Curs 12

2017-2018

Programare Logică

Cuprins

Semantica programelor - idei generale

Ce înseamnă semantica formală?

Ce definește un limbaj de programare?

Ce înseamnă semantica formală?

Ce definește un limbaj de programare?

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

Ce înseamnă semantica formală?

Ce definește un limbaj de programare?

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

Ce înseamnă semantica formală?

Ce definește un limbaj de programare?

- Sintaxa – Simboluri de operație, cuvinte cheie, descriere (formală) a programelor/expresiilor bine formate
- Practic – Un limbaj e definit de modul cum poate fi folosit
  - Manual de utilizare și exemple de bune practici
  - Implementare (compilator/interpretor)
  - Instrumente ajutătoare (analizor de sintaxă, depanator)
- Semantica – Ce înseamnă/care e comportamentul unei instrucțiuni?
  - De cele mai multe ori se dă din umeri și se spune că Practica e suficientă

Acest material urmează cursul introductiv:
T. Șerbănuță, Semantica Limbajelor de Programare, master, anul I.

La ce folosește semantica?

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

La ce folosește semantica?

- Să înțelegem un limbaj în profunzime
  - Ca programator: pe ce mă pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanții trebuie să ofer
- Ca instrument în proiectarea unui nou limbaj/a unei extensii
  - Înțelegerea componentelor și a relațiilor dintre ele
  - Exprimarea (și motivarea) deciziilor de proiectare
  - Demonstrarea unor proprietăți generice ale limbajului

La ce folosește semantica?

- Să înțelegem un limbaj în profunzime
  - Ca programator: pe ce mă pot baza când programez în limbajul dat
  - Ca implementator al limbajului: ce garanții trebuie să ofer
- Ca instrument în proiectarea unui nou limbaj/a unei extensii
  - Înțelegerea componentelor și a relațiilor dintre ele
  - Exprimarea (și motivarea) deciziilor de proiectare
  - Demonstrarea unor proprietăți 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
- Operațională – asocierea unei demonstrații pentru execuție
  - $\langle cod, \sigma \rangle \rightarrow \langle cod', \sigma' \rangle$
  - modelează un program prin execuția pe o mașină abstractă
  - utilă pentru implementarea de compilatoare și interpretoare

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Operațională – asocierea unei demonstrații pentru execuție
  - $\langle cod, \sigma \rangle \rightarrow \langle cod', \sigma' \rangle$
  - modelează un program prin execuția pe o mașină abstractă
  - utilă pentru implementarea de compilatoare și interpretoare
- Denotațională – asocierea unui obiect matematic (denotație)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Operațională – asocierea unei demonstrații pentru execuție
  - $\langle cod, \sigma \rangle \rightarrow \langle cod', \sigma' \rangle$
  - modelează un program prin execuția pe o mașină abstractă
  - utilă pentru implementarea de compilatoare și interpretoare
- Denotațională – asocierea unui obiect matematic (denotație)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice
- Axiomatică – descrierea folosind logică a efectelor unei instrucțiuni
  - $\vdash \{\varphi\} cod \{\psi\}$
  - modelează un program prin formulele logice pe care le satisface
  - utilă pentru demonstrarea corectitunii

Tipuri de semantică

- Limbaj natural – descriere textuală a efectelor
- Operațională – asocierea unei demonstrații pentru execuție
  - $\langle cod, \sigma \rangle \to \langle cod', \sigma' \rangle$
  - modelează un program prin execuția pe o mașină abstractă
  - utilă pentru implementarea de compilatoare și interpretoare
- Denotațională – asocierea unui obiect matematic (denotație)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice
- Axiomatică – descrierea folosind logică a efectelor unei instrucțiuni
  - $\vdash \{\varphi\} cod \{\psi\}$
  - modelează un program prin formulele logice pe care le satisface
  - utilă pentru demonstrarea corectitunii
- Statică – asocierea unui sistem de tipuri care exclude programe eronate

Limbajul IMP

IMP este un limbaj IMPerativ foarte simplu.

Ce conține:

- Expresii
  - Aritmetice `x + 3`
  - Booleene `(x > 7)`
- Blocuri de instrucțiuni
  - De atribuire `x = 5;`
  - Condiționale `if (x > 7) {x =5; } else {x = 0;}`
  - De ciclare `while (x > 7) {x = x - 1;}`

Ce nu conține:

- Expresii cu efecte laterale
- Proceduri și funcții
- Schimbări abrupte de control

Limbajul IMP

Exemplu

Un program în limbajul IMP

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

Sintaxa BNF a limbajului IMP

$$
\begin{aligned}
E &::= n \mid x \\
&\quad \mid E + E \mid E * E \\
\\
B &::= b \\
&\quad \mid E <= E \\
&\quad \mid ! B \mid B \,\&\&\, B \\
\\
C &::= \{\ C\ \} \mid \{\} \\
\\
C &::= C \mid C\ C \\
&\quad \mid x = E\ ; \\
&\quad \mid \texttt{if}\ (\ B\ )\ C\ \texttt{else}\ C \\
&\quad \mid \texttt{while}\ (\ B\ )\ C \\
\\
P &::= \texttt{int}\ x = n\ ;\ P \mid C
\end{aligned}
$$

Semantică în limbaj natural

Atribuirea: $x = \text{expr}$

- Expresia este evaluată în starea curentă a programului
- Variabilei i se atribuie valoarea calculată, înlocuind valoarea precedentă a acelei variabile.

