计算理论期末复习

张志心

papercloud@zju.edu.cn

计算机科学与技术学院
浙江大学

2025 年 1 月 2 日

复习笔记

- https://fla.cuijiacai.com/
- https://note.tonycrane.cc/cs/tcs/toc/

语言定义

- 字母表 (Alphabet) Σ
- 字符串 (string) w
  - 一般用 ϵ 表示空串
  - Σ^* 表示任意字符串，Σ^+ 表示至少有一个字符的字符串
  - Σ^n 表示长度为 n 的字符串
  - 字符串拼接：$w_1 w_2$ 表示 w_1 和 w_2 的串联
  - 字符串反转：w^R 表示 w 的反转（在 01 串中，要注意和翻转 flip 的区别）
    - 例如：$01011^R = 11010$，$flip(01011) = 10100$
  - 字符串重复：$w^n = ww \cdot w$ 表示 w 重复 n 次
- 语言 (langauage) $L \subseteq 2^{\Sigma^*}$

自动机

- 确定性有限自动机 (DFA)
  - 五元组：$M = (K, \Sigma, \delta, s, F)$
  - 状态集合：K
  - 输入符号集合：Σ
  - 转移函数：$\delta : K \times \Sigma \to K$
  - 初始状态：$s \in K$
  - 接受状态集合：$F \subseteq K$
- 非确定性有限自动机 (NFA)
  - 五元组：$M = (K, \Sigma, \Delta, s, F)$
  - 状态集合：K
  - 输入符号集合：Σ
  - 转移集合：$\Delta \subseteq (K \cup \{\varepsilon\}) \times \Sigma \times K$
  - 初始状态：$s \in K$
  - 接受状态集合：$F \subseteq K$

字符串在自动机上的推导

- $(q, aw) \vdash (q', w)$ 表示字符串在自动机上从状态 q 读入字符串 aw 的第一个字符后转移到状态 q'
- $(q, a_1 a_2 \cdots a_n w) \vdash^* (q', w)$ 表示字符串 w 在自动机上从状态 q 读入 $a_1 a_2 \cdots a_n w$ 后，经过 n 步转移后，最终转移到状态 q'
- $(s, w) \vdash^* (q, e), q \in F$，表示自动机接受字符串 w
- $L(M)$ 表示自动机接受的字符串组成的语言.

正则语言

- 被有限自动机接受的语言/存在正则表达式的语言
- 正则表达式：
  - 字符：$\alpha \in \Sigma$
  - $L(\emptyset) = \emptyset, \forall a, \in \Sigma L(a) = \{a\}$
  - $L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
  - $L(R_1 R_2) = L(R_1)L(R_2)$ $(A\emptyset = \emptyset, A\varepsilon = A)$
  - $L(R^*) = L(R)^*$
  - 优先级：$Star > Conxatenation > Union$
- 正则语言在交、并、补、连接、Kleene Star 运算下均封闭.
  - Union：$A \cup B = \{\alpha \mid \alpha \in A \vee \alpha \in B\}$
  - Concatenation：$AB = \{\alpha_1\alpha_2 \mid \alpha_1 \in A, \alpha_2 \in B\}$
  - Kleene Star：$A^* = \{\epsilon\} \cup AA^* = \{\alpha_1\alpha_2 \cdots \alpha_n \mid n \geq 0, \alpha_i \in A\}$

Pumpumping Theorem

是正则语言的**必要不充分**条件：令 L 为一个正则语言，存在一个整数 $p > 0$ （pumping length）使得对于任意长度大于等于 p 的字符串 $w \in L$，都可以划分为 $w = xyz$，其中：

- $|y| > 0$
- $|xy| \leq p$
- $\forall i \geq 0, xy^i z \in L$

正则语言的判定

正则语言：不可计数、没有记忆、只能存在有限种结构.
有限自动机可以识别的：有限的子串（注意对个数取模涉及的状态是有限的）
比如：（开头、结尾、中间）包含某一种模式串的所有串，某个字符出现个数模 K 意义下满足某种条件的串（或是出现个数具有一个有限的限制——大于 K 或者小于 K）

