Curs 9

2021-2022

Fundamentele limbajelor de programare

Cuprins

Limbajul IMP

Limbajul IMP

Vom implementa un limbaj care conţine:

- Expresii
  - Aritmetice
  - Booleene

```
x + 3
x >= 7
```

Limbajul IMP

Vom implementa un limbaj care conţine:

- Expresii
  - Aritmetice — `x + 3`
  - Booleene — `x >= 7`
- Instructiuni
  - De atribuire — `x = 5`

Limbajul IMP

Vom implementa un limbaj care conţine:

- Expresii
  - Aritmetice `x + 3`
  - Booleene `x >= 7`
- Instructiuni
  - De atribuire `x = 5`
  - Condiţionale `if(x >= 7, x = 5, x = 0)`

Limbajul IMP

Vom implementa un limbaj care conţine:

- Expresii
  - Aritmetice — `x + 3`
  - Booleene — `x >= 7`
- Instructiuni
  - De atribuire — `x = 5`
  - Condiţionale — `if(x >= 7, x = 5, x = 0)`
  - De ciclare — `while(x >= 7, x = x - 1)`

Limbajul IMP

Vom implementa un limbaj care conţine:

- Expresii
  - Aritmetice — `x + 3`
  - Booleene — `x >= 7`
- Instructiuni
  - De atribuire — `x = 5`
  - Condiţionale — `if(x >= 7, x = 5, x = 0)`
  - De ciclare — `while(x >= 7, x = x - 1)`
- Compunerea instrucţiunilor — `x=7; while(x>=0, x=x-1)`

Limbajul IMP

Vom implementa un limbaj care conţine:

- Expresii
  - Aritmetice — `x + 3`
  - Booleene — `x >= 7`
- Instructiuni
  - De atribuire — `x = 5`
  - Condiţionale — `if(x >= 7, x = 5, x = 0)`
  - De ciclare — `while(x >= 7, x = x - 1)`
- Compunerea instrucţiunilor — `x=7; while(x>=0, x=x-1)`
- Blocuri de instrucţiuni — `{x=7; while(x>=0, x=x-1)}`

Limbajul IMP

Exemplu

Un program în limbajul IMP

```
{x = 10 ; sum = 0;
while(0 =< x,
      {sum = sum + x; x = x-1}
      )},sum
```

- Semantica
  după executia programului, se evaluează `sum`

Sintaxa BNF a limbajului IMP

$$
\begin{aligned}
E &::= n \mid x \\
&\quad \mid E + E \mid E - E \mid E * E \\
\\
B &::= \texttt{true} \mid \texttt{false} \\
&\quad \mid E =< E \mid E >= E \mid E == E \\
&\quad \mid \texttt{not}(B) \mid \texttt{and}(B\ ,\ B) \mid \texttt{or}(B\ ,\ B) \\
\\
C &::= \texttt{skip} \\
&\quad \mid x = E \\
&\quad \mid \texttt{if}(\ B\ ,\ C\ ,\ C\ ) \\
&\quad \mid \texttt{while}(\ B\ ,\ C\ ) \\
&\quad \mid \{\ C\ \} \mid C\ ;\ C \\
\\
P &::= \{\ C\ \},\ E
\end{aligned}
$$

Semantica programelor - idei generale

Ce înseamnă semantica formală?

Ce defineşte un limbaj de programare?

Ce înseamnă semantica formală?

Ce defineşte un limbaj de programare?

- Sintaxa – Simboluri de operaţie, cuvinte cheie, descriere (formală) a programelor/expresiilor bine formate

Ce înseamnă semantica formală?

Ce defineşte un limbaj de programare?

- Sintaxa – Simboluri de operaţie, cuvinte cheie, descriere (formală) a programelor/expresiilor bine formate
- Practic – Un limbaj e definit de modul cum poate fi folosit
  - Manual de utilizare şi exemple de bune practici
  - Implementare (compilator/interpretor)
  - Instrumente ajutătoare (analizor de sintaxă, depanator)

Ce înseamnă semantica formală?

