Llenguatges Funcionals: Introducció

Albert Rubio

Llenguatges de Programació, FIB, UPC

Continguts

Continguts

Presentació

Aplicacions

- Microsoft. SLAM (OCaml). F# a .NET o Visual Studio
- Jane Street Capital (Wall Street), OCaml.
- Credit Siusse, Haskell, F#.
- Barclays, Haskell.
- EDF Trading (energia), Scala.
- Ericsson, Erlang, Haskell; Nokia també
- Facebook, Google, Haskell
- Twitter, Scala
- Conf.: Commercial Users of Functional Programming

Presentació

Aplicacions

- Derivative pricing.
- Data analysis (per exemple, fulls de càlcul).
- Hardware description languages.
- Geospatial search engine (geographical information).
- Desenvolupament de Software crític.
- ...

Presentació

Objectius

- Conèixer un llenguatge funcional pur
- Conèixer els fonaments del model de càlcul
- Sistema de tipus potent, ordre superior.
- Lambda-expressions. Funcions anònimes.
- Pattern matching, lazy evaluation
- Base per altres Llenguatges de Programació Funcionals: OCaml, Erlang, F#, Scala, ...
- Base per entendre aquestes construccions: C++, Java, Python, Ruby, ...

Continguts

Fonaments: λ-càlcul

- Model de computació funcional.
- Alonzo Church, 1932
- Totes les funcions recursives poden ser representades en λ-càlcul (Kleene,Rosser 1936)
- El λ-càlcul pot expressar exactament les funcions computables per una Màquina de Turing (Turing 1937).
- Abstracció i aplicació (variables lligades i substitució)
- Origen del llenguatges funcionals

Fonaments: λ-càlcul

Construcció de λ-termes. Dues "operacions":

- Abstracció: $\lambda x.u$ on u és un λ-terme.
- Aplicació: $(u\ \ v)$ on u i v són λ-termes.
  Associa a l'esquerra: $(((u\ \ v_1)\ \ v_2)\ \ \ldots\ \ v_n) = (u\ \ v_1\ \ v_2\ \ \ldots\ \ v_n)$

Intuïció: funcions matemàtiques.
$f(x, y, z) = x^2 + 2y + z + 4$
és
$f = \lambda x.\lambda y.\lambda z.x^2 + 2y + z + 4$

i
$f(6, 3, 12)$ és $(f\ \ 6\ \ 3\ \ 12)$, o millor $(\ (\ (f\ \ 6)\ 3)\ 12)$

Fonaments: λ-càlcul

Computació. Una sola regla:

- β-reducció: $(\lambda x.u\ \ v) \rightarrow_\beta u[x := v]$

Intuïció:
$(\lambda x.\lambda y.\lambda z.x^2 + 2y + z + 4\ \ 6\ \ 3\ \ 12) \rightarrow_\beta^3$
$x^2 + 2y + z + 4[x := 6][y := 3][z := 12] =$
$6^2 + 2 \cdot 3 + 12 + 4 = 58$

Variables

- Lligades. Variables que apareixen en una abstracció.
  En $\lambda x.(y\ \ \lambda z.(x\ \ z))$, x i z són lligades
- Lliures. Les demés (a l'exemple y). Es poden substituir!

α-conversió: $\lambda x.u =_\alpha \lambda y.u[x := y]$ si y és nova

Fonaments: λ-càlcul

Exemple: codificació dels nombres naturals

- 0: $c_0 = \lambda x.\lambda y.x = \lambda x\ y.y$
- n: $c_n = \lambda x.\lambda y.(\underbrace{x...(x}_{n}\ y)) = \lambda x\ y.x^n(y)$
- increment: $A_s = \lambda z\ x\ y.(x\ (z\ x\ y))$
- suma: $A_+ = \lambda p\ q\ x\ y.(p\ x\ (q\ x\ y))$
- producte: $A_* = \lambda p\ q\ x\ y.(p\ (q\ x)\ y)$

- $(A_s\ c_n) \rightarrow_\beta c_{n+1}$
- $(A_+\ c_n\ c_m) \rightarrow_\beta c_{n+m}$
- $(A_*\ c_n\ c_m) \rightarrow_\beta c_{n\cdot m}$

Fonaments: λ-càlcul

Propietats de la β-reducció:

- β-reducció és confluent.

  Si $t \to_\beta \ldots \to_\beta t_1$ i $t \to_\beta \ldots \to_\beta t_2$ llavors
  $t_1 \to_\beta \ldots \to_\beta t_3$ i $t_2 \to_\beta \ldots \to_\beta t_3$
- estratègia normalitzant de la β-reducció:

  left-most outer-most.

  Redueix la λ sintàcticament més a l'esquerra.

  Si existeix un terme que no es pot reescriure més, aquesta estratègia el troba!

Fonaments: λ-càlcul amb tipus

Només considerarem tipus simples

- a la Curry
  $\lambda x.x : \alpha \to \alpha$
- a la Church
  $\lambda x : \alpha.x : \alpha \to \alpha$

Declaracions

- A la Curry el tipus dels paràmetres es dedueixen del context.
- Cal declarar el tipus de las variables lliures.

Fonaments: λ-càlcul amb tipus

Exemple: Tipus simples: només tipus bàsics i funcionals.

Sigui Γ un context (declaració) de la forma $\{x : \alpha, y : \beta\}$ un λ-terme t és tipable en Γ si

$$\Gamma \vdash t : \sigma$$

es pot deduir de les següents regles (a la Curry)

Abstracció:

$$\frac{\Gamma \cdot \{x : \sigma\} \vdash t : \tau}{\Gamma \vdash (\lambda x.t) : \sigma \to \tau}$$

Variable:

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

Aplicació:

$$\frac{\Gamma \vdash s : \sigma \to \tau \quad \Gamma \vdash t : \sigma}{\Gamma \vdash (s\ \ t) : \tau}$$

Fonaments: λ-càlcul amb tipus

Exemple: Tipus simples: només tipus bàsics i funcionals.

Sigui Γ un context (declaració) de la forma $\{x : \alpha, y : \beta\}$ un λ-terme t és tipable en Γ si

$$\Gamma \vdash t : \sigma$$

es pot deduir de les següents regles (a la Church)

Abstracció:

$$\frac{\Gamma \cdot \{x : \sigma\} \vdash t : \tau}{\Gamma \vdash (\lambda x{:}\sigma.t) : \sigma \to \tau}$$

Variable:

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}$$

Aplicació:

$$\frac{\Gamma \vdash s : \sigma \to \tau \quad \Gamma \vdash t : \sigma}{\Gamma \vdash (s\ \ t) : \tau}$$

Fonaments: λ-càlcul amb tipus

$\lambda x.(x\ \ x)$ no és tipable.

Exemple de derivació de tipus

$$\dfrac{\dfrac{\dfrac{\{f:\beta\to\alpha, x:\beta\}\vdash f:\beta\to\alpha \qquad \{f:\beta\to\alpha, x:\beta\}\vdash x:\beta}{\{f:\beta\to\alpha, x:\beta\}\vdash (f\ \ x):\alpha}}{\{f:\beta\to\alpha\}\vdash \lambda x.(f\ \ x):\beta\to\alpha}}{\emptyset\vdash \lambda f\lambda x.(f\ \ x):(\beta\to\alpha)\to\beta\to\alpha}$$

- Si t es tipable amb aquest sistema la β-reducció acaba.
- El λ-càlcul amb tipus simples no és Turing complet
- Cal afegir polimorfisme i definició de noves funcions