1. $\{a^m b^n \mid 3m = 2n, m, n \in \mathbb{N}\}$ （不是）
2. $\{a^{2m} b^{3n+2021} \mid m, n \in \mathbb{N}\}$ （是）
3. $\{a^m b^n \mid m + n = 2021\}$ （是）
4. $\{a^m b^n \mid m \neq n \pmod 3, m, n \in \mathbb{N}\}$ （是）
5. $\{a^m b^n c^k \mid m + k \neq n \pmod 3, m, n, k \in \mathbb{N}\}$ （是）
6. $\{a^m b^n \mid m - n = 3, m, n \in \mathbb{N}\}$ （不是）
7. $\{a^m b^n c^k \mid m \neq k, m, n, k \in \mathbb{N}\}$ （不是）
8. $\{a^m \mid m \geq 2024\}$ （是）
9. $\{w \mid w = xabby, x, y \in \{a, b\}^*, x$ has odd number of a's and y has at least 2024 b's$\}$ （是）
10. A is a regular language, PREFIX$(A) = \{u : uv \in A$ for some $v\}$ （是）

上下文无关文法

- 四元组：$G = (V, \Sigma, S, R)$
- 其中 V 是所有符号集，而 Σ 是所有终结符号（terminal）.
- $S \in V \setminus \Sigma$ 表示起始符号（start symbol）.
- $R \subseteq (V \setminus \Sigma) \times V^*$ 表示产生式.
- $w \Rightarrow_G^* w'$ 表示串 w 可以根据产生式推导出 w'. (单步推导 $xAy \Rightarrow_G xwy, (A, w) \in R$)
- 生成字符串：$S \Rightarrow_G^* w$
- 语言：$L(G) = \{w \in \Sigma^* \mid S \Rightarrow_G^* w\}$
- 上下文无关文法生成上下文无关语言.
- 上下文无关文法可以转化成 CNF 形式（Chomsky Normal Form）：保证生成一个长度为 n 的串经过 $2n - 1$ 步.
- 关于并、连接、Kleene Star 运算封闭，关于交和补不一定封闭.

构造上下文无关文法

- 生成 $\Sigma(\{a,b\})^*$：$S \to aS|bS|\varepsilon$
- 生成回文串：$S \to aSa|bSB|\varepsilon$
- 生成括号序列：$S \to (S)S|\varepsilon$

构造一个上下文无关文法，使得它生成的语言为：

$$\{w\#x : x, w \in \{0,1\}^*, w^R \text{ is a substring of } x\}$$

答案：$S \to AB, B \to 1B|0B|\varepsilon, A \to 0A0|1A1|\#B$

$$\{w \in \{0,1\}^* \text{no prefix of w has more } 0\text{'}s \text{ than } 1\text{'}s\}$$

答案：$S \to 1S|1S0S|\varepsilon$

下推自动机

- NFA + stack
- 六元组：$P = (K, \Sigma, \Gamma, \Delta, s, F)$
- Γ 表示中会出现的符号集合.
- Δ 表示转移集合 $(K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^*) \times (K \times \Gamma^*)$
  - $((q_1, a, x), (q_2, y))$ 表示状态 q_1 读入符号 a 后，弹出栈顶串 x，转移到状态 q_2，并依次把 y 的每个字符压入栈顶.
- 推导 $(q, ww', x) \vdash_P^* (q', w', y)$ 表示下推自动机从状态 q 读入串 ww' 的前缀 w，转移到状态 q，同时栈中元素从底到顶部由 x 变成 y.
- 接受字符串 $(s, w, \varepsilon) \vdash_P^* (q, \varepsilon, \varepsilon), q \in F$，表示下推自动机接受串 w.
- 语言：$L(P) = \{w \in \Sigma^* \mid P \text{ accepts } w\}$
- PDA 都可以转化成 simple PDA 使得每步对栈的操作要么是弹出一个字符，要么是压入一个字符.

