{\alpha} P_2'}{P_1|P_2 \xrightarrow{\alpha} P_1|P_2'}$$

$$\frac{P_1 \xrightarrow{a} P_1' \quad P_2 \xrightarrow{\bar{a}} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'} \qquad \frac{P_1 \xrightarrow{\bar{a}} P_1' \quad P_2 \xrightarrow{a} P_2'}{P_1|P_2 \xrightarrow{\tau} P_1'|P_2'}$$

EXAMPLE

- $A \triangleq a.A'$
- $A' \triangleq \bar{b}.A$
- $B \triangleq b.B'$
- $B' \triangleq \bar{c}.B$



$$\dfrac{\dfrac{a.A' \xrightarrow{a} A'}{A \xrightarrow{a} A'}\, A \triangleq a.A'}{A|B \xrightarrow{a} A'|B}$$

$$\dfrac{\dfrac{b.B' \xrightarrow{b} B'}{B \xrightarrow{b} B'}\, B \triangleq b.B'}{A|B \xrightarrow{b} A|B'}$$

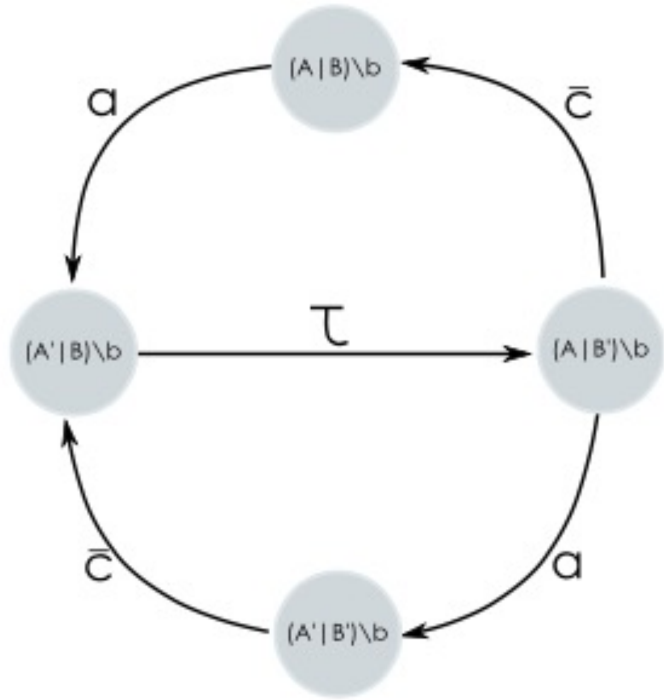
$$\dfrac{\dfrac{\bar{b}.A \xrightarrow{\bar{b}} A}{A' \xrightarrow{\bar{b}} A}\, A' \triangleq \bar{b}.A \qquad \dfrac{b.B' \xrightarrow{b} B'}{B \xrightarrow{b} B'}\, B \triangleq b.B'}{A'|B \xrightarrow{\tau} A|B'}$$

In the construction of the LTS we loose the consciousness of the parallel

→ It is indeed possible, by having the new set of actions, to obtain the previous LTS through the syntax we considered last class

The usefulness of the parallel is two-fold:

- it is the fundamental operator in concurrency theory
- it allows for a compact and intuitive writing of processes.



EXAMPLE (cont'd): (A|B)\b

$$\cfrac{\cfrac{\cfrac{a.A' \xrightarrow{a} A'}{A \xrightarrow{a} A'}\, A \triangleq a.A'}{A|B \xrightarrow{a} A'|B}}{(A|B)\backslash b \xrightarrow{a} (A'|B)\backslash b}\, a \notin \{b, \bar{b}\}$$

$$\cfrac{\cfrac{\cfrac{b.B' \xrightarrow{b} B'}{B \xrightarrow{b} B'}\, B \triangleq b.B'}{A|B \xrightarrow{b} A|B'}}{(A|B)\backslash b}\, b \notin \{b, \bar{b}\}$$

The effect of the restriction on b is that we have deleted the transitions involving b

→ hide all transitions labelled with b and $\bar{b}$

Notice that the τ, even if it has been generated by synchronizing on b, it is still present after applying the restriction on b

