

μ – *Recursive* functions and their
Turing-computability

Abhijit P. Pai

29th November, 2007

Outline

- ▶ Computability notions
- ▶ Primitive Recursive Functions
- ▶ μ – *Recursive* Functions
- ▶ μ – *Recursive* Functions are Turing-Computable

Computability

- ▶ Church-Turing thesis states that Turing Machines form a bound on what can be computed.
- ▶ The computational equivalence of the following have been shown:
 - ▶ Turing Machines
 - ▶ μ – *Recursive* functions
 - ▶ Post systems
 - ▶ Type 0 Grammars
 - ▶ λ – *Calculus*

Terms

- ▶ *Function*: A function from \mathbf{N}^k to \mathbf{N} for some $k \geq 0$; (called a k -place function).
- ▶ *Primitive Recursive function*: Class of functions defined by 3 types of initial functions and two combining rules.

Initial Functions

- ▶ The 0-place **zero** function ζ from \mathbf{N}^0 to \mathbf{N} such that $\zeta() = 0$
- ▶ Let $k \geq 1$ and $1 \leq i \leq k$. The i^{th} k -place **projection** function π_i^k from \mathbf{N}^k to \mathbf{N} such that $\pi_i^k(n_1, \dots, n_k) = n_i$ for any $n_1, \dots, n_k \in \mathbf{N}$
- ▶ The **successor** function σ from \mathbf{N} to \mathbf{N} such that $\sigma(n) = n + 1$ for each $n \in \mathbf{N}$
- ▶ An **initial** function is ζ, σ , or one of the π_i^k

Notation

- ▶ \bar{n} represents the sequence n_1, \dots, n_k where $k \geq 0$
- ▶ \bar{n} will also be used to represent the k-tuple (n_1, \dots, n_k)

Combining Rules - Composition

- ▶ Let $l > 0$ and $k \geq 0$
- ▶ Let g be an l -place function
- ▶ Let h_1, \dots, h_l be k -place functions
- ▶ Let f be the k -place function such that for every $\bar{n} \in \mathbf{N}^k$,
 $f(\bar{n}) = g(h_1(\bar{n}), \dots, h_l(\bar{n}))$
- ▶ Then f is said to be obtained from g, h_1, \dots, h_l by **composition**

Combining Rules - Primitive Recursion

- ▶ Let $k \geq 0$, g be a k -place function, h be a $(k+2)$ -place function.
- ▶ Let f be the $(k+1)$ -place function such that for every $\bar{n} \in \mathbf{N}^k$,
 $f(\bar{n}, 0) = g(\bar{n})$
- ▶ and for every $\bar{n} \in \mathbf{N}^k$ and $m \in \mathbf{N}$,
 $f(\bar{n}, m + 1) = h(\bar{n}, m, f(\bar{n}, m))$
- ▶ Then f is said to be obtained from g and h by **primitive recursion**

Primitive Recursive Functions

- ▶ A function is **primitive recursive** if it is an initial function or can be generated from the initial functions by some sequence of composition and primitive recursion.

Examples

- ▶ Constant functions

- ▶ $k = 0$: K_0^0 is ζ , $K_{j+1}^0() = \sigma(K_j^0())$ for each $j \geq 0$
- ▶ $k > 0$: $K_j^{k+1}(\bar{n}, 0) = K_j^k(\bar{n})$
- ▶ $k > 0$: $K_j^{k+1}(\bar{n}, m + 1) = \pi_{k+2}^{k+2}(\bar{n}, m, K_j^{k+1}(\bar{n}, m))$

Examples

- ▶ $plus(a, m) = a + m$
 - ▶ $plus(a, 0) = \pi_1^1(a)$
 - ▶ $plus(a, m + 1) = h(m, n, plus(a, m)), h = \sigma \circ \pi_3^3$
- ▶ $mult(a, m) = a * m$
 - ▶ $mult(a, 0) = K_0^1$
 - ▶ $mult(a, m + 1) = h(m, n, mult(a, m)), h = plus(\pi_1^3, \pi_3^3)$

Why primitive recursive functions couldn't possibly define computability

- ▶ Each primitive recursive function has a finite encoding
- ▶ Generate a list of all possible primitive recursive functions in lexicographic order A_1, A_2, A_3, \dots
- ▶ Let f_i be the p.r.f. defined by A_i
- ▶ Consider the 1-place function f such that for every $n \in \mathbf{N}$, $f(n) = f_n(n, \dots, n) + 1$
- ▶ f is computable. But f must be f_i for some i .
- ▶ When evaluated with argument i , f and f_i differ by 1.

Unbounded Minimization

- ▶ Let $k \geq 0$ and g be a $(k+1)$ -place function.
- ▶ The **unbounded minimization** of g is that k -place function f such that, for any $\bar{n} \in \mathbf{N}^k$,
 $f(\bar{n}) = \{ \text{The least } m \text{ such that } g(\bar{n}, m) = 0, \text{ if such } m \text{ exists}$
 $f(\bar{n}) = \{ 0 \text{ otherwise}$
- ▶ We write $f(\bar{n}) = \mu m [g(\bar{n}, m) = 0]$

Regular Functions

- ▶ A $(k+1)$ -place function g is **regular** iff, for every $\bar{n} \in \mathbf{N}^k$, there is an m such that $g(\bar{n}, m) = 0$.

