Complexidade 2007: folha prática nº 5

Funções recursivas e funções definidas por recursão primitiva (PR)

FCUP/DCC

Docente: Armando Matos

Notas. Nesta folha estudamos duas caracterizações das funções definidas por recursão primitiva ("primitivas recursivas"): a caracterização indutiva e linguagem FOR. Os exercícios 9 e 10 destinam-se à segunda semana de utilização desta folha.

Exercícios

1. Estudou a caracterização indutiva das funções "primitivas recursivas". Usando essa caracterização, mostre que as seguintes funções são PR. Ao resolver uma alínea, pode utilizar os resultados das alíneas anteriores.

   (a) • $f(x,y,z) = y$

   (b) $f(x,y) = x + y$

   (c) $f(x,y) = x - y$, definindo-se $x - y = 0$ quando $y > x$. Pode supor que a função $\mathrm{dec}(x) = x - 1$ (sendo $\mathrm{dec}(0) = 0$) já foi definida anteriormente.

   (d) • $f(x,y,z) = x + z$

   (e) $f(x,y) = x^2 + 2x$

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

   (g) • $f(x,y) = \max\{x,y\}$. **Sugestão.** $\max\{x,y\} = x + (y - x) = y + (x - y)$, definindo-se $x - y = 0$ quando $y > x$.

   (h) $f(x,y) = x^y$

2. Mostre que as seguintes funções são PRs, usando a caracterização pela linguagem FOR.

   (a) • $f(x,y) = x - y$, definindo-se $x - y = 0$ quando $y \geq x$.

   (b) • $f(x,y,z) = y$

   (c) • $f(x,y) = x^2 + 2x$

   (d) $f(x,y,z) = \begin{cases} g(y,z) & \text{se } x > 0 \\ h(y,z) & \text{se } x = 0 \end{cases}$
   onde $g(y,z)$ e $h(y,z)$ são por hipótese PRs (existindo pois programas FOR que as implementam).

   (e) • $f(x,y) = \max\{x,y\}$

   (f) $f(x,y) = x^y$

3. • Mostre que qualquer função PR é total; use a caracterização pela linguagem FOR..

4. • Escreva um "macro" para a linguagem FOR que troque os valores dos registos x_i e x_j.

5. Se na definição da linguagem FOR retirarmos a condição "todos os outros registos... contêm 0 no início", passa a haver menos (ou mais) funções caracterizáveis na linguagem? Porquê?

6. Escreva um interpretador da linguagem FOR.

7. Como se sabe, numa instrução "`FOR x {P}`" o registo x não pode ser mencionado (e como consequência não pode ser modificado) no programa `P`. Suponha agora que a linguagem é alterada de modo que `x` pode ser usado dentro de `P`, mas:

   – `P` é executado um número de vezes que é o conteúdo *inicial* de `x`.

   – Após a última execução de `P` é colocado 0 em `x`

   Mostre que a nova linguagem caracteriza também a classe das funções "primitivas recursivas" (é pois equivalente à linguagem `FOR`).

---

8. • *Nem num ponto!*
   Implemente em linguagem `WHILE` a função $f : \mathbb{N} \to \mathbb{N}$ não definida em nenhum ponto.

9. • *Esta encrava nos ímpares...*
   Implemente em linguagem `WHILE` a seguinte função

$$f(x) = \begin{cases} 1 & \text{se x é par} \\ \uparrow & \text{se x é ímpar} \end{cases}$$

10. • FOR⊆WHILE

    (a) Mostre que se $f(x)$ é computável por um programa `FOR`, também é computável por um programa `WHILE`. Fica assim provado que toda a função primitiva recursiva é recursiva.

    (b) Mostre que há funções recursivas que não são primitivas recursivas.

11. *Outra definição...*
    Como sabe, uma linguagem L diz-se r.e. (recursivamente enumerável) quando existe uma MT M_L tal que

$$M_L(x) = \begin{cases} 1 & \text{se } x \text{ pertence a } L \\ \uparrow & \text{caso contrário} \end{cases}$$

    Mostre que a seguinte definição de linguagem r.e. é equivalente: uma linguagem L diz-se r.e. quando existe uma MT M'_L tal que

$$M'_L(x) = \begin{cases} \text{pára} & \text{se } x \text{ pertence a } L \\ \uparrow & \text{caso contrário} \end{cases}$$

    (nesta definição quando $M'_L(x)$ pára, é indiferente o conteúdo da fita.)

