

理论计算机基础

Little Wolf

2024 年 10 月 25 日

目录

1 \mathcal{S} 程序和可计算函数	2
2 原始递归函数	9
3 通用程序	13
4 Turing机	17

1 \mathcal{S} 程序和可计算函数

题目. P_9 1.3.6 对程序:

\mathcal{P}_2 :

```

IF   X ≠ 0  GOTO  A
Y ← Y + 1
Z ← Z + 1
IF   Z ≠ 0  GOTO  E
[A]X ← X - 1
[B]IF  X ≠ 0  GOTO  B

```

给出它从输入变量 X 分别等于0, 1, 5的初始状态开始的计算.

解答. (1) 思路: $X = 0$, 不跳转到[A], 之后 $Y = 1, Z = 1$, 此时 $Z \neq 0$, 跳转到[E], 结束程序. 程序结束时, 输出变量 $Y = 1$.

计算:

- (1, { $X = 0, Y = 0, Z = 0$ })
- (2, { $X = 0, Y = 0, Z = 0$ })
- (3, { $X = 0, Y = 1, Z = 0$ })
- (4, { $X = 0, Y = 1, Z = 1$ })
- (7, { $X = 0, Y = 1, Z = 1$ })** (终点快相)

(2) 思路: $X = 1$, 跳转到[A], 执行 $X \leftarrow X - 1$ 后, $X = 0$, 不执行[B], 结束程序. 程序结束时, 输出变量 $Y = 0$.

计算:

- (1, { $X = 1, Y = 0, Z = 0$ })
- (5, { $X = 1, Y = 0, Z = 0$ })
- (6, { $X = 0, Y = 0, Z = 0$ })
- (7, { $X = 0, Y = 0, Z = 0$ })**

(3) 思路: $X = 5$, 跳转到[A], 执行 $X \leftarrow X - 1$ 后, $X = 4$, 执行[B], 进入死循环. 程序结束时, 输出变量 $Y = 0$.

计算:

- (1, { $X = 5, Y = 0, Z = 0$ })
- (5, { $X = 5, Y = 0, Z = 0$ })
- (6, { $X = 4, Y = 0, Z = 0$ })
- (6, { $X = 4, Y = 0, Z = 0$ })
- ... (死循环)

□

题目. P_9 1.3.7 对程序

\mathcal{P}_3 :

```

 $X_1 \leftarrow X_1 + 1$ 
 $X_1 \leftarrow X_1 + 1$ 
[A] $X_1 \leftarrow X_1 - 1$ 
    IF  $X_1 \neq 0$  GOTO C
[B] $Z \leftarrow Z + 1$ 
    IF  $Z \neq 0$  GOTO B
[C] $X_1 \leftarrow X_1 - 1$ 
    IF  $X_1 \neq 0$  GOTO A
    IF  $X_2 \neq 0$  GOTO D
 $Y \leftarrow Y + 1$ 
[D] $Y \leftarrow Y$ 
```

设输入变量的初始状态的值如下:

- (1) $X_1 = 2, X_2 = 0$
 - (1) $X_1 = 4, X_2 = 3$
 - (1) $X_1 = 1, X_2 = 4$
- 写出计算

解答. (1) 分析: 执行了[A]后, $X_1 = 3$, 跳转到C, 之后 $X_1 = 2$, 跳转回[A], $X_1 = 1$, 再跳转到[C], $X_1 = 0$, 而 $X_2 = 0$, 执行 $Y \leftarrow Y + 1$, 之后进入空指令[D]. 最后输出变量 $Y = 1$.

计算:

- (1, $\{X_1 = 2, X_2 = 0, Y = 0, Z = 0\}$)
- (2, $\{X_1 = 3, X_2 = 0, Y = 0, Z = 0\}$)
- (3, $\{X_1 = 4, X_2 = 0, Y = 0, Z = 0\}$)
- (4, $\{X_1 = 3, X_2 = 0, Y = 0, Z = 0\}$)
- (7, $\{X_1 = 3, X_2 = 0, Y = 0, Z = 0\}$)
- (8, $\{X_1 = 2, X_2 = 0, Y = 0, Z = 0\}$)
- (3, $\{X_1 = 2, X_2 = 0, Y = 0, Z = 0\}$)
- (4, $\{X_1 = 1, X_2 = 0, Y = 0, Z = 0\}$)
- (7, $\{X_1 = 1, X_2 = 0, Y = 0, Z = 0\}$)
- (8, $\{X_1 = 0, X_2 = 0, Y = 0, Z = 0\}$)
- (9, $\{X_1 = 0, X_2 = 0, Y = 0, Z = 0\}$)
- (10, $\{X_1 = 0, X_2 = 0, Y = 0, Z = 0\}$)
- (11, $\{X_1 = 0, X_2 = 0, Y = 1, Z = 0\}$)
- (12, $\{X_1 = 0, X_2 = 0, Y = 1, Z = 0\}$)

(2) 思路: 执行[A]之后, $X_1 = 5$, 跳转[C], 之后 $X_1 = 4$, 跳转回[A], 在[C],[A]间来回跳转, 根据 X_1 的奇偶性, 最后在执行[C]的第一步之后 $X_1 = 0, X_2 = 3 \neq 0$, 跳转到空指令D. 最后输出变量 $Y = 0$.

计算:

```
(1, {X1 = 4, X2 = 3, Y = 0, Z = 0})
(2, {X1 = 5, X2 = 3, Y = 0, Z = 0})
(3, {X1 = 6, X2 = 3, Y = 0, Z = 0})
(4, {X1 = 5, X2 = 3, Y = 0, Z = 0})
(7, {X1 = 5, X2 = 3, Y = 0, Z = 0})
(8, {X1 = 4, X2 = 3, Y = 0, Z = 0})
(3, {X1 = 4, X2 = 3, Y = 0, Z = 0})
(4, {X1 = 3, X2 = 3, Y = 0, Z = 0})
(7, {X1 = 3, X2 = 3, Y = 0, Z = 0})
(8, {X1 = 2, X2 = 3, Y = 0, Z = 0})
(3, {X1 = 2, X2 = 3, Y = 0, Z = 0})
(4, {X1 = 1, X2 = 3, Y = 0, Z = 0})
(7, {X1 = 1, X2 = 3, Y = 0, Z = 0})
(8, {X1 = 0, X2 = 3, Y = 0, Z = 0})
(9, {X1 = 0, X2 = 3, Y = 0, Z = 0})
(11, {X1 = 0, X2 = 3, Y = 0, Z = 0})
(12, {X1 = 0, X2 = 3, Y = 0, Z = 0})
```

(3) 思路: 执行[A]之后, $X_1 = 2$, 跳转[C], 之后 $X_1 = 1$, 跳转回[A], $X_1 = 0$, 执行[B], $Z = 1$, 在[B]中进入死循环. 最后输出变量 $Y = 0$.

