Tipuri

Definirea unui sistem de tipuri

Traian Florin Șerbănuță

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

30 octombrie 2014

Este IMP prea expresiv?

- Sunt programe care n-aș vrea să le pot scrie, dar le pot?

  $3 + \mathit{false}$ `if 2 then skip else skip`
- Putem detecta programe greșite înainte de rulare?
- 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

Scop (ideal)

Progamele „bine formate“, i. e., cărora li se poate asocia un tip nu eșuează

Sisteme de tipuri

Intuiție

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

Exemple

$$\vdash \quad \texttt{if } \mathit{true} \texttt{ then } 2 \texttt{ else } 3 + 4 : \texttt{int}$$

$$l_1 : \texttt{intref} \quad \vdash \quad \texttt{if!}l_1 \geq 3 \texttt{ then!}l_1 \texttt{ else } 3 \quad : \texttt{int}$$

$$\nvdash \quad 3 + \mathit{false} \quad : T \qquad \text{pentru orice } T$$

$$\nvdash \quad \texttt{if } \mathit{true} \texttt{ then } 3 \texttt{ else } \mathit{false} : \texttt{int}$$

Sisteme de tipuri

Intuiție

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

Exemple

$$\vdash \quad \texttt{if } \mathit{true} \texttt{ then } 2 \texttt{ else } 3 + 4 : \texttt{int}$$

$$l_1 : \texttt{intref} \quad \vdash \quad \texttt{if!}l_1 \geq 3 \texttt{ then!}l_1 \texttt{ else } 3 \quad : \texttt{int}$$

$$\nvdash \quad 3 + \mathit{false} \quad : T \qquad \text{pentru orice } T$$

$$\nvdash \quad \texttt{if } \mathit{true} \texttt{ then } 3 \texttt{ else } \mathit{false} : \texttt{int}$$

Tipuri în limbajul IMP

Tipurile expresiilor = tipurile valorilor

$$T ::= \texttt{int} \mid \texttt{bool} \mid \texttt{unit}$$

Tipurile locațiilor

$$T_{loc} ::= \texttt{intref}$$

Fie $\mathbb{T}$ și $\mathbb{T}_{loc}$ mulțimile defiinite de T și respectiv T_{loc}.

Γ — Mediul de tipuri

- Asociază tipuri locațiilor
- Funcție parțială (finită) de la $\mathbb{L}$ la $\mathbb{T}_{loc}$, asemănătoare memoriei

$$\Gamma : \mathbb{L} \xrightarrow{\circ} \mathbb{T}_{loc}$$

- Notație: o listă de perechi locație-tip

$$l_1 : \texttt{intref}, \ldots, l_n : \texttt{intref}$$

Observații pentru limbajul IMP

- Toate locațiile din Γ au același tip: `intref`
- Apariția unei locații în Γ înseamnă că locația e de fapt definită

IMP: Reguli pentru tipuri

Expresii aritmmetice

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

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

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

$$(\textsc{op}\leq) \quad \frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \texttt{ <= } e_2 : \texttt{bool}}$$

$$(\textsc{if}) \quad \frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : T \quad \Gamma \vdash e_3 : T}{\Gamma \vdash \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 : T}$$

IMP: Reguli pentru tipuri

Exemplu

Arătați că $\vdash \texttt{if}\ \mathit{false}\ \texttt{then}\ 2\ \texttt{else}\ 3 + 4\ : \texttt{int}$

IMP: Reguli pentru tipuri

Exemplu

Arătați că $\vdash \texttt{if}\ \mathit{false}\ \texttt{then}\ 2\ \texttt{else}\ 3 + 4\ : \texttt{int}$

$$(\textsc{if})\ \frac{(\textsc{bool})\ \dfrac{}{\vdash\ \mathit{false}\ : \texttt{bool}}\ \checkmark \qquad (\textsc{int})\ \dfrac{}{\vdash\ 2\ : \texttt{int}}\ \checkmark \qquad \vdash\ 3 + 4\ : \texttt{int}}{\vdash \texttt{if}\ \mathit{false}\ \texttt{then}\ 2\ \texttt{else}\ 3 + 4\ : \texttt{int}}$$

unde

$$(\textsc{op+})\ \frac{(\textsc{int})\ \dfrac{}{\vdash\ 3\ : \texttt{int}}\ \checkmark \qquad (\textsc{int})\ \dfrac{}{\vdash\ 4\ : \texttt{int}}\ \checkmark}{\vdash\ 3 + 4\ : \texttt{int}}$$