Ce defineşte un limbaj de programare?

- Sintaxa – Simboluri de operaţie, cuvinte cheie, descriere (formală) a programelor/expresiilor bine formate
- Practic – Un limbaj e definit de modul cum poate fi folosit
  - Manual de utilizare şi exemple de bune practici
  - Implementare (compilator/interpretor)
  - Instrumente ajutătoare (analizor de sintaxă, depanator)
- Semantica – Ce înseamnă/care e comportamentul unei instrucţiuni?

La ce foloseşte semantica?

- Să înţelegem un limbaj în profunzime
  - Ca programator: pe ce mă pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanţii trebuie să ofer

La ce foloseşte semantica?

- Să înţelegem un limbaj în profunzime
  - Ca programator: pe ce mă pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanţii trebuie să ofer
- Ca instrument în proiectarea unui nou limbaj/a unei extensii
  - Înţelegerea componentelor şi a relaţiilor dintre ele
  - Exprimarea (şi motivarea) deciziilor de proiectare
  - Demonstrarea unor proprietăţi generice ale limbajului

La ce foloseşte semantica?

- Să înţelegem un limbaj în profunzime
  - Ca programator: pe ce mă pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanţii trebuie să ofer
- Ca instrument în proiectarea unui nou limbaj/a unei extensii
  - Înţelegerea componentelor şi a relaţiilor dintre ele
  - Exprimarea (şi motivarea) deciziilor de proiectare
  - Demonstrarea unor proprietăţi generice ale limbajului
- Ca bază pentru demonstrarea corectitudinii programelor

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Axiomatică – descrierea folosind logică a efectelor unei instrucţiuni
  - $\vdash \{\varphi\} cod \{\psi\}$
  - modelează un program prin formulele logice pe care le satisface
  - utilă pentru demonstrarea corectitunii

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Axiomatică – descrierea folosind logică a efectelor unei instrucţiuni
  - $\vdash \{\varphi\} cod \{\psi\}$
  - modelează un program prin formulele logice pe care le satisface
  - utilă pentru demonstrarea corectitunii
- Denotaţională – asocierea unui obiect matematic (denotaţie)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Axiomatică – descrierea folosind logică a efectelor unei instrucţiuni
  - $\vdash \{\varphi\} cod \{\psi\}$
  - modelează un program prin formulele logice pe care le satisface
  - utilă pentru demonstrarea corectitunii
- Denotaţională – asocierea unui obiect matematic (denotaţie)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice
- Operaţională – asocierea unei demonstraţii pentru execuţie
  - $\langle cod, \sigma \rangle \to \langle cod', \sigma' \rangle$
  - modelează un program prin execuţia pe o maşină abstractă
  - utilă pentru implementarea de compilatoare şi interpretoare

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Axiomatică – descrierea folosind logică a efectelor unei instrucţiuni
  - $\vdash \{\varphi\} cod \{\psi\}$
  - modelează un program prin formulele logice pe care le satisface
  - utilă pentru demonstrarea corectitunii
- Denotaţională – asocierea unui obiect matematic (denotaţie)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice
- Operaţională – asocierea unei demonstraţii pentru execuţie
  - $\langle cod, \sigma \rangle \to \langle cod', \sigma' \rangle$
  - modelează un program prin execuţia pe o maşină abstractă
  - utilă pentru implementarea de compilatoare şi interpretoare
- Statică – asocierea unui sistem de tipuri care exclude programe eronate

Semantica operaţională small-step

Imagine de ansamblu

- Semantica operaţională descrie cum se execută un program pe o masină abstractă (ideală).

