Verifica dei programmi

Studia le relazioni che valgono fra oggetti definiti in linguaggi di descrizione di diverso livello

- Un linguaggio di programmazione $\mathcal{L}$ che può essere compilato ed eseguito
- Un linguaggio di specifica $\mathcal{SP}$ di più alto livello per rappresentare requisiti e specifiche

Data una specifica $SP \in \mathcal{SP}$ ed un programma $S \in \mathcal{L}$ controllare se essi sono compatibili cioé se il significato di S soddisfa SP

Verifica dei programmi (ctnd.)

Tratteremo il problema in maniera generale

- Introduzione di alcuni linguaggi di programmazione e relativa semantica

- Definizione delle specifiche che descrivono i risultati di correttezza che vogliamo provare

- Introduzione dei metodi di verifica classici

Alcuni richiami di logica del I Ordine - Sintassi

Simboli logici:

- Connettivi proposizionali: $\neg, \wedge, \vee, \supset, \equiv$
- Costanti logiche: *true*, *false*
- Quantificatori: $\forall, \exists$
- Uguaglianza: $=$
- Parentesi: (,)
- Var: un'infinità contabile di variabili

Simboli non logici: una base $B = (\mathcal{F}, \mathcal{P})$

- $\mathcal{F}$: collezione contabile di simboli funzionali
- $\mathcal{P}$: collezione contabile di simboli relazionali

Alcuni richiami di logica del I Ordine - Sintassi (ctnd 1.)

L'insieme T_B dei termini del linguaggio del I ordine di base B viene costruito induttivamente

- $x \in Var$, allora $x \in T_B$
- $c \in \mathcal{F}$, allora $c \in T_B$
- $f \in \mathcal{F}$ con arietà $n \geq 1$ e $t_1, \ldots, t_n \in T_B$, allora $f(t_1, \ldots, t_n) \in T_B$

Alcuni richiami di logica del I Ordine - Sintassi (ctnd 2.)

L'insieme WFF_B delle formule ben formate del linguaggio del I ordine di base B viene costruito induttivamente

- $t_1, t_2 \in T_B$, allora $t_1 = t_2 \in WFF_B$
- $c \in \mathcal{P}$, allora $c \in WFF_B$
- $R \in \mathcal{P}$ con arietà $n \geq 1$ e $t_1, \ldots, t_n \in T_B$, allora $R(t_1, \ldots, t_n) \in WFF_B$
- $\varphi, \psi \in WFF_B$, allora
  - $\neg\varphi \in WFF_B$
  - $\varphi \circ \psi \in WFF_B, \qquad \circ \in \{\neg, \wedge, \vee, \supset, \equiv\}$
- $x \in Var$ e $\varphi \in WFF_B$, allora $(\forall x)\varphi \in WFF_B$ e $(\exists x)\varphi$ sono in WFF_B

Alcuni richiami di logica del I Ordine - Semantica

Sia $B = (\mathcal{F}, \mathcal{P})$. Un'interpretazione per B, $\boldsymbol{\mathcal{I}}$, è una coppia ordinata $(\boldsymbol{\mathcal{D}}, \boldsymbol{\mathcal{I}}_0)$ tale che

- $\boldsymbol{\mathcal{D}} \neq \emptyset$
- $\boldsymbol{\mathcal{I}}_0$ è una mappa sui simboli di B definita come segue:
  - $c \in \mathcal{F}$, $\boldsymbol{\mathcal{I}}_0(c) = d$, $d \in \boldsymbol{\mathcal{D}}$
  - $c \in \mathcal{P}$, $\boldsymbol{\mathcal{I}}_0(c) \in \{\mathbf{true}, \mathbf{false}\}$
  - $f \in \mathcal{F}$ con arietà $n \geq 1$, $\boldsymbol{\mathcal{I}}_0(f) : \boldsymbol{\mathcal{D}}^n \rightarrow \boldsymbol{\mathcal{D}}$
  - $R \in \mathcal{P}$ con arietà $n \geq 1$, $\boldsymbol{\mathcal{I}}_0(R) : \boldsymbol{\mathcal{D}}^n \rightarrow \{\mathbf{true}, \mathbf{false}\}$