上下文无关语言的判定

CFL 可以在 RL 的基础上识别一组数量关系（倍数关系，大小关系），但同一时刻只能识别一组．用状态转移图来识别子串的构造，用栈来识别数量关系．

1. $\{a^n b^n c^n \mid n \in \mathbb{N}\}$ （不是）
2. $\{a^m b^n c^k \mid n = m + k, m, n, k \in \mathbb{N}\}$ （是）
3. $\{a^m b^n c^k \mid 2024m + 4047n \leq k, m, n, k \in \mathbb{N}\}$ （是）
4. $\{a^{3m} b^{5n} c^{4k} \mid 2024m + 4047n \leq k, m, n, k \in \mathbb{N}\}$ （是）
5. $\{a^m b^n \mid n \neq m, m, n \in \mathbb{N}\}$ （是）
6. $\{w \in \{a, b\}^* : \#a = \#b$abaa and babb are not substrings of $w\}$ （是）
7. $\{a^m b^n c^k d^l \mid m = n, k \neq l, m, n, k, l \in \mathbb{N}\}$ （是）
8. $\{wcuw^R v \mid u, v, w \in \{a, b\}^*,$ and $|v|$ is odd$\}$ （是）
9. A is a context-free language, PREFIX$(A) = \{u \mid uv \in A$ for some $v\}$. （是）
10. A is a context-free language, REVERSE$(A) = \{u \mid u^R \in A\}$ （是）
11. A is a context-free language, SUFIX$(A) = \{u \mid vu \in A$ for some $v\}$. （是）
12. A is a context-free language，SUBSTRING$(A) = \{u \mid u$ is a substring of $A\}$ （是）
13. A is a context-free language, SUBSEQUENCE$(A) = \{u \mid u$ is a subsequence of $A\}$ （是）
14. $\{a^i b^j c^k d^l \mid i = k$ and $j = l\}$ （不是）
15. $\{a^i b^j c^k d^l \mid i = l$ and $j = k\}$ （是）

上下文无关语言的判定

证明是 CFL/RL 一般采用直接构造自动机/文法的方法，证明不是 CFL/RL 一般使用 pumping theorem.

除此之外，利用 RL / CFL 的运算的封闭性质，来证明一些语言不可判定/可判定也是常见的思路. 注意 $\emptyset$ 和 Σ^* 都是 CFL. 一个 CFL L 和一个 RL R 满足 $L - R$ 是 CFL，但 $R - L$ 不一定是.

我们还可以对闭包性质做一个复合，比如，要证明如下语言是 CFL:

$$M(L) = \{w \mid (w \in L) \wedge (\forall v \in \Sigma^+ . vw \notin L)\}, L \text{ is a CFL}$$

注意到 $M(L) = REVERSE(NOT - TRUE - PREFIX(REVERSE(L)))$，因此考虑下一步证明如下语言是 CFL 的即可.

$$NOT - TRUE - PREFIX(L) = \{w \mid (w \in L) \wedge (\forall v \in \Sigma^+, wv \in L)\}$$

Pumping Theorem

是上下文无关语言的**必要不充分**条件：令 L 为一个上下文无关语言，存在一个整数 $p > 0$ (pumping length) 使得对于任意长度大于等于 p 的字符串 $w \in L$，都可以划分为 $w = uvxyz$，其中：

- $|v| + |y| > 0$
- $|vxy| \leq p$
- $\forall i \geq 0, uv^i xy^i z \in L$

例题

证明以下语言不是 CFL:

1. $L = \{0^p \mid p \text{ is a prime}\}$
2. $L = \{0^i 1^j \mid i^2 \leq j\}$
3. $L = \{0^i 1^j 2^k \mid i = \max(j, k)\}$
4. $L = \{t_1 \# t_2 \# \cdots \# t_k \mid k \leq 2 \wedge (\forall i, t_i \in \{0,1\}^*) \wedge (\exists i \neq j, t_i = t_j)\}$

