Funções Numéricas

Prof^a Jerusa Marchi

Departamento de Informática e Estatística

Universidade Federal de Santa Catarina

e-mail: jerusa@inf.ufsc.br

Funções Numéricas

- As funções numéricas computáveis através de Máquinas de Turing são exatamente aquelas que podem ser definidas recursivamente a partir de um conjunto adequado de **funções básicas**

Funções Numéricas

- Funções Básicas ($f : \mathcal{N}^k \mapsto \mathcal{N}$):
  - função nula:

$$zero(n) = 0 \text{ para todo } n$$

  - função sucessor:

$$succ(n) = n + 1$$

  - projeção:

$$proj_k^i(x_1, \cdots, x_k) = x_i \text{ para } 1 \leq i \leq k$$

$proj_i^i = i$ é chamada de função identidade

Operações Básicas

- Composição:
  - Se g é uma função m-ária e $h_1, \cdots, h_m$ são funções k-árias, então a composição gera a função:

$$f(x_1, \cdots, x_k) = g(h_1(x_1, \cdots, x_k), \cdots, h_m(x_1, \cdots, x_k))$$

    ou simplesmente $f(\overrightarrow{x}) = g(h_1(\overrightarrow{x}), \cdots, h_m(\overrightarrow{x}))$

Operações Básicas

- Recursão Primitiva:
  - Para funções de uma variável:

$$f(0) = d$$
$$f(n+1) = h(f(n), n)$$

onde d é um número e h é uma função já definida

  - Para funções de duas ou mais variáveis, se g e h já estão definidas, então f é dada por *recursão primitiva em h com base em g*:

$$f(0, \overrightarrow{x}) = g(\overrightarrow{x})$$
$$f(n+1, \overrightarrow{x}) = h(f(n, \overrightarrow{x}), n, \overrightarrow{x})$$

n e $\overrightarrow{x}$ aparecem em h para manter uma memória da entrada no passo atual

Funções Numéricas

- Definição indutiva da classe de funções
  - Funções recursivas primitivas são exatamente as funções básicas e as que podem ser obtidas a partir delas por qualquer número de aplicações sucessivas das operações de composição e de recursão primitiva.

Funções Numéricas

- Exemplos:
  - Constantes: Obtidas pela composição da função nula com a função sucessor.

$$succ(succ(...succ(zero(x))...))$$

  - Adição: Obtida pela combinação das funções identidade, nula e sucessor.

$$\begin{aligned} plus(m, 0) &= m \\ plus(m, n+1) &= succ(plus(m, n)) \end{aligned}$$

Funções Numéricas

- Exemplos:
  - Multiplicação: $mult(m,n) = m.n$

$$\begin{aligned} mult(m,0) &= zero(m) \\ mult(m,n+1) &= plus(m, mult(m,n)) \end{aligned}$$

  - Exponencial: $exp(b,e) = b^e$

$$\begin{aligned} exp(m,0) &= succ(zero(m)) \\ exp(m,n+1) &= mult(m, exp(m,n)) \end{aligned}$$

Funções Numéricas

- Exemplos:
  - Predecessor:

$$\begin{aligned} pred(0) &= 0 \\ pred(n+1) &= n \end{aligned}$$

  - Subtração não negativa ($m - n$ = max($m - n$,0))

$$\begin{aligned} sub(m,0) &= m \\ sub(m,n+1) &= pred(sub(m,n)) \end{aligned}$$

Funções Numéricas

- Pode-se definir um *Predicado recursivo primitivo* como sendo uma função recursiva primitiva que assume os valores 0 e 1
  - Exemplo:
    - is_zero: vale 1 se $n = 0$ e 0 se $n > 0$

$$\begin{aligned} is\_zero(0) &= 1 \\ is\_zero(m+1) &= 0 \end{aligned}$$

    - is_one:

$$\begin{aligned} is\_one(0) &= 0 \\ is\_one(n+1) &= is\_zero(n) \end{aligned}$$

Funções Numéricas

- Exemplos:
  - greater-than-or-equal:

$$geq(m,n) \quad = is\_zero(sub(n,m))$$

  - less-than:

$$le(m,n) \quad = (sub(1, geq(m,n))$$

Funções Numéricas

- Pode-se definir uma função *por casos*. Sejam f e g funções recursivas primitivas e p um predicado recursivo primitivo, todos k-ários, então:

$$f(n_1, \cdots, n_k) = \begin{cases} g(n_1, \cdots, n_k) & \text{se} \quad p(n_1, \cdots, n_k) \\ h(n_1, \cdots, n_k) & \text{caso contrário} \end{cases}$$

Funções Numéricas

- Exemplos:
  - resto da divisão:

$$
\begin{aligned}
mod(0,n) &= 0 \\
mod(m+1,n) &= \begin{cases} 0 & \text{se } equal(mod(m,n),pred(n)) \\ plus(mod(m,n),1) & \text{caso contrário} \end{cases}
\end{aligned}
$$

  - divisão inteira:

$$
\begin{aligned}
div(0,n) &= 0 \\
div(m+1,n) &= \begin{cases} plus(div(m,n),1) & \text{se } equal(mod(m,n),pred(n)) \\ div(m,n) & \text{caso contrário} \end{cases}
\end{aligned}
$$

Funções Numéricas

- Exercícios: Mostre que as seguintes funções são recursivas primitivas:
  1. fatorial(n) = n!
  2. gcd(m,n) - máximo divisor comum de m e n
  3. prime(n) - o predicado é 1 se n é um número primo

Funções Numéricas

- Partindo das funções básicas é possível mostrar que várias funções extremamente complexas são também recursivas primitivas
- Porém, seriam as funções recursivas primitivas exatamente a classe de funções que poderíamos considerar como *computáveis*??

