Esempi sull'equivalenza osservazionale $\approx$

Nel seguito confronteremo **sistemi** con le loro **specifiche**. La specifica sarà costituita da un processo pressochè privo di composizione parallela. Mentre il sistema (cioè l'implementazione) conterrà un certo numero di componenti in parallelo che interagiscono tra loro. In ciascun caso proveremo un teorema della forma:

$$Sistema \approx Specifica.$$

A tal scopo possiamo usare per lo meno le seguenti quattro tecniche:

1. Fornire una bisimulazione $\mathcal{S}$ t.c. $(Sistema\,, Specifica) \in \mathcal{S}$
2. Fornire una bisimulaz. up to $\sim$, $\mathcal{S}$ t.c. $(Sistema, Specifica) \in \mathcal{S}$
3. Leggi algebriche (Es: expansion law, $P \approx \tau.P$, ect)
4. Usare il Teorema 6.19 di pag. 59 dell'**unicità della soluzione di un sistema di equazioni** (lo vediamo dopo).

Esempio della lotteria

La specifica:

$$Lotspec \stackrel{\text{def}}{=} \tau.b_1.Lotspec + \tau.b_2.Lotspec + \ldots + \tau.b_n.Lotspec$$

L'implementazione:

$$\begin{array}{rcl}
A(x,y,z) & \stackrel{\text{def}}{=} & \overline{x}.C\langle x,y,x\rangle \\
C(x,y,z) & \stackrel{\text{def}}{=} & \tau.B\langle x,y,z\rangle + z.A\langle x,y,z\rangle \\
B(x,y,z) & \stackrel{\text{def}}{=} & y.C\langle x,y,z\rangle
\end{array}$$

$$\begin{array}{rcccl}
A_i & \stackrel{\text{def}}{=} & A\langle a_i, b_i, a_{i+1}\rangle & = & \overline{a_i}.C_i \\
B_i & \stackrel{\text{def}}{=} & B\langle a_i, b_i, a_{i+1}\rangle & = & b_i.C_i \\
C_i & \stackrel{\text{def}}{=} & C\langle a_i, b_i, a_{i+1}\rangle & = & \tau.B_i + a_{i+1}.A_i
\end{array}$$

$$\begin{array}{lcl}
L_1 \stackrel{\text{def}}{=} (\nu a_1a_2a_3)(C_1 \mid A_2 \mid A_3) & \stackrel{\tau}{\longrightarrow} & (\nu a_1a_2a_3)(B_1 \mid A_2 \mid A_3) \stackrel{\text{def}}{=} L_1' \\
L_2 \stackrel{\text{def}}{=} (\nu a_1a_2a_3)(A_1 \mid C_2 \mid A_3) & \stackrel{\tau}{\longrightarrow} & (\nu a_1a_2a_3)(A_1 \mid B_2 \mid A_3) \stackrel{\text{def}}{=} L_2' \\
L_3 \stackrel{\text{def}}{=} (\nu a_1a_2a_3)(A_1 \mid A_2 \mid C_3) & \stackrel{\tau}{\longrightarrow} & (\nu a_1a_2a_3)(A_1 \mid A_2 \mid B_3) \stackrel{\text{def}}{=} L_3'
\end{array}$$

Teorema 56 $L_1 \approx Lotspec$.

Prova Proviamo qualcosa di più generale e cioè che $L_i \approx Lotspec$, per ogni i. Sia

$$\mathcal{R} \stackrel{\text{def}}{=} \{(L_i, Lotspec) \mid 1 \le i \le n\} \cup \{(L'_i, b_i.Lotspec) \mid 1 \le i \le n\}$$

Mostriamo allora che la relazione $\mathcal{R}$ è una **bisimulazione debole**, ovvero che sia $\mathcal{R}$ che $\mathcal{R}^{-1}$ sono simulazioni deboli: Consideriamo prima le coppie della forma $(L'_i, b_i.Lotspec)$ e poi quelle della forma $(L_i, Lotspec)$. Ci sono quattro casi.

1. Sia $(L'_i, b_i.Lotspec) \in \mathcal{R}$, entrambi i processi possono eseguire solo l'azione $\xrightarrow{b_i}$, raggiungendo la coppia $(L_i, Lotspec)$ che per definizione sta in $\mathcal{R}$.