Assegnamento $\sigma : Var \rightarrow \boldsymbol{\mathcal{D}}$
L'insieme degli assegnamenti per $\boldsymbol{\mathcal{I}}$ è $\Sigma_{\boldsymbol{\mathcal{I}}}$
Funzionale che mappa ogni termine $t \in T_B$ in $\boldsymbol{\mathcal{I}}(t) : \Sigma_{\boldsymbol{\mathcal{I}}} \rightarrow \boldsymbol{\mathcal{D}}$ ed ogni formula $w \in WFF_B$ in $\boldsymbol{\mathcal{I}}(w) : \Sigma_{\boldsymbol{\mathcal{I}}} \rightarrow \{\mathbf{true}, \mathbf{false}\}$

Linguaggio di programmazione flowchart

- Linguaggio di programmazione imperativo costruito a partire da assegnamenti e salti (condizionali ed incondizionali)
- Viene chiamato "flowchart" perché può essere anche rappresentato come un diagramma di flusso
- La sua definizione si basa sulla logica predicativa

Simboli utilizzati dai programmi flowchart

- armamentario comune in logica predicativa (simboli logici e non logici)
- $\{:=, ;, :, \texttt{goto}, \texttt{if}, \texttt{then}, \texttt{else}, \texttt{fi}\}$
- un'infinità contabile di etichette $\texttt{Lab}$ con due elementi distinti: $begin$ ed end

Preliminari alla costruzione dei programmi flowchart

1. Termini e formule della logica predicativa
2. Comandi
   - Assegnamento parallelo:
     $l_1 :\ (x_1, \ldots, x_n) := (t_1, \ldots, t_n)\ ;$ goto l_2, $\qquad n \geq 1$
     con $l_1, l_2 \in$ Lab, $l_1 \neq end$, $x_1, \ldots, x_n$ variabili distinte in Var, $t_1, \ldots, t_n$ termini della logica predicativa

   - Jump condizionale:
     $l_1 :$ `if` e `then goto` l_2 `else goto` l_3 `fi`
     con $l_1, l_2, l_3 \in$ Lab, $l_1 \neq end$, $l_2 \neq l_3$, e una formula della logica predicativa senza quantificatori

Costruzione dei programmi flowchart

Un programma flowchart sulla base $B = (\mathcal{F}, \mathcal{P})$ viene definito come una sequenza finita e non vuota di comandi su B separati da ;

Esso soddisfa le seguenti condizioni:

1. Le etichette delle occorrenze definenti sono tutte differenti
2. Ogni etichetta in una occorrenza applicata diversa da end ha anche una occorrenza definente
3. L'etichetta $begin$ ha un'occorrenza definente

L'insieme di tutti i programmi flowchart viene indicato con L_1^B

Semantica di un programma flowchart

Sia $B = (\mathcal{F}, \mathcal{P})$ una base, $\mathcal{I}$ una interpretazione di B.

Costruiamo un funzionale $\mathcal{M}_{\mathcal{I}}$ che associ ad ogni oggetto *programma flowchart* il suo significato

Stati

La collezione $\Sigma_{\mathcal{I}}$ degli assegnamenti dell'interpretazione $\mathcal{I}$

Semantica di un programma flowchart (ctnd.)

Configurazioni

Una configurazione per una base B, un'interpretazione $\boldsymbol{\mathcal{I}}$ e un programma S è una coppia (l_i, σ_i) in
$\{l \in \mathtt{Lab} | l$ occorre in $S\} \times \Sigma_{\boldsymbol{\mathcal{I}}}$

Come correlare insieme queste configurazioni?

E' possibile definire una relazione sulle configurazioni che descriva le transizioni eseguite durante l'esecuzione del programma

Relazione di transizione

Sia $B = (\mathcal{F}, \mathcal{P})$ una base, $\boldsymbol{\mathcal{I}}$ una interpretazione di B ed S un programma flowchart

La relazione di transizione $\Rightarrow^S$ sull'insieme di configurazioni di S viene definita

$$(l_1, \sigma_1) \Rightarrow^S (l_2, \sigma_2)$$

se e solo se vale una delle seguenti tre condizioni

1. C'è in S un comando

   $l_1 : \ (x_1, \ldots, x_n) := (t_1, \ldots, t_n);$ `goto` l_2

   e

   $\sigma_2 = \sigma_1[x_1/\boldsymbol{\mathcal{I}}(t_1)(\sigma_1), \ldots, x_n/\boldsymbol{\mathcal{I}}(t_n)(\sigma_1)]$

Relazione di transizione (ctnd.)