μ – Recursive Functions

- ▶ A function is μ – recursive iff it can be obtained from ζ , π_i^k , and σ by:
 - ▶ Composition
 - ▶ Primitive Recursion
 - ▶ Application of Unbounded minimization to regular functions

Computing Turing Machine Model

- ▶ Input surrounded by blank on each side, written on leftmost squares of tape
- ▶ Head is initially at the blank just past the input string
- ▶ The TM M computes f if, for all $w \in \sigma^*$, if $f(w) = u$, then $(s, \#w\#) \vdash_M^* (h, \#u\#)$

Turing computability of μ – Recursive functions - Some TM's

- ▶ $R_{\#}$ searches for the first blank square to the right of the square currently scanned
- ▶ $L_{\#}$ searches for the first blank square to the left of the square currently scanned
- ▶ S_L shifts left - $\#w\underline{\#}$ to $w\underline{\#}$
- ▶ S_R shifts right - $\#w\underline{\#}$ to $\#\#w\underline{\#}$
- ▶ E (eraser) - $\#w\underline{\#}$ to $\underline{\#}$, where w can be empty
- ▶ B (backup) - $\#u\#w\underline{\#}$ to $\#w\underline{\#}$, u and w can be empty
- ▶ k-copier $C_k (k \geq 1)$ - $\#w_1\#w_2\#\dots\#w_k\underline{\#}$ to $\#w_1\#w_2\#\dots\#w_k\#w_1\underline{\#}$
- ▶ C_k^l - $\#w_1\#w_2\#\dots\#w_k\underline{\#}$ to $\#w_1\#w_2\#\dots\#w_k\#w_1\#w_2\#\dots\#w_1\underline{\#}$

The initial functions are Turing-computable

- ▶ ζ is computed by a TM that moves its head one square to the right and halts.
- ▶ Initial config: $(s, \underline{\#})$
- ▶ Final config: $(h, \#\underline{\#})$

The initial functions are Turing-computable

- ▶ π_i^k is computed by $E^{k-i} B^{i-1}$
- ▶ E^{k-i} - $\#w_1\#w_2\#\dots\#w_k\underline{\#}$ to $\#w_1\#w_2\#\dots\#w_i\underline{\#}$
- ▶ B^{i-1} transforms this to $\#w_i\underline{\#}$

The initial functions are Turing-computable

- ▶ σ - A TM which transforms $\#1^n\underline{\#}$ ($n \geq 0$) into $\#1^{n+1}\underline{\#}$

Composition is Turing-computable

- ▶ Composition

- ▶ $l > 0, k \geq 0$, g an l -place function, h_1, \dots, h_l k -place functions
- ▶ f , the k -place function such that for every $\bar{n} \in \mathbf{N}^k$,
 $f(\bar{n}) = g(h_1(\bar{n}), \dots, h_l(\bar{n}))$

- ▶ If $g, h_1 \dots h_l$ are Turing-computable, so is f .

- ▶ Let G, H_1, \dots, H_l be TM's that compute g, h_1, \dots, h_l .

- ▶ The function f is computed by the TM

$$C_k^k H_1 C_{k+1}^k H_2 \dots C_{k+l-1}^k H_l G B^k$$

- ▶ $C_k^k H_1 - \#w_1\#w_2\#\dots\#w_k\#\underline{\#}$ to
 $\#w_1\#w_2\#\dots\#w_k\#w_1\#w_2\#\dots\#w_k\#\underline{\#}$ to
 $\#w_1\#w_2\#\dots\#w_k\#u_1\#\underline{\#}$ where $u_i = 1^{h_i(\bar{n})}$

- ▶ After $C_k^k H_1 \dots C_{k+l-1}^k H_l, \#w_1\#w_2\#\dots\#w_k\#u_1\#u_2\#\dots\#u_l\#\underline{\#}$

Composition is Turing-computable

- ▶ G makes this into $\#w_1\#w_2\#\dots\#w_k\#\underline{\underline{v\#}}$ where v will be $1^{f(\bar{n})}$
- ▶ B^k makes this into $\#v\#$

Primitive recursion is Turing-computable

- ▶ Primitive Recursion
 - ▶ $k \geq 0$, g , a k -place function, h , a $(k+2)$ -place function
 - ▶ f , the $(k+1)$ -place function s.t. for every $\bar{n} \in \mathbf{N}^k$,
 $f(\bar{n}, 0) = g(\bar{n})$
 - ▶ and for every $\bar{n} \in \mathbf{N}^k$ and $m \in \mathbf{N}$,
 $f(\bar{n}, m + 1) = h(\bar{n}, m, f(\bar{n}, m))$
- ▶ If g and h are Turing-computable, so is f .
- ▶ Let G and H be TM's that compute g and h
- ▶ We write \bar{w} for $w_1 \# w_2 \# \dots w_k$

