# 计算理论导论 课程笔记

酥雨
zusuyu@stu.pku.edu.cn

April 10, 2022

## 目录

# 1 正则语言

**定义 1.1 (Deterministic Finite Automaton, DFA).** (确定性) 有限自动机是一个五元组 $(Q, \Sigma, \delta, q_0, F)$, 其中

- $Q$ 是称为**状态**的有限集.
- $\Sigma$ 是称为**字符集**的有限集.
- $\delta : Q \times \Sigma \to Q$ 被称为**转移函数**.
- $q_0 \in Q$ 称为**起始态**.
- $F \subseteq Q$ 称为**接受态 (终止态) 集合**.

称字符串 $w = w_1 w_2 \cdots w_m (w_i \in \Sigma)$ 可以被 DFA $M = (Q, \Sigma, \delta, q_0, F)$ 接受, 如果存在状态序列 $r_0, r_1, \cdots, r_m \in Q$ 满足(i)$r_0 = q_0$, (ii)$r_{i+1} = \delta(r_i, w_{i+1})$ $(\forall i = 0, 1, \cdots, m-1)$, (iii)$r_m \in F$.

所有可被 $M$ 识别的字符串 $w$ 构成集合 $A$, 则称 $A$ 是 DFA $M$ 的语言 (或者说 DFA $M$ 识别/接受 $A$), 记为 $L(M) = A$.

**定义 1.2 (正则语言).** 正则语言就是能够被有限自动机识别的语言.

**定义 1.3 (正则操作).** 定义如下三种正则操作

- **Union**: $A \cup B = \{x | x \in A \text{ or } x \in B\}$.
- **Concatenation**: $A \circ B = \{xy | x \in A \text{ and } y \in B\}$.
- **Star**: $A^* = \{x_1 x_2 \cdots x_k | k \geqslant 0 \text{ and each } x_i \in A\}$.

**注 1.1.** 补集 $\overline{A} = \Sigma^* - A$ 操作在正则语言下是封闭的: 只需要把终止态集合 $F$ 改成 $Q - F$ 即可.

**定理 1.1.** 正则操作 union 在正则语言下是封闭的: 把两个自动机放在一起跑就行了.

由于只利用已有的有限自动机模型证明 concatenation 和 star 的封闭性是困难的, 我们引入“非确定性”.

**定义 1.4 (Nondeterministic Finite Automaton, NFA).** 非确定性有限自动机是一个五元组 $(Q, \Sigma, \delta, q_0, F)$, 其中 $\delta$ 不再是 $Q \times \Sigma \to Q$ 的函数, 而是 $Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ 的, 其中 $\mathcal{P}$ 表示幂集, $\Sigma_\varepsilon$ 表示 $\Sigma \cup \{\varepsilon\}$.

相应的, 称字符串 $w = w_1 w_2 \cdots w_m (w_i \in \Sigma)$ 可以被 NFA $N = (Q, \Sigma, \delta, q_0, F)$ 接受, 如果 $w$ 可以写成 $w = y_1 y_2 \cdots y_{m'} (y_i \in \Sigma_\varepsilon)$, 且存在状态序列 $r_0, r_1, \cdots, r_{m'} \in Q$ 满足(i)$r_0 = q_0$, (ii)$r_{i+1} \in \delta(r_i, y_{i+1})$ $(\forall i = 0, 1, \cdots, m'-1)$, (iii)$r_{m'} \in F$.

**注 1.2.** DFA 的每个状态对每种字符都有恰好一条转移出边, 而相对的, NFA 可能有零条、一条或者多条, 有几条出边就表示会创建出多少个独立的“后继进程”. 此外还存在 $\varepsilon$ 的出边, 表示可以不输入任何字符创建进程.

**定理 1.2 (NFA 与 DFA 的等价性).** 任何 NFA 都存在等效的 DFA.

证明. 对 $k$ 个状态的 NFA, 构造一个 $2^k$ 个状态的 DFA, 每个状态表示“可能处在的 NFA 状态”的子集.

形式化的, 对于 NFA $M = (Q, \Sigma, \delta, q_0, F)$, 构造 DFA $M' = (Q', \Sigma, \delta', q_0', F')$, 其中

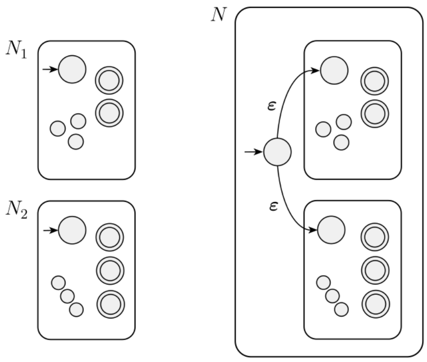
- $Q' = \mathcal{P}(Q)$.
- $\forall R \in Q', a \in \Sigma, \delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$.
- $q_0' = \{q_0\}$.
- $F' = \{R \in Q' | R \cap F \neq \varnothing\}$.

□

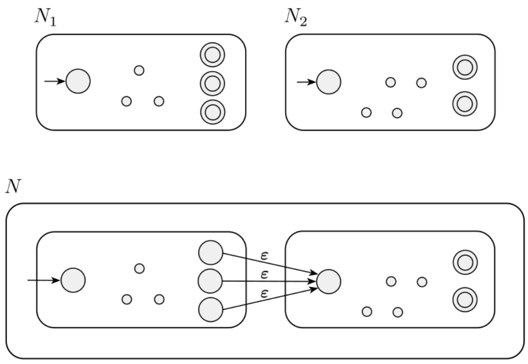
**推论 1.1.** 一个语言是正则的当且仅当可以被一台非确定性有限自动机识别.

**定理 1.3.** union, concatenation 和 star 在正则语言下都是封闭的.

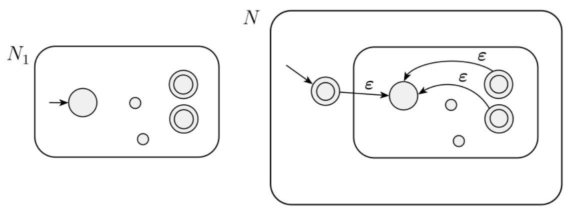
证明. 不多说了看图.