计算:

```
(1, {X1 = 1, X2 = 4, Y = 0, Z = 0})
(2, {X1 = 2, X2 = 4, Y = 0, Z = 0})
(3, {X1 = 3, X2 = 4, Y = 0, Z = 0})
(4, {X1 = 2, X2 = 4, Y = 0, Z = 0})
(7, {X1 = 2, X2 = 4, Y = 0, Z = 0})
(8, {X1 = 1, X2 = 4, Y = 0, Z = 0})
(3, {X1 = 1, X2 = 4, Y = 0, Z = 0})
(4, {X1 = 0, X2 = 4, Y = 0, Z = 0})
(5, {X1 = 0, X2 = 4, Y = 0, Z = 0})
(6, {X1 = 0, X2 = 4, Y = 0, Z = 1})
(5, {X1 = 0, X2 = 4, Y = 0, Z = 1})
(6, {X1 = 0, X2 = 4, Y = 0, Z = 2})
... (进入死循环)
```

□

题目. P_{12} 1.1 写出计算下述函数的 \mathcal{S} 程序(允许使用宏指令):

- (1) $f(x) = \lfloor x/2 \rfloor$ (向下取整)
- (2) x 偶数, $f(x) = 1$; x 奇数, $f(x)$ 无定义.

解答. (1) 思路: 除以2可以用一直减2表示.

\mathcal{P}_1 :

```

 $Z \leftarrow Z + 1$ 
 $X \leftarrow X + 1$  (+1的目的是为了保证2的输出是1, 以此类推)
[A] $X \leftarrow X - 1$ 
 $X \leftarrow X - 1$ 
    IF  $X \neq 0$  GOTO  $B$ 
    IF  $Z \neq 0$  GOTO  $E$ 
[B] $Y \leftarrow Y + 1$ 
    IF  $Y \neq 0$  GOTO  $A$ 
```

使用宏指令的版本:

\mathcal{P}_1^* :

```

 $X \leftarrow X + 1$ 
[A] $X \leftarrow X - 2$ 
    IF  $X \neq 0$  GOTO  $B$ 
    GOTO  $E$ 
[B] $Y \leftarrow Y + 1$ 
    GOTO  $A$ 
```

(2) 思路: 对输入的 X , 循环减两次1, 但每次都检查 X 是否是0, 来判断奇偶性, 为了兼容0, 首先加上1. 简单来说, 就是看减去的是奇数个还是偶数个1来进行出口的分类.

\mathcal{P}_2^* :

```

 $X \leftarrow X + 1$ 
[A] $X \leftarrow X - 1$ 
    IF  $X = 0$  GOTO  $B$ 
     $X \leftarrow X - 1$ 
    IF  $X \neq 0$  GOTO  $A$ 
    GOTO  $C$ 
[B] $Y \leftarrow Y + 1$ 
    GOTO  $E$ 
[C] $Z \leftarrow Z + 1$ 
    IF  $Z \neq 0$  GOTO  $C$ 
```

如果不允许判断 $X = 0$, 可以这么写:

\mathcal{P}_2^* :

```

 $X \leftarrow X + 1$ 
[A] $X \leftarrow X - 1$ 
    IF  $X \neq 0$  GOTO  $B$ 
    GOTO  $C$  (偶数出口)
```

```

[B]X ← X - 1
IF  X ≠ 0  GOTO  A
GOTO  D  (奇数出口)
[C]Y ← Y + 1
GOTO  E
[D]Z ← Z + 1
IF  Z ≠ 0  GOTO  D  (死循环)

```

□

题目的注记. 可供使用的宏指令:

- GOTO A
- $V \leftarrow V^l$
- 判断 $X = 0$ 和跳转

题目. $P_{12} 1.2$ 给出下列程序 \mathcal{P} 计算的函数 $\psi_{\mathcal{P}}^{(1)}(x)$:

```

(1)[A]X ← X + 1
      X ← X - 1
      IF  X ≠ 0  GOTO  A
(2)[A]X ← X - 1
      IF  X = 0  GOTO  A
      X ← X - 1
      IF  X ≠ 0  GOTO  A
(3)空程序

```

$$\text{解答. (1)} \psi_{\mathcal{P}_1}^{(1)}(x) = \begin{cases} \uparrow (\text{未定义}) & \text{if } x \in \mathbb{N}^*, \\ 0 & \text{if } x = 0. \end{cases}$$

$$(2) \psi_{\mathcal{P}_1}^{(1)}(x) = \begin{cases} 0, & x \text{是正偶数} \\ \uparrow, & x = 0 \text{或} x \text{是奇数} \end{cases}$$

$$(3) \psi_{\mathcal{P}_1}^{(1)}(x) = 0, \forall x \in \mathbb{N}$$

□

题目. $P_{12} 1.3$ 证明下面的函数是部分可计算的:

- (1) $x_1 + x_2$; (2) $x_1 - x_2$; (3) $x_1 x_2$; (4) 空函数

解答. 要证明一个函数是部分可计算的, 实际上就是可以用 S 函数把它写出来, “部分”指的是可以在某些点上没有定义

(1) 思路: x_2 一直减1, x_1 一直加1, 直到减到零

```

Y ← X1
Z ← X2
[A]IF  Z ≠ 0  GOTO  B
      GOTO  E

```

```
[B]Z ← Z - 1
    Y ← Y + 1
    GOTO A
```

(2) 思路: 如果 $x_1 < x_2$, 那么无定义, 因此看谁先减到0.

```
Z1 ← X1
Z2 ← X2
[A]IF Z1 ≠ 0 GOTO B
    GOTO D1
[B]IF Z2 ≠ 0 GOTO C
    GOTO D2
[C]Z1 ← Z1 - 1
    Z2 ← Z2 - 1
    GOTO A
[D1]Y ← Z2
    GOTO E
[D2]Y ← Z1
    GOTO E
```

(3) 思路: 如果有0, 返回0; 如果都不是0, 一直减 x_2 , 同时对 x_1 做加法(已经证明是部分可计算函数)

```
Z1 ← X1
Z2 ← X2
[A]IF Z1 ≠ 0 GOTO B
    GOTO E
[B]IF Z2 ≠ 0 GOTO C
    GOTO E
[C]Z2 ← Z2 - 1
    Z1 ← Z1 + Z1
    GOTO B
```

(4) 思路: 空函数就是处处无定义, 直接进入死循环即可.

```
[A]Z ← Z + 1
    IF Z ≠ 0 GOTO A
```

□

题目. P₁₃ 1.4 证明下述谓词是可计算的

- (1) $x \geq a$, a 是正整数
- (2) $x_1 \leq x_2$
- (3) $x_1 = x_2$

解答. 要证明一个谓词是可计算的, 实际上就是证明这个谓词(判断过程)可以用S语言表示

(1) 思路: 两边一直减1, 看谁先减到0, 又因为是大于等于, 所以可以先验证 Z_2

```

 $Z_1 \leftarrow X$ 
 $Z_2 \leftarrow a$ 
[A]IF  $Z_2 \neq 0$  GOTO B
      GOTO  $D_2$ 
[B]IF  $Z_1 \neq 0$  GOTO C
      GOTO  $D_1$ 
[C] $Z_1 \leftarrow Z_1 - 1$ 
 $Z_2 \leftarrow Z_2 - 1$ 
      GOTO A
[D1]GOTO E
[D2] $Y \leftarrow Y + 1$ 
      GOTO E