Semantică în limbaj natural

Atribuirea: $x = \text{expr}$

- Expresia este evaluată în starea curentă a programului
- Variabilei i se atribuie valoarea calculată, înlocuind valoarea precedentă a acelei variabile.

Avantaje și dezavantaje

+ Ușor de prezentat
– Potențial ambiguă
– Imposibil de procesat automat

Semantica operațională

Imagine de ansamblu

- Semantica operațională descrie cum se execută un program pe o mașină abstractă (ideală).

Imagine de ansamblu

- Semantica operațională descrie cum se execută un program pe o mașină abstractă (ideală).
- Semantica operațională **small-step**
  - semantica structurală, a pașilor mici
  - descrie cum o execuție a programului avansează în funcție de reduceri succesive.

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

Imagine de ansamblu

- Semantica operațională descrie cum se execută un program pe o mașină abstractă (ideală).
- Semantica operațională **small-step**
  - semantica structurală, a pașilor mici
  - descrie cum o execuție a programului avansează în funcție de reduceri succesive.

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

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

Starea execuției

- Starea execuției unui program IMP la un moment dat este dată de valorile deținute în acel moment de variabilele declarate în program.
- Formal, starea executiei unui program IMP la un moment dat este o funcție parțială (cu domeniu finit):

$$\sigma : Var \rightharpoonup Int$$

Starea execuției

- Starea execuției unui program IMP la un moment dat este dată de valorile deținute în acel moment de variabilele declarate în program.
- Formal, starea executiei unui program IMP la un moment dat este o funcție parțială (cu domeniu finit):

$$\sigma : Var \rightharpoonup Int$$

- Notații:
  - Descrierea funcției prin enumerare: $\sigma = n \mapsto 10, sum \mapsto 0$
  - Funcția vidă $\perp$, nedefinită pentru nicio variabilă
  - Obținerea valorii unei variabile: $\sigma(x)$
  - Suprascrierea valorii unei variabile:

$$\sigma[v/x](y) = \begin{cases} \sigma(y), \text{ dacă } y \neq x \\ v, \text{ dacă } y = x \end{cases}$$

Semantica small-step

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește cel mai mic pas de execuție ca o relație „de tranziție” între configurații:

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

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește cel mai mic pas de execuție ca o relație „de tranziție" între configurații:

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

- Execuția se obține ca o succesiune de astfel de tranziții:

$$\langle \texttt{int}\ x = 0\ ;\ x = x + 1\ ;\ ,\ \bot\rangle \quad \to \quad \langle x = x + 1\ ;\ ,\ x \mapsto 0\rangle$$

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește cel mai mic pas de execuție ca o relație „de tranziție" între configurații:

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

- Execuția se obține ca o succesiune de astfel de tranziții:

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

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește cel mai mic pas de execuție ca o relație „de tranziție" între configurații:

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

- Execuția se obține ca o succesiune de astfel de tranziții:

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

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește cel mai mic pas de execuție ca o relație „de tranziție" între configurații:

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

- Execuția se obține ca o succesiune de astfel de tranziții:

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

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește cel mai mic pas de execuție ca o relație „de tranziție" între configurații:

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

- Execuția se obține ca o succesiune de astfel de tranziții:

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

- Cum definim această relație?

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică Operațională Structurală
  - semantică prin tranziții
  - semantică prin reducere
- Definește cel mai mic pas de execuție ca o relație „de tranziție" între configurații:

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

- Execuția se obține ca o succesiune de astfel de tranziții:

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

- Cum definim această relație? Prin inducție după elementele din sintaxă.

Redex. Reguli structurale. Axiome

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

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

Redex. Reguli structurale. Axiome

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

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

Redex. Reguli structurale. Axiome

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

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

- 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

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

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

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

Redex. Reguli structurale. Axiome

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

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

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

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

- Axiome
  - Realizează pasul computațional

Redex. Reguli structurale. Axiome

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

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

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

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

- Axiome
  - Realizează pasul computațional

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

Sintaxa BNF a limbajului IMP

$$
\begin{aligned}
E &::= n \mid x \\
&\quad \mid E + E \mid E * E \\
\\
B &::= b \\
&\quad \mid E <= E \\
&\quad \mid ! B \mid B \,\&\&\, B \\
\\
C &::= \{\ C\ \} \mid \{\} \\
\\
C &::= C \mid C\ C \\
&\quad \mid x = E\ ; \\
&\quad \mid \texttt{if}\ (\ B\ )\ C\ \texttt{else}\ C \\
&\quad \mid \texttt{while}\ (\ B\ )\ C \\
\\
P &::= \texttt{int}\ x = n\ ;\ P \mid C
\end{aligned}
$$