Primitive recursion is Turing-computable

- ▶ Add a left end marker $\$$ to input ($\$ \notin G, H$) using $(S_R L_{\#}^2)^{k+1} \$ R_{\#}^{k+1}$
- ▶ The tape is now $\$ \# \bar{w} \# I^m \#$
- ▶ Change tape to $\$ \# \bar{w} \# I^{m-1} \# \bar{w} \# I^{m-2} \# \dots \# \bar{w} \# I^1 \# \bar{w} \# I^0 \# \bar{w} \#$ using steps:
 - ▶ Copy I^m to tape 2
 - ▶ Copy \bar{w} to tape 3
 - ▶ Erase tape 1 to get $\underline{\$} \#$
 - ▶ Move right twice to get $\$ \# \#$
 - ▶ Repeat till tape 2 becomes zero
 - ▶ Decrement tape 2
 - ▶ Copy tape 3 to tape 1
 - ▶ Put $\#$ in tape 1
 - ▶ Copy tape 2 to tape 1
 - ▶ Put $\#$ in tape 1
 - ▶ Copy tape 3 to tape 1
 - ▶ Put $\#$ in tape 1

Primitive recursion is Turing-computable

- ▶ Now G and H compute successive values $f(\bar{n}, i)$ for $i = 0, \dots, m$ as:
- ▶ $\$ \# \bar{w} \# l^{m-1} \# \bar{w} \# l^{m-2} \# \dots \# \bar{w} \# l^1 \# \bar{w} \# l^0 \# \bar{w} \# \underline{\underline{\#}}$
- ▶ to $\$ \# \bar{w} \# l^{m-1} \# \bar{w} \# l^{m-2} \# \dots \# \bar{w} \# l^1 \# \bar{w} \# l^0 \# v_0 \# \underline{\underline{\#}}$
- ▶ to $\$ \# \bar{w} \# l^{m-1} \# \bar{w} \# l^{m-2} \# \dots \# \bar{w} \# l^1 \# v_1 \# \underline{\underline{\#}}$
- ▶ to $\$ \# \bar{w} \# l^{m-1} \# v_{m-1} \# \underline{\underline{\#}}$
- ▶ to $\$ \# v_m \# \underline{\underline{\#}}$
- ▶ Now remove the \$, and the output value is the value computed by f.

Unbounded Minimization is Turing-computable

- ▶ g , regular $(k+1)$ -place Turing-computable.
- ▶ If f is got from g by unbounded minimization, f is Turing-computable.
- ▶ Let G be the TM that computes g , we construct F that computes f , where for all $\bar{n} \in \mathbf{N}^k$, $f(\bar{n}) =$ the least m such that $g(\bar{n}, m) = 0$.
- ▶ g regular $\Rightarrow m$ exists for every \bar{n}
- ▶ F gets $\#w_1\#w_2\#\dots\#w_k\#\underline{\underline{}}$
- ▶ Change to $\#w_1\#w_2\#\dots\#w_k\#1^0\#\underline{\underline{}}$
- ▶ Create a copy of this.
- ▶ Apply G to the copy. If result is nonzero, add one more 1 at the end of the original and repeat.
- ▶ At some point, for $\#w_1\#w_2\#\dots\#w_k\#1^m\#\underline{\underline{}}$, G gives 0.
- ▶ Now, eliminate using B 1st k arguments and keep 1^m on tape.

Functions with strings as arguments

- ▶ Define a function known as the Gödel number $gn(w)$ of the string w as the number corresponding to the base $\beta = |\Delta| + 1$ encoding of the string.
- ▶ If $\Delta = a, b, c, \beta = 4$.
- ▶ We fix $d_1 = a, d_2 = b, d_3 = c$
- ▶ $gn(\epsilon) = 0$
- ▶ $gn(a) = gn(d_1) = 1$
- ▶ $gn(ba) = gn(d_2 d_1) = 4^1 \cdot 2 + 1 = 9$
- ▶ We add d_0 so that numbers such as β correspond to some string.
- ▶ Let $D = \epsilon \cup \Delta(\Delta \cup d_0)^*$
- ▶ gn^{-1} is a well defined function from \mathbf{N} to D .

Functions with strings as arguments

- ▶ $gn^{-1}(0) = \epsilon$
- ▶ $gn^{-1}(1) = d_1 = a$
- ▶ $gn^{-1}(3) = d_3 = c$
- ▶ $gn^{-1}(4) = d_1d_0 = ad_0$
- ▶ $gn^{-1}(5) = d_1d_1 = aa$
- ▶ ...

References

- ▶ **Elements of the Theory of Computation**
 - ▶ *Harry R. Lewis*
 - ▶ *Christos H. Papadimitriou*

Thank You

► Questions