2. Sia $(L_i, Lotspec) \in \mathcal{R}$,

   (a) Se $L_i \xrightarrow{\tau} L_{i+1}$
   allora $Lotspec \Rightarrow Lotspec$ con $(L_{i+1}, Lotspec) \in \mathcal{R}$.

(b) Se $L_i \xrightarrow{\tau} L'_i$
allora $Lotspec \xrightarrow{\tau} b_i.Lotspec$ con $(L'_i\,,\, b_i.Lotspec) \in \mathcal{R}$.

3. Sia $(b_i.Lotspec\,,\, L'_i) \in \mathcal{R}^{-1}$, la prova è simile al caso (1).

4. Sia $(Lotspec\,,\, L_i) \in \mathcal{R}^{-1}$, le possibili azioni di *Lotspec* sono della forma $Lotspec \xrightarrow{\tau} b_k.Lotspec$, ma allora $L_i \Rightarrow L_k \xrightarrow{\tau} L'_k$ e $(b_k.Lotspec\,,\, L'_k) \in \mathcal{R}^{-1}$.

Job shop (bottega di un fabbro)

- In una bottega ci sono due agenti A uguali tra loro che lavorano in una catena di assemblaggio.
- Ci sono tre tipi di lavori: easy (E), neutral (N) e difficult (D).
- Gli agenti ricevono i lavori da fare attraverso tre nastri trasportatori in ingresso modellati dalle porte in input i_E, i_N, e i_D, uno per ogni tipo di lavoro, e forniscono il lavoro finito su un unico nastro trasportatore in uscita modellato dalla porta in output $\overline{o}$.

- Poichè gli agenti hanno una lunga esperienza, essi trattano i vari lavori allo stesso modo, indipendentemente dal grado di difficoltà. Perciò la loro definizione è:

$$\begin{array}{rcl} A & \stackrel{\text{def}}{=} & i_E.A' + i_N.A' + i_D.A' \\ A' & \stackrel{\text{def}}{=} & \overline{o}.A \end{array}$$

  Il sistema composto dai due agenti che lavorano insieme (ovvero la specifica) è dato da:

$$Agency \stackrel{\text{def}}{=} A \mid A$$

- Ad un certo punto gli agenti vengono promossi a manager e rimpiazzati da due operai non specializzati chiamati J (jobbers).

- Un jobber J può fare

  – un lavoro difficult (D) solo utilizzando un martello (hammer)

  – un lavoro neutral (N) utilizzando un martello (hammer) oppure un maglio (mallet).

- un lavoro easy (E) a mani nude.

- Sfortunamente i due jobbers hanno a disposizione un solo hammer ed un solo mallet (**condivisione di risorse da usare in mutua esclusione**).

- Ovviamente i due strumenti sono in pratica due semafori. Vengono acquisiti e poi rilasciati per essere acquisiti possibilmente da un altro jobber.

$$
\begin{array}{lcllcl}
H & \stackrel{\text{def}}{=} & S\langle gh, ph\rangle & \quad M & \stackrel{\text{def}}{=} & S\langle gm, pm\rangle \\
H & \equiv & gh.H' & \quad M & \equiv & gm.M' \\
H' & \equiv & ph.H & \quad M' & \equiv & pm.M
\end{array}
$$

- La definizione di jobbers è leggermente più complicata:

$$
\begin{aligned}
J &\stackrel{\text{def}}{=} i_E.J_E + i_N.J_N + i_D.J_D \\
J_E &\stackrel{\text{def}}{=} \overline{o}.J \\
J_N &\stackrel{\text{def}}{=} \overline{gh}.\overline{ph}.J_E + \overline{gm}.\overline{pm}.J_E \\
J_D &\stackrel{\text{def}}{=} \overline{gh}.\overline{ph}.J_E
\end{aligned}
$$

- L'intero sistema, costituito dai due jobbers un hammer ed un mallet, è dato da:

$$Jobshop \stackrel{\text{def}}{=} (\nu\vec{t})(J \mid J \mid H \mid M)$$

dove $\vec{t}$ sta per la sequenza di nomi gh, ph, gm, pm.

- Guardare il grafo di flusso a pag. 63.

