Interpretação e Compilação de Linguagens de Programação

Licenciatura em Engenharia Informática
Departamento de Informática
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa

Luís Caires

Sistema de Tipos para a Linguagem CORE

- **Expressões**

num: **integer** → EXP

id: **string** → EXP

add: EXP × EXP → EXP

and: EXP × EXP → EXP

eq: EXP × EXP → EXP

var: EXP → EXP

deref: EXP → EXP

do: COM × EXP → EXP

let: **string** × EXP × EXP → EXP

fun: **string** × EXP → EXP

call: EXP × EXP → EXP

assign: EXP × EXP → EXP

if: EXP × EXP × EXP → EXP

while: EXP × EXP → EXP

Um programa em CORE

```
let
  Acum = (fun  r,v => r := !r + v end)
  inc = (fun j => Acum (j, 1) end)
  looper = (fun x,f =>
              let
                  i = var(0)
                  s = var(0)
              in
                    while (!i < x) do
                       Acum(s, f(!i) ) ;
                       inc(i)
                    end;
                 !s end
              end end
  google = (fun x => x * x end)
in  print (looper(10,google))
end
```

Sistema de Tipos para CORE

- Tipos ($\mathcal{T}$)

  int

  bool

  Ref($\mathcal{T}$) : Tipo das referências que guardam valores de tipo $\mathcal{T}$.

  Fun($\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$)$\mathcal{T}$: Tipo das funções que recebem *n* argumentos, de tipos resp. $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, e devolvem um valor de tipo $\mathcal{T}$.

- Exemplos de asserções de tipificação válidas:

  x: Ref(int) $\vdash$ **while** (!x<0) **do** x:=!x+1 **end** ok

  y: int $\vdash$ (**fun** x: int -> $y + x$) **: Fun(int)int**

A Linguagem CORE

- Para se poder definir um algoritmo de tipificação simples, e garantir a propriedade de unicidade do tipo, extendemos a linguagem CORE etiquetando os parâmetros das funções com o seu tipo.
- Etiquetam-se também as declarações de modo a facilitar a tipificação das chamadas recursivas.

let: **string** × TYPE × EXP × EXP → EXP

fun: **string** × TYPE × EXP → EXP

Exemplos:

fun x:int → $y + x$ **end**

let f:Fun(int)int = **fun** x:int → $f(x\text{-}1)$ **in** ... **end**

Sistema de Tipos para Core

- **Tipificação de expressões (cálculo lambda)**

R1 (axioma)

$$Env,\ x{:}\mathcal{T},\ Env' \vdash \mathbf{id}(x) : \mathcal{T}$$

R2 (função)

$$\frac{Env,\ x{:}\ \mathcal{T} \vdash \mathrm{E} : \mathcal{U}}{Env \vdash \mathbf{fun}(x, \mathcal{T}, \mathrm{E}) : \mathbf{Fun}(\mathcal{T})\ \mathcal{U}}$$

R3 (chamada)

$$\frac{Env \vdash \mathrm{M} : \mathbf{Fun}(\mathcal{T})\ \mathcal{U} \qquad Env \vdash \mathrm{N} : \mathcal{T}}{Env \vdash \mathbf{call}(\mathrm{M},\mathrm{N}) : \mathcal{U}}$$

Sistema de Tipos para Core

- Tipificação de expressões (Declarações)

R4 (decl)

$$\frac{Env,\ x\colon \mathcal{T} \vdash \mathrm{M} : \mathcal{T} \qquad Env,\ x\colon \mathcal{T} \vdash \mathrm{N} : \mathcal{U}}{Env \vdash \textbf{let}\ x\colon \mathcal{T} = \mathrm{M}\ \textbf{in}\ \mathrm{N}\ \textbf{end} : \mathcal{U}}$$

- Note-se que para permitir declarações recursivas, é necessário usar o tipo do identificador declarado na tipificação da sua expressão de definição.
- Se M não denotar um valor funcional, poderíamos omitir a declaração do seu tipo (porquê?).

```
let
  f : Fun(int)int = fun x:int → f(x-1) end
  y: int = x+3
in ... end
```

Sistema de Tipos para Core

- **Tipificação de expressões (inteiros)**

R5 (número)

$$Env \vdash \mathbf{num}(\mathrm{x}) : \mathbf{int}$$

R6 (add)

$$\frac{Env \vdash \mathrm{N} : \mathbf{int} \qquad Env \vdash \mathrm{M} : \mathbf{int}}{Env \vdash \mathbf{add}(\mathrm{M}, \mathrm{N}) : \mathbf{int}}$$

R7 (equal)

$$\frac{Env \vdash \mathrm{N} : \mathbf{int} \qquad Env \vdash \mathrm{M} : \mathbf{int}}{Env \vdash \mathbf{eq}(\mathrm{M}, \mathrm{N}) : \mathbf{bool}}$$

Sistema de Tipos para Core

- Tipificação de expressões (booleanos)

R6 (true) $Env \vdash \textbf{true} : \textbf{bool}$ $\quad$ $Env \vdash \textbf{false} : \textbf{bool}$ **R7 (false)**

R8 (and)

$$\frac{Env \vdash \mathrm{N} : \textbf{bool} \qquad Env \vdash \mathrm{M} : \textbf{bool}}{Env \vdash \textbf{and}(\mathrm{M}, \mathrm{N}) : \textbf{bool}}$$

R9 (equal)

$$\frac{Env \vdash \mathrm{N} : \textbf{bool} \qquad Env \vdash \mathrm{M} : \textbf{bool}}{Env \vdash \textbf{eq}(\mathrm{M}, \mathrm{N}) : \textbf{bool}}$$

Sistema de Tipos para Core

- **Tipificação de expressões (referências)**

R10 (var)

$$\frac{Env \vdash \mathrm{E} : \mathcal{T}}{Env \vdash \textbf{new}\ \mathrm{E} : \textbf{Ref}(\mathcal{T})}$$

R11 (deref)

$$\frac{Env \vdash \mathrm{E} : \textbf{Ref}(\mathcal{T})}{Env \vdash \textbf{!}\,\mathrm{E} : \mathcal{T}}$$

Sistema de Tipos para Core

- Tipificação de expressões (afetação, condicionais, ciclo)

$$\frac{Env \vdash \mathrm{N}:\mathrm{Ref}(\mathcal{T}) \quad Env \vdash \mathrm{E}:\mathcal{T}}{Env \vdash \mathrm{N} := \mathrm{E} : \mathcal{T}}$$ **R13 (assign)**

$$\frac{Env \vdash \mathrm{E}:\mathrm{bool} \quad Env \vdash \mathrm{C}:\mathcal{T} \quad Env \vdash \mathrm{C}':\mathcal{T}}{Env \vdash \textbf{if } \mathrm{E} \textbf{ then } \mathrm{C} \textbf{ else } \mathrm{C}' \textbf{ end} : \mathcal{T}}$$ **R14 (if)**

$$\frac{Env \vdash \mathrm{E}:\mathrm{bool} \quad Env \vdash \mathrm{C}:\mathcal{T}}{Env \vdash \textbf{while } \mathrm{E} \textbf{ do } \mathrm{C} \textbf{ end} : \mathrm{bool}}$$ **R15 (while)**

- **Leituras adicionais**

https://en.wikipedia.org/wiki/Type_system

https://en.wikipedia.org/wiki/Type_rule

Wikipedia reasonable reading list

https://dl.acm.org/citation.cfm?doid=234313.234418

Type systems, Luca Cardelli
Systems Research Center, Digital Equipment Corporation