(a) proof for union




(b) proof for concatenation




(c) proof for star

□

**定义 1.5 (正则表达式).** 称 $R$ 是正则表达式, 如果 $R$ 为

- $\{a\}$, 其中 $a$ 是字符集 $\Sigma$ 中的某个元素
- $\{\varepsilon\}$, 其中 $\varepsilon$ 表示空串
- $\varnothing$
- $(R_1 \cup R_2)$, 其中 $R_1, R_2$ 是某两个正则表达式
- $(R_1 \circ R_2)$, 其中 $R_1, R_2$ 是某两个正则表达式
- $(R_1^*)$, 其中 $R_1$ 是某个正则表达式

**例 1.1.** 对于任意正则表达式 $R$, $R \cup \varnothing = R \circ \varepsilon = R$, $R \circ \varnothing = \varnothing$, $\varnothing^* = \{\varepsilon\}$.

**定理 1.4 (正则表达式与有限自动机的等价性).** 一个语言是正则的当且仅当它可以被一个正则表达式描述.

证明. "$\Leftarrow$" 的证明是简单的, 只需要根据正则表达式 $R$ 构造 NFA, 利用 "union, concatenation, star 的封闭性" 的构造性证明即可.

"$\Rightarrow$" 的证明中, 我们引入 GNFA 的定义 (每条转移边上的 label 是一个正则表达式), 然后分别展示如何把 DFA 转化成 GNFA 以及如何根据 GNFA 构造正则表达式.

DFA 转 GNFA 是简单的——只需要额外加入两个状态表示 $q_{\text{start}}$ 和 $q_{\text{accept}}$ 即可.

观察到一个 GNFA 有 $k \geqslant 2$ 个状态. 如果 $k = 2$, 那么 $q_{\text{start}}$ 到 $q_{\text{accept}}$ 的转移边上的正则表达式就是该有限自动机对应的正则表达式. 如果 $k > 2$, 那么考虑选出一个状态 $q_{\text{rip}}$ 删除, 此时对于 $q_i, q_j \in Q \setminus \{q_{\text{rip}}\}$, 如果

$\delta(q_i, q_{\text{rip}}) = R_1, \delta(q_{\text{rip}}, q_{\text{rip}}) = R_2, \delta(q_{\text{rip}}, q_j) = R_3, \delta(q_i, q_j) = R_4$, 则修改 $\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$. 归纳即可. □

**定义 1.6 (Generalized Nondeterministic Finite Automaton, GNFA).** 广义非确定性有限自动机是一个五元组 $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, 其中 $\delta$ 是 $(Q \setminus \{q_{\text{accept}}\}) \times (Q \setminus \{q_{\text{start}}\}) \to \mathcal{R}$ 的转移函数, $\mathcal{R}$ 表示字符集 $\Sigma$ 上的所有正则表达式. 注意不失一般性地要求了只有唯一的接受态, 以及 $q_{\text{start}} \neq q_{\text{accept}}$.

**定理 1.5 (Pumping Lemma for Regular Language).** 如果 $A$ 是正则语言, 那么存在一个数 $p$ (称为 **pumping length**), 使得对于任意 $A$ 中长度至少为 $p$ 的字符串 $s$, $s$ 都可分成三部分 $s = xyz$ 满足

- for each $i \geqslant 0$, $xy^iz \in A$,
- $|y| > 0$,
- $|xy| \leqslant p$.

*证明.* 取 pumping length $p$ 为识别此正则语言的 DFA $M$ 的状态集大小 $|Q|$. 对于任意长度至少为 $p$ 的 $s \in A$, 其经过的状态序列至少长为 $p+1$. 根据**鸽巢原理**, 存在一个状态 $q$ 经过了至少两次, 于是把从 $q_{\text{start}}$ 走到 $q$ 的部分视作 $x$, $q$ 回到自身的环视作 $y$, 从 $q$ 走到 $q_{\text{accept}}$ 的部分视作 $z$, 便构造出了划分. □

**注 1.3.** 利用 pumping lemma 可以证明某个语言 $B$ 不是正则语言, 通用的方式是: 先假设 $B$ 是正则的, 导出 pumping length $p$ 的存在性, 然后根据这个 $p$ 构造 $s \in B$, 并验证其**不能**被划分为 $s = xyz$. 第三个条件 $|xy| \leqslant p$ 有时也是有用的.

**例 1.2.** $B = \{0^n1^n | n \geqslant 0\}$ 不是正则语言.

*证明.* 假设 $B$ 是正则语言, 那么就存在 pumping length $p$. 考虑串 $0^p1^p$, 无论 $y$ 取其何种子串, $xyyz$ 都不可能 $\in B$. 因此 $B$ 不是正则语言. □

**例 1.3.** $C = \{w | w \text{ has an equal number of 0s and 1s}\}$ 不是正则语言.

*证明.* 假设 $C$ 是正则语言, 那么就存在 pumping length $p$. 考虑串 $0^p1^p$, 注意到我们要求了 $|xy| \leqslant p$, 所以 $y$ 只能包含 0, 此时 $xyyz \notin B$. 因此 $C$ 不是正则语言.

*另一种证法是: 考虑 $C \cap 0^*1^* = B$, 正则语言在 intersection 下是封闭的, 所以 $C$ 正则会导出 $B$ 正则.* □

**例 1.4.** $F = \{ww | w \in \{0,1\}^*\}$ 不是正则语言.

*证明.* 考虑串 $0^p10^p1$, 注意到 $y$ 只能包含 0, 从而 $xyyz \notin F$, 因此 $F$ 不是正则语言. □

**例 1.5.** $D = \{1^{n^2} | n \geqslant 0\}$ 不是正则语言.

*证明.* 考虑串 $1^{p^2}$. 由于 $|y| \leqslant p$, 所以 $|xyyz| = p(p+1) < (p+1)^2$ 不可能是完全平方数, $xyyz \notin D$, 说明 $D$ 不是正则语言. □

**例 1.6.** $E = \{0^i1^j | i > j\}$ 不是正则语言.

*证明.* 考虑串 $0^{p+1}1^p$, $y$ 只能包含 0, 且 $|y| > 0$, 因此 $xz$ 中 0 的个数不超过 1 的个数, $xz \notin E$, 说明 $E$ 不是正则语言. □

