Processi Concorrenti

- Fino adesso abbiamo visto solo processi sequenziali. Adesso vedremo come due o più processi possano evolvere **concorrentemente**, possibilmente **interagendo** tra loro, **sincronizzandosi** attraverso canali comuni. Cioè introdurremo nel nostro piccolo linguaggio i processi concorrenti.

- Inoltre, distingueremo le azioni in due categorie: quelle **osservabili dall'esterno** e quelle **interne** al sistema e quindi non osservabili.

- Se pensiamo a sistemi come a delle scatole con pulsanti, le azioni esterne corrispondono alla pressione dei pulsanti, mentre quelle interne (come dice la parola stessa) avvengono all'interno della scatola e quindi non sono osservabili.

- Nel seguito, azioni del tipo b o $\overline{b}$ verranno chiamate **azioni esterne** o **osservabili**. Esse rappresentano il tentativo da parte del sistema di sincronizzarsi con l'esterno attraverso il **canale** (o **porta**) b. L'osservabilità deriva dal fatto che l'azione b, eseguita da una componente del sistema, può essere osservata dall'esterno interagendo con essa, cioè fornendo l'azione complementare $\overline{b}$ nel tentativo di sincronizzarsi, e viceversa.

- Le **azioni interne** ad un sistema **non sono osservabili** e modellano la sincronizzazione su un certo canale da parte di due componenti interne ad un sistema. Non vi è alcuna interazione con l'esterno. Ecco perchè vengono dette non osservabili. Dal punto di vista di un osservatore esterno le azioni interne sono tutte uguali tra loro e quindi non le distingueremo attraverso nomi a, b, c, ... ma le rappresenteremo con un **unico simbolo speciale** τ. Essendo non osservabile, l'azione τ non ha un'azione complementare.

- Così adesso l'insieme di azioni possibili è dato da $Act \stackrel{\text{def}}{=} \mathcal{L} \cup \{\tau\}$.

- Per rappresentare la **struttura** di un sistema, cioè il modo in cui le componenti sono **collegate** tra loro, useremo dei **grafi di flusso** (vedi esempi a pag. 27). Da non confondere con i **grafi di transizioni.**

Definizione 14 *La* **sintassi** *dell'insieme $\mathcal{P}$ dei* **processi concorrenti** *è data attraverso la seguente grammatica:*

$$P ::= A\langle a_1, \dots a_n \rangle \quad | \quad \sum_{i \in I} \alpha_i.P_i \quad | \quad P_1 \mid P_2 \quad | \quad (\nu a)P$$

dove I è un insieme finito di indici ed A è definito da una equazione del tipo $A(\vec{x}) \stackrel{def}{=} P_A$ in cui P_A è un processo somma.

- $A\langle a_1, \dots a_n \rangle$ viene chiamato **processo costante**
- $\sum_{i \in I} \alpha_i.P_i$ è la **somma** o **scelta non deterministica**
- $P_1 \mid P_2$ è la **composizione parallela**
- $(\nu a)P$ si chiama **restrizione** o **new**.
  - $(\nu a)P$ è una abbreviazione per $(\mathbf{new}\, a)P$, che può essere visto come una **dichiarazione** di un nuovo nome di canale a (ovvero un indirizzo vergine). Lo **scope** della dichiarazione è il processo P. Il nome a è **legato** in $(\nu a)P$: al di fuori di P "quel" nome a non ha senso.

- **Domanda:** è possibile che "quel" nome a non abbia senso neppure in P pur apparendo occorrenze di a?

- **Abbreviazioni** (i) abbrevieremo suffissi del tipo $a.b.\mathbf{0}$ con $a.b$; (ii) a volte scriveremo $(\nu ab)P$ al posto di $(\nu a)(\nu b)P$.

Precedenza degli operatori

Gli operatori unari hanno la precedenza rispetto agli operatori binari e la composizione parallela ha la precedenza rispetto alla somma:

$$\begin{array}{rcl}
(\nu a)P \mid Q & = & ((\nu a)P) \mid Q \\
a.P + M & = & (a.P) + M \\
\\
P \mid Q + R & = & (P \mid Q) + R \\
P + Q \mid R & = & P + (Q \mid R) \\
\\
\{{}^{a}\!/\!{}_{b}\}M_1 + M_2 & = & (\{{}^{a}\!/\!{}_{b}\}M_1) + M_2 \\
\{{}^{a}\!/\!{}_{b}\}M_1 \mid M_2 & = & (\{{}^{a}\!/\!{}_{b}\}M_1) \mid M_2
\end{array}$$

Nomi liberi e nomi legati

- $(\nu a)P$ **lega** a in P.
- a occorre **legato** in R se appare in un sottotermine $(\nu a)Q$ di R.
- a occorre **libero** in R se non si trova all'interno di una (νa).
- Definiamo per induzione i **nomi liberi** per processi concorrenti.

