Fundamentele limbajelor de programare
CURS 2

Traian Florin Șerbănuță și Andrei Sipoș

Facultatea de Matematică și Informatică, DL Info, Anul II
Semestrul II, 2023/2024

Programare imperativă

Programarea imperativă

Paradigma de programare cel mai transparent de descris formal este cea imperativă, având în vedere că posedă foarte puține construcții care au inerent nevoie de un aparat matematic complex (cum ar fi rezoluția, funcțiile de ordin înalt etc. pe care le vom vedea în alte paradigme), și de aceea poate servi drept caz de jucărie pentru abstracțiunile pe care le vom dezvolta ulterior.

Vom descrie un limbaj imperativ auster, numit IMP, o variantă simplificată[1] a limbajelor imperative folosite în mod uzual (cum ar fi C, Go etc.).

[1] Denumirea IMP își are originea, cel mai probabil, în cartea (indicată în bibliografie): G. Winskel, *The Formal Semantics of Programming Languages: An Introduction*, The MIT Press, 1993.

Limbajul IMP

Fixăm o mulțime (probabil numărabilă, dar nu intrăm în detalii) L de „variabile”, unde cuvântul se vrea să aibă același sens ca într-un limbaj de programare imperativ uzual: practic, ne referim la locuri în memorie.

Descrierea limbajului va cuprinde diverse „tipuri de date”: expresii aritmetice, expresii booleene, instrucțiuni (așa cum aveam și la logica de ordinul I întâi termeni și abia apoi formule), pe care le vom descrie pe rând.

Expresii aritmetice

O expresie aritmetică are exact una dintre următoarele forme:

- un număr întreg n;
- o variabilă X (element al lui L);
- $a_0 + a_1$, $a_0 - a_1$, $a_0 * a_1$, unde a_0 și a_1 sunt expresii aritmetice.

Acesta (și cele care vor urma) sunt (la modul informal) tipuri de date inductive, iar, în genere, principiile exprimate și demonstrate în cursul trecut pentru exemplul particular al formulelor logicii propoziționale (inducție structurală, recursie structurală, citire unică etc.) vor fi valabile și aici, *mutadis mutandis*, și ca atare nu le vom mai menționa.

Expresii booleene

O expresie booleană are exact una dintre următoarele forme:

- o valoare booleană (**true** sau **false**);
- $a_0 = a_1$, $a_0 \leq a_1$, unde a_0 și a_1 sunt expresii aritmetice;
- $\neg b_0$, $b_0 \wedge b_1$, $b_0 \vee b_1$, unde b_0 și b_1 sunt expresii booleene.

Instrucțiuni

O instrucțiune are exact una dintre următoarele forme:

- **skip**;
- $X := a$, unde $X \in L$, iar a este o expresie aritmetică;
- $c_0; c_1$, unde c_0 și c_1 sunt instrucțiuni;
- **if** b **then** c_0 **else** c_1, unde b este expresie booleană, iar c_1 și c_2 sunt instrucțiuni;
- **while** b **do** c, unde b este expresie booleană, iar c este instrucțiune.

Stări

Vom numi stare o funcție de la L la $\mathbb{Z}$. Practic, fiecărui loc din memorie i se va asocia o valoare, în mod asemănător cu funcțiile de evaluare din logică. Mulțimea stărilor o vom nota cu Σ. Așadar, $\Sigma = \mathbb{Z}^L$.

Tot ca în logică, pentru orice $\sigma \in \Sigma$, $X \in L$ și $N \in \mathbb{N}$, definim starea $\sigma_{X \mapsto N}$, pentru orice $Y \in L$, prin:

$$\sigma_{X \mapsto N}(Y) := \begin{cases} N, & \text{dacă } Y = X, \\ \sigma(Y), & \text{altfel.} \end{cases}$$

Evaluarea expresiilor aritmetice

Fie $\sigma \in \Sigma$. Definim o funcție e_σ care evaluează expresii aritmetice în numere întregi, în mod recursiv, în felul următor:

- pentru orice $N \in \mathbb{Z}$, $e_\sigma(N) := N$;
- pentru orice $X \in L$, $e_\sigma(X) := \sigma(X)$;
- pentru orice expresii aritmetice a_0, a_1, avem
  $e_\sigma(a_0 + a_1) := e_\sigma(a_0) + e_\sigma(a_1)$,
  $e_\sigma(a_0 - a_1) := e_\sigma(a_0) - e_\sigma(a_1)$,
  $e_\sigma(a_0 * a_1) := e_\sigma(a_0) * e_\sigma(a_1)$.

