Funções Parciais Recursivas e Cálculo-λ

Carlos A. P. Campani

4 de novembro de 2008

1 Introdução

- Funções parciais recursivas são equivalentes ao formalismo Máquina de Turing;
- Representam as funções que podem ser computadas em uma máquina.

Tipos de formalismos para especificar algoritmos:

Operacional Máquinas abstratas (Máquina de Turing);

Axiomático Regras definem os componentes da linguagem (Gramáticas);

Funcional ou Denotacional Funções construídas de forma a serem compostas – *linguagem-*λ (Alonzo Church) e *funções parciais recursivas* (Kleene).

Equivalências entre máquinas e gramáticas (hierarquia de Chomsky):

- Autômatos finitos $\equiv$ Gramáticas regulares;
- Autômatos de pilha não-determinísticos $\equiv$ Gramáticas livres de contexto;
- Máquinas universais (Máquina de Turing) $\equiv$ Gramáticas irrestritas.

2 Cálculo-λ

- Formalismo para representar funções proposto por Alonzo Church (1941);
- Fornece um sistema axiomático para o cálculo com as expressões da linguagem-λ;
- Linguagem-$\lambda \equiv$ linguagem funcional (LISP).

2.1 Definição de Expressão-λ

Definição 1 *(*expressão-λ *ou* termo-λ*)*

1. *Uma variável é uma expressão-λ;*
2. *Se M é uma expressão-λ e x é uma variável, então $\lambda x M$ é uma expressão-lambda, interpretada como "uma função com argumento x";*
3. *Se F e A são expressões-λ, então (FA) é uma expressão-λ, interpretada como "F aplicado ao argumento A";*
4. *Nada mais é expressão-λ.*

Exemplos 1

1. $\lambda x \overbrace{x}^{M}$;
2. $(\overbrace{\lambda x x}^{F} \underbrace{(yz)}_{A})$;
3. $(\overbrace{\lambda x \underbrace{(xx)}_{M}}^{F} \overbrace{y}^{A})$;
4. $(\underbrace{\lambda x x}_{F} \underbrace{\lambda x x}_{A})$;
5. $\lambda x \underbrace{\lambda y (xy)}_{M}$.

Exercício 1 *Determine as expressões válidas:*

1. $\lambda xx;$
2. $\lambda x;$
3. $\lambda x \lambda y x.$

2.2 Variáveis Livres e Limitadas

Se uma ocorrência de uma variável x está no escopo de um λx, então sua ocorrência é dita *limitada*, caso contrário é dita *livre*.

Exemplo 1 $(x\lambda x\lambda y(xy))$

Primeira ocorrência de x é livre, a segunda é limitada.

2.3 Substituição de Variáveis

$M[x \leftarrow A]$ denota a substituição uniforme de todas as ocorrências *livres* de x por A.

Exemplo 2 $(x\lambda x\lambda y(xy))[x \leftarrow \lambda zz] = (\lambda zz\lambda x\lambda y(xy))$.

2.4 Reduções do Cálculo-λ

$$(FA)$$

F funcional;

A argumento.

$$(\underbrace{\lambda x M}_{F} A) \Rightarrow M[x \leftarrow A]$$

Exemplo 3

$$(\lambda x x(yz)) \Rightarrow (yz)$$

Exemplos 2

1. $(\lambda xx\lambda xx) \Rightarrow \lambda xx$;
2. $((\lambda x\lambda y(xy)\lambda xx)x) \Rightarrow (\lambda y(\lambda xxy)x) \Rightarrow (\lambda xxx) \Rightarrow x$;
3. $(\lambda x(xx)\lambda x(xx)) \Rightarrow (\lambda x(xx)\lambda x(xx))$ *(irredutível)*;
4. $(\lambda xyz) \Rightarrow y$ *(jogar fora alguma coisa)*.

Exercício 2 *Efetue as seguintes reduções:*

1. $(\lambda z(\lambda yzx)(xx))$
2. $(\lambda xx\lambda xx)$
3. $(\lambda x(xx)\lambda yy(xx))$

2.5 Currying

- Ocorre quando da aplicação de um termo-λ em que existem menos argumentos que variáveis limitadas;

$$(\lambda x \lambda y (xy) z) \Rightarrow \lambda y (zy)$$

- Na matemática: $f(x, y)$, fixando um x qualquer, resulta em uma função de y;
- Natural de fazer na programação funcional/difícil de fazer na programação procedural (necessário editar o fonte e atribuir os valores que não serão lidos).