Imagine de ansamblu

- Semantica operaţională descrie cum se execută un program pe o masină abstractă (ideală).
- Semantica operaţională **small-step**
  - semantica structurală, a pa silor mici
  - descrie cum o execuţie a programului avansează în functie de reduceri succesive.

$$\langle cod, \sigma \rangle \rightarrow \langle cod', \sigma' \rangle$$

Imagine de ansamblu

- Semantica operaţională descrie cum se execută un program pe o masină abstractă (ideală).
- Semantica operaţională **small-step**
  - semantica structurală, a pa silor mici
  - descrie cum o execuţie a programului avansează în functie de reduceri succesive.

$$\langle cod, \sigma \rangle \rightarrow \langle cod', \sigma' \rangle$$

- Semantica operaţională **big-step**
  - semantică naturală, într-un pas mare

Starea execuţiei

- Starea execuţiei unui program IMP la un moment dat este dată de valorile deţinute în acel moment de variabilele declarate în program.
- Formal, starea execuţiei unui program IMP la un moment dat este o funcţie parţială (cu domeniu finit):

$$\sigma : \mathit{Var} \rightharpoonup \mathit{Int}$$

Starea execuţiei

- Starea execuţiei unui program IMP la un moment dat este dată de valorile deţinute în acel moment de variabilele declarate în program.
- Formal, starea execuţiei unui program IMP la un moment dat este o funcţie parţială (cu domeniu finit):

$$\sigma : \mathit{Var} \rightharpoonup \mathit{Int}$$

- Notatii:
  - Descrierea funcţiei prin enumerare: $\sigma = n \mapsto 10, \mathit{sum} \mapsto 0$
  - Funcţia vidă $\perp$, nedefinită pentru nicio variabilă
  - Obţinerea valorii unei variabile: $\sigma(x)$
  - Suprascrierea valorii unei variabile:

$$\sigma_{x \leftarrow v}(y) = \begin{cases} \sigma(y), \text{ dacă } y \neq x \\ v, \text{ dacă } y = x \end{cases}$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma \rangle \rightarrow \langle cod'\ ,\ \sigma' \rangle$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma \rangle \rightarrow \langle cod'\ ,\ \sigma' \rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma \rangle \rightarrow \langle cod'\ ,\ \sigma' \rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

$\langle x = 0\ ;\ x = x + 1\ ;\ ,\ \perp \rangle$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma\rangle \rightarrow \langle cod'\ ,\ \sigma'\rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

$$\langle x = 0\ ;\ x = x + 1\ ;\ ,\ \perp\rangle \quad \rightarrow \quad \langle x = x + 1\ ;\ ,\ x \mapsto 0\rangle$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma\rangle \to \langle cod'\ ,\ \sigma'\rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

$$\begin{aligned}\langle x = 0\ ;\ x = x + 1\ ;\ ,\ \bot\rangle &\to \langle x = x + 1\ ;\ ,\ x \mapsto 0\rangle \\ &\to \langle x = 0 + 1\ ;\ ,\ x \mapsto 0\rangle\end{aligned}$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma\rangle \rightarrow \langle cod'\ ,\ \sigma'\rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

$$\begin{aligned}\langle x = 0\ ;\ x = x + 1\ ;\ ,\ \bot\rangle &\rightarrow \langle x = x + 1\ ;\ ,\ x \mapsto 0\rangle\\ &\rightarrow \langle x = 0 + 1\ ;\ ,\ x \mapsto 0\rangle\\ &\rightarrow \langle x = 1\ ;\ ,\ x \mapsto 0\rangle\end{aligned}$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma \rangle \rightarrow \langle cod'\ ,\ \sigma' \rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

$$\begin{aligned} \langle x = 0\ ;\ x = x + 1\ ;\ ,\ \bot \rangle &\rightarrow \langle x = x + 1\ ;\ ,\ x \mapsto 0 \rangle \\ &\rightarrow \langle x = 0 + 1\ ;\ ,\ x \mapsto 0 \rangle \\ &\rightarrow \langle x = 1\ ;\ ,\ x \mapsto 0 \rangle \\ &\rightarrow \langle \{\}\ ,\ x \mapsto 1 \rangle \end{aligned}$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma \rangle \rightarrow \langle cod'\ ,\ \sigma' \rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

$$\begin{aligned} \langle x = 0\ ;\ x = x + 1\ ;\ ,\ \bot \rangle &\rightarrow \langle x = x + 1\ ;\ ,\ x \mapsto 0 \rangle \\ &\rightarrow \langle x = 0 + 1\ ;\ ,\ x \mapsto 0 \rangle \\ &\rightarrow \langle x = 1\ ;\ ,\ x \mapsto 0 \rangle \\ &\rightarrow \langle \{\}\ ,\ x \mapsto 1 \rangle \end{aligned}$$

