Controllare la comunicazione in π-calcolo

- Abbiamo già visto come simulare la trasmissione di tipi di dato quali booleani e liste: trasmettendo il nome del canale dove sono allocati.

- Adesso vediamo come garantire che lo scambio di tali tipi di dato sia ben tipato.

- Introduciamo un semplice *type system* per il π-calcolo.

- I nostri tipi in π-calcolo li chiameremo 'sorts' in quanto descrivono quale 'sorta' di infomazione viene scambiata attraverso un canale.

- Un esempio di processo mal tipato:

$$P = x(y).\overline{y}\langle u, v\rangle \qquad Q = \overline{x}\langle y'\rangle.y'(w).Q'$$

  La composizione parallela $P \mid Q$ non è ben tipata perchè non viene rispettata l'arietà sul canale y'.

Sorting

- Assumiamo una collezione Σ di *sorts*.
- Ad ogni nome x viene associato un sort $\sigma \in \Sigma$ e scriviamo $x : \sigma$.
- Liste di sorts $\vec{\sigma} = \sigma_1, \ldots, \sigma_n$.
- $\Sigma^\star$ è l'insieme di tutte le liste di sorts.
- Convenzioni:
  - Se $\vec{x} = x_1, \ldots, x_n$ allora scriveremo $\vec{x} : \vec{\sigma}$.
  - Se $F = (\vec{x}).P$ e $\vec{x} : \vec{\sigma}$, allora scriveremo $F : \vec{\sigma}$.
  - Per ogni processo P, $P : \epsilon$.
- **Definizione 94** *Dato un insieme Σ di sorts, un* **sorting** *su Σ è una funzione parziale*

  $$ob : \Sigma \rightarrow \Sigma^\star$$

  Diciamo che un processo rispetta un sorting ob se per ogni sotto-termine della forma $x(\vec{y}).P$ o $\overline{x}\langle\vec{y}\rangle.P$ se $x : \sigma$ allora $\vec{y} : ob(\sigma)$.

- **Sorting nell'esempio della telefonia mobile** Scegliamo

$$\Sigma = \{TALK, SWITCH, GAIN, LOSE\}$$

con $talk_i$:$TALK$, $switch_i$:$SWITCH$, $gain_i$:$GAIN$, $lose_i$:$LOSE$ per $(i = 1, 2)$.

Allora l'unica funzione di sorting rispettata dal sistema è:

$$ob : \quad \left(\begin{array}{rcl} TALK & \mapsto & \epsilon \\ SWITCH & \mapsto & TALK, SWITCH \\ GAIN & \mapsto & TALK, SWITCH \\ LOSE & \mapsto & TALK, SWITCH \end{array} \right.$$

- Notare che nell'esempio dato precedentemente non esiste un sorting che possa essere rispettato dalla composizione parallela $P \mid Q$.

- Fare Esercizio 11.4.

- **Proposizione 95 Le funzioni di sorting sono invarianti rispetto alla congruenza strutturale ed alle regole di tran-**

sizione; *cioè se* $P \equiv P'$ *o* $P \xrightarrow{\pi} P'$ *e* P *rispetta un sorting ob allora anche* P' *rispetta ob.*

- **Esempio 96 (Il sorting assiste nell'analisi)** *Supponiamo di avere una composizione parallela* $P \mid Q$ *con nomi liberi* $x_i : A$, $y_i : B$, $u_i : C$, *e* $v_i : D$, *e che rispetti un sorting ob secondo cui*

$$A \mapsto B, \qquad B \mapsto \epsilon, \qquad C \mapsto D, \qquad D \mapsto \epsilon.$$

Se P *contiene solo* x_i *tra i suoi nomi liberi, allora* P *può ricevere* y_i *da* Q*, ed usarlo, ma non riceverà mai* u_i *o* v_i*.*

Verso un linguaggio di sorts

Una funzione di sorting contiene informazioni sulle comunicazioni di un sistema.

