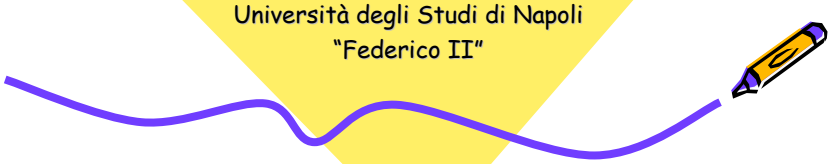
Nondeterminismo e Parallelismo (Concorrenza)

Aniello Murano
Università degli Studi di Napoli
"Federico II"



Prefazione

- In questa lezione introduciamo l'ultimo argomento del corso:
  Sistemi e Programmi Nondeterministici e Paralleli (Concorrenti) e alla loro semantica
- Inizieremo mostrando un semplice linguaggio parallelo (IMPP) ottenuto estendendo il linguaggio IMP
- Poi analizzeremo modelli di comunicazione tra sistemi che condividono variabili. Saranno trattati i seguenti argomenti:
  - Linguaggio di Dijkstra dei comandi con guardia (**IMPCG**);
  - Communicating Sequential Processes (**CSP**) di Tony Hoare
  - Calculus of Communicating system (**CCS**) di Robin Milner
- Infine, in riferimento alla verifica di sistemi introdotta nella lezione precedente, brevemente mostreremo:
  - Una Logica modale da utilizzare come linguaggio di specifica
  - Un algoritmo per la verifica del soddisfacimento di una specifica da parte di un processo a stati finiti
- Tutti questi strumenti sono alla base dello studio degli strumenti di verifica formale dei sistemi paralleli !!!

Linguaggio IMP parallelo

- Un semplice linguaggio parallelo può essere introdotto estendo l'insieme dei comandi del linguaggio imperativo IMP con una operazione di composizione parallela $\mathbf{c_0 \parallel c_1}$:
- c::= skip | X:=a |c_0;c_1| if b then c_0 else c_1| while b do c | $c_0 \parallel c_1$
- Semantica informale:
  - Dati c_0 e c_1, l'esecuzione della loro composizione parallela $c_0 \parallel c_1$ avverrà come se c_0 e c_1 fossero eseguiti contemporaneamente!
- Osservazioni:
  - Cosa succede se c_0 e c_1 accedono alla stessa variabile (per esempio due nuovi assegnamenti alla stessa variabile?
  - Uno dei due comandi eseguirà con successo l'assegnamento eventualmente seguito dall'altro assegnamento;
  - Questo significa che
    1. L'interpretazione dei comandi non è certa !
    2. non possiamo descrivere l'esecuzione di comandi paralleli utilizzando la stessa relazione fra configurazioni di comandi e stati finali finora utilizzata per IMP.

Semantica dell'operazione parallela

- La semantica operazionale della composizione parallela è data dalle seguenti regole:

$$\frac{\langle c_0, \sigma\rangle \rightarrow_1 \sigma'}{\langle c_0 \parallel c_1, \sigma\rangle \rightarrow_1 \langle c_1, \sigma'\rangle} \qquad \frac{\langle c_0, \sigma\rangle \rightarrow_1 \langle c_0', \sigma'\rangle}{\langle c_0 \parallel c_1, \sigma\rangle \rightarrow_1 \langle c_0' \parallel c_1, \sigma'\rangle}$$

$$\frac{\langle c_1, \sigma\rangle \rightarrow_1 \sigma'}{\langle c_0 \parallel c_1, \sigma\rangle \rightarrow_1 \langle c_0, \sigma'\rangle} \qquad \frac{\langle c_1, \sigma\rangle \rightarrow_1 \langle c_1', \sigma'\rangle}{\langle c_0 \parallel c_1, \sigma\rangle \rightarrow_1 \langle c_0 \parallel c_1', \sigma'\rangle}$$

- La relazione di esecuzione rappresenta singoli passi non interrompibili in modo da permettere ad un comando di modificare lo stato di un altro comando con cui è posto in parallelo.
- Le prime due regole si riferiscono al caso in cui c_0 sia eseguita prima di c_1 e le ultime due al caso simmetrico (c_1 prima di c_0)
- La prima e la terza regola si riferiscono al caso in cui l'esecuzione di c_0 e c_1 siano terminate, mentre la seconda e la quarta al caso che esse non lo siano

Comunicazione e nondeterminismo

- In una operazione di composizione parallela $\mathbf{c_0} \parallel \mathbf{c_1}$, se c_0 e c_1 hanno locazioni in comune, allora le loro esecuzioni possono risultare reciprocamente influenzate.
- Si può pensare che i due comandi comunichino tramite queste locazioni (**comunicazione tramite variabili condivise**)
- Come si vede dalle regole semantiche date per la composizione parallela, l'interpretazione di IMPP non è certa. Per esempio, per il seguente comando

  $$c \equiv (X:=0 \parallel X:=1);\ \text{if } X:=0 \text{ then } c_0 \text{ else } c_1$$

  non possiamo sapere se questo si comporterà come c_0 o come c_1
- Questa incertezza viene chiamata **nondeterminismo**.
- È importante sapere che nella programmazione parallela il nondeterminismo è un elemento ineliminabile. Questo vale non solo per i programmi semplici presentati in questo corso, ma anche per i complessi sistemi utilizzati nella realtà.