- Cum definim această relaţie?

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operaţională Structurală
  - semantică prin tranziţii
  - semantică prin reducere
- Defineşte cel mai mic pas de execuţie ca o relaţie „de tranziţie” între configuraţii:

$$\langle cod\ ,\ \sigma\rangle \to \langle cod'\ ,\ \sigma'\rangle$$

- Execuţia se obţine ca o succesiune de astfel de tranziţii:

$$\begin{aligned}\langle x = 0\ ;\ x = x + 1\ ;\ ,\ \bot\rangle &\to \langle x = x + 1\ ;\ ,\ x \mapsto 0\rangle\\ &\to \langle x = 0 + 1\ ;\ ,\ x \mapsto 0\rangle\\ &\to \langle x = 1\ ;\ ,\ x \mapsto 0\rangle\\ &\to \langle \{\}\ ,\ x \mapsto 1\rangle\end{aligned}$$

- Cum definim această relaţie? Prin inductie după elementele din sintaxă.

Redex. Reguli structurale. Axiome

- Expresie reductibilă (redex)
  - Fragmentul de sintaxă care va fi procesat la pasul următor

$$\texttt{if}\,(0 <= 5 + 7 * x\ ,\ r = 1\ ,\ r = 0)$$

Redex. Reguli structurale. Axiome

- Expresie reductibilă (redex)
  - Fragmentul de sintaxă care va fi procesat la pasul următor

$$\texttt{if}\,(0 \texttt{<=} 5 + 7 \texttt{ * } x \texttt{ , } r = 1 \texttt{ , } r = 0)$$

Redex. Reguli structurale. Axiome

- Expresie reductibilă (redex)
  - Fragmentul de sintaxă care va fi procesat la pasul următor

$$\texttt{if}\ (0 <= 5 + 7 * x\ ,\ r = 1\ ,\ r = 0)$$

- Reguli structurale
  - Folosesc la identificarea următorului redex
  - Definite recursiv pe structura termenilor

Redex. Reguli structurale. Axiome

- Expresie reductibilă (redex)
  - Fragmentul de sintaxă care va fi procesat la pasul următor

$$\texttt{if}\ (0 \texttt{ <= } 5 + 7 \texttt{ * } x\ \texttt{ , } r = 1\ \texttt{ , } r = 0)$$

- Reguli structurale
  - Folosesc la identificarea următorului redex
  - Definite recursiv pe structura termenilor

$$\frac{\langle b\ ,\ \sigma\rangle \to \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma\rangle \to \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

Redex. Reguli structurale. Axiome

- Expresie reductibilă (redex)
  - Fragmentul de sintaxă care va fi procesat la pasul următor

$$\texttt{if}\ (0 <= 5 + 7 * x\ ,\ r = 1\ ,\ r = 0)$$

- Reguli structurale
  - Folosesc la identificarea următorului redex
  - Definite recursiv pe structura termenilor

$$\frac{\langle b\ ,\ \sigma\rangle \rightarrow \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

- Axiome
  - Realizează pasul computaţional

Redex. Reguli structurale. Axiome

- Expresie reductibilă (redex)
  - Fragmentul de sintaxă care va fi procesat la pasul următor

$$\texttt{if}\ (0 <= 5 + 7 * x\ ,\ r = 1\ ,\ r = 0)$$

- Reguli structurale
  - Folosesc la identificarea următorului redex
  - Definite recursiv pe structura termenilor

$$\frac{\langle b\ ,\ \sigma\rangle \rightarrow \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

