Lezione 10 MSC
CCS – semantica operazionale ed osservazioni su guardatezza

Roberto Gorrieri

Dalla Sintassi alla Semantica

- Una volta definito l'insieme $\mathcal{P}$ dei processi CCS, bisogna definire una funzione di interpretazione semantica che associa ad ogni processo un LTS, ovvero il suo "significato"
- In realtà, noi definiremo un grandissimo LTS per tutto CCS: per individuare quello di un particolare processo p, bisognerà estrarre il sotto-LTS raggiungibile dallo stato iniziale p.
- Questo LTS "globale" avrà infiniti stati (ogni processo CCS è uno stato), infinite labels (se il calcolo non è finito, ovvero può usare infinite azioni), infinite transizioni.
- Come rappresentare in modo finito l'insieme infinito contabile degli stati e delle transizioni? Per gli stati, abbiamo definito nella lezione precedente una grammatica, che è una rappresentazione implicita finita di questo insieme infinito. Ma per le transizioni … come fare?

Semantica Operazionale Strutturata (SOS)

- Le transizioni sono definite per mezzo di un inference system, cioè un insieme finito di regole di inferenza la cui definizione è syntax-driven.
- Una tipica regola di inferenza SOS ha la forma

$$\frac{\text{Premessa-1 .... Premessa-n}}{\textbf{Conclusione}} \quad \text{side-condition}$$

dove ogni premessa/**conclusione** è una transizione e side-condition è un predicato che deve essere vero per poter applicare la regola. Se n = 0, la regola si chiama assioma.

- Il transition system per CCS è la tripla ($\mathcal{P}$,Act, $\rightarrow$) where $\rightarrow \subseteq \mathcal{P} \times \text{Act} \times \mathcal{P}$ è la minima relazione generata dall'assioma e dalle regole d'inferenza dei prossimi lucidi.

Regole per la somma/scelta

- Queste regole chiariscono perché diciamo che le regole sono definite per induzione sulla struttura del termine: per derivare una transizione da p + q, dobbiamo prima risolvere il problema più "semplice" di trovare una transizione da p (Sum-1) o da q (Sum-2):

$$(\mathrm{Sum}_1) \quad \frac{p \xrightarrow{\mu} p'}{p+q \xrightarrow{\mu} p'} \qquad (\mathrm{Sum}_2) \quad \frac{q \xrightarrow{\mu} q'}{p+q \xrightarrow{\mu} q'}$$

- Nota che l'alternativa non selezionata viene scartata e che l'operatore sparisce (dinamico)
- Nota che possiamo scegliere di eseguire solo un processo che può muovere: 0 + a.b –a-> b   ma 0 non può essere scelto

Assioma per il prefisso e regola per la costante

- Non c'è nessuna regola per nil 0; allora da 0 non parte nessuna transizione.

$$\text{(Pref)} \quad \frac{}{\mu.p \xrightarrow{\mu} p} \qquad\qquad \text{(Cons)} \quad \frac{p \xrightarrow{\mu} p'}{C \xrightarrow{\mu} p'} \quad C \stackrel{def}{=} p$$

- Osserva che l'occorrenza del prefisso scompare nello stato raggiunto (operatore dinamico)
- Osserva che la regola (Cons) non è definita induttivamente, perché la premessa parte da un processo più complicato

Regole del parallelo “asincrono”

- Queste regole descrivono l’esecuzione indipendente/asincrona di azioni da parte di un componente di un sistema parallelo

$$(\text{Par}_1) \quad \frac{p \xrightarrow{\mu} p'}{p\,|\,q \xrightarrow{\mu} p'\,|\,q} \qquad\qquad (\text{Par}_2) \quad \frac{q \xrightarrow{\mu} q'}{p\,|\,q \xrightarrow{\mu} p\,|\,q'}$$

- Nota che il sottocomponente che sta fermo (idle) non viene scartato e che l’operatore permane (statico).
- Base dell’assunzione di interleaving: i due processi mescolano le loro azioni in tutti i modi possibili perché non si fa nessuna ipotesi sulla velocità relativa dei due.