Evaluarea expresiilor booleene

Tot pentru un $\sigma \in \Sigma$, vom defini o funcție notată tot e_σ care evaluează expresii booleene în mulțimea $\{0, 1\}$, în mod recursiv, în felul următor:

- $e_\sigma(\textbf{true}) = 1$, $e_\sigma(\textbf{false}) = 0$;
- pentru orice expresii aritmetice a_0, a_1, avem
  $e_\sigma(a_0 = a_1) = 1$ dacă și numai dacă $e_\sigma(a_0) = e_\sigma(a_1)$,
  $e_\sigma(a_0 \leq a_1) = 1$ dacă și numai dacă $e_\sigma(a_0) \leq e_\sigma(a_1)$;
- pentru orice expresii booleene b_0, b_1, avem
  $e_\sigma(\neg b_0) = \neg e_\sigma(b_0)$,
  $e_\sigma(b_0 \wedge b_1) = e_\sigma(b_0) \wedge e_\sigma(b_1)$,
  $e_\sigma(b_0 \vee b_1) = e_\sigma(b_0) \vee e_\sigma(b_1)$.

Semantica operațională structurală

În acest moment, putem trece la definirea primei semantici pentru limbajul IMP. Ea va fi o semantică operațională structurală big-step, în sensurile următoare:

- operațională: seamănă cu execuția de către o mașină;
- structurală: este definită conform principiului de recursie structurală (pe instrucțiuni);
- big-step: se va referi la starea finală după execuția unui fragment de cod (de aceea vom folosi semnul $\Downarrow$).

Semantica instrucțiunilor

Vom considera aserțiuni de forma $(c, \sigma) \Downarrow \sigma'$, unde c este o instrucțiune, iar σ, $\sigma' \in \Sigma$, cu semnificația informală „instrucțiunea c executată în starea σ produce starea σ'”.

Vom prezenta niște **reguli de deducție** care ne vor permite să derivăm asemenea aserțiuni. Am putea nota faptul că aserțiunea $(c, \sigma) \Downarrow \sigma'$ este derivabilă prin $\vdash (c, \sigma) \Downarrow \sigma'$, dar o notăm, mai simplu, doar cu $(c, \sigma) \Downarrow \sigma'$.

Vom putea face și inducție pe derivări/demonstrații (ori structural, ori după lungime), precum și pe aserțiuni derivabile (putem face și recursie? în ce sens? dacă da/nu în vreun sens, de ce?).

Regulile de deducție big-step

$$\frac{}{(\textbf{skip}, \sigma) \Downarrow \sigma}$$

$$\frac{}{(X := a, \sigma) \Downarrow \sigma_{X \mapsto e_\sigma(a)}} \qquad \frac{(c_0, \sigma) \Downarrow \sigma'' \quad (c_1, \sigma'') \Downarrow \sigma'}{(c_0; c_1, \sigma) \Downarrow \sigma'}$$

$$\frac{e_\sigma(b) = 1 \quad (c_0, \sigma) \Downarrow \sigma'}{(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma) \Downarrow \sigma'} \qquad \frac{e_\sigma(b) = 0 \quad (c_1, \sigma) \Downarrow \sigma'}{(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma) \Downarrow \sigma'}$$

$$\frac{e_\sigma(b) = 0}{(\textbf{while } b \textbf{ do } c, \sigma) \Downarrow \sigma}$$

$$\frac{e_\sigma(b) = 1 \quad (c, \sigma) \Downarrow \sigma'' \quad (\textbf{while } b \textbf{ do } c, \sigma'') \Downarrow \sigma'}{(\textbf{while } b \textbf{ do } c, \sigma) \Downarrow \sigma'}$$

Un exemplu de raționament

În acest moment, putem vedea cum putem folosi semantica pentru a raționa despre programe.

Propoziție

Fie $w :=$ **while true do skip**. Nu există σ, σ' cu $(w, \sigma) \Downarrow \sigma'$.

Demonstrație

Presupunem că ar exista o derivare a acelei aserțiuni. Considerăm una de lungime minimă. Ultima regulă a derivării trebuie să fie a doua regulă pentru **while**, așadar trebuie să existe o stare σ'' astfel încât în premisele regulii să apară $(\textbf{skip}, \sigma) \Downarrow \sigma''$ și $(w, \sigma'') \Downarrow \sigma'$. Prima premisă poate fi dedusă doar aplicând regula pentru **skip**, așadar $\sigma = \sigma''$, deci a doua premisă trebuie să fie $(w, \sigma) \Downarrow \sigma'$, care este exact aserțiunea despre care vorbim, contrazicându-se minimalitatea lungimii derivării.

