Curs 10

2018-2019

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 are la bază 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
- 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
- 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
- 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
- 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
- Denotațională – asocierea unui obiect matematic (denotație)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice
- 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
- 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
- Denotațională – asocierea unui obiect matematic (denotație)
  - $[\![cod]\!]$
  - modelează un program ca obiecte matematice
  - utilă pentru fundamente matematice
- 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
- 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 y = 1;
while (0 < x) {
  y = y * x;
  x = x + -1;
}
```

Sintaxa BNF a limbajului IMP

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

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 axiomatică

Semantica Axiomatică

- Inventată de 1969 Tony Hoare în 1969 (insiprată de rezultatele lui Robert Floyd).
- Definește triplete (triplete Hoare) de forma

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

unde:

  - S este o instrucțiune (Stmt)
  - *Pre* (precondiție), respectiv *Post* (postcondiție) sunt aserțiuni logice asupra stării sistemului înaintea, respectiv după execuția lui S
  - Limbajul aserțiunilor este un limbaj de ordinul I.

- Tripletul $\{Pre\}\ S\ \{Post\}$ este (parțial) *corect* dacă:
  - dacă programul se execută dintr-o stare inițială care satisface *Pre*
  - și execuția se termină
  - atunci se ajunge într-o stare finală care satisface *Post*.

Semantica Axiomatică

Definește triplete (triplete Hoare) de forma

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

- Tripletul $\{Pre\}\ S\ \{Post\}$ este (parțial) *corect* dacă:
  - dacă programul se execută dintr-o stare inițială care satisface *Pre*
  - și execuția se termină
  - atunci se ajunge într-o stare finală care satisface *Post*.

Exemplu

- $\{\mathtt{x} = 1\}\ \mathtt{x} = \mathtt{x}+1\ \{\mathtt{x} = 2\}$ este corect
- $\{\mathtt{x} = 1\}$ `x = x+1` $\{\mathtt{x} = 3\}$ **nu** este corect
- $\{\top\}$ `if (x<=y) z=x; else z=y;` $\{\mathtt{z} = \mathit{min}(\mathtt{x}, \mathtt{y})\}$ este corect

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.

Sistem de reguli pentru logica Floyd-Hoare

$$(\rightarrow) \quad \frac{P1 \rightarrow P2 \quad \{P2\}\ c\ \{Q2\} \quad Q2 \rightarrow Q1}{\{P1\}\ c\ \{P2\}}$$

$$(\vee) \quad \frac{\{P1\}\ c\ \{Q\} \quad \{P2\}\ c\ \{Q\}}{\{P1 \vee P2\}\ c\ \{Q\}}$$

$$(\wedge) \quad \frac{\{P\}\ c\ \{Q1\} \quad \{P\}\ c\ \{Q2\}}{\{P\}\ c\ \{Q1 \wedge Q2\}}$$

Logica Floyd-Hoare pentru IMP

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

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

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

$$\text{(If)} \quad \frac{\{b \wedge P\}\ c1\ \{Q\} \quad \{\neg b \wedge P\}\ c2\ \{Q\}}{\{P\}\ \texttt{if}\ (b) c1\ \texttt{else}\ c2\ \{Q\}}$$

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

Logica Floyd-Hoare pentru IMP

- regula pentru atribuire

$$(\textsc{Asign}) \quad \frac{}{\{P[x/e]\}\ x = e;\ \{P\}}$$

Exemplu

$\{\mathtt{x} + \mathtt{y} = \mathtt{y} + 10\}$ `x = x+ y` $\{\mathtt{x} = \mathtt{y} + 10\}$

Logica Floyd-Hoare pentru IMP

- regula pentru atribuire

$$(\textsc{Asign}) \quad \frac{}{\{P[x/e]\}\ x = e;\ \{P\}}$$

Exemplu

$\{\mathtt{x} + \mathtt{y} = \mathtt{y} + 10\}$ `x = x+ y` $\{\mathtt{x} = \mathtt{y} + 10\}$

- regula pentru condiții

$$(\textsc{If}) \quad \frac{\{b \wedge P\}\ c1\ \{Q\} \quad \{\neg b \wedge P\}\ c2\ \{Q\}}{\{P\}\ \mathtt{if}\ (b) c1\ \mathtt{else}\ c2\ \{Q\}}$$

Exemplu

Pentru a demonstra $\{\top\}$ `if (x<=y) z=x; else z=y;` $\{\mathtt{z} = \mathit{min}(\mathtt{x}, \mathtt{y})\}$
este suficient să demonstrăm $\{\mathtt{x} \leq \mathtt{y}\}$ `z=x;` $\{\mathtt{z} = \mathit{min}(\mathtt{x}, \mathtt{y})\}$
și $\{\neg(\mathtt{x} \leq \mathtt{y})\}$ `z=y;` $\{\mathtt{z} = \mathit{min}(\mathtt{x}, \mathtt{y})\}$

Invarianți pentru `while`

Cum demonstrăm $\{P\}$ `while (b)c` $\{Q\}$?