Vantaggi del nondeterminismo

- Spesso il raggiungimento di un certo risultato non dipende da quale strada viene utilizzata, fra quelle disponibili.
- Un utilizzo "disciplinato" del nondeterminismo può portare allo sviluppo di algoritmi più efficienti.
- Il linguaggio dei **comandi con guardia di Dijkstra** è basato su questa filosofia: non è richiesto al programmatore di specificare nei dettali il metodo di soluzione.

- Edsger Wybe Dijkstra (1930–2002) was a Dutch computer scientist.
- He received the 1972 A. M. Turing Award for fundamental contributions in the area of programming languages (**ALGOL**, late 1950's).
- **Dijkstra's algorithm:** an algorithm that solves the single-source shortest path problem for a directed graph with nonnegative edge weights.

Sintassi del linguaggio di Dijkstra

- Questo linguaggio contiene le stesse espressioni aritmetiche e booleane di IMP, (Aexp e Bexp). Inoltre, possiede due nuove categorie sintattiche: i comandi (c) e i comandi con guardia (gc).
- La sintassi dei comandi è data dalle seguenti regole:

  c::= skip | abort | X:=a| c_0;c_1 | if gc fi | do gc od
- La sintassi dei comandi con guardia è data dalle seguenti regole:

  gc::= b $\rightarrow$ c | gc_0 [] gc_1
- [] è l'operatore alternativo. Un comando con guardia solitamente ha la forma

  ($b_1 \rightarrow c_1$) [] … [] ($b_n \rightarrow c_n$)
- In questo caso le b_i sono chiamate guardie. Se b_i è true allora c_i può essere eseguito. Se nessuna guardia è valutata true il comando fallisce. Altrimenti non deterministicamente viene eseguito uno tra i c_i abilitati ad essere eseguiti.

Significato degli altri comandi

- Sintassi dei comandi e dei comandi con guardia:

  c::= skip | abort | X:=a| c_0;c_1 | if gc fi | do gc od

  gc::= b $\rightarrow$ c | gc_0 [] gc_1
- Skip, assegnamento e composizione sequenziale hanno lo stesso significato dato in IMP;
- Abort non produce uno stato finale a partire da qualsiasi stato
- Il comando **if gc fi** si comporta come il comando di guardia gc se gc non fallisce, altrimenti si comporta come abort
- Il comando **do gc od** si comporta come l'esecuzione ripetuta del comando di guardia gc fino a che gc non fallisce. Esso si comporta come skip quando gc fallisce.

Regole di derivazione per c e gc

- Una configurazione di un comando è <c, σ> oppure solo σ . Si raggiunge la configurazione σ solo quando uno stato è stato completamento eseguito.
- Una configurazione iniziale di un comando con guardia è <gc, σ> e un solo passo di esecuzione può portare a una configurazione di comando o ad una configurazione di fallimento (fail)
- Regole per i comandi:

$$\langle \mathbf{skip}, \sigma \rangle \to \sigma \qquad \frac{\langle a, \sigma \rangle \to n}{\langle X := a, \sigma \rangle \to \sigma[n/X]}$$

$$\frac{\langle c_0, \sigma \rangle \to \sigma'}{\langle c_0; c_1, \sigma \rangle \to \langle c_1, \sigma' \rangle} \qquad \frac{\langle c_0, \sigma \rangle \to \langle c_0', \sigma' \rangle}{\langle c_0; c_1, \sigma \rangle \to \langle c_0'; c_1, \sigma' \rangle} \qquad \frac{\langle gc, \sigma \rangle \to \langle c, \sigma' \rangle}{\langle \mathbf{if}\ gc\ \mathbf{fi}, \sigma \rangle \to \langle c, \sigma' \rangle}$$

$$\frac{\langle gc, \sigma \rangle \to \mathbf{fail}}{\langle\ \mathbf{do}\ gc\ \mathbf{od}, \sigma \rangle \to \sigma} \qquad \frac{\langle gc, \sigma \rangle \to \langle c, \sigma' \rangle}{\langle\ \mathbf{do}\ gc\ \mathbf{od}, \sigma \rangle \to \langle c;\ \mathbf{do}\ gc\ \mathbf{od}, \sigma' \rangle}$$

