计算理论基础

教师：李元 yuan_li@fudan.edu.cn

助教：

- 5n+1 高怡雯
- 2 李可欣
- 3 李阳年
- 4 万顺
- 5 张双俊

期末：60%

无期中考

作业不迟交

可以3个薄本/订好的纸

计算模型 Computation Model

算法：图灵机　　可计算性 Computability　　问题 { 可解的 / 不可解的

↑ 高效计算？

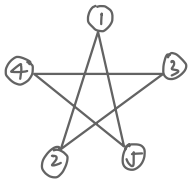
停机问题不能被计算.

复杂性理论 Complexity　　困难程度：消耗的资源（时间、空间）

易算的　　难算的

NP完全（NPC）：如果其中一个NP问题可解，则其他NP问题都可解.　e.g. 汉密尔顿回路

图的描述：$G=(V,E)$



$G=(\{1,2,3,4,5\},\{(1,2),(2,3),(3,4),(4,5),(5,1)\})$

Big O Notation

Def 1.1 Let $f, g: \mathbb{N} \to \mathbb{R}$. Write $f = O(g)$ iff $(\exists C > 0)(\exists N)(\forall n \geq N)(|f(n)| \leq C \cdot g(n))$ (常数)

O：小于等于　$n \geq N$ 时总有 $|f(n)| \leq C \cdot g(n)$（给出上界）

Ω：大于等于　（给出下界）

Θ：$\begin{cases} f = O(g) \\ f = \Omega(g) \end{cases} \Rightarrow f = \Theta(g)$　$(\exists C_1, C_2 > 0)(\exists N)(\forall n \geq N)(C_1 g(n) \leq |f(n)| \leq C_2 g(n))$

（给出上+下界）

o：$f = o(g)$ if $(\forall \varepsilon > 0)(\exists N)(\forall n \geq N)(|f(n)| \leq \varepsilon \cdot g(n))$　在渐近情况下，$f \leq g(n)$

For example：

$f(n) = 6n^4 - 2n^3 + 5 = O(n^4)$

$|f(n)| \leq 6n^4 + 2n^3 + 5 \leq 6n^4 + 2n^4 + 5n^4 = 13n^4 \Rightarrow f(n) = O(n^4)$　（超级粗放缩）

根据定义，$f(n) = O(n^5)$，$f(n) = O(n^6)$．$f(n) = \Omega(n^4)$．$f(n) = \Theta(n^4)$

$1 + 2 + \cdots + n = \frac{n(n+1)}{2} = O(n^2)$　或 $\frac{n(n+1)}{2} = \frac{1}{2}n^2 + O(n)$（便计算变干净）

$1^2 + 2^2 + \cdots + n^2 = \frac{1}{6}n(n+1)(2n+1) = O(n^3)$

Practice.

1) $5 + 0.001n^3 + 0.25n = O(n^3)$

2) $500n + 100n^{1.5} + 50 n \log_{10} n = O(n^{1.5})$

3) $n^2 \log_2 n + n(\log_2 n)^2 = O(n^2 \log n)$

4) $3\log_8 n + \log_2 \log_2 n = O(\log n)$

Prop 1.2（命题）

1) $f_1 = O(g_1)$，$f_2 = O(g_2) \Rightarrow f_1 \cdot f_2 = O(g_1 \cdot g_2)$

2) $f \cdot O(g) = O(f \cdot g)$

3) $f_1 = O(g_1)$，$f_2 = O(g_2) \Rightarrow f_1 + f_2 = O(\max(g_1, g_2))$

4) $f_1 = O(g)$，$f_2 = O(g) \Rightarrow f_1 + f_2 = O(g)$ →（常数被 O 吃掉）

5) if k is a constant and $f = O(g)$, then $k \cdot f = O(g)$.

constant：$O(1)$（常数）　e.g. $1 + \frac{1}{2} + \cdots + \frac{1}{n} = \ln n + O(1)$　$1 + \frac{1}{2^2} + \cdots + \frac{1}{n^2} = O(1)$（收敛）

double logarithmic：$O(\log\log n)$（双对数．不用写底）

$O(\log n)$

polylogarithmic：$O((\log n)^c)$，$c > 0$　/　$O(\log^{O(1)} n)$

Linear : $O(n)$ (线性)

quasiLinear : $O(n \log^c n)$, $c>0$ / $O(n \log^{O(1)} n)$

quadratic : $O(n^2)$

polynomial : $O(n^c)$, $c>0$ (理论上有多项式时间的算法都被视作可解的)

exponential : $O(c^n)$, $c>1$

factorial : $O(n!)$

Def 1.3 $f = \omega(g)$ if $(\forall c>0)(\exists N)(\forall n \geq N)(f(n) \geq c \cdot g(n))$

(在渐近意义下左大于等于右)

$f = \theta(g)$ (等价于 $f \sim g$) if $(\forall \varepsilon>0)(\exists N)(\forall n \geq N)(|f(n)-g(n)| < \varepsilon \cdot g(n))$

O: big O ; o: little o ; Θ: big theta ; θ: small θ (on the order of)

Ω: big Omega ; ω: small theta.

Alphabets and Languages

Def 1.4 An Alphabet is a set of symbols.

Roman Alphabet : $\{a, b, \ldots, z\}$

binary Alphabet : $\{0, 1\}$

Def 1.5 A string (over an alphabet) is a finite sequence of symbols from the alphabet.

Empty string : a string of no symbols, denoted by ε.

The set of all strings is denoted by Σ^*.

Denote by Σ^n the set of all strings of length n.

So, $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$.

For example, $\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, \ldots\}$

Denote the length of a string ω by $|\omega|$. (e.g. $|\varepsilon| = 0$, $|0110| = 4$)

Def 1.6 Two strings over the same alphabet can be combined by the operation of concatenation. The concatenation of X and Y is denoted by XY.

(以上定义是为了对问题进行编码)

if $w = xy$, then $|w| = |x| + |y|$.

Def 1.7 A string v is a substring of w iff ∃ string x and y such that $w = xvy$.

if $w = xv$ for some x, the v is the suffix of w. (后缀: suffix)

if $w = vy$ for some y, then v is the preffix of w. (前缀: preffix)

Def 1.8 The string w^n is defined

$$w^0 = \varepsilon.$$

$$w^{i+1} = w^i w \quad \text{for each } i \geq 0$$

example: $01^0 = \varepsilon$, $01^1 = 01$, $01^2 = 0101$

Def 1.9 The reverse of string w denoted w^R is the string "spelled backwards".
(逆转字符串, 启动!) $w = av$, $w^R = v^R a$.

A formal defination can be given by induction on Length.

1) if $w = \varepsilon$, $w^R = w = e$.

2) if w has Length $n+1$. where $w = ua$, $(a \in \Sigma)$ then $w^R = au^R$.

Def 1.10 Language is a set of strings over an alphabet. (语言是字符串的集合)

That is, $L \subseteq \Sigma^*$. For example, $\emptyset$, Σ^*, Σ are all languages. (拥有集合的特性)

$\Sigma = \{0, 1\}$, $Even = \{0, 10, 100, 110, 1000, \dots\}$

$Prime = \{10, 11, 101, 111, 1011, \dots\}$ → 质数集合是一个语言.

$Palindrome = \{$ all strings w s.t. $w^R \} = \{\varepsilon, 0, 1, 00, 11, 000, 010, \dots\}$
(回文)

Def 1.11 Let L be a language. The complement of L denoted by $\bar{L}$. is $\Sigma^* - L$. So $\bar{\bar{L}} = L$.

Since L is a set, we can define union($\cup$), intersection($\cap$), and difference.

The concatenation of L_1 and L_2 is defined by $L_1L_2 = \{w \in \Sigma^*, w = xy$ for some $x \in L_1$ and $y \in L_2\}$.

Def 1.12 The Kleene star of L, denoted by L^*, is the set of strings obtained by concatenation zero or more strings from L.

Thus, $L^* = \{w \in \Sigma^*, w = w_1 w_2 \cdots w_k, k \geq 0$, and $w_1, w_2, \ldots, w_k \in L\}$

For example, if $L = \{01, 1, 100\}$, then $1100011100 \in L^*$.

Write L^+ for the language LL^*. ($L^+ = L^* - \varepsilon$)

Equivalently, $L^+ = \{w \in \Sigma^* : w = w_1 w_2 \cdots w_k, k \geq 1$, and $w_1, w_2, \ldots w_k \in L\}$

Encoding of problems (问题的编码)

Example 1) (integer multiplication) Given 2 nonnegtive integers compete xy.

2) (Primality testing) Given $n \in \mathbb{N}$, decide if n is a prime.

3) (Hamiltonian cycle) Given an undirected graph G, test if G has a Ham cycle.

(Decision Problem → Language)

e.g. 2) 设质数的二进制编码集 L_{prime}. 给出 n 的二进制字符串 x. 判 $x \in ?\ L_{prime}$

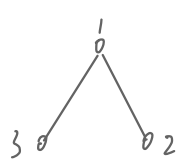
(Computation Problem → Function) $f: \Sigma^* \to \Sigma^*$

By encoding, any decision problem is a language, any computation problem is a function from Σ^* to Σ^*.

Often time, the way of encoding does not matter.

