LOGICA DI HOARE

INTRODUZIONE

- Linguaggi di programmazione: sintassi e semantica
- Sintassi fornita con strumenti formali:
  - es. grammatica in BNF
- Semantica spesso data in modo informale
  - di stile operazionale
  - comprensibile per non esperti
  - lascia spazio a ambiguità
  - non sufficiente per applicazioni “critiche”
- Necessità di ***dimostrare*** proprietà relative a programmi
- Varie semantiche formali: operazionale, denotazionale, **assiomatica**
- Semantica assiomatica di Hoare [1969] - Dijkstra [1976]

LINGUAGGIO IMPERATIVO MINIMO

- Espressioni:

  Exp ::= Const | Ide | (Exp) | Exp Op Exp | not Exp

  Op ::= + | - | * | div | mod |

  = |≠ | < | ≤ | > | ≥ | or | and

  Const ::= Num | Bool

- Comandi:

  Com ::= **skip** | Ide_List := Exp_List |

  Com ; Com |

  if Exp **then** Com **else** Com **fi** |

  while Exp **do** Com **endw**

  Ide_List ::= Ide | Ide, Ide_List

  Exp_List ::= Exp | Exp, Exp_List

CONCETTO DI STATO

- Modelliamo lo stato di un programma come una funzione da Identificatori di variabile a valori (booleani e interi)
- $\sigma : Ide \rightarrow B \cup Z$
- Poiché in uno stato l'insieme delle variabili è finito, una tipica rappresentazione è

  $\sigma = \{<x,18>,<y,true>,<z,-8>\}$
- con $E(e,\sigma)$ indichiamo il valore dell'espressione e nello stato σ

  ***E(_,_)* : funzione di interpretazione semantica**
- Esempi nello stato precedente
  - $E(x+z,\sigma) = 10$
  - $E(y,\sigma) = true$

SEMANTICA DELLE ESPRESSIONI

$$
\begin{array}{rcll}
\mathcal{E}(true, \sigma) & = & \mathbf{tt} & \\
\mathcal{E}(false, \sigma) & = & \mathbf{ff} & \\
\mathcal{E}(n, \sigma) & = & \mathbf{n} & \text{se} \quad n \in \texttt{Num} \\
\mathcal{E}(x, \sigma) & = & \sigma(x) & \text{se} \quad x \in \texttt{Ide} \\
\mathcal{E}(E \ op \ E', \sigma) & = & \mathcal{E}(E, \sigma) \ \mathbf{op} \ \mathcal{E}(E', \sigma) & \text{se} \quad op \in \{+, -, div, mod, =, \neq, <, >, \leq, \geq\} \\
 & & & \quad \mathcal{E}(E, \sigma) \in \mathbb{Z} \ \text{ e } \ \mathcal{E}(E', \sigma) \in \mathbb{Z} \\
\mathcal{E}(E \ op \ E', \sigma) & = & \mathcal{E}(E, \sigma) \ \mathbf{op} \ \mathcal{E}(E', \sigma) & \text{se} \quad op \in \{and, or, =, \neq\} \\
 & & & \quad \mathcal{E}(E, \sigma) \in \mathbb{B} \ \text{ e } \ \mathcal{E}(E', \sigma) \in \mathbb{B} \\
\mathcal{E}(not \ E, \sigma) & = & \neg \mathcal{E}(E, \sigma) & \text{se} \quad \mathcal{E}(E, \sigma) \in \mathbb{B} \\
\mathcal{E}((E), \sigma) & = & \mathcal{E}(E, \sigma) &
\end{array}
$$

SIGNIFICATO INFORMALE DEI COMANDI

- L'esecuzione di **skip** a partire dallo stato σ porta nello stato σ
- L'esecuzione dell'assegnamento **x1, ..., xn := E1, ..., En** a partire dallo stato σ porta nello stato σ[*E(E1,σ)*/x1, ..., *E(En,σ)*/xn]
- L'esecuzione di **C;C'** a partire dallo stato σ porta nello stato σ' ottenuto eseguendo **C'** a partire dallo stato σ" ottenuto dall'esecuzione di **C** nello stato σ.
- L'esecuzione di **if E then C else C' fi** a partire da σ porta nello stato σ' che si ottiene dall'esecuzione di **C** in σ, se *E(E,σ)= **tt**,* e dall'esecuzione di **C'** in σ, *se E(E,σ)= **ff***
- L'esecuzione del comando **while E do C endw** a partire da σ porta in σ se *E(E,σ)= **ff**,* altrimenti porta nello stato σ' ottenuto dall'esecuzione di **while E do C endw** a partire dallo stato σ" ottenuto con l'esecuzione di **C** nello stato σ.