- **Teorema 57** $Agency \approx Jobshop$

Prova Esibiamo una bisimulazione (debole) up to $\sim$. Essa contiene le seguenti coppie (e le loro inverse), dove $X, Y \in \{E, N, D\}$.

$$
\begin{array}{lll}
(1) & A \mid A, & (\nu\vec{t})(J \mid J \mid H \mid M) \\
(2) & A \mid A' & (\nu\vec{t})(J \mid J_X \mid H \mid M) \\
 & A \mid A' & (\nu\vec{t})(J \mid \overline{ph}.J_E \mid H' \mid M) \\
 & A \mid A' & (\nu\vec{t})(J \mid \overline{pm}.J_E \mid H \mid M') \\
(3) & A' \mid A' & (\nu\vec{t})(J_X \mid J_Y \mid H \mid M) \\
 & A' \mid A' & (\nu\vec{t})(J_X \mid \overline{ph}.J_E \mid H' \mid M) \\
 & A' \mid A' & (\nu\vec{t})(J_X \mid \overline{pm}.J_E \mid H \mid M') \\
 & A' \mid A' & (\nu\vec{t})(\overline{ph}.J_E \mid \overline{pm}.J_E \mid H' \mid M')
\end{array}
$$

- Perchè abbiamo bisogno di lavorare up to $\sim$? Sarebbe stato sufficiente lavorare up to $\equiv$? In altre parole, perchè $\mathcal{S}$ così definita non è una bisimulazione?

- Provare a fare a casa l'esercizio 7.4.

Soluzioni uniche di equazioni

Vediamo adesso la nostra quarta tecnica di prova per risolvere un sistema di equazioni della forma:

$$
\begin{array}{rcl}
X_1 & \approx & \alpha_{11}.E_{11} + \ldots + \alpha_{1n_1}.E_{1n_1} \\
X_2 & \approx & \alpha_{21}.E_{21} + \ldots + \alpha_{2n_2}.E_{2n_2} \\
\multicolumn{3}{l}{\ldots\ldots\ldots\ldots}
\end{array}
$$

dove E_{ij} contengono le variabili di processo X_1, $X_2, \ldots$. Un esempio:

$$
\begin{array}{rcl}
X & \approx & a.X + b.Y \\
Y & \approx & c.X
\end{array}
$$

Quante soluzioni ha questo sistema di equazioni? Certamente i processi A e B definiti come

$$
\begin{array}{rcl}
A & \stackrel{\mathrm{def}}{=} & a.A + b.B \\
B & \stackrel{\mathrm{def}}{=} & c.A
\end{array}
$$

sono banalmente una soluzione al sistema sopra esposto. Ma sono l'unica soluzione a meno di $\approx$? In questo caso la risposta è si. Cioè per ogni P e Q che soddisfino il sistema sopra esposto, avremo $P \approx A$ e $Q \approx B$.

Comunque le cose non vanno sempre così. Ad esempio il sistema costituito dalla sola equazione

$$X \approx \tau.X$$

Ha infinite soluzioni. Più precisamente ogni $P \in \mathcal{P}$ soddisfa l'equazione.

Esercizio 58 (Da fare a casa) *Considera l'equazione*

$$X \approx a.P + \tau.X$$

per P fissato. Mostrare che ha soluzione $a.P + \tau.Q$, per ogni processo Q.

Teorema 59 (Unicità della soluzione) *Sia $\vec{X} = X_1, X_2, \ldots$ una sequenza (eventualmente infinita) di variabili di processo. Se nel sistema di equazioni*

$$\begin{array}{lcl} X_1 & \approx & \alpha_{11}.X_{k(11)} + \ldots \alpha_{1n_1}.X_{k(1n_1)} \\ X_2 & \approx & \alpha_{21}.X_{k(21)} + \ldots \alpha_{2n_2}.X_{k(2n_2)} \\ \ldots\ldots\ldots\ldots & & \end{array}$$

vale $\alpha_{ij} \neq \tau$, per ogni i e j, allora esiste un'unica sequenza di processi P_1, $P_2, \ldots$ che soddisfa il sistema di equazioni.

Scheduler

1. Un insieme di agenti P_i, per $1 \leq i \leq n$, deve essere schedulato.
2. Ciascun P_i desidera eseguire un certo task ripetutamente. Lo scheduler si occupa di far sì che gli agenti **inizino** i loro task in ordine ciclico, cominciando da P_1.
3. Ogni agente segnala la terminazione del proprio task allo scheduler.
4. Più processi possono eseguire il proprio task allo stesso tempo.
5. Comunque, lo scheduler non consente ad un processo P_i di iniziare un nuovo task se prima non ha terminato il task precedente.
6. Ogni processo P_i richiede di iniziare il proprio task “premendo” il tasto a_i; la fine del task viene segnalata “premendo” il tasto b_i.