By preprocessing, one can switch between encodings.

e.g.



n=3

	1	2	3
1	0	1	1
2	1	0	0
3	1	0	0

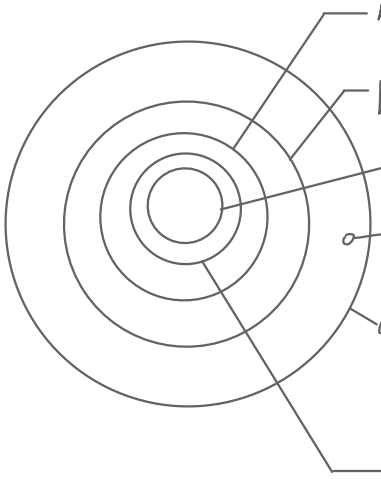
adj. matrix

n=3

(1,2) (1,3)

adj. List

2024.3.7
Week 2

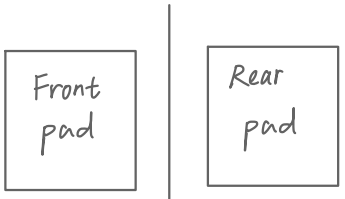


Upper bound： Given L, prove L is decidable in time T(n).

Lower bound： Given L, prove L is not decidable in Time T(n).

Finite Automaton

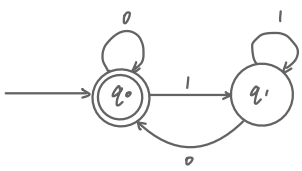
有限自动机 (?改动) → 图灵机(通用计算模型)



Status：
front：门外有人
rear：门内有人
both：两侧有人
nether：两边没人

判断字符串是否以0为结尾：

$L = \{w \in \{0,1\}^* \mid w = w_1 w_2 \dots w_n, w_n = 0\}$



: Accept state

$Q = \{q_0, q_1\}$

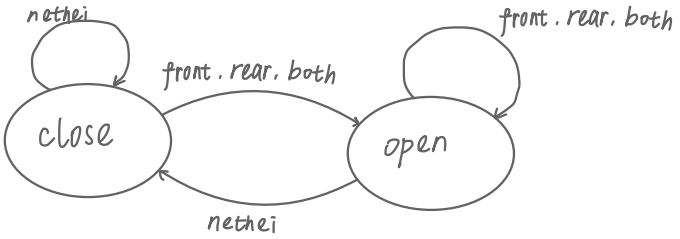
$\Sigma = \{0, 1\}$

δ：

	0	1
q_0	q_0	q_1
q_1	q_0	q_1

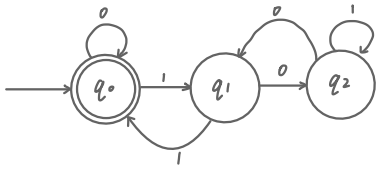
$Q \times \Sigma \to Q$

w = 011 reject ; w = 100 accept



$L = \{w \in \{0,1\}^* \mid w = w_1 w_2 \cdots w_n, w_n + 2w_{n-1} + 2^2 w_{n-2} + \cdots + 2^{n-1} w_1 \equiv 0 \pmod 3\}$

判断一个二进制数是否为3的倍数.



e.g. 10110
(22 % 3 = 1)

$X \equiv 0 \pmod 3$ $X \equiv 1 \pmod 3$ $X \equiv 2 \pmod 3$

$2X \equiv 0 \pmod 3$ $2X \equiv 2 \pmod 3$ $2X \equiv 1 \pmod 3$

$2X+1 \equiv 1 \pmod 3$ $2X+1 \equiv 0 \pmod 3$ $2X+1 \equiv 2 \pmod 3$

Def 2.1 A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where：

(q_0：起始状态)

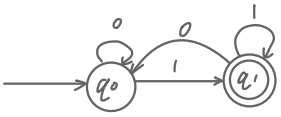
Q：状态集 Σ：字母表 δ：转移函数 F：接受状态

1. Q is a finite set called the states.
2. Σ is the alphabet.
3. $\delta: Q \times \Sigma \to Q$ is the transition function.
4. $q_0 \in Q$ is the start state.
5. $F \subseteq Q$ is the set of accept states.

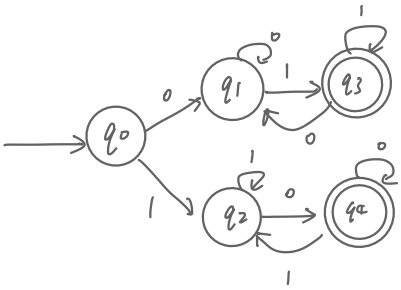
Def 2.3 if L is the set of strings that M accepts, we say L is the language of M, and write $L(M)=L$

We say M recognizes / decides / accepts L.

If M accepts no string, it recognizes one language, namely the empty language. (空语言)



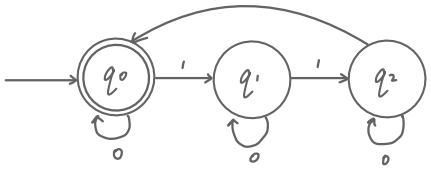
$L(M)=\{w\in\{0,1\}^*,\ w=w_1w_2\cdots w_n,\ w_n=1\}$ 接受以1结尾的字符串



$L(M)=\{w\in\{0,1\}^*,\ w=w_1w_2\cdots w_n,\ w_1\neq w_n\}$

input：开头为1：接受以0结尾；开头为0：结尾为1

$L=\{w\in\{0,1\}^*,$ the number of 1's is a multiple of 3$\}$



$L=\{w\in\{0,1\}^*,\ w^R=w\}$ (回文串, Palindrome)

无法由有限自动机判定.

Def 2.2 Let $M=(Q,\Sigma,\delta,q_0,F)$ be a finite automaton. Let $w=w_1w_2\cdots w_n$ be a string, where each $w_i\in\Sigma$. Then M accepts w if there is a sequence of states $r_0r_1\cdots r_n\in Q$ such that

1. $r_0=q_0$
2. $\delta(r_i,w_{i+1})=r_{i+1}$, for $i=0,1,\cdots,n-1$
3. $r_n\in F$

Def 2.4 $L\subseteq\Sigma^*$ is a regular language if there is a finite automaton that accept L.

(正则语言：可由有限自动机接受)

Let $A,B\subseteq\Sigma^*$. Define:

正则语言对这些运算封闭.

- (union) $A\cup B=\{x\in\Sigma^*,\ x\in A$ or $x\in B\}$
- (concateration) $A\circ B$ or $AB=\{xy,\ x\in A,\ y\in B\}$
- (star) $A^*=\{x_1x_2\cdots x_k,\ k\geq 0,\ x_i\in A\}$

$\Sigma=\{0,1\}$. $A=\{\varepsilon,0,00,000,\cdots\}$, $B=\{\varepsilon,1,11,111,\cdots\}$

$AB=\{0^i1^j,\ i,j\geq 0\}$. $A^*=A$, $B^*=B$, $(AB)^*=\Sigma^*$. (可以表示所有01串)

Thm 2.5 If A_1, A_2 are regular languages, so is $A_1 \cup A_2$.

Proof: Let $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ accepts A_1, and $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ accepts A_2,

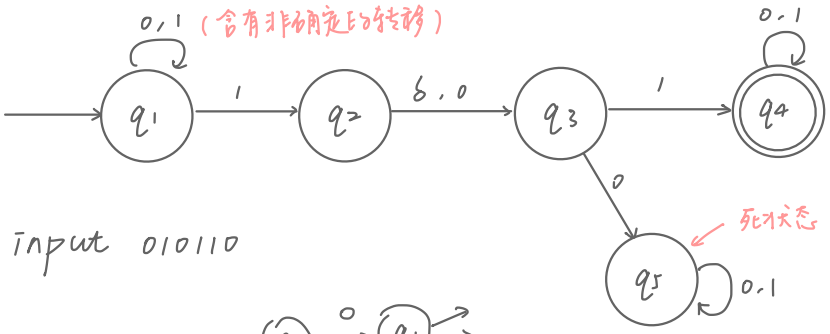
Construct M to accept $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.

1. $Q = Q_1 \times Q_2 = \{(r_1, r_2), r_1 \in Q_1, r_2 \in Q_2\}$
2. $\delta: Q \times \Sigma \to Q$ is defined as for each $(r_1, r_2) \in Q$, and each $a \in \Sigma$, let $\delta(r_1, r_2) = (\delta(r_1), \delta(r_2))$.
3. $q_0 = (q_1, q_2)$
4. $F = \{(r_1, r_2): r_1 \in F_1$ or $r_2 \in F_2\}$ or 改为 and 即可识别 $A_1 \cap A_2$.

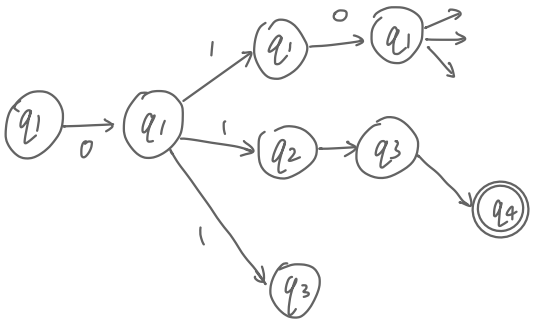
Thm 2.6 If A_1, A_2 are regular languages, so is $A_1 A_2$.

DFA: deterministic finite automaton

NFA: non-deterministic finite automaton (非确定的)



input 010110

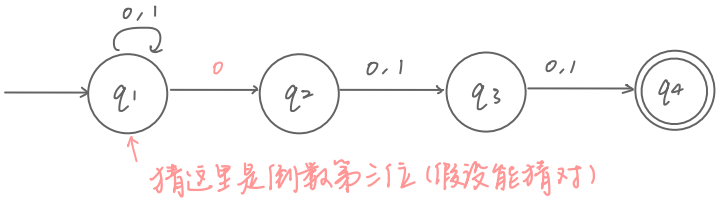


NFA

执行过程类似一棵树

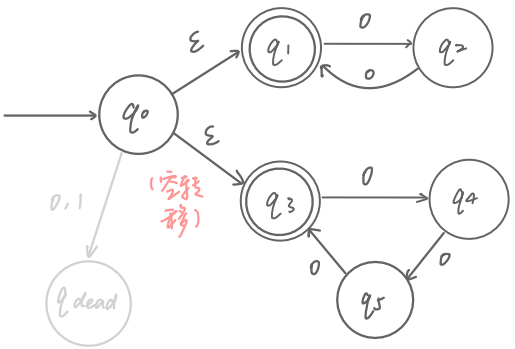
只要有一条分支接受就算接受.

