第一章 计算模型

Lecture 1 Models of Computation

字母表、符号串、语言

定义1.1[字母表，alphabet]

字母表 Σ 是一个满足如下条件的有限符号(symbol)集合：两个由 Σ 中符号组成的符号序列等同当且仅当其符号个数相等且对应符号相同

定义1.2[字符串，string]

一个字符串(符号串, string)是一个字母表 Σ 的有限符号序列，空序列 ϵ 也是一个字符串

- 字符串 x 的长度，表示为 $|x|$，是 x 中的符号个数，空串 ϵ 的长度为0

定义1.3[语言，language]

一个语言 L 是字母表Σ的符号串构成的集合，$L \subseteq \Sigma^*$

- Σ, $\emptyset$以及Σ^*都是语言
- 由语言组成的类(class)称为语言类
- 任意字母表都能用$\{0,1\}$字母表编码表示，即$\Sigma = \{0,1\}$

运算符

- 连接(concatenation)运算
  - 符号串 x 和 y 的连接：$x \cdot y = xy$
  - 连接运算满足结合律：$x(yz) = (xy)z$
  - 语言 A 和 B 的连接 $A \cdot B = \{xy | x \in A, y \in B\}$，简写为$AB$
- Kleen 闭包
  - $A^0 = \{\lambda\}$, $A^n = AA^{n-1}$
  - Kleen 闭包 $A^\star = \bigcup_{n=0}^{\infty} A^n$
  - 字母表 Σ 的Kleen闭包 $\Sigma^\star$ 是 Σ 上全部符号串的集合

计算 vs. 计算模型

- 什么是计算的理论模型？
  - 有效计算的严格数学定义
  - 独立于机器（Machine Independent）
- 为什么需要计算模型？
  - 算术的公理化 ⇒ 希尔伯特第二问题 ⇒ 证明论、元数学 ⇒ 哥德尔不完备定理（1931）
  - 丢番图方程的可解问题 ⇒ 希尔伯特第十问题 ⇒ 算法 ⇒ 不可判定性 ⇒ 马蒂亚塞维奇（1970）
  - 什么是可以计算的、多久可以算完等
  - 随机性、量子计算、并行计算等

确定图灵机

阿兰.图灵于1936年提出，图灵机 M 是由如下直观想法实现的

- M 有一个无穷带（infinite tape）存储信息
- M 有一个带头（tape head）可以在带上移动、读写
- 无穷带用输入初始化，带头上有状态标记
- 如果 M 需要存储信息，可以向带上写；如果需要读取某位置的信息，带头需要先移动到该位置
- M 在计算过程中可以进入接受（accept）或拒绝（reject）状态，或者死机（不停机）

确定图灵机—形式化定义

定义1.6[Deterministic Turing Machine, DTM]

一个确定图灵机可以表示为7元组$M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$，其中

- Q是有穷的状态集合
- Σ是有穷的输入符号集合
- Γ是有穷的带上符号集合，显然$\Sigma\subseteq\Gamma$
- q_0是起始状态
- $F\subseteq Q$是停机（halting）状态（接受或拒绝）集合
- $B\in\Gamma\setminus\Sigma$是空字符（blank），带上除了初始字符串外均是空字符
- δ是形如$(Q\setminus F)\times\Gamma\to Q\times\Gamma\times\{R,L\}$的转移函数

$$\delta(q,X)=(p,Y,D),\ D=L或R$$

- 看一个例子

确定图灵机—状态转移图

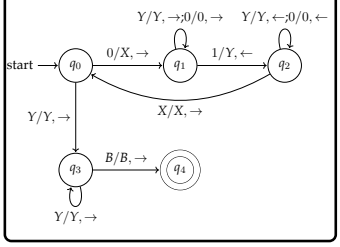
- 图灵机的瞬时描述：$X_1X_2\dots X_{i-1}qX_iX_{i+1}\dots X_n$
- 瞬时描述的迁移：$\delta(q,X_i)=(p,Y,L)$
  - $X_1X_2\dots X_{i-1}qX_iX_{i+1}\dots X_n\vdash X_1X_2\dots pX_{i-1}YX_{i+1}\dots X_n$
  - $\alpha_1q\beta_1\vdash^*\alpha_2p\beta_2$