Funções Numéricas

- Certamente, o conjunto das funções recursivas primitivas é enumerável
  - Cada função recursiva primitiva pode ser, a princípio, definida nos temos das funções básicas e, portanto, representada como uma cadeia sobre um alfabeto finito

Funções Numéricas

- Suponha que listemos todas as funções recursivas primitivas *unárias*, definidas para um argumento apenas, como cadeias, em ordem lexicográfica

$$f_0, f_1, f_2, f_3, \cdots$$

- Nessas condições, dado qualquer $n \geq 0$, podemos localizar f_n, a n-ésima função recursiva primitiva unária na lista e usá-la para computar o número $g(n) = f_n(n) + 1$
- $g(n)$ é computável, mas não é uma função recursiva primitiva, caso fosse, existiria algum $m \geq 0$ tal que $g(m) = f_m$, e em consequência, deveríamos ter $f_m(m) = f_m(m) + 1$, o que é absurdo

Funções Numéricas

- Qual é então o limite do que é computável?
  - a aplicação do operador de composição não parece aumentar a complexidade das funções básicas
  - já com a recursão, podemos ter a operação de adição e, de fato, todas as outras funções recursivas primitivas
  - que outra operação poderia ser aplicada às funções que pudesse extrapolar o conjunto das funções recursivas primitivas?

Funções Numéricas

- Operador de Busca Ilimitada (Recursão ou Iteração Ilimitada)
  - similar a um comando While
  - Extrapola a noção de recursão para além do limite onde a função é definida

$$f(x) = \text{o menor } y \text{ tal que } y + x = 10$$

para todo $x \geq 10$, $f(x)$ é indefinida, mas ainda assim é computável

Funções Numéricas

- Um operador de busca ilimitada pode ser reescrito como uma **Operação de Minimização**
  - Parcial
  - Total

Funções Numéricas

- Operação de Minimização Total
  - Seja $f : \mathcal{N}^{k+1} \mapsto \mathcal{N}$ uma função total
  - Seja $n = (n_1, n_2, \cdots n_k) \in \mathcal{N}^k$ fixo
  - A operação de Minimização definida sob f é escrita como:

$$\mu y(f(n, y) = 0)$$

e é especificada nos termos de um operador de busca ilimitada como:

$$\mu y(f(n, y) = 0) = \begin{cases} \text{o menor } y \in \mathcal{N} \text{ t.q } f(n, y) = 0 & : \text{se existe tal } y \\ \text{indefinida} & : \text{caso contrário} \end{cases}$$

Funções Numéricas

- Operação de Minimização Parcial
  - Seja $f : \mathcal{N}^{k+1} \mapsto \mathcal{N}$ uma função parcial
  - Seja $n = (n_1, n_2, \cdots n_k) \in \mathcal{N}^k$ fixo
  - A operação de Minimização definida sob f é escrita como:

$$\mu y(f(n,y) = 0)$$

e é especificada nos termos de um operador de busca ilimitada como:

$$\mu y(f(n,y) = 0) = \begin{cases} z & : f(n,z) = 0 \text{ e } f(n,y) \text{ é definida e} \\ & \forall y : 0 \leq y \leq z : f(n,y) \neq 0 \\ \text{indefinida} & : \text{caso contrário} \end{cases}$$

Funções Numéricas

- Operação de Minimização Parcial
  - Não é suficiente que exista um z tal que $f(n, z) = 0$
  - $f(n, y)$ precisa ser definida para todo $y \leq z$.
  - Caso contrário, ao tentar encontrar z por meio de recursão partindo de 0, poderiamos encontrar uma parte do domínio (algum y) onde f não é definida

Funções Numéricas

- Ao fechar o conjunto das funções recursivas primitivas sob as operações de composição, recursão primitiva e minimização, obtém-se a classe das **Funções Recursivas Parciais**
  - Uma Função Recursiva Total é uma Função Recursiva Parcial definida para toda a entrada
- Toda a função recursiva primitiva é uma função recursiva total, mas nem toda função recursiva total é recursiva primitiva
  - Exemplo: Função de Ackermann (como mostrado pelo argumento da diagonalização)

Funções Numéricas

- Apesar de a minimização ser bem definida, não há método óbvio para computá-la, o método óbvio da iteração ilimitada não é um algoritmo, pois pode, eventualmente não parar nunca.
- Uma função é dita *minimizável* se a iteração ilimitada sempre terminar
- Dizemos que uma função é μ-recursiva se ela puder ser obtida a partir das funções básicas e pela aplicação das operações de composição, recursão primitiva e minimização de funções minimizáveis
- Uma função $f : \mathcal{N}^k \mapsto \mathcal{N}$ é μ-recursiva se e somente se ela for recursiva, ou seja, computável por uma máquina de Turing