Semantică operațională

Limbajul IMP

Traian Florin Șerbănuță

Departamentul de Informatică, FMI, UNIBUC
traian.serbanuta@fmi.unibuc.ro

14 octombrie 2014

Limbajul IMP

IMP este un limbaj IMPerativ foarte simplu.

Ce conține

- Operații aritmetice
  - Adunare
  - Comparare
- Operații cu memoria
  - Dereferențiere
  - Atribuire
- Instrucțiuni
  - Instrucțiunea vidă
  - Compunere secvențială
  - Instrucțiuni condiționale
  - Instrucțiuni de ciclare

$l_2 := 0 ;$
while $(0 <= !\, l_1)$ **do**
$(l_2 := !\, l_2 + !\, l_1 ;$
$l_1 := !\, l_1 + -1$
$)$

Unde starea inițială a memoriei este $\{l_1 \mapsto 3, l_2 \mapsto 0\}$.

Sintaxa limbajului IMP

Backus Naur Form

$$e ::= n \mid b \mid l \mid e\ op\ e \mid \texttt{if}\ e\ \texttt{then}\ e\ \texttt{else}\ e \mid\ !\ e \mid e\ \texttt{:=}\ e \mid \texttt{skip} \mid e\ \texttt{;}\ e \mid \texttt{while}\ e\ \texttt{do}\ e$$

$l ::=$ locație de memorie ($\mathbb{L} = \{l, l_0, l_1, l_2, \ldots\}$)
$n ::=$ număr întreg ($\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$)
$b ::=$ valoare de adevăr ($\mathbb{B} = \{\mathit{true}, \mathit{false}\}$)
$op ::=$ operație binară ($+, \leq$)

Backus Naur Form

- Pentru gramatici (generative) independente de context
- Producții — generează termeni prin expandare (rescriere)

  e ::= `if` e `then` e `else` e

  | l `:=` e
- Categorii sintactice (neterminale)
  - descriu tipurile de sintaxă
    - Tipuri lexicale: întregi (n), booleeni (b), locații (l), simboluri de operații (op)
    - Tipuri construite: expresii (e)
- Cuvinte cheie (terminale)
  - descriu elementele lexicale: `if`, `then`, `else`, `while`, do, `:=`, `!`, `;` ...

Sisteme de tranziție

Definiție (Sistem de tranziție)

Un sistem de tranziție este dat de

- O mulțime Config a configurațiilor (a stărilor)
- O relație „de tranziție“ $\rightarrow \subseteq \mathit{Config} \times \mathit{Config}$

Dacă $(c, c') \in \rightarrow$ scriem $c \rightarrow c'$ și citim c se poate transforma în c' (direct)

Exemplu: evaluarea expresiilor întregi

- Config e mulțimea expresiilor pe numere întregi cu operații $+$ și $*$
- $\rightarrow$ descrie un pas de evaluare
- $(3 + 5) * (7 + 3) \rightarrow 8 * (7 + 3) \rightarrow 8 * 10 \rightarrow 80$

Definiții derivate

$\rightarrow^*$ $c \rightarrow^* c'$ dacă c se transformă în c' in zero, unul, sau mai mulți pași (închiderea reflexiv-tranzitivă a relației $\rightarrow$)
$(3 + 5) * (7 + 3) \rightarrow^* 8 * 10$

$\not\rightarrow$ $c \not\rightarrow$ dacă c nu se mai poate transforma în nimic
$80 \not\rightarrow$, dar și $3/0 \not\rightarrow$

$\rightarrow^!$ $c \rightarrow^! c'$ dacă $c \rightarrow^* c'$ și $c' \not\rightarrow$, adică c' nu se mai poate transforma
$(3 + 5) * (7 + 3) \rightarrow^! 80$

Determinism Un sistem se numește puternic determinist dacă relația $\rightarrow$ e injectivă: $\forall c, c_1, c_2 \in \mathit{Config}, c \rightarrow c_1 \wedge c \rightarrow c_2 \implies c_1 = c_2$
Un sistem se numește determinist dacă relația $\rightarrow^!$ e injectivă.
Evaluarea expresiilor e deterministă, dar nu tare deterministă.
$(3 + 5) * (7 + 3) \rightarrow 8 * (7 + 3)$ și
$(3 + 5) * (7 + 3) \rightarrow (3 + 5) * 10$