DTM

$\delta(q_0,0)=(q_1,X,R)$
$\delta(q_0,Y)=(q_3,Y,R)$
$\delta(q_1,0)=(q_1,0,R)$
$\delta(q_1,Y)=(q_1,Y,R)$
$\delta(q_1,1)=(q_2,Y,L)$
$\delta(q_2,0)=(q_2,0,L)$
$\delta(q_2,Y)=(q_2,Y,L)$
$\delta(q_2,X)=(q_0,X,R)$
$\delta(q_3,Y)=(q_3,Y,R)$

$\Rightarrow$

状态转移图



确定图灵机—语言识别器

定义1.7[确定图灵机识别(recognize)的语言]

给定图灵机$M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$，$M$识别（接受）的语言可以表示为

$$L(M)=\{w|w\in\Sigma^*,q_0w\vdash^*\alpha p\beta,p\in F,\alpha,\beta\in\Gamma^*\}$$

- 构造图灵机识别语言
  - $\{0^n1^n|n>0\}$
  - $\{0110\}$

确定图灵机—函数计算器

- 非负整数的编码
  - 非负整数n用字符串0^n表示
  - 对函数$f(n_1,n_2,\dots,n_k)$的编码
    - ✓ $0^{n_1}10^{n_2}1\dots10^{n_k}$表示$f$的$k$个变元
    - ✓ 0^m表示$f(n_1,n_2,\dots,n_k)$的结果m

定义1.9[确定图灵机计算的函数]

图灵机$M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$计算$k$元函数$f(n_1,n_2,\dots,n_k)$，当且仅当$M$在输入串$0^{n_1}10^{n_2}1\dots10^{n_k}$运行后停机并且输出符号为$0^m$，此时称$f$是图灵可计算的

- 构造一个计算$m+n$的图灵机
- 构造一个计算$m\dot{-}n$的图灵机

确定图灵机—一些讨论

- 停机状态中是否可以只包含拒绝（reject）状态？
- 停机状态中是否可以只包含接受（accept）状态？
  - 如何制造“不停机”？
- 是否可以只包含一个停机状态？
- 是否可以在移动中引入“停在原处”？

语言与函数的可计算性定义

- 语言的可计算性

定义1.10[图灵可接受语言]

若对于Σ上语言A存在图灵机M，使得A=$L(M)$，则A称为图灵可接受语言，或递归可枚举语言。

定义1.11[图灵可判定语言]

若Σ上的语言A及其补集$\Sigma^*\setminus A$皆为图灵可接受语言，则A为图灵可判定语言，或递归语言。

语言与函数的可计算性定义

- 函数的可计算性

定义1.12[部分递归函数]

若对于部分函数$f:\Sigma^*\to\Sigma^*$，存在图灵机M使得：(1)把f定义域中w作为输入，M的输出是$f(w)$；(2)把f定义域外的w作为输入，M不停机；则称f为可计算函数，或部分递归函数.

定义1.13[递归函数]

若部分递归函数f的定义域为Σ^*，则f称为完全递归函数，或递归函数.

Church-Turing命题

命题1.1[Church-Turing Thesis]

如果一个函数在某个合理的计算模型上可计算，当且仅当它在图灵机上也是可计算的。

什么是合理的计算模型

- 计算一个函数只要有限条指令
- 每条指令由模型的有限个计算步骤完成
- 执行指令的过程是确定型的

Church-Turing命题的不可证明性

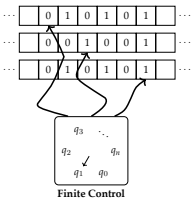
- 不存在“合理”模型的准确定义，命题不可证明
- 历史上被认为是合理的模型，均被证明符合命题

多带图灵机

定义1.14[多带图灵机，Multi-Tape TM]

K-带图灵机可以表示为$M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$，

- Q: 有穷状态集合
- Σ: 有穷输入符号集合
- $\Gamma\subseteq\Sigma$: 有穷带符号集合
- q_0: 起始状态
- $B\in\Gamma\setminus\Sigma$: 空白符号
- $F\subseteq Q$: 终止或接受状态集合
- δ: $(Q-F)\times\Gamma^k\to Q\times\Gamma^k\times\{R,L,S\}^k$



多带TM与单带多道TM的区别

✓多道TM: $\delta(q,(x_1,\ldots,x_k))=(p,(y_1,\ldots,y_k),D)$

✓多带TM: $\delta(q,(x_1,\ldots,x_k))=(p,(y_1,\ldots,y_k),(d_1,\ldots,d_k))$

多带图灵机—语言识别器

定义1.15[多带图灵机接受的语言]