→ the purpose of the τ is exactly to signal that a synchronization has happened but to hide the event on which the involved processed synchronized

In general, it is possible that whole states disappear upon restriction of some names: this would be the case, e.g., if we consider the LTS arising from (A' | B)\a,b:

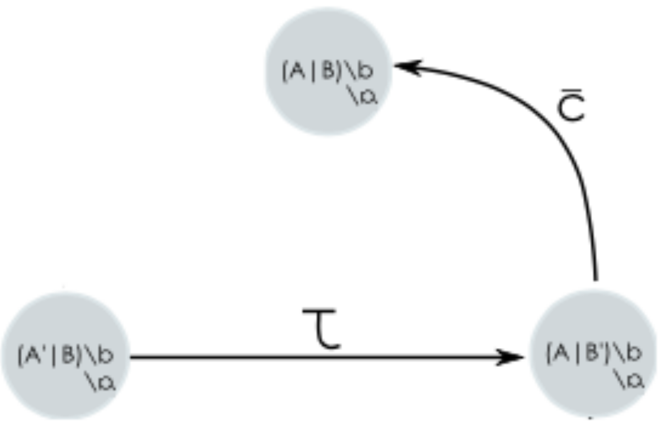


Image Finiteness

Theorem 3.1. $|\{P' : \exists \alpha . P \xrightarrow{\alpha} P'\}| < \infty$.

Proof. For every process P, let $\mathcal{T}(P)$ be the set of derivation trees for every possible transition $P \xrightarrow{\alpha} P'$, for every α and P'. Let us denote with k_P the maximum height of a tree in $\mathcal{T}(P)$. By induction on k_P, let us show that $\{P' : \exists \alpha . P \xrightarrow{\alpha} P'\}$ is finite.

Base case ($k_P = 0$):

In this case, it must be that $P \triangleq \sum_{i \in I} \alpha_i . P_i$. Then, we have that

$$|\{P' : \exists \alpha . P \xrightarrow{\alpha} P'\}| = |\{P_i : i \in I\}| \leq |I| < \infty$$

Inductive step:

We have to analyze three cases, according to the outmost operator in P.

1. $P \triangleq Q \backslash a$. All inferences for a reduction from P must have as last rule the one dealing with restriction, i.e.:

$$\frac{Q \xrightarrow{\alpha} Q'}{Q \backslash a \xrightarrow{\alpha} Q' \backslash a} \alpha \notin \{a, \bar{a}\}$$

   for every possible inference starting from process Q.

Since $k_Q < k_P$, we can use the induction hypothesis and obtain that

$$\mid \{Q' : \exists \alpha . Q \xrightarrow{\alpha} Q'\} \mid \; < \infty$$

the thesis holds by observing that

$$\{P' : \exists \alpha . P \xrightarrow{\alpha} P'\} \subseteq \{Q' \backslash a : \exists \alpha . Q \xrightarrow{\alpha} Q'\}$$

2. $P \triangleq A(a_1 \ldots a_n)$. In this case, all inferences of P will be of the form

$$\frac{Q\{a_1/x_1 \ldots a_n/x_n\} \xrightarrow{\alpha} Q'}{A(a_1 \ldots a_n) \xrightarrow{\alpha} Q'} \; A(x_1 \ldots x_n) \triangleq Q$$

where the inference trees for $Q\{a_1/x_1 \ldots a_n/x_n\}$ have height lower than k_P. By induction,

$$\mid \{R : \exists \alpha . Q\{a_1/x_1 \ldots a_n/x_n\} \xrightarrow{\alpha} R\} \mid \; < \infty$$

The thesis holds by observing that

$$\{P' : \exists \alpha . P \xrightarrow{\alpha} P'\} = \{R : \exists \alpha . Q\{a_1/x_1 \ldots a_n/x_n\} \xrightarrow{\alpha} R\}$$