例题

1. 假设 L 是 CFL，设 p' 为其 pumping length，考虑 $z = 0^p \in L, p > p'$，那么 $z = uvwxy$，其中 $|vx| \geq 1$，那么 $uv^{p+1}wx^{p+1}y = 0^{p+|vx|\times p} = 0^{p\times(1+|vx|)}$，因此 $uv^{p+1}wx^{p+1}y \notin L$，矛盾.
2. 假设 L 是 CFL，设 p 为其 pumping length，考虑 $0^{p+1}1^{(p+1)^2}$，根据 pumping thm 划分为 $vwxyz$，若 wy 中只含有 1，则 vw^0xy^0z 中 0 的个数为 $p+1$，而 1 的个数不足 $(p+1)^2$，不满足条件，因此 wy 中至少含有一个 0，至多有 $(p+1)^2$ 个 1．因此 vw^2xy^2z 中至少含有 $p+2$ 个 0，至多含有 $(p+1)^2+(p+1)$ 个 1，而 $(p+2)^2 = (p+1)^2+2(p+1)+1 > (p+1)^2+(p+1)$，不满足条件.

例题

3 假设 L 是 CFL，设 n 为其 pumping length，考虑 $0^{n+1}1^n2^{n+1} \in L$，由 pumping thm 划分 $vwxyz$. 因为 $|wxy| \leq n$ 所以 wxy 不同时含有 $0, 1, 2$，若 wxy 仅含 0，则 vw^2xy^2z 中 0 的数量至少为 $n+2$，而 $1, 2$ 的数量为 $n, n+1$，矛盾；若 wxy 中仅含 1 或 2，则 vw^3xy^3z 中 $1, 2$ 数量较大的值至少为 $n+2$ 矛盾；若 wxy 同时含有 0 和 1，设 $wxy = 0^p1^q, pq > 0$，则 vw^3xy^3z 中 $1, 2$ 数量较大值为 $n+2q$，0 的数量为 $n+1+2p$，奇偶性不同，矛盾.

4 假设 L 是 CFL，设 p 为其 pumping length，考虑 $0^p1^p\#0^p1^p \in L$，由 pumping thm 划分为 $vwxyz$，若 wy 中包含 $\#$，那么 vw^0xy^0z 中不包含 $\#$，不属于 L；若 wxy 位于字符串前 $2p$ 位，那么很显然 vw^0xy^0z 中 $\#$ 前后两段长度不同，也不属于 L；同理，wxy 也不能位于字符串后 $2p$ 位；因为 $|wxy| \leq p$，所以 $w = 1^{k_1}, y = 0^{k_2}, 0 \leq k_1, k_2 \leq p\ k_1k_2 > 0$，因此若 $k_1 > 0$，那么 vw^2xy^2z 中 $\#$ 前 1 的个数多于 $\#$ 后 1 的个数，因此前后不相等，否则 $k_2 > 0$ 也会导出同样的结果. 所以 L 不满足 pumping thm，不是 CFL.

例题

定义 $\bowtie$ 运算：

$$L \bowtie R = \{w \mid w = x_1 y_1 x_2 y_2 \cdots x_n y_n \text{ for some } n, x_1 x_2 \cdots x_n \in L, y_1 y_2 \cdots y_n \in R, x_i, y_i \in \Sigma\}$$

1. 如果 L 是 CFL，R 是 RL，则 $L \bowtie R$ 是 CFL（直接考虑在 L 的 PDA 中的自动机中加入一维来跑 R 的 DFA）

例题

2. $\bowtie$ 在 CFL 中不封闭:

令 $L = \{0^i 1^i 2^j : i, j \geq 2\}, R = \{0^j 1^i 2^i : i, j \geq 2\}$，其中 L, R 都是 context-free 的，下证 $L \bowtie R$ 不是 context-free 的，从而导出 CFL 不在 $\bowtie$ 运算下不封闭.