2.6 Aplicação-λ e Abstração-λ

Abstração-λ $M \Rightarrow \lambda x M$;

Aplicação-λ $(\lambda x M A) \Rightarrow M[x \leftarrow A]$.

$$\underbrace{(\lambda x M A)}_{\text{redex}} \Rightarrow \underbrace{M[x \leftarrow A]}_{\text{contractum}}$$

Definição 2 *Uma expressão-λ que não pode ser mais reduzida é chamada* forma normal.

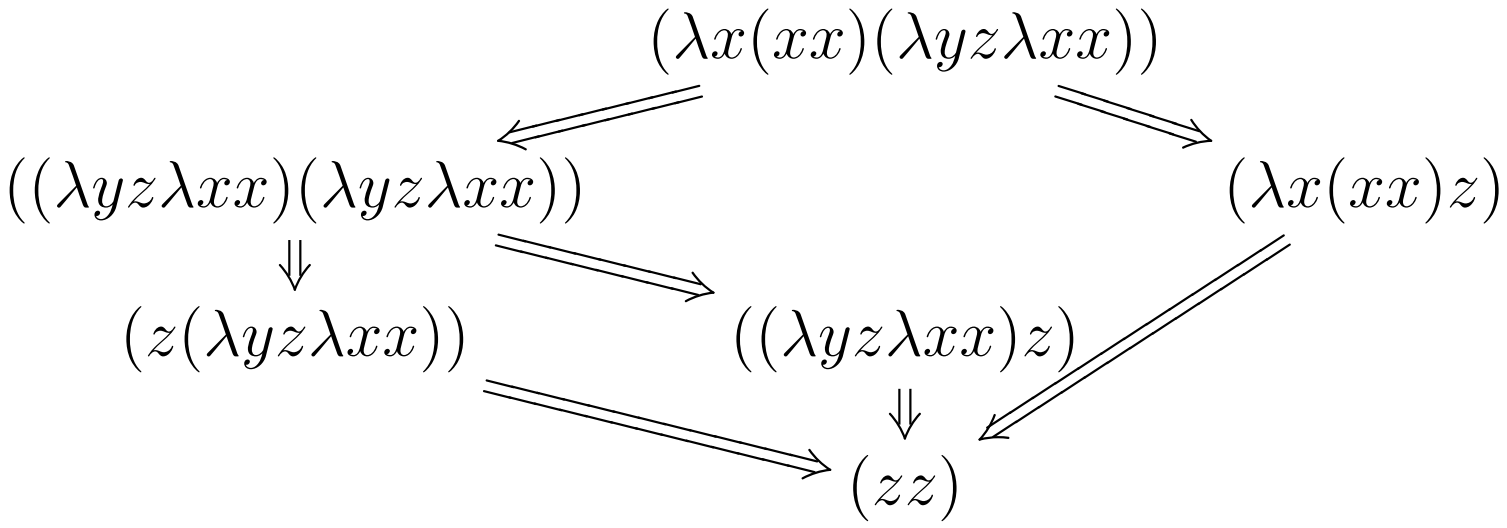
Exemplo 4 *λxx é uma forma normal.*

Exemplo 5 *$(\lambda xx(yz))$ não é uma forma normal.*

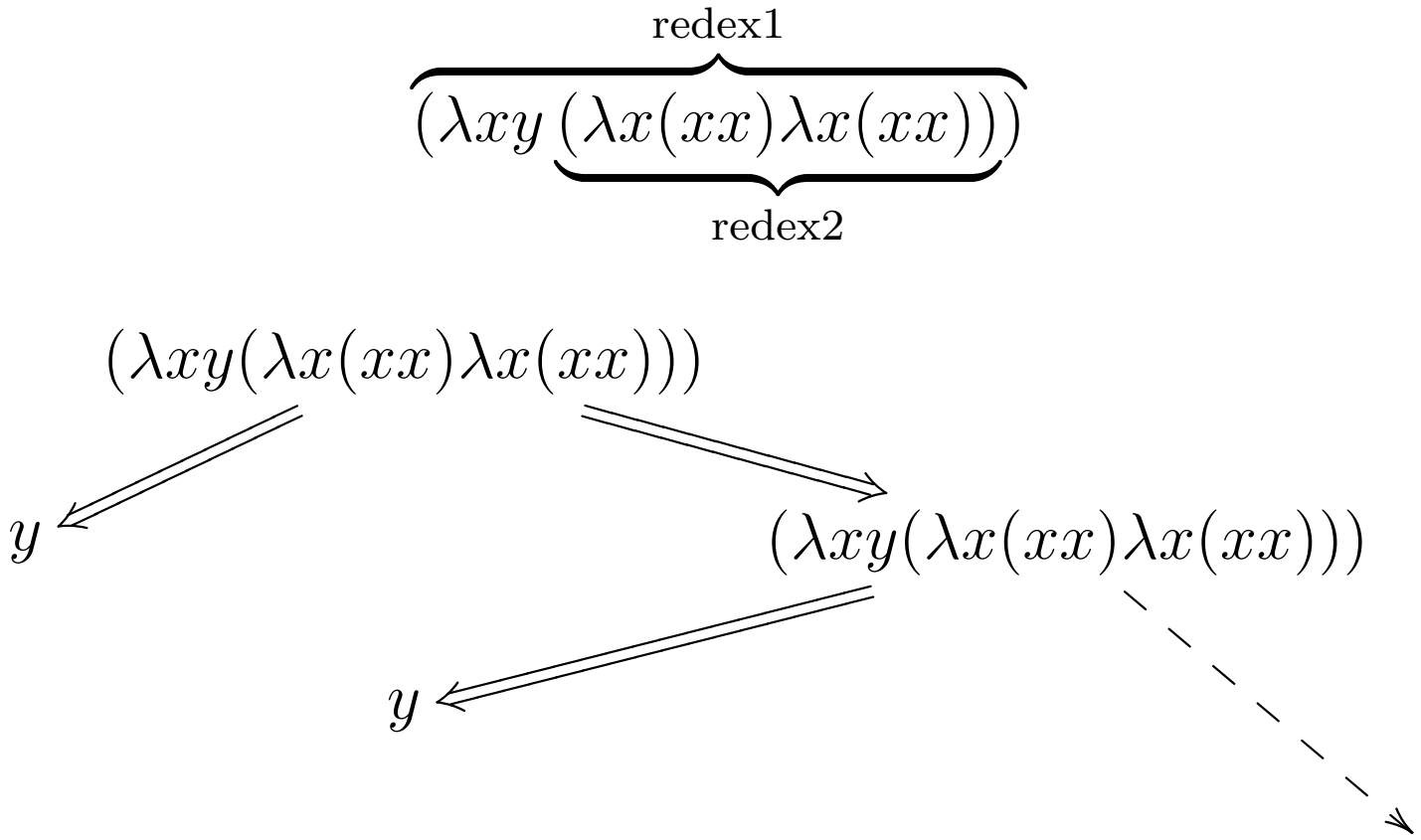
Exemplo 6 *$(\lambda x(xx)\lambda x(xx))$ não é uma forma normal.*

2.7 Teorema de Church-Rosser

- Podem existir mais de uma redução possíveis (mais de um redex)



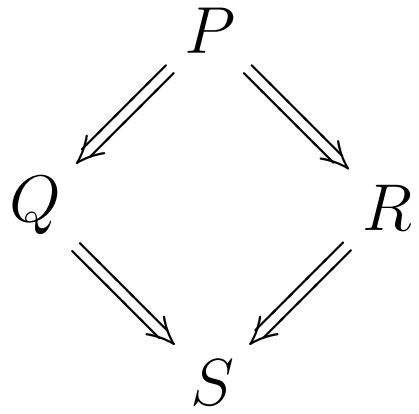
- Pode haver caminhos sem saída



- Considerando os diversos caminhos, seria a resposta da avaliação das expressões única? Ou seja, seriam as formas normais idênticas?

Teorema 1 *(Teorema de Church-Rosser)*

Para qualquer expressão-λ P e para quaisquer Q e R, se $P \Rightarrow Q$ e $P \Rightarrow R$, então existe um S tal que $Q \Rightarrow S$ e $R \Rightarrow S$.



