Introdução à Programação Funcional

PPGIA - PUCPR
Prof. Fabrício Enembreck

1

Conteúdo

- Introdução ao Cálculo Lambda e à Programação Funcional
- Introdução ao LISP e ao CLisp
- Funções Lambda e binding
- Funções de sequências
- Listas e funções recursivas
- Grafos em LISP

2

Bibliografia

- Graham, Paul, Ansi Common Lisp, 1996.
- Steele, Guy, Common Lisp: the langage, 1990.
- Reade, Chris, Elements of functional programming, 1989.
- Bird, Richard, Introduction to funcional programming, 1988.
- http://www.paulgraham.com/lib/paulgraham/onlisp.pdf
- http://www.paulgraham.com
- http://lib1.store.vip.sc5.yahoo.com/lib/paulgraham/acl2.txt
- John McCarthy: *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I*

Utilizado nesta apresentação

- http://www.ufasoft.com/lisp/

Motivação

- A programação funcional introduziu muitos recursos novos de programação:
  - **Condicionais**: *if-then-else* foi inventado por John McCarthy
  - **Função**: Uma função é uma estrutura de dados (Lista)
  - **Recursão**: Existia apenas em definições matemáticas. LISP foi a primeira linguagem a suportar recursividade
  - **Ponteiros**: Toda variável é um ponteiro
  - **Garbage-Collection**

Introdução ao Cálculo Lambda

- O que é?
  - A notação Lambda é um formalismo matemático que inspirou algumas das linguagens de programação funcionais (LISP por exemplo).
- Conceitos Básicos
  - ***Abstração de função***: "λx.E" e denota uma função cujo parâmetro formal (variável ligada) é x e cujo corpo é a expressão E.
  - ***Aplicação da função***: Uma aplicação ou chamada de função é escrita na forma "E1 E2". A sub-expressão E1 deve avaliar uma determinada função, tendo como parâmetro atual E2.

5

Exemplo de Abstrações de Função

- λx.x Função identidade;
- λn.subtract n 1 Função predecessor;
- λf.f 0 O argumento é uma função f sendo o resultado a aplicação deste com 0;
- λf . λx . f(f x) Esta função recebe f e x como parâmetro; o resultado é uma função que aplica f duas vezes a x;
- λp . λx . not(p x) O argumento é uma função booleana p e um valor x. O resultado é a negação do resultado de p sobre x;

6

Exemplo de Aplicações de Função

- suc n — Sucessor de n;
- zero (pred n) — Aplica a função zero sobre o resultado de (pred n);
- add 1 n — Soma 1 e n;
- (λx.x) y — Aplica a função identidade a y;

7

Exemplo de Avaliação de Função

- zero (pred 4)
  - ⇒ zero 3
  - ⇒ false zero retorna true caso o argumento seja 0 e false caso contrário
- (λn . add n n) 3
  - ⇒ add 3 3
  - ⇒ 6

8

Sintaxe do Cálculo Lambda

Expression ::= S-Expression
| “λ” Identifier “.” Expression

S-Expression ::= P-Expression
| S-Expression P-Expression

P-Expression ::= Literal
| Identifier
| (Expression)

S-Expression: **Symbolic Expression** P-Expression: **Propositional Expression**

Semântica do Cálculo Lambda

- ***Não tipado***: função identidade pode ser aplicada sobre qualquer valor
- ***Variável ligada***: Identificador que sucede o símbolo “λ”
- ***Ocorrência aplicada***: Qualquer outra ocorrência de um identificador “ligado”
- Exemplos

  λn. add m n (“add” e “m” estão livres)
  λf . λx . f (f (x)) (“f” e “x” são parâmetros)

Exemplos de Avaliação de Funções

(λp . λx . not (p x)) zero 2

⇒ λx . not (zero x) 2

⇒ not (zero 2)

⇒ not false

⇒ true

(λf . λx . f (f x)) suc 8

⇒ λx . suc (suc x) 8

⇒ suc (suc 8)

⇒ suc 9

⇒ 10

11

Ordem de Avaliação de Funções

Ordem Normal

(λn . multiply n n) (add 2 3)

⇒ multiply (add 2 3) (add 2 3)

⇒ multiply 5 (add 2 3)

⇒ multiply 5 5

⇒ 25

Ordem Primária

(λn . multiply n n) (add 2 3)

⇒ (λn . multiply n n) 5

⇒ multiply 5 5

⇒ 25

12

Porque a ordem é importante?

