Fundamentele Informatica 3

voorjaar 2014

http://www.liacs.nl/home/rvvliet/fi3/

Rudy van Vliet
kamer 124 Snellius, tel. 071-527 5777
rvvliet(at)liacs(dot)nl

(werk-)college 13, 6 mei 2014

10. Computable Functions
10.2. Quantification, Minimalization, and μ-Recursive Functions

Exercise 7.37.

Show that if there is TM T computing the function $f : \mathbb{N} \to \mathbb{N}$, then there is another one, T', whose tape alphabet is $\{1\}$.

Exercise.

How many Turing machines are there having n nonhalting states $q_0, q_1, \ldots, q_{n-1}$ and tape alphabet $\{0, 1\}$?

Exercise 10.2.

The *busy-beaver function* $b : \mathbb{N} \to \mathbb{N}$ is defined as follows.
The value $b(0)$ is 0.
For $n > 0$, there are only a finite number of Turing machines having n nonhalting states $q_0, q_1, \ldots, q_{n-1}$ and tape alphabet $\{0, 1\}$. Let $T_0, T_1, \ldots, T_m$ be the TMs of this type that eventually halt on input 1^n, and for each i, let n_i be the number of 1's that T_i leaves on its tape when it halts after processing the input string 1^n. The number $b(n)$ is defined to be the maximum of the numbers $n_0, n_1, \ldots, n_m$.

Show that the total function $b : \mathbb{N} \to \mathbb{N}$ is not computable.
Suggestion: Suppose for the sake of contradiction that T_b is a TM that computes b. Then we can assume without loss of generality that T_b has tape-alfabet $\{0, 1\}$.

A slide from lecture 12:

Definition 10.1. Initial Functions

The initial functions are the following:

1. *Constant* functions: For each $k \geq 0$ and each $a \geq 0$, the constant function $C_a^k : \mathbb{N}^k \to \mathbb{N}$ is defined by the formula

$$C_a^k(X) = a \quad \text{for every } X \in \mathbb{N}^k$$

2. The *successor* function $s : \mathbb{N} \to \mathbb{N}$ is defined by the formula

$$s(x) = x + 1$$

3. *Projection* functions: For each $k \geq 1$ and each i with $1 \leq i \leq k$, the projection function $p_i^k : \mathbb{N}^k \to \mathbb{N}$ is defined by the formula

$$p_i^k(x_1, x_2, \ldots, x_k) = x_i$$

A slide from lecture 12:

Definition 10.2. The Operations of Composition and Primitive Recursion

1. Suppose f is a partial function from $\mathbb{N}^k$ to $\mathbb{N}$, and for each i with $1 \leq i \leq k$, g_i is a partial function from $\mathbb{N}^m$ to $\mathbb{N}$.
   The partial function obtained from f and $g_1, g_2, \ldots, g_k$ by composition is the partial function h from $\mathbb{N}^m$ to $\mathbb{N}$ defined by the formula

$$h(X) = f(g_1(X), g_2(X), \ldots, g_k(X)) \text{ for every } X \in \mathbb{N}^m$$

A slide from lecture 12:

Definition 10.2. The Operations of Composition and Primitive Recursion (continued)

2. Suppose $n \geq 0$ and g and h are functions of n and $n+2$ variables, respectively. (By "a function of 0 variables," we mean simply a constant.)
   The function obtained from g and h by the operation of *primitive recursion* is the function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ defined by the formulas

$$\begin{aligned} f(X,0) &= g(X) \\ f(X,k+1) &= h(X,k,f(X,k)) \end{aligned}$$

   for every $X \in \mathbb{N}^n$ and every $k \geq 0$.

A slide from lecture 12:

n*-place predicate* P is function from $\mathbb{N}^n$ to $\{\text{true}, \text{false}\}$

characteristic function χ_P defined by

$$\chi_P(X) = \begin{cases} 1 & \text{if } P(X) \text{ is true} \\ 0 & \text{if } P(X) \text{ is false} \end{cases}$$

We say P is primitive recursive...

A slide from lecture 12:

Theorem 10.6.

The two-place predicates LT, EQ, GT, LE, GE, and NE are primitive recursive.
(LT stands for "less than," and the other five have similarly intuitive abbreviations.)
If P and Q are any primitive recursive n-place predicates, then $P \wedge Q$, $P \vee Q$ and $\neg P$ are primitive recursive.

Proof. . .

A slide from lecture 12:

Theorem 10.7.

Suppose $f_1, f_2, \ldots, f_k$ are primitive recursive functions from $\mathbb{N}^n$ to $\mathbb{N}$,
$P_1, P_2, \ldots, P_k$ are primitive recursive n-place predicates,
and for every $X \in \mathbb{N}^n$,
exactly one of the conditions $P_1(X), P_2(X), \ldots, P_k(X)$ is true.
Then the function $f : \mathbb{N}^n \to \mathbb{N}$ defined by

$$f(X) = \begin{cases} f_1(X) & \text{if } P_1(X) \text{ is true} \\ f_2(X) & \text{if } P_2(X) \text{ is true} \\ \quad \ldots & \\ f_k(X) & \text{if } P_k(X) \text{ is true} \end{cases}$$

is primitive recursive.