2.8 Teorema da Normalização

Sempre usar o redex mais à esquerda e mais externo primeiro em uma redução.

estratégia normal ≈ call by name ≈ eal ≈ menor ponto fixo

2.9 Representação dos Conectivos da Lógica

```
if A then B else C
```

$$T \equiv \lambda x \lambda y x$$

$$((Ta)b) \equiv ((\lambda x \lambda y x a)b) \Rightarrow (\lambda y a b) \Rightarrow a$$

$$F \equiv \lambda x \lambda y y$$

$$((Fa)b) \equiv ((\lambda x \lambda y y a)b) \Rightarrow (\lambda y y b) \Rightarrow b$$

$$\text{not} \equiv \lambda x((xF)T)$$

Exemplo 7 *(*not$F$*)*

$(\lambda x((xF)T)F) \Rightarrow ((FF)T) \Rightarrow T$

$$\text{and} \equiv \lambda x \lambda y((xy)F)$$

$$\text{or} \equiv \lambda x \lambda y((xT)y)$$

$$\rightarrow \equiv \lambda x \lambda y((xy)T)$$

2.10 Manipulação de Listas

Usar F e T como seletores de elementos de listas (if-then-else aninhados).

- $T \equiv \lambda x \lambda y x$ (primeiro elemento da lista);
- $FT \equiv \lambda x \lambda y (y \lambda x \lambda y x) \equiv \lambda x \lambda y (yT)$ (segundo elemento da lista);
- $F^2T \equiv \lambda x \lambda y (y \lambda x \lambda y (y \lambda x \lambda y x)) \equiv \lambda x \lambda y (yFT)$ (terceiro elemento da lista);
- $F^{i+1}T \equiv \lambda x \lambda y (yF^iT)$ (o $(i+2)$-ésimo elemento).

$$\langle \phi_0, \phi_1, \ldots, \phi_{n-1} \rangle$$

- $\langle \phi_0 \rangle \equiv \lambda x((x\phi_0)\psi)$ (ψ é o terminador de lista);
- $\langle \phi_0, \phi_1 \rangle \equiv \lambda x((x\phi_0)\lambda x((x\phi_1)\psi)) \equiv \lambda x((x\phi_0)\,\langle \phi_1 \rangle)$;
- $\langle \phi_0, \phi_1, \ldots, \phi_{n-1} \rangle \equiv \lambda x((x\phi_0)\,\langle \phi_1, \ldots, \phi_{n-1} \rangle)$.

(obtendo o primeiro elemento de uma lista)

$$\begin{aligned}(\langle\phi_0\rangle\, T) \quad &\equiv \quad (\lambda x((x\phi_0)\psi)\lambda x\lambda y x) \Rightarrow ((\lambda x\lambda y x\phi_0)\psi) \Rightarrow \\ &\Rightarrow (\lambda y\phi_0\psi) \Rightarrow \phi_0\end{aligned}$$

(obtendo o segundo elemento de uma lista)

$$(\langle \phi_0, \phi_1, \phi_2 \rangle \, FT) \equiv$$

$$\equiv (\lambda x((x\phi_0)\lambda x((x\phi_1)\lambda x((x\phi_2)\psi)))\, \underbrace{\lambda x \lambda y(y\lambda x \lambda y x)}) \Rightarrow$$

$$\Rightarrow ((\lambda x \lambda y(y\lambda x \lambda y x)\, \underbrace{\phi_0}\,)\lambda x((x\phi_1)\lambda x((x\phi_2)\psi))) \Rightarrow$$

$$\Rightarrow (\lambda y(y\lambda x \lambda y x)\, \underbrace{\lambda x((x\phi_1)\lambda x((x\phi_2)\psi))}) \Rightarrow$$

$$\Rightarrow (\lambda x((x\phi_1)\lambda x((x\phi_2)\psi))\, \underbrace{\lambda x \lambda y x}) \Rightarrow$$

$$\Rightarrow ((\lambda x \lambda y x\, \underbrace{\phi_1}\,)\lambda x((x\phi_2)\psi)) \Rightarrow$$

$$\Rightarrow (\lambda y \phi_1\, \underbrace{\lambda x((x\phi_2)\psi)}) \Rightarrow \phi_1$$