2. C'è in S un comando
   $l_1 :$ `if` e `then goto` l_2 `else goto` l_3 `fi`,
   $\boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{true}$ e $\sigma_2 = \sigma_1$

3. C'è in S un comando
   $l_1 :$ `if` e `then goto` l_3 `else goto` l_2 `fi`,
   $\boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{false}$ e $\sigma_2 = \sigma_1$

Sequenze di computazione dei programmi flowchart

Sia S un programma flowchart e $\sigma \in \Sigma_{\boldsymbol{\mathcal{I}}}$

Una sequenza di computazione di S su input σ è una sequenza possibilmente infinita di configurazioni
$(l_0, \sigma_0), \ldots, (l_k, \sigma_k), [\ldots]$
tale che

$l_0 = begin$
$\sigma_0 = \sigma$
e per ogni coppia di configurazioni della sequenza
$(l_i, \sigma_i), (l_{i+1}, \sigma_{i+1})$,

$$(l_i, \sigma_i) \Rightarrow^S (l_{i+1}, \sigma_{i+1}) \text{ per } i \geq 0$$

Computazioni dei programmi flowchart

Una sequenza di computazione di un programma S su un input σ che è

- infinita
- o finisce con una configurazione (l_k, σ_k) tale che $l_k = end$

viene detta computazione di S su input σ

Qualora la computazione sia finita, σ_k viene detto lo *stato di output*

Il programma S termina su input σ se
$(begin, \sigma) \Rightarrow^{S*} (end, \sigma_k)$

Semantica operazionale dei programmi flowchart

Sia $S \in \mathrm{L}_1^B$ ed $\boldsymbol{\mathcal{I}}$ una interpretazione di B

Il significato del programma di flowchart S (nell'interpretazione $\boldsymbol{\mathcal{I}}$)
è la funzione $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S) : \Sigma_{\boldsymbol{\mathcal{I}}} \rightsquigarrow \Sigma_{\boldsymbol{\mathcal{I}}}$ definita da

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma) = \begin{cases} \sigma' & \text{se, con input } \sigma, S \text{ termina con output } \sigma' \\ \text{indefinita} & \text{altrimenti} \end{cases}$$

Programmi while

I programmi while sono di tipo imperativo e costituiscono il nucleo di ogni programma Algol-like.

Essi sono basati sulla nozione di

- assegnamento
- if-then-else statements
- while loops

La loro definizione è basata sulla logica predicativa

Sintassi dei programmi while

Sia $B = (\mathcal{F}, \mathcal{P})$ una base per la logica predicativa

Aggiungiamo i seguenti simboli

$\{:=, ;, \texttt{if}, \texttt{then}, \texttt{else}, \texttt{fi}, \texttt{while}, \texttt{do}, \texttt{od}\}$

La definizione dei programmi while viene fatta per induzione

A differenza dei programmi flowchart ogni statement è un programma

Costruzione di un programma while

Sia $B = (\mathcal{F}, \mathcal{P})$ una base, allora l'insieme L_2^B dei programmi while su B viene definito induttivamente

a. Statement di assegnamento

   Per ogni variabile $x \in Var$ e ogni termine t della logica predicativa

$$x := t \qquad \text{è un programma while}$$

b. Statement composto

   Siano S_1 ed S_2 due programmi while, allora $S_1; S_2$ è un programma while

Costruzione di un programma while (ctnd.)

c. Statement condizionale

Siano S_1, S_2 due programmi while ed e una formula della logica predicativa priva di quantificatori,

allora

`if` e `then` S_1 `else` S_2 `fi`

è un programma while

d. While loop

Sia S_1 un programma while ed e una formula della logica predicativa priva di quantificatori,

allora

`while` e `do` S_1 `od`

è un programma while

Semantica di un programma while

Sia $B = (\mathcal{F}, \mathcal{P})$ una base, $\mathcal{I}$ una interpretazione di B.

Costruiamo un funzionale $\mathcal{M}_{\mathcal{I}}$ che associ ad ogni oggetto *programma while* il suo significato

Stati

La collezione $\Sigma_{\mathcal{I}}$ degli assegnamenti dell'interpretazione $\mathcal{I}$

Semantica di un programma while (ctnd.)