- Axiome
  - Realizează pasul computaţional

$$\langle \texttt{if}\ (\texttt{true}, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle bl_1\ ,\ \sigma\rangle$$

Semantica expresiilor aritmetice

- Semantica unui întreg este o valoare
  - nu poate fi redex, deci nu avem regulă

Semantica expresiilor aritmetice

- Semantica unui întreg este o valoare
  - nu poate fi redex, deci nu avem regulă
- Semantica unei variabile

(Id) $\langle x\ ,\ \sigma\rangle \to \langle i\ ,\ \sigma\rangle$ *dacă* $\sigma(x) = i$

Semantica expresiilor aritmetice

- Semantica unui întreg este o valoare
  - nu poate fi redex, deci nu avem regulă
- Semantica unei variabile

  (Id) $\langle x\ ,\ \sigma\rangle \to \langle i\ ,\ \sigma\rangle \quad \textit{dacă}\ \sigma(x) = i$
- Semantica adunării a două expresii aritmetice

  (Add) $\langle i_1 + i_2\ ,\ \sigma\rangle \to \langle i\ ,\ \sigma\rangle \quad \textit{dacă}\ i_1 + i_2 = i$

Semantica expresiilor aritmetice

- Semantica unui întreg este o valoare
  - nu poate fi redex, deci nu avem regulă
- Semantica unei variabile

(Id) $\langle x\ ,\ \sigma\rangle \to \langle i\ ,\ \sigma\rangle \quad \textit{dacă}\ \sigma(x) = i$

- Semantica adunării a două expresii aritmetice

(Add) $\langle i_1 + i_2\ ,\ \sigma\rangle \to \langle i\ ,\ \sigma\rangle \quad \textit{dacă}\ i_1 + i_2 = i$

$$\frac{\langle a_1\ ,\ \sigma\rangle \to \langle a_1'\ ,\ \sigma\rangle}{\langle a_1 + a_2\ ,\ \sigma\rangle \to \langle a_1' + a_2\ ,\ \sigma\rangle} \qquad \frac{\langle a_2\ ,\ \sigma\rangle \to \langle a_2'\ ,\ \sigma\rangle}{\langle a_1 + a_2\ ,\ \sigma\rangle \to \langle a_1 + a_2'\ ,\ \sigma\rangle}$$

Observatie: ordinea de evaluare a argumentelor este nespecificată.

Semantica expresiilor booleene

- Semantica operatorului de comparaţie

(LEQ-FALSE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \rightarrow \langle \texttt{false} \ , \ \sigma \rangle \quad$ *dacă* $i_1 > i_2$

(LEQ-TRUE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \rightarrow \langle \texttt{true} \ , \ \sigma \rangle \quad$ *dacă* $i_1 \leq i_2$

Semantica expresiilor booleene

Semantica operatorului de comparaţie

(LEQ-FALSE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \to \langle \texttt{false} \ , \ \sigma \rangle \quad$ *dacă* $i_1 > i_2$

(LEQ-TRUE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \to \langle \texttt{true} \ , \ \sigma \rangle \quad$ *dacă* $i_1 \leq i_2$

$$\frac{\langle a_1 \ , \ \sigma \rangle \to \langle a_1' \ , \ \sigma \rangle}{\langle a_1 =< a_2 \ , \ \sigma \rangle \to \langle a_1' =< a_2 \ , \ \sigma \rangle} \qquad \frac{\langle a_2 \ , \ \sigma \rangle \to \langle a_2' \ , \ \sigma \rangle}{\langle a_1 =< a_2 \ , \ \sigma \rangle \to \langle a_1 =< a_2' \ , \ \sigma \rangle}$$

Semantica expresiilor booleene

Semantica operatorului de comparaţie

(LEQ-FALSE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \rightarrow \langle \texttt{false} \ , \ \sigma \rangle$ *dacă* $i_1 > i_2$

