Funções Recursivas

Prof.: Edson Holanda

edsonholanda@gmail.com

Teoria da computação - Diverio e Menezes

Tipos de Formalismos

- **Operacional**

  Define-se uma máquina abstrata, baseada em estados, em instruções primitivas e na especificação de como cada instrução modifica cada estado.

Exemplos: formalismos Máquina Norma e Máquina de Turing

Tipos de Formalismos

- **Axiomático.**

  Associam-se regras às componentes da linguagem. As regras permitem afirmar o que será verdadeiro após a ocorrência de cada cláusula, considerando o que era verdadeiro antes da ocorrência.

Exemplo: Gramáticas e lógicas.

Tipos de Formalismos

- **Denotacional ou Funcional**

  Funções construídas a partir de funções elementares de modo que o algoritmo denotado pela função pode ser determinado em termos de suas funções componentes.

Exemplo: Funções Recursivas Parciais (Kleene, 1936).

Funções Recursivas

- Funções naturais simples:
  - constante zero;
  - sucessor;
  - projeção;
- Juntamente com recursão e minimização.

Linguagem Lambda

- É apresentada a notação conhecida como **Linguagem Lambda** (λ-Linguagem, introduzida por Alonzo Church em 1941) no Cálculo Lambda (λ-Cálculo), cujo principal objetivo é evitar ambigüidades de notação.

Funções e Funcionais

- Uma função parcial é uma relação $f \subseteq A \times B$ onde cada elemento do domínio (conjunto A) está relacionado com, no máximo, um elemento do contra-domínio (conjunto B).
- $f \subseteq A \times B$ é denotada por $f: A \rightarrow B$;
  - o *tipo* de f é $A \rightarrow B$;
  - $(a, b) \in f$ é denotado por $f(a) = b$.

Exemplo: Funções e seus tipos

$f(x) = x^3 + 4$ $\quad$ $f: \mathbb{N} \rightarrow \mathbb{N}$

$g(x, y) = (x^2, y - x)$ $\quad$ $g: \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N} \times \mathbb{N}$ **ou**

$g: \mathbb{N}^2 \rightarrow \mathbb{N}^2$

$h(f, x) = f(\, f(x)\,)$ $\quad$ $h: (\,(\mathbb{N} \rightarrow \mathbb{N}) \times \mathbb{N}) \rightarrow \mathbb{N}$

Definição: (Funcional)

- Funcional é uma função que possui uma ou mais funções como argumentos.
- Portanto, no exemplo anterior, a função h é um funcional.

Observação:

- *funções com mais de um argumento podem ser vistas como funcionais com um argumento.*
- *funcionais constituem um dispositivo útil na simplificação de notação e facilita a manipulação de expressões.*

Exemplo:

- Seja **f:** $\mathbb{N} \rightarrow \mathbb{N}$

- h(f, x) = f(f(x))

- h'(f) (x) = f(f(x))

- h: $(N \rightarrow N) \times N \rightarrow N$
- h': $(N \rightarrow N) \rightarrow (N \rightarrow N)$

Introdução

- Considere novamente a função f: $N \rightarrow N$ tal que: $f(x) = x^3 + 4$
- f(x) é o resultado da aplicação de f ao parâmetro x;
- f(x) não é uma função, mas uma equação (polinômio).

Construções da Linguagem Lambda

- ***a) Abstração Lambda*. Permite abstrair a definição da função.**
- *Por exemplo, a função f acima é denotada pelo seguinte termo Lambda:*
- $\lambda x.x^3 + 4$ função tal que, para um argumento arbitrário x, resulta em $x^3 + 4$

Construções da Linguagem Lambda

- *b) Aplicação Lambda*

- Determina o valor da função aplicada a um dado parâmetro.

