# FUN-IMP

## Tipuri și recursie

Traian Florin Șerbănuță

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

2 decembrie 2014

# Reguli pentru tipuri în FUN-IMP

## Pregătiri

### Mediul de tipuri

Extindem mediul de tipuri de la o funcție de la locații la tipuri referință la o funcție care în plus dă și tipuri pentru variabile.
Formal $\Gamma = \Gamma_l \uplus \Gamma_v$ (reuniune disjunctă) unde

- $\Gamma_l : \mathbb{L} \xrightarrow{\circ} T_{loc}$ asociază tipuri referință la locații
- $\Gamma_v : \mathbb{X} \xrightarrow{\circ} T$ asociază tipuri variabilelor

Notăm cu $\Gamma[x \mapsto T]$ mediul de tipuri care e definit la fel ca $\Gamma$ peste tot, mai puțin în $x$ unde este $T$.

# Reguli pentru tipuri

## Funcții

$$(\textsc{tVar}) \quad \Gamma \vdash x : T \quad \textit{dacă } \Gamma(x) = T$$

$$(\textsc{tFun}) \quad \frac{\Gamma' \vdash e : T'}{\Gamma \vdash \texttt{fun}\,(x : T) \to e : T \to T'} \quad \textit{dacă } \Gamma' = \Gamma[x \mapsto \tau]$$

$$(\textsc{tApp}) \quad \frac{\Gamma \vdash e_1 : T \to T' \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1\ e_2 : T'}$$

# Exemplu

$$(\textsc{tApp})\ \frac{\vdash \texttt{fun}\,(x : \texttt{int}) \to x + 2 \ :\ \texttt{int} \to \texttt{int} \quad (\textsc{tInt})\ \dfrac{}{\vdash\ 2\ :\ \texttt{int}}\ \checkmark}{\vdash\ (\texttt{fun}\,(x : \texttt{int}) \to x + 2)\ 2\ :\texttt{int}}$$

# Exemplu

$$(\textsc{tApp})\ \frac{\vdash \texttt{fun}\,(x : \texttt{int}) \to x + 2 \ :\ \texttt{int} \to \texttt{int} \quad (\textsc{tInt})\ \dfrac{}{\vdash\ 2\ :\ \texttt{int}}\ \checkmark}{\vdash\ (\texttt{fun}\,(x : \texttt{int}) \to x + 2)\ 2\ :\texttt{int}}$$

$$(\textsc{tFun})\ \frac{(\textsc{t+})\ \dfrac{(\textsc{tVar})\ \dfrac{}{x : \texttt{int} \vdash\ x\ :\ \texttt{int}}\ \checkmark \qquad (\textsc{tInt})\ \dfrac{}{x : \texttt{int} \vdash\ 2\ :\ \texttt{int}}\ \checkmark}{x : \texttt{int} \vdash\ x + 2\ :\ \texttt{int}}}{\vdash\ \texttt{fun}\,(x : \texttt{int}) \to x + 2\ :\ \texttt{int} \to \texttt{int}}$$

# Proprietăți ale sistemului de tipuri

**Teoremă (Progres)**

*Dacă $e$ închisă și dacă $\Gamma \vdash e : T$ și $Dom\,\Gamma \subseteq Dom\,s$, atunci fie $e$ este valoare, fie există $e'$, $s'$ astfel încât $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$.*

Demonstrație: prin inducție structurală. □

**Teoremă (Conservarea tipului)**

*Dacă $e$ închisă și dacă $\Gamma \vdash e : T$, $Dom\,\Gamma \subseteq Dom\,s$ și $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$, atunci $\Gamma \vdash e' : T$, $e'$ închisă și $Dom\,\Gamma \subseteq Dom\,s'$.*

Demonstrație: prin inducție deductivă. □

**Lemma (Substituția conservă tipul)**

*Dacă $\Gamma \vdash e : T$ și $\Gamma[x \mapsto T] \vdash e' : T'$, atunci $\Gamma \vdash e'\ [e/x] : T'$.*

# Definiții locale de variabile

## Sintaxă

$$e ::= \ldots \mid \texttt{let}\ x : T = e\ \texttt{in}\ e$$

### Exemple

```
let n : int = 10 in
  sum := 0 ;
  i := n ;
  while ! i > 0 do
     sum  := !sum + !i;
     i := !i - 1;
  done
```

```
let x : int = 10 in
   x + (let x : int = 20 + x in
           x + x)
```

# Definire prin funcții și aplicații

## Semantica declarațiilor locale

$$\texttt{let}\ x : T = e_1\ \texttt{in}\ e_2 \stackrel{def}{=} (\texttt{fun}\ (x : T) \to e_2)\ e_1$$

## Tipul asociat (regulă derivată)

$$(\textsc{tLet})\quad \frac{\Gamma \vdash e_1 : T_1 \quad \Gamma' \vdash e_2 : T_2}{\Gamma \vdash \texttt{let}\ x : T = e_1\ \texttt{in}\ e_2 : T_2} \quad \textit{dacă}\ \Gamma' = \Gamma[x \mapsto T]$$

