Partial Recursive Functions

Problem solving in Automata, Languages and Automata

Ding-Zhu Du and Keri Ko

John Wieley and Sons, 2001

4.1 Turing Machine

4.5 Unrestricted Grammar
4.6 Primitive Resursive Function
4.7 Pairing Function and Gödel Numbering
4.8 Partial Recursive Function

4.1 Turing Machine

One-tape deterministic Turing machine $M = (Q, \Sigma, \Gamma, \delta, s, h, B)$,

1. Q *is a finite set of* ***states***,
2. Σ *is a finite set of* ***input*** *symbols*,
3. Γ *is a finite set of* ***tape*** *symbols (where* $\Sigma \subseteq \Gamma$*)*,
4. δ *is a* ***transition function(instructions)***,

$$\delta: (Q - \{h\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$$

5. $s \in Q$ *is an* ***initial*** *state*,
6. $h \in Q$ *is a* ***halting(final)*** *state, and*
7. $B \in \Gamma$ *is a* ***blank*** *tape symbol.*

Configuration *of TM*

$(\alpha, q, \beta) \in \Gamma^* \times Q \times \Gamma^*$.

1:$\beta \in \Gamma$ *is the* ***current tape*** *symbol.*

Blank symbols are assumed to be surrounding the tape string

$$(\alpha, q, \beta) = (B^*\alpha, q, \beta B^*).$$

Let (α, q, β) be a current configuration. Then

i) if $\delta(q, 1{:}\beta) = (p, X, R)$ then $(\alpha, q, \beta) \Rightarrow_M (\alpha X, p, \beta{:}|\beta|\text{-}1)$

note that if $\beta{=}\varepsilon$ and $\delta(q,B)=(p,X,R)$ then $(\alpha,q,\varepsilon) \Rightarrow_M (\alpha X,p,B)$

ii) if $\delta(q, 1{:}\beta) = (p, X, L)$ then $(\alpha, q, \beta) \Rightarrow_M (|\alpha|\text{-}1{:}\alpha, p, \alpha{:}1X\beta{:}|\beta|\text{-}1)$

note that if $\alpha{=}\varepsilon$ then $(\varepsilon, q, \beta) \Rightarrow_M (B, p, BX\beta{:}|\beta|\text{-}1)$

We may write $\Rightarrow$ instead of $\Rightarrow_M$ when it is clear.

We write reflexive-transitive closure of $\Rightarrow$ as $\Rightarrow^*$.

*A string $x \in \Sigma^*$ is **accepted** by a DTM M, if*

$(\varepsilon, s, x) \Rightarrow_M{}^*(\alpha, h, \beta)$ for some $\alpha, \beta \in \Sigma^*$.

*For configuration (α, q, β), if $\nexists\delta(q, 1{:}\beta)$, then M is said to be **hang on**.*

*A string $x \in \Sigma^*$ is **not accepted** by a DTM M, if*

*either M **hangs on** x or M **does not halt(loops forever)** on x.*

$L(M) = \{x \in \Sigma^* \mid (\varepsilon, s, x) \Rightarrow_M{}^*(\alpha, h, \beta)\}$

TM as a partial function

If $\exists$a DTM M .э. L = L(M). Then L is called ***Turing-accpetable****.*

Let f: $(\Sigma_1^*)^k \rightarrow \Sigma_2^*$ be a partial function.

We write $f(x_1, x_2, \ldots, x_k)\downarrow$ to denote f is ***defined*** *at $(x_1, x_2, \ldots, x_k)$ and*

$f(x_1, x_2, \ldots, x_k)\uparrow$ to denote f is ***not defined*** *at $(x_1, x_2, \ldots, x_k)$.*

A partial function f is called ***Turing-computable****, if $\exists$a DTM M .э.*

if $f(x_1, x_2, \ldots, x_k) = y$, then $(\varepsilon, s, x_1Bx_2B\ldots Bx_kB) \Rightarrow_M^* (\varepsilon, h, y)$; and

if $f(x_1, x_2, \ldots, x_k)\uparrow$, then $(\varepsilon, s, x_1Bx_2B\ldots Bx_kB)$

does not halt(loops forever)*.*

f: $(\Sigma_1^*)^k \rightarrow \Sigma_2^*$ *is called (**total**) function,*

if $^{\forall}(x_1, x_2, \ldots, x_k) \in (\Sigma_1^*)^k$, $f(x_1, x_2, \ldots, x_k)\downarrow$.

*f is called **Turing-decidable**, if f is a **total Turing-computable** function.*

*A language L is **Turing-decidable**, if its characteristics function*

$\chi_L(x) = 1$, *if* $x \in L$,

0, otherwise.