Guardate la figura a pag. 23 del libro. Informalmente la specifica dello scheduler è la seguente:

1. I pulsanti $a_1, \ldots, a_n$ possono essere premuti ciclicamente partendo da a_1.

2. Per ciascun processo P_i i tasti a_i e b_i devono essere premuti in modo alternato.

3. Ogni pulsante può essere premuto, purchè si rispettino i precedenti due punti.

Scriviamo la specifica come un **processo sequenziale**.

- A tal fine definiamo uno stato $Sched_{i,X}$ per ogni $i \in \{1, \ldots, n\}$ e $X \subseteq \{1, \ldots, n\}$.

- Il parametro i indica che è il turno del processo P_i di iniziare.

- Il parametro X rappresenta l'insieme di agenti che in quel momento stanno eseguendo il loro task.

Così in uno stato $Sched_{i,X}$:

1. solo P_i può iniziare (purchè $i \notin X$)
2. ciascun P_j, con $j \in X$, può completere il proprio task.

La **specifica formale** è data allora dalle seguenti equazioni:

$$
\begin{array}{rcl}
Scheduler & \stackrel{\text{def}}{=} & Sched_{1,\emptyset} \\
Sched_{i,X} & \stackrel{\text{def}}{=} & \left(\begin{array}{ll} \sum_{j \in X} b_j.Sched_{i,X-j} & (i \in X) \\ \sum_{j \in X} b_j.Sched_{i,X-j} + a_i.Sched_{i+1,X \cup i} & (i \notin X) \end{array} \right)
\end{array}
$$

dove $i+1$ è interpretato modulo n, ed abbreviamo $X \cup \{i\}$ e $X \setminus \{i\}$, con $X \cup i$ e $X - i$, rispettivamente.

Adesso supponiamo di voler **implementare** uno scheduler attraverso un **anello** di n celle identiche $A(a, b, c, d)$. La i-esima cella ha **due porte** per controllare l'agente P_i ed altre **due porte** per comunicare con le celle adiacenti (guardare figura pag. 65).

La definizione di A potrebbe essere la seguente:

$$A(a,b,c,d) \stackrel{\text{def}}{=} a.c.b.\overline{d}.A$$

Introducendo degli *stati intermedi* potremmo riscrivere la definizione:

$$\begin{array}{rclcrcl}
A(a,b,c,d) & \stackrel{\text{def}}{=} & a.C\langle a,b,c,d\rangle & \qquad & C(a,b,c,d) & \stackrel{\text{def}}{=} & c.B\langle a,b,c,d\rangle \\
B(a,b,c,d) & \stackrel{\text{def}}{=} & b.D\langle a,b,c,d\rangle & \qquad & D(a,b,c,d) & \stackrel{\text{def}}{=} & \overline{d}.A\langle a,b,c,d\rangle
\end{array}$$

Al fine di collegare le n celle definiamo:

$$\begin{array}{rclcl}
A_i & \stackrel{\text{def}}{=} & A\langle a_i,b_i,c_i,c_{i-1}\rangle & = & a_i.C_i \\
C_i & \stackrel{\text{def}}{=} & C\langle a_i,b_i,c_i,c_{i-1}\rangle & = & c_i.B_i \\
B_i & \stackrel{\text{def}}{=} & B\langle a_i,b_i,c_i,c_{i-1}\rangle & = & b_i.D_i \\
D_i & \stackrel{\text{def}}{=} & D\langle a_i,b_i,c_i,c_{i-1}\rangle & = & \overline{c_{i-1}}.A_i
\end{array}$$

Nello stato A_i il processo P_i può iniziare. Mentre in B_i il processo P_i può terminare. Nello stato D_i, P_i attende il proprio turno per iniziare. Infine in C_i, P_i può consentire a P_{i+1} di iniziare.