- $(\lambda x.x^3 + 4)(2)$ aplicação da função $\lambda x.x^3 + 4$ ao valor 2

Construções da Linguagem Lambda

- Termos da Linguagem Lambda serão denotados por letras maiúsculas M, N, P,…
- $(\lambda x.x^3 + 4)(2)$
- M denota $x^3 + 4$ $(\lambda x.M)(P)$
- N denota $\lambda x.x^3 + 4$
- P denota 2 $(N)(P)$

Construções da Linguagem Lambda

- *Regra de Redução Beta* (*Regra de Redução β*)

- $(\lambda x.M)(P) = [P/x]M$

substituição de x por P em M:

- $(\lambda x.x^3 + 4)(2)$

$= [2/x]\ x^3 + 4$

$= 2^3 + 4$

$= 12$

Construções da Linguagem Lambda

- Linguagem Lambda é constituída de termos lambda.
- **Definição: (Termo Lambda)**
- Sejam X e C conjuntos contáveis de *variáveis* e de *constantes*
- Então um *Termo Lambda*, *Expressão Lambda* ou *Palavra Lambda* sobre X e C é indutivamente definido por:

Construções da Linguagem Lambda

- a) Toda variável $x \in X$ é um termo lambda;
- b) Toda constante $c \in C$ é um termo lambda;
- c) Se M e N são termos lambda, e x é uma variável, então:
- 1) $(M\ N)$ é um termo lambda;
- 2) $(\lambda x.M)$ é um termo lambda.

Simplificações de notação adotadas:

- a) Parênteses podem ser eliminados, respeitando a associatividade à esquerda. Sejam M, N e P termos. M N P é uma simplificação de ((M N) P);
- b) Parênteses podem ser eliminados, respeitando o escopo de uma variável em uma abstração. Sejam M, N e P termos, e x a variável. $\lambda x.M\ N\ P$ é uma simplificação de $(\lambda x.(M\ N\ P))$;

Simplificações de notação adotadas:

- c) Os seguintes termos lambda denotam a função adição equivalentemente:

- $\lambda(x, y).x + y$ e $\lambda(r, s).r + s$

Definição: (Linguagem Lambda)

- Sejam X e C conjuntos contáveis de variáveis e de constantes, respectivamente. Uma *Linguagem Lambda* L sobre X e C é o conjunto de todos os termos lambda sobre esses conjuntos.

Observação:

- A Linguagem Lambda inspirou a linguagem de programação *LISP*.
- LISP é uma linguagem sem tipos, onde programas e dados não são distinguíveis.

Observação:

- Até hoje são discutidas as vantagens e desvantagens das *linguagens* *tipadas* e das *linguagens* *não-tipadas*, ou seja se:
  - Tipos devem constituir um esquema básico do conhecimento;
  - Entidades devem ser organizadas em subconjuntos com propriedades uniformes a partir de um universo não-tipado.

Observação:

- Após a introdução da linguagem Algol, onde variáveis são associadas a tipos quando de suas declarações, permitindo verificações sintáticas e semânticas de suas instâncias no programa em tempo de compilação, foi verificado que tipos têm sido considerados uma facilidade essencial para o desenvolvimento de programas.

Observação:

- A associação de tipo a um termo lambda é como em uma definição tradicional de função.

- Exemplo: Tipos das funções parciais, onde as variáveis **x** e **y** têm seus valores em **N**:
- $f(x) = x^3 + 4$
- $f = \lambda x.x^3 + 4: N \to N$

Definição: (Var. Ligada e Var. Livre)

- Uma variável em um termo lambda é denominada:
- a) *Variável Ligada*, se está dentro do escopo de uma abstração lambda;
- b) *Variável Livre*, caso contrário.

Exemplo:

- a)No termo $\lambda x.x^k$, as variáveis x e k são ditas ligada e livre, respectivamente.
- b) No termo $\lambda k.\lambda x.x^k$, as variáveis x e k são ditas ligadas.

Definição: (Substituição de Var. Livre)

- Sejam x uma variável e M, P termos lambda.
- A *Substituição de uma Variável Livre* x por P em M, denotada por:

$$[P/x]M$$

é simplesmente a substituição de todas as ocorrências de x em M pelo termo P.

Exemplo:

- A substituição da variável livre x por 5 em $\lambda k.x^k$ é

$$[5/x]\ \lambda k.x^k = \lambda k.5^k$$

Definição: Regra de Redução Beta

- Sejam x uma variável e, M e P termos lambda. A *Regra de Redução Beta* de um termo $(\lambda x.M)(P)$ é dada pela substituição de x por P em M, ou seja:

$$[P/x]M$$

- Note-se que x é variável ligada em $\lambda x.M$, mas é livre em M, o que garante a coerência da definição acima.