$^{\forall}x \in \Sigma^*$, $^{\exists}$*a DTM M .э. if* $x \in L$, *DTM halts; if* $x \notin L$, *it hangs on.*

***Turing-decidable**(type 1) is a **proper subclass** of **Turing-acceptable**(type 0).*

4.5 Unristricted grammar

G = (V, Σ, P, S) is a unristricted grammat if

$\alpha \rightarrow \beta \in P$ *with* $\alpha \in (V \cup \Sigma)^+$ *and* $\beta \in (V \cup \Sigma)^*$.

Example 4.15 $L(G) = \{a^n b^n c^n \mid n \geq 0\}$.

$S \rightarrow aSBC \mid \varepsilon$, $CB \rightarrow BC$,

$aB \rightarrow ab$, $bB \rightarrow bb$,

$bC \rightarrow bc$, $cC \rightarrow cc$.

$S \Rightarrow^{n+1} a^n(BC)^n \Rightarrow^{n(n-1)/2} a^n B^n C^n \Rightarrow^{2n} a^n b^n c^n$.

Example 4.16 *Find a grammar G .ə. L(G) = {a^{2^n} | n ≥ 0}.*

$S \to [Ra] \mid a,$	$Ra \to aaR,$	*doubles a, move R right*
$R] \to L] \mid L_h,$		*change R] to L] or L_h(terminate)*
$aL \to La,$	$[L \to [R,$	*move L left and change it to [R*
$aL_h \to L_h a,$	$[L_h \to \varepsilon.$	*move L_h left and remove [.*

basis *i)* $n=0,\ S \Rightarrow_{S\to a} a = a^{2^0}.$

ii) $n = 1,\ S \Rightarrow_{S\to[Ra]} [Ra] \Rightarrow_{Ra\to aaR} [aaR] \Rightarrow_{RL_h]\to L_h} [aaL_h$

$$\Rightarrow_{aL_h\to L_h a} [aL_h a \Rightarrow_{aL_h\to L_h a} [L_h aa \Rightarrow_{[L_h\to\varepsilon} aa = a^{2^1}.$$

induction $S \Rightarrow_{S\to[Ra]} [Ra] \Rightarrow_{Ra\to aaR}^{*} [Ra^{2^k}] \Rightarrow_{Ra\to aaR}^{2^k} [a^{2^{k+1}}R]$

$$\Rightarrow_{R]\to L]} [a^{2^{k+1}}L] \Rightarrow_{aL\to La}^{2^{k+1}} [La^{2^{k+1}}] \Rightarrow_{[L\to R} [Ra^{2^{k+1}}] \Rightarrow^{*} a^{2^{k+1}}.$$

Nonterminal R ***cannot*** *move to left until it meets], R doubles a's.*

Example 4.17 *Find a grammar G .ə. L(G) = {ww| w ∈ {a, b}*.*

$S \to T\ ]$, $T \to a\ T\ A \mid b\ T\ B \mid [\ R$,

$R\ A \to A\ R$, $R\ B \to B\ R$,

$A\ R\ ] \to L_a\ ]$, $B\ R\ ] \to L_b\ ]$,

$A\ L_a \to L_a\ A$, $B\ L_b \to L_b\ B$,

$[\ L_a \to a\ [\ R$, $[\ L_b \to a\ [\ R$, $[\ R\ ] \to \varepsilon$.

$S \Rightarrow T\ ] \Rightarrow^4 abaaTAABA] \Rightarrow abaa[RAABA] \Rightarrow^4 abaa[AABAR]$

$\Rightarrow abaa[AABL_a] \Rightarrow^3 abaa[L_aAAB] \Rightarrow abaaa[RAAB] \Rightarrow^3 abaaa[AABR]$

$\Rightarrow abaaa[AAL_b] \Rightarrow^2 abaaa[L_bAA] \Rightarrow abaaab[RAA] \Rightarrow^2 abaaab[AAR]$

$\Rightarrow^* abaaaba[AR] \Rightarrow^* abaaabaa[R] \Rightarrow abaaabaa.$

$S \Rightarrow T\ ] \Rightarrow^n wTw^{\sim R}] \Rightarrow w[Rw^{\sim R}]$ *where* $w^{\sim} = w$*(replace a, b to A, B resp.)*

$[Rw^{\sim R}A] \Rightarrow^n [w^{\sim R}AR] \Rightarrow_{AR]\to L_a} [w^{\sim R}L_a] \Rightarrow^{n-1} [L_aw^{\sim R}] \Rightarrow_{[L_a\to a[R} a[Rw^{\sim R}].$

Example 4.18 *Find a grammar G .ə. L(G) =* $\{a^n b^m c^{nm} \mid n, m \geq 0\}$.