Semantica tranzițională

- Introdusă în 1981 de Gordon Plotkin cu numele de
  Semantică Operațională Structurală (SOS)
- Denumiri alternative:
  „semantica pașilor mici”, „semantică prin reducere"
- Definește cel mai mic pas de execuție
- Relație „de tranziție” între configurații definită recursiv prin reguli:

$$\langle Cod, Stare\rangle \to \langle Cod', Stare'\rangle$$

- Fiecare pas de execuție este concluzia unei demonstrații
- Execuția se obține ca o succesiune de astfel de tranziții:
  $\langle l := ! \; l + 1, \{l \mapsto 0\}\rangle \longrightarrow \langle l := 0 + 1, \{l \mapsto 0\}\rangle \longrightarrow$
  $\langle l := 1, \{l \mapsto 0\}\rangle \longrightarrow \langle \texttt{skip}, \{l \mapsto 1\}\rangle \not\to$

Configurații finale și configurații blocate

Scopul semanticii tranziționale

Să descrie execuțiile posibile ca transformări ale programului și memoriei, pas cu pas, dintr-o configurație inițială într-una finală (sau blocată).

Configurații finale

Sunt configurații care conțin valori ca fragmente de program:

$\langle n, s\rangle$ $\langle \mathit{true}, s\rangle$ $\langle \mathit{false}, s\rangle$ $\langle \texttt{skip}, s\rangle$

Configurații blocate

Sunt configurații care nu sunt finale dar care nu mai pot tranziționa

$\langle 5 \leq 3 + \,!\, l_0, \{l_1 \mapsto 2\}\rangle$ $\langle \texttt{if } 0 \texttt{ then } e_1 \texttt{ else } e_2, s\rangle$ $\langle 3/0, s\rangle$

Starea memoriei

Starea memoriei unui program IMP la un moment dat este dată de valorile deținute în acel moment de locațiile folosite în program.

Matematic: o funcție parțială $s : \mathbb{L} \xrightarrow{\circ} \mathbb{Z}$ de domeniu finit.

Notații

- Descrierea funcției prin enumerare: $s = \{l_1 \mapsto 10, l_5 \mapsto 0\}$
- Funcția vidă $\emptyset$, nedefinită pentru nici o variabilă
- Obținerea valorii unei variabile: $s(x)$
- Suprascrierea valorii unei variabile:

$$s[l \mapsto n](l') = \begin{cases} s(l'), \text{ dacă } l' \neq l \\ n, \text{ dacă } l' = l \end{cases}$$

Redex. Reguli structurale. Axiome

Expresie reductibilă — redex

Reprezintă fragmentul de sintaxă care va fi modificat la următorul pas.

$$\texttt{if } 0 \texttt{ <= } 5 \texttt{ + } (7 \texttt{ * } \texttt{ ! } l_1) \texttt{ then } l_2 \texttt{ := } 1 \texttt{ else } l_2 \texttt{ := } 0$$

Reguli structurale — Folosesc la identificarea următorului redex

- Definite recursiv pe structura termenilor

$$\frac{\langle e, s\rangle \to \langle e', s'\rangle}{\langle \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2, s\rangle \to \langle \texttt{if } e' \texttt{ then } e_1 \texttt{ else } e_2, s'\rangle}$$

Axiome — Realizează pasul computaţional

$$\langle \texttt{if true then } e_1 \texttt{ else } e_2, s\rangle \to \langle e_1, s\rangle$$

Expresii aritmetice

- Axiomele efectuează operația în domeniu

(OP+) $\langle n_1 + n_2, s\rangle \to \langle n, s\rangle \quad \textit{dacă } n = n_1 + n_2$

(OP≤) $\langle n_1 <= n_2, s\rangle \to \langle b, s\rangle \quad \textit{dacă } b = (n_1 \leq n_2)$

- Regulile structurale descriu ordinea evaluării argumentelor

(OPS) $$\frac{\langle e_1, s\rangle \to \langle e_1', s'\rangle}{\langle e_1 \ op \ e_2, s\rangle \to \langle e_1' \ op \ e_2, s'\rangle}$$