Una implementazione dello scheduler potrebbe essere:

$$
\begin{array}{rcl}
S & \stackrel{\text{def}}{=} & (\nu c_1, \ldots, c_n)(A_1 \mid D_2 \mid \ldots \mid D_n) \\
& = & (\nu c_1, \ldots, c_n)(a_1.C_1 \mid \overline{c_1}.A_2 \mid \ldots \mid \overline{c_{n-1}}.A_n)
\end{array}
$$

In cui P_1 è il primo a poter iniziare mentre i restanti $n-1$ processi attendono (pazientemente) il loro turno. Così, ad esempio

$$
\begin{array}{rcl}
S & \xrightarrow{a_1} & (\nu c_1, \ldots, c_n)(C_1 \mid D_2 \mid \ldots \mid D_n) \\
& = & (\nu c_1, \ldots, c_n)(c_1.B_1 \mid \overline{c_1}.A_2 \mid D_3 \mid \ldots \mid D_n) \\
& \xrightarrow{\tau} & (\nu c_1, \ldots, c_n)(B_1 \mid A_2 \mid D_3 \mid \ldots \mid D_n) \\
& = & (\nu c_1, \ldots, c_n)(b_1.D_1 \mid a_2.C_2 \mid \overline{c_2}.A_3 \mid \ldots \mid \overline{c_{n-1}}.A_n) \stackrel{\text{def}}{=} T
\end{array}
$$

Adesso vi sono due possibilità (i) o P_1 termina (eseguendo una $\xrightarrow{b_1}$) oppure (ii) inizia P_2 (eseguendo una $\xrightarrow{a_2}$). Nella seconda ipotesi avremmo

$$
\begin{array}{rcl}
T & \xrightarrow{a_2}\xrightarrow{\tau} & (\nu c_1, \ldots, c_n)(B_1 \mid B_2 \mid A_3 \mid D_4 \mid \ldots \mid D_n) \\
& = & (\nu c_1, \ldots, c_n)(b_1.D_1 \mid b_2.D_2 \mid a_3.C_3 \mid \overline{c_3}.A_4 \mid \ldots \mid \overline{c_{n-1}}.A_n) \stackrel{\text{def}}{=} U
\end{array}
$$

Adesso P_3 può iniziare mentre P_1 e P_2 possono entrambi terminare.

Adesso, nell'ipotesi che P_2 termini il sistema evolverebbe in:

$$\begin{array}{lcl} U & \xrightarrow{b_2} & (\nu c_1, \ldots, c_n)(B_1 \mid D_2 \mid A_3 \mid D_4 \mid \ldots \mid D_n) \\ & = & (\nu c_1, \ldots, c_n)(b_1.D_1 \mid D_2 \mid a_3.C_3 \mid \overline{c_3}.A_4 \mid \ldots \mid \overline{c_{n-1}}.A_n) \\ & \stackrel{\text{def}}{=} & V \end{array}$$

Allora, il nostro processo S rappresenta una proposta di implementazione per lo scheduler. Ovviamente vorremmo provare che tale implementazione è in effetti corretta, vale a dire che

$$S \approx Scheduler.$$

Curiosamente *ciò non è vero*!!!
In effetti quello sopra è un tipico esempio di un *baco* che può affliggere una implementazione per anni senza essere "scovato". Quel che è peggio è che anche quando appare non è affatto facile riprodurlo.

Chi di voi trova il problema che affligge l'implementazione S?

Il problema è dovuto al fatto che S può raggiungere uno stato per cui non ne esiste uno corrispondente in $Scheduler$. Tale stato lo si produce quando tutti i processi hanno iniziato il loro task, uno dopo l'altro, senza che nessuno termini. Cioè

$$
\begin{array}{lcl}
S & \xrightarrow{a_1}\xrightarrow{\tau} & (\nu c_1,\ldots,c_n)(B_1 \mid A_2 \mid D_3 \mid \ldots \mid D_n) \\
 & \xrightarrow{a_2}\xrightarrow{\tau} & (\nu c_1,\ldots,c_n)(B_1 \mid B_2 \mid A_3 \mid D_4 \mid \ldots \mid D_n) \\
 & \ldots & \ldots \\
 & \xrightarrow{a_{n-1}}\xrightarrow{\tau} & (\nu c_1,\ldots,c_n)(B_1 \mid B_2 \mid \ldots \mid B_{n-1} \mid A_n) \\
 & \xrightarrow{a_n} & (\nu c_1,\ldots,c_n)(B_1 \mid B_2 \mid \ldots \mid B_{n-1} \mid C_n) \\
 & = & (\nu c_1,\ldots,c_n)(b_1.D_1 \mid b_2.D_2 \mid \ldots \mid b_{n-1}.D_{n-1} \mid c_n.B_n) \\
 & = & (\nu c_1,\ldots,c_n)(b_1.\overline{c_n}.A_1 \mid b_2.D_2 \mid \ldots \mid b_{n-1}.D_{n-1} \mid c_n.B_n)
\end{array}
$$

In tale stato, curiosamente, solo i primi $n-1$ processi possono terminare. Mentre il processo P_n può terminare solo dopo che P_1 abbia terminato. **Ma questa non era la nostra specifica?** Come risolviamo il problema?