TRIPLE DI HOARE

- {Q} C {R}

  dove Q e R sono *asserzioni* ovvero formule ben formate in cui compaiono le variabili dello stato e C è un comando
- Significato intuitivo:

  a partire da uno stato che soddisfi Q, l'esecuzione del comando C termina in uno stato che soddisfa R
- {x>1} x := x + 1 {x>2}

NOTAZIONE

- *free*(P) è l'insieme delle variabili libere nell'asserzione P

  p.e. *Free*($x>y \wedge z \leq 1$) = {x,y,z}
- Sia P un'asserzione e σ uno stato.

  $P\sigma$ indica l'asserzione P istanziata sullo stato σ, ovvero in cui a tutte le variabili sono sostituiti i loro valori nello stato

  p.e. $\sigma = \{\langle x,18\rangle,\langle y,true\rangle,\langle z,-8\rangle\}$

  $P = (x > 0 \wedge z < 0)$

  $P\sigma = (18 > 0 \wedge -8 < 0) \equiv T$

PROPRIETÀ

- $\sigma \models P$ (lo stato σ soddisfa l'asserzione P) sse $P\sigma \equiv T$
- Con $\not\models$ indichiamo la non soddisfazione
- Con {P} indichiamo l'insieme degli stati che soddisfano P
- Sia E una espressione, P un'asserzione
  - Se per ogni $x \in \mathit{free}(E)$, $\sigma(x) = \sigma'(x)$ allora

    $E(E, \sigma) = E(E, \sigma')$
  - Se per ogni $x \in \mathit{free}(P)$, $\sigma(x) = \sigma'(x)$ allora

    $\sigma \models P$ se e solo se $\sigma' \models P$
  - Per ogni variabile x

    $\sigma[E(E, \sigma)/x] \models P$ se e solo se $\sigma \models P[E/x]$

GENERALITÀ SULLE TRIPLE DI HOARE

- Data la tripla di Hoare {Q} C {R}
  - Q è detta precondizione
  - R è detta postcondizione
  - La tripla è soddisfatta se dato uno stato σ che soddisfa Q:
  1. L'esecuzione del comando C a partire dallo stato σ termina producendo lo stato σ'
  2. Lo stato σ' soddisfa R

INTERPRETAZIONE DELLE TRIPLE DI HOARE (1)

- *Semantica*: dati Q e C, determinare R in modo che sia soddisfatta {Q} C {R} è un modo per descrivere il comportamento di C.

  Per esempio sia Q ≡ {x>10} e

  C ≡ **if** (x < 10) **then** x := 1 **else** x :=2 **fi**

  allora R ≡ {x = 2} descrive il comportamento del comando

- *Correttezza*: dati Q, C ed R, dimostrare che la tripla

  {Q} C {R} è soddisfatta corrisponde ad una dimostrazione di correttezza del comando C rispetto alla pre e alla postcondizione

INTERPRETAZIONE DELLE TRIPLE DI HOARE (2)

- *Specifica*. Data una precondizione Q e una postcondizione R determinare un comando C che soddisfa la tripla {Q} C {R} equivale a scrivere il programma che realizza le specifiche Q ed R.
- Per esempio le specifiche $Q \equiv \{x > 0\}$ e

  $R \equiv \{y = 2 \times x\}$ possono essere soddisfatte dal comando

  y := x * 2

  oppure dal comando

  y := x + x

  oppure dal comando

  y := x; y := y * 2

  ecc..