(LEQ-TRUE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \rightarrow \langle \texttt{true} \ , \ \sigma \rangle$ *dacă* $i_1 \leq i_2$

$$\frac{\langle a_1 \ , \ \sigma \rangle \rightarrow \langle a_1' \ , \ \sigma \rangle}{\langle a_1 =< a_2 \ , \ \sigma \rangle \rightarrow \langle a_1' =< a_2 \ , \ \sigma \rangle} \qquad \frac{\langle a_2 \ , \ \sigma \rangle \rightarrow \langle a_2' \ , \ \sigma \rangle}{\langle a_1 =< a_2 \ , \ \sigma \rangle \rightarrow \langle a_1 =< a_2' \ , \ \sigma \rangle}$$

Semantica negaţiei

(!-FALSE) $\langle \texttt{not(true)} \ , \ \sigma \rangle \rightarrow \langle \texttt{false} \ , \ \sigma \rangle$

(!-TRUE) $\langle \texttt{not(false)} \ , \ \sigma \rangle \rightarrow \langle \texttt{true} \ , \ \sigma \rangle$

Semantica expresiilor booleene

Semantica operatorului de comparaţie

(LEQ-FALSE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \rightarrow \langle \texttt{false} \ , \ \sigma \rangle \quad$ *dacă* $i_1 > i_2$

(LEQ-TRUE) $\langle i_1 =< i_2 \ , \ \sigma \rangle \rightarrow \langle \texttt{true} \ , \ \sigma \rangle \quad$ *dacă* $i_1 \leq i_2$

$$\frac{\langle a_1 \ , \ \sigma \rangle \rightarrow \langle a_1' \ , \ \sigma \rangle}{\langle a_1 =< a_2 \ , \ \sigma \rangle \rightarrow \langle a_1' =< a_2 \ , \ \sigma \rangle} \qquad \frac{\langle a_2 \ , \ \sigma \rangle \rightarrow \langle a_2' \ , \ \sigma \rangle}{\langle a_1 =< a_2 \ , \ \sigma \rangle \rightarrow \langle a_1 =< a_2' \ , \ \sigma \rangle}$$

Semantica negaţiei

(!-FALSE) $\langle \texttt{not(true)} \ , \ \sigma \rangle \rightarrow \langle \texttt{false} \ , \ \sigma \rangle$

(!-TRUE) $\langle \texttt{not(false)} \ , \ \sigma \rangle \rightarrow \langle \texttt{true} \ , \ \sigma \rangle$

$$\frac{\langle a \ , \ \sigma \rangle \rightarrow \langle a' \ , \ \sigma \rangle}{\langle \texttt{not} \ (a) \ , \ \sigma \rangle \rightarrow \langle \texttt{not} \ (a') \ , \ \sigma \rangle}$$

Semantica expresiilor booleene

- Semantica şi-ului

(AND-FALSE) $\langle \texttt{and}\ (\texttt{false},\ b_2)\ ,\ \sigma\rangle \to \langle \texttt{false}\ ,\ \sigma\rangle$

(AND-TRUE) $\langle \texttt{and}\ (\texttt{true},\ b_2)\ ,\ \sigma\rangle \to \langle b_2\ ,\ \sigma\rangle$

$$\frac{\langle b_1\ ,\ \sigma\rangle \to \langle b_1'\ ,\ \sigma\rangle}{\langle \texttt{and}\ (b_1\ ,\ b_2)\ ,\ \sigma\rangle \to \langle \texttt{and}\ (b_1'\ ,\ b_2)\ ,\ \sigma\rangle}$$

Semantica compunerii şi a blocurilor

Semantica blocurilor

(BLOCK) $\langle \{\ s\ \}\ ,\ \sigma \rangle \rightarrow$

Semantica compunerii şi a blocurilor

Semantica blocurilor

(BLOCK) $\langle \{\ s\ \}\ ,\ \sigma \rangle \rightarrow \langle s\ ,\ \sigma \rangle$