Example: Design an NFA that accepts the set of strings containing a 0 in the third position from the end.



猜这里是倒数第三位(很没能猜对)

Example: $L = \{0^k, k$ is a multiple of 2 or 3$\}$



$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

	0	1	ε
q_0	q_{dead}	q_{dead}	q_1, q_3
q_1	q_0	q_{dead}	q_{dead}

⋮

For any set Q, write $\mathcal{P}(Q)$ to be the collection of all subsets of Q, called the power set of Q. ($\mathcal{P}(Q)$ 为 Q 的幂集 (power set))

Def 2.7 An NFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

1. Q is a finite set of states
2. Σ is an alphabet
3. $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$ is the transition function.
4. $q_0 \in Q$ is the start state.
5. $F \subseteq Q$ is the set of accept states.

转移函数：可以有空转移，且映射到某个集合（不只一个状态）

↑ NFA 与 DFA 的唯一区别

Def 2.8 Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA, and let $w \in \Sigma^*$. Say N accepts w if we can write $w = y_1 y_2 \cdots y_m$, where $y_i \in \Sigma \cup \{\varepsilon\}$, and there exist $r_0 r_1 \cdots r_m \in Q$ such that:

1. $r_0 = q_0$. (初始状态为 q_0)
2. $r_{i+1} = \delta(r_i, y_{i+1})$ for $i = 0, 1, \ldots, n-1$
3. $r_m \in F$ (结束状态被接受)

(只要有一条路径接受就算接受)

DFA 与 NFA 是等价的. (不确定的转移) 下节课证明

Thm 2.9 Every NFA has an equivalent DFA.

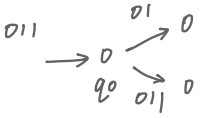
Last Class

- DFA: 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$
- regular language: exist a DFA that recognize L. (language $L \subseteq \Sigma^*$)

  closed under union, intersection, concatenation

DFA $\delta: Q \times \Sigma \to Q$

NFA $\delta: Q \times (\Sigma \cup \{\varepsilon\}) \to \mathcal{P}(Q)$



Variant 1: $\delta: Q \times \Sigma^* \to Q$

We guarantee there is at most one applicable transition.

Variant 2:

If there are multiple applicable transitions, non-deterministically choose one.

ToC

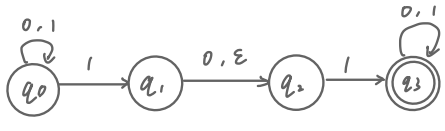
1. Computation models
2. computability
3. complexity

If L can be accepted by a DFA, then its time complexity is $O(n)$, ($x \in \Sigma^*, |x| = n$)

its space complexity is $O(1)$.

正则语言的时间开销为 $O(n)$，空间开销为 $O(1)$.

Exercise. Prove the variants are equavalent to DFA or NFA.



$Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_3\}$, $\delta: Q \times \Sigma \to Q$

$10001 \notin L$

$L = \{w \in \{0,1\}^* \mid w \text{ contains substring } 11 \text{ or } 101\}$

含有 "101" 或 "11" 的串.

$Q \backslash \Sigma$	0	1	ε
q_0	$\{q_0\}$	$\{q_0, q_1\}$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$	$\{q_2\}$
q_2	$\emptyset$	$\{q_3\}$	$\emptyset$
q_3	$\{q_3\}$	$\{q_3\}$	$\emptyset$

Equivalence of DFA & NFA

Thm 2.9 Every NFA has an equivalent DFA. (将n个不确定状态打成至2^n个确定状态即可用DFA模拟NFA)

Proof: Let $N=(Q,\Sigma,\delta,q_0,F)$ be a NFA recognizing A. Construct a DFA $M=(Q',\Sigma,\delta',q_0',F')$ recognizing A.

($R\subseteq Q$. Let $E(R)=\{q\in Q: q$ can be reached from R by traveling along zero or more ε arrows$\}$) (闭包)

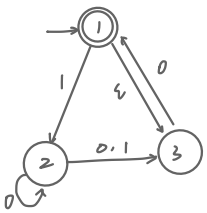
在DFA中加入ε转移能达到的状态.

Define M as follows:

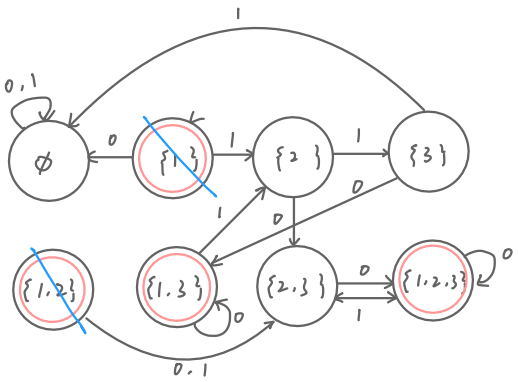
1. $Q'=\mathcal{P}(Q)$ (DFA的状态集为NFA的幂集) ($R\subseteq Q \Rightarrow R\in Q'=\mathcal{P}(Q)$, R是NFA的状态子集. 同时是DFA的一个状态)
2. For $R\in Q'$ and $a\in\Sigma$, let $\delta'(R,a)=\{q\in Q: q\in E(\delta(r,a))$ for some $r\in R\}=\bigcup_{r\in R}E(\delta(r,a))$
3. $q_0'=E(\{q_0\})$
4. $F'=\{R\in Q': R\cap F\neq\emptyset\}$

Example

NFA



DFA



F所有含1的状态都可以接受.

∅表示死状态.

{1,2}: 可能在1或2

{1}和{1,2}不会到达,故可以删去.

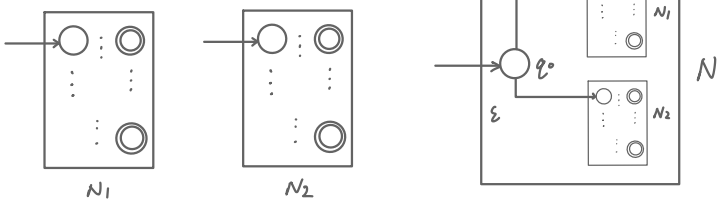
$\delta: Q\times(\Sigma\cup\{\varepsilon\})\to\mathcal{P}(Q)$

(推论)

Corollary 2.10 A language is regular iff some NFA recognize it.

Thm 2.5 The class of regular language is closed under union. 并

Second Proof

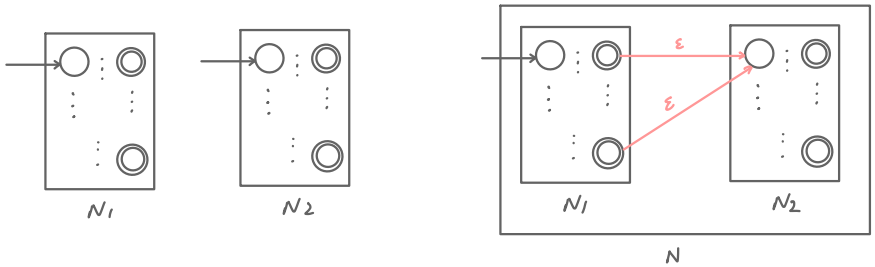


Let $N_1=(Q_1,\Sigma,\delta_1,q_1,F_1)$ recognize A_1, and $N_2=(Q_2,\Sigma_2,\delta_2,q_2,F_2)$ ... A_2, Construct $N=(Q,\Sigma,\delta,q_0,F)$ as follows:

1. $Q=Q_1\cup Q_2\cup\{q_0\}$
2. q_0 is the start state
3. $F=F_1\cup F_2$
4. For $q\in Q$, $a\in\Sigma\cup\{\varepsilon\}$, let $\delta(q,a)=\begin{cases}\delta_1(q,a) & q\in Q_1\\ \delta_2(q,a) & q\in Q_2\\ \{q_1,q_2\} & q=q_0 \text{ and } a=\varepsilon\\ \emptyset & q=q_0 \text{ and } a\neq\varepsilon\end{cases}$

Thm 2.11 ... closed under concatenation 连接

Proof



(N_1, N_2 同上) N_1的初始状态 N_2的接受状态

construct $N=(Q,\Sigma,\delta,q_1,F_2)$

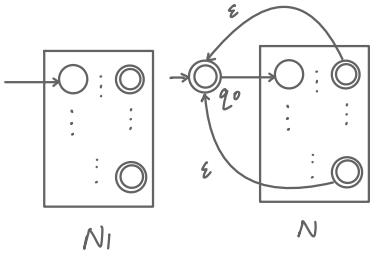
where:

1. $Q=Q_1\cup Q_2$.
2. $\delta(q,a)=\begin{cases}\delta_1(q,a) & q\in Q_1\\ \delta_2(q,a) & q\in Q_2\\ q_2 & q\in F_1 \text{ and } a=\varepsilon\\ \emptyset & q\in F_1 \text{ and } a\neq\varepsilon\end{cases}$ ($q\in Q$, $a\in\Sigma\cup\{\varepsilon\}$)

(□内的部分由记录员补充,如有错误请指出orz)

Thm 2.12 closed under star (*) 星号

Proof:



N_1 recognizes A $\quad$ N recognizes A^* (A中的字符串重复0次及以上) $\quad$ $\varepsilon \in A^*$

$N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

construct $N = (Q, \Sigma, \delta, q_0, F)$

where: 1. $Q = Q_1 \cup \{q_0\}$

2. q_0: the start state

3. $\delta(q,a) = \begin{cases} \delta(q,a) & q \in Q_1 \\ q_0 & q \in F_1 \text{ and } a = \varepsilon \\ \emptyset & q \in F_1 \text{ and } a \neq \varepsilon \end{cases}$