$S \to [\ A\ ]$, $A \to a\ A \mid B$, $B \to b\ B \mid L$,

$a\ L \to L\ a$, $b\ L \to L\ b$, $\overline{b}\ L \to L\ \overline{b}$,

$[\ L\ a \to a\ [\ R_b$, $[\ L\ b \to b\ R$, $[\ L\ ] \to \varepsilon$,

$R_b\ a \to a\ R_b$, $R_b\ b \to \overline{b}\ R_c$, $R_b\ ] \to L\ ]$,

$R_c\ b \to b\ R_c$, $R_c\ ] \to L_b\ ]\ c$,

$b\ L_b \to L_b\ b$, $\overline{b}\ L_b \to \overline{b}\ R_b$,

$R\ b \to b\ R$, $R\ ] \to \varepsilon$.

$S \Rightarrow [\ A\ ] \Rightarrow^n [a^n A] \Rightarrow [a^n B] \Rightarrow^m [a^n b^m B] \Rightarrow [a^n b^m L] \Rightarrow^{n+m} [L a^n b^m]$

$a^k[L a^{n-k} b^m] c^{km} \Rightarrow a^{k+1}[R_b a^{n-k-1} b^m] c^{km} \Rightarrow^* a^{k+1}[a^{n-k-1} R_b b b^{m-1}] c^{km}$

$\Rightarrow a^{k+1}[a^{n-k-1} \overline{b} R_c b^{m-1}] c^{km} \Rightarrow^* a^{k+1}[a^{n-k-1} \overline{b} b^{m-1} R_c] c^{km}$

$\Rightarrow a^{k+1}[a^{n-k-1} \overline{b} b^{m-1} L_b] c c^{km} \Rightarrow^* a^{k+1}[a^{n-k-1} \overline{b} L_b b^{m-1}] c^{km+1}$

$\Rightarrow a^{k+1}[a^{n-k-1} \overline{b} R_b b^{m-1}] c^{km+1} \Rightarrow^* \ldots \Rightarrow^* a^{k+1}[a^{n-k-1} \overline{b}^2 R_b b^{m-2}] c^{km+2}$

$\Rightarrow^* a^{k+1}[a^{n-k-1} \overline{b}^m R_b] c^{km+m} \Rightarrow a^{k+1}[a^{n-k-1} \overline{b}^m L] c^{(k+1)m}$

$\Rightarrow^* a^{k+1}[L a^{n-k-1} b^m] c^{(k+1)m}$.

Theorem 4.19 *For any one-tape DTM* $M = (Q, \Sigma, \Gamma, \delta, s)$,

$\exists$ *a grammar* $G_M = (Q \cup (\Gamma - \Sigma) \cup \{[,]\}, \Sigma, P, S)$.ə.

$\forall$ *configurations* $(\alpha, q, X\beta)$ *and* $(\alpha', p, Y\beta')$

where $p, q \in Q, X, Y \in \Gamma, \alpha, \beta, \alpha', \beta' \in \Gamma^*$,

$(\alpha, q, X\beta) \Rightarrow_M^* (\alpha', p, Y\beta')$ *if and only if* $[\alpha qX\beta] \Rightarrow_{G_M}^* [\alpha' pY\beta']$.

Proof

i) $\forall \delta(q, X)=(p,Y, L)$ *where* $Y \neq B$, $\quad \forall Z \in \Gamma, ZqX \rightarrow pZY \in P$.

ii) $\forall \delta(q, X) = (p, B, L)$, $\quad \forall Z,V \in \Gamma, ZqXV \rightarrow pZBV \in P$,

$\forall Z \in \Gamma, ZqX] \rightarrow pZ] \in P$.

iii) $\forall \delta(q, X) = (p, Y, R)$, $\quad \forall Z \in \Gamma, qXZ \rightarrow YpZ, qX] \rightarrow Yp] \in P$.

$[\alpha qX\beta] \Rightarrow_{G_M} [\alpha' pY\beta']$ *if and only if* $(\alpha, q, X\beta) \Rightarrow_M (\alpha', p, Y\beta')$.

$[\alpha qX\beta] \Rightarrow_{G_M}^* [\alpha' pY\beta']$ *if and only if* $(\alpha, q, X\beta) \Rightarrow_M^* (\alpha', p, Y\beta')$.

Theorem 4.20 *If a language $L \subseteq \Sigma^*$ is* ***Turing-acceptable****, then $L = L(G)$ for some grammar G.*

proof *Assume $L = L(M)$ TM $^{\exists}M$.ə. M always halts with empty output.*
Thus the configuration of M is always (B, h, B).
The rules of G includes the rules of G_M, plus

$S \to [\ B\ h\ B\ ]$, $\quad s\ B\ ] \to L$,

$a\ L \to L\ a$, $\quad [\ B\ L \to \varepsilon$. *for* $^{\forall}a \in \Sigma$, *where* $L \in V$.