多带图灵机$M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$接受的语言为

$$L(M)=\{w|w\in\Sigma^\star,\alpha_{q_0}\Rightarrow^\star\alpha_p,p\in F\}$$

定义1.16[图灵可接受集]

若对Σ上语言A，存在多带TM M，使$A=L(M)$，则A称为图灵可接受集，或递归枚举集。

定义1.17[图灵可判定集]

若Σ上的语言A及其补集$\Sigma^\star-A$皆为图灵可接受集，则A为图灵可判定集，或递归集。

- 类似的，可以依据多带图灵机定义可计算函数

多带图灵机—与单带图灵机等价性

定理1.2[多带图灵机与单带图灵机的等价性]

多带TM与单带TM是计算等价的.

证明：

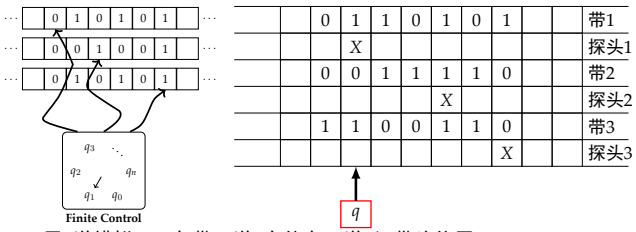
- 由于单带TM是多带TM的特例，只需要证明：给定一个多带TM M，可构造一个单带TM N，N能够完成M的计算
- 设M=(Q, Σ, Γ, δ, q_0, B, F)
- 构造N=(Q', Σ, Γ', δ', p_0, B', F')，使之与M等价

多带图灵机—与单带图灵机等价性

用N的多道单带模拟M的K带

M=(Q, Σ, Γ, δ, q_0, B, F)

N=(Q', Σ, Γ', δ', p_0, B', F')



- N用2道模拟M一条带：道1存信息，道2记带头位置；
- $\Gamma'=(\Gamma\times\{B,X\}^k,X\notin\Gamma)$;
- $B'=\{B,B,\ldots,B\}$: 由$2k$个B构成的向量

多带图灵机—与单带图灵机等价性

定理1.3[单带与多带图灵机时间关系]

对于任何多带图灵机M，存在单带图灵机N，使得$L(M)=L(N)$，且对于任意输入w，有$Time_N(w)=O(Time_M(w)^2)$.

证明：N按照如下模拟M的一个动作：

- 从左到右扫描k个带头指向的符号; (k个右移)
- 修改带符号为$(y_1,y_2,\ldots,y_k)$; (k个左移)
- 移动带头, 即修该X的位置; (k个右移)
- 形成新状态$(p,B,\ldots,B)$, 带头到最左; (k个左移)

N模拟一个M的动作需要移动数是最远两个带头距离; M的第i个移动可能导致的最远两个带头距离为$2i$.
(带头一个一直向左,一个一直向右).

N模拟M的$Time_M(w)$个移动最多$\sum_{i=1}^{Time_M(w)}4\times 2i=O(Time_M(w)^2)$

即,$Time_N(w)=O(Time_M(w)^2)$.

不确定图灵机

定义1.20[不确定图灵机，NTM]

不确定图灵机可以表示为$M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$，其中

Q：有穷状态集合

Σ：有穷输入符号集合

$\Gamma\subseteq\Sigma$：有穷带符号集合

q_0：起始状态

$B\in\Gamma-\Sigma$：空白符号

$F\subseteq Q$：终止或接受状态集合

δ：$(Q-F)\times\Gamma\to 2^{Q\times\Gamma\times\{R,L\}}$

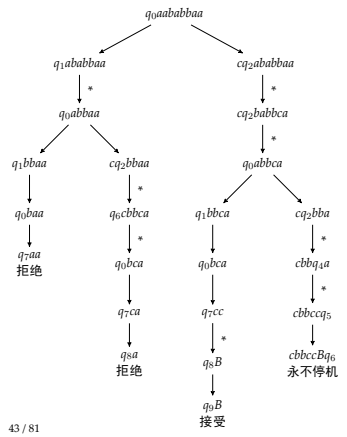
即$\delta(q,x)=\{(p,y,D)|p\in Q,y\in\Gamma,D\in\{L,R\}\}$

NTM与DTM的区别：DTM: $\delta(q,x)=(p,y,D)$

NTM: $\delta(q,x)=\{(p,y,D)|p\in Q,y\in\Gamma,D\in\{L,R\}\}$

不确定图灵机—瞬时描述

- NTM的瞬时描述同DTM;
- 计算过程是瞬时描述的树;
- 包含接受状态的瞬时描述称接受瞬时描述;
- 从初始瞬时描述到接受瞬时描述的路是接受路.