4. $F = F_1 \cup \{q_0\}$

Thm 2.13 closed under complement (作业题)

$A \subseteq \Sigma^*$, $\bar{A} = \Sigma^* \setminus A$

Proof: Let DFA $M = (Q, \Sigma, \delta, q_0, Q_{accept})$ be a DFA recognzing A. Construct $M' = (Q, \Sigma, \delta, q_0, Q'_{accept})$ where $Q'_{accept} = Q \setminus Q_{accept}$. It is easy to verify $L(Q') = \bar{A}$. (将接受状态与非接受状态交换即可)

Regular Expressions 正则表达式

$(0 \cup 1 \cup 2 \cdots \cup 9)^* (1 \cup 3 \cup 5 \cup 7 \cup 9)$ 奇数 (student ID ending with an odd digit)

$(0 \cup 1 \cup 2 \cdots \cup 9)^* (0 \cup 2 \cup 4 \cup 6 \cup 8)$ 偶数 ... even digit

$(0 \cup 1)0^*$ start with 0 or 1 then append zero or more 0's (任意个)

$(0 \cup 1)^* = \Sigma^*$ $\quad (\Sigma = \{0, 1\})$

$(0\Sigma^*) \cup (\Sigma^* 1)$ all strings that start with a 0 or end with 1

Def 2.14 R is a regular expression if R is

1. a for some $a \in \Sigma$ 字母表中的单个字符
2. ε 空串
3. $\emptyset$ 空集
4. $(R_1 \cup R_2)$, where R_1, R_2 are regular expressions.
5. $(R_1 \circ R_2)$ where R_1, R_2 are regular expressions. ("$\circ$"可有可无)
6. (R_1^*) where R_1 is a regular expression

正则表达式 { 内容: 字母、空串、空集 / 运算: 并、连接、星号

(Parentheses may be omitted) 括号可以忽略

Example

1. $0^*10^* = \{w \in \{0,1\}^*, w \text{ contains a single } 1\}$
2. $\Sigma^*1\Sigma^* = \{w \in \{0,1\}^*, w \text{ contains at least one } 1\}$
3. $\Sigma^*001\Sigma^* = \{w \in \{0,1\}^*, w \text{ contains substring } 001\}$
4. $1^*(01^+)^* = \{w \in \{0,1\}^*, \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$
5. $(\Sigma\Sigma)^* = \{w : \text{the length of } w \text{ is even}\}$
   ↑ Σ represents a single letter (not ε)
6. $01 \cup 10 = \{01, 10\}$
7. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
8. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
9. $1^*\emptyset = \emptyset$
10. $\emptyset^* = \{\varepsilon\}$

Example:

1. w contains substring 110. → $\Sigma^*110\Sigma^*$

2. w does not contain 00 as a substring. → $(\varepsilon \cup 0\cup 1)(10\cup 1)^*$

3. the number of 1's is a multiple of 3. → $(0^*10^*10^*10^*)^*$

4. contains at least two 1's and one 0's → $(\Sigma^*1\Sigma^*1\Sigma^*0\Sigma^*)\cup(\Sigma^*0\Sigma^*1\Sigma^*1\Sigma^*)$
$\cup(\Sigma^*1\Sigma^*0\Sigma^*1\Sigma^*)$

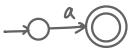
$w=(1\cdots1)_{\geq 0}0(1\cdots1)_{\geq 1}0(1\cdots1)_{\geq 0}$
$=1^{a_1}01^{a_2}0\cdots01^{a_n}$, $a_1\geq 0$, $a_2,\cdots,a_{n-1}\geq 1$, $a_n\geq 0$ claim $w\in L$

Thm 2.15 A language is regular iff (if & only if (⇔)) some regular expression describes it.

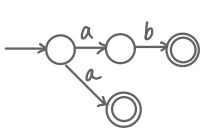
(Lemma: 引理)

Lem 2.16 (⇐) If a language is described by a regular expression, then it is regular.

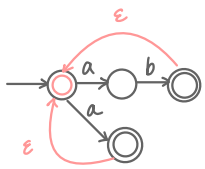
可以被正则表达式表达的语言是正则语言.
(构造式证明)

Proof:
1. $R=a$, $a\in\Sigma$. $L(R)=\{a\}$ 
2. $R=\varepsilon$. 
3. $R=\emptyset$ 
4. $R=R_1\cup R_2$ (by Thm 2.5)
5. $R=R_1\circ R_2$ (by Thm 2.6、2.11)
6. $R=R_1^*$ (by Thm 2.12)

Example $(ab\cup a)^*$

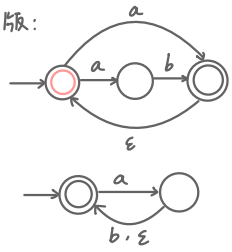


$ab\cup a$



$(ab\cup a)^*$

简化版:



正则语言的数量: $\aleph_0$
所有语言的数量: $\aleph_1$

Last class:
1. Equivalence of DFA and NFA.
2. Regular Expression
3. Lem 2.16

(⇒) If a language is regular, then it is described by a regular expression.

GNFA (G: Generalized)

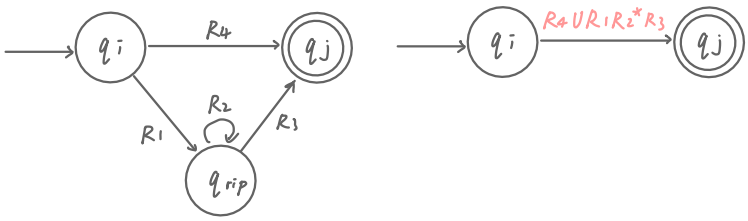
Def 2.17 A generalized nondeterministic finite automaton (GNFA) is a 5-tuple $(Q,\Sigma,\delta,q_s,q_a)$ (q_{start}, q_{accept})

where:
1. Q is a finite set of states
2. Σ is the alphabet.
3. $\delta:(Q-\{q_{accept}\})\times R\to(Q-\{q_{start}\})$ (R: the set of all regular expression) (初始状态无入边,接受状态无出边.)
4. q_{start} is the start state.
5. q_{accept} is the accept state.

Note that GNFA requires:
1. Start state has no incoming edge;
2. Accept state has no outgoing edge;
3. Except for the start and accept state, there is one arrow from one state to every other state, and also from each state to itself.

(为了收缩到只剩起始与接收状态,边上是待证的正则表达式.)

状态数-1



Proof: Let M be the DFA for language A. construct M to a GNFA as follows: (DFA → GNFA)

1. add a new start state with ε arrow to the old start state,
2. add a new accept state with ε arrows from old accept states.
3. replace multiple edges by one edge using $\cup$.
4. add arrows with $\emptyset$ between states that has no arrows.

Convert(G)

1. Let k be the number of states in G
2. If $k=2$, return $\delta(q_{start}, q_{accept})$
3. If $k>2$, choose $q_{rip} \in Q \setminus \{q_s, q_a\}$

Let G' be the GNFA $(Q', \Sigma, \delta', q_s, q_a)$, where:

(a) $Q' = Q - \{q_{rip}\}$

(b) $\forall q_i \in Q' - \{q_a\}$, $\forall q_j \in Q' - \{q_s\}$,

Let $\delta'(q_i, q_j) = \delta(q_i, q_j) \cup \delta(q_i, q_{rip})\,\delta(q_{rip}, q_{rip})^*\,\delta(q_{rip}, q_j)$

Notice q_i can be equal to q_j

4. return Convert(G')

Claim for any GNFA G, Convert(G) is equivalent:

Convert(G')

Proof: 1) $L(G) \subseteq L(G')$

For any $w \in \Sigma^*$, if G accepts w, then G' accepts w.

$q_{start}, q_{t1}, q_{t2}, q_{t3}, \cdots, q_{accept}$

If none of them is q_{rip}, then G' also accepts w.

If q_{rip} appears, $q_{start}, q_{t1}, \cdots, q_{ti}, q_{ti+1}, q_{ti+2}$

q_{rip} $\quad \delta(q_{ti}, q_{ti+1})\,\delta(q_{ti+1}, q_{ti+2})$

Then G' also accepts w.

2) $L(G') \subseteq L(G)$

Non-regular Language 非正则语言

Pumping Lemma

(Almost all languages are non-regular)

Lemma 2.17 If A is a regular language, then $\exists p \in \mathbb{N}$, such that for any string s of length at least p, $\exists x, y, z \in \Sigma^*$, s.t. $s = xyz$ and:

1) $xy^iz \in A$ for every $i \geq 0$.

2) $|y| > 0$.

3) $|xy| \leq p$.

(由于状态数有限，令 $p = |Q|$，则任意长度大于 p 的字符串必然含有重复状态)

PHP：鸽笼原理

Proof: Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing A. Let $p = |Q|$.

Let $s = s_1 s_2 \cdots s_n \in \Sigma^n$, where $n \geq p$. Let $r_1, r_2, \ldots, r_{n+1} \in Q$ be the sequence of n states.

That is $\begin{cases} r_1 = q_s \\ r_{i+1} = \delta(r_i, s_i), \text{ for } i = 1, 2, \ldots, n \end{cases}$

By PHP, there exists at least two states that are same. Consider the first occurrence of repeated states. As such, $l - 1 \leq p$, i.e. $|xy| \leq p$. Call the first r_j, the second r_L.