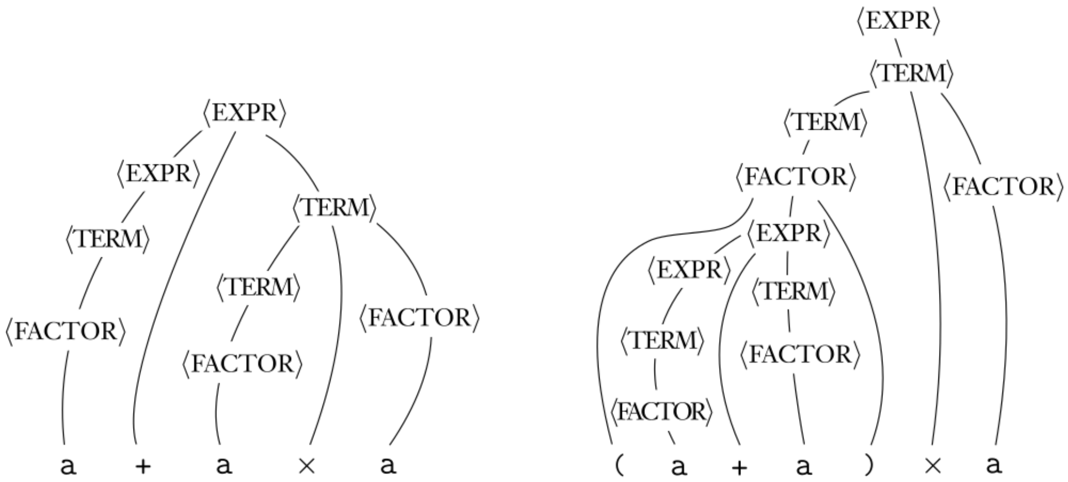
## 2 上下文无关文法

**定义 2.1 (Context-Free Grammar/Language, CFG/CFL).** 一个上下文无关文法是一个四元组 $(V, \Sigma, R, S)$, 其中

- $V$ 是称为**变量**的有限集,
- $\Sigma$ 是称为**终止符**的有限集, 与 $V$ 不交,
- $R$ 是称为**规则**的有限集, 是从 $V$ 到 $(V \cup \Sigma)^*$ 的映射,
- $S \in V$ 称为**起始变量**.

上下文无关语言就是上下文无关文法导出/生成的语言, 即 $\{w \in \Sigma^* | S \overset{*}{\Rightarrow} w\}$.

**定义 2.2 (parse tree).** 形如这样子的东西.




**命题 2.1.** CFG 的描述能力严格强于有限自动机 (或者正则表达式).

*证明.* 对于任意的 DFA, 都可以构造与其等价的 CFG: 对每个状态 $q_i$ 构造一个变量 $R_i$, 起始变量 $R_0$ 对应起始态 $q_0$, 如果 $\delta(q_i, a) = q_j$, 就添加规则 $R_i \to aR_j$, 而如果 $q_i$ 是接受态, 就添加规则 $R_i \to \varepsilon$.

而显然存在可被 CFG 描述的非正则语言, 比如 $\{0^n 1^n | n \in \mathbb{N}\}$. □

**定义 2.3 (歧义性).** 称一个串 $w$ 由 CFG $G$ 歧义生成, 如果存在 $G$ 下 $w$ 的两种 lestmost derivation (每次只替换最左边的变量) 方式, 或者说存在两棵不同的 parse tree 可以生成 $w$. 称一个 CFG $G$ 是歧义的, 如果它可以歧义生成某些串.

**定义 2.4 (固有歧义).** 称一个 CFL $L$ 是固有歧义的, 如果 $L$ 只能由歧义的 CFG $G$ 生成.

**例 2.1.** $\{a^i b^j c^k | i = j \text{ or } j = k\}$ 是固有歧义的.

我们希望能给有限自动机做一些加强, 使其能够达到与 CFG 相同的表达能力. 最终决定了为其添加栈结构, 于是得到了如下定义的“下推自动机”:

**定义 2.5 (Pushdown Automaton, PDA).** 下推自动机是一个六元组 $(Q, \Sigma, \Gamma, \delta, q_0, F)$, 其中

- $Q$ 是状态集,
- $\Sigma$ 是输入字符集,
- $\Gamma$ 是栈字符集,
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$ 是转移函数,

- $q_0 \in Q$ 是起始态,
- $F \subseteq Q$ 是接受态集合.

其中 $\Sigma_\varepsilon, \Gamma_\varepsilon$ 分别表示 $\Sigma \cup \{\varepsilon\}, \Gamma \cup \{\varepsilon\}$. $(q', b) \in \delta(q, c, a)$ 表示在状态 $q$ 上被输入 $c$ 字符时, 会先从栈顶 pop 出字符 $a$, 再向栈顶 push 进字符 $b$, 最后转移到状态 $q'$. 幂集 $\mathcal{P}$ 暗含了下推自动机是 nondeterministic 的.

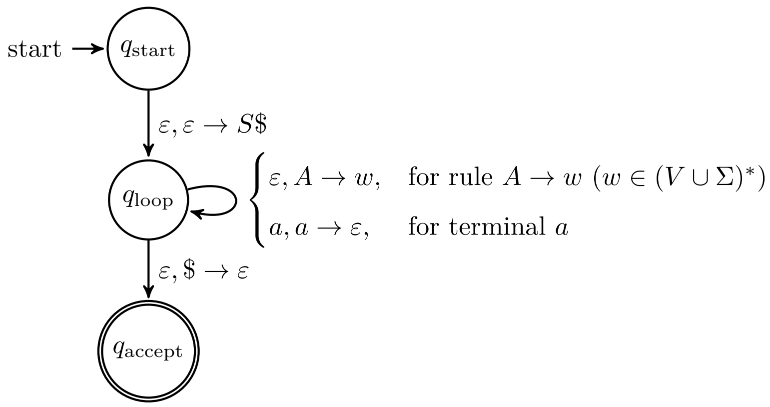
称字符串 $w = w_1 w_2 \cdots w_m (w_i \in \Sigma_\varepsilon)$ 可以被 PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ 接受, 如果存在状态序列 $r_0, r_1, \cdots, r_m \in Q$ 和字符串 (栈) 序列 $s_0, s_1, \cdots, s_m \in \Gamma^*$, 满足

- $r_0 = q_0, s_0 = \varepsilon$,
- for $i = 0, 1, \cdots, m-1$, $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at, s_{i+1} = bt$ for some $a, b \in \Gamma_\varepsilon$ and $t \in \Gamma^*$,
- $r_m \in F$.

类似的, 可以定义 $L(M)$ 表示 PDA $M$ 接受的所有字符串构成的集合, 即 $M$ 所识别的语言.

**定理 2.1** (**下推自动机与上下文无关文法的等价性**). 一个语言是上下文无关的, 当且仅当存在某个下推自动机可以识别它.

证明. "$\Rightarrow$" : 需要根据 CFG 来构造 PDA. 一开始把 CFG 的起始变量写在栈上, 并保证在替换过程中栈顶始终是一个尚未替换的变量. 利用 nondeterminism 尝试每一种变量的替换方式. 每次只考虑替换栈顶的变量, 而如果栈顶是一个终止符, 就直接和输入匹配掉. 当输入匹配完且栈为空时, 代表输入串可接受.