(OPD) $$\frac{\langle e_2, s\rangle \to \langle e_2', s'\rangle}{\langle n_1 \ op \ e_2, s\rangle \to \langle n_1 \ op \ e_2', s'\rangle}$$

Exemplu

Exercițiu

Folosind regulile semantice de mai sus, găsiți configurația finală pentru $\langle (3+5)+(7+9), \emptyset \rangle$.

$$(\text{OpS}) \quad \frac{(\text{Op+}) \quad \frac{}{\langle 3+5, \emptyset \rangle \to \langle 8, \emptyset \rangle}\checkmark}{\langle (3+5)+(7+9), \emptyset \rangle \to \langle 8+(7+9), \emptyset \rangle}$$

$$(\text{OpD}) \quad \frac{(\text{Op+}) \quad \frac{}{\langle 7+9, \emptyset \rangle \to \langle 16, \emptyset \rangle}\checkmark}{\langle 8+(7+9), \emptyset \rangle \to \langle 8+16, \emptyset \rangle}$$

$$(\text{Op+}) \quad \frac{}{\langle 8+16, \emptyset \rangle \to \langle 24, \emptyset \rangle}\checkmark$$

Programare „imperativă“

Operații cu memoria

(Loc) $\langle !\ l, s\rangle \to \langle n, s\rangle \quad$ *dacă* $l \in Dom(s), n = s(l)$

(Atrib) $\langle l := n, s\rangle \to \langle \texttt{skip}, s[l \mapsto n]\rangle \quad$ *dacă* $l \in Dom(s)$

(AtribD) $$\frac{\langle e, s\rangle \to \langle e', s'\rangle}{\langle l := e, s\rangle \to \langle l := e', s'\rangle}$$

Compunerea secvențială

(Secv) $\langle \texttt{skip; } e_2, s\rangle \to \langle e_2, s\rangle$

(SecvS) $$\frac{\langle e_1, s\rangle \to \langle e_1', s'\rangle}{\langle e_1 \ ; \ e_2\rangle, s \to \langle e_1' \ ; \ e_2, s'\rangle}$$

Exemple

- $\langle l := 7 \ ; !l, \{l \mapsto 5\}\rangle \xrightarrow[SecvS]{Atrib} \langle \texttt{skip}; !l, \{l \mapsto 7\}\rangle \xrightarrow{Secv} \langle !l, \{l \mapsto 7\}\rangle \xrightarrow{Loc} \langle 7, \{l \mapsto 7\}\rangle$
- $\langle l := 7 \ ; \ l := !l, \{l \mapsto 5\}\rangle \longrightarrow ?$
- $\langle 15 + !l, \emptyset\rangle \longrightarrow ?$

Instrucțiunile condiționale și de ciclare

(IfTrue) $\langle \texttt{if true then } e_1 \texttt{ else } e_2, s\rangle \to \langle e_1, s\rangle$

(IfFalse) $\langle \texttt{if false then } e_1 \texttt{ else } e_2, s\rangle \to \langle e_2, s\rangle$

(IfS) $$\frac{\langle e, s\rangle \to \langle e', s'\rangle}{\langle \texttt{if } e \texttt{ then } e_1 \texttt{ else } e_2, s\rangle \to \langle \texttt{if } e' \texttt{ then } e_1 \texttt{ else } e_2, s'\rangle}$$

(While) $$\langle \texttt{while } e_1 \texttt{ do } e_2, s\rangle \to \langle \texttt{ if } e_1 \texttt{ then } (e_2 \texttt{ ; while } e_1 \texttt{ do } e_2) \texttt{ else skip}, s\rangle$$

Exemplu

Exercițiu

Dacă $e = \texttt{while}\ (1 <= !l_1)\ \text{do}\ (l_2 := !l_2 + !l_1\ ;\ l_1 := !l_1 + -1)$

