La Mobilità in sistemi concorrenti

- Introduciamo adesso il **π-calcolo** di Robin Milner, Joachim Parrow e David Walker.
- Il π-calcolo è un **modello algebrico** attraverso cui è possibile rappresentare una **rete** di processi in grado scambiare **messaggi**. Tali messaggi possono contenere:
  - **nomi di canali**, ovvero links a processi attivi (e parleremo di first-order π-calculus), oppure
  - **processi** attivabili a destinazione che possono a loro volta contenere links ad altri processi (e parleremo di higher-order π-calculus).
- La **nozione primitiva** su cui si fonda il π-calcolo è quella di **interazione** (tra componenti di un sistema). Così come, ad esempio, nel λ-calcolo la nozione cruciale è quella di **funzione matematica**.

- L'aspetto fondamentale che distingue il π-calcolo dal linguaggio per processi concorrenti visto finora è la capacità di modellare sistemi in cui **la struttura di comunicazione** tra processi **cambia a tempo di esecuzione**.

- Ma cosa ha a che vedere tutto ciò con la **mobilità**?

- È bene subito dire che il termine mobilità viene usato con differenti accezioni.

- Tali differenze possono essere desunte rispondendo a due domande:

  - **Che tipo di entità si muovono?**
  - **In quale spazio si muovono?**

  Ci sono varie possibilità:

  (A) Migrazione di **agenti** in corso di esecuzione che si spostano dallo **spazio fisico** di un sito a quello di un altro. (Es: telefoni cellulari, aerei, ect.)

(B) Migrazione di **codice** (non in esecuzione) che viene spostato (downloaded) dallo **spazio fisico** di un sito a quello di un altro (Es: Applets).

(C) Migrazione di **codice** all'interno di uno **spazio virtuale** composto da processi collegati attraverso canali.

(D) Migrazione di **links**, cioè creazione e/o distruzione di connessioni all'interno di uno **spazio virtuale** di processi interconnessi.

- Il π-calcolo modella la forma più semplice di mobilità, cioè la quarta (D). Le ragioni di tale scelta si possono così riassumere:
  - Ragionando in astratto, la locazione di un processo all'interno di uno spazio virtuale è determinata dai links che il processo ha verso altri processi.
  - In quest'ottica la migrazione di codice da uno spazio virtuale ad un altro può essere rappresentata interamente attraverso la migrazione dei suoi links. Perciò il caso C si può ridurre al caso D

- I casi A e B riguardano sistemi che oltre ad essere concorrenti sono anche *distribuiti* cioè basati su una nozione di locazione/sito in cui un processo può essere eseguito. Il π-calcolo non modella la nozione di locazione.

- Per quanto riguarda la mobilità di tipo A, i ricercatori stanno ancora cercando di capire se questa può essere ricondotta a quella di tipo B. Certamente quella di tipo A è più difficile da implementare poichè riguarda la mobilità di agenti in esecuzione e quindi comprensivi di stato.

- In questo corso vedremo brevemente modelli per sistemi distribuiti incentrati sulla mobilità di tipo A come l'**Ambient Calculus** (Cardelli e Gordon 1998).

- E modelli incentrati sulla mobilità di tipo B come il **Distributed π-calculus** (Hennessy e Riely 1998).

Forme limitate di mobilità di tipo (D)

- I processi visti nelle lezioni precedenti consentono di modellare alcune forme di mobilità (di tipo D, ovviamente).
- Consideriamo ad esempio il processo

$$S \stackrel{\text{def}}{=} (\nu c)(A \mid C) \mid B$$

dove A e B comunicano col mondo esterno attraverso le porte a e b (guardate figura (1) a pag. 79).

- Con la sintassi per processi concorrenti vista sinora possiamo cambiare la struttura dei links solo in due modi.
- Supponiamo che

$$A \stackrel{\text{def}}{=} a.(\nu d)(A \mid A') + c.A''$$

allora A può interagire con l'ambiente esterno attraverso il canale a per poi dividersi in due componenti, A stesso ed A', tra loro connesse da un canale privato d.

- In tal caso la struttura di comunicazione cambierebbe come mostrato in figura (2) a pag. 79.
- Adesso, supponendo che $A' \stackrel{\text{def}}{=} c.\mathbf{0}$ e $C \stackrel{\text{def}}{=} \overline{c}.C'$, un'interazione sul canale c darebbe luogo ad una implosione della struttura di comunicazione come esemplificato in figura (3) a pag 80.
- Comunque coi processi visti finora non è possibile in nessun modo creare nuovi link di comunicazione tra componenti già esistenti. Come ad esempio collegare le componenti C e B di figura (4) a pag. 80.
- Questa limitazione è abbastanza importante in quanto molti sistemi reali creano connessioni a tempo di esecuzione.
- Come facciamo allora a superare questa limitazione? **Trasmettendo sui canali (nomi di) altri canali**.
- **outputs:** $\overline{a}\langle b\rangle$ trasmette il canale b attraverso il canale a.
- **inputs:** $a(x)$ riceve sul canale a un qualsiasi canale b ed istanzia x con b.