假设 $L \bowtie R$ 是 CFL 的，设其 pumping length 为 n，考虑 $(00)^n(11)^n(22)^n \in L \bowtie R$，则其存在满足 pumping 引理的一个划分 $vwxyz$, 若 wxy 中仅含 0/仅含 1/仅含 2，都会导致 vw^3xy^3z 中奇数位置上 0 与 1 的数量不相同或者偶数位置上 1 与 2 的数量不相同. 由于 $|wxy| \leq n$，所以 wxy 为 $0^{k_1}1^{k_2}$ 或者 $1^{k_3}2^{k_4}$ 的形式，其中 $k_1, k_2, k_3, k_4 > 0$. 若 $wxy = 0^{k_1}1^{k_2}$，则会导致 vw^3xy^3z 中偶数位置上 1 与 2 的数量不相同；若 $wxy = 1^{k_3}2^{k_4}$，则会导致 vw^3xy^3z 中奇数位置上 0 与 1 的数量不相同. 所以 $L \bowtie R$ 不满足 pumping 引理，所以不是上下文无关语言.

确定性图灵机

- 五元组：$M = (K, \Sigma, \delta, s, H)$
- 状态集合：K，$s \in K$ 表示起始状态，$H \subseteq K$ 表示停机状态集合.
- 纸带上可以出现的集合：Σ，包括 $\triangleright$ $\sqcup$ 两种特殊符号 (分别是纸带左端和空格)
- 转移函数：$\delta : (K \setminus H) \times \Sigma \to K \times (\{\leftarrow, \rightarrow\} \cup (\Sigma \setminus \{\triangleright\}))$
  - 非停机状态读入一个字符后可以转移到另一个状态，并在纸带上左右移动读写头/在读写头的位置写一个符号.
- 接受字符串：$(s, \triangleright \sqcup w) \vdash^* (q, w'), q \in H$
- 半判定 (semidecides)：$L(M) = \{w \in \Sigma^* \mid (s, \triangleright \sqcup w) \vdash^* (q, w'), q \in H\}$
- 判定 (decides)：两类 Halting states：y 和 n 表示接受和不接受，$L(M) = \{w \in \Sigma^* \mid (s, \triangleright \sqcup w) \vdash^* (y, w')\}$，并且 M 无论输入时什么都会停机.
- 被图灵机判定的语言是 recursive / decidable 的.
- 被图灵机半判定的语言是 recursively enumerate / recoginizable 的.
- 可计算函数：存在图灵机 M，$(s, \triangleright \sqcup w) \vdash^* (q, \triangleright \sqcup f(w)), q \in H, w \in \Sigma_0^*$

非确定图灵机（NTM）

- 五元组：$M = (K, \Sigma, \Delta, s, H)$
- 转移集合：$\Delta \subseteq ((K \setminus H) \times \Sigma) \times (K \times (\{\leftarrow, \rightarrow\} \cup (\Sigma \setminus \{\triangleright\}))$
- 半判定：存在一个转移分支可以停机
- 判定：所有分支在有限步内停机并且存在一个分支走到 yes 状态.
- NTM 可以由 DTM 来模拟.（以指数级的代价）

判定问题

- 编码：自动机、图灵机，单个字符串可以被编码.
- 语言不可以被编码：Σ^* 的幂集不可数（用对角线方法来证明）.
- 可判定问题：将问题编码成字符串组成的语言，并存在图灵机判定它.
- 一个语言和其关于一个可判定语言的补集如果都是半判定的，则这个语言是判定的.（可以用来证明一个语言不是半判定的）
- 规约：$A \leq B$ 如果存在 $f: \Sigma^* \to \Sigma^*$ 满足 $x \in A$ 当且仅当 $f(x) \in B$.