3. $P \triangleq P_1 \mid P_2$. In this case, notice that:

$$\begin{aligned}\{P' : \exists \alpha . P \xrightarrow{\alpha} P'\} = \quad & \{P_1'|P_2 : \exists \alpha . P_1 \xrightarrow{\alpha} P_1'\} \\ & \cup\{P_1|P_2' : \exists \alpha . P_2 \xrightarrow{\alpha} P_2'\} \\ & \cup\{P_1'|P_2' : \exists a . P_1 \xrightarrow{a} P_1' \wedge P_2 \xrightarrow{\bar{a}} P_2'\} \\ & \cup\{P_1'|P_2' : \exists a . P_1 \xrightarrow{\bar{a}} P_1' \wedge P_2 \xrightarrow{a} P_2'\}\end{aligned}$$

The required cardinality is thus at most the sum of the cardinalities of the four sets depicted above. By induction, we have that every such a set has a finite cardinality. This easily allows us to conclude. $\square$

Renamings

Let us now consider a *renaming*, i.e. a function $\sigma : \mathcal{N} \to \mathcal{N}$.

By definition, we let $\sigma(\bar{a})$ to be $\overline{\sigma(a)}$ and $\sigma(\tau)$ be τ itself.

We now define the result of applying a renaming σ to a process P:

$$\begin{array}{rcl}
\sigma(\sum_{i \in I} \alpha_i.P_i) & = & \sum_{i \in I} \sigma(\alpha_i).\sigma(P_i) \\
\sigma(P \backslash a) & = & \sigma(P) \backslash \sigma(a) \\
\sigma(P_1 \mid P_2) & = & \sigma(P_1) \mid \sigma(P_2) \\
\sigma(A(a_1, \ldots, a_n)) & = & A^\sigma(\sigma(a_1), \ldots, \sigma(a_n))
\end{array}$$

where, for every process definition $A(x_1, \ldots, x_n) \triangleq P$, we assume the new process definition

$$A^\sigma(\sigma(x_1), \ldots, \sigma(x_n)) \triangleq \sigma(P)$$

Theorem 3.2. *For every injective renaming $\sigma : \mathcal{N} \to \mathcal{N}$, if $P \xrightarrow{\alpha} P'$ then $\sigma(P) \xrightarrow{\sigma(\alpha)} \sigma(P')$.*

Proof. The proof is by induction on the height of the inference for $P \xrightarrow{\alpha} P'$.

Base case: The only possible such case is with a sum, i.e. $P = \sum_{i \in I} \alpha_i.P_i \xrightarrow{\alpha_j} P_j$.

In this case, $\alpha = \alpha_j$ and $P' = P_j$, for some $j \in I$.

We now have that $\sigma(P) = \sum_{i \in I} \sigma(\alpha_i).\sigma(P_i)$

and $\sigma(P) \xrightarrow{\sigma(\alpha_j)} \sigma(P_j)\ \ \forall j \in I$.

Inductive step:
We have to consider three possible cases:

1. $P \triangleq A(a_1 \dots a_n)$, $A(x_1 \dots x_n) \triangleq Q$ and $Q\{a_1/x_1 \dots a_n/x_n\} \xrightarrow{\alpha} P'$.

   By definition of renaming, we have that $\sigma(P) = A^\sigma(\sigma(a_1) \dots \sigma(a_n))$.

   By inductive hypothesis on Q, we have that

$$\sigma(Q)\{\sigma(a_1)/\sigma(x_1) \dots \sigma(a_n)/\sigma(x_n)\} \xrightarrow{\sigma(\alpha)} \sigma(P')$$

   and we conclude.

2. $P \stackrel{\triangle}{=} Q\backslash a$, $Q \xrightarrow{\alpha} Q'$ and $\alpha \notin \{a, \bar{a}\}$.

   By definition of renaming, $\sigma(Q\backslash a) = \sigma(Q)\backslash\sigma(a)$

   By induction, $\sigma(Q) \xrightarrow{\sigma(\alpha)} \sigma(Q')$

   Since $\sigma(\alpha) \notin \{\sigma(a), \overline{\sigma(a)}\}$, we have the desired $\sigma(P) \xrightarrow{\sigma(\alpha)} \sigma(P')$

   Notice that here injectiveness of σ is crucial.

   For example, let $Q = b.\mathbf{0}$ and $\sigma(a) = \sigma(b) = a$; then, $P \xrightarrow{b} \mathbf{0}\backslash a$ whereas $\sigma(P) \stackrel{\sigma(b)}{\not\rightarrow} \mathbf{0}\backslash\sigma(a)$.