2.11 Relação com a Programação Funcional (LISP)

$T \equiv$ `CAR` $\quad F \equiv$ `CDR` $\quad \psi \equiv$ `nil`

`(CAR (CDR (CAR QUOTE((A B C) D))))=B`

2.12 Representação de Números Inteiros

$$i \equiv F^i T$$

$$0 \equiv T$$

$$1 \equiv FT$$

$$2 \equiv FFT$$

$$\vdots$$

$$\text{suc} \equiv \lambda z \lambda x \lambda y (yz)$$

$$(\text{suc } 1) \equiv (\lambda z \lambda x \lambda y (yz) \lambda x \lambda y (y \lambda x \lambda y x)) \Rightarrow$$

$$\lambda x \lambda y (\underbrace{y \, \lambda x \lambda y (y \, \underbrace{\lambda x \lambda y x}_{T})}_{FT}) \equiv FFT \equiv 2$$

Da observação que podemos escrever as expressões-λ para pred, $+$, $-$, mult etc. concluimos que

$$\text{Cálculo-}\lambda \approx \text{máquina de Turing}$$

2.13 Igualdade do Cálculo-λ

Definição 3 *(Redução beta)* $(\lambda x M A) \Rightarrow M[x \leftarrow A]$.

Definição 4 *(Redução alfa)* $\lambda x M \Rightarrow \lambda y M[x \leftarrow y]$.

As reduções alfa e beta induzem uma igualdade das expressões-λ (igualdade extensional).

$=$ igualdade extensional;

$\equiv$ igualdade intencional (baseada na equivalência de abreviaturas).

2.14 Sistema Axiomático do Cálculo-λ

Serve para julgar a igualdade extensional entre termos do cálculo-λ.

$\lambda \vdash M = N$ se e somente se existe uma dedução de $M = N$.

2.14.1 Axiomas/Regras de Inferência

$$M = M$$

$$(\lambda x M A) = M[x \leftarrow A]$$

$$\frac{M = N}{N = M}$$

$$\frac{M = N, N = K}{M = K}$$

$$\frac{M = N}{(MA) = (NA)}$$

$$\frac{M = N}{(FM) = (FN)}$$

$$\frac{M = N}{\lambda x M = \lambda x N}$$

Cálculo-λ=linguagem-λ+sistema axiomático

3 Funções Parciais Recursivas

- Propostas por Kleene (1936);
- Equivalentes ao formalismo Máquina de Turing e linguagem-λ.

3.1 Funções e Funcionais

Definição 5 *Uma* função parcial *é uma relação* $f \subseteq A \times B$ *onde cada elemento de* A *se relaciona com, no máximo, um elemento de* B*. O conjunto* A *é chamado de* domínio *da função e o conjunto* B *de* co-dominio.

Notação: Denotamos a função $f \subseteq A \times B$ como $f : A \to B$ e diz-se que o *tipo* de f é $A \to B$. $< a, b > \in f$ é denotado por $f(a) = b$.

Exemplo 8 *Seja a função* $f : \mathbb{N} \to \mathbb{N}$, *definida como* $f(x) = x^2$. *Assim,* $f(3) = 9$.

- Uma função é *total* se ela está definida para todo o seu domínio;
- Uma função $f : A \to B$ é *parcial* se $\exists x \in A(\nexists y \in B f(x) = y)$. Exemplo: $f : \mathbb{R} \to \mathbb{R}$, $f(x) = 1/x$. Observe que $f(0)$ não está definido.

Definição 6 *Um* funcional *é uma função que possui uma ou mais funções como argumentos.*

Exemplo 9 *Seja o funcional* $h : (\mathbb{N} \to \mathbb{N}) \times \mathbb{N} \to \mathbb{N}$, *tal que* $h(f, x) = f(x)$ *e* $f : \mathbb{N} \to \mathbb{N}$.

3.2 Definição de Função Parcial Recursiva

- Funções parciais recursivas são funções construídas sobre funções básicas usando cinco tipos de construções: composição; condicional; recursão primitiva, recursão while e minimização;
- Função Turing-computável $\equiv$ função parcial recursiva;
- Função Turing computável para máquina que sempre pára $\equiv$ função recursiva (total).

3.2.1 Funções Básicas

Função sucessor $s : \mathbb{N} \to \mathbb{N}$, definida como

$s(x) = x + 1$;

Função predecessor $p : \mathbb{N} \to \mathbb{N}$, definida como

$$p(x) = \begin{cases} x - 1 & \text{se } x > 0 \\ 0 & \text{se } x = 0 \end{cases};$$

Projeção $p_n^i : \mathbb{N}^n \to \mathbb{N}$, definida como

$p_n^i(x_1, x_2, \ldots x_n) = x_i$, para $1 \leq i \leq n$.