Semantica compunerii şi a blocurilor

- Semantica blocurilor

  (BLOCK) $\langle \{\ s\ \}\ ,\ \sigma \rangle \rightarrow \langle s\ ,\ \sigma \rangle$

- Semantica compunerii secvenţiale

  (NEXT-STMT) $\langle \texttt{skip;}\ s_2\ ,\ \sigma \rangle \rightarrow \langle s_2\ ,\ \sigma \rangle$

$$\frac{\langle s_1\ ,\ \sigma \rangle \rightarrow \langle s_1'\ ,\ \sigma' \rangle}{\langle s_1\ ;\ s_2\ ,\ \sigma \rangle \rightarrow \langle s_1'\ ;\ s_2\ ,\ \sigma' \rangle}$$

Semantica compunerii şi a blocurilor

- Semantica blocurilor

(Block) $\langle \{\ s\ \}\ ,\ \sigma \rangle \to \langle s\ ,\ \sigma \rangle$

- Semantica compunerii secvenţiale

(Next-stmt) $\langle \texttt{skip;}\ s_2\ ,\ \sigma \rangle \to \langle s_2\ ,\ \sigma \rangle$

$$\frac{\langle s_1\ ,\ \sigma \rangle \to \langle s_1'\ ,\ \sigma' \rangle}{\langle s_1\ ;\ s_2\ ,\ \sigma \rangle \to \langle s_1'\ ;\ s_2\ ,\ \sigma' \rangle}$$

- Semantica atribuirii

(Asgn) $\langle x = i\ ,\ \sigma \rangle \to \langle \texttt{skip}\ ,\ \sigma' \rangle \quad \textit{dacă}\ \sigma' = \sigma_{x \leftarrow i}$

$$\frac{\langle a\ ,\ \sigma \rangle \to \langle a'\ ,\ \sigma \rangle}{\langle x = a\ ,\ \sigma \rangle \to \langle x = a'\ ,\ \sigma \rangle}$$

Semantica lui `if`

Semantica lui `if`

(IF-TRUE) $\langle \texttt{if}\ (\texttt{true}, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle bl_1\ ,\ \sigma\rangle$

(IF-FALSE) $\langle \texttt{if}\ (\texttt{false}, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle bl_2\ ,\ \sigma\rangle$

$$\frac{\langle b\ ,\ \sigma\rangle \rightarrow \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

Semantica lui `if`

- Semantica lui `if`

(IF-TRUE) $\langle \texttt{if}\ (\texttt{true}, bl_1, bl_2)\ ,\ \sigma\rangle \to \langle bl_1\ ,\ \sigma\rangle$

(IF-FALSE) $\langle \texttt{if}\ (\texttt{false}, bl_1, bl_2)\ ,\ \sigma\rangle \to \langle bl_2\ ,\ \sigma\rangle$

$$\frac{\langle b\ ,\ \sigma\rangle \to \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma\rangle \to \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

- Semantica lui `while`

(WHILE) $\langle \texttt{while}\ (b, bl)\ ,\ \sigma\rangle \to \langle \texttt{if}\ (b, bl\ \texttt{;while}\ (b, bl), \texttt{skip})\ ,\ \sigma\rangle$

Semantica lui `if`

- Semantica lui `if`

(IF-TRUE) $\langle \texttt{if}\ (\texttt{true}, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle bl_1\ ,\ \sigma\rangle$

(IF-FALSE) $\langle \texttt{if}\ (\texttt{false}, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle bl_2\ ,\ \sigma\rangle$

$$\frac{\langle b\ ,\ \sigma\rangle \rightarrow \langle b'\ ,\ \sigma\rangle}{\langle \texttt{if}\ (b, bl_1, bl_2)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b', bl_1, bl_2)\ ,\ \sigma\rangle}$$

- Semantica lui `while`

(WHILE) $\langle \texttt{while}\ (b, bl)\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b, bl\ \texttt{;while}\ (b, bl), \texttt{skip})\ ,\ \sigma\rangle$