Regole per i comandi con guardia

$$\frac{\langle b, \sigma \rangle \to \mathbf{true}}{\langle b \to c, \sigma \rangle \to \langle c, \sigma \rangle}$$

$$\frac{\langle gc_0, \sigma \rangle \to \langle c, \sigma' \rangle}{\langle gc_0 [\!] gc_1, \sigma \rangle \to \langle c, \sigma' \rangle} \qquad \frac{\langle gc_1, \sigma \rangle \to \langle c, \sigma' \rangle}{\langle gc_0 [\!] gc_1, \sigma \rangle \to \langle c, \sigma' \rangle}$$

$$\frac{\langle b, \sigma \rangle \to \mathbf{false}}{\langle b \to c, \sigma \rangle \to \mathbf{fail}} \qquad \frac{\langle gc_0, \sigma \rangle \to \mathbf{fail}\ \langle gc_1, \sigma \rangle \to \mathbf{fail}}{\langle gc_0 [\!] gc_1, \sigma \rangle \to \mathbf{fail}}$$

- Si noti come i comandi alternativi introducano il nondeterminismo

Equivalenza di IMPG con gli altri comandi di IMP

- Conditionals and while-loops of IMP can easily be modeled in IMPGC.
- We reintroduce them as abbreviations of IMPGC commands:
- if b then c_0 else $c_1 \equiv$ if $b \rightarrow c_0 \; [] \; \neg b \rightarrow C_1$ fi and
- while b do c $\equiv$ do $b \rightarrow c$ od

Come utilizzare IMPGC

- Given an expression in pseudocodice:

  if a $\geq$ b then print "More or equal";
  else if a < b then print "Less";
- The equivalent in guarded commands is:

  if
  a $\geq$ b $\rightarrow$ print "More or equal" []
  a < b $\rightarrow$ print "Less"
  fi
- The power of guarded commands is illustrated in the following expression:

  if
  a $\geq$ b $\rightarrow$ print "More or equal" []
  a $\leq$ b $\rightarrow$ print "Less or equal"
  fi
- When a = b, the result of command can be one "More or equal" or "Less or equal".

Esempio (1)

- Il seguente esempio mostra un utilizzo efficiente del comando if del linguaggio IMPCG per calcolare il massimo tra due locazioni

$$\begin{array}{l}\textbf{if}\\ X \geq Y \rightarrow MAX := X\\ [\!]\\ Y \geq X \rightarrow MAX := Y\\ \textbf{fi}\end{array}$$

Esempio (2)

- Il seguente esempio mostra un modo elegante dell'applicazione dell'algoritmo di Euclide per il calcolo del massimo comun divisore di due numeri, utilizzo il comando do del linguaggio IMPCG

$$\begin{array}{l}\textbf{do}\\ X > Y \rightarrow X := X - Y\\ [\!]\\ Y > X \rightarrow Y := Y - X\\ \textbf{od}\end{array}$$

Osservazioni sulla terminazione

- For every IMP command c and for every state σ there exists a configuration $\langle c', \sigma' \rangle$ (or σ') such that $\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$ (or $\langle c, \sigma \rangle \rightarrow_1 \sigma'$, respect.)
  - The proof is a straightforward induction on the structure of c.
- In IMP, every command c has from every state σ precisely one computation.
  - The prof follows by structural induction on commands.
- Le precedenti proprietà non valgono per IMPGC. La seconda è stata già discussa in precedenza. per mostrare la non valiedità della prima è sufficiente utilizzare il seguente comando
- $c \equiv \langle$if false $\rightarrow$ skip, $\sigma \rangle$. Infatti, per le regole date, non esiste una configurazione $\langle c', \sigma' \rangle$ (o uno stato σ) tale che $\langle c, \sigma \rangle \rightarrow_1 \langle c', \sigma' \rangle$ ($\langle c, \sigma \rangle \rightarrow_1 \sigma'$).

Esercizio 1

- Si dimostri che per ogni espressione booleana b e comando c, i comandi

$$\text{do } b \rightarrow c \,[]\, b \rightarrow c \text{ od}$$

$$\text{if } b \rightarrow (c; \text{do } b \rightarrow c \,[]\, b \rightarrow c \text{ od}) \,[]\, \neg b \rightarrow \text{skip fi}$$

sono semanticamente equivaelnti

La valutazione di $\neg$b è semplicemente data dall'opposto della valutazione di b

Esercizio 2

- Si dimostri la terminazione del seguente programma

$$\mathbf{do}\ (2|X \to X := (3 \times X)/2) [] (3|X \to X := (5 \times X)/3)\ \mathbf{od}$$

dove 2|X significa 2 divide X e (5 x X)/3 significa 5 x X diviso 3