```

(2) 思路: 和第一问思路类似, 注意取等条件对判定顺序的影响

```

 $Z_1 \leftarrow X_1$ 
 $Z_2 \leftarrow X_2$ 
[A]IF  $Z_1 \neq 0$  GOTO B
      GOTO  $D_2$ 
[B]IF  $Z_2 \neq 0$  GOTO C
      GOTO  $D_1$ 
[C] $Z_1 \leftarrow Z_1 - 1$ 
 $Z_2 \leftarrow Z_2 - 1$ 
      GOTO A
[D1]GOTO E
[D2] $Y \leftarrow Y + 1$ 
      GOTO E

```

(3) 思路: 可以使用(1)和(2)的判定了

```

 $Z_1 \leftarrow X_1$ 
 $Z_2 \leftarrow X_2$ 
[A]IF  $Z_1 \leq Z_2$  GOTO B
      GOTO E
[B]IF  $Z_2 \leq Z_1$  GOTO C
      GOTO E
[C] $Y \leftarrow Y + 1$ 
      GOTO E

```

□

2 原始递归函数

题目. P₁₆ 2.1.5用基本的原始递归函数来表示下面的函数, 从而它们也是原始递归函数

$$(1) E(x) = \begin{cases} 1, & x \text{偶数} \\ 0, & x \text{奇数} \end{cases}$$

$$(3) \max(x, y) = \begin{cases} x, & x \geq y \\ y, & x < y \end{cases}$$

解答. (1) $E(0) = 1, E(x + 1) = \alpha(E(x))$.

(2) $\max(x, y) = x\alpha(y - x) + y\alpha(x - p(y))$

这是因为, 当 $x \geq y$ 时, $y - x = 0$, 因此 $\alpha(y - x) = 1$. 但为了防止 $x = y$ 且取非零值($x = y = 0$ 不影响)的时候, 得到 $2x$, 因此要保证 $x = y$ 的时候, y 的系数不能是1, 因此取 $x - p(y)$ □

题目. P₂₂ 2.3.4利用极小化给出下述函数 $f(x)$:

$$(1) f(x) = \begin{cases} \sqrt{x}, & x \text{是完全平方数} \\ \uparrow, & \text{否则} \end{cases}$$

(2) $g(x)$ 是全函数, 若存在 t , 使得 $g(t) = x$, 则 $f(x)$ 等于使得 $g(t) = x$ 成立的最小的 t , 否则 $f(x) \uparrow$.

解答. (1) $f(x) = \min_t(t^2 = x)$

(2) $f(x) = \min_t(g(t) = x)$ □

题目的注记. 应该是对的吧: 使用极小化定义的函数中, 加入我想要定义的函数不是全函数(即在某些情况下无定义), 那么一定是用无界极小化定义的. 因为有界极小化+原始递归, 定义出的都是全函数

题目. P₃₆ 2.1 证明: 仅在有穷个点处非零值, 在其余点取零的函数一定是原始递归函数.

解答. 思路: 因为只在有限个地方取非零值, 那么只需要对这有限个点做有限递归

设 $f(x)$ 在 x_1, \dots, x_k 处取非零值 c_1, \dots, c_k , 其余点取零.

定义:

$$\chi_{x_i}(x) = \begin{cases} 1, & x = x_i \\ 0, & x \neq x_i \end{cases} = (x = x_i)$$

是原始递归的谓词, 那么定义:

$$f(x) = \sum_{i=1}^k c_i \cdot \chi_{x_i}(x)$$

(有限求和, 常数乘法原始递归)因此也是原始递归的. □

题目的注记. 也可以写成:

$$f(x) = \sum_{i=1}^k c_i \alpha(x - x_i)$$

题目的注记. 更严格的题目描述, 应该是, 把这有限个点给出, 即应该是给定的有限点

以及书上的描述: 任何给定的有限集是原始递归的.

题目. P₃₆ 2.3 设 $f(0) = 1, f(1) = 1, f(2) = 2^2, f(3) = 3^{3^3} = 3^{27}$ 等. 一般地, $f(n)$ 等于高度为 n 的一叠 n , 都是指数, 试证明 f 是原始递归的.

解答. 自己没有想出来, 看了答案才想出来的, 一直纠结于单变量的情况, 却没有想到构建多变量的函数, 之后给参数赋值来变回单变量

设 $h(n, m)$ 是 $m + 1$ 个 n 的指数堆叠, 那么 $h(n, 0) = n, h(n, t + 1) = n^{h(n, t)}$, 所以 $h(n, m)$ 原始递归, 因此 $h(n, n - 1) = f(n)$ 原始递归.

虽然最后一步更自然的说法应该是取 $n = m + 1$, 那么 $h(n, m) = h(m + 1, m) = h(n, n - 1) = f(n)$. 因为上面的 n 作为参数, 我是证明了 $h(n, m)$ 这个二元函数关于 m 是原始递归的. \square

题目. P₃₆ 2.5 设 $\sigma(0) = 0$; 当 $x \neq 0$ 时, $\sigma(x)$ 是 x 的所有因子的和. 证明 $\sigma(x)$ 是原始递归函数.

解答. 写成原始递归的形式:

$$\sigma(x) = \sum_{i=1}^x \min_{t \leq i} (t|x) = \sum_{i=1}^x \min_{t \leq n} ((t|x) \wedge (i \leq t)), x \neq 0$$

谓词 $(t|x)$ 和 $(i \leq t)$ 是原始递归的, 原始递归谓词的“且” \wedge 是原始递归的, 有界极小化 $\min_{t \leq n} ((t|x) \wedge (i \leq t))$ 是原始递归的, 因此 $\sigma(x)$ 也是原始递归的 \square

题目. P₃₆ 2.7 设 $\phi(x)$ 是小于等于 x 且与 x 互素的正整数的个数, 证明 Euler 函数 $\phi(x)$ 是原始递归函数.

解答. 先考虑判断 x, y 是否互素的量词 $P(x, y)$ 是原始递归的.

$$P(x, y) = (\forall)_{t \leq x} ((t = 1) \vee \neg((t|x) \wedge (t|y))) = (\forall)_{t \leq x} ((t = 1) \vee \neg(t|x) \vee \neg(t|y))$$

因此:

$$\phi(x) = \sum_{t=1}^x P(x, t)$$

\square

题目的注记. 奇怪的是, 书上的答案是这么写的:

$$P(x, y) = (x > 0 \vee y > 0) \wedge (\forall)_{t \leq x} ((t = 1) \vee \neg(t|x) \vee \neg(t|y))$$

即 x, y 不可以同时为 0, 这是必要的吗?