Echivalențe

Pe instrucțiuni, putem defini relația de echivalență $\sim$ în felul următor: pentru orice c_1, c_2, avem $c_1 \sim c_2$ exact atunci când, pentru orice σ, $\sigma' \in \Sigma$, $(c_0, \sigma) \Downarrow \sigma'$ dacă și numai dacă $(c_1, \sigma) \Downarrow \sigma'$.

Propoziție

Fie b o expresie booleană și c o instrucțiune. Notăm $w := \textbf{while } b \textbf{ do } c$. Atunci $w \sim (\textbf{if } b \textbf{ then } (c; w) \textbf{ else skip})$.

Demonstrație

Fie σ, $\sigma' \in \Sigma$. Vrem să arătăm că

$$(w, \sigma) \Downarrow \sigma' \Leftrightarrow (\textbf{if } b \textbf{ then } (c; w) \textbf{ else skip}, \sigma) \Downarrow \sigma'.$$

Demonstrăm implicația „$\Rightarrow$”. Presupunem că avem $(w, \sigma) \Downarrow \sigma'$. În primul caz, avem $e_\sigma(b) = 0$ și $\sigma = \sigma'$, iar, folosind faptul că $(\textbf{skip}, \sigma) \Downarrow \sigma$, deducem că $(\textbf{if } b \textbf{ then } (c; w) \textbf{ else skip}, \sigma) \Downarrow \sigma$.

Demonstrație (cont.)

În al doilea caz, avem că $e_\sigma(b) = 1$ și există σ'' cu $(c, \sigma) \Downarrow \sigma''$ și $(w, \sigma'') \Downarrow \sigma'$. Putem deduce că $(c; w, \sigma) \Downarrow \sigma'$ și apoi că $(\textbf{if } b \textbf{ then } (c; w) \textbf{ else skip}, \sigma) \Downarrow \sigma'$.

Demonstrăm acum implicația „$\Leftarrow$”. Presupunem că avem $(\textbf{if } b \textbf{ then } (c; w) \textbf{ else skip}, \sigma) \Downarrow \sigma'$.

În primul caz, avem $e_\sigma(b) = 0$ și $\sigma = \sigma'$, de unde deducem imediat $(w, \sigma) \Downarrow \sigma$.

În al doilea caz, avem $e_\sigma(b) = 1$ și $(c; w, \sigma) \Downarrow \sigma'$, de unde scoatem că există σ'' cu $(c, \sigma) \Downarrow \sigma''$ și $(w, \sigma'') \Downarrow \sigma'$. Putem apoi deduce imediat că $(w, \sigma) \Downarrow \sigma'$.

Semantica small-step

Există și un mod „small-step" de a defini semantica operațională structurală, care se referă la modul în care se modifică starea după rularea „unui pas" din program (cât anume reprezintă un pas este la latitudinea celui care definește semantica; de exemplu, aici am făcut deja o asemenea alegere prin faptul că expresiile sunt evaluate funcțional, iar nu prin reguli de deducție).

Ne vom referi acum la aserțiuni de forma $(c, \sigma) \to (c', \sigma')$ unde c și c' sunt instrucțiuni, iar σ, $\sigma' \in \Sigma$, cu semnificația informală „după ce executăm instrucțiunea c în starea σ, starea devine σ', iar instrucțiunea care mai rămâne de executat este c'". Vom nota cu $\to^*$ închiderea reflexiv-tranzitivă a relației $\to$ (cum o putem defini formal? discuție!).

Prezentăm în continuare regulile de deducție.

Regulile de deducție small-step

$$\frac{}{(X := a, \sigma) \to (\textbf{skip}, \sigma_{X \mapsto e_\sigma(a)})}$$

$$\frac{}{(\textbf{skip}; c_1, \sigma) \to (c_1, \sigma)} \qquad \frac{(c_0, \sigma) \to (c_0', \sigma')}{(c_0; c_1, \sigma) \to (c_0'; c_1, \sigma')}$$

$$\frac{e_\sigma(b) = 1}{(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma) \to (c_0, \sigma)}$$

$$\frac{e_\sigma(b) = 0}{(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma) \to (c_1, \sigma)}$$

$$\frac{}{(\textbf{while } b \textbf{ do } c, \sigma) \to (\textbf{if } b \textbf{ then } (c; (\textbf{while } b \textbf{ do } c)) \textbf{ else skip}, \sigma)}$$

Echivalența între semantici

Teoremă

Fie c o instrucțiune și σ, $\sigma' \in \Sigma$. Atunci

$$(c, \sigma) \Downarrow \sigma' \text{ dacă și numai dacă } (c, \sigma) \to^* (\textbf{skip}, \sigma').$$

Demonstrație

Demonstrăm implicația „$\Rightarrow$”. Facem inducție după regulile pentru relația $\Downarrow$.