IMP: Reguli pentru tipuri

Referințe

$$(\textsc{atrib}) \quad \frac{\Gamma \vdash e : \texttt{int}}{\Gamma \vdash l := e : \texttt{unit}} \quad \textit{dacă } \Gamma(l) = \texttt{intref}$$

$$(\textsc{loc}) \quad \Gamma \vdash !l : \textit{int} \quad \textit{dacă } \Gamma(l) = \texttt{intref}$$

- Pentru IMP, $\Gamma(l) = \texttt{intref}$ poate fi citit ca: locația l e definită

IMP: Reguli pentru tipuri

Programare imperativă

$$(\textsc{skip}) \quad \Gamma \vdash \texttt{skip} : \texttt{unit}$$

$$(\textsc{secv}) \quad \frac{\Gamma \vdash e_1 : \texttt{unit} \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 \texttt{ ; } e_2 : T}$$

$$(\textsc{while}) \quad \frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{unit}}{\Gamma \vdash \texttt{while } e_1 \texttt{ do } e_2 \texttt{ done} : \texttt{unit}}$$

Proprietăți

ale sistemului de tipuri pentru limbajul IMP

Teoremă (Proprietatea de a progresa)

Dacă $\Gamma \vdash e : T$ *și* $Dom(\Gamma) \subseteq Dom(s)$ *atunci* e *este valoare sau* $\langle e, s\rangle$ *poate progresa: există* e', s' *astfel încât* $\langle e, s\rangle \to \langle e', s'\rangle$.

Teoremă (Proprietatea de conservare a tipului)

Dacă $\Gamma \vdash e : T$, $Dom(\Gamma) \subseteq Dom(s)$ *și* $\langle e, s\rangle \to \langle e', s'\rangle$, *atunci* $\Gamma \vdash e' : T$ *și* $Dom(\Gamma) \subseteq Dom(s')$.

Teoremă (Siguranță—programele bine formate nu se împotmolesc)

Dacă $\Gamma \vdash e : T$, $Dom(\Gamma) \subseteq Dom(s)$ *și* $\langle e, s\rangle \longrightarrow^* \langle e', s'\rangle$, *atunci* e' *este valoare sau există* e'', s'' *astfel încât* $\langle e', s'\rangle \to \langle e'', s''\rangle$.

Probleme computaționale

Verificarea tipului

Date fiind Γ, e și T, verificați dacă $\Gamma \vdash e : T$.

Determinarea (inferarea) tipului

Date fiind Γ și e, găsiți (sau arătați ce nu există) un T astfel încât $\Gamma \vdash e : T$.

- A doua problemă e mai grea în general decât prima
- Algoritmi de inferare a tipurilor
  - Colectează constrângeri asupra tipului
  - Folosesc metode de rezolvare a constrângerilor (programare logică)
- Pentru limbajul nostru ambele probleme sunt ușoare

Probleme computaționale

Proprietăți

Teoremă (Determinarea tipului este decidabilă)

Date fiind Γ *și* e*, poate fi găsit (sau demonstrat că nu există) un* T *astfel încât* $\Gamma \vdash e : T$.

Teoremă (Verificarea tipului este decidabilă)

Date fiind Γ*,* e *și* T*, problema* $\Gamma \vdash e : T$ *este decidabilă.*

Teoremă (Unicitatea tipului)

Dacă $\Gamma \vdash e : T$ *și* $\Gamma \vdash e : T'$*, atunci* $T = T'$.

Determinarea tipului

Implementare

- Reprezentăm tipurile ca variante

  type tip = TInt | TBool | TUnit

  type tipL = TIntRef

- Reprezentăm mediul de tipuri ca listă de perechi locație-tip de locație

  type mediuTip = (l * tipL) list

- Implementăm regulile de tipuri ca o funcție parțială

  val infertype : MediuTip -> e -> tipL option

Algoritm de determinare a tipurilor in IMP

```
function Int  n –> Some TInt
        | Bool b –> Some TBool
        | Op(e1,Plus,e2) –> (match (infertype m e1, infertype m e2) with
            | (Some TInt, Some TInt) –> Some TInt
            | _ –> None)
        | Op(e1,Mic,e2) –> (match (infertype m e1, infertype m e2) with
            | (Some TInt, Some TInt) –> Some TBool
            | _ –> None)
        |  If (e1,e2,e3) –> (match (infertype m e1, infertype  m e2, infertype  m e3) with
            | (Some TBool, Some t, Some t') when t=t' –> Some t
            | _ –> None)
        | Loc l  –> (match lookup l m with
            | Some (Ref TInt) –> Some TInt
            | _ –> None)
        |  Atrib (l ,e)  –> (match (lookup l m, infertype  m e) with
            | (Some (Ref TInt), Some TInt) –> Some TUnit
            | _ –> None)
        | Skip –> Some TUnit
        | Secv (e1,e2) –> (match (infertype m e1) with
            | Some TUnit –> infertype m e2
            | _ –> None)
        | While (e1,e2) –> (match (infertype m e1, infertype  m e2) with
            | (Some TBool, Some TUnit) –> Some TUnit
            | _ –> None)
```