- Ad esempio in π-calcolo:

$$\overline{c}\langle d\rangle.P \mid c(x).\overline{x}\langle a\rangle.Q \xrightarrow{\tau} P \mid \{^{d}/_{x}\}\overline{x}\langle a\rangle.Q = P \mid \overline{d}\langle a\rangle.\{^{d}/_{x}\}Q$$

- La comunicazione è detta **poliadica** quando si trasmettono più nomi di canale allo stesso tempo:
  - **outputs:** $\overline{a}\langle\vec{b}\rangle$ trasmette la sequenza di canali $\vec{b} = b_1 \ldots b_n$ attraverso il canale a.
  - **inputs:** $a(\vec{x})$ riceve sul canale a una qualsiasi sequenza di canali $\vec{b}$ ed istanzia $\vec{x}$ con $\vec{b}$.

Modellare in π la telefonia mobile (pag. 81)

- Per introdurre π-calcolo mostriamo un esempio in cui la mobilità riveste un ruolo centrale: una rete di telefonia mobile (ovviamente vedremo solo una semplificazione).

- Supponente di avere dei veicoli in movimento i cui conduttori sono in possesso di un cellulare.

- Ogni cellulare sarà connesso ad un antenna fissa T (transmitter) con un raggio d'azione limitato.

- Tutte le antenne T sono collegate via cavo ad una centrale di controllo.

- Quando un veicolo si muove, ed esce dal raggio d'azione di una antenna, la comunicazione viene garantita grazie ad altre antenne, sparse nel territorio, che sono in grado di captare il segnale del cellulare ed inviarlo alla centrale di controllo.

- Non confondiamo lo **spostamento fisico** di una vettura con lo **spostamento virtuale** all'interno dello spazio di comunicazione, cioè la creazione di nuovi links.

- Le figure di pag. 81 mostrano come cambia il sistema allo spostarsi di un'autovettura. Tale interconnessione richiede un **protocollo** chiamato "hand-over". Per semplicità considereremo il caso di una *sola vettura e due antenne*.

- Una antenna è collegata sia con la vettura che con la centrale di controllo. Con la vettura comunicherà attraverso un canale *talk* mentre con la centrale di controllo comunicherà attraverso un canale *lose* per sapere quando sconnettersi con la vettura.

- Modelliamo le due antenne con le espressioni $Trans_i$ per $i \in \{1, 2\}$; useremo $Idtrans_i$ per rappresentare l'antenna i in uno stato di non collegamento (idle); il sistema di controllo può trovarsi in due stati $Control_1$ e $Control_2$, a seconda di dove sia la vettura.

- La vettura può trovarsi in due stati: Car_1 e Car_2, a seconda dell'antenna con cui è connessa.

$$
\begin{array}{rcll}
Trans_i & \stackrel{\mathrm{def}}{=} & Trans\langle talk_i, switch_i, gain_i, lose_i\rangle & i \in \{1,2\} \\
Trans(t,s,g,l) & \stackrel{\mathrm{def}}{=} & t.Trans\langle t,s,g,l\rangle + l(t',s').\overline{s}\langle t',s'\rangle.Idtrans\langle g,l\rangle & \\
Idtrans_i & \stackrel{\mathrm{def}}{=} & Idtrans\langle gain_i, lose_i\rangle & i \in \{1,2\} \\
Idtrans(g,l) & \stackrel{\mathrm{def}}{=} & g(t,s).Trans\langle t,s,g,l\rangle &
\end{array}
$$

$$
\begin{array}{rcl}
Control_1 & \stackrel{\mathrm{def}}{=} & \overline{lose_1}\langle talk_2, switch_2\rangle.\overline{gain_2}\langle talk_2, switch_2\rangle.Control_2 \\
Control_2 & \stackrel{\mathrm{def}}{=} & \overline{lose_2}\langle talk_1, switch_1\rangle.\overline{gain_1}\langle talk_1, switch_1\rangle.Control_1
\end{array}
$$

$$
\begin{array}{rcll}
Car_i & \stackrel{\mathrm{def}}{=} & Car\langle talk_i, switch_i\rangle & i \in \{1,2\} \\
Car(t,s) & \stackrel{\mathrm{def}}{=} & \overline{t}.Car\langle t,s\rangle + s(t',s').Car\langle t',s'\rangle &
\end{array}
$$

$$
\begin{array}{rcl}
System_1 & \stackrel{\mathrm{def}}{=} & (\nu\ talk_1, switch_1, gain_1, lose_1, talk_2, switch_2, gain_2, lose_2) \\
 & & (Car_1 \mid Trans_1 \mid IdTrans_2 \mid Control_1)
\end{array}
$$