$^{\forall}x \in \Sigma^*$.ə. $x \in L(M)$, $(\varepsilon, s, x) \Rightarrow_{G_M}^{*} (\varepsilon, h, \varepsilon)$

$S \Rightarrow_G [\ B\ h\ B\ ] \Rightarrow_G^{*} [\ B\ s\ x\ B\ ] \Rightarrow_G [\ B\ x\ s\ L \Rightarrow_G^{*} [\ B\ L\ x\ \Rightarrow_G x$.

4.6 Primitive Recursive Functions

Three initial functions *and* two operations on functions

Initial Functions:

Zero: $\zeta: N \to \{0\}$.э. $^{\forall} n \in N: \zeta(n) \cong 0$.

Successor: $\sigma: N \to N$.э. $^{\forall} n \in N: \sigma(n) \cong n+1$.

Projection: $^{\forall} k \in N,\ 1 \leq ^{\forall} i \leq n: \pi_i^k: N^k \to N$.э. $\pi_i^k(n_1, \ldots, n_i, \ldots, n_k) \cong n_i$.

Compositon: *Given* $^{\forall} k \in N: g: N^k \multimap N,\ 1 \leq ^{\forall} i \leq k: h_i: N^m \multimap N$; *define*

$f: N^k \multimap N$.э. $^{\forall} X \in N^m: f(X) \cong g(h_1(X), \ldots, h_k(X)) \cong [g^{\circ}(h_1, \ldots, h_k)]\ (X)$

Primitive recursion: *Let* $k \geq 0$.

Given $^{\forall} k \in N: g: N^k \multimap N,\ h: N^{k+2} \multimap N$; *define*

$f: N^{k+1} \multimap N$.э. $^{\forall} X \in N^k$:

$f(X, 0) \cong g(X)$, **basis**

$f(X, \sigma(m)) \cong h(X, m, f(X, m)),\ m \geq 0$. **(primitive) recursion**

Definition** of **primitive recursive functions

*(1) Initial functions are **primitive recursive**.*

(2) If $g: N^k \to N$, $h_1, \ldots, h_k: N^m \to N$ are primitive recursive,
***composition** $g \circ (h_1, \ldots, h_k)$ is also **primitive recursive**.*

(3) If $g: N^k \to N$ and $h: N^{k+2} \to N$ are primittive recursive,
*the function obtained from g and h by **primitive recursion***
*is also **primitive recursive**.*

*(4) No other function is **primitive recursive**.*

Primitive recursive function
*obtained from the **initial functions** by **finite mumber** of applications*
*of **composition** and **primitive recursive** operations.*

Constants *are primitive recursive*

Example 4.23 ***Constants*** $^{\forall}k \in 1, {}^{\forall}a \geq 0$: K_a^k: $N^k \rightharpoonup N$

.ə. $^{\forall}X \in N^k$: $K_a^k(X) \cong a$ *are primitive recursive.*

$K_j^k = \sigma(\sigma\ldots(\sigma(\zeta(\pi_1^k(n_1, \ldots, n_k)))\ldots)$

$= [\sigma^{(j)} \circ \zeta \circ \pi_1^k](n_1, \ldots, n_k)$

Arithmetic functions *are primitive recursive*

Example 4.21 $add(m, n) \cong m + n$ *is primitive recursive*

$add(m, 0) = \pi_1^1(m) = m,$

$add(m, n+1) = [\sigma \circ \pi_3^3](m, n, add(m, n)) = \sigma(add(m, n)).$

Example 4.22 $mult(m, n) \cong m \cdot n$ *is primitive recursive*

$mult(m, 0) = \zeta(m) = 0.$

$multi(m, n+1) = [add \circ (\pi_1^3, \pi_3^3)](m, n, mult(m, n))$

$= add(m, multi(m, n))$

Example 4.24 minus(m, n) ≅ m ∸ n = 0, if m ≤ n; m - n, if m > n is p.r.

$minus(m, 0) = \pi_1^1(m)$

$minus(m, n+1) = [pred \circ \pi_3^3](m, n, minus(m, n)))$

$= pred(minus(m, n))$

where $pred(0) = 0$

$pred(m+1) = pred(m) = \pi_1^2(m, pred(m))$

Example 4.25 $g(m, n) \cong f^{(n)}(m) = f(f(\ldots(f(m))\ldots))$ *is primitive recursive*

$g(m, 0) = \pi_1^1(m)$

$g(m, n+1) = f(g(m, n))$

Boolean functions *and* ***if-then-else*** *are primitive recursive*

Example 4.26 *Following functions are primitive rcursive.*

(a) neg(x) ≅ 0 if x ≥ 1; 1 if x = 0.

neg(x) = minus(1, x).

(b) and(x, y) ≅ 1 if x ≥ 1 ∧ y ≥ 1; 0 otherwise.

and(x) = neg(neg(mult(x, y))).