经典的可判定语言

以下语言都是可判定的：

- $A_{DFA} = \{\text{"}D\text{""}w\text{"} : D \text{ is a DFA that accepts } w\}$
- $A_{NFA} = \{\text{"}N\text{""}w\text{"} : N \text{ is a NFA that accepts } w\}$
- $A_{REX} = \{\text{"}R\text{""}x\text{"} : R \text{ is a regular expression that matches } x\}$
- $E_{DFA} = \{\text{"}D\text{"} : D \text{ is a DFA that accepts none strings}\}$
- $EQ_{DFA} = \{\text{"}D_1\text{""}D_2\text{"} : D_1, D_2 \text{ are DFAs that accept the same language}\}$
- $ALL_{DFA} = \{\text{"}D\text{"} : D \text{ is a DFA that accepts all strings}\}$
- $A_{CFG} = \{\text{"}C\text{""}w\text{"} : C \text{ is a CFG that generates } w\}$
- $A_{PDA} = \{\text{"}P\text{""}w\text{"} : P \text{ is a PDA that accepts } w\}$
- $E_{CFG} = \{\text{"}C\text{"} : C \text{ is a CFG that generates none strings}\}$
- $E_{PDA} = \{\text{"}P\text{"} : P \text{ is a PDA that accepts none strings}\}$

注意：$ALL_{PDA}, EQ_{PDA}, ALL_{CFG}, EQ_{CFG}$ 都不是可判定的语言．而 ALL_{DFA} 是可判定的，因为正则表达式可计算．

停机问题

以下语言是不可判定的语言：

- $H = \{\text{"}M\text{""}w\text{"} : M \text{ is a TM that halts on } w\}$
- $H_e = \{\text{"}M\text{"} : M \text{ is a TM that halts on } e\}$
- $SOME_{TM} = \{\text{"}M\text{"} : M \text{ is a TM that halts on some input}\}$
- $E_{TM} = \{\text{"}M\text{"} : M \text{ is a TM that halts on none input}\}$
- $ALL_{TM} = \{\text{"}M\text{"} : M \text{ is a TM that halts on all input}\}$
- $EQ_{TM} = \{\text{"}M_1\text{""}M_2\text{"} : M_1, M_2 \text{ are TMs that halts on the same input}\}$
- $R_{TM} = \{\text{"}M\text{"} : M \text{ is a TM that } L(M) \text{ is regular}\}$
- $CF_{TM} = \{\text{"}M\text{"} : M \text{ is a TM that } L(M) \text{ is context-free}\}$
- $REC_{TM} = \{\text{"}M\text{"} : M \text{ is a TM that } L(M) \text{ is recursive}\}$

证明是否可判定的方法

- 可以规约到一个可（半）判定语言则可判定，由一个不可（半）判定语言规约到它则它本身也不可（半）判定.
- 可以通过枚举的方法来检验是否满足一个性质则可半判定.（这里的性质必须是“至少有 k 个”、" 存在..." 这样的形式，那么可以根据字典序从小到大逐步增加并行地图灵机来同时检验结果，当检验到满足条件之后退出. 注意检验问题也应该是递归可枚举的 (否则无法停机). 具体可以参照 29 页例题.
- 如果 $\overline{L}$ 是不可判定的，则 L 是不可判定的.

例题

$L = \{ \langle M_1, M_2 \rangle \mid M_1, M_2$ are two turing machines and $L(M_1) = L(M_2) \}$ is decidable.
(**Hint**: $\mathrm{E_{TM}}$ is undecidable.)
只需证明 $\mathrm{E_{TM}} \leq L$，由于 $\mathrm{E_{TM}}$ 不可判定，则 L 也不可判定．证明如下：
考虑 $\mathrm{E_{TM}} \leq L$ 归约到 L 的过程，(对于输入 $\langle M \rangle$)

1. 首先计构造 $M_1 = M$, M_2 对于任何输入都执行 loop 循环操作，使其用不停机;
2. 判断是否 $\langle M_1, M_2 \rangle \in L$;
3. - 若是，因为 $L(M_2) = \emptyset$，所以 $L(M) = \emptyset$，则接受;
4. - 若否，则拒绝．

例题

如果 L 是一个不可判定的语言，那么

$$R = \{w \mid (w = 0x \wedge x \in L) \vee (w = 1x \wedge x \in \overline{L})\}$$

不是递归可枚举的.
否则，你可以给 R 的自动机同时输入 $0x$ 和 $1x$，若 R 递归可枚举，则总有一边会停机，因此导出 L 可判定，矛盾.

