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

college 15, 19 mei 2014

A slide from lecture 13:

Example 10.13. The nth Prime Number

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

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

A slide from lecture 13:

Example 10.13. The nth Prime Number

Let

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

Then

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

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

A slide from lecture 14:

Definition 10.15. μ-Recursive Functions

The set $\mathcal{M}$ of μ-recursive, or simply *recursive*, partial functions is defined as follows.

1. Every initial function is an element of $\mathcal{M}$.
2. Every function obtained from elements of $\mathcal{M}$ by composition or primitive recursion is an element of $\mathcal{M}$.
3. For every $n \geq 0$ and every total function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ in $\mathcal{M}$, the function $M_f : \mathbb{N}^n \to \mathbb{N}$ defined by
$$M_f(X) = \mu y[f(X,y) = 0]$$
is an element of $\mathcal{M}$.

A slide from lecture 14:

Theorem 10.16.

All μ-recursive partial functions are computable.

Proof. . .

A slide from lecture 14:

Definition 10.17.
The Gödel Number of a Sequence of Natural Numbers

For every $n \geq 1$ and every finite sequence $x_0, x_1, \ldots, x_{n-1}$ of n natural numbers, the *Gödel number* of the sequence is the number

$$gn(x_0, x_1, \ldots, x_{n-1}) = 2^{x_0} 3^{x_1} 5^{x_2} \ldots (PrNo(n-1))^{x_{n-1}}$$

where $PrNo(i)$ is the ith prime (Example 10.13).

A slide from lecture 14:

Example 10.18.

The Power to Which a Prime is Raised in the Factorization of x

Function *Exponent* : $\mathbb{N}^2 \to \mathbb{N}$ defined as follows:

$$Exponent(i,x) = \begin{cases} \text{the exp. of } PrNo(i) \text{ in } x\text{'s prime fact.} & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

Configuration of Turing machine determined by

- state
- position on tape
- tape contents

A slide from lecture 4:

Assumptions:

1. Names of the states are irrelevant.

2. Tape alphabet Γ of every Turing machine T is subset of infinite set $\mathcal{S} = \{a_1, a_2, a_3, \ldots\}$, where $a_1 = \Delta$.

A slide from lecture 4:

Definition 7.33. An Encoding Function

Assign numbers to each state:
$n(h_a) = 1$, $n(h_r) = 2$, $n(q_0) = 3$, $n(q) \geq 4$ for other $q \in Q$.

Assign numbers to each tape symbol:
$n(a_i) = i$.

Assign numbers to each tape head direction:
$n(R) = 1$, $n(L) = 2$, $n(S) = 3$.

Now different numbering

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be Turing machine

States:

h_a	h_r	q_0	...	.
0	1	2	...	s_T

with $s_T = \ldots$

Tape symbols:

Δ	...	.
0	...	ts_T

with $ts_T = \ldots$

Now different numbering

Let $T = (Q, \Sigma, \Gamma, q_0, \delta)$ be Turing machine

States:

h_a	h_r	q_0	...	.
0	1	2	...	s_T

with $s_T = |Q| + 1$

Tape symbols:

Δ	...	.
0	...	ts_T

with $ts_T = |\Gamma|$

$$\begin{aligned} \textit{tapenumber}(\Delta ab \Delta a \Delta) &= 2^0 3^1 5^2 7^0 11^1 13^0 \ldots \\ \textit{confignumber} &= 2^q 3^P 5^{\textit{tapenumber}} \end{aligned}$$

10.4. All Computable Functions are μ-Recursive

We must show that $f : \mathbb{N}^n \to \mathbb{N}$ defined by

$$f(X) = \textit{Result}_T(f_T(\textit{InitConfig}^{(n)}(X)))$$

is μ-recursive.

Step 1

The function $\textit{InitConfig}^{(n)} : \mathbb{N}^n \to \mathbb{N}$

Exercise 10.34.

Show using mathematical induction that if $tn^{(n)}(x_1, \ldots, x_n)$ is the tape number containing the string

$$\Delta 1^{x_1} \Delta 1^{x_2} \Delta \ldots \Delta 1^{x_n}$$

then $tn^{(n)} : \mathbb{N}^n \to \mathbb{N}$ is primitive recursive.

Use $nr(\Delta) = 0$ and $nr(1) = 1$.

Exercise 10.34.

Show using mathematical induction that if $tn^{(n)}(x_1, \ldots, x_n)$ is the tape number containing the string

$$\Delta 1^{x_1} \Delta 1^{x_2} \Delta \ldots \Delta 1^{x_n}$$

then $tn^{(n)} : \mathbb{N}^n \to \mathbb{N}$ is primitive recursive.
Suggestion: In the induction step, show that

$$tn^{(m+1)}(X, x_{m+1}) = tn^{(m)}(X) * \prod_{j=1}^{x_{m+1}} PrNo(m + \textstyle\sum_{i=1}^{m} x_i + j)$$

Use $nr(\Delta) = 0$ and $nr(1) = 1$.