不确定图灵机

定义1.21[不确定图灵机接收的语言]

不确定图灵机 M=(Q, Σ, Γ, δ, q_0, B, F)接受的语言为

$$L(M) = \{w | w \in \Sigma^\star, \text{ 存在序列} q_0 w \Rightarrow^\star \alpha p \beta, p \in F\}$$

定义1.22[NTM可计算函数]

NTM M=(Q, Σ, Γ, δ, q_0, B, F)可计算函数 $f: \Sigma^\star \to \Sigma^\star$,如果

1. M 的计算树上的所有接受瞬时描述中的输出皆相同;
2. 若对于输入x, M 输出 $f(x)$ 当且仅当 f 在 x 处有定义.

定义1.23[NTM完全可计算函数]

如果NTM可计算函数 f 的定义域为Σ^*, 则称 f 是NTM完全可计算的函数.

不确定图灵机—与确定图灵机等价

定理1.4[不确定图灵机与确定图灵机的等价性]

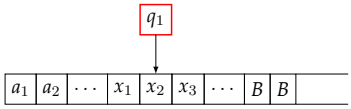
NTM与DTM是计算等价的.

证明：由于DTM是NTM的特例，只需要证明:对于任意一个NTM M，可构造一个DTM N，N能够完成M的计算

- 设M=(Q, Σ, Γ, δ, q_0, B, F)，构造N=(Q', Σ, Γ', δ', p_0, B, F')
- 思想：模拟所有可能移动序列

单向无穷带图灵机

- 直观模型



定义1.24[单向无穷带图灵机]

单向无穷带TM是一个系统M=(Q,Σ,Γ,δ,q_0,B,F),其中Q,Σ,Γ,q_0, B,F与双向无穷带图灵机相同,$\delta(q,x) = (p, Y, D)$,$D = L/R$, 达到左界时不可左移.

- 计算能力

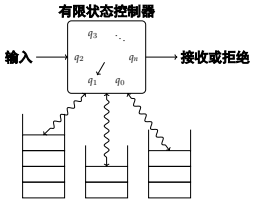
定理1.6[单双向无穷带图灵机等价性]

单向无穷带图灵机与双向无穷带图灵机等价.

证明:

多栈图灵机

- 直观模型



定义1.25[k栈图灵机]

k栈图灵机可以表示为$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$，$Q$是状态集，$\Sigma$是输入字母表，$\Gamma$是栈字母表，$F$是接受状态集，$q_0$是起始状态，$Z_0$是栈底符号，令$X_i$是栈$i$的栈顶符号$\delta(q, a, X_1, \ldots, X_k) = (p, \gamma_1, \ldots, \gamma_k)$，其中$a \in \Sigma \cup \{\epsilon\}$，$\gamma_i \in (\text{push } x \in \Gamma, \text{pop})$.

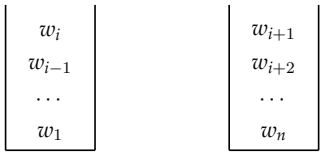
多栈图灵机

定理1.9[多栈图灵机与图灵机计算能力的等价性]

图灵机与2-栈图灵机计算能力等价.

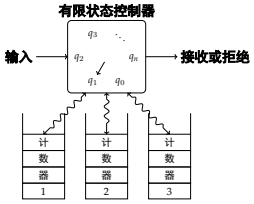
证明:

基本思想是:用2个栈模拟TM的带,栈1存储TM带头左侧的符号行,栈2存储TM带头所指和右侧符号行,符号行左右侧的无限个B不存储在栈中.



计数图灵机

- 直观模型



定义1.26[计数图灵机]

计数图灵机是$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, Q是状态集, Σ是输入字母表, Γ是计数器字母表, F是接受状态集, q_0是起始状态, 令C_i是第i计数器, $\delta(q, a, C_1, C_2, \ldots, C_i, \ldots, C_k) = (p, o_1, o_2, \ldots, o_i, \ldots, o_k)$, $q \in \Sigma \cup \{\epsilon\}$, $o_i \in \{+, -\}$表示计数器C_i加1或减1, 但计数器不能为负.

计数图灵机

定理1.10[计数图灵机与图灵机的计算能力等价]

图灵机与3-计数图灵机计算能力等价.