Considere a função: (λn . 7) (divide 1 0)

Ordem Normal

(λn . 7) (divide 1 0)

⇒ 7

Ordem Primária

(λn . 7) (divide 1 0)

⇒ ⊥ (Indeterminação)

13

Funções Estritas e não Estritas

Estrita se f ⊥⇒ ⊥

Exemplo : succ

Não-estrita se f ⊥⇒ x e x ≠ ⊥ (em alguma ordem de avaliação)

Exemplo: (λn . 7)

Funções estritas, ou parciais, aplicam-se apenas em parte de um domínio, pois combinações de valores de parâmetros podem levar a função a não terminar a computação.

14

Introdução à Programação Funcional

- Em 1956-1958 John McCarthy criou LISP*
- Fortran era a linguagem mais conhecida
- Limitações de Fortran:
  - Seqüencial (sem subrotinas ou comandos aninhados)
  - Condicionais “if goto”
  - Sem recursividade
- LISP introduziu os conceitos de base para muitas das linguagens de hoje

*** LIS**t **P**rocessing Language

15

Elementos da Programação Funcional

- ***Funções Parciais*** (já visto no cálculo lambda)
- ***Expressões Proposicionais***: uma expressão cujos valores possíveis são T (true) ou F (false)
- ***Expressões condicionais***

$$(p_1 \rightarrow e_1, p_2 \rightarrow e_2, ..., p_n \rightarrow e_n)$$

Lê-se: se p1 então e_1 senão se p_2 *então* e_2...

16

Elementos da Programação Funcional (cont.)

- Exemplos de Expressões condicionais

$(1 < 2 \rightarrow 4, 1 > 2 \rightarrow 3) = 4$

$(2 < 1 \rightarrow 4, 2 > 2 \rightarrow 3, 2 > 1 \rightarrow 2) = 2$

$(2 < 1 \rightarrow 3, T \rightarrow 0/0)$ indefinida

$(2 < 1 \rightarrow 3, 4 < 1 \rightarrow 4)$ indefinida

*** LIS**t **P**rocessing Language

17

Elementos da Programação Funcional (cont.)

- ***Definições recursivas de função***

$$n! = (n = 0 \rightarrow 1, T \rightarrow n * (n -1)!)$$

Avaliação de 2!

$2! = (2 = 0 \rightarrow 1, T \rightarrow 2 * (2 - 1)!)$
$= 2 * 1!$
$= 2 * (1 = 0 \rightarrow 1, T \rightarrow 1 * (1 – 1)!)$
$= 2 * 1 * 0!$
$= 2 * 1 * (0 = 0 \rightarrow 1, T \rightarrow 0 * (0 - 1)!)$
$= 2 * 1 * 1$
$= 2$

18

Elementos da Programação Funcional (cont.)

- ***Expressões para funções recursivas***
  - Lambda Cálculo não permite chamar uma função dentro da definição da mesma função
  - Ex:

sqrt(a,x,e) = (|x^2-a| < e → x, T → sqrt(a, ½ * (x+a/x), e))

Pode ser definida como:

sqrt = (λ.a.x.e).(|x^2-a| < e → x, T → sqrt(a, ½ * (x+a/x), e))

mas o que é "sqrt"?

19

Elementos da Programação Funcional (cont.)

- Notação para definição de funções recursivas
- Ex.: label(l, Expression)

label(sqrt, (λ.a.x.e).(|x^2-a| < e → x, T → sqrt(a, ½ * (x+a/x), e))))

20

Elementos da Programação Funcional (cont.)