Configurazioni

Una configurazione per una base B e un'interpretazione $\boldsymbol{\mathcal{I}}$ di B è una coppia (S, σ) di elementi di
$(\mathrm{L}_2^B \cup \{\epsilon\}) \times \Sigma_{\boldsymbol{\mathcal{I}}}$

Come correlare insieme queste configurazioni?

E' possibile definire una relazione sulle configurazioni che descriva le transizioni eseguite durante l'esecuzione del programma

Relazione di transizione

Sia $B = (\mathcal{F}, \mathcal{P})$ una base ed $\boldsymbol{\mathcal{I}}$ una interpretazione di B,

allora la relazione di transizione $\Rightarrow$ sul prodotto cartesiano $(\mathrm{L}_2^B \cup \{\epsilon\}) \times \Sigma_{\boldsymbol{\mathcal{I}}}$ delle configurazioni possibili in B viene definita

$$(S_1, \sigma_1) \Rightarrow (S_2, \sigma_2)$$

se e solo se vale una delle seguenti condizioni

1. S_1 è il programma $x := t;\ S_2$ e $\sigma_2 = \sigma_1[x/\boldsymbol{\mathcal{I}}(t)(\sigma_1)]$

Relazione di transizione (ctnd 1.)

2. Ci sono dei programmi while S_1', S_2', S_3', e una e della logica predicativa senza quantificatori, tali che S_1 è
   `if` e `then` S_1' `else` S_2' `fi` ; S_3'
   S_2 è $\begin{cases} S_1'; S_3' & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{true} \\ S_2'; S_3' & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{false} \end{cases}$
   e $\sigma_2 = \sigma_1$
3. Ci sono dei programmi while S_1', S_2', e una e della logica predicativa senza quantificatori, tali che S_1 è
   `while` e `do` S_1'`od` ; S_2'

$$S_2 \text{ è } \begin{cases} S_1'; S_1 & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{true} \\ S_2' & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{false} \end{cases}$$

e $\sigma_2 = \sigma_1$

Relazione di transizione (ctnd 2.)

4. S_1 è il programma $x := t$,
   S_2 è il programma vuoto e $\sigma_2 = \sigma_1[x/\boldsymbol{\mathcal{I}}(t)(\sigma_1)]$

5. Ci sono programmi S_1', S_2', e una formula e della logica predicativa, priva di quantificatori, tali che S_1 è
   `if` e `then` S_1' `else` S_2' `fi`

$$S_2 \text{ è } \begin{cases} S_1' & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{true} \\ S_2' & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{false} \end{cases}$$

   e $\sigma_2 = \sigma_1$

Relazione di transizione (ctnd 3.)

6. C'è un programma S_1', e una formula e della logica predicativa, priva di quantificatori, tali che

   S_1 è `while` e `do` S_1'`od`

$$S_2 \text{ è } \begin{cases} S_1'; S_1 & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{true} \\ \epsilon & \text{se } \boldsymbol{\mathcal{I}}(e)(\sigma_1) = \mathbf{false} \end{cases}$$

   e $\sigma_2 = \sigma_1$

Sequenze di computazione

Una sequenza di computazione è una sequenza possibilmente infinita di configurazioni
$(S_0, \sigma_0), \ldots, (S_k, \sigma_k), [\ldots]$

tale che per ogni $i \geq 0$ ci sia una transizione

$(S_i, \sigma_i) \Rightarrow (S_{i+1}, \sigma_{i+1})$
dal programma S_i nello stato σ_i al programma S_{i+1} nello stato σ_{i+1}

Computazioni

Una sequenza di computazione si dice una computazione se è

- infinita
- o finisce con una configurazione (S_k, σ_k) tale che $S_k = \epsilon$

Qualora la computazione sia finita, σ_k viene detto lo *stato di output*

Semantica operazionale dei programmi while

Sia $S \in \mathrm{L}_1^B$ ed $\boldsymbol{\mathcal{I}}$ una interpretazione di B

Il significato del programma while S (nell'interpretazione $\boldsymbol{\mathcal{I}}$) è la funzione $\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S) : \Sigma_{\boldsymbol{\mathcal{I}}} \rightsquigarrow \Sigma_{\boldsymbol{\mathcal{I}}}$ definita da

$$\boldsymbol{\mathcal{M}}_{\boldsymbol{\mathcal{I}}}(S)(\sigma) = \begin{cases} \sigma' & \text{se, su input } \sigma, \ S \text{ termina con output } \sigma' \\ \text{indefinita} & \text{altrimenti} \end{cases}$$