Semantica small-step a lui IMP

Semantica expresiilor aritmetice

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

Semantica small-step a lui IMP

Semantica expresiilor aritmetice

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

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

Semantica small-step a lui IMP

Semantica expresiilor aritmetice

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

- Semantica unei variabile

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

- Semantica adunării a două expresii aritmetice

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

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

Semantica small-step a lui IMP

Semantica expresiilor aritmetice

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

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

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

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

- Semantica înmulțirii a două expresii aritmetice – similar

Semantica small-step a lui IMP

Semantica expresiilor Booleene

- Semantica constantelor Booleene sunt valori
  - nu pot fi redex, deci nu avem reguli

Semantica small-step a lui IMP

Semantica expresiilor Booleene

- Semantica constantelor Booleene sunt valori
  - nu pot fi redex, deci nu avem reguli

- Semantica operatorului de comparație

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

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

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

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

Semantica small-step a lui IMP

- Semantica negației

Semantica small-step a lui IMP

Semantica negației

(!-TRUE) $\langle \texttt{!true} \ , \ \sigma \rangle \to \langle \texttt{false} \ , \ \sigma \rangle$

(!-FALSE) $\langle \texttt{!false} \ , \ \sigma \rangle \to \langle \texttt{true} \ , \ \sigma \rangle$

$$\frac{\langle a \ , \ \sigma \rangle \to \langle a' \ , \ \sigma \rangle}{\langle \texttt{!} \ a \ , \ \sigma \rangle \to \langle \texttt{!} \ a' \ , \ \sigma \rangle}$$

Semantica small-step a lui IMP

Semantica negației

(!-TRUE) $\langle \texttt{!true} \ , \ \sigma \rangle \to \langle \texttt{false} \ , \ \sigma \rangle$

(!-FALSE) $\langle \texttt{!false} \ , \ \sigma \rangle \to \langle \texttt{true} \ , \ \sigma \rangle$

$$\frac{\langle a \ , \ \sigma \rangle \to \langle a' \ , \ \sigma \rangle}{\langle \texttt{!} \ a \ , \ \sigma \rangle \to \langle \texttt{!} \ a' \ , \ \sigma \rangle}$$

Semantica și-ului

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

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

$$\frac{\langle b_1 \ , \ \sigma \rangle \to \langle b_1' \ , \ \sigma \rangle}{\langle b_1 \ \texttt{\&\&} \ b_2 \ , \ \sigma \rangle \to \langle b_1' \ \texttt{\&\&} \ b_2 \ , \ \sigma \rangle}$$

Semantica small-step a lui IMP

Semantica comenzilor

Semantica blocurilor

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

$$\frac{\langle s \ , \ \sigma \rangle \to \langle s' \ , \ \sigma' \rangle}{\langle \{ \ s \ \} \ , \ \sigma \rangle \to \langle \{ \ s' \ \} \ , \ \sigma' \rangle}$$

Atenție! O instrucțiune poate modifica starea curentă!

Semantica small-step a lui IMP

Semantica comenzilor

- Semantica blocurilor

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

$$\frac{\langle s \ , \ \sigma \rangle \to \langle s' \ , \ \sigma' \rangle}{\langle \{ \ s \ \} \ , \ \sigma \rangle \to \langle \{ \ s' \ \} \ , \ \sigma' \rangle}$$

Atenție! O instrucțiune poate modifica starea curentă!

- Semantica compunerii secvențiale

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

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

Semantica small-step a lui IMP

Semantica atribuirii

(ASGN) $\langle x = i\ ;\ ,\ \sigma\rangle \to \langle \{\}\ ,\ \sigma'\rangle \quad \text{dacă } \sigma' = \sigma[i/x]$

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

Semantica small-step a lui IMP

Semantica atribuirii

(Asgn) $\langle x = i\ ;\ ,\ \sigma\rangle \to \langle \{\}\ ,\ \sigma'\rangle \quad \text{dacă } \sigma' = \sigma[i/x]$

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

Semantica lui if

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

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

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

Semantica small-step a lui IMP

- Semantica lui while

(While) $\langle \texttt{while}\ (b)\ bl\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b)\ \ \texttt{\{}\ bl\ \texttt{while}\ (b)\ bl\ \texttt{\}else\{\}}\ ,\ \sigma\rangle$

Semantica small-step a lui IMP

- Semantica lui while

(While) $\langle \texttt{while}\ (b)\ bl\ ,\ \sigma\rangle \rightarrow \langle \texttt{if}\ (b)\ \texttt{\{}\ bl\ \texttt{while}\ (b)\ bl\ \texttt{\}else\{\}}\ ,\ \sigma\rangle$

- Semantica inițializărilor

(Init) $\langle \texttt{int}\ x = i\ \texttt{;}\ p\ ,\ \sigma\rangle \rightarrow \langle p\ ,\ \sigma'\rangle \quad \textit{dacă}\ \sigma' = \sigma[i/x]$