- Expressões simbólicas (S-Expressions)
  - “.”, “(”, “)”
  - Símbolos atômicos: Letras maiúsculas e/ou dígitos
    - Ex.: CASA, 3, APPLE, APPLE-3

Logo, uma S-Expression pode ser:

- Um símbolo atômico
- Se e_1 e e_2 são S-Expressions, então $(e_1 . e_2)$ são S-Expressions

- Exemplos de S-Expressions:
  - AB
  - (A . B)
  - ((AB . C) . D)

21

Elementos da Programação Funcional (cont.)

- Listas
  - Uma lista (m_1 m_2 ... m_n) de elementos corresponde à seguinte expressão simbólica:
  - (m_1. (m_2 . (... (m_n . NIL) ...)))

Onde:

- (m) equivale a (m . NIL)
- (m1 m2 ... m_n) equivale a (m1. (m2 . (... (m_n . NIL) ...)))
- (m1 m2 ... m_n . x) equivale a (m1. (m2 . (... (m_n . x) ...)))

22

Elementos da Programação Funcional (cont.)

- Abreviações de Listas
  - ((AB . (C . NIL)) . (D . NIL)) equivale a:

    ((AB C) D)
  - ((A . (B . NIL)) . (C . (D . E))) equivale a:

    ((A . B) C D . E)

23

Elementos da Programação Funcional (cont.)

- Funções simbólicas elementares e predicados
  - (atom x): true se x é átomo
  - (eq x y): x e y são átomos e idênticos
  - (car l): primeiro elemento da lista l
  - (cdr l): corpo da lista l
  - (cons x y): (x . y)
  - (equal x y): x e y são expressões simbólicas idênticas

24

Elementos da Programação Funcional (cont.)

- A função universal *apply*
  - (apply f args) é equivalente a:

(eval (cons f x))

EX.: (apply (quote +) (quote (1 2 3))) = 6

25

Elementos da Programação Funcional (cont.)

- Representação de funções simbólicas ε com expressões simbólicas ε*
  1. Se ε é uma S-Function, ε* é (QUOTE ε)
  2. Toda string s em ε é colocada em maiúsculo "s*" em ε*
  3. ($p_1 \rightarrow e_1$, $p_2 \rightarrow e_2$, ..., $p_n \rightarrow e_n$) é transformado em (COND (p_1* e_1*) (p_2* e_2*) ... (p_n* e_n*))
  4. $\lambda x_1. \lambda x_2. ... \lambda x_n.\varepsilon$ é transformado em
     (LAMBDA (x_1* x_2* ... x_n*) ε*)
  5. label(l, param, ε) é transformado em
     (defun l* param* ε*)

26

Exemplos de Expressões Simbólicas em LISP

?> (lambda (x) x) ;;; função identidade

?> (lambda (x) (* x x)) ;;; quadrado

?> (lambda (f) (funcall f 0)) ;; uma função f sendo o resultado a aplicação deste com 0;

?> (lambda (p x) (not (funcall p x)))

?> (cond ((> 0 1) (print “nao”))
((> 1 0) (print “sim”)))

27

Exercícios

1) Da lista (3 4 5) diga quais são os seus elementos e de que tipo são (átomo/lista). Qual é o segundo elemento?

2) Qual é o resultado da avaliação da expressão:
(+ 3 (* 7 6) 5 (/ 4 2))? Diga quais são os seus elementos e de que tipo são (átomo/lista).

3) Defina uma função **lambda** que implemente a seguinte função f(x)=x-1. Dê um exemplo de utilização.

4) Qual a cabeça da lista (3 4 5)? Qual a expressão simbólica em LISP que nos permite obtê-la?

28

Exercícios

5) Qual é o corpo da lista ((3) (4 5))? Qual a expressão simbólica em LISP que nos permite obtê-la?

6) Implemente uma função **lambda** que obtém a cabeça da lista passada como argumento. Dê um exemplo de utilização.

7) Qual a cauda da lista (3 4 5)? E da lista ((3) 4 5)? E da lista ((3) 4 5)? E da lista nil? E da lista contendo nil?

8) Implemente uma função **lambda** que obtém a cauda da lista passada como argumento. Dê um exemplo de utilização.

9) Qual a cabeça da cauda das seguintes listas: (), (3), (3 4), ((3) 4), ((3) (4 5)), ((3) (4 5) (7))

29