"$\Leftarrow$" : 需要根据 PDA 来构造 CFG. 不妨假设[1]该 PDA 有如下特性: (i)只有一个接受态 $q_{\text{accept}}$, (ii)会在接受前清空栈, (iii)每次转移都会要么 push 要么 pop, 没有 both 和 neither 的情况. 构造变量 $A_{pq}$ 表示所有能够使 PDA 从 "状态 $p$ 且栈空" 转移到 "状态 $q$ 且栈空" 的串组成的语言, 其中 $A_{q_0 q_{\text{accept}}}$ 是该 CFG 的起始变量. 按如下方式构造 CFG 的规则集合:

- 对于任意 $p, q, r, s \in Q, u \in \Gamma, a, b \in \Sigma_\varepsilon$, 如果 $(r, u) \in \delta(p, a, \varepsilon), (q, \varepsilon) \in \delta(s, b, u)$, 就添加规则 $A_{pq} \to a A_{rs} b$,
- 对于任意 $p, q, r \in Q$, 添加规则 $A_{pq} \to A_{pr} A_{rq}$,
- 对于任意 $p \in Q$, 添加规则 $A_{pp} \to \varepsilon$.

构造思路来源于考虑压栈弹栈的括号序列, 该序列要么被一个大括号包裹 (第一种), 要么由两个括号序列组成 (第二种). 可以归纳证明 $A_{pq}$ 的构造方式与其含义的等价性. $\square$

**定理 2.2** (**Pumping Lemma for CFL**). 如果 $A$ 是上下文无关语言, 那么存在一个数 $p$(称为 **pumping length**), 使得对于任意 $A$ 中长度至少为 $p$ 的字符串 $s$, $s$ 都可以分成五部分 $s = uvxyz$ 满足

[1]需要简短地说明转化的可行性. 前两条只需要添加额外的结束状态和转移函数即可, 第三条需要在所有 both 和 neither 的转移中间插入中间状态.

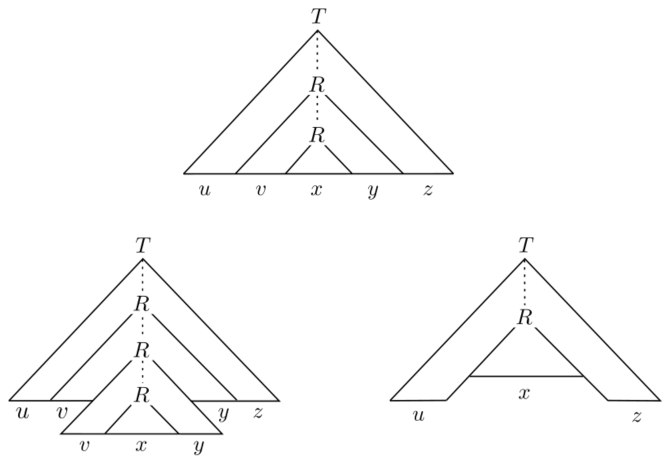
- for each $i \geqslant 0, uv^ixy^iz \in A$,
- $|vy| > 0$,
- $|vxy| \leqslant p$.

证明. 设 $b$ 为规则中的最大"度数", 即替换字符串的最大长度. 如果 parse tree 的树高是 $h$(根的深度是 0), 那么生成的字符串长度至多为 $b^h$.

取 pumping length $p$ 为 $b^{|V|+1}$. 长度至少为 $p$ 的串对应的 parse tree 树高至少为 $|V|+1$, 故存在一条"直链"上有至少 $|V|+1$ 个变量, 根据**鸽巢原理**, 存在一个变量出现至少两次, 记为 $R$, 那么对于 $R$ 就可以无限复制或者把两次出现压缩成一次 (如图).

为了满足第二个条件, 我们要求 parse tree 必须是"最简"的, 因为只有冗余的替换方式才会导致两次 $R$ 的出现之间没有任何字符实际生成.

为了满足第三个条件, 取 $R$ 为满足条件的"深度最大"的, 即两个 $R$ 都出现在最底下 $|V|+1$ 层. 此时 $|vxy|$ 对应上面的 $R$ 的子树大小, 受深度限制不超过 $b^{|V|+1} = p$.




□

**例 2.2.** $B = \{a^nb^nc^n | n \geqslant 0\}$ 不是上下文无关语言.

证明. 假设 B 是正则语言, 那么就存在 pumping length $p$. 考虑串 $a^pb^pc^p$, 注意到 $|vxy| \leqslant p$ 故不可能含有超过两种字符, 那么在重复时就不可能保证三种字符出现次数仍然相同, 从而 $B$ 不是上下文无关语言. □

**例 2.3.** $C = \{a^ib^jc^k | 0 \leqslant i \leqslant j \leqslant k\}$ 不是上下文无关语言.

证明. 考虑串 $a^pb^pb^p$, 注意 $v, y$ 分别只能包含一种字符, 否则就会出现顺序错乱. 无论分别包含什么字符, 考虑 pumping up 或者 pumping down, 总可以使得生成的新字符串不属于 $C$, 从而 $C$ 不是上下文无关语言. □

**例 2.4.** $D = \{ww | w \in \{0,1\}^*\}$ 不是上下文无关语言.

证明. 考虑 $s = 0^p1^p0^p1^p$.

首先指出 $vxy$ 必须跨域 $s$ 的中点. 假设 $vxy$ 只出现在 $s$ 的左半边, 那么 $uv^2xy^2z$ 中, 中点右侧的字符一定是 1(因为 $|vy| \leqslant p$, 只会把 $\frac{|vy|}{2} \leqslant \frac{p}{2}$ 个 1 推到右半边), 而起始字符是 0 说明该串不是 $ww$ 形式的.

但如果 $vxy$ 跨越 $s$ 的中点, 那么就一定跟前 $p$ 个 0 与后 $p$ 个 1 无交, 因此 $uxz$ 就会形如 $0^p1^i0^j1^p$, 其中 $i, j < p$, 这显然不属于 $D$. □

# 3 图灵机

图灵机是有限自动机的进一步加强 (也严格强于下推自动机), 在状态集的基础上额外添加了外部存储设备——“纸带”. 一条纸带是一列无限长的存储单元 (称为“格子”), 其中每个格子上只能存储有限的信息, 在单步计算中也只能读取/写入一个格子, 读取/写入的格子位置 (称为“纸带头”) 在相邻两步计算间也至多移动一格.