题目. P₂₄ 2.4.3 验证:

$$(x)_i = \min_{t \leq x} \{\neg(p_i^{t+1}|x)\}$$

对于 $(0)_i$ 和 $(x)_0$ 成立, 其中 x 和 i 是任意的自然数.

解答. 实际含义: 第 i 个素数 p_i 作为 x 的因子的次数

$$(0)_i = \min_{t \leq 0} \{\neg(p_i^{t+1}|0)\} = \neg(p_i|0) = \neg 1 = 0$$

$$(x)_0 = \min_{t \leq x} \{\neg(0^{t+1}|x)\} = \min_{t \leq x} \{\neg(0|x)\} = \min_{t \leq x} \{1\} = 0$$

解释: $(0|x) = (\exists)_{t \leq x} (t \cdot 0 = x) = 0$ \square

题目. P_{37} 2.11 设 $R(x, t)$ 是原始递归谓词, 定义有界极大化:

$$g(x, y) = \max_{t \leq y} R(x, t)$$

当存在 $t \leq x$ 使得 $R(x, t)$ 为真时, $g(x, y)$ 等于这样的 t 的最大值; 当不存在这样的 t 的时候, $g(x, y) = 0$. 证明 $g(x, y)$ 原始递归.

解答. 第一反应是, 已经知道了有界极小化原始递归, 可以尝试用有界极小化来表示有界极大化. 那么可以这么想: 使得 $R(x, t)$ 成立的最大的 t 就是: 使得 $k = t$ 到 y 的 $R(x, k)$ 只有 $R(x, t)$ 成立的最小的 t .

$$g(x, y) = \min_{t \leq y} \left\{ R(x, t) \wedge \left(\left((t < y) \wedge \alpha \left(\sum_{k=t+1}^y R(x, k) \right) \right) \vee (t = y) \right) \right\}$$

当不存在这样的 t 的时候, $R(x, t)$ 恒为 0, 因此 $g(x, y) = 0$, 满足题设 \square

题目的注记. 书上给的两种解法也很有意思:

(1) 方法一: 和我的思路是一样的, 即首先 $R(x, t)$ 要成立, 且要么 $s \leq t$, 要么 $R(x, s)$ 不成立. 但是他比我聪明的地方在于对 \forall 的量词和 \vee 的使用

$$g(x, y) = \min_{t \leq y} \{R(x, t) \wedge (\forall s)_{s \leq y} [(s \leq t) \vee \neg R(x, s)]\}$$

(2) 方法二: 倒着开始用有界极小化, 找到了再减回去得到正着数的下标

$$g(x, y) = \begin{cases} y - \min_{t \leq x} R(x, y - t), & (\exists z)_{z \leq y} R(x, z) \\ 0, & \text{否则} \end{cases}$$

题目. P_{37} 2.13

- (1) Cantor 编码 $\pi(x, y)$ 的定义如下所示
- (2) 若 $\pi(x, y) = z$, 则 $\sigma_1(z) = x, \sigma_2(z) = y$
- (3) $\sigma(z) = \sigma_1(z) + \sigma_2(z)$

证明 $\pi(x, y), \sigma_1(z), \sigma_2(z), \sigma(z)$ 都是原始递归的.

解答. (1) $\pi(x, y)$ 是 x 行 y 列, 首先, 第 0 行, 第 y 列是 $\frac{y(y+1)}{2}$, 所以 (x, y) 元素是:

$$\frac{y(y+1)}{2} + (y+2) + (y+3) + \dots + (x+y+1) = \frac{(x+y)(x+y+1)}{2} + x$$

因此是原始递归函数

(2) 答案是这么写的, 但是还有点疑惑

$$\begin{aligned} \sigma_1(z) &= \min_{x \leq z} [(\exists y)_{\leq z} \pi(x, y) = z] \\ \sigma_2(z) &= \min_{y \leq z} [(\exists x)_{\leq z} \pi(x, y) = z] \end{aligned}$$

(3) $\sigma_1(z), \sigma_2(z)$ 原始递归, 所以加和原始递归 \square

题目. P_{38} 2.21 证明下述字函数是原始递归的:

- (1) 长度函数 $|w|$
 $|w|$ 等于 w 的长度, 即 w 中的字符个数;
- (2) 连接函数 $\text{CONCAT}^{(m)}(w_1, w_2, \dots, w_m)$, 其中 m 是正整数,

$$\text{CONCAT}^{(m)}(w_1, w_2, \dots, w_m) = w_1 w_2 \cdots w_m$$

(3) 字尾函数 $\text{RTEND}(w)$

当 $w = \epsilon$ 时, $\text{RTEND}(w) = \epsilon$; 当 $w \neq \epsilon$ 时, $\text{RTEND}(w)$ 等于 w 最右端的字符;

(4) 字头函数 $\text{LTEND}(w)$

当 $w = \epsilon$ 时, $\text{LTEND}(w) = \epsilon$; 当 $w \neq \epsilon$ 时, $\text{LTEND}(w)$ 等于 w 最左端的字符;

(5) 截尾函数 w^-

当 $w = \epsilon$ 时, $w^- = \epsilon$; 当 $w \neq \epsilon$ 时, w^- 等于 w 删去最右端的字符后的子串;

(6) 截头函数 ${}^{\sim}w$

当 $w = \epsilon$ 时, ${}^{\sim}w = \epsilon$; 当 $w \neq \epsilon$ 时, ${}^{\sim}w$ 等于 w 删去最左端的字符后的子串。

解答. 要证明字函数是原始递归函数, 需要证明这个字函数对应的数论函数是原始递归的; 或者是由其他的原始递归的字函数构造得到的

指定字母表 $A = \{a_1, \dots, a_n\}$

(1) 从直观上说, 给定一个字符串 w , 获取 w 的长度是不需要知道 w 的字符所属的字母表的, 但是, 这里为了分析这个过程的可计算性, 需要指定一个字母表.

那么在 n 进制下, 长度为 $k + 1$ 的最小字符串是 $a_1 a_1 \dots a_1 = \sum_{t=0}^k n^t$, 那么长度是(控制 $k \leq x$, 可以是有界极小化, 这里的 x 在比较的时候, 就是数), 根据有界极小化和原始递归谓词定义的函数是原始递归的:

$$\min_{k \leq x} \left\{ \sum_{t=0}^k n^t \right\} > x$$

(2) 连接函数, w_1 向前挪动一格, 值乘上 n , 而向前挪动的格子数可以由第一问的长度函数得到, 不妨设长度函数是 $l(w)$, 那么:

$$\begin{aligned} & \text{CONCAT}^{(m)}(w_1, \dots, w_m) \\ &= w_m + w_{m-1} \cdot n^{l(w_m)} + w_{m-1} \cdot n^{l(w_m) + l(w_{m-1})} + \dots + w_1 \cdot n^{l(w_m) + \dots + l(w_2)} \\ &= \sum_{i=0}^{m-1} w_{m-i} \cdot n^{l_{w_m} + \dots + l_{w_{m-i+1}}} \end{aligned}$$