- Se determină un invariant I și se folosește următoarea regulă:

$$\text{(Inv)} \quad \frac{P \to I \quad \{b \wedge I\}\ c\ \{I\} \quad (I \wedge \neg b) \to Q}{\{P\}\ \texttt{while}\ (b)\ c\ \{Q\}}$$

Invarianți pentru `while`

Cum demonstrăm $\{P\}$ `while (b)c` $\{Q\}$?

- Se determină un invariant I și se folosește următoarea regulă:

$$(\textsc{Inv}) \quad \frac{P \to I \quad \{b \wedge I\}\ c\ \{I\} \quad (I \wedge \neg b) \to Q}{\{P\}\ \texttt{while}\ (b)\ c\ \{Q\}}$$

Invariantul trebuie să satisfacăurmătoarele proprietăți:

- să fie adevărat inițial
- să rămână adevărat după executarea unui ciclu
- să implice postcondiția la ieșirea din buclă

Invarianți pentru `while`

$\{\mathtt{x} = n \ \wedge \ 0 \leq x \ \wedge \mathtt{y} = 1\}$

```
while (0 < x) {y = y * x;  x = x + -1;}
```

$\{\mathtt{y} = n!\}$

Invarianți pentru `while`

$\{\mathtt{x} = n \ \wedge \ 0 \leq x \ \wedge \mathtt{y} = 1\}$

```
while (0 < x) {y = y * x;  x = x + -1;}
```

$\{\mathtt{y} = n!\}$

- Invariantul *I* este $\mathtt{y} * x! = n! \ \wedge 0 \leq x$

Invarianți pentru `while`

$\{\mathtt{x} = n \ \wedge \ 0 \leq x \ \wedge \mathtt{y} = 1\}$

```
while (0 < x) {y = y * x;  x = x + -1;}
```

$\{\mathtt{y} = n!\}$

- Invariantul I este $\mathtt{y} * x! = n! \ \wedge 0 \leq x$
- $(\mathtt{x} = n \ \wedge \ 0 \leq x \ \wedge \mathtt{y} = 1) \rightarrow I$
- $\{I \wedge (0 < x)\}$ `y = y * x; x = x + -1;` $\{I\}$
- $I \wedge \neg(0 < \mathtt{x}) \rightarrow (\mathtt{y} = n!)$

Semantica denotațională

Semantica denotațională

- Introdusă de Christopher Strachey și Dana Scott (1970)
- 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

- Definim stările memoriei ca fiind funcții parțiale de la mulțimea identificatorilor la mulțimea valorilor:

$$State = Id \rightharpoonup \mathbb{Z}$$

- Asociem fiecărei categorii sintactice o categorie semantică.
- Fiecare construcție sintactică va avea o denotație (interpretare) în categoria semantică respectivă.