Step 2

The predicate $IsConfig_T$ defined by

$$IsConfig_T(m) = (m \text{ is configuration number for } T)$$

Step 2 (continued)

The function $IsAccepting_T$ defined by

$$IsAccepting_T(m) = \begin{cases} 0 & \text{if } m \text{ represents accepting config. of } T \\ 1 & \text{otherwise} \end{cases}$$

Step 2 (continued)

The function $IsAccepting_T$ defined by

$$IsAccepting_T(m) = \begin{cases} 0 & \text{if } IsConfig_T(m) \wedge Exponent(0, m) = 0 \\ 1 & \text{otherwise} \end{cases}$$

Step 3

The function $Result_T \ldots$

Step 3

The function $Result_T$

$$Result_T(m) = \begin{cases} HighestPrime(Exponent(2, m)) & \text{if } IsConfig_T(m) \\ 0 & \text{otherwise} \end{cases}$$

Exercise 10.22.

Show that the function *HighestPrime* introduced in Section 10.4 is primitive recursive.

$$HighestPrime(k) = \begin{cases} 0 & \text{if } k \leq 1 \\ \max\{i \mid Exponent(i, k) > 0\} & \text{if } k \geq 2 \end{cases}$$

Step 4

$$\begin{aligned} State(m) &= Exponent(0, m) \\ Posn(m) &= Exponent(1, m) \\ TapeNumber(m) &= Exponent(2, m) \\ Symbol(m) &= Exponent(Posn(m), TapeNumber(m)) \end{aligned}$$

Step 4

$$\begin{aligned} \mathit{NewState}(m) &= \ldots \\ \mathit{NewSymbol}(m) &= \ldots \\ \mathit{NewPosn}(m) &= \ldots \\ \mathit{NewTapeNumber}(m) &= \ldots \end{aligned}$$

Exercise 10.35.

Show that the function *NewTapeNumber* discussed in Section 10.4 is primitive recursive.

Suggestion: Determine the prime factor of $\mathit{TapeNumber}(m)$ that may change by a move of the Turing machine, when the tape head is at position $\mathit{Posn}(m)$.

Step 5

The function $\mathit{Move}_T : \mathbb{N} \to \mathbb{N}$ defined by

$$\mathit{Move}_T(m) = \begin{cases} gn(\mathit{NewState}(m), \mathit{NewPosn}(m), \mathit{NewTapeNumber}(m)) \\ \quad \text{if } \mathit{IsConfig}_T(m) \\ 0 \quad \text{otherwise} \end{cases}$$

Step 6

The function $\mathit{Moves}_T : \mathbb{N}^2 \to \mathbb{N}$ defined by

$$\begin{aligned} \mathit{Moves}_T(m, 0) &= \begin{cases} m & \text{if } \mathit{IsConfig}_T(m) \\ 0 & \text{otherwise} \end{cases} \\ \mathit{Moves}_T(m, k+1) &= \begin{cases} \mathit{Move}_T(\mathit{Moves}_T(m, k)) & \text{if } \mathit{IsConfig}_T(m) \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

Step 7

The function $\mathit{NumberOfMovesToAccept}_T : \mathbb{N} \to \mathbb{N}$ defined by

$$\mathit{NumberOfMovesToAccept}_T(m) = \mu\, y[\mathit{IsAccepting}_T(\mathit{Moves}_T(m, y)) = 0]$$

Step 7

The function $\mathit{NumberOfMovesToAccept}_T : \mathbb{N} \to \mathbb{N}$ defined by

$$\mathit{NumberOfMovesToAccept}_T(m) = \mu\, y[\mathit{IsAccepting}_T(\mathit{Moves}_T(m, y)) = 0]$$

The function $f_T : \mathbb{N} \to \mathbb{N}$ defined by

$$f_T(m) = \mathit{Moves}_T(m, \mathit{NumberOfMovesToAccept}_T(m))$$

We must show that $f : \mathbb{N}^n \to \mathbb{N}$ defined by

$$f(X) = \mathit{Result}_T(f_T(\mathit{InitConfig}^{(n)}(X)))$$

is μ-recursive.

Theorem 10.20.

Every Turing computable partial function from $\mathbb{N}^n$ to $\mathbb{N}$ is μ-recursive.

The Rest of the Proof. . .

A slide from lecture 14:

Definition 10.15. μ-Recursive Functions

The set $\mathcal{M}$ of μ-recursive, or simply *recursive*, partial functions is defined as follows.

1. Every initial function is an element of $\mathcal{M}$.

2. Every function obtained from elements of $\mathcal{M}$ by composition or primitive recursion is an element of $\mathcal{M}$.

3. For every $n \geq 0$ and every total function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ in $\mathcal{M}$, the function $M_f : \mathbb{N}^n \to \mathbb{N}$ defined by
$$M_f(X) = \mu\, y[f(X, y) = 0]$$
is an element of $\mathcal{M}$.

10.5. Other Approaches to Computability

Let
- $G = (V, \Sigma, S, P)$ be unrestricted grammar
- f be partial function from Σ^* to Σ^*

Then G is said to compute f, if there are $A, B, C, D \in V$, such that for every x and y in Σ^*

$$f(x) = y \quad \text{if and only if} \quad AxB \Rightarrow^* CyD$$

Exercise.

Describe an unrestricted grammar that computes the function $f : \mathbb{N} \to \mathbb{N}$ defined by $f(x) = 2^x$.

Both the input x and the answer 2^x are unary numbers.

Computer programs vs. Turing machines

Computer programs vs. μ-recursive functions

En verder. . .

Tentamen: vrijdag 6 juni 2014, 14:00–17:00

Vragenuur. . . ?