Regola di sincronizzazione

- Se i due componenti eseguono simultaneamente due azioni complementari, allora ha luogo la sincronizzazione (handshake)

$$\text{(Com)}\quad \frac{p \xrightarrow{\alpha} p' \qquad q \xrightarrow{\overline{\alpha}} q'}{p \mid q \xrightarrow{\tau} p' \mid q'}$$

- Nota che la sincronizzazione è strettamente binaria (point-to-point), poiché la transizione risultante è etichettata tau, che non può essere usata da un terzo processo per sincronizzarsi.
- Problema: come riuscire ad avere sincronizzazioni multi-way? Regole diverse, operatori diversi , …??? In CCS è impossibile (si veda Multi-CCS, capitolo 6 del libro)
- Problema: comunicazione asincrona? Implementabile per mezzo di comunicazione sincrona (buffer intermedio).

Regola per restrizione

- Questa regola spiega che il ruolo della restrizione è di legare un nome (a, in questo caso), affinchè non sia più disponibile per l'ambiente esterno.
- (va)p impedisce qualunque transizione etichettata a o $\bar{a}$ che p può produrre, mentre non ha effetto sulle altre transizioni di p. In particolare, dentro p possono avvenire sincronizzazioni su a.

$$\text{(Res)} \quad \frac{p \xrightarrow{\mu} p'}{(\nu a)p \xrightarrow{\mu} (\nu a)p'} \; \mu \neq a, \bar{a}$$

Regole SOS - tabella riassuntiva

(Pref)	$\dfrac{}{\mu.p \xrightarrow{\mu} p}$	(Cons)	$\dfrac{p \xrightarrow{\mu} p'}{C \xrightarrow{\mu} p'}$	$C \stackrel{def}{=} p$
(Sum$_1$)	$\dfrac{p \xrightarrow{\mu} p'}{p+q \xrightarrow{\mu} p'}$	(Sum$_2$)	$\dfrac{q \xrightarrow{\mu} q'}{p+q \xrightarrow{\mu} q'}$	
(Par$_1$)	$\dfrac{p \xrightarrow{\mu} p'}{p \mid q \xrightarrow{\mu} p' \mid q}$	(Par$_2$)	$\dfrac{q \xrightarrow{\mu} q'}{p \mid q \xrightarrow{\mu} p \mid q'}$	
(Com)	$\dfrac{p \xrightarrow{\alpha} p' \qquad q \xrightarrow{\bar{\alpha}} q'}{p \mid q \xrightarrow{\tau} p' \mid q'}$	(Res)	$\dfrac{p \xrightarrow{\mu} p'}{(\nu a)p \xrightarrow{\mu} (\nu a)p'}$	$\mu \neq a, \bar{a}$

Table 3.1 Structural Operational Semantics: syntax-driven axiom and inference rules.

Come si deriva una transizione?

Prendiamo (va)((a.E + b.**0**) | ‘a.F) –tau-> (va) (E | F) e dimostriamola

-
a.E –a-> E

|
a.E + b.**0** –a-> E

-
‘a.F –’a-> F

(a.E + b.**0**) | ‘a.F –tau-> E | F

|
(va)((a.E + b.**0**) | ‘a.F) –tau-> (va)(E | F)

- Induzione sulla struttura sintattica
- “Albero di prova” con Assiomi come “foglie” e il “teorema” come “radice”
- Facile implementazione di queste regole in PROLOG => semplice interprete sequenziale per CCS => **prototipazione!**

