MATA51 – Teoria da Computação
Profa. Laís Salvador

TEORIA DAS FUNÇÕES RECURSIVAS - Funções Recursivas Parciais

Revisitando a definição:

Seja $g\ (x_1, ..., x_n, y)$ uma função total e computável.

Seja também, $\mu y\ [g\ (x_1, ..., x_n, y) = 0]$ o menor valor de y – **se existir** – tal que $g\ (x_1, ..., x_n, y) = 0$. Então podemos construir a função:

$$f\ (x_1, ..., x_n) = \mu y\ [g\ (x_1, ..., x_n, y) = 0]$$

Intuitivamente, a minimização busca - começando a busca de 0 e prosseguindo para cima - o menor argumento y que faz com que a função g retorne zero; se não houver tal argumento, então a pesquisa nunca termina, e f não estará definida para os argumentos $x_1, ..., x_n$.

Assim, pode acontecer de $f(x_1, ..., x_n) = \mu y\ [g\ (x_1, ..., x_n, y) = 0]$ ser uma função parcial, ou seja, definida apenas para algumas *n-tuplas* de números naturais $x_1, ..., x_n$.

Por esse motivo, as funções caracterizadas pela adição desse construtor (operador de minimização) aos construtores das funções recursivas primitivas são definidas **Funções Recursivas Parciais**.

➔ As **Funções Recursivas Parciais** correspondem exatamente ao **conjunto de funções computáveis.**

➔ **Funções Recursivas Parciais**: Funções definidas a partir das **funções iniciais** e pela aplicação dos construtores de **substituição**, **recursão** e **minimização**.

Exemplos já discutidos:

(1) f (x) = μ y [x+ y = 0]

(2) max (x,y) = μ z [*(x ~ z) + (y ~ z)* = 0]

Outros exemplos:

(3) *f(x) =μy [rem (x, 2) + y = 0]*

É um exemplo de função recursiva parcial, pois

$$f(x) = \begin{cases} 0, se\ x\ \text{é}\ par \\ \uparrow\ (indefinida), se\ x\ \text{é}\ \text{ímpar} \end{cases}$$

Como *f* é parcial, *f* não é primitiva recursiva.

(4) Seja *g(x, y) = | x – y²|* e

Seja *f(x) =μy [g(x, y) = 0]*, isto é,

f é obtida de *g* através do operador de minimização.

Então temos que,

$$f(x) = \begin{cases} \sqrt{x}, se\ x\ \text{é}\ raiz\ quadrada\ perfeita \\ \uparrow\ (indefinida), caso\ contr\text{á}rio \end{cases}$$

(5) Seja *g(x, y) = equals (x, 3.y)*

Seja *f(x) = μy [g(x, y) = 1],*

(definição da operação de minimização segundo Lewis & Papadimitriou)

Então temos que:

$$f(x) = \begin{cases} x/3, se\ x \text{ é } divisível\ por\ 3 \\ \uparrow (indefinida), caso\ contrário \end{cases}$$

onde,

$$equals\ (x, y) = \begin{cases} 1, se\ x = y \\ 0, se\ x \neq y \end{cases}$$

Funções Recursivas Parciais e Máquinas de Turing

As mTs servem para reconhecer linguagens, computar funções computáveis, definir conceitos abstratos como chamadas de sub-rotinas e realocação de espaço de memória pelo deslocamento de dados na fita.

- As Funções Recursivas Parciais correspondem exatamente ao conjunto de funções computáveis.
  - Por que? (Hopcroft)

- Uma Máquina de Turing (MT) pode ser vista como um computador de funções de naturais para naturais. Podemos assumir a codificação unária, onde um número $i \in N$ é representado pelo string 0^i.

- Se a função tem *k* argumentos, $i_1, i_2, \ldots, i_k$, então esses valores são inicialmente armazenados sobre uma fita separada por *1's* como:

$$0^{i1}10^{i2}1 \ldots 10^{ik}$$

- Se a MT pára com uma fita consistindo de 0^m para algum m, então $f(i_1, i_2, \ldots, i_k) = m$, onde f é a função de k argumentos computada pela máquina.

- Note que se a MT computa uma função f de k argumentos, então f **não** necessita assumir valores para todas a *k-tuplas* de $i_1, i_2, \ldots, i_k$.

- Logo, de uma forma geral, uma função $f(i_1, i_2, ..., i_k)$ computada por uma Máquina de Turing é chamada de **Função Recursiva Parcial.**
    - Neste caso a MT pode ou não parar sobre uma dada entrada.

$$f(i_1, i_2, ..., i_k) = \begin{cases} m, \text{ para algum } m \text{ (a MT pára)} \\ \uparrow \text{ (a MT não pára)} \end{cases}$$

- Por outro lado, se $f(i_1, i_2, ..., i_k)$ é definida para $\forall\, i_1, i_2, ..., i_k$ então f é uma **Função Recursiva Total.**

Resumo (segundo Silva e Melo)

As Máquinas de Turing computam as funções recursivas parciais, que são análogas às linguagens recursivamente enumeráveis (RE).