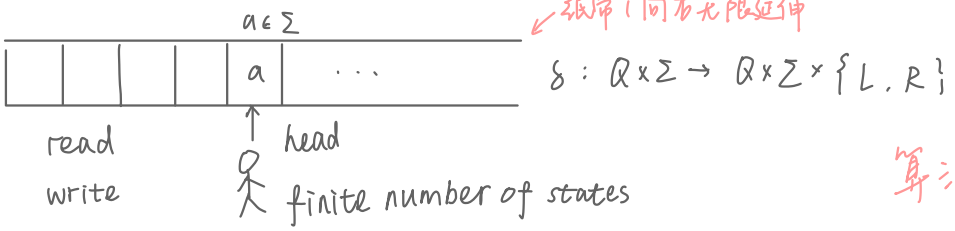
Let

What is an algorithm ?

An algorithm is a machanical process to be followed in calculations or other problem-solving operation.

Example: 1. +、-、×、÷ ; 2. GCD ; 3. Dijkstra

In 1936. for the first time , Alan Turing rigorously defined algorithms



算法就是图灵机.

An algorithm is a TM (Turing Machine)

图灵机的定义

Def 3.1 (Turing Machine) A TM is a 7-tuple ($Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject}$)

输入字母表

where: 1. Q is the set of states.

2. Σ is the input alphabet, $\sqcup$ (space) $\notin \Sigma$.

3. Γ is the tape alphabet, where $\sqcup \in \Gamma$, $\triangleright \in \Gamma$, $\Sigma \subseteq \Gamma$

$\Sigma \subseteq \Gamma$

4. $\delta : Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R, S\}$ is the transition function.

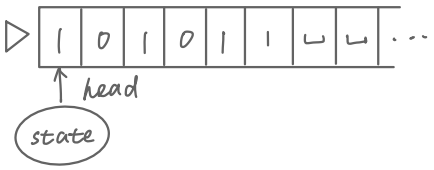
5. q_0 is the start state.

δ(状态，字符) = (新状态、新字符、移动方向)

6. q_{ac} is the ...

7. q_{re} is ...

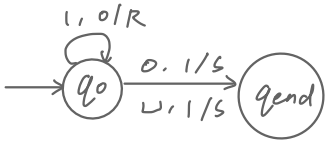
Example : $\Sigma = \{0, 1\}$



1) Binary add 1

Input a binary number n, least significant bit first, output n+1.

$\Gamma = \{0, 1, \sqcup, \triangleright\}$.



Input	Output
101⊔⊔	011⊔⊔
11⊔⊔	001⊔⊔

2) add 2 : A. first add 1, then go left until ▷. then add 1 again.

B. head go right for 1 bit, then add 1.

(3) Decide if $w \in \Sigma^*$ is a palindrome, i.e. $w = w^R$.

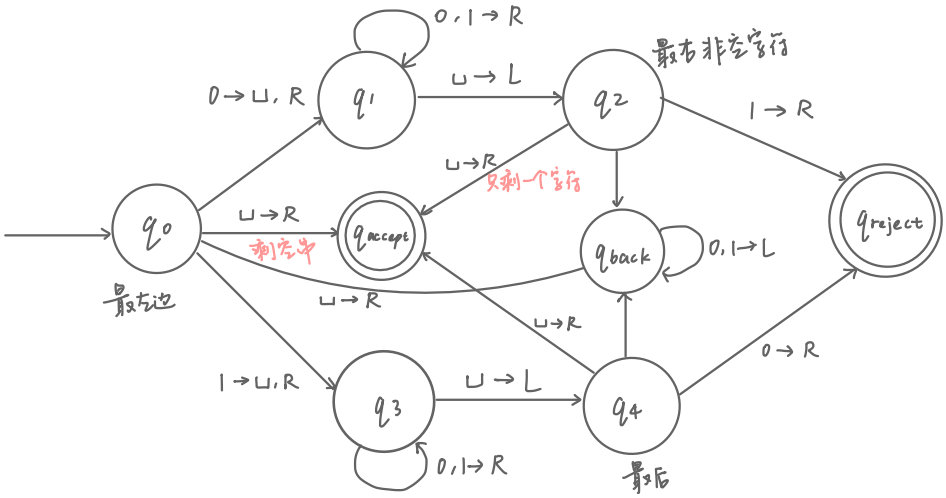
idea: 1. read the leftmost symbol, memorize and erase.

2. go to the rightmost position, and read the symbol, check if it matches the leftmost one. If not, reject.

Erase the rightmost one.

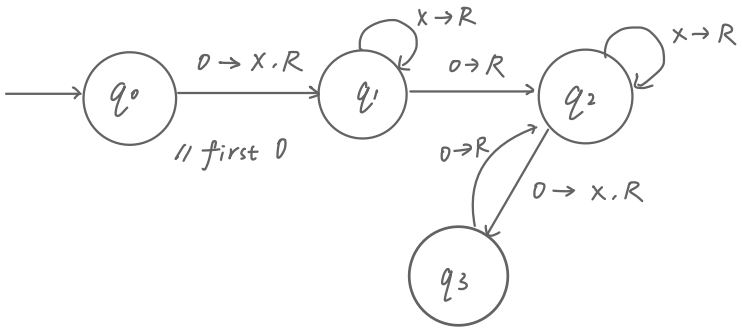
3. Go to the leftmost non-blank symbol.

4. Repeat 1-3.



(4) Decide $L = \{0^{2^n}, n \geq 0\}$, where $\Sigma = \{0\}$. (书上有，不记了)

δ:



Exercise:

1) Binary Comparison. $L = \{x, y, x, y \in \{0,1\}^*\}$ $x \geq y$. MSB, no leading 0 unless it's 0

2) Binary add 1. MSB.

3) $L = \{0^n 1^n, n \geq 0\}$

Ideas:

1)

2)

3)

Def 3.2 Let $L \subseteq \{0,1\}^*$. Let M be a TM. Say M decides L in time $T(n)$ if every

判定

$x \in \{0,1\}^*$.

(1) M halts in $T(n)$ steps;

(2) If $x \in L$, then M accepts x;

(3) If $x \notin L$, then M rejects x.

图灵可判定的

Def 3.3 Let $L \in \{0,1\}^*$. Call L (Turing) decidable if there is a TM decides it.

Note that on an input x, a TM may accept, reject, or loop forever.

In Def 3.2, the TM either accept or reject a string, doesn't loop forever.

图灵可识别的语言

Def 3.4 Let M be a TM, the set of strings that M accepts is the language recognized by M, denoted by $L(M)$.

L内的字符串被接受，L之外的字符串随意

Def 3.5 Let $L \subseteq \{0,1\}$. Call L Turing Recognizable if there is some TM recognizes it.

可识别

(Obviously, every decidable language is recognizable. The converse is not true.)

可判定比可识别更严格

in time $T(n)$

Def 3.6 Let $f: \{0,1\}^* \to \{0,1\}^* \cup \{\text{undefined}\}$. Say TM M computes f if for every $x \in \{0,1\}^*$, with $f(x) \neq$ undefined, M halts with $f(x)$ on its tape in at most $T(|x|)$ steps.

What is an algorithm? An algorithm is a TM. (1936, Alan Turing)

Despite its simplicity, it's capable of impementing any computer algorithm.

"Everything should be made as simple as possible, but no simpler"

Next: 定义一些图灵机的变种，然后证明它们等价.

Lemma 3.7 If Language $L \subseteq \{0,1\}^*$ is decidable in time $T(n)$ by a TM on alphabet Γ, then it is decidable in time $O(\log|\Gamma| T(n)) = O_\Gamma(T(n))$ by a TM on alphabet $\Gamma = \{0,1,\sqcup\}$

Proof: Encode any symbol in Γ using $k = \lceil \log_2|\Gamma| \rceil = O(\log_2|\Gamma|)$ bits. To simulate one step of M, TM M' will:

1) Use k steps to read a symbol $a \in \Gamma$.

2) Transit to next step q', and get the new symbol b (to overwrite a)

3) Overwrite a by b

4) Go left of right for k steps, or stay.

In total, the simulation (for one step) takes $\leq k+1+k+k = O(k)$.

Def 3.8 A k-tape TM M is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, where:

k带图灵机

$$\delta: Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$$

Usually, the first tape is the input tape, the last one is the output tape, the remainings are work tapes.

Lem 3.9 Let $L \subseteq \{0,1\}^*$. If L is decidable by a k-tape TM in time $T(n)$, then L is decidable in time $O(kT(n^2))$ by a single-tape TM.

4.18

Encoding of the TMs satisfy the following properties:

1. every string $\alpha \in \{0,1\}^*$ represents some TM;
2. Every TM is represented by infinitely many strings.

(On invalid encoding α, M_α always reject)

TM: $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{ac}, q_{re})$. $\Sigma = \{0,1\}$

$\delta: Q \times \Gamma^k \to Q \times \Gamma^k \times \{L, R, S\}^k$, (has size $O_M(1)$)

Th 4.1 (Universal TM) There exists a multitape TM U s.t. for all $x, \alpha \in \{0,1\}^*$, $U(x, \alpha) = M_\alpha(x)$. Moreover, if M_α halts on input x within T steps, then $U(x, \alpha)$ halts in $O_M(T \log T)$ steps. (Weak version: $O_M(T^2)$)

Proof of the weaker version:

By lemma 3.9, every multitape TM can be converted to a single-tape TM with time $O_M(T^2)$.

For each step of M, our UTM U:

1. reads the symbol a on the work tape of M;
2. reads the current state q_1 of M;
3. scan through the description β of M to find $\delta(q_1, \alpha)$
4. update the symbol (on the work tape of M) and move the head if needed.
5. update "the current state of M" from q_1 to q_2.