Altri esempi di derivazioni

$$
(\mathrm{Sum}_1)\ \dfrac{(\mathrm{Pref})\ \dfrac{}{a.b.\mathbf{0} \xrightarrow{a} b.\mathbf{0}}}{a.b.\mathbf{0} + b.a.\mathbf{0} \xrightarrow{a} b.\mathbf{0}}
\qquad
(\mathrm{Sum}_2)\ \dfrac{(\mathrm{Pref})\ \dfrac{}{b.a.\mathbf{0} \xrightarrow{b} a.\mathbf{0}}}{a.b.\mathbf{0} + b.a.\mathbf{0} \xrightarrow{b} a.\mathbf{0}}
$$

$$
(\mathrm{Res})\ \dfrac{(\mathrm{Com})\ \dfrac{(\mathrm{Pref})\ \dfrac{}{a.c.\mathbf{0} \xrightarrow{a} c.\mathbf{0}} \qquad (\mathrm{Sum}_1)\ \dfrac{(\mathrm{Pref})\ \dfrac{}{\bar{a}.\mathbf{0} \xrightarrow{\bar{a}} \mathbf{0}}}{\bar{a}.\mathbf{0} + c.\mathbf{0} \xrightarrow{\bar{a}} \mathbf{0}}}{a.c.\mathbf{0} \,|\, (\bar{a}.\mathbf{0} + c.\mathbf{0}) \xrightarrow{\tau} c.\mathbf{0} \,|\, \mathbf{0}}}{(\nu c)(a.c.\mathbf{0} \,|\, (\bar{a}.\mathbf{0} + c.\mathbf{0})) \xrightarrow{\tau} (\nu c)(c.\mathbf{0} \,|\, \mathbf{0})}
$$

Come si derivano tutte le transizioni?

- Cerchiamo tutte le transizioni da (νa)((a.E + b.**0**) | ‘a.F)
- Induzione sulla struttura sintattica: allora in questo caso prima provare a ritroso la regola (Res)
- Man mano si impongono vincoli sul tipo di etichetta e sulla forma dello stato raggiunto
- Adesso l’operatore principale è | => 3 diverse regole da provare! Alcune porteranno a fallimento, altre a successo.

$$\dfrac{\dfrac{}{(a.E + b.\mathbf{0}) \mid {}`a.F \;\text{–}a?\text{->}\; G}\;\mid}{(\nu a)((a.E + b.\mathbf{0}) \mid {}`a.F) \;\text{–}a?\text{->}\; G?}$$

due variabili

G? deve essere (νa)G e a? deve essere diversa da a e ‘a

Esercizi SOS

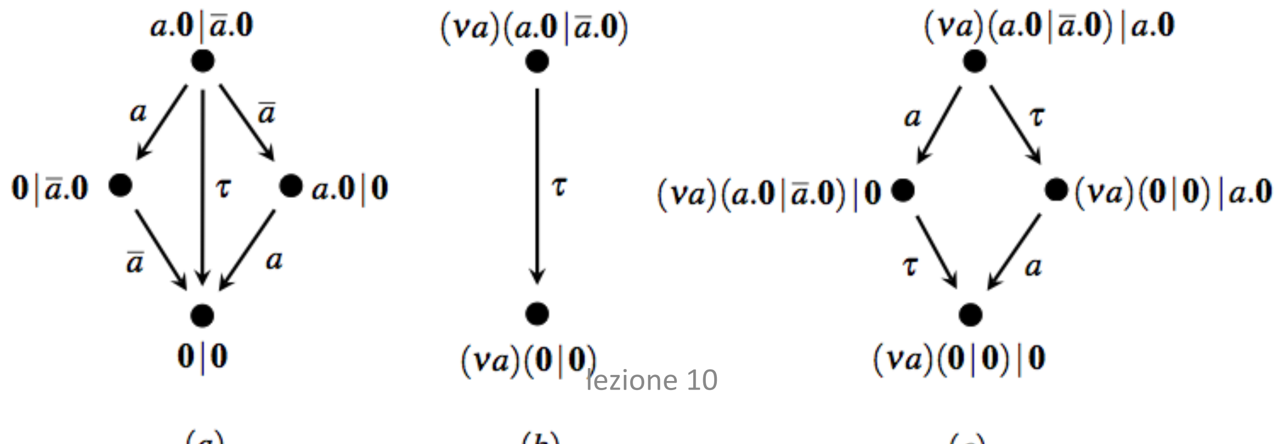
Exercise 3.12. Use the operational rules of Table 3.1 to prove that the following two transitions are derivable:

$$(\nu c)(a.c.\mathbf{0} \mid (b.\mathbf{0} + c.\mathbf{0})) \xrightarrow{b} (\nu c)(a.c.\mathbf{0} \mid \mathbf{0}) \quad (a.\mathbf{0} \mid \bar{a}.\mathbf{0}) \mid \mathbf{0} \xrightarrow{\tau} (\mathbf{0} \mid \mathbf{0}) \mid \mathbf{0}$$ $\square$

Exercise 3.13. Use the operational rules of Table 3.1 to prove that the following two transitions are *not* derivable:

$$(\nu c)(a.c.\mathbf{0} \mid b.\mathbf{0}) \xrightarrow{\tau} (\nu c)(c.\mathbf{0} \mid \mathbf{0}) \quad (\nu a)(a.\mathbf{0} \mid \bar{a}.\mathbf{0}) \xrightarrow{a} (\nu a)(\mathbf{0} \mid \bar{a}.\mathbf{0})$$ $\square$

Exercise 3.14. Use the operational rules to derive the portion of the CCS transition system reachable from $(a.\mathbf{0} \mid \bar{a}.\mathbf{0})$, $(\nu a)(a.\mathbf{0} \mid \bar{a}.\mathbf{0})$ and $(\nu a)(a.\mathbf{0} \mid \bar{a}.\mathbf{0}) \mid a.\mathbf{0}$. Compare the resulting lts's with those in Figure 3.4. $\square$



Altri esercizi

- Deriva l'lts per $a.b.0 \mid \bar{a}.0$
- Deriva l'lts per $(a.b.0 + a.c.0) \mid \bar{a}.0.$
- Deriva l'lts per $(\nu d)(a.(d.b.0|d.c.0)|\bar{d}.0)$
- Deriva l'lts per $(\nu d)(a.(b.d.0+c.d.0)|\bar{d}.e.0)$
- Deriva l'lts per $(\nu d)(a.(b.d.0|c.d.0)|\bar{d}.\bar{d}.e.0)$

SOS ben formata

- A partire da processi CCS, si raggiungono solo processi CCS => la semantica SOS è ben formata

Proposition 3.1. *For any $p \in \mathscr{P}$, if $p \xrightarrow{\mu} p'$, then $p' \in \mathscr{P}$.*

Proof. It is enough to observe, by induction on the proof of $p \xrightarrow{\mu} p'$, that if p is a CCS process term, then also p' is a CCS process term. Additionally, we can prove that $Const(p') \subseteq Const(p)$, so that if p satisfies the guardedness condition, the same holds for p'. □

Notation: For any $p \in \mathscr{P}$, the reachable lts from p (see Definition 2.3) is $\mathscr{C}_p = (\mathscr{P}_p, sort(p), \to_p, p)$, where $\mathscr{P}_p$ is the set of states reachable from p, $sort(p)$ is the set of actions that can be performed by p and $\to_p$ is the restriction of the transition relation on the set $\mathscr{P}_p \times sort(p) \times \mathscr{P}_p$.