3. $P \stackrel{\triangle}{=} P_1 \mid P_2$. EXERCISE $\square$

Restrictions

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Prop.: a.P\a ~ **0**

Proof.

S = {(a.P\a , **0**)} is a bisimulation

Which challenges can (a.P)\a have?

- a.P can only perform a (and become P)
- however, because of restriction, a.P\a is stuck

No challenge from a.P\a, nor from 0 → bisimilar!

QED

Prop.: $\bar{a}$.P\a ~ **0**

Proof.

Similar.

Idempotency of Sum

Prop.: $\alpha.P+\alpha.P+M \sim \alpha.P+M$, where M denotes a sum $\Sigma_{i\in I}\ \beta i.Pi$

Proof.

$$S = \{ (\alpha.P+\alpha.P+M\ ,\ \alpha.P+M) \}$$

Is it a bisimulation?

NO: the problem is that, for example:

- $\alpha.P+\alpha.P+M\ \ -\alpha\!\!\to\ \ P$
- $\alpha.P+M\ \ -\alpha\!\!\to\ \ P$
- BUT (P,P) in general does NOT belong to S!

So, we can try with

$$S = \{ (\alpha.P+\alpha.P+M\ ,\ \alpha.P+M) \} \cup \{(P,P)\}$$

Is it a bisimulation?

NOT YET: $P -\beta\!\!\to P'$ (challenge and reply), but (P',P') is not in S

So, we try with

$$S = \{ (\alpha.P+\alpha.P+M\ ,\ \alpha.P+M) \} \cup Id$$

This is a bisimulation (try to prove!) and contains the desired pair.

QED

EXAMPLE: Semaphores

An n-ary semaphore $S^{(n)}(p,v)$ is a process used to ensure that there are no more than n istances of the same activity concurrently in execution.

An activity is started by action p and is terminated by action v.

The specification of a unary semaphore is the following:

$$
\begin{array}{lcl}
S^{(1)} & \triangleq & p \cdot S_1^{(1)} \\
S_1^{(1)} & \triangleq & v \cdot S^{(1)}
\end{array}
$$

The specification of a binary semaphore is the following:

$$
\begin{array}{lcl}
S^{(2)} & \triangleq & p \cdot S_1^{(2)} \\
S_1^{(2)} & \triangleq & p \cdot S_2^{(2)} + v \cdot S^{(2)} \\
S_2^{(2)} & \triangleq & v \cdot S_1^{(2)}
\end{array}
$$

If we consider $S^{(2)}$ as the specification of the expected behavior of a binary semaphore and $S^{(1)} \mid S^{(1)}$ as its concrete implementation, we can show that

$$S^{(1)} \mid S^{(1)} \sim S^{(2)}$$

This means that the implementation and the specification do coincide

To show this equivalence, it suffices to show that relation

$$\begin{aligned} R = \{ \quad & (S^{(1)} \mid S^{(1)}\,,\, S^{(2)})\,,\, (S_1^{(1)} \mid S^{(1)}\,,\, S_1^{(2)})\,, \\ & (S^{(1)} \mid S_1^{(1)}\,,\, S_1^{(2)})\,,\, (S_1^{(1)} \mid S_1^{(1)}\,,\, S_2^{(2)}) \quad \} \end{aligned}$$

is a bisimulation

Congruence

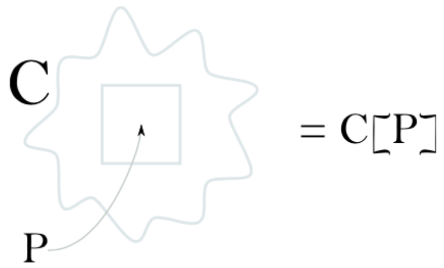
SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

One of the main aims of an equivalence notion between processes is to make equational reasonings of the kind: “if P and Q are equivalent, then they can be interchangeably used in any execution context”

This feature on an equivalence makes it a *congruence*

Not all equivalences are necessarily congruences (even though most of them are)

To properly define a congruence, we first need to define an execution context, and then what it means to run a process in a context. Intuitively:



where C is a context (i.e., a process with a hole ☐), P is a process, and C[P] denotes filling the hole with P

Example: if C = (☐ | Q)\a, then C[P] = (P | Q)\a

The set $\mathcal{C}$ of CCS contexts is given by the following grammar:

$$C \ ::= \ \Box \ | \ C|P \ | \ C\backslash a \ | \ \alpha.C + M$$

where M denotes a sum.