$$
\begin{aligned}
\mathsf{fn}(a) &\stackrel{\text{def}}{=} \{a\} \\
\mathsf{fn}(\overline{a}) &\stackrel{\text{def}}{=} \{a\} \\
\mathsf{fn}(\vec{\alpha}) &\stackrel{\text{def}}{=} \mathsf{fn}(\alpha_1) \cup \ldots \cup \mathsf{fn}(\alpha_n) \\
\mathsf{fn}(A\langle\vec{\alpha}\rangle) &\stackrel{\text{def}}{=} \mathsf{fn}(\vec{\alpha}) \\
\mathsf{fn}(\textstyle\sum_{i \in I} \alpha_i.P_i) &\stackrel{\text{def}}{=} \cup_{i \in I}(\mathsf{fn}(\alpha_i) \cup \mathsf{fn}(P_i)) \\
\mathsf{fn}(P_1 \mid P_2) &\stackrel{\text{def}}{=} \mathsf{fn}(P_1) \cup \mathsf{fn}(P_2) \\
\mathsf{fn}((\nu a)P) &\stackrel{\text{def}}{=} \mathsf{fn}(P) \setminus \{a\}
\end{aligned}
$$

- Definiamo formalmente l'insieme dei **nomi legati**, $\mathsf{bn}(P)$, di un processo P, dove al solito $A(\vec{x}) \stackrel{\text{def}}{=} P_A$:

$$
\begin{array}{lcl}
\mathsf{bn}(A\langle\vec{a}\rangle) & \stackrel{\text{def}}{=} & \mathsf{bn}(P_A) \\
\mathsf{bn}(\sum_{i\in I} \alpha_i.P_i) & \stackrel{\text{def}}{=} & \cup_{i\in I}\mathsf{bn}(P_i) \\
\mathsf{bn}(P_1 \mid P_2) & \stackrel{\text{def}}{=} & \mathsf{bn}(P_1) \cup \mathsf{bn}(P_2) \\
\mathsf{bn}((\nu a)P) & \stackrel{\text{def}}{=} & \{a\} \cup \mathsf{bn}(P)
\end{array}
$$

α-conversione e sostituzione

- Rivediamo la definizione di **sostituzione** per processi concorrenti: $\{{}^{\vec{b}}/_{\vec{a}}\}P$ si ottiene rimpiazzando tutte le occorrenze **libere** di a_i in P con b_i evitando la "cattura di nomi liberi".

- Ma allora come si calcola $\{{}^{b}/_{a}\}(\nu b)b.a.\mathbf{0}$?

- Per evitare la **cattura di nomi liberi**, abbiamo bisogno dell'**α-conversione**, scritta $=_\alpha$, attraverso cui è possibile rinominare i nomi legati con **nomi fresh** cioè nomi "vergini" non utilizzati nel nostro termine.

Ad esempio:

$$
\begin{array}{rcl}
(\nu a)(\overline{a}.\mathbf{0} \mid b.\mathbf{0}) & =_\alpha & (\nu c)(\overline{c}.\mathbf{0} \mid b.\mathbf{0}) \\[2ex]
\{^a/_b\}((\nu b)\overline{b}.\mathbf{0} \mid b.\mathbf{0}) & = & ((\nu b)\overline{b}.\mathbf{0} \mid a.\mathbf{0}) \\[2ex]
\{^a/_b\}((\nu a)\overline{b}.a.\mathbf{0} \mid a.b.\mathbf{0}) & =_\alpha & \{^a/_b\}((\nu c)\overline{b}.c.\mathbf{0} \mid a.b.\mathbf{0}) \\
& = & ((\nu c)\overline{a}.c.\mathbf{0} \mid a.a.\mathbf{0})
\end{array}
$$

Importante: Nel seguito identificheremo processi che differiscono solo per α-conversione. Ovvero consideremo uguali processi che differiscono solo per α-conversione.
Esercizio: Calcolare i seguenti termini:

1. $\{^s/_h\}\{^h/_r\}((\nu h)\overline{r}.h.\mathbf{0} \mid h.r.\mathbf{0})$
2. $\{^s/_h\}\{^h/_r\}((\nu h)r(s).h(r).\overline{r}\langle s\rangle.\mathbf{0} \mid h(a).r(h).\overline{h}\langle\rangle.\mathbf{0})$
3. $\{^s/_h\}\{^h/_r\}((\nu h)r(s).(h(r).\overline{r}\langle s\rangle.0 \mid \overline{r}\langle s\rangle) \mid h(a).r(h).\overline{h}\langle r\rangle.0)$

L'importanza di lavorare con congruenze

- Il nostro prossimo obiettivo è quello di riformulare la definizione di *congruenza strutturale* per processi concorrenti.
- Ma cosa vuol dire **congruenza**?
- Una congruenza è una relazione binaria con le seguenti proprietà:
  - È una relazione di equivalenza
  - È preservata da tutti i contesti del linguaggio.
- Quindi la definizione di congruenza è legata a quella di **contesto**. Informalmente un contesto in un linguaggio non è altro che del *codice non completamente definito*, cioè con uno spazio (o buco) vuoto al suo interno che attende di essere istanziato.