Semantica small-step a lui IMP

Execuție pas cu pas

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

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

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

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

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

$$\langle \texttt{if } (0 \texttt{ <= } 3) \texttt{ \{\{ } i = i + -4 \texttt{ ;\} while } (0 \texttt{ <= } i) \texttt{ \{ } i = i + -4 \texttt{ ;\} \} else \{\}} , i \mapsto 3 \rangle \xrightarrow{\textsc{Leq-true}}$$

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

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

$$\langle \texttt{if } (0 \texttt{ <= } 3) \texttt{ \{\{ } i = i + -4 \texttt{ ;\} } \texttt{ while } (0 \texttt{ <= } i) \texttt{ \{ } i = i + -4 \texttt{ ;\} } \texttt{ \} else \{\} } , i \mapsto 3 \rangle \xrightarrow{\textsc{Leq-true}}$$

$$\langle \texttt{if } (\mathit{true}) \texttt{ \{\{ } i = i + -4 \texttt{ ;\} } \texttt{ while } (0 \texttt{ <= } i) \texttt{ \{ } i = i + -4 \texttt{ ;\} } \texttt{ \} else \{\} } , i \mapsto 3 \rangle \xrightarrow{\textsc{If-true}}$$

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

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

$$\langle \texttt{if } (0 \texttt{ <= } 3) \texttt{ \{\{ } i = i + -4 \texttt{ ;\} while } (0 \texttt{ <= } i) \texttt{ \{ } i = i + -4 \texttt{ ;\} \} else \{\}} , i \mapsto 3 \rangle \xrightarrow{\textsc{Leq-true}}$$

$$\langle \texttt{if } (\mathit{true}) \texttt{ \{\{ } i = i + -4 \texttt{ ;\} while } (0 \texttt{ <= } i) \texttt{ \{ } i = i + -4 \texttt{ ;\} \} else \{\}} , i \mapsto 3 \rangle \xrightarrow{\textsc{If-true}}$$

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

Semantica small-step a lui IMP

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

Semantica small-step a lui IMP

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

$$\langle \{\{\ i = -1\ ;\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto 3 \rangle \xrightarrow{\textsc{Asgn}}$$

Semantica small-step a lui IMP

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

$$\langle \{\{\ i = -1\ ;\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto 3 \rangle \xrightarrow{\textsc{Asgn}}$$

$$\langle \{\{\{\}\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1 \rangle \xrightarrow{\textsc{Block-end}}$$

Semantica small-step a lui IMP

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

$$\langle \{\{\ i = -1\ ;\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto 3 \rangle \xrightarrow{\textsc{Asgn}}$$

$$\langle \{\{\}\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1 \rangle \xrightarrow{\textsc{Block-end}}$$

$$\langle \{\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1 \rangle \xrightarrow{\textsc{Next-stmt}}$$

Semantica small-step a lui IMP

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

$$\langle \{\{\ i = -1\ ;\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto 3 \rangle \xrightarrow{\textsc{Asgn}}$$

$$\langle \{\{\}\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1 \rangle \xrightarrow{\textsc{Block-end}}$$

$$\langle \{\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1 \rangle \xrightarrow{\textsc{Next-stmt}}$$

$$\langle \{\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1 \rangle \xrightarrow{\textsc{While}}$$

Semantica small-step a lui IMP

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

$$\langle \{\{\ i = -1\ ;\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto 3\rangle \xrightarrow{\textsc{Asgn}}$$

$$\langle \{\{\}\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{Block-end}}$$

$$\langle \{\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{Next-stmt}}$$

$$\langle \{\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{While}}$$

$$\langle \{\ \texttt{if}\ (0 <= i)\ \{\{\ i = i + -4\ ;\}\ \texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\ \}\ \texttt{else}\ \{\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{Id}}$$

Semantica small-step a lui IMP

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

$$\langle \{\{\ i = -1\ ;\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto 3\rangle \xrightarrow{\textsc{Asgn}}$$

$$\langle \{\{\}\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{Block-end}}$$

$$\langle \{\}\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{Next-stmt}}$$

$$\langle \{\texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{While}}$$

$$\langle \{\ \texttt{if}\ (0 <= i)\ \{\{\ i = i + -4\ ;\}\ \texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\ \}\ \texttt{else}\ \{\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{Id}}$$

$$\langle \{\ \texttt{if}\ (0 <= -1)\ \{\{\ i = i + -4\ ;\}\ \texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\ \}\ \texttt{else}\ \{\}\}\ ,\ i \mapsto -1\rangle \xrightarrow{\textsc{Leq-false}}$$

Semantica small-step a lui IMP

$$\langle \texttt{\{ if (false) \{\{ } i \texttt{ = } i \texttt{ + } -4 \texttt{ ;\} while (0 <= } i \texttt{) \{ } i \texttt{ = } i \texttt{ + } -4 \texttt{ ;\} \} else \{\} \}} \;,\; i \mapsto -1 \rangle \xrightarrow{\textsc{If-false}}$$