(3) 首先, 判断 $w = \epsilon$ 当然是原始递归的谓词, 因为这就是 $w = 0$ 的判断, 其次, 获取 w 最右端字符就是 $R^+(w, n)$, 因此是原始递归的

$$\text{RTEND}(w) = \begin{cases} \epsilon, & \text{if } w = \epsilon \\ R^+(w, n), & \text{else} \end{cases}$$

(4) 长度为 $l(w)$ 的字符串的首位是 $n^{l(w)-1}$ 的阶, 因此用 $Q^+(w, n^{l(w)-1})$ 获取最左端的字符(即 w 用 $n^{l(w)-1}$ 来除之后得到的整数部分), 因此是原始递归的

$$\text{LTEND}(w) = \begin{cases} \epsilon, & \text{if } w = \epsilon \\ Q^+(w, n^{l(w)-1}), & \text{else} \end{cases}$$

当然, 也可以用书上讲过的, 可以直接拿来用的函数, 更加安全, $h(m, n, x)$ 表示字符串 x 在 n 进制表示下第 m 位的字符, 因此可以写成:

$$\text{LTEND}(w) = \begin{cases} \epsilon, & \text{if } w = \epsilon \\ h(l(w) - 1, n, x), & \text{else} \end{cases}$$

(5) 截尾函数 w^- , 除 n 后留下来的整数部分就是向右挪动一格的结果, 很容易用原始递归函数来表示, 注意 $R^+(w, n)$ 是余数部分, $Q^+(w, n)$ 才是整数部分:

$$w^- = \begin{cases} \epsilon, & \text{if } w = \epsilon \\ Q^+(w, n), & \text{else} \end{cases}$$

(6) 可以获得长度 $l(w)$, 最左端字符的阶是 $n^{l(w)-1}$, 可以根据 $\text{LTEND}(w)$ 获得首位字符是什么, 那么减去即可:

$$w = \begin{cases} \epsilon, & \text{if } w = \epsilon \\ w - \text{LTEND}(w) \cdot n^{l(w)-1}, & \text{else} \end{cases}$$

□

题目. P₃₈ 2.22 设两个字母表 $A = \{s_1, s_2, \dots, s_n\}$ 和 $\tilde{A} = \{s_1, s_2, \dots, s_m\}$, 其中 $n < m$ 。任给一个数 x , 设 $w \in A^*$ 是 x 的 n 进制表示, w 也是 \tilde{A}^* 上的字符串, 把以 m 为底所表示的数记为 $\text{UPCHANGE}_{n,m}(x)$.

反之, 任给一个数 x , 设 $w \in \tilde{A}^*$ 是 x 的 m 进制表示, 删去 w 中不属于 A 的符号后得到 $w' \in A^*$, 把以 n 为底 w' 所表示的数记作 $\text{DOWNCHANGE}_{n,m}(x)$ 。例如, $\text{DOWNCHANGE}_{2,5}(36) = 9$. 又如, $s_1s_4s_2s_3$ 以 5 为底表示 238, 删去 s_4 和 s_3 得到 s_1s_2, s_1s_2 , 以 2 为底表示 4, 故 $\text{DOWNCHANGE}_{2,5}(238) = 4$.

试证明: $\text{DOWNCHANGE}_{n,m}(x)$ 和 $\text{UPCHANGE}_{n,m}(x)$ 是原始递归的.

解答. (1) 对 $\text{UPCHANGE}_{n,m}(x)$, 用原始递归的函数来表示它. 通过 $l_n(x)$ 来获取 x 在 n 进制表示下的长度, 用 $h(k, n, x), 0 \leq k \leq l_n(x) - 1$ 来获取每一位在字母表 A 中的序数, 因为 $A \subseteq \tilde{A}$, 因此也是在字母表 \tilde{A} 中的序数, 然后通过 m 进制表示即可:

$$\text{UPCHANGE}_{n,m}(x) = \sum_{k=0}^{l_n(x)-1} h(k, n, x) \cdot m^k$$

(2) 对于 $\text{DOWNCHANGE}_{n,m}(x)$, 用 $l_m(x)$ 来获取在 m 进制下的长度, 用 $h(k, m, x), 0 \leq k \leq l_m(x) - 1$ 来获取每一位的字符的操作只需要考察 $h(k, m, x)$ 是否小于等于 n 即可, 而对于删去一些字符后的位数, 可以用累加计数的方法来实现, 函数 $P(x) = \begin{cases} 1, & \text{if } x \leq n \\ 0, & \text{else} \end{cases}$

$$\text{DOWNCHANGE}_{n,m}(x) = \sum_{k=0}^{l_m(x)-1} h(k, m, x) \cdot P(h(k, m, x)) \cdot n^{\sum_{i=0}^k P(h(k, m, x))}$$

□

题目的注记. (1) 的书上的答案中写的是:

$$\text{UPCHANGE}_{n,m}(x) = \sum_{k=0}^{l_n(x)} h(k, n, x) \cdot m^k$$

(1) 的书上的答案中写的是:

$$\text{DOWNCHANGE}_{n,m}(x) = \sum_{k=0}^{l_m(x)} h(k, m, x) \cdot P(h(k, m, x)) \cdot n^{\sum_{i=0}^{k-1} P(h(k, m, x))}$$

我仔细检查了, 应该是答案写错了?

3 通用程序

题目. 计算 $\text{STP}(x_1, \dots, x_n, y, 0) = ?$

解答.

$$\text{STP}(x_1, \dots, x_n, y, 0) = \begin{cases} 1, & \text{if } y = 0 \\ 0, & \text{否则} \end{cases}$$

□

题目. 3.1 证明HALT($0, x$)是不可计算的

解答. 反证法, 假设HALT($0, x$)可计算, 构造:

[A]IF HALT($0, x$) GOTO A

这是可计算的, 有程序 P 可以计算, 程序 P 的编号是 k .

那么HALT($0, k$) $\iff P(k) = 0 \iff \neg\text{HALT}(0, k)$, 矛盾. □

题目的注记. 令

$$f(t, x) = \begin{cases} 0, \text{HALT}(x, x) \\ \uparrow, \neg\text{HALT}(x, x) \end{cases}$$

即 $f(t, x) = n(\Phi(x, x))$ 是部分可计算的(n 表示零函数)

则存在程序 P , 编码是 y_0 , 使得 $f(t, x) = \Phi^{(2)}(t, x, y)$

对任意给定的 x_0 , 构造程序 Q , 即执行 x_0 次 $X_2 \leftarrow X_2 + 1$, 然后执行 P

则程序 Q 关于输入 t 的计算, 与程序 P 关于输入 (t, x_0) 的计算相同

那么程序 Q 的编码: $[] - 1 = \prod_{i=1}^{x_0} p_i^{26} \prod_{j=1}^{L_t(y_0+1)} p_{x_0+j}^{(y_0+1)j} - 1$