通俗地来讲, 一台拥有 $k$ 条纸带的图灵机在一步计算中, 会首先查询其状态并分别在 $k$ 个纸带头位置读取 $k$ 个字符, 然后根据这些信息, 决定变换到什么状态, 将 $k$ 个纸带头处的字符改写成什么, 以及如何移动 $k$ 个纸带头 (向左或右移动一格, 或者保持不动).

**定义 3.1 (Deterministic Turing Machine, TM).** 一台 $k$ 条纸带的 (确定性) 图灵机是一个七元组 $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$, 其中

- $Q$ 是状态集.
- $\Sigma$ 是不包含空格符 $\square$ 的输入字符集, $\Gamma$ 是纸带字符集. $\Sigma \subseteq \Gamma$.
- $\delta : Q \times \Gamma^k \to Q \times \Gamma^k \times \{\text{L(eft)}, \text{S(tay)}, \text{R(ight)}\}^k$ 是转移函数.
- $q_0, q_{\text{accept}}, q_{\text{reject}} \in Q$ 分别是起始态, 接受态和拒接态.

初始时, 第一条纸带的第一个格子上标有 $\triangleright$ 字符, 表示输入串的开始, 随后紧接着是输入串 $w \in \Sigma^*$. 除了这 $|w|+1$ 个格子外, 所有纸带的所有格子都被初始化为空格符 $\square$.

一旦图灵机运行到 $q_{\text{accept}}$ 或者 $q_{\text{reject}}$ 状态时, 它就会**停机**. 因此可以把一台图灵机看成一个 $\Sigma^* \to \{0,1\}$ 的函数. 一般地, 对于图灵机 $M$ 和字符串 $x \in \Sigma^*$, 我们有 $M(x) \in \{0,1\}$, 其中 $M(x) = 1$ 当且仅当对于输入 $x$, $M$ 会运行到 $q_{\text{accept}}$, $M(x) = 0$ 当且仅当对于输入 $x$, $M$ 会运行到 $q_{\text{reject}}$ **或者不停机**.

对于图灵机 $M$, 记 $L(M) = \{x \in \{0,1\}^* | M(x) = 1\}$ 为 $M$ 所**识别**的语言.

**注 3.1.** 图灵机还存在另一种设定: 并没有 $q_{\text{accept}}, q_{\text{reject}}$ 两个特殊状态, 取而代之的是单一的停机状态 $q_{\text{halt}}$, 同时最后一条纸带被用作“输出纸带”. 图灵机运行到 $q_{\text{halt}}$ 时停机, 此时输出纸带上的内容就是该图灵机的输出.

这种设定下的图灵机可以看作一个 $\Sigma^* \to \Sigma^*$ 的函数, 但不难验证其计算能力与原本设定下的是相同的. 之后为了省事可能会混淆使用两种设定, 不妨碍理解就好.

**定义 3.2 (图灵可识别与图灵可判定).** 称一个语言是**图灵可识别 (Turing-recognizable)** 的, 如果存在一台图灵机可以识别它 (i.e. 接受其中的每一个字符串). 称一台图灵机是一个 **decider**, 如果它对于任何输入都不会无限循环. 称一个语言是**图灵可判定 (Turing-decidable)** 的, 如果存在一台 decider 可以识别它.

**定义 3.3 (函数的计算, 运行时间).** 考虑函数 $f : \{0,1\}^* \to \{0,1\}^*$ 以及 $T : \mathbb{N} \to \mathbb{N}$, 令 $M$ 为一图灵机. 我们称 $M$ 计算了函数 $f$, 如果对于任意 $x \in \{0,1\}^*$, 只要 $M$ 被初始化为输入 $x$, 它就能在输出纸带上写下 $f(x)$ 并停机. 称 $M$ 在 $T(n)$ 的时间内计算了 $f$, 如果它计算每个 $x$ 都可以在 $T(|x|)$ 步内停机.

**定义 3.4 (Time-constructible functions).** 称一个函数 $T : \mathbb{N} \to \mathbb{N}$ 是 time-constructible 的, 如果 $T(n) \geqslant n$ 且存在运行时间为 $T(n)$ 的计算函数 $x \to \llcorner T(|x|) \lrcorner$ 的图灵机 $M$, 其中 $\llcorner x \lrcorner$ 表示 $x$ 的 binary representation.

**例 3.1.** $A = \{w\#w | w \in \{0,1\}^*\}$ 可以被 (单纸带) 图灵机识别.

*证明.* 给待匹配的两个位置打上标记, 每次前后移动找标记, 用状态记录已经看过的字符, 若比较失败则直接 reject, 否则向后移动标记继续比较直到全部比完. 以上的描述的图灵机的运行时间是 $T(n) = O(n^2)$ 的. $\square$

**例 3.2.** $B = \{ww | w \in \{0,1\}^*\}$ 可以被双纸带图灵机识别.

*证明.* 没有 # 记号, 无法方便地找到中间位置. 可以在二号纸带上把输入复制一遍, 然后二号纸带头移动到中间 (一号纸带头走一步, 二号纸带头走两步), 顺序比较即可. 运行时间是 $O(n)$. $\square$

图灵机由字符集大小, 纸带数量等的不同, 产生了许多不同的变种. 它们在计算能力上会有所不同吗?

**命题 3.1 (小字符集模拟大字符集).** 对于任意 $f:\{0,1\}^*\to\{0,1\}$ 以及 time-constructible function $T:\mathbb{N}\to\mathbb{N}$, 如果 $f$ 可以被图灵机 $M$ 以 $T(n)$ 时间计算, 那么它也可以被一台字符集为 $\{0,1,\square,\triangleright\}$ 的图灵机 $\tilde{M}$ 以 $O(T(n)\log|\Gamma|)$ 的时间计算.

*证明.* 任意 $\Gamma$ 中的字符都可以用 $\log|\Gamma|$ 个比特表示. 每步转移时先用 $\log|\Gamma|$ 步读出纸带上一个字符的 encoding 并存入状态, 再根据转移函数进行移动, 最后用 $\log|\Gamma|$ 步写下新字符的 encoding 表示. □

**命题 3.2 (单纸带模拟多纸带).** 对于任意 $f:\{0,1\}^*\to\{0,1\}$ 以及 time-constructible function $T:\mathbb{N}\to\mathbb{N}$, 如果 $f$ 可以被有 $k$ 条纸带的图灵机 $M$ 以 $T(n)$ 时间计算, 那么它也可以被一台单纸带图灵机 $\tilde{M}$ 以 $O(kT^2(n))$ 的时间计算. 单纸带指的是只有一条可读可写的纸带, 它同时扮演了输入、工作和输出纸带的角色.