Semantica small-step a lui IMP

$$\langle \{\ \texttt{if}\ (\texttt{false})\ \{\{\ i = i + -4\ ;\}\ \texttt{while}\ (0 <= i)\ \{\ i = i + -4\ ;\}\ \}\ \texttt{else}\ \{\}\ \},\ i \mapsto -1 \rangle \xrightarrow{\textsc{If-false}} \langle \{\{\}\},\ i \mapsto -1 \rangle \xrightarrow{\textsc{Block-end}}$$

Semantica small-step a lui IMP

$$\langle \texttt{\{ if (false) \{\{ } i = i + -4 \texttt{ ;\}} \quad \texttt{while (0 <= } i\texttt{) \{ } i = i + -4 \texttt{ ;\}} \quad \texttt{\} else \{\} \}} \;,\; i \mapsto -1 \rangle \xrightarrow{\textsc{If-false}}$$

$$\langle \texttt{\{\{\}\}} \;,\; i \mapsto -1 \rangle \xrightarrow{\textsc{Block-end}}$$

$$\langle \texttt{\{\}} \;,\; i \mapsto -1 \rangle$$

Semantica small-step

Avantaje

- Definește precis noțiunea de pas computațional
- Semnalează erorile, oprind execuția
- Execuția devine ușor de urmărit și depanat
- Nedeterminismul și concurența pot fi definite și analizate

Semantica small-step

Avantaje

- Definește precis noțiunea de pas computațional
- Semnalează erorile, oprind execuția
- Execuția devine ușor de urmărit și depanat
- Nedeterminismul și concurența pot fi definite și analizate

Dezavantaje

- Regulile structurale sunt evidente și deci plictisitor de scris
- Schimbarea abruptă a controlului rămâne o o sarcină dificilă
- Nemodular: adăugarea unei trăsături noi poate solicita schimbarea întregii definiții

Semantica big-step

Semantica big-step

- Introdusă de Gilles Kahn (1987)
- Denumiri alternative:
  - semantică relațională
  - semantică naturală
- Relaționează fragmente de program într-o stare cu valoarea corespunzătoare evaluării lor în acea stare
  - Expresiile aritmetice se evaluează la întregi: $\langle a\ ,\ \sigma\rangle \Downarrow \langle i\rangle$
  - Expresiile Booleene se evaluează la $true/false$: $\langle b\ ,\ \sigma\rangle \Downarrow \langle t\rangle$
  - Instrucțiunile se evaluează la stări: $\langle ins\ ,\ \sigma\rangle \Downarrow \langle \sigma'\rangle$
  - Blocurile se evaluează la stări: $\langle bl\ ,\ \sigma\rangle \Downarrow \langle \sigma'\rangle$
  - Programul se evaluează la o stare: $\langle p\rangle \Downarrow \langle \sigma\rangle$ sau $\langle p\ ,\ \sigma\rangle \Downarrow \langle \sigma'\rangle$
- Valoarea este obținută într-un singur pas (mare)
- Reguli structurale, având ca premize secvenți corespunzători subtermenilor.

Semantica big-step

Exemplu

- $\langle 3 + x\ ,\ (x \mapsto 5, y \mapsto 7)\rangle \Downarrow \langle 8\rangle$
- $\langle x = 3 + y\ ,\ (x \mapsto 5, y \mapsto 7)\rangle \Downarrow \langle x \mapsto 10, y \mapsto 7\rangle$

Semantica big-step a lui IMP

Expresii aritmetice

(Int) $\langle i\ ,\ \sigma\rangle \Downarrow \langle i\rangle$

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

(Add) $$\frac{\langle a_1\ ,\ \sigma\rangle \Downarrow \langle i_1\rangle \quad \langle a_2\ ,\ \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 + a_2\ ,\ \sigma\rangle \Downarrow \langle i\rangle} \quad \text{dacă } i = i_1 + i_2$$

(Mul) $$\frac{\langle a_1\ ,\ \sigma\rangle \Downarrow \langle i_1\rangle \quad \langle a_2\ ,\ \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 * a_2\ ,\ \sigma\rangle \Downarrow \langle i\rangle} \quad \text{dacă } i = i_1 * i_2$$

Semantica big-step a lui IMP

Expresii booleene

(Bool) $\langle t\ ,\ \sigma\rangle \Downarrow \langle t\rangle$

(Cmp) $$\frac{\langle a_1\ ,\ \sigma\rangle \Downarrow \langle i_1\rangle \quad \langle a_2\ ,\ \sigma\rangle \Downarrow \langle i_2\rangle}{\langle a_1 \texttt{ <= } a_2\ ,\ \sigma\rangle \Downarrow \langle t\rangle} \quad \textit{dacă } t = i_1 \leq i_2$$

(Not-True) $$\frac{\langle b\ ,\ \sigma\rangle \Downarrow \langle \mathit{false}\rangle}{\langle \texttt{!}\ b\ ,\ \sigma\rangle \Downarrow \langle \mathit{true}\rangle}$$

(Not-False) $$\frac{\langle b\ ,\ \sigma\rangle \Downarrow \langle \mathit{true}\rangle}{\langle \texttt{!}\ b\ ,\ \sigma\rangle \Downarrow \langle \mathit{false}\rangle}$$