- $\mathcal{P}_p$ è davvero un sottoinsieme di $\mathcal{P}$

LTS raggiungibile da p è ridotto

Proposition 3.2. *For any* $p \in \mathscr{P}$, *the lts* $\mathscr{C}_p = (\mathscr{P}_p, sort(p), \rightarrow_p, p)$ *reachable from* p *is a reduced rooted lts.* □

Exercise 3.16. Use the operational rules to derive the part of the CCS transition system reachable from the following six states:

$$a.b.\mathbf{0}+\mathbf{0} \qquad a.(b.\mathbf{0}+c.d.\mathbf{0}) \qquad A \stackrel{def}{=} a.(b.A+c.\mathbf{0})$$

$$a.(b.\mathbf{0}+c.d.\mathbf{0}) \mid \bar{b}.\mathbf{0} \qquad (\nu b)(a.(b.\mathbf{0}+c.d.\mathbf{0}) \mid \bar{b}.\mathbf{0}) \qquad (\nu a)(B \mid C)$$

where $B \stackrel{def}{=} a.B+b.\mathbf{0}$ and $C \stackrel{def}{=} \bar{a}.C+c.\mathbf{0}$. □

State space explosion problem

Exercise 3.22. (Counting states) Recall that $\mathscr{P}_p$ denotes the set of processes reachable from p and that $|J|$ is the cardinality of set J. Looking at the SOS rules, argue that, if $|\mathscr{P}_{p_1}| = k_1$ and $|\mathscr{P}_{p_2}| = k_2$, then $|\mathscr{P}_{p_1+p_2}| \leq k_1 + k_2 + 1^3$ and $|\mathscr{P}_{p_1 \mid p_2}| = k_1 \times k_2$. Moreover, if $q = (\nu a)p_1$, conclude that $|\mathscr{P}_q| \leq k_1$. $\square$

Remark 3.7. **(State space explosion problem)** The exercise above explains that if we have a compound process $p_1 \mid p_2 \mid \ldots \mid p_n$, where each p_i generates an lts with 10 states, then the lts for $p_1 \mid p_2 \mid \ldots \mid p_n$ has 10^n states, i.e., the state space of a compound system grows exponentially w.r.t. the number of components. This phenomenon is sometimes called the *state space explosion problem*. $\square$

Esempio: Verificare deadlock-freeness composizionalmente

Tecnica composizionale per verificare se $P = p_1 \mid \ldots \mid p_n$ è deadlock-free: basta che uno degli n p_i sia deadlock-free per dedurre che tutto P è deadlock-free. Quindi anziché costruire l'lts per P e poi verificare che esso è deadlock-free, ci basta costruire gli lts per gli n processi p_i e verificare se almeno uno è deadlock-free.

(N.B: non stiamo osservando il deadlock parziale, ma solo quello totale.)

Unguardedness può causare infinite prove di una transizione…

- Considera una costante non guardata A = a.0 + A.
- Quante transizioni in uscita da A? UNA

  A –a--> 0

- Quante prove per questa transizione? INFINITE diverse!
- Anche una “divergenza” …. nel procedimento di dimostrazione della transizione!

Unguardedness may imply infinite branching

- Considera C = (a.0 | C). Quante transizioni in uscita?

$$
\begin{array}{rc}
\text{(Pref)} & \\ \cline{2-2}
 & a.\mathbf{0} \xrightarrow{a} \mathbf{0} \\
\text{(Par}_1\text{)} & \\ \cline{2-2}
 & a.\mathbf{0} \,|\, C \xrightarrow{a} \mathbf{0} \,|\, C \\
\text{(Cons)} & \\ \cline{2-2}
 & C \xrightarrow{a} \mathbf{0} \,|\, C
\end{array}
$$

$$
\begin{array}{rc}
\text{(Pref)} & \\ \cline{2-2}
 & a.\mathbf{0} \xrightarrow{a} \mathbf{0} \\
\text{(Par}_1\text{)} & \\ \cline{2-2}
 & a.\mathbf{0} \,|\, C \xrightarrow{a} \mathbf{0} \,|\, C \\
\text{(Cons)} & \\ \cline{2-2}
 & C \xrightarrow{a} \mathbf{0} \,|\, C \\
\text{(Par}_2\text{)} & \\ \cline{2-2}
 & a.\mathbf{0} \,|\, C \xrightarrow{a} a.\mathbf{0} \,|\, (\mathbf{0} \,|\, C) \\
\text{(Cons)} & \\ \cline{2-2}
 & C \xrightarrow{a} a.\mathbf{0} \,|\, (\mathbf{0} \,|\, C)
\end{array}
$$