(c) or(x, y) ≅ 1 if x ≥ 1 ∨ y ≥ 1; 0 otherwise.

and(x) = neg(and(neg(x), neg(y))).

(d) if-then-else(x, y, z) ≅ y if x ≥ 1; z othewise.

if-then-else(x, y, z) = add(mult(neg(neg(x)), y), mult(neg(x), z)).

We write "x and y" for add(x, y), "x or y" for or(x, y), and

*"**if** x **then** y **else** z **fi**" for if-then-else(x, y, z).*

Example 4.27 *Following functions are primitive rcursive.*

(a) eq(x, y) = 1 if x = y, 0 if x ≠ y.

eq(x, y) = neg(add(minus(x, y), minus(y, x))).

(b) gr(x, y) = 1 if x > y, 0 if x ≤ y.

gr(x, y) = neg(neg(minus(x, y))).

(c) geq(x, y) = 1 if x ≥ y, 0 if x < y.

geq(x, y) = gr(x, y) or eq(x, y).

(d) ls(x, y) = 1 if x < y, 0 if x ≥ y.

ls(x, y) = neg(geq(x, y)).

(d) leq(x, y) = 1 if x ≤ y, 0 if x > y.

geq(x, y) = neg(gr(x, y)).

Example 4.28 $max^k(n_1, \ldots, n_k) \cong max\{n_1, \ldots, n_k\}$ *is p.r. for* $k \geq 1$.

Solution $max^1(n_1) = \pi_1^1(n_1)$

$max^k(n_1,\ldots, _k) =$ ***if*** $n_k > max^{k-1}(n_1,\ldots,n_{k-1})$ ***then*** n_k ***else*** $max^{k-1}(n_1,\ldots,n_{k-1})$

$R: (\{0, 1\}^*)^k \to \{0, 1\}$ is a ***Boolean function*** *or* ***predicate****.*

We use 1 to represent TRUE and 0 to represent FALSE

If $R(n_1, \ldots, n_k)$ is a ***predicate****, $R(n_1, \ldots, n_k)$ means $[R(n_1, \ldots, n_k) = 1]$*

We open write ***formula*** *instaed of the name of the* ***predicate***

$g(m) \cong 1$ if $(\exists n)[n^2 = m]$; 0 otherwise.

We write $(\exists n)[n^2 = m]$ instead of $g(m)$.

Theorem 4.29 *Let $f: N^{k+1} \to \{0, 1\}$ be a* ***primitive recursive predicate****.*
*Then following t****hree operations*** *on f are also* ***primitive recursive****.*

(a) ***bounded universal quantifier***

$g(n_1, \ldots, n_k, m) \cong (\forall i)_{i \le m} f(n_1, \ldots, n_k, i)$.

(b) ***bounded existential quantifier***

$h(n_1, \ldots, n_k, m) \cong (\exists i)_{i \le m} f(n_1, \ldots, n_k, i)$.

(c) ***bounded minimization***

$t(n_1, \ldots, n_k, m) \cong (\min i)_{i \le m} f(n_1, \ldots, n_k, i)$ *if* $(\exists i)_{i \le m} f(n_1, \ldots, n_k, i)$,

$\cong 0$ *otherwise.*

Proof

(a) $g(n_1, \ldots, n_k, 0) = f(n_1, \ldots, n_k, 0)$

$g(n_1, \ldots, n_k, m+1) = [f(n_1, \ldots, n_k, m+1)$ *and* $g(n_1, \ldots, n_k, m)]$

(b) $h(n_1, \ldots, n_k, 0) = f(n_1, \ldots, n_k, 0)$

$h(n_1, \ldots, n_k, m+1) = [f(n_1, \ldots, n_k, m+1)$ *or* $h(n_1, \ldots, n_k, m)]$

(c) $t(n_1, \ldots, n_k, 0) = f(n_1, \ldots, n_k, 0)$

$t(n_1, \ldots, n_k, m+1) = [$***if*** $(\exists i)_{i \le m} f(n_1, \ldots, n_k, i)$ ***then*** $t(n_1, \ldots, n_k, m)$

else if $f(n_1, \ldots, n_k, m+1)$ ***then*** $m+1$ ***else*** 0 ***fi fi***$]$

Example 4.30 *The following functions are primitive recursive.*

(a) quot(m, n) ≅ $\lfloor m/n \rfloor$ if n > 0; 0 otherwise.

quot(m, n) = ***if*** *n > 0* ***then*** *$(\min i)_{i \le m}$ [(i+1) · n > m]* ***else*** *0.*

(b) mod(m, n) ≅ m - $\lfloor m/n \rfloor$ · n if n > 0; 0 otherwise.

quot(m, n) = ***if*** *n > 0* ***then*** *$(\min i)_{i \le m}$ [quot(m, n) · n + i = m]* ***else*** *0.*

(c) prime(n) ≅ 1 if n is prime; 0 otherwise.

prime(n) = (n ≥ 2) and $(\forall j)_{2 \le j \le n \dot{-} 1}$ [mod(n, j) > 0]; or

prime(n) = prime'(n, pred(n))

where prime'(n, k) = (n ≥ 2) and $(\forall j)_{j \le k}$ [j ≤ 1 or mod(n, j) > 0].