(And-True) $$\frac{\langle b_1\ ,\ \sigma\rangle \Downarrow \langle \mathit{true}\rangle \quad \langle b_2\ ,\ \sigma\rangle \Downarrow \langle t\rangle}{\langle b_1 \texttt{ \&\& } b_2\ ,\ \sigma\rangle \Downarrow \langle t\rangle}$$

(And-False) $$\frac{\langle b_1\ ,\ \sigma\rangle \Downarrow \langle \mathit{false}\rangle}{\langle b_1 \texttt{ \&\& } b_2\ ,\ \sigma\rangle \Downarrow \langle \mathit{false}\rangle}$$

Semantica big-step a lui IMP

Instrucțiuni simple

(SECV) $$\frac{\langle s_1\ ,\ \sigma\rangle \Downarrow \langle \sigma'\rangle \quad \langle s_2\ ,\ \sigma'\rangle \Downarrow \langle \sigma''\rangle}{\langle s_1\ \ s_2\ ,\ \sigma\rangle \Downarrow \langle \sigma''\rangle}$$

(ASGN) $$\frac{\langle a\ ,\ \sigma\rangle \Downarrow \langle i\rangle}{\langle x = a\ ;\ ,\ \sigma\rangle \Downarrow \langle \sigma'\rangle} \quad \text{dacă } \sigma' = \sigma[i/x]$$

(IF-TRUE) $$\frac{\langle b\ ,\ \sigma\rangle \Downarrow \langle true\rangle \quad \langle bl_1\ ,\ \sigma\rangle \Downarrow \langle \sigma_1\rangle}{\langle \texttt{if}\ \ (b)\ bl_1\ \texttt{else}\ bl_2\ ,\ \sigma\rangle \Downarrow \langle \sigma_1\rangle}$$

(IF-FALSE) $$\frac{\langle b\ ,\ \sigma\rangle \Downarrow \langle false\rangle \quad \langle bl_2\ ,\ \sigma\rangle \Downarrow \langle \sigma_2\rangle}{\langle \texttt{if}\ \ (b)\ bl_1\ \texttt{else}\ bl_2\ ,\ \sigma\rangle \Downarrow \langle \sigma_2\rangle}$$

Semantica big-step a lui IMP

Blocuri și instrucțiuni de ciclare

(Skip) $\langle \{\} \ , \ \sigma \rangle \Downarrow \langle \sigma \rangle$

(Block) $$\frac{\langle s \ , \ \sigma \rangle \Downarrow \langle \sigma' \rangle}{\langle \{ \ s \ \} \ , \ \sigma \rangle \Downarrow \langle \sigma' \rangle}$$

(While-True) $$\frac{\langle b \ , \ \sigma \rangle \Downarrow \langle true \rangle \quad \langle bl \ , \ \sigma \rangle \Downarrow \langle \sigma' \rangle \quad \langle \texttt{while} \ \ (b) \ \ bl \ , \ \sigma' \rangle \Downarrow \langle \sigma'' \rangle}{\langle \texttt{while} \ \ (b) \ \ bl \ , \ \sigma \rangle \Downarrow \langle \sigma'' \rangle}$$

(While-False) $$\frac{\langle b \ , \ \sigma \rangle \Downarrow \langle false \rangle}{\langle \texttt{while} \ \ (b) \ \ bl \ , \ \sigma \rangle \Downarrow \langle \sigma \rangle}$$

Semantica big-step a lui IMP

Inițializări. Semantica programului

$$\text{(Init)}\quad \frac{\langle p\ ,\ \sigma'\rangle \Downarrow \langle \sigma''\rangle}{\langle \texttt{int}\ x = i\ ;\ p\ ,\ \sigma\rangle \Downarrow \langle \sigma''\rangle} \quad \textit{dacă}\ \sigma' = \sigma[i/x]$$

$$\text{(Pgm)}\quad \frac{\langle p\ ,\ \bot\rangle \Downarrow \langle \sigma\rangle}{\langle p\rangle \Downarrow \langle \sigma\rangle}$$

Arbori de derivare

$$\textsc{(Pgm)}\ \frac{\textsc{(Init)}\ \dfrac{\textsc{(Asgn)}\ \dfrac{\textsc{(Add)}\ \dfrac{}{\langle a + 4\ ,\ a \mapsto 3\rangle \Downarrow \langle ?? \rangle}}{\langle a = a + 4\ ;\ ,\ a \mapsto 3\rangle \Downarrow \langle a \mapsto ?? \rangle}}{\langle \texttt{int } a = 3;\ a = a + 4\ ;\ ,\ \bot\rangle \Downarrow \langle a \mapsto ?? \rangle}}{\langle \texttt{int } a = 3;\ a = a + 4\ ;\rangle \Downarrow \langle a \mapsto ?? \rangle}$$