(fim dos exercícios)

Caracterização indutiva das funções PR
Método indutivo

Funções $\mathbb{N}^m \to \mathbb{N}$ com $m \geq 1$.

Funções de base As seguintes funções são PR.

1. *Função 0*: $0 : \mathbb{N} \to \mathbb{N}$, $0(x) = 0$
2. *Função sucessor*: $s : \mathbb{N} \to \mathbb{N}$, , $s(x) = x + 1$
3. *Projecção*: as funções $\pi^n_m : \mathbb{N}^n \to \mathbb{N}$ onde $1 \leq m \leq n$, sendo $\pi^n_m(x_1, x_2, \ldots, x_n) = x_m$

Funções PR / Regras para a formação de novas funções PR

1. *Composição.* Dada a função PR $f : \mathbb{N}^n \to \mathbb{N}$ e as n funções PR $g_i : \mathbb{N}^m \to \mathbb{N}$ para $i = 1, \ldots, n$ então a função

   $$\mathrm{comp}^n_m(f, g_1, \cdots, g_n) : \mathbb{N}^m \to \mathbb{N}$$

   definida por

   $$\mathrm{comp}^n_m(\mathbf{x}) = f(g_1(\mathbf{x}), \cdots, g_n(\mathbf{x}))$$

   onde $\mathbf{x}$ designa o vector de variáveis $(x_1, \cdots, x_m)$, também é PR.

2. *Recursão primitiva.* Sejam 2 funções PR $f : \mathbb{N}^n \to \mathbb{N}$ e $g : \mathbb{N}^{n+2} \to \mathbb{N}$ onde $n \geq 1$. Então a função $h : \mathbb{N}^{n+1} \to \mathbb{N}$ definida abaixo (onde $\mathbf{x}$ representa $x_1, \cdots, x_m$) também é PR.

   $$\begin{cases} h(\mathbf{x}, 0) = f(\mathbf{x}) \\ h(\mathbf{x}, y+1) = g(\mathbf{x}, y, h(\mathbf{x}, y)) \end{cases}$$

   (Note que a função f definida por este processo tem pelo menos 2 argumentos).

Caracterização indutiva das funções PR
Método baseado na linguagem "FOR"

Registos: x_0, x_1, x_2,.... Por conveniência usamos também outras designações para os registos: x, y,... Cada registo contém um elemento de $\mathbb{N}$ (inteiro não negativo)

Instruções:

Instrução	Significado
$\mathtt{x_i}++$	$\mathtt{x_i} = \mathtt{x_i} + 1$
$\mathtt{x_i}--$	$\mathtt{x_i} = \mathtt{x_i} - 1$ (definindo-se 0-1=0)
`for` $\mathtt{x_i}$ {P}	P é executado $\mathtt{x_i}$ vezes; $\mathtt{x_i}$ não pode ser mencionado em P. No fim $\mathtt{x_i}$ fica com 0.

Programa: Sequência de 0 ou mais instruções (o programa pode pois ser vazio). Usamos por vezes ";" para separar instruções

Cálculo da função $f(x, y, ..., z)$ de aridade k (convenções): No início x em x_1, y em x_2, ... , z em x_k; todos os outros registos (incluindo x_0) contêm 0 no início. No final: resultado em x_0.

Definição. Dizemos que $f(x, ..., y)$ é "primitiva recursiva" (PR) se existe um programa FOR que a calcula.

Exemplo 1. `for x {}` equivale a: `x = 0 (atribuição)`

Exemplo 2.

```
for x {u++}                Nota: u, y são registos novos,
for y {}                   isto é, não usados em mais nenhum lado.
for u {y++;x++}            O programa equivale a:  y = x (atribuição)
```

Exemplo 3.

```
for x1 {
   for x2 {x0++; x3++}
   for x3 {x2++} }         calcula a função f(x,y)=xy+1
   x0++                    (logo xy+1 é PR)
```