证明: 图灵机与2-栈图灵机等价，只需证明2-栈图灵机可以由3-计数图灵机模拟.

设S是2-栈机, 如下构造3-计数器机C, 模拟S:

1. 设S具有$r-1$个带符号, 可编码为1、2、. . . 、$r-1$. 栈符号串$X_1X_2 \ldots X_n$可视为r-进制数$X_1 + rX_2 + r^2X_3 + \cdots + r^{n-1}X_n$.
2. C的计数器C_1和C_2存储S的2个栈符号串, C_3辅助计算.
3. C_1存储S的栈1的初始符号串, C_2存储S的栈2初始符串.
4. C模拟S的弹出栈顶符号、改变栈顶符号、压符号入栈:

计数图灵机

证明续:

4. C模拟S的弹出栈顶符号、改变栈顶符号、压符号入栈:

A. 弹出栈i的顶符号, 即计算$C_i = (C_i - X_1)/r$, $i = 1$或2:
   a. $C_3 = 0$, 反复执行[$C_i = C_i - (r$个1)]和[$C_3 = C_3 + 1$], 直至$C_i = X_1$;
   b. C_i清0, 反复执行[$C_3 = C_3 - 1$]和[$C_i = C_i + 1$], 直至$C_3 = 0$.

B. 栈i的顶X_1改为Y, 即计算$C_i = C_i - X_1 + Y$:
   a. $C_3 = 0$, 反复执行[$C_i = C_i - (r$个1)]和[$C_3 = C_3 + 1$], 直至$C_i = X_1$;
   b. C_i变为Y, 反复执行[$C_3 = C_3 - 1$]和[$C_i = C_i + r$], 直至$C_3 = 0$.

C. 压X到栈i, 即计算$C_i = rC_i + X$:
   a. $C_3 = 0$开始, 反复执行[$C_i = C_i - 1$]和[$C_3 = C_3 + r$], 直至$C_i = 0$;
   b. 反复执行[$C_3 = C_3 - 1$]和[$C_i = C_i + 1$], 直至$C_3 = 0$, $C_i = C_i + X$.

计数图灵机

定理1.11[计数图灵机与图灵机的计算能力等价]

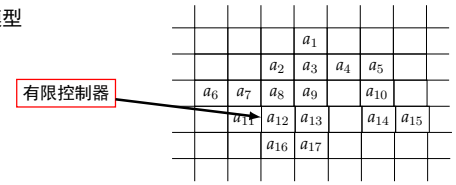
图灵机与2-计数图灵机计算能力等价.

证明: 只需证明可用2-计数器机C'模拟3-计数器机C.
C'用计数器C_1存储整数$2^i3^j5^k$表示3个计数器的i、j、k. C'的计数器C_2辅助计算.C'如下模拟C:

1. i、j、k加1:
   C_1=$C_1\times 2$实现$i+1$; C_1=$C_1\times 3$实现$j+1$; C_1=$C_1\times 5$实现$k+1$.
2. 判定i、j或k是否为0:
   C_1能被2整除$\Leftrightarrow i$不为0;
   C_1能被3整除$\Leftrightarrow j$不为0;
   C_1能被5整除$\Leftrightarrow k$不为0.
3. i、j、k减1:
   C_1=$C_1/2$实现$i-1$; C_1=$C_1/3$实现$j-1$; C_1=$C_1/5$实现$k-1$.

多维图灵机

▷ 直观模型



定义1.27[2-维图灵机]

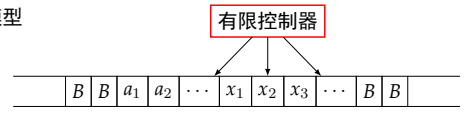
2-维图灵机是一个系统M=$(Q,\Sigma,\Gamma,\delta,q_0,B,F)$,其中$Q,\Sigma,\Gamma,q_0,B,F$与确定图灵机相同,$\delta(q,x)=(p,Y,D)$,$D=\{L,R,U,D\}$

定理1.12

多维图灵机与确定图灵机等价.

多头图灵机

▷ 直观模型



定义1.28[多头图灵机]

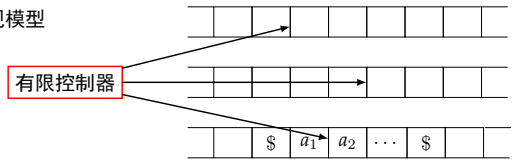
多头图灵机可表示为$M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$,其中$Q,\Sigma,\Gamma,q_0,B,F$与确定图灵机相同, $\delta(q,(x_1,\ldots,x_k))=(p,(y_1,\ldots,y_k),(d_1,\ldots,d_k))$,$D=\{L,R\}$.

