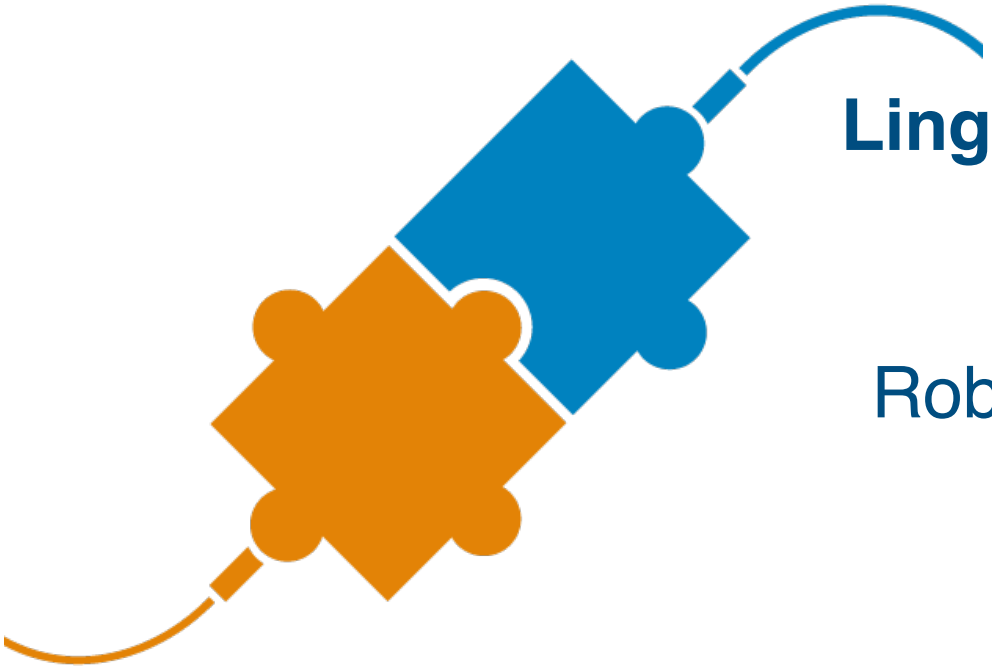


Linguaggi di Programmazione

Roberta Gori

Applicazioni del CCS

CCS: sintassi

p, q	$::=$	$\mathbf{nil}$	processo inattivo
	$\mid$	x	variabile di processo (per la ricorsione)
	$\mid$	$\mu.p$	prefisso azione
	$\mid$	$p\backslash\alpha$	canale ristretto
	$\mid$	$p[\phi]$	rietichettatura del canale
	$\mid$	$p + q$	scelta nondeterministica (somma)
	$\mid$	$p \mid q$	composizione parallela
	$\mid$	$\mathbf{rec}\ x.\ p$	ricorsione

(gli operatori sono elencati in ordine di precedenza)

Un po' di notazione

scriviamo $\sum_{i=1}^{n} p_i$ invece di $p_1 + \cdots + p_n$

scriviamo $\prod_{i=1}^{n} p_i$ invece di $p_1 | \cdots | p_n$

scriviamo $p \backslash \{a_1, ..., a_n\}$ invece di $p \backslash a_1 \cdots \backslash a_n$

scriviamo $\mu^n.p$ invece di $\underbrace{\mu.\mu....\mu.}_{n} p$

CCS op. semantica

Act) $\dfrac{}{\mu.p \xrightarrow{\mu} p}$ Res) $\dfrac{p \xrightarrow{\mu} q \quad \mu \notin \{\alpha, \overline{\alpha}\}}{p\backslash\alpha \xrightarrow{\mu} q\backslash\alpha}$ Rel) $\dfrac{p \xrightarrow{\mu} q}{p[\phi] \xrightarrow{\phi(\mu)} q[\phi]}$

SumL) $\dfrac{p_1 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q}$ SumR) $\dfrac{p_2 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q}$

ParL) $\dfrac{p_1 \xrightarrow{\mu} q_1}{p_1 | p_2 \xrightarrow{\mu} q_1 | p_2}$ Com) $\dfrac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\overline{\lambda}} q_2}{p_1 | p_2 \xrightarrow{\tau} q_1 | q_2}$ ParR) $\dfrac{p_2 \xrightarrow{\mu} q_2}{p_1 | p_2 \xrightarrow{\mu} p_1 | q_2}$

Rec) $\dfrac{p[^{\mathbf{rec}\ x.\ p}/_x] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$

CCS
Codificare un linguaggio imperativo

Prima cosa: terminazione

Un canale dedicato *done*:

viene inviato un messaggio quando il comando corrente termina

$$\text{Done} \triangleq \overline{done}$$

$$\text{Done} \xrightarrow{\overline{done}} \mathbf{nil}$$

Terminazione

Skip

skip

non fa nulla invia *done*

τ.Done

- $\xrightarrow{\tau}$ Done $\xrightarrow{\overline{done}}$ **nil**

Variabili

x che varia su $V = \{v_1, ..., v_n\}$

un processo dedicato alla gestione di ogni variabile

possiamo leggere il suo valore attuale (xr_i canale)

possiamo scrivere qualsiasi valore (canale xw_i)