Semântica de um Termo Lambda

- A semântica de um Termo Lambda dado por uma função aplicada a um parâmetro é dada pelas aplicações sucessivas possíveis da regra de redução beta.

Exemplo:

- A semântica do termo (λk.(λx.xk)(5))(2) é dada pela sucessiva aplicação da regra de redução beta como segue:

  ($\lambda k.(\lambda x.x^k)(5))(2)$ aplicação da rb em k

  $= [2/k](\lambda x.x^k)(5)$ substituição da var. livre k

  $= (\lambda x.x^2)(5)$ aplicação da rb em x

  $= [5/x](x^2)$ substituição da var. livre x

  $= 5^2$

  $= 25$

Funções Recursivas de Kleene

- As funções recursivas parciais propostas por Kleene são funções construídas sobre funções básicas usando três tipos de construções:
  - Composição;
  - Recursão;
  - Minimização.

Definição: Composição de Funções

- Sejam g, f_1, f_2, …, f_k funções parciais tais que:

- $g = \lambda(y_1, y_2, \ldots, y_k).g(y_1, y_2, \ldots, y_k): N^k \to N$
- $f_i = \lambda(x_1, x_2, \ldots, x_n).f_i(x_1, x_2, \ldots, x_n): N^n \to N$, para $i \in \{ 1, 2, \ldots, k \}$

Definição: Composição de Funções

- A função parcial h tal que:
- $h = \lambda(x_1, x_2, \ldots, x_n).h(x_1, x_2, \ldots, x_n): N^n \rightarrow N$

é a *Composição de Funções* definida a partir de g, f_1, f_2, ..., f_k por:

$$h(x_1, x_2, \ldots, x_n) = g(\, f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_k(x_1, x_2, \ldots, x_n)\,)$$

Definição: Composição de Funções

- A função parcial h é dita *definida* para $(x_1, x_2, \ldots x_n)$ se, e somente se:

- $f_i(x_1, x_2, \ldots, x_n)$ é definida para todo $i \in \{ 1, 2, \ldots, k \}$
- $g(f_1(x_1, x_2, \ldots, x_n), f_2(x_1, x_2, \ldots, x_n), \ldots, f_k(x_1, x_2, \ldots, x_n))$ é definida.

Exemplo:

- Suponha as seguintes funções:
  - zero() = $\lambda x.0: N \rightarrow N$ (constante zero)
  - suc(x) = $\lambda x.x + 1: N \rightarrow N$ (sucessor)

- A seguinte função é definidas usando composição de funções:
  - um = $\lambda x.suc(zero(x)): N \rightarrow N$ (constante um)

Definição: Recursão

- Sejam f e g funções parciais tais que:

$f = \lambda(x_1, x_2,\ldots, x_n).f(x_1, x_2,\ldots, x_n): N^n \to N$

$g = \lambda(x_1, x_2,\ldots, x_n, y, z).g(x_1, x_2,\ldots, x_n, y, z): N^{n+2} \to N$

A função parcial h tal que:

$h = \lambda(x_1, x_2,\ldots, x_n, y).h(x_1, x_2,\ldots, x_n, y): N^{n+1} \to N$

é definida por *Recursão* a partir de f e g como segue:

Definição: Recursão

$h(x_1, x_2,\ldots, x_n, 0) = f(x_1, x_2,\ldots, x_n)$

$h(x_1, x_2,\ldots, x_n, y + 1) = g(x_1, x_2,\ldots, x_n, y, h(x_1, x_2,\ldots, x_n, y))$

A função h é dita *definida* para $(x_1, x_2,\ldots, x_n, y)$ se, e somente se:

$f(x_1, x_2,\ldots, x_n)$ é definida;

$g(x_1, x_2,\ldots, x_n, i, h(x_1, x_2,\ldots, x_n, i))$ é definida para todo $i \in \{ 1, 2, \ldots, y \}$