și $s = \{l_1 \mapsto 3, l_2 \mapsto 0\}$

atunci $\langle e, s\rangle \longrightarrow ?$

Execuție pas cu pas

$$\langle \texttt{while } 0 <= \texttt{!}\ l \texttt{ do } l \texttt{ := !}\ l + -4, \{l \mapsto 3\}\rangle \xrightarrow{\textsc{While}}$$

$$\langle \texttt{if } 0 <= \texttt{!}\ l \texttt{ then } (l \texttt{ := !}\ l + -4 \texttt{ ; while } 0 <= \texttt{!}\ l \texttt{ do } l \texttt{ := !}\ l + -4) \texttt{ else skip}, \{l \mapsto 3\}\rangle \xrightarrow[\textsc{IfS, OpD}]{\textsc{Loc}}$$

$$\langle \texttt{if } 0 <= 3 \texttt{ then } (l \texttt{ := !}\ l + -4 \texttt{ ; while } 0 <= \texttt{!}\ l \texttt{ do } l \texttt{ := !}\ l + -4) \texttt{ else skip}, \{l \mapsto 3\}\rangle \xrightarrow[\textsc{IfS}]{\textsc{Op}\leq}$$

$$\langle \texttt{if } \mathit{true} \texttt{ then } (l \texttt{ := !}\ l + -4 \texttt{ ; while } 0 <= \texttt{!}\ l \texttt{ do } l \texttt{ := !}\ l + 4) \texttt{ else skip}, \{l \mapsto 3\}\rangle \xrightarrow{\textsc{IfTrue}}$$

$$\langle l \texttt{ := !}\ l + -4 \texttt{ ;while } 0 <= \texttt{!}\ l \texttt{ do } l \texttt{ := !}\ l + -4, \{l \mapsto 3\}\rangle \xrightarrow[\mathit{SecvS},\mathit{AtribD},\mathit{OpS}]{\textsc{Loc}}$$

$$\langle l \texttt{ := } 3 + -4 \texttt{ ;while } 0 <= \texttt{!}\ l \texttt{ do } l \texttt{ := !}\ l + -4, \{l \mapsto 3\}\rangle \xrightarrow[\textsc{SecvS, AtribD}]{\textsc{Op}+}$$

Execuție pas cu pas

$$\langle l := -1 \ ; \texttt{while}\ 0 <= !\ l\ \texttt{do}\ l := !\ l + -4, \{l \mapsto 3\}\rangle \xrightarrow[\text{SecvS}]{\text{Atrib}}$$

$$\langle\ \texttt{skip;while}\ 0 <= !\ l\ \texttt{do}\ l := !\ l + -4, \{l \mapsto -1\}\rangle \xrightarrow{\text{Secv}}$$

$$\langle\ \texttt{while}\ (0 <= !\ l)\ \texttt{do}\ l := !\ l + -4, \{l \mapsto -1\}\rangle \xrightarrow{\text{While}}$$

$$\langle \texttt{if}\ 0 <= \ !\ l\ \texttt{then}\ l := !\ l + -4\ ; \texttt{while}\ 0 <= !\ l\ \texttt{do}\ l := !\ l + -4\ \texttt{else skip}, \{l \mapsto -1\}\rangle \xrightarrow[\text{IfS,OpD}]{\text{Loc}}$$

$$\langle \texttt{if}\ 0 <= -1\ \texttt{then}\ l := !\ l + -4\ ; \texttt{while}\ 0 <= !\ l\ \texttt{do}\ l := !\ l + -4\ \texttt{else skip}, \{l \mapsto -1\}\rangle \xrightarrow[\text{IfS}]{\text{Op}\leq}$$

$$\langle\ \texttt{if}\ \textit{false}\ \texttt{then}\ l := !\ l + -4\ ; \texttt{while}\ 0 <= !\ l\ \texttt{do}\ l := !\ l + 4\ \texttt{else skip}, \{l \mapsto -1\}\rangle \xrightarrow{\text{IfFalse}}$$

$$\langle \texttt{skip}, \{l \mapsto -1\}\rangle$$

Determinism

Teoremă

Limbajul IMP este puternic determinist, adică, dacă $\langle e, s\rangle \to \langle e_1, s_1\rangle$ *și* $\langle e, s\rangle \to \langle e_2, s_2\rangle$, *atunci* $e_1 = e_2$ *și* $s_1 = s_2$.

Demonstrație

Va urma . . .