$$\mathsf{XW} \triangleq \sum_{i=1}^{n} xw_i.\mathsf{X}_i$$
$$= xw_1.\mathsf{X}_1 + \cdots + xw_n.\mathsf{X}_n$$

$$\mathsf{X}_i \triangleq \overline{xr}_i.\mathsf{X}_i + \mathsf{XW}$$

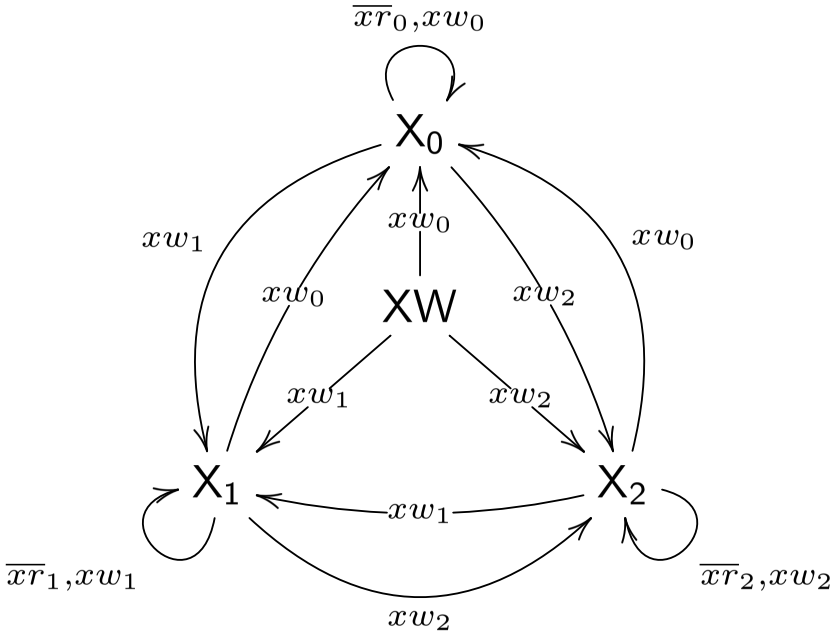
Dichiarazione di variabili

var x

rilascia una variabile non inizializzata e termina

XW|Done

$V = \{v_0, v_1, v_2\}$



| Done $\xrightarrow{\overline{done}}$ **nil**

Assignamento

$$x := v_i$$

invia un messaggio per cambiare lo stato della variabile e poi termina

$$\overline{xw}_i.\text{Done}$$

- $\xrightarrow{\overline{xw}_i}$ Done $\xrightarrow{\overline{done}}$ **nil**

Composizione sequenziale

$$c_1 \; ; \; c_2$$

assumiamo p_1 modella c_1

p_2 modella c_2

$$p_1 | done.p_2$$

scelta infelice: non è scalabile

$$c_1; c_2; c_3$$

$$p_1 | done.p_2 | done.p_3$$

p_3 puo' iniziare dopo p_1

Sequenziale?

Sequential

Composizione sequenziale

$$c_1 \; ; \; c_2$$

assumiamo p_1 modella c_1

p_2 modella c_2

$$\phi_d(done) = d$$

$$p_1 \frown p_2 \triangleq (p_1[\phi_d]|d.p_2)\backslash d$$

ora d e' locale a p_1 e p_2

$$(\ ((p_1[\phi_{d_1}] \mid d_1.p_2)\backslash d_1)[\phi_{d_2}] \mid d_2.p_3)\backslash d_2$$

$$(\ ((p_1[\phi_d] \mid d.p_2)\backslash d)[\phi_d] \mid d.p_3)\backslash d$$

$$(p_1 \frown p_2) \frown p_3$$

Condizionale

if $x = v_i$ then c_1 else c_2

assumiamo p_1 modella c_1

p_2 modella c_2

riceve lo stato della variabile
e poi sceglie di conseguenza

$$xr_i.p_1 + \sum_{j \neq i} xr_j.p_2$$

Condizionale

Iterazione

while $x = v_i$ do c

assumiamo p modella c

riceve lo stato della variabile
e poi sceglie di conseguenza, eventualmente ricorrendo

$$\mathbf{rec}\ y.\ xr_i.(p[\phi_d]|d.y)\backslash d + \sum_{j \neq i} xr_j.\mathsf{Done}$$

$$Y \triangleq xr_i.(p[\phi_d]|d.Y)\backslash d + \sum_{j \neq i} xr_j.\mathsf{Done}$$

$$Y \triangleq xr_i.(p \frown Y) + \sum_{j \neq i} xr_j.\mathsf{Done}$$

Riassumendo

tutti i canali di comunicazione con le variabili
devono essere ristretti per garantire che le richieste di lettura/scrittura siano sincronizzate

$$p\backslash\{xw_1, xr_1, \cdots\}$$

sono possibili diverse ottimizzazioni:

prefisso d'azione invece del collegamento per la composizione sequenziale
guardie più espressive
rimuovere le transizioni silenziose

Example: optimisation

$$x := 1; y := 2$$

$$\overline{xw}_1.\overline{done} \quad \frown \quad \overline{yw}_2.\overline{done}$$

$$(\ (\overline{xw}_1.\overline{done})[\phi_d] \mid d.\overline{yw}_2.\overline{done}\ )\backslash d$$

$$\overline{xw}_1.\overline{yw}_2.\overline{done}$$