Rice's Theorem

令 $\mathcal{L}(P)$ 表示具有非平凡性质 (不为空或全集) P 的所有递归可枚举语言的集合，$R(P) = \{"M" : M \text{ is a TM that} L(M) \in \mathcal{L}(P)\}$，则 $R(P)$ 不可判定.

考虑证明 $R(P)$ 或者 $\overline{R(P)}$ 可以被 H 规约到，具体的，考虑不包含 $\emptyset$ 的那一边. 统一的证明方法: 设 $L \in \mathcal{L}(P)$，M_L 是 L 的 TM.

构造图灵机 $M^* =$ on input x: a. run M on w, b. run M_L on x.

那么 $"M^*" \in R(P) \Leftrightarrow "M""w" \in H$.

例题

证明一个语言 L 是递归可枚举的当且仅当存在一个递归语言 R 满足：(这里用 $\langle\cdot\rangle$ 表示编码)

$$L = \{x \mid \exists y, \langle x, y\rangle \in R\}$$

"$\Leftarrow$"：对于一个递归语言 R，$L = \{x : \exists y, \langle x, y\rangle \in R\}$ 是递归可枚举语言．理由如下：对于 R 的图灵机 M，$L(M) = R$，对 M 增加转移，使得 M 在到达拒绝状态 q_{rej} 后执行 'loops'，使其永远无法停机．接下来构造另一台可编程的图灵机 M' 使得 $L(M') = L$．其工作流程如下：对于输入 $w \in \Sigma^*$，将 Σ^* 中所有字符串，按照长度从小到大，字典序从小到大的顺序排序，令 $\Sigma = \{s_i\}_{i=1}^{\infty}$：

1. 创建图灵机 $M_1 = M$，输入 $\langle w, s_1\rangle$，并将其转移至第 1 步，若其停机，则 M^* 停机；
2. 创建图灵机 $M_2 = M$，输入 $\langle w, s_2\rangle$，并将 M_1, M_2 转移至第 2 步，若 M_1 或 M_2 停机，则 M^* 停机；
3. 创建图灵机 $M_3 = M$，输入 $\langle w, s_3\rangle$，并将 M_1, M_2, M_3 转移至第 3 步，若 M_1 或 M_2 或 M_3 停机，则 M^* 停机；
4. .....
5. 第 i 步，创建图灵机 $M_i = M$，输入 $\langle w, s_i\rangle$，并将 $M_1, M_2, \cdots, M_i$ 转移至第 i 步，若 $M_1, \cdots, M_i$ 有一台停机，则 M^* 停机；

可以证明，M^* 对于输入 x 停机，当前仅当存在 $y \in \Sigma^*$，$\langle w, y\rangle \in R$．

例题

"⇒"：对于递归可枚举语言 L，L 的图灵机 M 满足 $L(M) = L$，对于 $w \in L$，令

$$n_w = \{\text{number of steps for TM M on string w to halt}\}$$

构造 $R = \{\langle x, y\rangle : x \in L \wedge y \in \mathbb{N} \wedge y \geq n_x\}$，显然 $L = \{x : \exists y, \langle x, y\rangle \in R\}$，下证明 R 是递归语言：
构造图灵机 M'，对于输入 w，其工作流程如下：
1. 获取 $w = \langle x, y\rangle$，其中 y 为一个整数，x 为一个字符串（若获取失败则拒绝该串）；
2. 创建图灵机 M，输入 w，并将其转移到第 y 步，若其停机，则接受该串，否则拒绝该串.
容易证明 $L(M') = R$. 所以 R 是递归语言.

复杂度理论

- 令 n 表示输入的长度.
- DTIME$(t(n))$ 表示所有能在 $O(t(n))$ 时间内停机的标准（单头、单带）图灵机集合.
- $P = \bigcup_{k\geq 0}$ DTIME(n^k)
- NP：非确定图灵机在 $O(poly(n))$ 内判定的语言. 等价于确定性图灵机多项式时间内可判定.
- $A \leq_P B$ 表示 A 在多项式时间内可以规约到 B，规约函数 f 在多项式时间内可计算.
- 如果 $A \leq_P B, B \in P$ 则 $A \in P$.
- 如果一个语言满足任意一个 NP 可以在多项式时间内规约到它，则它是 NP-完全的. (如：SAT 问题).
- 如果 A 是 NPC，$B \in NP$，$A \leq_P B$，则 $B \in NPC$.