Semantica denotațională

- Definim stările memoriei ca fiind funcții parțiale de la mulțimea identificatorilor la mulțimea valorilor:

$$State = Id \rightharpoonup \mathbb{Z}$$

- Asociem fiecărei categorii sintactice o categorie semantică.
- Fiecare construcție sintactică va avea o denotație (interpretare) în categoria semantică respectivă.De exemplu:
  - denotația unei expresii aritmetice este o funcție parțială de la mulțimea stărilor memoriei la mulțimea valorilor ($\mathbb{Z}$):

$$[[\_]] : AExp \to (State \rightharpoonup \mathbb{Z})$$

  - denotația unei instrucțiuni este o funcție parțială de la mulțimea stărilor memoriei la mulțimea stărilor memoriei:

$$[[\_]] : Stmt \to (State \rightharpoonup State)$$

Semantica denotațională

$State = Id \rightharpoonup \mathbb{Z}$
$[\![\_]\!] : AExp \to (State \rightharpoonup \mathbb{Z})$
$[\![\_]\!] : Stmt \to (State \rightharpoonup State)$

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

$State = Id \rightharpoonup \mathbb{Z}$
$[[\_]] : AExp \to (State \rightharpoonup \mathbb{Z})$
$[[\_]] : Stmt \to (State \rightharpoonup State)$

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 a limbajului IMP

$$State = Id \rightharpoonup \mathbb{Z}$$

- Domenii semantice:
  $[\![\_]\!] : AExp \to (State \rightharpoonup \mathbb{Z})$
  $[\![\_]\!] : BExp \to (State \rightharpoonup \{T, F\})$
  $[\![\_]\!] : Stmt \to (State \rightharpoonup State)$

Semantica denotațională a limbajului IMP

$$State = Id \rightharpoonup \mathbb{Z}$$

- Domenii semantice:
  $[[\_]] : AExp \to (State \rightharpoonup \mathbb{Z})$
  $[[\_]] : BExp \to (State \rightharpoonup \{T, F\})$
  $[[\_]] : Stmt \to (State \rightharpoonup State)$
- Semantica denotațională este compozițională:
  - semantica expresiilor aritmetice
    $[[n]](s) = n$
    $[[x]](s) = s(x)$
    $[[e1 + e2]](s) = [[e1]](s) + [[e2]](s)$

Semantica denotațională a limbajului IMP

$$State = Id \rightharpoonup \mathbb{Z}$$

- Domenii semantice:
  $[[\_]] : AExp \to (State \rightharpoonup \mathbb{Z})$
  $[[\_]] : BExp \to (State \rightharpoonup \{T, F\})$
  $[[\_]] : Stmt \to (State \rightharpoonup State)$
- Semantica denotațională este compozițională:
  - semantica expresiilor aritmetice
    $[[n]](s) = n$
    $[[x]](s) = s(x)$
    $[[e1 + e2]](s) = [[e1]](s) + [[e2]](s)$
  - semantica expresiilor booleene
    $[[true]](s) = T$, $[[false]](s) = F$
    $[[!b]](s) = \neg b$
    $[[e1 <= e2]](s) = [[e1]](s) <= [[e2]](s)$

Semantica denotațională a limbajului IMP

$$State = Id \rightharpoonup \mathbb{Z}$$

- Domenii semantice:
  $[\![\_]\!] : AExp \to (State \rightharpoonup \mathbb{Z})$
  $[\![\_]\!] : BExp \to (State \rightharpoonup \{T, F\})$
  $[\![\_]\!] : Stmt \to (State \rightharpoonup State)$

Semantica denotațională a limbajului IMP

$$State = Id \rightharpoonup \mathbb{Z}$$

- Domenii semantice:
  $[[\_]] : AExp \to (State \rightharpoonup \mathbb{Z})$
  $[[\_]] : BExp \to (State \rightharpoonup \{T, F\})$
  $[[\_]] : Stmt \to (State \rightharpoonup State)$
- Semantica instrucțiunilor:

$$[[\texttt{skip}]] = id$$

$$[[\texttt{c1;c2}]] = [[c2]] \circ [[c1]]$$

$$[[\texttt{x = e}]](s)(y) = \begin{cases} s(y), \text{ dacă } y \neq x \\ [[e]](s), \text{ dacă } y = x \end{cases}$$

Semantica denotațională a limbajului IMP

$$State = Id \rightharpoonup \mathbb{Z}$$

- Domenii semantice:
  $[[\_]] : AExp \to (State \rightharpoonup \mathbb{Z})$
  $[[\_]] : BExp \to (State \rightharpoonup \{T, F\})$
  $[[\_]] : Stmt \to (State \rightharpoonup State)$
- Semantica instrucțiunilor:

$$[[\texttt{skip}]] = id$$

$$[[\texttt{c1;c2}]] = [[c2]] \circ [[c1]]$$

$$[[\texttt{x = e}]](s)(y) = \begin{cases} s(y), \text{ dacă } y \neq x \\ [[e]](s), \text{ dacă } y = x \end{cases}$$

$$[[\texttt{if (b) c1 else c2}]](s) = \begin{cases} [[c1]](s), \text{ dacă } [[b]](s) = T \\ [[c2]](s), \text{ dacă } [[b]](s) = F \end{cases}$$

Semantica denotațională a limbajului IMP

Exemplu

```
if (x<= y) z=x; else z=y;
```

$$[[pgm]](s) = \begin{cases} [[z = x;]](s), \text{ dacă } [[x <= y]](s) = T \\ [[z = y;]](s), \text{ dacă } [[x <= y]](s) = F \end{cases}$$

Semantica denotațională a limbajului IMP

Exemplu

```
if (x<= y) z=x; else z=y;
```

$$[[pgm]](s) = \begin{cases} [[z = x;]](s), \text{ dacă } [[x <= y]](s) = T \\ [[z = y;]](s), \text{ dacă } [[x <= y]](s) = F \end{cases}$$

$$[[pgm]](s)(v) = \begin{cases} s(v), \text{ dacă } s(x) \leq s(y), v \neq z \\ s(x), \text{ dacă } s(x) \leq s(y), v = z \\ s(v), \text{ dacă } s(x) > s(y), v \neq z \\ s(y), \text{ dacă } s(x) > s(y), v = z \end{cases}$$

Semantica denotațională a limbajului IMP

Exemplu

```
if (x<= y) z=x; else z=y;
```

$$[[pgm]](s) = \begin{cases} [[z = x;]](s), \text{ dacă } [[x <= y]](s) = T \\ [[z = y;]](s), \text{ dacă } [[x <= y]](s) = F \end{cases}$$

$$[[pgm]](s)(v) = \begin{cases} s(v), \text{ dacă } s(x) \leq s(y), v \neq z \\ s(x), \text{ dacă } s(x) \leq s(y), v = z \\ s(v), \text{ dacă } s(x) > s(y), v \neq z \\ s(y), \text{ dacă } s(x) > s(y), v = z \end{cases}$$

Cum definim semantica denotațională pentru `while`?

Mulțimea funcțiilor parțiale

Fie X și Y două mulțimi.

- $Pfn(X, Y)$ mulțimea funcțiilor parțiale de la X la Y, adică $Pfn(X, Y) = X \rightharpoonup Y$
- Pentru $f \in Pfn(X, Y)$ notăm cu $dom(f)$ mulțimea elementelor din X pentru care funcția este definită.
  Atunci $dom(f) \subseteq X$ și $f|_{dom(f)} : dom(f) \to Y$ este funcție.

Mulțimea funcțiilor parțiale

Fie X și Y două mulțimi.

- $Pfn(X, Y)$ mulțimea funcțiilor parțiale de la X la Y, adică $Pfn(X, Y) = X \rightharpoonup Y$
- Pentru $f \in Pfn(X, Y)$ notăm cu $dom(f)$ mulțimea elementelor din X pentru care funcția este definită.
  Atunci $dom(f) \subseteq X$ și $f|_{dom(f)} : dom(f) \to Y$ este funcție.