Arbori de derivare

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\langle a\ ,\ a \mapsto 3\rangle \Downarrow \langle 3\rangle}(\textsc{Id}) \qquad \dfrac{}{\langle 4\ ,\ a \mapsto 3\rangle \Downarrow \langle 4\rangle}(\textsc{Int})}{\langle a + 4\ ,\ a \mapsto 3\rangle \Downarrow \langle 7\rangle}(\textsc{Add})}{\langle a = a + 4\ ;\ ,\ a \mapsto 3\rangle \Downarrow \langle a \mapsto 7\rangle}(\textsc{Asgn})}{\langle \texttt{int } a \texttt{ =3; } a = a + 4\ ;\ ,\ \bot\rangle \Downarrow \langle a \mapsto 7\rangle}(\textsc{Init})}{\langle \texttt{int } a \texttt{ =3; } a = a + 4\ ;\rangle \Downarrow \langle a \mapsto 7\rangle}(\textsc{Pgm})$$

Proprietăți ale limbajului IMP

- Intuitiv, două instrucțiuni sunt echivalente dacă produc același rezultat în orice stare de execuție a programului.
- Formal, două instrucțiuni c_1 și c_2 sunt echivalente (notat $c_1 \sim c_2$) dacă, pentru orice stări σ și σ'

$$\langle c_1 \ , \ \sigma \rangle \Downarrow \langle \sigma' \rangle \quad \Leftrightarrow \quad \langle c_2 \ , \ \sigma \rangle \Downarrow \langle \sigma' \rangle$$

Proprietăți ale limbajului IMP

- Intuitiv, două instrucțiuni sunt echivalente dacă produc același rezultat în orice stare de execuție a programului.
- Formal, două instrucțiuni c_1 și c_2 sunt echivalente (notat $c_1 \sim c_2$) dacă, pentru orice stări σ și σ'

$$\langle c_1 \ , \ \sigma \rangle \Downarrow \langle \sigma' \rangle \quad \Leftrightarrow \quad \langle c_2 \ , \ \sigma \rangle \Downarrow \langle \sigma' \rangle$$

De exemplu, putem arăta următorul rezultat:

Teoremă

$$\text{while (b) c} \quad \sim \quad \text{if (b) c ; while (b) c else \{\}}$$

Proprietăți ale limbajului IMP

Teoremă (Determinism)

Dacă $\langle c\ ,\ \sigma\rangle \Downarrow \langle\sigma'\rangle$ *și* $\langle c\ ,\ \sigma\rangle \Downarrow \langle\sigma''\rangle$ *atunci* $\sigma' = \sigma''$.

Semantica big-step

Avantaje

- Compozițională: arborii de demonstrație sunt compoziționali
- Ușor și relativ eficient de implementat și executat
- Foarte folosită pentru definirea sistemelor de tipuri

Semantica big-step

Avantaje

- Compozițională: arborii de demonstrație sunt compoziționali
- Ușor și relativ eficient de implementat și executat
- Foarte folosită pentru definirea sistemelor de tipuri

Dezavantaje

- Lipsa granularității - computația e un monolit
- Greu de capturat nedeterminismul/concurența
- Greu de capturat schimbările de control
- Nemodulară - extensiile solicită modificarea regulilor existente.

Semantica denotațională

Semantica denotațională

- Semantica operațională, ca un interpretor, descrie cum să evaluăm un program.
- Semantica denotațională, ca un compilator, descrie o traducere a limbajului într-un limbaj diferit cu semantică cunoscută, anume matematica.
- Semantica denotațională definește ce înseamnă un program ca o funcție matematică.

Semantica denotațională

Atribuirea: $x = \text{expr}$

- Asociem expresiilor aritmetice funcții de la starea memoriei la valori:

- Asociem instrucțiunilor funcții de la starea memoriei la starea (următoare) a memoriei.

Semantica denotațională

Atribuirea: $x = \text{expr}$

- Asociem expresiilor aritmetice funcții de la starea memoriei la valori:
  - Funcția constantă $[[1]](s) = 1$
  - Funcția care selectează valoarea unui identificator $[[x]](s) = s(x)$
  - „Morfismul de adunare” $[[e1 + e2]](s) = [[e1]](s) + [[e2]](s)$.
- Asociem instrucțiunilor funcții de la starea memoriei la starea (următoare) a memoriei.
  - $[[x = e]](s)(y) = \begin{cases} s(y), \text{ dacă } y \neq x \\ [[e]](s), \text{ dacă } y = x \end{cases}$

Semantica denotațională

Avantaje și dezavantaje

- \+ Formală, matematică, foarte precisă
- \+ Compozițională (morfisme și compuneri de funcții)
- – Greu de stăpânit (domeniile devin din ce în ce mai complexe)

Semantica denotațională a lui IMP

```
type Id = String
type State = Id -> Int
type DAExp = State -> Int
type DBExp = State -> Bool
type DStmt = State ->
    State

denotAExp :: AExp -> DAExp
denotAExp (Int n) s = n
denotAExp (e1 :+: e2) s
  = denotAExp e1 s
  + denotAExp e2 s

denotBExp :: BExp -> DBExp
denotBExp (Bool b) s = b
denotBExp (a1 :<=: a2) s
  = denotAExp a1 s
  <= denotAExp a2 s
```

```
denotStmt :: Stmt -> DStmt
denotStmt Skip = id
denotStmt (s1 ::: s2) =
  denotStmt s2 . denotStmt s1
denotStmt (If c t e) s
 |denotBExp c s = denotStmt t
      s
 |otherwise     = denotStmt e
      s
denotStmt (While c b) =
  fix (\w s ->
    if denotBExp b s
      then w (denotStmt c s)
      else s)
```