其中, $X_2 \leftarrow X_2 + 1$ 的编码是 $<0, <1, 3>> = 26$

那么 Q 的编码这个数的计算是 x_0 的原始递归函数

即 $\#(Q)$ 是原始递归函数 $q(x)$

于是 $f(t, x) = \Phi^{(2)}(t, x, y_0) = \Phi^{(1)}(t, q(x))$

所以 $\Phi^{(1)}(t, q(x)) \downarrow \iff f(t, x) \downarrow \iff \text{HALT}(x, x) \downarrow$

特别的, $\Phi^{(1)}(0, q(x)) \downarrow \iff f(0, x) \downarrow \iff \text{HALT}(x, x) \downarrow$

其中 $\Phi^{(1)}(0, q(x)) \downarrow \iff \text{HALT}(0, q(x)) \downarrow$

若 $\text{HALT}(0, x)$ 可计算, 那么 $\text{HALT}(0, q(x))$ 可计算, 矛盾

题目的注记. 参数定理:

任意给定的 $n, m > 0$, 存在原始递归函数 $S_m^n(u_1, \dots, u_n, y)$ 使得, $\Phi^{(m+n)}(x_1, \dots, x_m, u_1, \dots, u_n, y) = \Phi^{(m)}(x_1, \dots, x_m, S_m^n(u_1, \dots, u_n, y))$

S_m^n 只和 m, n 有关

题目的注记. 递归定理:

设 $g(z, x_1, \dots, x_m)$ 是 $m + 1$ 元的部分可计算函数, 则存在 r , 使得 $g(r, x_1, \dots, x_m) = \Phi_r^{(m)}(x_1, \dots, x_m)$, r 是程序编码, 而等式左边的 r 是取定的变量取值

证明: 考虑部分可计算函数,

$g(S_m^{(1)}(t, t), x_1, \dots, x_m)$, 因为 $S_m^{(1)}(t, t)$ 原始递归, 所以它和部分可计算函数的复合是部分可计算的.

于是, 存在 y_0 , 使得

$g(S_m^{(1)}(t, t), x_1, \dots, x_m) = \Phi^{(m+1)}(t, x_1, \dots, x_m, y_0)$

再利用参数定理, 得到 $\Phi^{(m+1)}(t, x_1, \dots, x_m, y_0) = \Phi^{(m)}(t, x_1, \dots, x_m, S_m^1(t, y_0))$

令 $t = y_0$, 得到了 $r = S_m^1(y_0, y_0)$

题目的注记. 定理(Rice定理):

设 F 表示的是一元部分可计算函数的全体, $\forall \Gamma \subset F$, 记 $R_\Gamma = \{t \in N \mid \Phi_t \in \Gamma\}$ (称为 Γ 的下标集)

若存在部分可计算函数 f, g , 使得 $f \in \Gamma, g \notin \Gamma$ (这个集合既不是空集, 也不是全体, 但是为了可计算严格表述, 需要说出来具体的 f, g), 那么 R_Γ 不是递归集.

证明:

假设 R_Γ 是递归的(下面定义的 $h(t, x)$ 是部分可计算的), 令

$$h(t, x) = \begin{cases} g(x), t \in R_\Gamma \\ f(x), t \notin R_\Gamma \end{cases}$$

由递归定理, 存在一个 r 使得, $\Phi_r(x) = h(r, x) = \begin{cases} g(x), r \in R_\Gamma \\ f(x), r \notin R_\Gamma \end{cases}$

于是, $r \in R_\Gamma$ 吗?

$r \in R_\Gamma \iff \Phi_r(x) = g(x) \notin \Gamma \iff r \notin R_\Gamma$, 矛盾

题目. 3.3 证明: 不存在可计算函数 $f(x)$, 使得当 $\Phi(x, x) \downarrow$ 时 $f(x) = \Phi(x, x) + 1$.

解答. 反证法, 如果是可计算的, 那么函数 $f(x)$ 有编号 k , 那么 $f(x) = \Phi(x, k)$, 因此有 $f(k) = \Phi(k, k) = \Phi(k, k) + 1$, 矛盾. \square

题目. 设 f 是一个全函数, $B = \{f(n) | n \in N\}$, 证明:

- (1) 若 f 是可计算的, 那么 B 是r.e.
- (2) 若 f 是可计算的, 严格增加的($f(n) < f(n+1), \forall n \in N$), 则 B 是递归的.

解答. (1) 构造一个可计算函数, 使得输入的 x 如果在 B 里面, 就停机(输出什么都行), 否则不停机(无定义)

$$h(x) = \begin{cases} 0, & \text{if } \exists n (f(n) = x) \\ \uparrow, & \text{else} \end{cases}$$

可计算函数:

```
[A]IF  f(N) = X  GOTO  E
      N ← N + 1
      GOTO  A
```

(2) 构造一个可计算函数, 使得输入的 x 如果在 B 里面, 就输出1, 如果不在里面, 就输出0
又因为函数是严格单调的, 因此我只需要遍历 N , 直到 $f(N)$ 大于 X 或者 $f(N) = X$ 即可.

```
[A]IF  f(N) = X  GOTO  C
[B]IF  f(N) > X  GOTO  E
      N ← N + 1
      GOTO  A
[C]Y ← Y + 1
      GOTO  E
```

\square

题目的注记.

- 存在不是可计算的全函数, 例子(这是一个不可计算的谓词): $\text{HALT}(x, x)$
- 存在不是部分可计算的部分函数, 例子: $f(x) = \begin{cases} 1, & \text{if } \neg \text{HALT}(x, x) \\ 0, & \text{if } \text{HALT}(x, x) \end{cases}$

题目. 3.6 设 A, B 是 N 的非空子集, 定义:

$$\begin{aligned} A \odot B &= \{2x \mid x \in A\} \cup \{2x + 1 \mid x \in B\} \\ A \otimes B &= \{\langle x, y \rangle \mid x \in A, y \in B\} \end{aligned}$$

证明:

- (1) $A \odot B$ 是递归的当且仅当 A 和 B 是递归的
- (2) $A \otimes B$ 是递归的当且仅当 A 和 B 是递归的

题目的注记. 实际上, 上述两个集合的构造方式都是“双射”的编码, 因此只要知道 x 是否属于 $A \odot B$, 那么就一定知道 x 是否属于 A 或者 B , 反之亦然(显然)

解答. (1) 分析谓词: $(x \in A \odot B), (x \in A), (x \in B)$:

$$(x \in A \odot B) \iff ((2|x) \wedge (\lfloor x/2 \rfloor \in A)) \vee (\neg(2|x) \wedge (\lfloor (x-1)/2 \rfloor) \in B)$$

(2) 对于配对函数 $z = \langle x, y \rangle$, 有原始递归函数 $l(z), r(z)$.

分析谓词: $(x \in A \otimes B), (x \in A), (x \in B)$:

$$(z \in A \otimes B) \iff ((x = l(z)) \wedge (x \in A)) \vee ((x = r(z)) \wedge (x \in B))$$

□

题目. 3.7 证明集合 $B = \{x \in N \mid a \in \text{ran } \Phi_x\}$ 是递归可枚举的, a 是常数.