Formalmente nel nostro linguaggio per processi concorrenti:

Definizione 15 *Un* **contesto** *è dato dalla seguente grammatica:*

$$\begin{array}{rcl} C[\cdot] & ::= & [\cdot] \\ & | & \alpha.C[\cdot] + M \\ & | & M + \alpha.C[\cdot] \\ & | & (\nu a)C[\cdot] \\ & | & C[\cdot] \mid P \\ & | & P \mid C[\cdot]. \end{array}$$

Nota che in un contesto, il buco $[\cdot]$ può trovarsi ad una profondità arbitraria. Una categoria più semplice di contesti sono quelli in cui il buco si trova al più a profondita uno.

Definizione 16 *I* **contesti elementari** *sono definiti dalla seguente grammatica:*

$$E[\cdot] ::= [\cdot] \quad | \quad \alpha.[\cdot]+M \quad | \quad M+\alpha.[\cdot] \quad | \quad (\nu a)[\cdot] \quad | \quad [\cdot] \mid P \quad | \quad P \mid [\cdot].$$

Dato un contesto $C[\cdot]$ ed un processo Q, allora $C[Q]$ è il **processo** ottenuto istanziando lo spazio (o buco) $[\cdot]$ in $C[\cdot]$ con Q. Ad esempio, se $C[\cdot] \stackrel{\text{def}}{=} a.b.[\cdot] + c.\mathbf{0}$ allora $C[Q]$ è dato da: $a.b.Q + c.\mathbf{0}$.

Definizione 17 (Congruenza tra processi) *Sia $\cong$ una relazione di equivalenza sull'insieme di processi (o anche linguaggio) $\mathcal{P}$. Diremo che la relazione $\cong$ è una congruenza se è preservata da tutti i contesti elementari; cioè, se $P \cong Q$ allora per ogni α, M ed R:*

$$
\begin{array}{rcl}
M + \alpha.P & \cong & M + \alpha.Q \\
\alpha.P + M & \cong & \alpha.Q + M \\
(\nu a)P & \cong & (\nu a)Q \\
P \mid R & \cong & Q \mid R \\
R \mid P & \cong & R \mid Q
\end{array}
$$

Il seguente risultato afferma che è sufficiente definire una congruenza per contesti elementari.

Proposizione 18 *Una relazione di equivalenza $\cong$ è una congruenza* **se e solo se** *per ogni contesto $C[\cdot]$, se $P \cong Q$ allora $C[P] \cong C[Q]$.*

Domanda: *Ma perchè è importante che una equivalenza tra processi sia una congruenza?*
Risposta: *Perchè ciò ci consente di sostituire moduli (processi) equivalenti all'interno di sistemi, anche grossi, senza cambiarne la semantica, ovvero il comportamento osservabile dall'esterno.*

Esempio 19 *Supponiamo che due moduli (processi) P e Q siano equivalenti, $P \approx Q$, e supponiamo anche che $\approx$ sia una congruenza. Consideriamo allora un generico sistema da istanziare:*

$$S[\cdot] \stackrel{def}{=} (\nu a)(a.\overline{b}.R + S \mid b.c.\overline{a}.W + c.Z \mid \overline{b}.[\cdot])$$

allora, il suo comportamento non cambia se istanziamo il buco con P oppure con Q. Detto in altri termini:

$$S[P] \approx S[Q].$$

- Nel seguito ci concentreremo sempre su equivalenze tra processi che siano anche **congruenze**. Questo perchè una congruenza tra due processi è valida in qualsiasi contesto, cioè per qualsiasi osservatore esterno che tenti di interagire con i due processi.

- A volte una congruenza può essere definita come la **congruenza indotta da un insieme di leggi o equazioni algebriche**. Cioè, dato un insieme di equazioni, due termini P e Q sono congruenti se Q può essere trasformato in P applicando le leggi algebriche, in entrambe le direzioni, ed anche all'interno di sottotermini. È questo il caso della congruenza strutturale.

- In prima approssimazione possiamo dire che due processi P e Q sono strutturalmente congruenti, scritto $P \equiv Q$, se sono uguali a meno di

  - alpha-conversione
  - permutazione dei termini delle somme
  - permutazione delle componenti parallele
  - permutazione delle dichiarazioni di nome.