- Semantica programelor

(PGM)
$$\frac{\langle a_1\ ,\ \sigma_1\rangle \rightarrow \langle a_2\ ,\ \sigma_2\rangle}{\langle (\texttt{skip}, a_1)\ ,\ \sigma_1\rangle \rightarrow \langle (\texttt{skip}, a_2)\ ,\ \sigma_2\rangle}$$

$$\frac{\langle s_1\ ,\ \sigma_1\rangle \rightarrow \langle s_2\ ,\ \sigma_2\rangle}{\langle (s_1, a)\ ,\ \sigma_1\rangle \rightarrow \langle (s_2, a)\ ,\ \sigma_2\rangle}$$

Semantica small-step a lui IMP

Execuţie pas cu pas

$$\langle i = 3\ ;\texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ \bot\rangle \xrightarrow{\text{PGM}}$$

Semantica small-step a lui IMP

Execuţie pas cu pas

$$\langle i = 3 \texttt{ ;while } (0 \texttt{ <= } i \texttt{ ,\{ } i = i + -4 \texttt{ \}}) \texttt{ , } \bot \rangle \xrightarrow{\textsc{Pgm}}$$

$$\langle \texttt{while } (0 \texttt{ <= } i \texttt{ ,\{ } i = i + -4 \texttt{ \}}) \texttt{ , } i \mapsto 3 \rangle \xrightarrow{\textsc{While}}$$

Semantica small-step a lui IMP

Execuţie pas cu pas

$$\langle i = 3\ ;\texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ \bot\rangle \xrightarrow{\text{PGM}}$$

$$\langle \texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{WHILE}}$$

$$\langle \texttt{if}\ (0 <= i\ ,\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{ID}}$$

Semantica small-step a lui IMP

Execuţie pas cu pas

$$\langle i = 3\ ;\texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ \bot\rangle \xrightarrow{\text{PGM}}$$

$$\langle \texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{WHILE}}$$

$$\langle \texttt{if}\ (0 <= i\ ,\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{ID}}$$

$$\langle \texttt{if}\ (0 <= 3\ ,\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{LEQ-TRUE}}$$

Semantica small-step a lui IMP

Execuţie pas cu pas

$$\langle i = 3\ ;\texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ \bot\rangle \xrightarrow{\text{PGM}}$$

$$\langle \texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{WHILE}}$$

$$\langle \texttt{if}\ (0 <= i\ ,\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{ID}}$$

$$\langle \texttt{if}\ (0 <= 3\ ,\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{LEQ-TRUE}}$$

$$\langle \texttt{if}\ (\texttt{true},\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{IF-TRUE}}$$

Semantica small-step a lui IMP

Execuţie pas cu pas

$$\langle i = 3\ ;\texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ \bot\rangle \xrightarrow{\text{PGM}}$$

$$\langle \texttt{while}\ (0 <= i\ ,\{\ i = i + -4\ \})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{WHILE}}$$

$$\langle \texttt{if}\ (0 <= i\ ,\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{ID}}$$

$$\langle \texttt{if}\ (0 <= 3\ ,\ i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{LEQ-TRUE}}$$

$$\langle \texttt{if}\ (\texttt{true}, i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\texttt{skip})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{IF-TRUE}}$$

$$\langle i = i + -4\ ;\texttt{while}\ (0 <= i, \{\ i = i + -4\ \})\ ,\ i \mapsto 3\rangle \xrightarrow{\text{ID}}$$

$\ldots$

Avantaje şi dezavantaje

Semantica operaţională

- \+ Defineşte precis noţiunea de pas computaţional
- \+ Semnalează erorile, oprind execuţia
- \+ Execuţia devine uşor de urmărit şi depanat
- – Regulile structurale sunt evidente şi deci plictisitor de scris
- – Nemodular: adăugarea unei trăsături noi poate solicita schimbarea întregii definiţii

Pe săptămâna viitoare!