Tratăm cazul regulii $(\textbf{skip}, \sigma) \Downarrow \sigma$. Atunci, clar, $(\textbf{skip}, \sigma) \to^* (\textbf{skip}, \sigma)$. Cazul regulii pentru atribuire este similar.

La regula pentru secvențiere, deducem, din ipoteza de inducție, $(c_0, \sigma) \to^* (\textbf{skip}, \sigma'')$ și $(c_1, \sigma'') \to^* (\textbf{skip}, \sigma')$, de unde scoatem

$$(c_0; c_1, \sigma) \to^* (\textbf{skip}; c_1, \sigma'') \to (c_1, \sigma'') \to^* (\textbf{skip}, \sigma').$$

Echivalența între semantici

Demonstrație (cont.)

La prima regulă pentru **if** (cea de-a doua se tratează similar), deducem, din ipoteza de inducție, $(c_0, \sigma) \to^* (\textbf{skip}, \sigma')$, de unde scoatem

$$(\textbf{if } b \textbf{ then } c_0 \textbf{ else } c_1, \sigma) \to (c_0, \sigma) \to^* (\textbf{skip}, \sigma').$$

Notăm $w := \textbf{while } b \textbf{ do } c$. La prima regulă pentru **while**, unde $e_\sigma(b) = 0$, scoatem

$$(w, \sigma) \to (\textbf{if } b \textbf{ then } (c; w) \textbf{ else skip}, \sigma) \to (\textbf{skip}, \sigma).$$

La a doua, știm $(c, \sigma) \to^* (\textbf{skip}, \sigma'')$ și $(w, \sigma'') \to^* (\textbf{skip}, \sigma')$ și scoatem

$$\begin{aligned}(w, \sigma) &\to (\textbf{if } b \textbf{ then } (c; w) \textbf{ else skip}, \sigma) \\ &\to (c; w, \sigma) \to^* (\textbf{skip}; w, \sigma'') \to (w, \sigma'') \to^* (\textbf{skip}, \sigma').\end{aligned}$$

Echivalența între semantici

Demonstrație (cont.)

Demonstrăm implicația „$\Leftarrow$”. Acum vom face inducție structurală după c.

Pentru instrucțiunea **skip**, dacă avem $(\textbf{skip}, \sigma) \to^* (\textbf{skip}, \sigma')$, atunci neapărat $\sigma = \sigma'$ și, clar, avem $(\textbf{skip}, \sigma) \Downarrow \sigma$. Cazul instrucțiunii de atribuire este similar.

Dacă știm $(c_0; c_1, \sigma) \to^* (\textbf{skip}, \sigma')$, clar trebuie să avem

$$(c_0; c_1, \sigma) \to^* (\textbf{skip}; c_1, \sigma'') \to (c_1, \sigma'') \to^* (\textbf{skip}, \sigma')$$

și în particular $(c_0, \sigma) \to^* (\textbf{skip}, \sigma'')$. De aici scoatem $(c_0, \sigma) \Downarrow \sigma''$ și $(c_1, \sigma'') \Downarrow \sigma'$, deci $(c_0; c_1, \sigma) \Downarrow \sigma'$.

Cazul instrucțiunii **if** rămâne ca exercițiu.

Demonstrație (cont.)

Tratăm cazul instrucțiunii **while**. Presupunem că avem $(\textbf{while } b \textbf{ do } c, \sigma) \to^* (\textbf{skip}, \sigma')$. Notăm $w := \textbf{while } b \textbf{ do } c$. Vom face inducție după numărul de pași din „$\to^*$”.

Pentru cazul $e_\sigma(b) = 0$, avem $(\textbf{while } b \textbf{ do } c, \sigma) \to^* (\textbf{skip}, \sigma)$, deci $\sigma' = \sigma$ și scoatem ușor $(w, \sigma) \Downarrow \sigma$.

Pentru cazul $e_\sigma(b) = 1$, avem $(\textbf{while } b \textbf{ do } c, \sigma) \to^* (c; w, \sigma) \to^* (\textbf{skip}; w, \sigma'') \to (w, \sigma'') \to^* (\textbf{skip}, \sigma')$. În particular, avem $(c, \sigma) \to^* (\textbf{skip}, \sigma'')$ și $(w, \sigma'') \to^* (\textbf{skip}, \sigma')$. Din ipoteza de inducție structurală pentru c, avem $(c, \sigma) \Downarrow \sigma''$. Din ipoteza de inducție pe numărul de pași, avem $(w, \sigma'') \Downarrow \sigma'$. Acum scoatem imediat $(w, \sigma) \Downarrow \sigma'$.