∴ $(\forall i)_{l(n) \le i \le u(n)}$, $(\exists i)_{l(n) \le i \le u(n)}$, and $(\min i)_{l(n) \le i \le u(n)}$

are ***primitive recursive****, if l(n) and u(n) are primitive recursive.*

Example 4.31 *f(0) ≅ 1*

f(n) ≅ nth digit to the right of the decimal point of srqt(2).

Solution

Let g(n) be the integer m whose decimal expansion is equal to f(0)…f(n)

e.g. g(3) = 1414. Then

$g(n) = (\min k)_{k \le 10^{n+1}}[(k+1)^2 > 2 \cdot 10^{2n}$.

f(n) = mod(g(n), 10).

4.7 Pairing Function and Gödel Numbering

To show Fibonacci function F: N → N

F(0) = 0, F(1) = 1,

F(n+2) = F(n) + F(n+1) is recursive primitive

Pairing function f: N^2 → N

*i) (**Bijectivity**) f is one-to-one and onto.*

*ii) (**Primitive recursiveness**) f is primitive recursive.*

*iii) (**Monotonicity**) f(i, j) < f(i+1, j), f(i, j) < f(i, j+1) $^{\forall}i, j \in N$.*

*Suppose f is a **pairing function**.*

l_f, r_f: N → N .ə. f(l_f(n), r_f(n)) ≅ n are

***well defined** and **primitive recursive**.*

$l_f(n) = (\min i)_{i \le m} (\exists j)_{j \le n} [f(i, j) = n]$,

$r_f(n) = (\min i)_{j \le n} [f(l_f(n), j) = n]$.

Example 4.32 *$\pi(i, j) = (i+j)(i+j+1)/2 + j$ is a* ***pairing*** *function.*
Solution *1-1 onto, primitive recursive, monotonic.*

We write $\langle i, j\rangle = \pi(i, j)$, $l(n) = l_\pi(n)$, $r(n) = r_\pi(n)$.

$\therefore \langle l(n), r(n)\rangle = n$.

Example 4.33 *Fibonacci function F is primitive recursive.*

$F(0) = 0$
$F(1) = 1$
$F(n+2) = F(n) + F(n+1)$

Solution $G(n) = \langle F(n), F(n+1)\rangle$

$G(0) = \langle 0, 1\rangle$
$G(n+1) = \langle F(n+1), F(n+2)\rangle = \langle F(n+1), F(n) + F(n+1)\rangle$
$= \langle r(G(n)), l(G(n)) + r(G(n+1))\rangle$
$\therefore F(n) = l(G(n))$ *is* ***primitive recursive***.

$f: \cup_{k\geq 1} N^k \rightarrow N$ *is a Gödel numbering(of integer sequences)*

*i) (**Bijectivity**) f is one-to-one and onto.*

*ii) (**Primitive recursiveness**)* $f|_{N^k}$ *is primitive recursive* $^{\forall}k \geq 1$.

where $f|_{N^k}$ *is is the function restricted to the domain* N^k.

*iii) (**Monotonicity**)* $1 \leq {}^{\forall}i \leq k$

$f(n_1, \ldots, n_{i-1}, n_i, n_{i+1}, \ldots, n_k) < f(n_1, \ldots, n_{i-1}, n_i+1, n_{i+1}, \ldots, n_k)$.

Example 4.34

$\tau(n_1, \ldots, n_{k-1}, n_k) = \langle k \dot{-} 1, \langle n_1, \langle \ldots \langle n_{k-1}, n_k \rangle \ldots \rangle \rangle \rangle$ *is a Gödel numbering*

Solution

Suppose $\tau(n_1, \ldots, n_k) = \tau(m_1, \ldots, m_s)$

$\langle k \dot{-} 1, \langle n_1, \langle \ldots \langle n_{k-1}, n_k \rangle \ldots \rangle \rangle \rangle = \langle s \dot{-} 1, \langle n_1, \langle \ldots \langle n_{s-1}, n_s \rangle \ldots \rangle \rangle \rangle$

$k = s$, *since* π *is one-to-one.*

$\tau|_{N^k}$ *is one-to-one*

If $k=1$, $\tau(n) = \langle 0, n\rangle$ *is one-to-one*

If $k > 1$, $\tau(n_1, n_2, \ldots, n_k) = \langle k \dot{-} 1, \langle n_1, r(\tau(n_2, \ldots, n_k))\rangle\rangle$