定理1.13[多头图灵机的计算能力]

多头图灵机与确定图灵机等价.

离线图灵机

▷ 直观模型



定义1.29[离线图灵机]

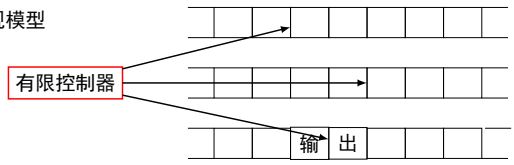
有一条带为输入带的多带图灵机称为离线图灵机，其中输入带是只读带，且具有界限，输入带头必须在界限内移动.

定理1.14[离线图灵机的计算能力]

离线图灵机与确定图灵机等价.

枚举图灵机

▷ 直观模型



定义1.30[枚举器图灵机]

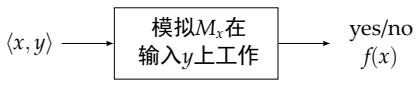
有一条带为输出带的多带图灵机称为枚举器图灵机，用于产生语言$G(M)$，输出带符号写入就不能更改，输出带头必须单向移动.

定理1.15[枚举器图灵机的计算能力]

语言L是递归可枚举的，当且仅当存在枚举TM M使得$L=G(M)$.

通用图灵机

▷ 什么是通用图灵机



▷ 有关DTM的假设
- DTM $M=(Q,\{0,1\},\{0,1,B\},\delta,q_0,F=\{q_f\})$
- M由δ确定
  - ✓ Q是δ定义中的所有状态(包括q_0和q_f)
  - ✓ Γ是δ定义中的所有符号
- 编码M只需编码所有δ表达式

通用图灵机

▷ DTM M 的编码
- ▷ 令Q=$\{q_0,\ldots,q_n\}$,X_0=0,X_1=1,X_2=B,D_0=L,D_1=R
  - δ的第i个表达式$\delta(q_i,X_j)=(q_k,X_l,D_h)$的编码$code_i$为

$$0^{i+1}10^{j+1}10^{k+1}10^{l+1}10^{h+1}$$

  - 设M具有m个δ表达式,M的编码为

$$code_1 11 code_2 11 \ldots 11 code_m$$

- ▷ 注意
  - δ表达式的不同顺序给出不同的编码,M的编码不唯一
  - 可以把如下符号行视为TM编码

$$code_1 11 code_2 11 \ldots 11 code_m 111 z, z\in\{0,1\}^\star$$

则每个DTM由无穷个编码且长度无上界.

通用图灵机

定理1.16[通用TM的存在性]

存在DTM M_u，对于任何输入$\langle x,y\rangle$=$0^{|y|}1yx$，M_u模拟x表示的TM M_x以y为输入的计算过程，使得

$$L(M_u) = \{\langle x,y\rangle | y \in L(M_x)\}$$

证明

- 构造4带DTM M_u:1条输入带,3条工作带.
- M_u如下工作：x=$code_1 11 code_2 11 \ldots 11 code_m 111 z, z \in \{0,1\}^\star$
  - 解码输入$\langle x,y\rangle$,分解y和x中111之前的有用部分w,分别写入带2和带3;
  - 如果(w不是单带TMM_w编码) $\vee$ (M_w不是确定型)则停机并拒绝$\langle x,y\rangle$；否则继续;
  - 在带3和带4上模拟M_w在y上的计算过程.

通用图灵机

- 推广到多带和不确定图灵机

定理1.17

存在DTM M_u,对于任何输入$\langle x,y\rangle$=$0^{|y|}1yx$,M_u模拟x表示的多带TM M_x以y为输入的计算过程,使得

$$L(M_u) = \{\langle x,y\rangle | y \in L(M_x)\}$$

证明. 先用单带DTM模拟M_x,再用命题2结果

定理1.18

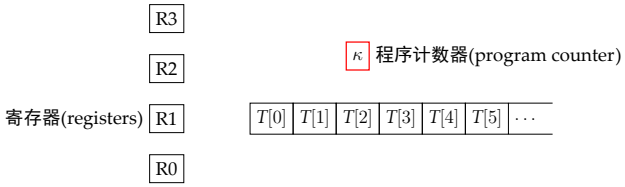
存在DTM M_u,对于任何输入$\langle x,y\rangle$=$0^{|y|}1yx$,M_u模拟x表示的NTM M_x以y为输入的计算过程,使得