4 25

Last class:

1. DTIME(T(n)): 能被多带TM在O(T(n))时间内判定的语言的集合.

2. $P = \bigcup_{c=1} DTIME(n^c)$

3 Encoding of a TM. $\alpha \in \{0,1\}^*$ 表示图灵机的编码(程序), $x \in \{0,1\}^*$ 为输入.

$U(\alpha, x) = M_\alpha(x)$ 为双带图灵机, 能根据输入 x 模拟 M_α. U: 通用图灵机(忠实的模拟)

定理: 若 $M_\alpha(x)$ 执行 T 步结束, 则 $U(\alpha, x)$ 在 $O_m(T \log T)$ 步能结束. (如果用 RAM 图灵机模拟则为 $O_m(T)$)

引理 4.2 Almost all languages are undecidable. (几乎所有语言都是不可判定的)

Proof: Languages $= \{L \in \{0,1\}^*\} = 2^{\aleph_0} = \aleph_1$ ($TMs = \aleph_0$)

Counting: |decidable| = |TM|
可数个: $\aleph_0$

diagonalization (对角化方法)

$L_{flip} \overset{def}{=} \{\alpha : M_\alpha \text{ does not accept } \alpha\}$ M_α 运行它自身的编码, 结果是接收、拒绝或死循环.

L_{flip} 是字符串集, 这些字符串所编码的 TM 不接收它自己.

证明 L_{flip} 是不可判定的:

假设 L_{flip} 可以被 M_β 判定, 即 $L(M_\beta) = L_{flip}$.

① 若 $\beta \in L_{flip}$, 则 M_β 不能接收 β, 即 M_β 拒绝. 与 M_β 判定 L_{flip} 矛盾.

② 若 $\beta \notin L_{flip}$, 则 M_β 必须接收 β. 但 M_β 判定 L_{flip} (不能接收自身的编码). 故 M_β 不能接受 β. 矛盾.

故 L_{flip} 不可判定.

Turing Halting Problem (图灵停机问题: 不存在一个算法能判断所有程序是否对于输入 x 停机)

$L_{halt} \overset{def}{=} \{(\alpha, x) : M_\alpha \text{ halts on input } x\}$

费马大定理: $(\forall m \geq 3)(\forall a,b,c \geq 1)(a^m + b^m \neq c^m)$ (Fermat Last Theorem, FLT)

M_α:
```
T = 2
While True:
    T = T + 1
    for d = 3 to T
        for (a, b, c ∈ {1, 2, ..., T}
            if a^d + b^d = c^d then exit
```

FLT 成立 $\Leftrightarrow (M_\alpha, \varepsilon) \notin L_{halt}$

归约 (reduction)

(L_1 可以归约到 L_2)

Def 4.4 Let $L_1, L_2 \in \{0,1\}^*$. Write $L_1 \leq L_2$ if there is a reduction from L_1 to L_2. That, there exists a TM $M: \{0,1\}^* \to \{0,1\}^*$ (On any input x, M always halts and outputs a string M(x)) s.t.

1) $(\forall x \in L_1)(M(x) \in L_2)$

2) $(\forall x \notin L_1)(M(x) \notin L_2)$

M 将 L_1 的输入 x 映射到 L_2 的输入 M(x)

L_1 对 x 的判定结果与 L_2 对 M(x) 的判定结果相同.

$L_1 \leq L_2$ 时, 若 L_2 可判定, 则 L_1 可判定; (逆否) 若 L_1 不可判定, 则 L_2 不可判定.

Th 4.5 L_{halt} is undecidable. (已知 L_{flip} 不可判定) ↑ 与已知矛盾

Proof $L_{flip} \le L_{halt}$: 假设 L_{halt} 可被 M_{halt} 判定，则 L_{flip} 可判定。构造如下的 M_{flip}:

Run M_{halt} on α on input (α, α).

① If M_{halt} rejects (α, α), Let M_{flip} accept α.

② If M_{halt} accepts (α, α), simulate M_α on input α using a UTM, and flip the output.

It's easy to verify M_{flip} decides L_{flip}. Contradiction.

$L_{accept} = \{(\alpha, x): M_\alpha \text{ accepts } x\}$ 也是不可判定的.

证明 $L_{halt} \le L_{accept}$: 假设 L_{accept} 可判定，则有在 M_{accept} 判定 L_{accept}. 构造 M_{halt} 如下判定 L_{halt}:

M_{halt} {
1) On input (α, x), create a new TM M_β 模拟 M_α on input x. 只要 M_α 停机就接收，否则死循环.
2) Run M_{accept} on input (β, x), and forward its output. (若 M_β 接收则 M_{accept} 接收，否则拒绝)
}

↓ M_{halt} 接收　↓ M_{halt} 拒绝

则 M_{halt} 判定 L_{halt}. 矛盾.

归约: 假设 L_{accept} 可判定，则 L_{halt} 可判定. 矛盾.

关键：利用判定前者的图灵机构造判定后者的图灵机.

(但实际上两个图灵机都不存在)

Lem 4.7 $L_{empty} = \{\langle M\rangle : M \text{ does not accept any input, i.e. } L(M) = \emptyset\}$ 不可判定.

证明：假设 L_{empty} 可被 M_{empty} 判定，则如下构造 M_{halt}:

1) 新TM M_β, input: $y \in \{0,1\}^*$: M_β: 只要停机就 accept，否则 $L(M_\beta) = \emptyset$.

M_{halt} {
a) simulate M_α on input x.
b) if a) halts, always accept y.
}

则 $L(M_\beta) = \emptyset$ 如果 M_α 在 x 上不停机，否则 $L(M_\beta) = \{0,1\}^*$.

2) Run M_{empty} on input M_β, 将输出反转 (如果 M_α 不停机则 M_{empty} 不接收，否则 M_{empty} 接收).

Th 4.8 $L_{regular} = \{\langle M\rangle$ M is a TM s.t. $L(M)$ is a regular language$\}$ 是不可判定的.

证 $L_{regular} \le L_{accept}$: 假设 $L_{regular}$ 可判定

(1) 构造 M_β, $\beta = \beta(\alpha, x)$. M_β 的输入记为 y.

a) If $y \in \{0^n1^n, n \ge 0\}$, accept. ← 不是正则语言

b) Otherwise. 模拟 M_α on x. 若 M_α accepts x, M_β accepts y. ↓ 顺序执行

(2) Run $M_{regular}$ on M_β, and forward its output.

Case 1: $(\alpha, x) \notin L_{accept}$. ($M_\alpha$ 不接收 x) 此时 $M_\beta: \{0^n1^n, n \ge 0\}$. $M_{regular}$ 拒绝 M_β. 则 M_{accept} 拒绝 M_β.

Case 2: $(\alpha, x) \in L_{accept}$. So $L(M_\beta) = \{0,1\}^*$. 此时 $M_{regular}$ 接收 β. M_{halt} 同样.

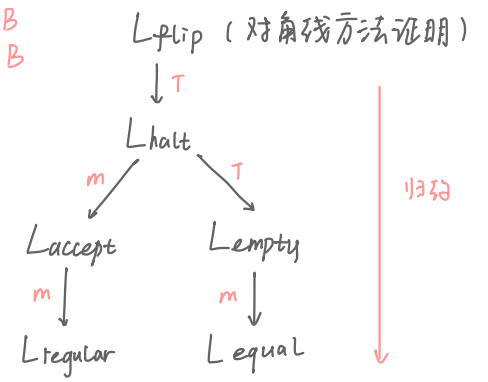
Lem 4.9 Let $L_{equal} = \{(\langle M_1\rangle, \langle M_2\rangle)$ M_1, M_2 are TMs s.t. $L(M_1) = L(M_2)\}$

对 $\langle M\rangle$ 构造 TM M_{empty} 如下:

归约：一种算法，将一个问题转化(transform)到另一个问题。

直观：$A \le B$ 时，B的解法可以用于解决A。若B可解则A可解。
若A不可解，则B不可解。

A→B
A≤B



若下面可判定，则上面可判定。
若上面不可判定，则下面不可判定。

m: mapping 可规约。

Examples of reduction:

1. Mapping reduction (映射归约，aka many-one reduction) $L_1 \le_m L_2$
2. Turing reduction $L_1 \le_T L_2$
3. Polynomial-time mapping reduction (Karp reduction) $L_1 \le_p L_2$.
4. Polynomial-time Turing reduction (Cook reduction)

Example: 有一个算x^2的算法来算 $x \cdot y$: $xy = \frac{1}{2}[(x+y)^2 - x^2 - y^2]$ (Turing reduction)
(调用三次)

将 max matching problem 转为 max flow problem (Mapping reduction)
(调用一次)

Def 4.4 (Mapping reduction) Many-one 归约

Let $L_1, L_2 \in \{0,1\}^*$. $L_1 \le_m L_2$ if there is a computable function $\phi: \{0,1\}^* \to \{0,1\}^*$,

s.t. $x \in L_1$ iff $\phi(x) \in L_2$ for any x.

Proposition 4.5

(1) $L \le L$;

(2) $L_1 \le_m L_2$ iff $\bar{L_1} \le_m \bar{L_2}$,

(3) if $L_1 \le_m L_2$, $L_2 \le_m L_3$, then $L_1 \le_m L_3$. (传递性)

Def 4.6 (Oracle Turing Machine) 神谕图灵机

k-tape Turing Machine;

additional states: q_{ask}, $q_{response}$;

Last tape: oracle tape (神谕带);

rest tapes: input & working tapes.

写一个问题，进入 q_{ask} 后：(一步内)

1. 擦掉问题
2. 得到答案，是：1，否：0。(写在最左边)
3. 指针移到最左边。
4. 进入 $q_{response}$.

Def 4.7 Let $L_1, L_2 \in \{0,1\}^*$, L_1 is Turing reducable to L_2, denoted $L_1 \leq_T L_2$, if there is an oracle TM, with an oracle for the language L_2, that decides L_1.
(图灵归约)

(理解为函数调用，但瞬间返回答案)　　L_2+神谕TM可以判定L_1.