$$
\begin{array}{rc}
\text{(Pref)} & \\ \cline{2-2}
 & a.\mathbf{0} \xrightarrow{a} \mathbf{0} \\
\text{(Par}_1\text{)} & \\ \cline{2-2}
 & a.\mathbf{0} \,|\, C \xrightarrow{a} \mathbf{0} \,|\, C \\
\text{(Cons)} & \\ \cline{2-2}
 & C \xrightarrow{a} \mathbf{0} \,|\, C \\
\text{(Par}_2\text{)} & \\ \cline{2-2}
 & a.\mathbf{0} \,|\, C \xrightarrow{a} a.\mathbf{0} \,|\, (\mathbf{0} \,|\, C) \\
\text{(Cons)} & \\ \cline{2-2}
 & C \xrightarrow{a} a.\mathbf{0} \,|\, (\mathbf{0} \,|\, C) \\
\text{(Par}_2\text{)} & \\ \cline{2-2}
 & a.\mathbf{0} \,|\, C \xrightarrow{a} a.\mathbf{0} \,|\, (a.\mathbf{0} \,|\, (\mathbf{0} \,|\, C)) \\
\text{(Cons)} & \\ \cline{2-2}
 & C \xrightarrow{a} a.\mathbf{0} \,|\, (a.\mathbf{0} \,|\, (\mathbf{0} \,|\, C))
\end{array}
$$

Guardedness implies finite branching

- Proposizione: per ogni processo CCS q, l'insieme delle transizioni in uscita da q è finito.

We can define an upper-bound $\gamma(q)$ on the number of transitions leaving a given state/process q. Function $\gamma : \mathscr{P} \to \mathbb{N}$ is such:

$$
\begin{array}{llll}
\gamma(\mathbf{0}) & = 0 & \gamma(\mu.p) & = 1 \\
\gamma(p_1 + p_2) & = \gamma(p_1) + \gamma(p_2) & \gamma(A) & = \gamma(p) \quad \text{if } A \stackrel{def}{=} p \\
\gamma((\nu a)p) & = \gamma(p) & \gamma(p_1 \mid p_2) & = \gamma(p_1) + \gamma(p_2) + \gamma(p_1) \times \gamma(p_2)
\end{array}
$$

By guardedness, we are sure that $\gamma(A)$ is always a finite number. It is not difficult then to check – by reasoning on the shape of the SOS inference rules – that indeed $\gamma(q)$ is an upper bound on the number of transitions leaving q. □

- Corollario: lts raggiungibile da un processo CCS p è finitely branching, cioè ogni suo stato raggiungibile ha un numero finito di transizioni in uscita.

Soluzione unica di equazioni?

- Problema: esiste uno / esistono infiniti processo/i CCS X tale che

$$X \sim E(X) \ ?$$

- Soluzioni: Tutti i p tali che p ~ E{p/X}
- La soluzione è "unica" (up to ~) quando:

  p ~ E{p/X} e q ~ E{q/X} implicano p ~ q.
- Esempio: X ~ a.0 + X (dove la variabile X occorre non guardata) ha infinite soluzioni non bisimili! Tutti i processi che abbiano un sommando a.0 (ad esempio a.0 o a.0 + b.0 o a.0 + a.b.0, ...)

Guardatezza implica soluzione unica

- Esempio: $X \sim b.X + a.0$ (X è guardata)
- Soluzione "unica": $A = b.A + a.0$

  Qualunque altra soluzione è bisimile ad A.

Proposition 3.4. *Let X be a process variable guarded in E and let $var(E) \subseteq \{X\}$. Then, if $p \sim E\{p/X\}$ and $q \sim E\{q/X\}$, then $p \sim q$.*