Exemplo:

- Suponha as seguintes funções:

(identidade)

id = $\lambda x.x$: N $\rightarrow$ N

(sucessor)

suc = $\lambda x.x + 1$: N $\rightarrow$ N

(projeção da 3ª componente da tripla)

$proj3_3 = \lambda(x,y,z).z$: $N^3 \rightarrow N$

Exemplo: adição

- A função adição nos naturais tal que: (adição)

$ad = \lambda(x,y).x + y: N^2 \rightarrow N$

é definida usando recursão, como:

$ad(x,0) = id(x)$

$ad(x,y+1)=proj3_3(x,y, suc(ad(x,y))$

Exemplo:

Por exemplo, ad(3, 2) é como segue:

ad(3, 2)

$= proj3_3\ (3, 1, suc(ad(3,1)))$

$= proj3_3\ (3, 1, suc(proj3_3(3, 0, suc(ad(3, 0)))))$

$= proj3_3\ (3, 1, suc(proj3_3(3,0, suc(id(3)))))$

$= proj3_3\ (3, 1, suc(proj3_3(3,0, suc(3))))$

$= proj3_3\ (3, 1, suc(proj3_3(3,0,4)))$

$= proj3_3\ (3, 1, suc(4))$

$= proj3_3\ (3, 1, 5)$

$= 5$

Definição: Minimização

- Seja f uma função parcial tal que:

$f = \lambda(x_1, x_2, \ldots, x_n, y).f(x_1, x_2, \ldots, x_n, {}_y): N^{n+1} \to N$

A função parcial h tal que:

$h = \lambda(x_1, x_2, \ldots, x_n).h(x_1, x_2, \ldots, x_n): N^n \to N$

Definição: Minimização

é dita definida por *Minimização* de f e é tal que:

$h = \lambda(x_1..x_n).\min\{\, y \mid f(x_1,\ldots,x_n, y)=0$ e, $(\forall z)\ \ z<y,$ $f(x_1,\ldots, x_n,z)$ é definida $\}$

Exemplo:

Suponha a função constante zero = $\lambda x.0: N \rightarrow N$.
Considere a seguinte função que identifica o número zero nos naturais:

$$const_{zero}: \rightarrow N$$

Note-se que é uma função *sem* variáveis, ou seja, é uma *constante*. A constante $const_{zero}$ é definida usando minimização como segue:

$$const_{zero} = min\{ y \mid zero(y) = 0 \}$$

Exemplo:

Sejam a constante zero $const_{zero}$, e a função de projeção:

$$proj2_1 = \lambda(x,y).x: N^2 \rightarrow N$$

A seguinte função antecessor nos naturais:

$$ant = \lambda x.ant(x): N \rightarrow N$$

Exemplo:

pode ser definida usando recursão (supondo que antecessor de 0 é 0)

$$ant(0) = const_{zero}$$

$$ant(y+1) = proj2_1(y, ant(y))$$

Exercício:

1) Calcule ant(3)

Exemplo:

A seguinte função subtração naturais:

$$sub = \lambda(x,y).sub(x, y): N^2 \rightarrow N$$

pode ser definida, usando recursão:

$sub(x, 0) = id(x)$

$sub(x, y + 1) = proj3_3 (x, y, ant(sub(x, y)))$

Exemplo:

sub(3, 2) =

= $proj3_3$ (3, 1, ant (sub(3, 1)))

=$proj3_3$ (3, 1,ant ($proj3_3$(3,0,ant (sub(3, 0)))))

= $proj3_3$ (3, 1, ant ($proj3_3$ (3, 0, ant (id(3)))))

= $proj3_3$ (3, 1, ant ($proj3_3$ (3, 0, ant (3))))

= $proj3_3$ (3, 1, ant ($proj3_3$ (3, 0, 2)))

= $proj3_3$ (3, 1, ant (2))

= $proj3_3$ (3, 1,1)

= 1

Função Recursiva Parcial e Total

- Funções recursivas parciais são definidas a partir de três funções básicas: constante zero, sucessor e projeção sobre o conjunto dos números naturais

Função Recursiva Parcial e Total

- Projeção não é uma função, mas uma família de funções, pois depende do número de componentes, bem como de qual a componente que se deseja projetar.