Ad esempio, se consideriamo le interazioni legate all'uso di booleani e liste, abbiamo:

$$True \stackrel{\text{def}}{=} (b).b(t,f).\overline{t} \qquad Nil \stackrel{\text{def}}{=} (l).l(n,c).\overline{n}$$

$$False \stackrel{\text{def}}{=} (b).b(t,f).\overline{f} \qquad Node \stackrel{\text{def}}{=} (l,v,l').l(n,c).\overline{c}\langle v,l'\rangle$$

$$b : BOOL,\ t : TRUE,\ f : FALSE \qquad l : LIST,\ n : NIL,\ c : CONS,\ v : VAL$$

$$\left(\begin{array}{rcl} BOOL & \mapsto & TRUE, FALSE \\ TRUE & \mapsto & \epsilon \\ FALSE & \mapsto & \epsilon \end{array}\right. \qquad \left(\begin{array}{rcl} LIST & \mapsto & NIL, CONS \\ CONS & \mapsto & VAL, LIST \\ NIL & \mapsto & \epsilon \end{array}\right.$$

Dove $BOOL$, $TRUE$, ect sono detti **sorts di base**. Si noti però che questi sorts sono sostanzialmente simili. Possiamo allora generalizzare la nozione di sorting!! Ma come?

- Potremmo non voler distinguere i sorts $TRUE$, $FALSE$, e NIL.
- Infatti, canali di tali sort sono utilizzati per spedire messaggi vuoti. E quindi sono essenzialmente dello stesso tipo.

- Potremmo allora introdurre un **costruttore di sort** $CHAN$ tale che, data una lista di sorts $\vec{\sigma}$, il sort $CHAN(\vec{\sigma})$ è quello dei canali che trasportano messaggi di sort $\vec{\sigma}$.

- In particolare i sorts $TRUE$, $FALSE$, e NIL possono essere sostituiti dal sort $CHAN(\epsilon)$.

- Notiamo inoltre che il sort $LIST$ non ci dice nulla sul **polimorfismo** di una lista, cioè sulla possibilità che la lista contenga valori di natura differente.

- Vorremmo piuttosto avere un sort $LIST(\sigma)$, parametrico nel sort σ dei valori che possono essere contenuti nella lista.

- In tal modo il nostro sorting adesso effettuerà la seguente associazione:
$$LIST(\sigma) \mapsto CHAN(\epsilon), CHAN(\sigma, LIST(\sigma))$$

In questo modo riusciamo a costruire sorts che non sono delle entità atomiche ma sono costruite utilizzando altri sort e costruttori di sort, generando una **gerarchia di tipi**.

Definizione 97 *Un* **linguaggio di sorts** Σ *è generato da un insieme* Γ *di costruttori di tipo, ognuno di arietà* ≥ 0*. Il linguaggio* Σ *consiste di sorts* σ *della forma*

$$C(\sigma_1, \ldots, \sigma_n)$$

dove $C \in \Gamma$ *ha arietà* n *e* $\sigma_1, \ldots, \sigma_n$ *sono sorts. Un* **sorting** *definito su* Σ *è una funzione parziale*

$$ob : \Sigma \longrightarrow \Sigma^\star$$

tale che ad ogni costruttore di sort $C \in \Gamma$ *corrisponde al più una regola di sorting della forma*

$$ob : C(s_1, \ldots, s_n) \mapsto \rho_1, \ldots, \rho_n$$

dove $s_1, \ldots, s_n$ *sono variabili di sorts che possono apparire nei sort* ρ_i*. Così per ciascun sort* $\sigma_1, \ldots, \sigma_n$ *la lista di sorts* $ob(C(\sigma_1, \ldots, \sigma_n))$ *è ottenuta sostituendo* $\vec{\sigma}$ *per* $\vec{s}$ *in* $\rho_1, \ldots, \rho_n$

Esempio 98 (Sorting per buffer unario di booleani) *Sia*

$$B \stackrel{def}{=} l(x).\overline{r}\langle x\rangle.B$$

allora potremmo dire che

$$x : BOOL, \quad l : CHAN(BOOL), \quad r : CHAN(BOOL).$$

Nota come la nuova disciplina di sorting sia solo un raffinamento della precedente.

Esempio 99 *Se riprendiamo la cella di memoria* $REF\langle r, v\rangle$*, possiamo definire un construtore di sort* REF *di arietà 1 con la seguente regola di sort:*

$$REF(s) \mapsto CHAN(\epsilon), CHAN(s).$$

Fare esercizio 11.13.

In seguito invece di assumere un assegnamento di un sort per ciascun nome, adotteremo la convenzione (tipica dei linguaggi tipati) di introdurre un nuovo nome legato insieme al suo sort. Esempio:

$$Node(l : LIST(\sigma)) \stackrel{\text{def}}{=} (v, l').l(n, c).\overline{c}\langle v, l'\rangle.$$