Spieghiamo un 'po cosa accade:

- $Trans_1$ può comunicare con la vettura nello stato Car_1 attraverso il canale $talk_1$.
- $Trans_1$ può anche ricevere dal controllo $Control_1$ il segnale, attraverso il canale $lose_1$, che Car_1 sta uscendo dal raggio di azione di $Trans_1$.
- In tale interazione $Control_1$ passa all'antenna i nuovi canali $talk_2$ e $switch_2$ che questa dovra passare alla vettura per poter comunicare con la seconda antenna $Trans_2$.
- Tali informazioni vengono passate da $Trans_1$ a Car_1 attraverso il canale $switch_1$.
- Tale comunicazione fa transitare $Trans_1$ nello stato $Idtrans_1$.
- Infine il controllo trasmette a $Idtrans_2$, attraverso il canale $gain_2$, i canali $talk_2$ e $switch_2$ con cui $Trans_2$ potrà comunicare con la vettura, la quale nel frattempo è transitata in uno stato Car_2.

Quello che abbiamo descritto a parole è il funzionamento del protocollo "hand-over" che consente al sistema di evolvere nella seguente maniera:

$$System_1 \xrightarrow{\tau}^{\star} System_2$$

dove

$$\begin{array}{rcl} System_2 & \stackrel{\text{def}}{=} & (\nu\, talk_1, switch_1, gain_1, lose_1, talk_2, switch_2, gain_2, lose_2) \\ & & (Car_2 \mid IdTrans_1 \mid Trans_2 \mid Control_2) \end{array}$$

Elastic Buffers

La definizione di buffer unario, già vista nelle lezioni precedenti, può essere riscritta in π-calcolo nel modo seguente:

$$\begin{array}{rcl} B(l,r) & \stackrel{\text{def}}{=} & l(x).C\langle x,l,r\rangle \\ C(x,l,r) & \stackrel{\text{def}}{=} & \overline{r}\langle x\rangle.B\langle l,r\rangle \end{array}$$

- Si ricordi che B denota il buffer vuoto mentre C il buffer pieno.

- Si noti che adesso i dati bufferizzati sono nomi di canale e non valori di un qualche insieme $V = \{1, \ldots, n\}$.

- Comunque, booleani, integer, arrays, ect possono tutti essere facilmente codificati in π-calcolo.

- Se abbreviamo $B\langle l, r\rangle$ con B sappiamo che possiamo utilizzare l'operatore di link per definire un buffer di dimensioni n;

$$B^{(n)} \stackrel{\text{def}}{=} \overbrace{B \frown \ldots \frown B}^{n \text{ volte}}$$

- Tale buffer ha dimensione fissa n. Se volessimo un buffer illimitato potremmo, ad esempio, creare una nuova cella ogni qualvolta che un nuovo dato viene inserito nel buffer:

$$\begin{array}{rcl} B(l,r) & \stackrel{\text{def}}{=} & l(x).B\langle l,r\rangle \frown C\langle x,l,r\rangle \\ C(x,l,r) & \stackrel{\text{def}}{=} & \overline{r}\langle x\rangle.B\langle l,r\rangle \end{array}$$

Per fare ciò non è necessario fare ricorso al passaggio di canali.

- Ma come potremmo compattare buffers contententi al loro interno delle celle vuote?

- Tale operazione di ricompattamento è molto utile nell'implementazione di meccanismi di garbage collection.

- Per capire l'intuizione di cosa intendiamo fare si guardi la figura finale di pag. 85.

- Intuitivamente possiamo "sopprimere" una cella B purchè questi passi al suo predecessere i canali verso il suo successore. Per far ciò dobbiamo introdurre altri due canali.

$$
\begin{aligned}
B(l_1, l_2, r_1, r_2) &\stackrel{\text{def}}{=} l_1(x).C\langle x, l_1, l_2, r_1, r_2\rangle + \overline{l_2}\langle r_1, r_2\rangle.\mathbf{0} \\
C(x, l_1, l_2, r_1, r_2) &\stackrel{\text{def}}{=} \overline{r_1}\langle x\rangle.B\langle l_1, l_2, r_1, r_2\rangle \\
&\quad + \\
&\quad r_2(r_1', r_2').C\langle x, l_1, l_2, r_1', r_2'\rangle.
\end{aligned}
$$