Dobbiamo cambiare leggermente la definizione dei moduli:

$$\begin{array}{rclcrcl} A(a,b,c,d) & \stackrel{\text{def}}{=} & a.C\langle a,b,c,d\rangle & \qquad & C(a,b,c,d) & \stackrel{\text{def}}{=} & c.E\langle a,b,c,d\rangle \\ B(a,b,c,d) & \stackrel{\text{def}}{=} & b.A\langle a,b,c,d\rangle & & D(a,b,c,d) & \stackrel{\text{def}}{=} & \overline{d}.A\langle a,b,c,d\rangle \end{array}$$

$$E(a,b,c,d) \stackrel{\text{def}}{=} b.D\langle a,b,c,d\rangle + \overline{d}.B\langle a,b,c,d\rangle$$

E quindi:

$$\begin{array}{rclcl} A_i & \stackrel{\text{def}}{=} & A\langle a_i,b_i,c_i,c_{i-1}\rangle & = & a_i.C_i \\ C_i & \stackrel{\text{def}}{=} & C\langle a_i,b_i,c_i,c_{i-1}\rangle & = & c_i.E_i \\ E_i & \stackrel{\text{def}}{=} & E\langle a_i,b_i,c_i,c_{i-1}\rangle & = & b_i.D_i + \overline{c_{i-1}}.B_i \\ B_i & \stackrel{\text{def}}{=} & B\langle a_i,b_i,c_i,c_{i-1}\rangle & = & b_i.A_i \\ D_i & \stackrel{\text{def}}{=} & D\langle a_i,b_i,c_i,c_{i-1}\rangle & = & \overline{c_{i-1}}.A_i \end{array}$$

Sostanzialmente, adesso un processo P_i, dopo aver iniziato un task e passato la mano a P_{i+1}, può decidere di terminare il proprio task oppure di “sbloccare” il processo P_{i-1}, se necessario.

Fare Esercizio 7.6 a pag. 66.

Considera, ad esempio, lo stato $(\nu\vec{c})(D_1 \mid E_2 \mid A_3 \mid E_4)$. In tale stato S può eseguire l'azione $\xrightarrow{a_3}$ ma nessun altra a_i. Può anche eseguire b_2 e b_4 ma non b_1 e b_3. Questo vuol dire che P_2 e P_4 sono in esecuzione ma non P_1 e P_3. Così S si trova in uno stato corrispondente allo stato $Sched(3, \{2, 4\})$ dello $Scheduler$.

Fare Esercizio 7.7 a pag. 66.

Esercizio 60 (Esercizio 7.8 pag. 66) *Usare l'***expansion law** *per mostrare che i seguenti stati di S sono equivalenti in senso forte a sommatorie della forma $\sum_r \alpha_r.S_r$ con α_r uguale ad a_i, b_i o τ e S_r stati accessibili.*

$$
\begin{aligned}
(\nu\vec{c})(D_1 \mid D_2 \mid A_3 \mid E_4) &\stackrel{def}{=} (\nu\vec{c})(\overline{c_4}.A_1 \mid \overline{c_1}.A_2 \mid a_3.C_3 \mid b_4.D_4 + \overline{c_3}.B_4) \\
&\sim a_3.(\nu\vec{c})(\overline{c_4}.A_1 \mid \overline{c_1}.A_2 \mid C_3 \mid b_4.D_4 + \overline{c_3}.B_4) \\
&\qquad + \\
&\quad\; b_4.(\nu\vec{c})(\overline{c_4}.A_1 \mid \overline{c_1}.A_2 \mid a_3.C_3 \mid D_4)
\end{aligned}
$$