3.2.2 Composição Generalizada

Definição 7 *Sejam $g, f_1, f_2, f_3, \ldots, f_k$ funções parciais tais que $g : \mathbb{N}^k \to \mathbb{N}$ e $f_i : \mathbb{N}^n \to \mathbb{N}$ para $1 \leq i \leq k$. A função parcial h, definida como*

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

é a composição das funções $g, f_1, f_2, f_3, \ldots, f_k$.

Exemplo 10 *A função* $\text{soma2} : \mathbb{N} \to \mathbb{N}$, *definida como* $\text{soma2}(x) = s(s(x))$, *usa a construção* composição *e resulta numa função que soma dois ao valor de seu argumento.*

3.2.3 Condicional

Definição 8 *A função* $\mathrm{cond} : \{V, F\} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, *definida como*

$$\mathrm{cond}(b, g_1, g_2) = \begin{cases} g_1 & \text{se } b = V \\ g_2 & \text{se } b = F \end{cases},$$

é a construção condicional, *onde* b *é uma expressão lógica,* g_1 *e* g_2 *são dois valores quaisquer.*

$$\text{cond}(b, g_1, g_2) = \text{se } b \text{ então } g_1 \text{ senão } g_2$$

Exemplo 11 $\text{maior} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, *definida como* $\text{maior}(x, y) = \text{cond}(x > y, x, y)$, *resulta no maior valor entre* x *e* y.

3.2.4 Recursão Primitiva

Definição 9 *A função* $f : \mathbb{N}^{n+1} \longrightarrow \mathbb{N}$, *definida como*

$$f(x_1, \ldots, x_n, y) = \text{cond}(y = 0, g(x_1, \ldots, x_n), h(x_1, \ldots, x_n, p(y), f(x_1, \ldots, x_n, p(y))))$$

é chamada de recursão primitiva. *Na definição,* $h : \mathbb{N}^{n+2} \longrightarrow \mathbb{N}$ *e* $g : \mathbb{N}^n \longrightarrow \mathbb{N}$ *são duas funções quaisquer.*

$$f(x_1, \ldots, x_n, y) = \begin{cases} g(x_1, \ldots, x_n) & \text{se } y = 0 \\ h(x_1, \ldots, x_n, p(y), f(x_1, \ldots, x_n, p(y))) & \text{se } y \neq 0 \end{cases}$$

Exemplo 12

$\text{zero}(x) = \text{cond}(x = 0, p_1^1(x), p_2^2(p(x), \text{zero}(p(x))))$, *define uma função que resulta no valor constante zero.*

3.2.5 Recursão While

Definição 10 *A função*

$$f(x_1, x_2, \ldots, x_n) = \begin{cases} g(x_1, \ldots, x_n) & \text{se } x_i = 0 \\ f(h_1(x_1, \ldots, x_n), \ldots, h_n(x_1, \ldots, x_n)) & \text{se } x_i > 0 \end{cases}$$

é chamada de recursão while.

3.2.6 Minimização

A função $f(x_1, \ldots, x_n) = \mu y : h(x_1, \ldots, x_n, y)$, definida como o menor valor y tal que $h(x_1, \ldots, x_n, y) = 0$ e para todo $z < y$, $h(x_n, \ldots, x_n, z)$ está definido, é chamada de *minimização*.

3.2.7 Definições

Definição 11 *As* funções while recursivas *compreendem a menor classe de funções que inclui as funções básicas e é fechado sobre a composição generalizada, condicional e recursão while.*

Definição 12 *As* funções primitivas recursivas *compreendem a menor classe de funções que inclui as funções básicas e é fechado sobre a composição generalizada e recursão primitiva.*

Definição 13 *As* funções parciais recursivas *compreendem a menor classe de funções que inclui as funções básicas e é fechado sobre a composição generalizada, recursão primitiva e minimização.*

funções Turing computáveis $\equiv$ funções while recursivas $\equiv$ funções parciais recursivas

funções primitivas recursivas $\sqsubseteq$ funções parciais recursivas

Exemplo 13 *(Função while recursiva)*

$$soma(x,y) = \begin{cases} x & \text{se } y = 0 \\ s(soma(x, p(y))) & \text{se } y > 0 \end{cases}$$

Exercício 3 *Escreva as seguintes funções while recursivas:*

1. $\mathrm{sub}(x, y) = x - y$;
2. $\mathrm{mult}(x, y) = x \times y$;
3. $\mathrm{fat}(x) = x!$.