- Fie $\perp : X \rightharpoonup Y$ unica funcție cu $dom(\perp) = \emptyset$ (funcția care nu este definită în nici un punct).
- Definim pe $Pfn(X, Y)$ următoarea relație:

$$f \sqsubseteq g \text{ dacă și numai dacă } dom(f) \subseteq dom(g) \text{ și } g|_{dom(f)} = f_{dom(f)}$$

Mulțimea funcțiilor parțiale

Fie X și Y două mulțimi.

- $Pfn(X, Y)$ mulțimea funcțiilor parțiale de la X la Y, adică $Pfn(X, Y) = X \rightharpoonup Y$
- Pentru $f \in Pfn(X, Y)$ notăm cu $dom(f)$ mulțimea elementelor din X pentru care funcția este definită.
  Atunci $dom(f) \subseteq X$ și $f|_{dom(f)} : dom(f) \to Y$ este funcție.
- Fie $\perp : X \rightharpoonup Y$ unica funcție cu $dom(\perp) = \emptyset$ (funcția care nu este definită în nici un punct).
- Definim pe $Pfn(X, Y)$ următoarea relație:

$$f \sqsubseteq g \text{ dacă și numai dacă } dom(f) \subseteq dom(g) \text{ și } g|_{dom(f)} = f_{dom(f)}$$

$(Pfn(X, Y), \sqsubseteq, \perp)$ este CPO

(mulțime parțial ordonată completă în care $\perp$ este cel mai mic element)

$(Pfn(X, Y), \sqsubseteq, \perp)$ este CPO

Exemplu

Definim $\mathbf{F} : Pfn(\mathbb{N}, \mathbb{N}) \to Pfn(\mathbb{N}, \mathbb{N})$ prin

$$\mathbf{F}(g)(k) = \begin{cases} 1, & \text{dacă } k = 0, \\ k * g(k-1) & \text{dacă } k > 0 \text{ și } (k-1) \in dom(g), \\ \text{nedefinit}, & \text{altfel} \end{cases}$$

- $\mathbf{F}$ este o funcție continuă,

$(Pfn(X, Y), \sqsubseteq, \perp)$ este CPO

Exemplu

Definim $\mathbf{F} : Pfn(\mathbb{N}, \mathbb{N}) \to Pfn(\mathbb{N}, \mathbb{N})$ prin

$$\mathbf{F}(g)(k) = \begin{cases} 1, & \text{dacă } k = 0, \\ k * g(k-1) & \text{dacă } k > 0 \text{ și } (k-1) \in dom(g), \\ \text{nedefinit}, & \text{altfel} \end{cases}$$

- $\mathbf{F}$ este o funcție continuă,deci putem aplica
- Teorema Knaster-Tarski
  Fie $g_n = \mathbf{F}^n(\perp)$ și $f = \bigvee_n g_n$.
  Știm că f este cel mai mic punct fix al funcției $\mathbf{F}$, deci $\mathbf{F}(f) = f$.

$(Pfn(X, Y), \sqsubseteq, \perp)$ este CPO

Exemplu

Definim $\mathbf{F} : Pfn(\mathbb{N}, \mathbb{N}) \to Pfn(\mathbb{N}, \mathbb{N})$ prin

$$\mathbf{F}(g)(k) = \begin{cases} 1, & \text{dacă } k = 0, \\ k * g(k-1) & \text{dacă } k > 0 \text{ și } (k-1) \in dom(g), \\ \text{nedefinit}, & \text{altfel} \end{cases}$$

- $\mathbf{F}$ este o funcție continuă,deci putem aplica
- Teorema Knaster-Tarski
  Fie $g_n = \mathbf{F}^n(\perp)$ și $f = \bigvee_n g_n$.
  Știm că f este cel mai mic punct fix al funcției $\mathbf{F}$, deci $\mathbf{F}(f) = f$.