Semantica axiomatică

Semantica Axiomatică

Definește triplete (triplete Hoare) de forma

$$\{Pre\}\ S\ \{Post\}$$

unde:

- S este o instrucțiune (Stmt)
- *Pre* (precondiție), respectiv *Post* (postcondiție) sunt asertiuni logice asupra stării sistemului înaintea, respectiv după execuția lui S

Se asociază fiecărei construcții sintactice Stmt o regulă de deducție care definește recursiv tripletele Hoare descrise mai sus.

Logica Floyd-Hoare pentru IMP

$$\text{(Skip)} \quad \frac{\cdot}{\{P\}\ \{\}\ \{P\}}$$

$$\text{(Seq)} \quad \frac{\{P\}\ c1\ \{Q\} \quad \{Q\}\ c2\ \{R\}}{\{P\}\ c1; c2\ \{R\}}$$

$$\text{(Asign)} \quad \{P[e/x]\}\ x = e\ \{P\}$$

$$\text{(If)} \quad \frac{\{c \wedge P\}\ t\ \{Q\} \quad \{\neg c \wedge P\}\ e\ \{Q\}}{\{P\}\ \texttt{if}\ c\ \texttt{then}\ t\ \texttt{else}\ e\ \{Q\}}$$

$$\text{(While)} \quad \frac{\{c \wedge P\}\ b\ \{P\}}{\{P\}\ \texttt{while}\ (c)\ b\ \{\neg c \wedge P\}}$$

Semantica statică

Semantică Statică - Motivație

- Este sintaxa unui limbaj de programare prea expresivă?
- Sunt programe pe care nu aș vrea să le pot scrie, dar le pot scrie?
- Putem detecta programe greșite înainte de rulare?

Semantică Statică - Motivație

- Este sintaxa unui limbaj de programare prea expresivă?
- Sunt programe pe care nu aș vrea să le pot scrie, dar le pot scrie?
- Putem detecta programe greșite înainte de rulare?
  De exemplu, în IMP, folosirea variabilelor fără a le declara
- Soluție: Sistemele de tipuri

Sisteme de tipuri

La ce folosesc?

- Descriu programele „bine formate“
- Pot preveni anumite erori
  - folosirea variabilelor nedeclarate/neințializate
  - detectarea unor bucați de cod inaccesibile
  - erori de securitate
- Ajută compilatorul
- Pot influența proiectarea limbajului

Sisteme de tipuri

- Vom defini o relație $\Gamma \vdash e : T$
- Citim e are tipul T dacă Γ, unde
- Γ — tipuri asociate locațiilor din e

Exemplu

$$\vdash \quad \texttt{if } true \texttt{ then } \{\} \texttt{ else } \{\} : \texttt{stmt}$$

$$x : \texttt{int} \quad \vdash \quad x + 13 \qquad : \texttt{int}$$

$$x : \texttt{int} \quad \nvdash \quad x = y + 1 \qquad : T \qquad \text{pentru orice } T$$

IMP: Reguli pentru tipuri

(LOC) $\Gamma \vdash x : \mathit{int} \quad \mathit{dacă}\ \Gamma(x) = \texttt{int}$

(INT) $\Gamma \vdash n : \texttt{int} \quad \mathit{dacă}\ n \in \mathbb{Z}$

(OP+) $$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 + e_2 : \texttt{int}}$$

(BOOL) $\Gamma \vdash b : \texttt{bool} \quad \mathit{dacă}\ b \in \{true, false\}$

(OP≤) $$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \texttt{ <= } e_2 : \texttt{bool}}$$

(ATRIB) $$\frac{\Gamma \vdash e : \texttt{int}}{\Gamma \vdash x \texttt{ = } e \texttt{ ; } : \texttt{stmt}} \quad \mathit{dacă}\ \Gamma(x) = \texttt{int}$$

(SECV) $$\frac{\Gamma \vdash c_1 : \texttt{stmt} \quad \Gamma \vdash c_2 : \texttt{stmt}}{\Gamma \vdash c_1 \quad c_2 : \texttt{stmt}}$$

(IF) $$\frac{\Gamma \vdash c : \texttt{bool} \quad \Gamma \vdash t : \texttt{stmt} \quad \Gamma \vdash e : \texttt{stmt}}{\Gamma \vdash \texttt{if } c \texttt{ then } t \texttt{ else } e : \texttt{stmt}}$$

(WHILE) $$\frac{\Gamma \vdash c : \texttt{bool} \quad \Gamma \vdash b : \texttt{stmt}}{\Gamma \vdash \texttt{while } (c)\ b : \texttt{stmt}}$$

Pe săptămâna viitoare!