解答. 这里的 $\text{ran } \Phi_x$ 就是值域

注意, 这里的函数 Φ_x 是部分可计算的, 而不是像3.5的全函数

我需要构造一个可计算函数来判断 x 是否属于 B , 即, 如果属于, 我知道, 但是不知道不属于
需要再添加一个程序来执行对 $\Phi(n_0, x)$ 是否有定义的判定.

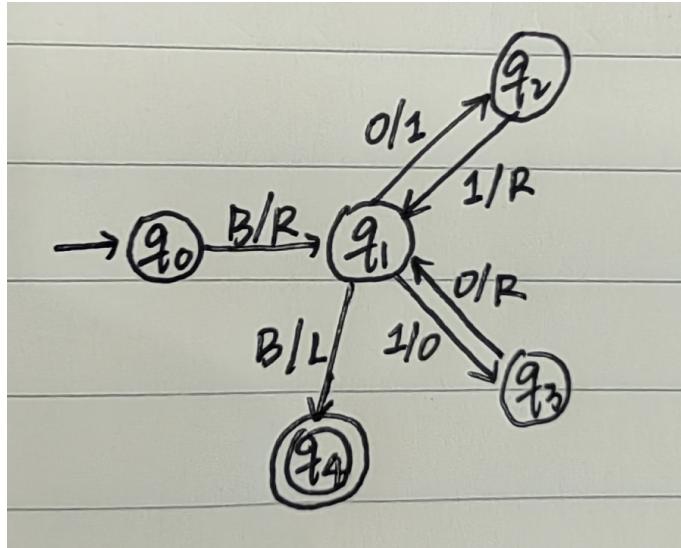
```
[A]IF   $\neg S T P(N, X, T)$   GOTO  B
    IF   $\Phi(N, X) = a$   GOTO  E      需要对输入N停机之后才能有输出来判断是不是a
[B]N  $\leftarrow N + 1$ 
    IF   $N \leq T$   GOTO  A
    T  $\leftarrow T + 1$ 
    N  $\leftarrow$ 
    GOTO  A
```

□

4 Turing机

题目. P₅₅ 4.1.4 设对 $w \in \{0, 1\}^*$, 有 $f(w) = \bar{w}$, 其中 \bar{w} 是 w 的反码, 即 $0 \rightarrow 1, 1 \rightarrow 0$. 构造一台TM计算 \bar{w} .

解答.



□

题目. P₆₄ 4.4.2 给出接受下列语言的TM

解答. 手写

□

题目. P₆₇ 4.5.1 设 NTM $\mathcal{M} = (Q, A, C, \delta, B, q_0, \{q_2\})$, 其中 $Q = \{q_0, q_1, q_2\}$, $A = \{0, 1\}$, $C = \{0, 1, B\}$, $\delta(q_0, B) = \{(R, q_0)\}$, $\delta(q_0, 0) = \{(R, q_0), (R, q_1), (R, q_2)\}$, $\delta(q_1, 1) = \{(R, q_0)\}$, $\delta(q_2, 0) = \{(L, q_0)\}$, $\delta(q_2, 1) = \{(L, q_0)\}$.

(0) 老师布置作业的时候添加的一个小问: 画出状态转移图

- (1) 画出关于输入 0010 的计算树;
- (2) 给出关于输入 0010 的一个停机在接受格局的计算, 一个停机在非接受格局的计算和一个永不停机的计算;
- (3) 给出 $L(\mathcal{M})$.

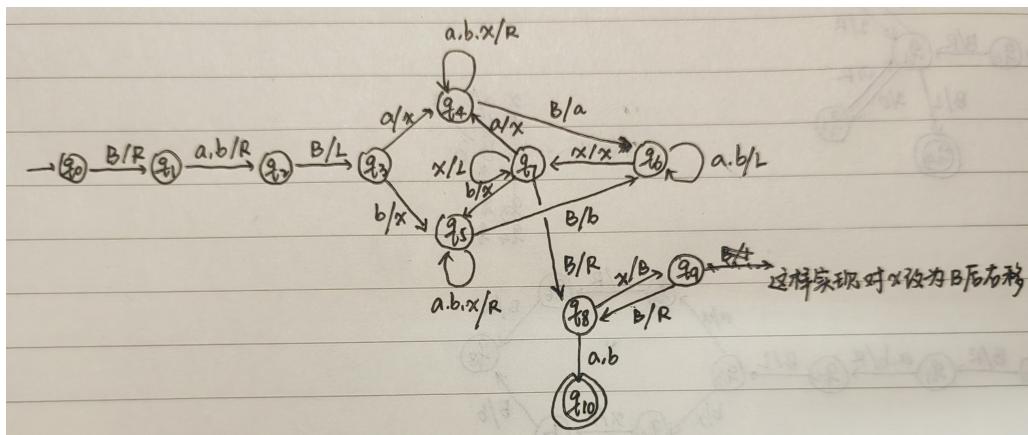
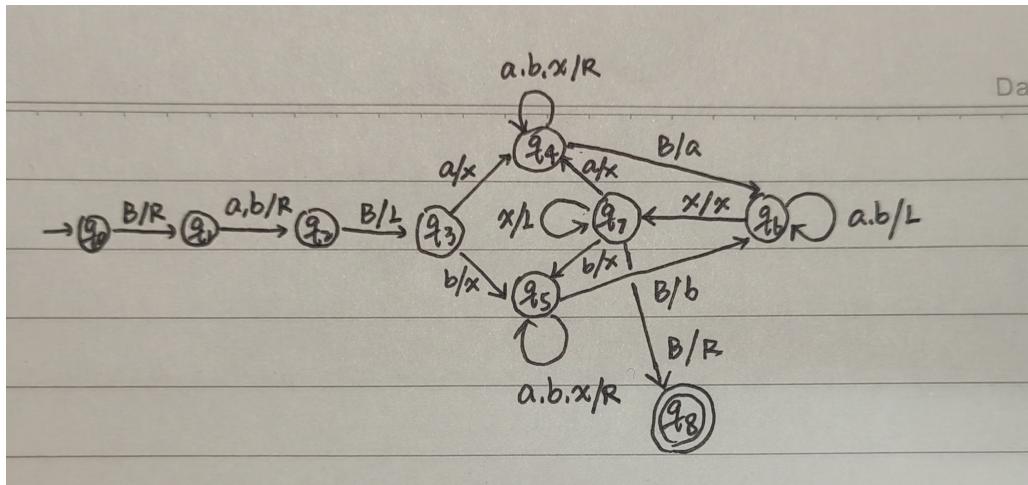
解答. 手写

□

题目. P₆₇ 4.1 设字母表 $A = \{a, b\}$, $f(x) = x^R$, x^R 是 x 的反转, 即颠倒 x 的符号的排列顺序所得到的字符串. 例如, $(abab)^R = baba$, 试构造一台TM计算 $f(x)$.