Prop 4.8 归约的性质

(1) For any decidable languages L_1, L_2, we have $L_1 \leq_T L_2$. (可判定语言可以互相归约)

(2) $L_1 \leq_T L_2$, $L_2 \leq_T L_3 \Rightarrow L_1 \leq_T L_3$.

(3) $L \leq_T L$. $\bar{L} \leq_T \bar{L}$.

(4) $L_1 \leq_m L_2 \Rightarrow L_1 \leq_T L_2$

(性质也是一种语言)

Def 4.14 (Nontrivial property of language) Property P is about the Language recognized by TMs if whenever $L(M) = L(N)$. P contains $\langle M \rangle$ iff P contains $\langle N \rangle$. ($P \subseteq \{0,1\}^*$)
语言的非平凡性质

不平凡的：$\exists$ TMα. s.t. $\alpha \in P$. $\exists$ TMβ s.t. $\beta \notin P$. (不是所有语言都有的性质)

Th 4.15 (Rice's Theorem) Any nontrivial property about the Language recognized by TMs is undecidable. (任何关于识别的语言的非平凡性质都不可判定)
莱斯定理　　无法判断M_T能否满足P (或能否实现M_α的功能)

Proof: ($\emptyset \notin P$) 假设P可被M_P判定。由于P不平凡，任选$\beta \in P$，不妨在关会拒绝的TM.

构造 M_{accept} 如下：

1) 对输入(α, x)，构造M_δ，$\delta = \delta(\alpha, x)$ on input y 如下：

a) 对于输入(α, x)，直到M_α接收 x 结束。否则永远循环.

b) 模拟 M_P 在输入 y 上运行，当且仅当 M_P 接收 y 时 M_{accept} 接受.

2) Run M_P on input δ. Accept iff M_P accepts.

$L_{accept} \leq_m P$

Claim M_{accept} decides L_{accept}

Case 1: $(\alpha, x) \in L_{accept}$, $L(M_\delta) = L(M_\beta)$. So M_P accepts δ.

Case 2: $(\alpha, x) \notin L_{accept}$, $L(M_\delta) = \emptyset$. So M_P rejects δ.

PCP (Post Correspondence Problem)

$P = \{ [\frac{t_i}{b_i}], t_i, b_i \in \Sigma^* \}$ 若有 $t_{i_1} t_{i_2} \cdots t_{i_L} = b_{i_1} b_{i_2} \cdots b_{i_L}$，则称 $i_1 i_2 \cdots i_L$ 为一个匹配 (match).

令 PCP 为一个语言，$PCP = \{\langle P \rangle : P$ 是含有匹配的牌组实例$\}$

Th 4.16 PCP 是不可判定问题.

思路：将 L_{accept} 归约到 PCP. 目标：构造一个 PCP 实例 $P = P(\alpha, x)$ 使 $(\alpha, x) \in L_{accept}$ iff $P \in PCP$.

MPCP: $\{\langle P \rangle : P \in PCP$ 且 P 含有从第一张牌起始的匹配$\}$

目标：1) $L_{accept} \le_m MPCP$；2) $MPCP \le_m PCP$.

$M_\alpha = (Q, \Sigma, \Gamma, \delta, q_0, q_{ac}, q_{re})$

Part 1：首张：$\left[\frac{\#}{\#q_0w_1w_2\cdots w_n\#}\right]$ （$w = w_1w_2\cdots w_n$ 为输入）

Part 2：若 $\exists\, \delta(q,a) = (q', b, R)$ $(q \neq q')$，则放入 $\left[\frac{qa}{bq'}\right]$.

Part 3：若 $\delta(q,a) = (q', b, L)$，放入 $\left[\frac{cqa}{q'cb}\right]$. c 为 Γ 中所有字符.

Part 4：对 $\forall a \in \Gamma$，放入 $\left[\frac{a}{a}\right]$.

例：$\Gamma = \{0, 1, 2, \sqcup\}$，$w = 0100$

q_0：$\delta(q_0, 0) = (q_1, 2, R)$.

Part 1：$\left[\frac{t_1}{b_1}\right] = \left[\frac{\#}{\#q_00100\#}\right]$　Part 2：$\left[\frac{q_00}{2q_1}\right]$　Part 3：（skip）　Part 4：$\left[\frac{0}{0}\right], \left[\frac{1}{1}\right], \left[\frac{2}{2}\right], \left[\frac{\sqcup}{\sqcup}\right]$

$\Rightarrow$ $\#q_00100\#$

$\#q_00100\#2q_1100\#$

step 1　step 2

（1、0、0：重张 $\left[\frac{1}{1}\right], \left[\frac{0}{0}\right]$）

Th 4.7 All consistent axiomatic formulation of number theory which include Peano arithmatic contains undecidable propositions.

(哥德尔(Gödel)不完备性定理：存在一些既无法证明又无法证伪的命题)

证明思路：枚举长度不超过k的证明，若完备则必停机 → 停机问题可判定 矛盾.

希尔伯特第十问题：丢番图方程是否有解是不可判定问题.(归约到图灵停机问题)
(Hilbelt 10-th problem)

Th 4.18 $L_{Dio} = \{ \langle P(x_1, \dots, x_n) \rangle : \text{Diophantine equation } P(x_1, \dots, x_n) = 0 \text{ has a solution} \}$

Complexity 复杂性

DTIME(T(n)) : 可被TM在O(T(n))时间内判定的语言集.

$P = \bigcup_{c \geq 1} DTIME(n^c)$: $EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$

Given a decision problem, i.e. $L \in \{0,1\}^*$, understand the resources required to solve it.

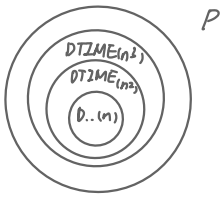
Resources: time, space, randomness, parallelism 资源：时间、空间、随机性、并发

Understand the relation between various problems (mainly be reduction.)

TCS (理论计算机) central problem: $P \neq NP$.

Def 5.1 Let $T: \mathbb{N} \to \mathbb{N}$. Language $L \in DTIME(T(n))$ iff there exists a multitape TM M that runs in time $O(T(n))$ and decides L.

Def 5.2 Let $P = \bigcup_{c \geq 1} DTIME(n^c)$, $EXP = \bigcup_{c \geq 1} DTIME(2^{n^c})$.



(P for Polynomial time) 可解问题一般 $\in P$ 或 $\in EXP$. $L \in P$ 时，通常 n^c 中的 c 很小.

For example: 最短路：$O(V^2)$, $O(E+V\log V)$; 最小生成树 $O(E\log V)$, $O(E+V\log V)$
最大流 $O(V \cdot E^2) \to O(V^3) \to O(E^{1+\alpha(n)})$

线性规划(Linear Programming)：n个变量，L为输入的位长度. $O(n^{2.5}L)$

BFS / DFS : $O(V+E)$ 字符串匹配：$O(n)$ 并查集：$O(m\alpha(n))$ m：操作次数；
$\alpha(n)$：小于5的某个常数

多项式是闭包的，多次调用后仍属于P.

efficient = polinomial time = $n^{O(1)}$

Th 5.3 PATH = {⟨G, s, t⟩ : G是有向图，有一条从s到t的路} ∈ P. (DFS / BFS)

Th 5.4 RELPRIME = {⟨x, y⟩ : x和y互质} ∈ P (n为输入长度)

Proof: Euclid GCD(x, y). (大除小，小除余.)

```
if x<y, swap x,y
while y>0 :
    x ← x mod y
    swap x,y
return x
```

if GCD(x,y)=1 accept

else reject

时间:

if $\frac{x}{2} \geq y$, then $x \% y < y \leq \frac{x}{2}$;

if $\frac{x}{2} < y$, then $x \% y = x - \lfloor \frac{x}{y} \rfloor y = x - y < \frac{x}{2}$

So, x drops by at least a half.

$O(\log_2 x) = O(\log_2 2^{O(n)}) = O(n) = O(n)$

取余数: $O(n^2)$

共计时间 $O(n^3)$

Th 5.5 PRIME = {x, x是[质数]} (输入为二进制)

暴力算法 ∈ EXP.

```
y ← 2
while y<x
  if x mod y = 0, reject
  y ← y+1
accept
```

n: 输入长度 $n = \lceil \log_2 x \rceil$ $x \leq 2^n$

最坏不用试不 x-2 即 $2^n - 2$ ($O(2^n)$)

(如果用一进制表示则 PRIME ∈ P)

Th 5.6 PRIME ∈ P

NP N for Non-deterministic P for Polynomial

NP: 在多项式时间内可判定.

P: 在多项式时间内可解.

if P = NP, 数学家失业 密码系统不再安全.

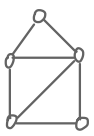
Def 5.7 $L \in \{0,1\}^*$ is in NP if exists a polynomial $p: N \to N$ and a poly-time TM M called the verifier for L, such that for every $x \in \{0,1\}^*$,

$x \in L$ iff $\exists w \in \{0,1\}^{p(|x|)}$ s.t. $M(x, w) = 1$. Such w is called a certificate or witness for x.

Example 5.8 图同构 ∈ NP. ($G \simeq H$)

找一个 $G \to H$ 的双射 w (↑证书)，将 w 喂给验证者 M. 若 M 接受 w，则接受.

Example 5.9 CLIQUE = {⟨G, k⟩ G contains a K_k subgraph} ∈ NP.