$\tau|_{N^{k-1}}$ *is one-to-one and* τ *is one-to-one.*

τ *is onto.*

$^{\forall}n \in N, {}^{\exists}(n_1, n_2, \ldots, n_k) \ni \tau(n_1, n_2, \ldots, n_k) = n.$

$k = l(n) + 1.$

$1 \le {}^{\forall}i \le k, item(n, i) \cong l(r^{(i)}(n))$ *if* $1 \le {}^{\forall}i \le k-1$; $r^{(k)}(n)$ *if* $i = k.$

$^{\forall}n \in N, {}^{\exists 1}(item(n, 1), \ldots, item(n, k)) \ni \tau(item(n,1), \ldots, item(n,k)) = n.$

$\therefore$ τ *is onto.*

Note that $\tau(n_1, \ldots, n_{k-1}, n_k) = \langle k \dot{-} 1, \langle n_1, \langle \ldots \langle n_{k-1}, n_k \rangle \ldots \rangle \rangle \rangle$

We write $\langle n_1, n_2, \ldots, n_k \rangle \cong r(\tau(n_1, n_2, \ldots, n_k)) = \langle n_1, \langle \ldots \langle n_{k-1}, n_k \rangle \ldots \rangle \rangle$.

We write $[n_1, n_2, \ldots, n_k] \cong \tau(n_1, n_2, \ldots, n_k)$. *We write* $size(n) \cong l(n) + 1$.

$\therefore [item(n, 1), \ldots, item(n, size(n))] = n$.

Example 4.35

(a) $list(m, 0) \cong 0$ and $list(m, k) = [m, m, \ldots, m]$, if $k \geq 1$.

$list(m, 0) = 0$

$list(m, 1) = \langle 0, m \rangle$

$list(m, k+1) = \langle k, \langle m, r(list(m, k)) \rangle \rangle$

$list(m, k+1) = \langle k, \langle m, r(list(m, k)) \rangle \rangle$

$= \langle k, \langle m, \langle m, r(list(m, k-2)) \rangle \rangle \rangle$

$= \ldots$

$= \langle k, \langle m, \langle m, \ldots, \langle m, r(list(m, 1)) \rangle \ldots \rangle \rangle \rangle$

$= \langle k, \langle m, \langle m, \ldots, \langle m, m \rangle \ldots \rangle \rangle \rangle$

$= [m, m, \ldots, m]$

(b) $find([n_1, \ldots, n_k], m) \cong min \{i | 1 \leq i \leq k, n_k = m\}$ if such an i exists,

$\cong 0$, otherwise.

$find(n, m) = (min\ i)_{1 \leq i \leq size(n)} [item(n, i) = m]$

(b) $replace([n_1, \ldots, n_k], m, i) \cong [n_1, \ldots, n_{i-1}, m, n_{i+1}, \ldots, n_k]$ if $1 \leq i \leq k$,

$\cong 0$ *otherwise.*

If $1 \leq i \leq size(n)$ ***then***

$replace(n, m, i) \; (\min t)_{t \leq list(n+m,\, size(n))} \; [size(t) = size(n)$

and

$find(n, m) = (\min i)_{1 \leq i \leq size(n)} \; [item(n, i) = m]$

$[n_1] = \langle 0, n_1 \rangle =$

$[n_1, n_2] = \langle 1, \langle n_1, n_2 \rangle \rangle$

$[n_1, n_2, n_3] = \langle 2, \langle n_1, \langle n_2, n_3 \rangle \rangle \rangle$

$[n_1, n_2, n_3, n_4] = \langle 3, \langle n_1, \langle n_2, \langle n_3, n_4 \rangle \rangle \rangle \rangle$

Example 4.36 *f: $N \rightarrow N$,*

f(0) = 1,

$f(n+1) = f(0)^{n+1} + f(1)^{n} + \ldots + f(n)^{1}$ is primitive recursive.

Solution

Theorem 4.37 *Let*

$f(n_1, \ldots, n_k, 0) = g(n_1, \ldots, n_k)$

$f(n_1, \ldots, n_k, m+1) = h(n_1, \ldots, n_k), m, [f_0, f_1, \ldots, f_m])$

where $f_i = f(n_1, \ldots, n_k, i)$. If g and h are primitive recursive, then f is also primitive recursive.

4.8 Partial Recursive Function

Unbounded minimization: *Given a* ***total*** *predicate* $g: N^{k+1} \rightarrow \{0, 1\}$, *define* $f: N^k \rightarrow N$.

$f(n_1, \ldots, n_k) \cong (\min m)\ g(n_1, \ldots, n_k, m)$ *if* $(\exists m)\ g(n_1, \ldots, n_k, m)$,

$\cong \uparrow$*(undefined)* *otherwise.*

We simply write $f(n_1, \ldots, n_k) = (\min m)\ g(n_1, \ldots, n_k, m)$.