Algoritm de determinare a tipurilor in IMP

If

Regula matematică

$$(\text{IF}) \quad \frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : T \quad \Gamma \vdash e_3 : T}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : T}$$

Implementarea în OCaml

```
let infertype m = function
  ...
| If (e1,e2,e3) -> (match (infertype m e1, infertype m e2, infertype m e3) with
    | (Some TBool, Some t, Some t') when t=t' -> Some t
    | _ -> None)
```

Algoritm de determinare a tipurilor in IMP

Dereferențierea

Regula matematică

$$\text{(Loc)} \quad \Gamma \vdash\ !l\ :\ \texttt{int} \quad \textit{dacă}\ \Gamma(l) = \texttt{intref}$$

Implementarea în OCaml

```
let infertype m = function
  ...
| Loc l -> (match lookup l m with
    | Some (Ref TInt) -> Some TInt
    | _ -> None)
```

IMP $\subseteq$ OCaml

- IMP este un fragment al limbajului OCaml
- Un program IMP este o expresie OCaml în care
  - Locațiile sunt valori de tip referință (predefinite)
  - instrucțiunea `skip` corespunde valorii ()

IMP $\subseteq$ OCaml

- IMP este un fragment al limbajului OCaml
- Un program IMP este o expresie OCaml în care
  - Locațiile sunt valori de tip referință (predefinite)
  - instrucțiunea `skip` corespunde valorii ()
- Date fiind
  - e o expresie IMP pentru care se poate determina tipul; și
  - o stare inițială a memoriei $s = \{l1 \mapsto n1, \ldots, lk \mapsto nk\}$ definită pentru toate locațiile lui e

e poate fi executată în starea s ca următorul program OCaml:

```
let skip = () and l1 = ref n1 and l2 = ref n2 and ... and lk=ref nk
 in e
   ; [("l1",!l1);("l2",!l2);...;("lk",!lk)]
```

Argumente împotriva sistemelor de tipuri

Argumente împotriva sistemelor de tipuri

- Sistemul de tipuri e prea restrictiv: programe „bune“ nu se compilează

$$\texttt{if}\ \mathit{false}\ \texttt{then}\ 1\ \texttt{else}\ \mathit{true}$$

Argumente împotriva sistemelor de tipuri

- Sistemul de tipuri e prea restrictiv: programe „bune“ nu se compilează

  `if` *false* `then 1 else` *true*

- E pierdere de vreme să tot urmărești tipurile și să le modifici (limbaje de scripting)

Argumente împotriva sistemelor de tipuri

- Sistemul de tipuri e prea restrictiv: programe „bune“ nu se compilează

  `if` *false* `then 1 else` *true*

- E pierdere de vreme să tot urmărești tipurile și să le modifici (limbaje de scripting)
- E prea mult de scris (e.g., tipuri STL in C++)
  Soluție: Detectarea tipurilor

Argumente împotriva sistemelor de tipuri

- Sistemul de tipuri e prea restrictiv: programe „bune“ nu se compilează

  `if` *false* `then` 1 `else` *true*

- E pierdere de vreme să tot urmărești tipurile și să le modifici (limbaje de scripting)
- E prea mult de scris (e.g., tipuri STL in C++)
  Soluție: Detectarea tipurilor
- Erorile de tipuri sunt greu/imposibil de citit
  Câteodată da — a se vedea erorile STL . . .

Argumente împotriva sistemelor de tipuri

- Sistemul de tipuri e prea restrictiv: programe „bune“ nu se compilează

  `if` *false* `then 1 else` *true*

- E pierdere de vreme să tot urmărești tipurile și să le modifici (limbaje de scripting)
- E prea mult de scris (e.g., tipuri STL in C++)
  Soluție: Detectarea tipurilor
- Erorile de tipuri sunt greu/imposibil de citit
  Câteodată da — a se vedea erorile STL . . .
- Tipurile nu mă lasă să scriu codul care îl vreau