$$L(M_u) = \{\langle x,y\rangle | y \in L(M_x)\}$$

证明. 先用单带DTM模拟M_x,再用命题2结果

随机访问图灵机

随机访问图灵机的直观表示：



- 寄存器初始化为0；程序计数器初始化为1；
- 机器最开始执行第一条指令
- 每次指令执行后κ加1，除非指令对κ赋值
- 指令一直运行，直到遇到停机指令(halt)

随机访问图灵机

指令集合

instruction	operand	semantics
read	j	$R_0 := T[R_j]$
write	j	$T[R_j] := R_0$
store	j	$R_j := R_0$
load	j	$R_0 := R_j$
load	$=c$	$R_0 := c$
add	j	$R_0 := R_0 + R_j$
add	$=c$	$R_0 := R_0 + c$
sub	j	$R_0 := \max\{R_0 - R_j, 0\}$
sub	$=c$	$R_0 := \max\{R_0 - c, 0\}$
half		$R_0 := \lfloor \frac{R_0}{2} \rfloor$
jump	s	$\kappa := s$
jpos	s	if $R_0 > 0$ then $\kappa := s$
jzero	s	if $R_0 = 0$ then $\kappa := s$
halt		$\kappa := 0$

随机访问图灵机

定义1.31[随机访问图灵机(random access Turing machine)]

一个随机访问图灵机被表示为 $M = (k, \Pi)$，其中 $k > 0$ 表示寄存器(registers)的数目，$\Pi = (\pi_1, \pi_2, \ldots, \pi_p)$ 是程序(program)，由一个有穷的指令(instructions)序列构成，假设最后一条指令 π_p 总是 halt 指令。

定义1.32[随机访问图灵机的瞬时描述]

一个随机访问图灵机 (k, Π) 的瞬时描述可以表示为一个 $k+2$ 元组

$$(\kappa, R_0, \ldots, R_{k-1}, T)$$

- κ是取值$[0,p]$之间的程序计数器；如果$\kappa = 0$则是停机状态；
- R_j是寄存器的值；
- T是带上内容，由一组(i,m)构成，(i,m)表示带上第i个格存储的是m

随机访问图灵机

- 假设**encode**将Σ中的符号映射到$\{0, \ldots, |\Sigma| - 1\}$
- 随机访问图灵机$M = (k, \Pi)$接收输入$w = a_1 a_2 \ldots a_n$，初始瞬时描述为

$$(\kappa = 1, R_0 = 0, \ldots, R_{k-1} = 0, T)$$

其中，$T = \{(1, \textbf{encode}(a_1)), (2, \textbf{encode}(a_2)), \ldots, (n, \textbf{encode}(a_n))\}$

- 用作语言识别器时，停机后，$R_0 = 1$表示接受，$R_0 = 0$表示拒绝
- 用作函数计数器时，停机后，$T[1]$开始的内容（结束符B）是计算结果

构造的例子

- 利用随机访问图灵机的指令集构造乘法multiply操作
- 构造 **while** $x > 0$ **do** $x := x - 3$

定理1.19[随机访问图灵机与图灵机的等价性]

图灵机与随机访问图灵机计算能力是等价的；假设随机访问图灵机运行时间为$T(n)$，那么图灵机模拟随机访问图灵机的代价是$poly(T(n))$。

文法

- 文法（Grammars）是一种语言生成器
- 上下文无关文法回顾：
  - 字母集合 V 被分为终结符集合 Σ 和非终结符集合 $V \setminus \Sigma$
  - 每条生成规则形如 $A \to u$，$A \in V \setminus \Sigma$ 并且 $u \in V^*$
- 文法对生成规则的左侧要求：至少包含一个非终结符

定义1.33[文法，Grammars]

一个文法是一个4元组 $G = (V, \Sigma, R, S)$，V 是字母集，Σ 是终结符集合，$S \in V \setminus \Sigma$ 是起始符，R 是生成规则集合，$R \subseteq (V^*(V \setminus \Sigma)V^*) \times V^*$