- Demonstrăm prin inducție după n că:

  $dom(g_n) = \{0, \ldots, n\}$ și $g_n(k) = k!$ oricare $k \in dom(g_n)$
- $f : \mathbb{N} \to \mathbb{N}$ este funcția factorial.

Semantica denotațională pentru `while`

`while (b) c`

- Definim $\mathbf{F} : \mathit{Pfn}(\mathit{State}, \mathit{State}) \to \mathit{Pfn}(\mathit{State}, \mathit{State})$ prin
- $\mathbf{F}$ este continuă
- Teorema Knaster-Tarski: $\mathit{fix}(\mathbf{F}) = \bigcup_n \mathbf{F}^n(\bot)$

Semantica denotațională pentru `while`

`while (b) c`

- Definim $\mathbf{F} : \mathit{Pfn}(\mathit{State}, \mathit{State}) \to \mathit{Pfn}(\mathit{State}, \mathit{State})$ prin

$$\mathbf{F}(g)(s) = \begin{cases} g([[c]](s)) & \text{dacă } [[b]](s) = T \\ s & \text{dacă } [[b]](s) = F \\ \text{nedefinit}, & \text{altfel} \end{cases}$$

- $\mathbf{F}$ este continuă
- Teorema Knaster-Tarski: $\mathit{fix}(\mathbf{F}) = \bigcup_n \mathbf{F}^n(\bot)$

Semantica denotațională pentru `while`

`while (b) c`

- Definim $\mathbf{F} : \mathit{Pfn}(\mathit{State}, \mathit{State}) \to \mathit{Pfn}(\mathit{State}, \mathit{State})$ prin

$$\mathbf{F}(g)(s) = \begin{cases} g([[c]](s)) & \text{dacă } [[b]](s) = T \\ s & \text{dacă } [[b]](s) = F \\ \text{nedefinit}, & \text{altfel} \end{cases}$$

- $\mathbf{F}$ este continuă
- Teorema Knaster-Tarski: $\mathit{fix}(\mathbf{F}) = \bigcup_n \mathbf{F}^n(\bot)$
- Semantica denotațională:

$$[[\_]] : \mathit{Stmt} \to (\mathit{State} \rightharpoonup \mathit{State})$$

$$[[\texttt{while (b) c}]](s) = \mathit{fix}(\mathbf{F})(s)$$

Semantica denotațională

Avantaje și dezavantaje

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

Semantica operațională (small-step)

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:

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

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică 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:

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

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:

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

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

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

Semantica small-step

- Introdusă de Gordon Plotkin (1981)
- Denumiri alternative:
  - Semantică 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:

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

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

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

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:

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

- Cum definim această 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:

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

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

Redex. Reguli structurale. Axiome

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

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

Redex. Reguli structurale. Axiome

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

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

Redex. Reguli structurale. Axiome

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

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

- 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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) $\langle x = i\ ;\ ,\ \sigma\rangle \to \langle \{\}\ ,\ \sigma'\rangle \quad \text{dacă } \sigma' = \sigma_{x \leftarrow i}$

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

Semantica small-step a lui IMP

Semantica atribuirii

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

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

Semantica lui if

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

- Semantica inițializărilor

(Init) $\langle \texttt{int}\ x = i\ ;\ p\ ,\ \sigma\rangle \to \langle p\ ,\ \sigma'\rangle \quad \textit{dacă}\ \sigma' = \sigma_{x \leftarrow i}$

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

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

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

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

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

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

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

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

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

Semantica small-step a lui IMP

Execuție pas cu pas

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

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

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

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

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

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

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

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

Semantica small-step a lui IMP

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

$$\langle \{\{\}\},\ i \mapsto -1 \rangle \xrightarrow{\text{Block-end}}$$

Semantica small-step a lui IMP

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

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

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

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 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

Vom defini limbajul IMP și semantica lui operațională în PROLOG!

Pe săptămâna viitoare!