SISTEMA DI PROVA (REGOLE DI INFERENZA)

(pre-post o conseguenza)

$$\frac{P \Rightarrow P' \quad \{P'\}\ C\ \{R'\} \quad R' \Rightarrow R}{\{P\}\ C\ \{R\}}$$

(pre)

$$\frac{P \Rightarrow P' \quad \{P'\}\ C\ \{R\}}{\{P\}\ C\ \{R\}}$$

(post)

$$\frac{\{P\}\ C\ \{R'\} \quad R' \Rightarrow R}{\{P\}\ C\ \{R\}}$$

SKIP E ASSEGNAMENTO

- Comando vuoto (**skip**)

$$\{P\}\ \textbf{skip}\ \{P\}$$

- Assegnamento

$$\{\mathit{def}(E) \wedge P[E/x]\}\ x := E\ \{P\}$$

  dove *def*(E) è vera se il calcolo dell'espressione E è ben definito, ovvero termina senza errori (p.e. divisioni per zero, out of bound per gli array ecc.).
- L'idea è che affinché l'asserzione E sia soddisfatta dopo l'assegnamento, la stessa asserzione deve essere soddisfatta dallo stato precedente l'assegnamento quando all'identificatore di variabile è sostituita l'espressione.

DEFINIZIONE DI ESPRESSIONI

- Introduciamo, per induzione strutturale, la funzione *def(E)* che, applicata a un'espressione *E*, restituisce una asserzione tale che $\sigma \models def(E)$ se esiste v tale che E(E, σ) = v.

$$
\begin{array}{rcll}
def(c) & = & \mathbf{tt} & \text{se} \quad c \in \texttt{Num o } c \in \texttt{Bool} \\
def(x) & = & \mathbf{tt} & \text{se} \quad x \in \texttt{Ide} \\
def(E \ op \ E') & = & def(E) \wedge def(E') & \text{se} \quad op \in \left\{ \begin{array}{l} +, -, <, >, \leq, \geq \\ =, \neq, and, or \end{array} \right\} \\
def(E \ op \ E') & = & def(E) \wedge def(E') \wedge E' \neq 0 & \text{se} \quad op \in \{div, mod\} \\
def(not \ E) & = & def(E) & \\
def((E)) & = & def(E) &
\end{array}
$$

ASSEGNAMENTO MULTIPLO

- L'assioma per l'assegnamento multiplo generalizza quello per l'assegnamento singolo

$$\{def(E_1) \wedge \ldots \wedge def(E_k) \wedge P^{E_1,\ldots,E_k}_{x_1,\ldots,x_k}\}\ x_1,\ldots,x_k := E_1,\ldots,E_k\ \{P\} \qquad (ass)$$

- Tutte le espressioni vengono valutate PRIMA di tutti gli assegnamenti

ESEMPI:

- Verificare le seguenti triple:
  - $\{x = 5\}\ x := x + 1\ \{x > 2\}$
  - $\{s = (\Sigma\, i \in [0, x).\ i\}\ x,s := x+1,\ s+x\ \{s = (\Sigma\, i \in [0, x).\ i\}$

SEQUENZA DI COMANDI

- Sequenza

$$\frac{\{P\}\ C\ \{R\} \qquad \{R\}\ C'\ \{Q\}}{\{P\}\ C\ ;C'\ \{Q\}}$$

ESEMPI:

- Si verifichino la seguente tripla
  - $\{x >= y - 1\}\ x := x+1;\ y := y - 1\ \{x > y\}$
- Determinare *E* affinché la seguente tripla sia corretta
  - $\{x = n \wedge y = m\}\ t := E;\ \ x := y;\ \ y := t\ \ \{x = m \wedge y = n\}$

COMANDO CONDIZIONALE

- Condizionale

$$\frac{P \Rightarrow \mathit{def}(E) \qquad \{P \wedge E\}\ C\ \{Q\} \qquad \{P \wedge \sim E\}\ C'\ \{Q\}}{\{P\}\ \textbf{if}\ E\ \textbf{then}\ C\ \textbf{else}\ C'\ \textbf{fi}\ \{Q\}}$$

- Esempio: si verifichi la seguente tripla:

$\{x >= 0 \wedge y > 0\}$

if x div y > 0

then z := x

else z := y

fi

$\{z = \max(x, y)\}$

COMANDO ITERATIVO

- While

$$\frac{P \Rightarrow Inv \wedge def(E) \qquad Inv \wedge \sim E \Rightarrow Q \qquad Inv \Rightarrow t \geq 0 \qquad \{Inv \wedge E\}\ C\ \{Inv \wedge def(E)\} \qquad \{Inv \wedge E \wedge t = V\}\ C\ \{t < V\}}{\{P\}\ \textbf{while}\ E\ \textbf{do}\ C\ \textbf{endw}\ \{Q\}}$$

- Con
  - t funzione di terminazione
  - Inv asserzione invariante