- $(u, v) \in R$，记作 $u \to v$
- $u \Rightarrow_G v$ 当且仅当 $w_1, w_2 \in V^*$ 和 $u' \to v'$ 使 $u = w_1 u' w_2$ 且 $v = w_1 v' w_2$
- $\Rightarrow_G^*$ 是 $\Rightarrow_G$ 的自反、传递闭包

文法

定义1.34[文法生成的语言]

字符串 $w \in \Sigma^*$ 是文法 G 生成的当且仅当 $S \Rightarrow_G^* w$；文法 G 生成的语言定义为 $L(G) = \{w | S \Rightarrow_G^* w\}$

例1.6[生成语言 $\{a^n b^n c^n : n \geq 1\}$ 的文法]

$G = (V, \Sigma, R, S)$，$V = \{S, a, b, c, A, B, C, T_a, T_b, T_c\}$，$\Sigma = \{a, b, c\}$，$R$：

- $S \to ABCS$, $S \to T_c$
- $CA \to AC$, $BA \to AB$, $CB \to BC$
- $CT_c \to T_c c$, $CT_c \to T_b c$, $BT_b \to T_b b$, $BT_b \to T_a b$, $AT_a \to T_a a$
- $T_a \to \epsilon$

定理1.20

语言 L 是由文法生成的当且仅当 L 是递归可枚举的

文法

定义1.35[文法计算的函数]

令 $G = (V, \Sigma, R, S)$ 是一个文法，$f : \Sigma^* \to \Sigma^*$ 是一个函数，G 计算 f，如果对任意 $w, v \in \Sigma^*$ 有 $SwS \Rightarrow_G^* v$ 当且仅当 $v = f(w)$。函数 f 被称作语法可计算的（grammatically computable）

定理1.21

函数 f 是图灵可计算的当且仅当是语法可计算的。

数值函数

直观想法，考虑如下函数：

$$f(m, n) = m \cdot n^2 + 3 \cdot m^{2 \cdot m + 17}$$

- 这个函数是否可以计算？为什么？
- 直观上，指数函数可以被递归地计算，$m^n = m \cdot m^{n-1}$
- 利用基本函数，构造复杂函数

数值函数

定义1.36[基本函数，basic functions]

形如 $\mathbb{N}^k \to \mathbb{N}$ 的基本函数包括：

- k-ary zero function. $\text{zero}_k(n_1, \ldots, n_k) = 0$
- j-th k-ary identity function. $\text{id}_{k,j}(n_1, \ldots, n_k) = n_j$
- successor function. $\text{succ}(n) = n + 1$

定义1.37[函数的合成，composition]

令 $g : \mathbb{N}^k \to \mathbb{N}$，$h_1, \ldots, h_k$ 是 l-元函数，g 和 h_i 的合成可以定义为

$$f(n_1, \ldots, n_l) = g\big(h_1(n_1, \ldots, n_l), \ldots, h_k(n_1, \ldots, n_l)\big)$$

定义1.38[函数的递归定义，composition]

令 $g : \mathbb{N}^k \to \mathbb{N}$，$h : \mathbb{N}^{k+2} \to \mathbb{N}$，由 g 和 h 递归定义的函数 f 是 $(k+1)$-元函数

$$f(n_1, \ldots, n_k, 0) = g(n_1, \ldots, n_k)$$
$$f(n_1, \ldots, n_k, m+1) = h\big(n_1, \ldots, n_k, m, f(n_1, \ldots, n_k, m)\big)$$

数值函数

定义1.39[primitive recursive functions]

基础递归函数由基本函数以及利用合成和递归构造的函数组成。

定义1.40[最小化函数，minimalization]

令 g 是一个 $k+1$-元函数，g 的最小化(minimalization)是一个 k-元函数 f

$$f(n_1, \ldots, n_k) = \begin{cases} \text{最小的 } m \text{ 使得 } g(n_1, \ldots, n_k, m) = 1，\text{如果} m \text{存在} \\ \text{否则为} 0 \end{cases}$$

最小化函数表示为 $\mu m[g(n_1, \ldots, n_k, m) = 1]$；如果对任意 $n_1, \ldots, n_k$ 都有一个 m 使得 $g(n_1, \ldots, n_k, m) = 1$，$g$ 被称作可最小化(minimalizable).

定义1.41[μ-recursive]

函数 f 是 μ-recursive的，如果 f 可由基础函数通过合成(composition)、递归(recursive)和最小化(minimalization)构造

数值函数

定理1.22[数值函数与图灵机的等价性]

数值函数 f 是 μ-recursive的，当且仅当 f 可以由图灵机计算。