Partial Recursive Function

(1) Initial functions and

(2)finite number of application of ***composition***, ***primitive recursion***, *and* ***unbounded minimization***.

A partial recursive function f is called ***recursive*** *function, if f is total.*

***Recursive set**: A set $A \subseteq N^k$ is recursive if its **characterristic** function*

$$\chi_A(n_1, \ldots, n_k) \cong 1, \text{ if } (n_1, \ldots, n_k) \in A$$
$$\cong 0, \text{ if } (n_1, \ldots, n_k) \notin A$$

*is a (**total**) **recursive** function.*

***Recursively enumerable set**: A set $A \subseteq N^k$ is recursively enumerable if its **semi-characterristic** function*

$$\sigma_A(n_1, \ldots, n_k) \cong 1, \text{ if } (n_1, \ldots, n_k) \in A$$
$$\cong \uparrow, \text{ if } (n_1, \ldots, n_k) \notin A$$

*is a **partial recursive** function.*

*We say a set $A \subseteq N^k$ is **primitive recursive** if χ_A is **primitive recursive**.*

***Example 4.39** $A = \{n | (\exists m) R(n, m)\}$ for some recursive set R, A is r.e.*
(a) $\sigma_A(n) = neg(neg(1 + (\min m)R(n, m)))$.

Any fixed alphabet $\Sigma = \{s_1, \ldots, s_k\}$ *with a fixed ordering* $s_1 \lessdot s_2 \lessdot \ldots \lessdot s_k$,

the ***lexicographic ordering*** $\lessdot$ *on strings in* Σ^* *is defined as follow:*

Let $x = a_1a_2 \ldots a_m$ *and* $y\ b_1b_2 \ldots b_n$ *where* $a_i, b_j \in \Sigma$ *for* $1 \leq^{\forall} i \leq m$, $1 \leq^{\forall} j \leq n$.

We say $x \lessdot y$, *if*

(a) $|x| < |y|$ *(i.e.,* $m < n$*), or*

(b) $|x| = |y|$ *and* $(\exists i)_{1 \leq i \leq m}\ [a_1 = b_1, \ldots, a_{i-1} = b_{i-1}, a_i \lessdot b_i]$.

$\Sigma^* = \{\varepsilon, s_1, s_2, \ldots, s_k, s_1s_1, s_1s_2, \ldots, s_ks_k, s_1s_1s_1, \ldots\}$

ι_Σ: $N \leftrightarrow \Sigma^*$.ə. $\iota_\Sigma(n)$ *is n-th string in* Σ^* *under*

the ***lexicographic ordering***. $\lessdot$ *(starting from* $\iota_\Sigma(0) = \varepsilon$*).*

Theorem 4.40 *Let* $\Sigma = \{s_1, \ldots, s_k\}$ *and* $s_1 \lessdot s_2 \lessdot \ldots \lessdot s_k$. *Then*

$^{\forall}x = s_{i_n}s_{i_{n-1}} \ldots s_{i_0} \in \Sigma^*$ *with* $n \geq 0$,

$\iota^{-1}(x) = i_n \cdot k^n + i_{n-1} \cdot k^{n-1} + \ldots + i_1 \cdot k^1 + i_0$.

Example 4.41 Suppose

A partial function $f: (\Sigma^*)^k \to \Sigma^*$ *is called* ***primitive recursive****(or* ***partial recursive****) if the function* $\tilde{f} : N^k \to N$ *is defined by*

$$\tilde{f}(n_1, \ldots, n_k) = \iota_\Sigma^{-1}(f(\iota_\Sigma(n_1), \ldots, \iota_\Sigma(n_k)))$$

is ***primitive recursive****(or, respectively,* ***partial recursive****)*

Lemma 4.43 *For any grammar G the predicate*

$d_G(\alpha, \beta, k) = [\alpha \Rightarrow^k \beta]$ *is primitive recursive.*

Proof $d_G(\alpha, \beta, 0) = eq(\alpha, \beta)$

$d_G(\alpha, \beta, k+1) = (\exists\gamma)_{|\gamma| \le |\beta|+l} [d_G(\alpha, \gamma, k)$ *and* $\gamma \Rightarrow \beta]$

where l is the length of the longest string in the LHS of any rules in G.

Theorem 4.44 *Every* ***Turing-computable*** *function is* ***partial recursive****.*

Proof *If* $f: (\Sigma^*)^k \rightarrow \Sigma^*$ *is Turing-computable, then* $^{\exists}$*a grammar* G_M*, .э.*

$$f(x_1, \ldots, x_k) = y \Leftrightarrow (\exists t)\, d_{G_M}([Bx_1Bx_2B\ldots Bx_ksB], [ByhB], t)$$