An equivalence relation $\Re$ is a congruence if and only if

$$\forall (P,Q) \in \Re,\ \forall C.(C[P], C[Q]) \in \Re$$

Theorem 3.7. *If $P \sim Q$ then $\forall C.\ C[P] \sim C[Q]$.*

Lemma 3.4. *If $P \sim Q$ then $\alpha.P + M \sim \alpha.Q + M$, for all α and M.*

Proof. It is sufficient to show that

$$S = \{(\alpha.P + M, \alpha.Q + M) : P \sim Q, \alpha \in \mathcal{A}, M = \sum_{i \in I} \alpha_i.P_i\} \ \cup \ \sim$$

is a simulation.

Let $(\alpha.P + M, \alpha.Q + M) \in S$ $\qquad \alpha.P + M \xrightarrow{\beta} P'.$ $\qquad M = \sum_{i \in I} \alpha_i.P_i$

1. $\beta = \alpha$ and $P' = P$

$\alpha.Q + M$ can reply with action α and become Q by hypothesis, is bisimilar to P

2. $\beta = \alpha_j$, for some $j \in I$, and $P' = P_j$

$\alpha.Q + M$ can reply with the same action and by reducing to the same process.

Lemma 3.5. *If $P \sim Q$ then $P\backslash a \sim Q\backslash a$, for all a.*

Proof. Also in this case, let us show that

$$S = \{(P\backslash a, Q\backslash a) : \forall P \sim Q, \forall a \in \mathcal{N}\}$$

is a simulation

Let $(P\backslash a, Q\backslash a) \in S$ and $P\backslash a \xrightarrow{\alpha} P'$.

$P' = P''\backslash a$, for $P \xrightarrow{\alpha} P''$ and $\alpha \notin \{a, \bar{a}\}$.

process Q can perform the same action α and reduce to process $Q' = Q''\backslash a$. Since $P \sim Q$, we have that $P'' \sim Q''$

Lemma 3.6. *If $P \sim Q$ then $P|R \sim Q|R$, for all R.*

Proof. Let us show that

$$S = \{(P|R, Q|R) \ : \ P \sim Q\}$$

is a simulation. Let $(P|R, Q|R) \in S$ and $P|R \xrightarrow{\alpha} P'$.

1. $P \xrightarrow{\alpha} P''$ and $P' = P''|R$:

   Since $P \sim Q$, we have that $Q \xrightarrow{\alpha} Q''$ and $P'' \sim Q''$; hence, $Q|R \xrightarrow{\alpha} Q''|R$ and $(P''|R, Q''|R) \in S$.

2. $R \xrightarrow{\alpha} R'$ and $P' = P|R'$:

   Trivially, $Q|R \xrightarrow{\alpha} Q|R'$ and $(P|R', Q|R') \in S$.

3. $P \xrightarrow{a} P'' \wedge R \xrightarrow{\bar{a}} R''$ (or viceversa) and $P' = P''|R''$, with $\alpha = \tau$.

   Since $P \sim Q$, we have that $Q \xrightarrow{a} Q''$ and $Q'' \sim P''$
   Hence, $Q|R \xrightarrow{\tau} Q''|R'' = Q'$ and $(P', Q') \in S$.

Theorem 3.7. *If* $P \sim Q$ *then* $\forall C.\ C[P] \sim C[Q]$.

Proof. By induction on the structure of C.

Base case: the only possible context to analyze is $C = \square$.

we have that $C[P] = P$, for all P. Hence, $C[P] \sim C[Q]$ by hypothesis.

Inductive step: Let us reason on the possible structure of C and exploit the previous lemmata.

For example, if $C = R|C'$, for some other context C', then $C[P] = R|C'[P]$ and $C[Q] = R|C'[Q]$.

By induction (since C' is smaller than C), we have that $C'[P] \sim C'[Q]$

by Lemma 3.6, we obtain that $R|C'[P] \sim R|C'[Q]$