Exemplo 14 *Avaliação da função parcial recursiva*
$\mathrm{f}(x) = \mathrm{cond}(x = 0, s(zero(x)), s(p_2^2(p(x), f(p(x)))))$*:*

$$
\begin{aligned}
\mathrm{f}(3) &= \mathrm{cond}(3 = 0, s(zero(3)), s(p_2^2(p(3), f(p(3))))) = \\
&= s(p_2^2(p(3), f(p(3)))) = s(f(p(3))) = \\
&= s(\mathrm{cond}(p(3) = 0, s(zero(p(3))), \\
&\quad s(p_2^2(p(p(3)), f(p(p(3)))))) = \\
&= s(s(p_2^2(p(p(3)), f(p(p(3)))))) = s(s(f(p(p(3))))) = \\
&= s(s(\mathrm{cond}(p(p(3))) = 0, \\
&\quad s(zero(p(p(3)))), s(p_2^2(p(p(p(3)))), \\
&\quad f(p(p(p(3)))))))))) = \\
&= s(s(s(p_2^2(p(p(p(3)))), f(p(p(p(3))))))))))) = \\
&= s(s(s(f(zero(x))))) =
\end{aligned}
$$

$$= \quad s(s(s(\mathrm{cond}(0=0, s(zero(0)), s(p_2^2(p(0),$$
$$f(p(0)))))))) = s(s(s(s(zero(0)))))$$

Observe-se que "3" é uma abreviatura para "$s(s(s(zero(x))))$*."*

4 Pontos Fixos

- Já vimos uma interpretação computacional das funções recursivas;
- Pontos fixos: Interpretação matemática das funções recursivas.

4.1 Definições

Definição 14 *A função* $\perp : \mathbb{N} \rightarrow \mathbb{N}$, *definida como* $\perp(x) = \underline{\text{undef}}$, *é chamada de* função totalmente indefinida.

Definição 15 *A relação* $\sqsubseteq$ *sobre* $\mathcal{F} \times \mathcal{F}$, *onde* $\mathcal{F}$ *é o conjunto das funções sobre* $\mathbb{N} \rightarrow \mathbb{N}$, *definida como* $f_1 \sqsubseteq f_2$ *se* $f_1(x) = y \rightarrow f_2(x) = y$, *onde* $f_1, f_2 \in \mathcal{F}$, *é uma relação de ordem parcial sobre* $\mathcal{F}$.

Observação: $\perp \sqsubseteq f$, para qualquer $f \in \mathcal{F}$.

Definição 16 *Um conjunto de funções* $\{f_i | i \geq 0\}$ *é chamado de* cadeia *se* $f_1 \sqsubseteq f_2 \sqsubseteq f_3 \sqsubseteq \cdots$.

Definição 17 *Uma função* f *é chamada* ponto fixo *do funcional* F *se* $F(f) = f$.

Teorema 2 *O ponto fixo f_0 do funcional F é o supremo da cadeia $F^i(\bot)$, $f_0 = \sqcup\{F^i(\bot)|i \geq 0\}$.*

Observações:

- $F^i = F \circ F^{i-1}$;
- $F^0(\bot) \sqsubseteq F^1(\bot) \sqsubseteq F^2(\bot) \sqsubseteq \cdots$;
- Identificamos o ponto fixo do funcional com a função computada pelo programa associado a este funcional.

Exemplo 15 *Seja* $f(x) = cond(x = 0, 1, x * f(x-1))$.
Cadeia:

1. $F^0(\perp) = \mathrm{id}(\perp) = \perp = \underline{\text{undef}}$;
2. $F^1(\perp) = F(\perp) = cond(x = 0, 1, x * \perp(x-1)) = cond(x = 0, 1, \underline{\text{undef}})$;
3. $F^2(\perp) = F \circ F^1(\perp) = cond(x = 0, 1, x * (F^1(\perp))(x-1)) = cond(x = 0, 1, x * cond(x - 1 = 0, 1, \underline{\text{undef}})) = cond(x = 0, 1, x * cond(x = 1, 1, \underline{\text{undef}}))$

Exercício 4 *Calcular* $F^3(\perp)$.

Observação: $\sqcup\{F^i(\perp)|i \geq 0\} =$ fatorial.