*证明.* 把单条纸带上的位置按照模 $k$ 余数分配给 $k$ 条纸带, 对每种字符 $a$ 新建字符 $\hat{a}$ 表示所在纸带的纸带头指向这个字符. 注意到运行时间为 $T(n)$ 的图灵机, 对于长度为 $n$ 的输入, 最多只会用到前 $T(n)$ 个位置, 所以每步转移时花费 $O(kT(n))$ 的代价搜索每个纸带头的位置即可[2], 运行时间为 $O(kT^2(n))$. □

**注 3.2 (健忘的图灵机, oblivious Turing Machine).** 纸带头的移动只与输入长度有关, 而与输入的具体内容无关, 即对于任意 $x\in\{0,1\}^*$ 以及 $i\in\mathbb{N}$, $M$ 在输入 $x$ 并执行到第 $i$ 步时, 所有纸带头的位置是关于 $|x|$ 和 $i$ 的函数. 可以证明健忘的图灵机可以以平方的 overhead 模拟一台标准图灵机.

**命题 3.3 (单向图灵机模拟双向图灵机).** 对于任意 $f:\{0,1\}^*\to\{0,1\}$ 以及 time-constructible function $T:\mathbb{N}\to\mathbb{N}$, 如果 $f$ 可以被双向图灵机 (纸带的两个方向都有无限长)$M$ 以 $T(n)$ 时间计算, 那么它也可以被一台单向图灵机 $\tilde{M}$ 以 $O(T(n))$ 的时间计算.

**论点 3.1 (Church-Turing Thesis).** 任何物理上可实现的计算设备都可以被图灵机实现.

**定理 3.1 (通用图灵机存在).** 存在图灵机 $\mathcal{U}$ 使得对于任意 $x,\alpha\in\{0,1\}^*$, $\mathcal{U}(\langle x,\alpha\rangle)=M_\alpha(x)$, 其中 $M_\alpha$ 为被 $\alpha$ 表示的图灵机. 进一步地, 如果 $M_\alpha$ 对于 $x$ 在 $T$ 步内停机, 则 $\mathcal{U}(\langle x,\alpha\rangle)$ 可以在 $CT\log T$ 步内停机, 其中 $C$ 是一个仅依赖于 $M_\alpha$ 的字符集大小、纸袋条数、状态数的常数.

*证明.* 构造 $\mathcal{U}$ 为一台五纸带图灵机, 五条纸带分别为

- Input, 被模拟图灵机 $M$ 的输入.
- Description of $M$, 主要为了记录转移函数 $\delta$ 以便查询.
- Simulation of $M$, 记录纸带信息. 这里需要用单纸带来模拟 $M$, 因而会产生平方的 overhead.
- Current state of $M$, 这部分不能简单地存在 $\mathcal{U}$ 的状态里, 因为对于 $\mathcal{U}$ 来说, $M$ 的状态集大小并不是常数.
- Output, $M$ 的输出.

注意每步模拟的过程中, 读取 $M$ 所在状态, 读取转移函数都是关于 $n$ 常数时间的.

可以设计一种类似势能分析的算法, 把单纸带模拟多纸带的 overhead 降到 $O(n\log n)$, 从而使通用图灵机模拟的复杂度优化到 $O(T\log T)$. □

[2] 可以考虑在已使用部分的“最远处”打上标记并维护, 这样每次搜索的代价就不超过当前写入过的格子数量

# 4 不可判定语言

**定义 4.1 (可识别/判定语言, recognizable/decidable Language).** 可识别语言就是能够被一台图灵机识别的语言. 可判定语言就是能够被一台 decider 识别的语言.

**定理 4.1.** 存在不可识别语言.

证明. 只需要考虑 "语言" 与 "可识别语言" 的基数. 前者的基数是 $2^{\aleph_0}$, 后者的基数不超过图灵机的基数 (因为存在可识别语言到图灵机的单射), 而图灵机可以被有限长的字符串描述, 因此是可数的. □

**定义 4.2 (可计算函数 (Computable Function)).** 可计算函数就是可以被一台图灵机计算的函数. 特别的, 考虑函数 $f : \{0,1\}^* \to \{0,1\}$, 则 $f$ 是可计算函数当且仅当 $L = \{x \in \{0,1\}^* | f(x) = 1\}$ 是可判定语言.

**定理 4.2.** 存在不可计算函数/存在不可判定语言.

(证明中的构造是重要的. 这个结论本身相比定理 4.1 是平凡的.)

证明. 我们认为存在一个映射 $\alpha \to M_\alpha$ 可以把任意字符串映到一台图灵机 (以某种既定格式编码, 再把非法格式的串映到某台特定的图灵机即可). 考虑函数 $\mathrm{UC}(\alpha) = 1 - M_\alpha(\alpha)$, 我们指出 UC 是不可计算函数.

假设 UC 可计算, 考虑计算 UC 的图灵机 $M$. 考虑 $\mathrm{UC}(\llcorner M \lrcorner)$, 由于 $M$ 计算了 UC, 我们知道 $\mathrm{UC}(\llcorner M \lrcorner) = M(\llcorner M \lrcorner)$, 但根据 UC 的定义, 又有 $\mathrm{UC}(\llcorner M \lrcorner) = 1 - M(\llcorner M \lrcorner)$, 产生了矛盾. □

**例 4.1 (停机问题不可判定).** $\mathsf{HALT} = \{\langle \llcorner M \lrcorner, \alpha \rangle | M \text{ halts on } \alpha\}$ 是不可判定语言.

证明. 假设存在 $M_{\mathsf{HALT}}$ 可以判定 HALT.

利用 $M_{\mathsf{HALT}}$ 可以构造判定语言 $L = \{\alpha | \mathrm{UC}(\alpha) = 1\}$ 的 decider: 计算 $M_{\mathsf{HALT}}(\langle \alpha, \alpha \rangle)$, 如果得到 0 (说明 $M_\alpha$ 对 $\alpha$ 不停机) 则直接输出 1, 否则输出 $M_\alpha(\alpha)$ 的结果. 这与 UC 不可计算相矛盾. 因此 HALT 不可判定. □

**例 4.2 (接受问题不可判定).** $\mathsf{AC} = \{\langle \llcorner M \lrcorner, \alpha \rangle | M \text{ accepts } \alpha\}$ 是不可判定语言.