G

⟨G, 4⟩ ∉ CLIQUE

⟨G, 3⟩ ∈ CLIQUE

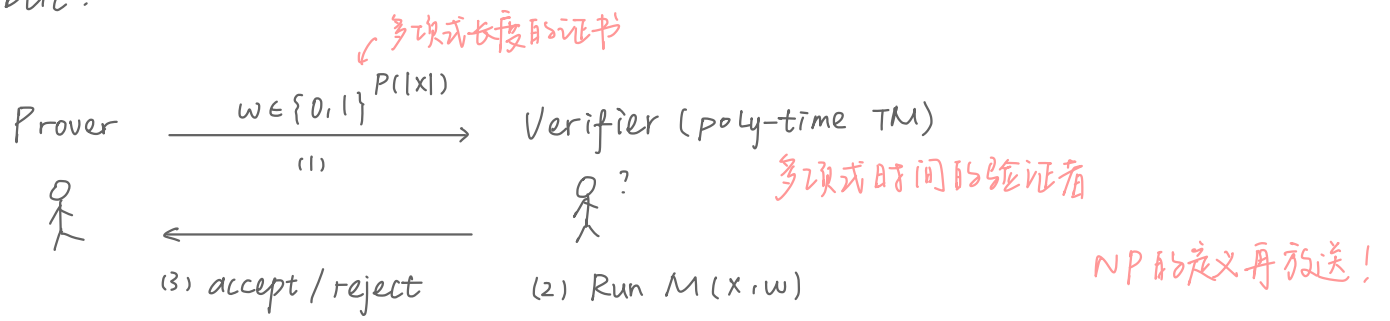
验证者: 输入 k 个点 $\{u_1, u_2, \ldots, u_k\} \subseteq V(G)$

检查是否对 $\forall i, j$ $(i \neq j)$ 都存在 $\langle u_i, u_j \rangle$.

NP: ① 可以被非确定型图灵机(NTM)在多项式时间内判定

② 可以被高效验证的问题.

Public:



If $x \in L$, $\exists w \in \{0,1\}^{p(|x|)}$ s.t. $M(x,w)=1$; (类比老师改卷)

If $x \notin L$, $\forall w \in \{0,1\}^{p(|x|)}$ s.t. $M(x,w)=0$. 不在乎如何得到 w.

Example 5.10 (Traveling Salesman, 旅行商问题) n个点, dij为i到j的距离. 总长度上限为K. 是否有经过所有点回到起点. ($TS \in NP$)

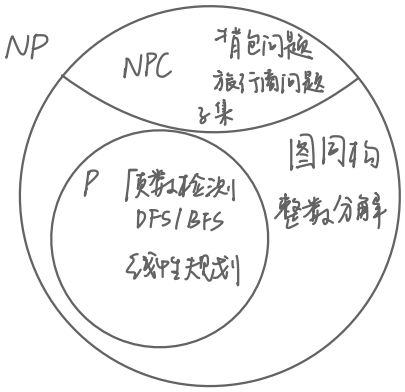
NP要求: 多项式时间的验证者. 给路径, 验证是否经过所有点; 是否距离总和小于等于K.

Example 5.11 Given N, L, R, decides if N has a prime factor in $[L,R]$.

certificate: $p \in [L,R]$ s.t. p is a prime and $p \mid N$.

Example 5.12 (0/1 integer programming) Given m linear inequalities with integral coefficients over n variables $u_1, u_2, \dots, u_n$, decide if there is an assignment of 0's and 1's to $u_1, u_2, \dots, u_n$ satisfying all equalities.

Example 5.13 背包问题 $\in NP$.



不是NP: 没有高效的验证者 (e.g. 图不同构问题、不含k-团问题)

多项式的闭包性质 $poly(n) = n^{O(1)}$

If: $f(n), g(n) \in O(n^{O(1)}) = poly(n)$, then

(1) $f(n)+g(n) \in poly(n)$

(2) $f(n) \cdot g(n) \in poly(n)$

(3) $f(g(n)) \in poly(n)$

练习: $\binom{n}{\log n} \notin poly(n)$

Th 5.16 $P \subseteq NP \subseteq EXP$

(1) $P \subseteq NP$: Let $L \in P$, $p(n)=0$. TM M decides L efficiently. (验证者接收空串.)

If $x \in L$, then $M(x,\varepsilon)=1$; if $x \notin L$, then $M(x,\varepsilon)=0$.

{ if $x \in L$ ...
  if $x \notin L$ ...

(2) $NP \subseteq EXP$: Since $L \in NP$, $\exists p: N \to N$ (polynomial) and a poly-time TM such that ...

Construct M_1', 枚举 $w \in \{0,1\}^{p(|x|)}$, 检查 $M(x,w)=1$ or 0. 若 $\exists w$ s.t. $M(x,w)=1$, M' accepts x.

Total time: $2^{p(|x|)} \cdot poly(n) = 2^{n^{O(1)}} \cdot n^{O(1)} = 2^{n^{O(1)}}$.

Non-deterministic TM (NTM: 非确定型图灵机)

每个分支都猜对. Li Yuan: 途猜必对, 直觉非常好.

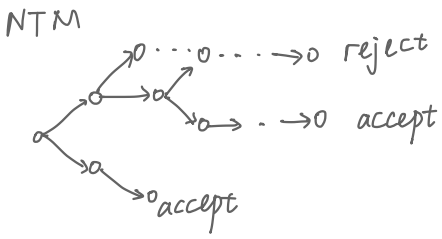
At any point, NTM 都有多种可能性. 形式上仍是 7元组, 仅转移函数不同.

$\delta: Q \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, R, S\})$

Accept: 存在一条可接受的路径即可.

运行时间: 最坏情况 $T(n)$: 对所有输入 $x \in \{0,1\}^n$, 在所有路径中最后一个停下的所在时间.

Def 5.18 NTM M decides L if for every $x \in \{0,1\}^*$, $x \in L$ iff $M(x)=1$.



运行时间: 最长时间.

Def 5.19 (Binary-choice NTM) 二择 NTM. 8元组, 有2个转移函数 $\delta_1, \delta_2: Q \times \Gamma \times \{L, R, S\}$, 每一步随便选 δ_1 或 δ_2 进行转移.

Lem 5.20 $L \subseteq \{0,1\}^*$, 若 L 可在 $T(n)$ 内被 NTM M 判定, 则 L 可在 $O_M(T(n))$ 内被二择 NTM 判定.

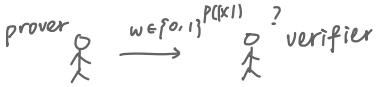
证明: 将多选一转成若干个二选一:

每步 M 至多有 $2^{|Q|\times|\Gamma|\times 3}$ 种选择.

可用 $\log_2 2^{|Q|\times|\Gamma|\times 3} = |Q|\times|\Gamma|\times 3$ 步模拟. 总时间 $\le |Q|\times|\Gamma|\times 3 \times T(n) = O_M(T(n))$.

Def 5.21 $T: \mathbb{N} \to \mathbb{N}$. $\mathrm{NTIME}(T(n))$: 可被 NTM 在 $O(T(n))$ 内判定的语言集合.

Thm 5.22 $NP = \bigcup_{c \ge 1} \mathrm{NTIME}(n^c)$



证明: ① 右 $\subseteq$ 左; ② 左 $\subseteq$ 右.

① 令 $L \subseteq \mathrm{NTIME}(n^c)$ $(c \ge 1)$, 存在二择 NTM N 在 $p(n) = d \cdot n^c$ ($d>0$ 常数) 时间内判定 L.

证书记录了采用的选择. 验证者根据该证书 w 检查 N 是否接受 x.

② 令 $L \in NP$, 构造一个 NTM N 来猜 w. 猜后模仿 verifier 验证即可.

NPC (NP完全问题)

Def 5.23 (Karp reduction)

Let $L, K \in \{0,1\}^*$. $L \le_p K$ (L is karp reducable to K) if $\exists$ poly-time TM M such that for all $x \in \{0,1\}^*$, $x \in L$ iff $M(x) \in K$.

和 Many-one 归约的唯一区别

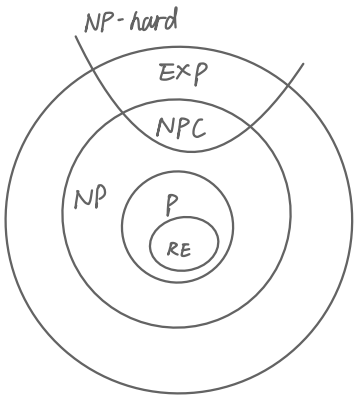
↑ 可计算即可

Lem 5.24 If $L_1 \le_p L_2$, $L_2 \le_p L_3$, then $L_1 \le_p L_3$.

$M_3 = M_2(M_1(x))$. $x \in L_1$ iff $M_3(x) \in L_3$.

Def 5.25 NP-hard (NP难) 比NPC更难.

$L \subseteq \{0,1\}^*$. $L \in$ NP-hard if for all $K \in NP$, $K \leq_p L$.



Def 5.26 NP-Complete (NPC, NP完全)

$L \in NPC$ if $L \in NP$ and $L \in$ NP-hard. 即 $NPC = NP \cap$ NP-hard.

Lem 5.27 If L is NP-hard and $L \in P$, the $P = NP$.

Lem 5.28 Let $L \in NPC$. $L \in P$ iff $P = NP$.

Cook-Levin Theorem

Cook(1971) Levin(1973) 证明了首个NPC问题. SAT.

Thm 5.29 SAT is NP-Complete.

variable: x, y, z. can take TRUE or FALSE.

Literal: $x, \neg x, y, \neg y, z, \neg z$ ..

clause: OR (disjunction, 析取). of ≥ 1 Literals.
(子句) $(\neg x \vee y, \neg y \vee z, \dots)$

formula: AND (conjunction, 合取) of ≥ 1 clauses.
(公式)

$\phi = (\neg x \vee y) \wedge (\neg y \vee z) \wedge (x \vee \neg z \vee y)$ $x=y=z$ (=1/0) 时成立.

$x=0$/ $y=1$ 且 $y=0$ 或 $z=1$ 且 $x=1$ 或 $z=0$ 或 $y=1$

$SAT = \{ \langle \phi \rangle, \phi$ is satisfiable $\} \in NP$.