# Evaluarea (strictă) a declarațiilor locale

## Reguli derivate

$$(\text{LETS}) \quad \frac{\langle e_1, s\rangle \to \langle e_1', s'\rangle}{\langle \texttt{let } x : T = e_1 \texttt{ in } e_2, s\rangle \to \langle \texttt{let } x : T = e_1' \texttt{ in } e_2, s'\rangle}$$

$$(\text{LET}) \quad \langle \texttt{let } x : T = v \texttt{ in } e_2, s\rangle \to \langle e_2\ [v/x], s\rangle$$

# $\alpha$-echivalență penrtu declarații locale

- Operatorul $\texttt{let}\ x : T = e_1\ \texttt{in}\ e_2$ este un operator de legare (derivat)
  - Leagă variabila $x$ în termenul $e_2$ (dar nu și în $e_1$)
  - Ușor de observat din definiția lui `let` în funcție de $\lambda$

- $var(\texttt{let}\ x : T = e_1\ \texttt{in}\ e_2) = var(e_1) \cup (var(e_2) \setminus \{x\})$

- ($\alpha$LET) $\texttt{let}\ x : T = e_1\ \texttt{in}\ e_2 \equiv_\alpha \texttt{let}\ x' : T = e_1\ \texttt{in}\ e_2'$
  dacă $x' \notin var(e_2)$ și $e_2[x'/x] = e_2'$
  - $\texttt{let}\ x : \texttt{int} = 10\ \texttt{in}\ x + (\texttt{let}\ x : \texttt{int} = 20 + x\ \texttt{in}\ x + x) \equiv_\alpha$
    $\texttt{let}\ x : \texttt{int} = 10\ \texttt{in}\ x + (\texttt{let}\ y : \texttt{int} = 20 + x\ \texttt{in}\ y + y)$

# Definiții recursive

Întrebare: La ce se evaluează următorul program?

$$\texttt{let } f : \texttt{int} \to \texttt{int} = \texttt{fun } (n : \texttt{int}).\ \texttt{if } n \leq 0 \texttt{ then } 0 \texttt{ else } n + f(n + -1) \texttt{ in } f\ 10$$

# Definiții recursive ?

Întrebare: La ce se evaluează următorul program?

$$\texttt{let } f : \texttt{int} \to \texttt{int} = \texttt{fun } (n : \texttt{int}).\ \texttt{if } n \leq 0 \texttt{ then } 0 \texttt{ else } n + f(n + -1) \texttt{ in } f\ 10$$

Răspuns: La nimic

Deoarece programul nu este închis: $f$ apare liber în definiția lui $f$.
Avem nevoie de un mecanism separat pentru a putea defini funcții recursive.

# Limbajul $\lambda^+$-IMP

## Sintaxă

$$e ::= \ldots \mid \texttt{let rec}\ x : T = e\ \texttt{in}\ e$$

### Exemplu

```
let rec sum : int -> int =
  fun (n : int) ->
    if n <= 0
    then 0
    else n + sum (n + -1)
in sum 10
```

# Variabile libere, substituție și $\alpha$-echivalență

- $\texttt{let rec}\ x = e_1\ \texttt{in}\ e_2$ leagă variabila x atât în $e_1$ cât și în $e_2$
  $\texttt{let rec}\ f : \texttt{int} \to \texttt{int} = \texttt{fun}\ (n : \texttt{int}) \to \texttt{if}\ n \leq 0\ \texttt{then}\ 0\ \texttt{else}$
  $n + f(n + -1)\ \texttt{in}\ f\ 10$
- $var(\texttt{let rec}\ x : T = e_1\ \texttt{in}\ e_2) = (var(e_1) \cup var(e_2)) \setminus \{x\}$
- $$\frac{e_1[e/y] = e_1' \quad e_2[e/y] = e_2'}{(\texttt{let rec}\ x : T = e_1\ \texttt{in}\ e_2)[e/y] = \texttt{let rec}\ x : T = e_1'\ \texttt{in}\ e_2'} \quad \text{dacă } x \notin var(e) \cup \{y\}$$
- $\texttt{let rec}\ x = e_1\ \texttt{in}\ e_2 \equiv_\alpha \texttt{let rec}\ y = e_1'\ \texttt{in}\ e_2'$
  *dacă* $y \notin var(e_1) \cup var(e_2)$, $e_1[y/x] = e_1'$ și $e_2[y/x] = e_2'$

# Semantică `let rec`

## Regula de tipuri

$$(\textsc{tLetRec})\quad \frac{\Gamma' \vdash e_1 : T \quad \Gamma' \vdash e_2 : T_2}{\Gamma \vdash \texttt{let rec } x : T = e_1 \texttt{ in } e_2 : T_2} \quad \textit{dacă } \Gamma' = \Gamma[x \mapsto T]$$

## Regula semantică

$$(\textsc{letrec})\quad \langle \texttt{let rec } x : T = e_1 \texttt{ in } e_2, s\rangle \to \langle e_2[\texttt{let rec } x : T = e_1 \texttt{ in } e_1/x], s\rangle$$