证明. 利用 $M_{\mathsf{AC}}$ 可以直接构造 $M_{\mathrm{UC}}$: 只要把 $M_{\mathsf{AC}}(\langle \alpha, \alpha \rangle)$ 的输出取反即可. □

**定义 4.3 (映射规约, Mapping Reduction).** 称语言 $A$ 可映射规约到 (is mapping reducible to) 语言 $B$, 如果存在可计算函数 $f : \Sigma^* \to \Sigma^*$, 使得 $w \in A$ 当且仅当 $f(w) \in B$, 记作 $A \leqslant_m B$.

**命题 4.1.** 如果 $A \leqslant_m B$, 则

- 如果 $B$ 可判定, 则 $A$ 也可判定.
- 如果 $A$ 不可判定, 则 $B$ 也不可判定.

**例 4.3.** $\mathsf{NAC} = \{\llcorner M \lrcorner | M \text{ accept nothing}\}$ 是不可判定语言.

证明. 考虑构造映射 $f$ 满足 $w \in \overline{\mathsf{AC}} \Leftrightarrow f(w) \in \mathsf{NAC}$. 令 $f(\langle \llcorner M \lrcorner, \alpha \rangle) = \llcorner M' \lrcorner$ 其中 $M'$ 不管输入直接运行 $M(\alpha)$. $f$ 显然是可计算的, 故 $\overline{\mathsf{AC}} \leqslant_m \mathsf{NAC}$, 而可判定语言关于补集的封闭性导致 $\overline{\mathsf{AC}}$ 是不可判定语言, 从而 NAC 是不可判定语言. □

**例 4.4.** $\mathsf{EQU} = \{\langle \llcorner M_1 \lrcorner, \llcorner M_2 \lrcorner \rangle | L(M_1) = L(M_2)\}$ 是不可判定语言.

证明. 考虑构造映射 $g$. 取 $g(\llcorner M \lrcorner) = \langle \llcorner M \lrcorner, \llcorner M' \lrcorner \rangle$ 其中 $M'$ 是拒绝一切输入的图灵机. 于是 $\llcorner M \lrcorner \in \mathsf{NAC} \Leftrightarrow \langle \llcorner M \lrcorner, \llcorner M' \lrcorner \rangle \in \mathsf{EQU}$, $\mathsf{NAC} \leqslant_m \mathsf{EQU}$, 从而 EQU 是不可判定语言. □

**定理 4.3.** 语言 $A$ 可判定当且仅当 $A$ 与 $\overline{A}$ 均可识别.

证明. $\Rightarrow$: 显然. $\Leftarrow$: 记 $A$ 可被 $M_1$ 识别, $\overline{A}$ 可被 $M_2$ 识别, 考虑<u>**并行**</u>运行 $M_1$ 和 $M_2$, 总有一个会给出结果. □

**推论 4.1.** HALT 和 AC 都是不可判定的可识别语言. 这说明 $\overline{\mathsf{HALT}}$ 和 $\overline{\mathsf{AC}}$ 都是不可识别语言.

## 5 时间复杂性, P 与 NP

**定义 5.1 (DTIME 与 P).** 对于函数 $T:\mathbb{N}\to\mathbb{N}$, 称语言 $L\in\mathbf{DTIME}(T(n))$, 如果存在常数 $c>0$ 和一台运行时间为 $c\cdot T(n)$ 的 decider 可以识别 $L$.

$\mathbf{P}=\bigcup_{c\geqslant 0}\mathbf{DTIME}(n^c)$.

**命题 5.1.** $\mathbf{P}$ 在多项式次集合操作下是封闭的.

**例 5.1.** $\mathsf{PATH}=\{\langle G,s,t\rangle : G$ is a direct graph in which there is a path from $s$ to $t\}\in\mathbf{P}$.

**定理 5.1 (Time Hierarchy Theorem).** $f,g$ 是满足 $f(n)\log f(n)=o(g(n))$ 的 time constructible 的函数, 则

$$\mathbf{DTIME}(f(n))\subsetneq\mathbf{DTIME}(g(n))$$

*证明.* 考虑这样的图灵机 $D$: 对于 $x\in\{0,1\}^*$, 用通用图灵机 $\mathcal{U}$ 模拟 $M_x(x)$ 运行至多 $g(|x|)$ 步 (是 $\mathcal{U}$ 的 $g(|x|)$ 步而不是 $M_x$ 的 $g(|x|)$ 步), 如果 $\mathcal{U}$ 在 $g(|x|)$ 步数内输出了 $b\in\{0,1\}$, 则 $D$ 输出 $1-b$, 否则 $D$ 输出 0.

根据定义, $D$ 对于任何输入 $x$ 都会在 $g(|x|)$ 步内停机, 因此 $L(D)\in\mathbf{DTIME}(g(n))$. 我们通过反证法证明 $L(D)\notin\mathbf{DTIME}(f(n))$. 先叙述否命题: 存在图灵机 $M$ 和常数 $c$, 使得对于任意输入 $x\in\{0,1\}^*$, $M$ 都能在 $cf(|x|)$ 步内输出与 $D$ 相同的结果.

对于输入 $x$, 用通用图灵机 $\mathcal{U}$ 模拟 $M$ 只需要 $c'cf(|x|)\log f(|x|)$ 步, 其中 $c'$ 是不依赖于 $|x|$ 的一个常数. 由于 $f(n)\log f(n)=o(g(n))$[3], 故存在充分大的 $n_0$ 使得 $g(n)>c'cf(n)\log f(n)$ 对于任意 $n\geqslant n_0$ 均成立. 令 $x'=\llcorner M\lrcorner$ 满足 $|x'|\geqslant n_0$, 那么

- $D$ 会输出与 $M$ 相同的结果, 因为这是 $M$ 的定义;
- $D$ 会输出与 $M$ 不同的结果, 因为 $c'cf(n)\log f(n)<g(n)$ 使得 $\mathcal{U}$ 对 $M$ 的模拟已经结束了, 根据 $D$ 的定义, $D$ 应该输出相反的结果.

产生了矛盾. 因此 $\mathbf{DTIME}(f(n))\subsetneq\mathbf{DTIME}(g(n))$. □

**定义 5.2 (NP).** 称语言 $L\subseteq\{0,1\}^*$ 属于 $\mathbf{NP}$, 如果存在一个多项式 $p:\mathbb{N}\to\mathbb{N}$ 和一个多项式时间图灵机 $M$(称为 $L$ 的 **verifier**) 使得对于任意的 $x\in\{0,1\}^*$, 都有

$$x\in L\Leftrightarrow\exists u\in\{0,1\}^{p(|x|)}, M(x,u)=1$$

如果 $x\in L$ 与 $u\in\{0,1\}^{p(|x|)}$ 满足 $M(x,u)=1$, 则称 $u$ 是 $x$ 的一个 **certificate**.