解答. 思路: 倒着读(从右往左读), 然后在字符串最右端的空白符的右边开始写(这样只需要把记录过的原字符串的字符改写成 x , 而不用最后还跑回去改写会空白了吗?) 这样的TM输出的会是 x^R , 之所以前面没有附带了相同长度的 x 字符串, 是因为 **图灵机在接收状态只会输出输出带上在 A 中的字符**. 当然, 如果想把 x 改为 B 也可以做到. 可以把状态 q_8 再添加一部分: 如果遇到的是 x , 那么把 x 改写为 B 之后向右移动一格, 那么就可以把所有的 x 都变为 B :

□



题目. P₆₈ 4.4 设计接受下述语言的基本TM, 可以直接简述思想

- (1) $\{0^n 1^n \mid n \in N\}$
- (2) $\{w \mid w \in \{0, 1\}^*\text{且} w\text{中}0\text{和}1\text{的个数相同}\}$

解答. 详细画图见手写部分

(1) q_0 遇到 B 右走到 q_1 , 如果遇到 0 , 右走到 q_2 ; 在 q_2 , 遇到 1 , 右走回到 q_1 . 在 q_1 遇到空白 B , 停机接受. 如果在 q_1 遇到 1 或者在 q_2 遇到 $0, B$, 停机不接受.

(2) q_0 遇到 B 右走到 q_1
如果 q_1 读到 0 , 标记为 x ; 然后右走(可能读到 $z, x, 0$) 直到遇见 1 , 标记为 z , 然后向左走(可能读到 $z, 0$) 直到之前标记的 x , 把 x 改为 z , 向右走一格, 回到状态 q_1

如果 q_1 读到 1 , 标记为 y ; 然后右走(可能读到 $z, y, 1$) 直到遇见 0 , 标记为 z , 然后向左走(可能读到 $z, 1$) 直到之前标记的 y , 把 y 改为 z , 向右走一格, 回到状态 q_1

如果 q_1 读到 z , 说明之前已经删除过这一字符, 向右走一格, 回到状态 q_1

如果 q_1 读到 B , 停机接受

其他状态读到 B 均停机不接受

□

题目. P₆₈ 4.7 不指定接受状态的TM和基本TM的区别是没有接受状态集, 并且把所有的停机格局都看成接受格局. 证明:

- (1) 函数 f 是 Turing 部分可计算的, 当且仅当存在不指定接受状态的 TM 计算 f
- (2) 语言 L 是 r.e. 当且仅当存在不指定接受状态的 TM 接受 L

解答. (1) 我们已知: 一个函数是Turing部分可计算的, 当且仅当存在TM计算 f 而不指定接收状态的TM是TM的一个特殊情况, 因此若存在TM计算 f (这个TM可以是不指定接收状态的TM), 有函数 f 是Turing部分可计算的

现在来证明另一边: 函数 f 是Turing部分可计算的 \Rightarrow 存在不指定接受状态的TM计算 f :

根据已知结论, 因为函数 f 是Turing部分可计算的, 所以存在一个基本TM, 记为 M , 计算 f . 那么可以构造不指定接受状态的TM, 记为 M_1 .

对于 M 的停机在非接受状态的 q , M_1 在此处进入死循环(虽然对所有停机状态都是接受的, 那么只需要这个状态永远不停机就等价于不接受了), 那么这样的 M_1 就可以计算 f . 得证.

(2) 我们已知: 语言 L 是r.e., 当且仅当存在TM接受 L

而不指定接收状态的TM是TM的一个特殊情况, 因此若存在TM接受 L (这个TM可以是不指定接收状态的TM), 有语言 L 是r.e.

现在来证明另一边: 语言 L 是r.e.的 \Rightarrow 存在不指定接受状态的TM接受 L :

根据已知结论, 因为语言 L 是r.e.的, 所以存在一个基本TM, 记为 M , 接受 L . 那么可以构造不指定接受状态的TM, 记为 M_1 .

对于 M 的停机在非接受状态的 q , M_1 在此处进入死循环(虽然对所有停机状态都是接受的, 那么只需要这个状态永远不停机就等价于不接受了), 那么这样的 M_1 就可以接受 L . 得证. \square

题目. 4.8 证明: A 上的语言 L 是r.e.当且仅当存在DTM: M 接受 L , 且 M 有唯一的接受状态 q_Y .

解答. 我们已知: 语言 L 是r.e., 当且仅当存在TM接受 L

而仅有一个接收状态的TM是TM的一个特殊情况, 因此若存在TM接受 L (这个TM可以是仅有一个接收状态的TM), 有语言 L 是r.e.

现在来证明另一边: 语言 L 是r.e.的 \Rightarrow 存在仅有一个接收状态的TM接受 L :

根据已知结论, 因为语言 L 是r.e.的, 所以存在一个基本TM, 记为 M , 接受 L . 那么可以构造仅有一个接收状态的TM, 记为 M_1 .

对于 M 的停机在接受状态的 q , 不妨设这些接受格局不止一个, 那么任意选定其中一个为 q_Y . 那么对于其他的不是 q_Y 的接受格局 q , 在 M_1 中, 这些格局不再是接受格局, 而是“不做操作”且跳转到 q_Y , 那么这样的 M_1 是仅有一个接收状态的TM, 且 M_1 接受 L , 得证. \square

题目. 4.9 证明: A 上的语言 L 是递归的, 当且仅当存在总停机的DTM: M 接受 L , 且 M 有唯一的接收状态 q_Y 和唯一的非接受的停机状态 q_N , 使得当 $x \in A$ 时, M 最终停机在 q_Y ; 当 $x \notin A$ 时, M 最终停机在 q_N .

解答. 我们已知: 语言 L 是递归的, 当且仅当存在总停机的TM接受 L

而有唯一的接收状态 q_Y 和唯一的非接受的停机状态 q_N 的总停机的TM是总停机的TM的一个特殊情况, 因此若存在总停机的TM接受 L (这个TM可以是有唯一的接收状态 q_Y 和唯一的非接受的停机状态 q_N 的总停机的TM), 有语言 L 是递归

现在来证明另一边: 语言 L 是递归的 \Rightarrow 存在有唯一的接收状态 q_Y 和唯一的非接受的停机状态 q_N 的总停机的TM接受 L :

根据已知结论, 因为语言 L 是递归的, 所以存在一个总停机的TM, 记为 M , 接受 L . 那么可以构造有唯一的接收状态 q_Y 和唯一的非接受的停机状态 q_N 的总停机的TM, 记为 M_1 .

对于 M 的所有停机格局, 对于所有的接受格局, 让它们不作操作, 然后跳转到新的一个接受格局为 q_Y ; 对于所有的非接受格局(当然是停机), 让它们不作操作, 然后跳转到新的一个接受格局为 q_N 这样的 M_1 是一个有唯一的接收状态 q_Y 和唯一的非接受的停机状态 q_N 的总停机的TM, 且 M_1 接受 L .

\square