- In altre parole la congruenza strutturale uguaglia processi che si differiscono sintatticamente solo apparentemente. In effetti i due processi sono strutturalmente "uguali', ovvero hanno la medesima struttura.

Definizione 20 *La* **congruenza strutturale**, *scritta* $\equiv$, *è una relazione binaria su processi definita come la congruenza indotta dagli* assiomi *e dalle* regole di inferenza *che seguono:*

1 $P =_\alpha Q$ *implica* $P \equiv Q$

2 *Possiamo permutare gli addendi di una somma*

3 $(\mathcal{P}, \mid, \mathbf{0})$ *è un monoide commutativo, cioè*

$$P \mid \mathbf{0} \equiv P, \quad P \mid Q \equiv Q \mid P, \quad P \mid (Q \mid R) \equiv (P \mid Q) \mid R$$

4 $(\nu a)(P \mid Q) \equiv P \mid (\nu a)Q \quad$ *se* $a \notin \mathsf{fn}(P)$
$(\nu a)\mathbf{0} \equiv \mathbf{0}$
$(\nu a)(\nu b)P \equiv (\nu b)(\nu a)P$

5 $A\langle \vec{b} \rangle \equiv \{\vec{b}/\vec{a}\}P_A \quad$ *se* $A(\vec{a}) \stackrel{def}{=} P_A$.

$$(\mathit{Ide})\ \frac{-}{P \equiv P} \qquad (\mathit{Sym})\ \frac{P \equiv Q}{Q \equiv P} \qquad (\mathit{Trans})\ \frac{P \equiv Q \qquad Q \equiv R}{P \equiv R}$$

$$(\mathit{Cxt})\ \frac{P \equiv Q}{C[P] \equiv C[Q]}$$

La congruenza strutturale è perciò formalmente la chiusura riflessiva, simmetrica, transitiva e per contesti dell'insieme di assiomi dati in precedenza.
Si noti che l'insieme dei nomi liberi è **invariante** rispetto alla congruenza strutt.

Proposizione 21 *Se* $P \equiv Q$ *allora* $\mathsf{fn}(P) \equiv \mathsf{fn}(Q)$*.*

Prova Qualche idea di come si provi?

Definizione 22 *Un processo* $(\nu\vec{a})(M_1 \mid \ldots \mid M_n)$*, dove ciascun* M_i *è una somma non vuota, è in* **standard form***. (Se* $n = 0$ *allora* $M_1 \mid \ldots \mid M_n$ *significa* $\mathbf{0}$*. Se* $\vec{a}$ *è vuoto allora omettiamo la restrizione.)*

Teorema 23 *Ogni processo è strutturalmente congruente ad uno in standard form.*

Prova Si portino all'esterno tutte le restrizioni (νa) che non si trovano dentro una somma. A tal fine usiamo l'α-conversione (se necessario) seguita dalla legge 20(4) ed eventualmente le leggi 20(3).

Esercizio 24 (Pag. 35, da fare a casa) *Mostrare che il processo visto in precedenza*

$$P = (\nu a)((a.Q_1 + b.Q_2) \mid \overline{a}.\mathbf{0} \mid \mathbf{0}) \Big| (\bar{b}.R_1 + \overline{a}.R_2)$$

ha la seguente standard form:

$$(\nu a')((a'.Q'_1 + b.Q'_2) \mid (\bar{b}.R_1 + \overline{a}.R_2) \mid \overline{a'}.\mathbf{0})$$

dove a' *è* **fresh** *e* $Q'_i = \{^{a'}\!/_a\}Q_i$.

Esercizio 25 (vedi pag. 32) *Provare che* $(\nu a)P \equiv P$ *se* $a \notin \mathsf{fn}(P)$.

Esempio 26 (Esempio 4.11 pag. 32) *Sia* P *un processo con due porte* l *ed* r *(vedi figura a pag. 32). Spesso si rivelerà utile collegare processi di questo tipo connettendo la porta destra di uno con la porta sinistra dell'altro.* **Operatore di link:**

$$P \frown Q \stackrel{def}{=} (\nu m)(\{^m\!/_r\}P \mid \{^m\!/_l\}Q)$$

per m **fresh**. Non importa quale m si scelga, perchè a meno di α-conversione otteniamo lo stesso processo. L'operatore di link può essere generalizzato al caso in cui si hanno n porte.

$$(\nu\vec{m})(\{\vec{m}/\vec{r}\}P \mid \{\vec{m}/\vec{l}\}Q)$$

Esercizio 27 (Da fare a casa) *Provare che l'operatore di link è associativo a meno di $\equiv$. Cioè*

$$P \frown (Q \frown R) \equiv (P \frown Q) \frown R.$$