Definição: Função Recursiva Parcial

- Uma *Função Recursiva Parcial* é indutivamente definida como segue:

a) *Funções Básicas*. As seguintes funções são recursivas parciais:

$const_{zero} = \lambda x.0: N \rightarrow N$ (função constante zero)

$suc = \lambda x.x + 1: N \rightarrow N$ (função sucessor)

$projn_i = \lambda(x_1,x_2,\ldots,x_n).x_i: N^n \rightarrow N$

(projeção: i-ésima componente da n-upla)

Definição: Função Recursiva Parcial

b) Se $f_1, ..., f_n$ são Funções Recursivas Parciais então a *Composição, a Recursão* e a *Minimização das funções dadas também são*.

Exemplo:

a) *Função Identidade*

A função identidade, definida como segue:

$$id = \lambda x.x: N \to N$$

é recursiva parcial, pois é uma função básica de projeção, ou seja:

$$id = proj1_1$$

Exemplo:

b) *Função Zero*

A função Zero, definida como segue:

$$zero = \lambda x.0 : N \rightarrow N$$

é recursiva parcial, pois é uma função básica de projeção, ou seja:

$zero(0) = conts_{zero}$

$zero(y+1) = Proj2_2(y, zero(y))$

Exemplo:

c) *Função adição (ad)*

A função adição, definida como segue:

$$ad = \lambda(x,y).x + y: N^2 \rightarrow N$$

é recursiva parcial, pois é uma função recursiva parcial, ou seja:

$ad(x, 0) = id(x)$

$ad(x, y + 1) = proj3_3(x, y, suc(ad(x, y)))$

Exemplo:

d) *Função antecessor (ant)*

A função antecessor, definida como segue:

$$ant = \lambda(x).x - 1: N \rightarrow N$$

é recursiva parcial, pois é uma função recursiva parcial, ou seja:

$ant(0) = const_{zero}$

$ant(y+1) = proj2_1(y,ant(y))$

Exemplo:

e) *Função subtração (sub)*

A função subtração, definida como segue:

$$sub = \lambda(x,y).x - y: N^2 \rightarrow N$$

é recursiva parcial, pois é uma função recursiva parcial, ou seja:

$sub(x, 0) = id(x)$

$sub(x, y + 1) = proj3_3\ (\ x, y, ant(sub(x, y))\ )$

Exemplo:

f) *Função multiplicação (mul)*

A função subtração, definida como segue:

$$mul = \lambda(x,y).x * y: N^2 \rightarrow N$$

é recursiva parcial, pois é uma função recursiva parcial, ou seja:

$mul(x, 0) = zero(x)$

$mul(x, y + 1) = ad(\ proj3_1(x, y, mul(x,y)),$
$\qquad proj3_3(x, y, mul(x,y)))$

Exemplo:

g) *Função exponenciação (exp)*

A função subtração, definida como segue:

$$exp = \lambda(x,y).x^y: N^2 \rightarrow N$$

é recursiva parcial, pois é uma função recursiva parcial, ou seja:

$exp(x, 0) = suc(zero(x))$

$exp(x, y + 1) = mul(\ proj3_1(x, y, exp(x,y)),$
$proj3_3(x, y, exp(x,y)))$

Exercício:

a) Mostre que a função fatorial é recursiva parcial.

Definição: (Função Recursiva Total)

- *Uma* Função Recursiva Total *é uma função recursiva parcial definida para todos os elementos do domínio.*

Teorema:

- As seguintes classes de funções são equivalentes:

a) Funções Recursivas Parciais e Funções Turing-Computáveis;

b) Funções Recursivas Totais e Funções Turing-Computáveis Totais.

Importante:

- A relação entre classes também pode ser estabelecida:

Funções Recursivas Parciais $\Leftrightarrow$ Linguagens Enumeráveis Recursivamente;

Funções Recursivas Totais $\Leftrightarrow$ Funções Turing-Computáveis Totais $\Leftrightarrow$ Linguagens Recursivas.