Similmente

$$
\begin{array}{rcl}
(\nu\vec{c})(E_1 \mid D_2 \mid C_3 \mid E_4) & \stackrel{def}{=} & (\nu\vec{c})(b_1.D_1 + \overline{c_4}.B_1 \mid \overline{c_1}.A_2 \mid c_3.E_3 \mid b_4.D_4 + \overline{c_3}.B_4) \\
& \sim & b_1.(\nu\vec{c})(D_1 \mid \overline{c_1}.A_2 \mid c_3.E_3 \mid b_4.D_4 + \overline{c_3}.B_4) \\
& & + \\
& & b_4.(\nu\vec{c})(b_1.D_1 + \overline{c_4}.B_1 \mid \overline{c_1}.A_2 \mid c_3.E_3 \mid D_4) \\
& & + \\
& & \tau.(\nu\vec{c})(b_1.D_1 + \overline{c_4}.B_1 \mid \overline{c_1}.A_2 \mid E_3 \mid B_4)
\end{array}
$$

Ma passiamo adesso al caso per n generico. Definiamo gli **stati accessibili** di S nel seguente modo, Siano $\{i\}$, X e Y una partizione di $\{1, \ldots, n\}$ (cioè disgiunti e la loro unione è $\{1, \ldots, n\}$). Allora:

$$
\begin{array}{rcl}
A_{i,X,Y} & \stackrel{\text{def}}{=} & (\nu\vec{c})(A_i \mid \prod_{j\in X} D_j \mid \prod_{k\in Y} E_k) \\
B_{i,X,Y} & \stackrel{\text{def}}{=} & (\nu\vec{c})(B_i \mid \prod_{j\in X} D_j \mid \prod_{k\in Y} E_k) \\
C_{i,X,Y} & \stackrel{\text{def}}{=} & (\nu\vec{c})(C_i \mid \prod_{j\in X} D_j \mid \prod_{k\in Y} E_k)
\end{array}
$$

Intuitivamente lo stato $A_{i,X,Y}$ è lo stato in cui (i) P_i può iniziare; (ii) i processi in Y sono in esecuzione e possono (se vogliono) terminare; (iii) i processi in X non sono in esecuzione ma sono in stand-by, aspettano cioè che qualcuno passi loro la mano per poter iniziare. Ragionando in maniera analoga si deduce il comportamento di $B_{i,X,Y}$ e $C_{i,X,Y}$.

Esercizio 61 (Esercizio 7.9 – da fare) *Usando l'Expansion Law possiamo mostrare le seguenti equivalenze forti:*

$$
\begin{array}{rcl}
A_{i,X,Y} & \sim & a_i.C_{i,X,Y} + \sum_{k \in Y} .b_k.A_{i,X \cup k, Y-k} \\
B_{i,X,Y} & \sim & b_i.A_{i,X,Y} + \sum_{k \in Y} .b_k.B_{i,X \cup k, Y-k} \\
C_{i,X,Y} & \sim & \sum_{k \in Y} .b_k.C_{i,X \cup k, Y-k} \\
 & & \qquad\qquad + \\
 & & \left(\begin{array}{ll} \tau.A_{i+1,X-(i+1),Y \cup i} & (i+1 \in X) \\ \tau.B_{i+1,X,Y-(i+1) \cup i} & (i+1 \in Y) \end{array} \right)
\end{array}
$$

Esercizio 62 (Esercizio 7.10 – da fare) *Sia $\mathcal{R}$ la relazione contenente le seguenti coppie (e le coppie inverse):*

$$
\begin{array}{lcl}
\big(A_{i,X,Y} & , & Sched_{i,Y}\big) \\
\big(B_{i,X,Y} & , & Sched_{i,Y\cup i}\big) \\
\big(C_{i,X,Y} & , & Sched_{i+1,Y\cup i}\big)
\end{array}
$$

Al variare di $\{i\}$, X e Y che rappresentino una partizione di $\{1, \ldots n\}$. Mostrare allora che $\mathcal{R}$ è una bisimulazione debole up to $\sim$, usando l'esercizio precedente.

Teorema 63 (Teorema 7.11 – correttezza dello scheduler)

$$S \approx Scheduler.$$

Prova È sufficiente notare che la coppia $(S\,,\,Scheduler)$ è contenuta in $\mathcal{R}$. Tale coppia corrisponde al caso $i = 1$, $X = \{2, \ldots, n\}$, e $Y = \emptyset$. Per la Proposizione 6.9 di pag. 55 (correttezza della bisim. debole up to $\sim$) possiamo concludere.