Proof. . .

Exercise.

Let $f : \mathbb{N}^{n+1} \to \mathbb{N}$ be a primitive recursive function.

Show that the predicate $P : \mathbb{N}^{n+1} \to \{\text{true}, \text{false}\}$ defined by

$$P(X, y) = \ (f(X, y) = 0)$$

is primitive recursive.

10.2. Quantification, Minimalization, and μ-Recursive Functions

A slide from lecture 12:

Theorem 10.4.

Every primitive recursive function is total and computable.

PR:	Turing-computable functions:
total and computable	not necessarily total

Unbounded quantification

$Sq(x, y) = \quad (y^2 = x)$

$H(x, y) = \quad T_u$ stopt na precies y stappen voor invoer s_x

Definition 10.9. Bounded Quantifications

Let P be an $(n+1)$-place predicate. The *bounded existential quantification* of P is the $(n+1)$-place predicate E_P defined by

$$E_P(X,k) = \text{(there exists } y \text{ with } 0 \leq y \leq k \text{ such that } P(X,y) \text{ is true)}$$

The *bounded universal quantification* of P is the $(n+1)$-place predicate A_P defined by

$$A_P(X,k) = \text{(for every } y \text{ satifying } 0 \leq y \leq k,\ P(X,y) \text{ is true)}$$

Theorem 10.10.

If P is a primitive recursive $(n+1)$-place predicate,
both the predicates E_P and A_P are also primitive recursive.

Proof. . .

A slide from lecture 12:

Theorem 10.4.

Every primitive recursive function is total and computable.

PR:
total and computable

Turing-computable functions:
not necessarily total

Definition 10.11. Bounded Minimalization

For an $(n+1)$-place predicate P, the *bounded minimalization* of P is the function $m_p : \mathbb{N}^{n+1} \to \mathbb{N}$ defined by

$$m_p(X,k) = \begin{cases} \text{min}\{y \mid \ 0 \leq y \leq k \text{ and } P(X,y)\} & \text{if this set is not empty} \\ k+1 & \text{otherwise} \end{cases}$$

Definition 10.11. Bounded Minimalization

For an $(n+1)$-place predicate P, the *bounded minimalization* of P is the function $m_P : \mathbb{N}^{n+1} \to \mathbb{N}$ defined by

$$m_P(X,k) = \begin{cases} \text{min}\{y \mid\ 0 \leq y \leq k \text{ and } P(X,y)\} & \text{if this set is not empty} \\ k+1 & \text{otherwise} \end{cases}$$

The symbol μ is often used for the minimalization operator, and we sometimes write

$$m_P(X,k) = \overset{k}{\mu}\, y[P(X,y)]$$

An important special case is that in which $P(X,y)$ is $(f(X,y) = 0)$, for some $f : \mathbb{N}^{n+1} \to \mathbb{N}$. In this case m_P is written m_f and referred to as the bounded minimalization of f.

Theorem 10.12.

If P is a primitive recursive $(n+1)$-place predicate, its bounded minimalization m_P is a primitive recursive function.

Proof. . .

Example 10.13. The nth Prime Number

$PrNo(0) = 2$
$PrNo(1) = 3$
$PrNo(2) = 5$

Example 10.13. The nth Prime Number

$PrNo(0) = 2$
$PrNo(1) = 3$
$PrNo(2) = 5$

$$\begin{aligned} Prime(n) \;=\; & (n \geq 2) \wedge \neg(\text{there exists } y \text{ such that} \\ & \qquad y \geq 2 \wedge y \leq n-1 \wedge Mod(n,y) = 0) \end{aligned}$$

Example 10.13. The nth Prime Number

Let

$$P(x, y) = (y > x \wedge \mathit{Prime}(y))$$

Then

$$\begin{aligned} \mathit{PrNo}(0) &= 2 \\ \mathit{PrNo}(k+1) &= m_P(\mathit{PrNo}(k), (\mathit{PrNo}(k))! + 1) \end{aligned}$$

is primitive recursive, with $h(x_1, x_2) = \ldots$

Exercise 10.19.

Show that each of the following functions is primitive recursive.

b. $f : \mathbb{N}^2 \to \mathbb{N}$ defined by $f(x, y) = \min\{x, y\}$

c. $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = \lfloor \sqrt{x} \rfloor$
(the largest natural number less than or equal to $\sqrt{x}$)

d. $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = \lfloor \log_2(x + 1) \rfloor$

Exercise 10.23.

In addition to the bounded minimalization of a predicate, we might define the bounded maximalization of a predicate P to be the function m^P defined by

$$m^P(X,k) = \begin{cases} \text{max}\{y \leq k \mid \; P(x,y) \text{ is true}\} & \text{if this set is not empty} \\ 0 & \text{otherwise} \end{cases}$$

a. Show m^P is primitive recursive by finding two primitive recursive functions from which it can be obtained by primitive recursion.

b. Show m^P is primitive recursive by using bounded minimalization.