结论：every CFL is in P. SAT 问题，极大团问题（无向图是否有一个团包含至少 k 个节点），顶点覆盖问题（是否存在至多 k 个顶点的覆盖）都是 NPC 的.

例题

Define

$$\text{01-ROOT} = \{\ \langle p \rangle \mid p \text{ is a polynomial in } n \text{ variables with integer coefficients such that } p(x_1, x_2, \cdots, x_n) = 0 \text{ for some } (x_1, x_2, \cdots, x_n) \in \{0,1\}^n \ \}.$$

Show that 01-ROOT is $\mathcal{NP}$-complete.

例题

要证明 01-ROOT 是 $\mathcal{NP}$-complete，只需证明 3-SAT $\leq_P$ 01-ROOT．只需构造一个函数：f: {Boolean Formulas} $\to$ {polynomials with coefficients be 0 or 1 and finite variables}，使得任意 Boolean Formula $\Phi = \Phi(x_1, x_2, \cdots, x_n)$，$f(\Phi) = p(x_1, x_2, \cdots, x_n)$，满足 Φ 是一个可满足的布尔表达式，当前仅当 $f(\Phi)$ 有零点．

以下给出在**多项式时间内**计算 $f(\Phi)$ 的方法：函数 g 按照如下方式将布尔表达式转化为多项式：

- $g(x_i) = x_i, x_i = 1, \cdots, n$;
- $g(\phi_1 \wedge \phi_2) = \phi_1 \phi_2$;
- $g(\neg\phi) = 1 - \phi$;
- $g(\phi_1 \vee \phi_2) = 1 - (1 - \phi_1)(1 - \phi_2)$

根绝表达式的优先级，对不等式按照如上规则进行递归构造，得到 $g(\Phi)$．令 $f(\Phi) = 1 - g(\Phi)$．考虑上述构造过程因为每个变量只设计进行不超过 n 次运算，所以在递归过程中，每个变量只会被 g 函数计算不超过 n 次，计算 $g(\Phi)$ 和 $f(\Phi)$ 的时间复杂度均在 $O(n^2)$ 之内．

例题

a. $\mathcal{P}$ 和 $\mathcal{NP}$ 在连接运算下封闭：**True**，对于 $\mathcal{P}$ 问题，设有两个 $\mathcal{P}$ 语言 A 和 B，并设对应的可在多项式时间内接受它们的图灵机为 M_A 和 M_B，对于一个串 w，要判断其是否属于 AB，可以按照如下过程，用 $O(n)$ 的时间复杂度，枚举断点，将 w 分成 $w = w_1 w_2$，接下来分别使用 $O(p_A(n))$ 和 $O(p_B(n))$ 的时间复杂度，判断 w_1 是否属于 A 并且 w_2 属于 B，如果上述结论对于某个断点成立，则接受 w，否则拒绝．时间复杂度为 $O(n(p_A(n) + p_B(n)))$，所以 AB 仍然是 $\mathcal{P}$ 语言．
对于 $\mathcal{NP}$ 问题，设有两个 $\mathcal{NP}$ 语言 A 和 B，并设对应的可在多项式时间内接受它们的非确定性图灵机为 M_A 和 M_B，与上述证明方法类似，同样可以通过枚举断点来构造接受 AB 的非确定性图灵机．所以 AB 仍然是 $\mathcal{NP}$ 语言．

b. $\mathcal{P}$ 在补运算下封闭：**True**，设 $A \in \mathcal{P}$，因为 A 是可判定语言，所以 $\overline{A} \leq_P A$，所以 $\overline{A} \in \mathcal{P}$．下给出将 $\overline{A}$ 归约到 A 的方法：
   - 对于输入 w;
   - 判断是否 $w \in A$：若是，则拒绝；若否，则接受．