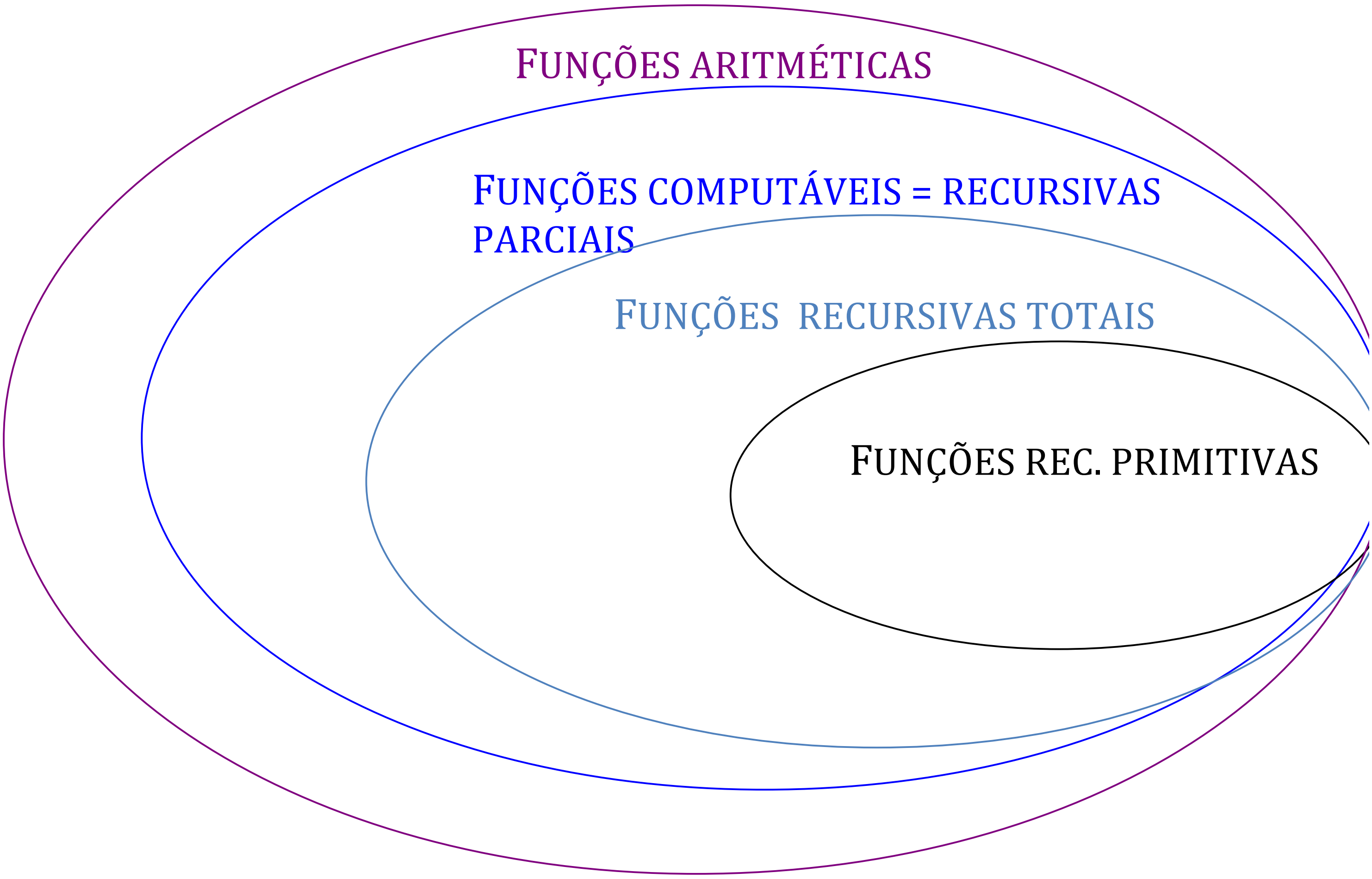
- Para as funções parciais, a MT pára para o conjunto de entradas definidas para a função, e pode não parar para as entradas não definidas no domínio da função. Nas linguagens RE, as suas MT reconhecedoras podem não parar para algumas cadeias de entrada.

Por sua vez, as funções recursivas totais (definidas para todo o domínio) são análogas às linguagens recursivas, e por isso, suas MTs páram para todos os dados de entrada.

- Quando implementamos/programamos funções recursivas em Computação, estamos falando de funções recursivas totais

Relações de Inclusão

Funções Primitivas Recursivas $\subset$ Funções Recursivas Totais $\subset$ Funções Recursivas Parciais $\subset$ Todas as Funções



A noção intuitiva de que uma **função computável** pode ser identificada com a classe das **funções recursivas parciais** é conhecida como a **Tese de Church-Turing**.

Outros modelos computacionais, como o **lambda-cálculo**, também foram desenvolvidos com o intuito de computar **funções recursivas parciais**.

Apesar de não poder ser provada formalmente, a **Tese de Church-Turing** estabelece que as **máquinas de Turing** sejam um modelo suficiente para as **funções computáveis**.

Referências

1. **Modelos Clássicos de Computação.** Flavio Soares Correa da Silva e Ana Cristina Vieira de Melo. Cengage, 2010.
2. **Elementos de teoria da computação.** LEWIS, Harry R.; PAPADIMITRIOU, Christos H. Bookman, 2000.
3. **Introduction to Automata Theory, Languages and Computation**. HOPCROFT, J. E., AND ULLMAN, J. D. (1979), Addison-Wesley, Reading, Mass.
4. **Machines, languages, and computation.** Peter J. Denning, Jack Bonnell Dennis, Joseph E. Qualitz Prentice-Hall, 1978.
5. **Conceitos Elementares da Teoria da Computação** – Módulos 1 e 2. Sonia Limoeiro MONTEIRO, Petrópolis, LNCC (Relatório de pesquisa), 2004.