**命题 5.2.** 定义 $\mathbf{EXP}=\bigcup_{c\geqslant 0}\mathbf{DTIME}(2^{n^c})$, 则 $\mathbf{P}\subseteq\mathbf{NP}\subseteq\mathbf{EXP}$.

**注 5.1.** 根据 Time Hierarchy Theorem 我们知道 $\mathbf{P}\subsetneq\mathbf{EXP}$, 因此 $\mathbf{P}\subsetneq\mathbf{NP}$ 与 $\mathbf{NP}\subsetneq\mathbf{EXP}$ 至少一者为真.

**定义 5.3 (非确定图灵机与 NTIME).** 非确定图灵机 (Nondeterministic Turing Machine, NDTM) 是有两个转移函数 $\delta_0,\delta_1$ 和一个特定状态 $q_{\text{accept}}$ 的图灵机 $M$, 每步转移时, 可以任意选择遵从某一个转移函数. 对于输入 $x$, 称 $M(x)=1$ 当且仅当存在一个选择序列可以使 $M$ 到达 $q_{\text{accept}}$ 状态, 否则——任意选择序列都无法在停机前到达 $q_{\text{accept}}$——就认为 $M(x)=0$. 称 $M$ 的运行时间为 $T(n)$, 如果对于任意输入 $x\in\{0,1\}^*$ 以及任意的选择序列, $M$ 都会在 $T(|x|)$ 步内到达 $q_{\text{accept}}$ 或者 $q_{\text{halt}}$.

对于 $T:\mathbb{N}\to\mathbb{N}$ 和语言 $L\subseteq\{0,1\}^*$, 称 $L\in\mathbf{NTIME}(T(n))$, 如果存在常数 $c>0$ 和一个运行时间为 $c\cdot T(n)$ 的非确定图灵机 $M$, 满足对于任意的 $x\in\{0,1\}^*$, $x\in L\Leftrightarrow M(x)=1$.

**定理 5.2.** $\mathbf{NP}=\bigcup_{c\geqslant 0}\mathbf{NTIME}(n^c)$.

*证明.* 非确定图灵机的选择序列可以看作 $x$ 的一个 certificate, 反之亦然. □

[3] little-$o$ 不能替换成 big-$O$, 我只能说懂的都懂.

**定义 5.4 (多项式时间规约, NP-hard 与 NP-complete).** 称语言 $L \subseteq \{0,1\}^*$ 可**多项式时间 (Karp) 规约**到语言 $L' \subseteq \{0,1\}^*$(记作 $L \leqslant_p L'$), 如果存在一个多项式时间可计算函数 $f : \{0,1\}^* \to \{0,1\}^*$ 使得对于任意 $x \in \{0,1\}^*$, $x \in L \Leftrightarrow f(x) \in L'$.

称 $L' \in \mathbf{NP}$-hard, 如果对于任意 $L \in \mathbf{NP}$, $L \leqslant_p L'$. $\mathbf{NP}$-complete $= \mathbf{NP} \cap \mathbf{NP}$-hard.

**定理 5.3 ($\leqslant_p$ 的传递性).**
- 若 $L \leqslant_p L'$ 且 $L' \leqslant_p L''$, 则 $L \leqslant_p L''$.
- 如果 $L \in \mathbf{NP}$-hard, 则 $L \in \mathbf{P} \Rightarrow \mathbf{P} = \mathbf{NP}$.
- 如果 $L \in \mathbf{NP}$-complete, 则 $L \in \mathbf{P} \Leftrightarrow \mathbf{P} = \mathbf{NP}$.

**定理 5.4 (Cook-Levin Theorem).** $\mathsf{SAT}, \mathsf{3SAT} \in \mathbf{NP}$-complete.

其中 $\mathsf{SAT} = \{\varphi \in \text{CNF} | \varphi \text{ is satisfiable}\}, \mathsf{3SAT} = \{\varphi \in \text{3CNF} | \varphi \text{ is satisfiable}\}$. CNF(Conjunctive Normal Form, 合取范式) 是一种形如 $\bigwedge_i \left(\bigvee_j v_{i_j}\right)$ 的特殊的 boolean formula, 其中每个 $v_{i_j}$ 都是某个变量或者其否定. 3CNF 是每个 $\bigvee_j v_{i_j}$ (称为**从句 (clause)**) 中都只含有不超过 3 项的 CNF.

*证明.* $\mathsf{SAT}, \mathsf{3SAT} \in \mathbf{NP}$ 是平凡的. 先考虑证明 $\mathsf{SAT} \in \mathbf{NP}$-hard, 即 $\forall L \in \mathbf{NP}, L \leqslant_p \mathsf{SAT}$. 回顾两者定义

$$
\begin{aligned}
x \in L \quad &\Leftrightarrow \quad \exists \text{ certificate } u \quad \text{s.t. } M(x,u) = 1 \\
\varphi_x \in \mathsf{SAT} \quad &\Leftrightarrow \quad \exists \text{ assignment } u \quad \text{s.t. } \varphi_x(u) = \text{True}
\end{aligned}
$$

因此考虑根据 $M, x$ 来构造**多项式规模的** $\varphi_x$, 满足 $M(x,u) = 1 \Leftrightarrow \varphi_x(u) = \text{True}$.

不妨假设 $M$ 是 (i) 双纸带且第一条纸带只读 (ii) oblivious 的 (参见注 3.2), 那么在 $M$ 运行到第 $i$ 步时, 其两个纸带头所指向的字符, 当前状态所组成的三元组 $z_i \in \Gamma \times \Gamma \times Q$ (称为 $M$ 运行到第 $i$ 步时的 **snapshot**) 将由后三者唯一决定: $z_{i-1}, z_{\text{prev}(i)}$ 和 $(x \circ u)_{\text{inputpos}(i)}$, 其中 $\text{prev}(i)$ 表示上一次工作纸带纸带头位置与第 $i$ 步相同的时刻, $\text{inputpos}(i)$ 表示第 $i$ 步时输入纸带的纸带头位置.

根据 $M$ 的转移函数 $\delta$, 可以构造出 $F : \{0,1\}^{2c+1} \to \{0,1\}^c$ 满足 $z_i = F(z_{i-1}, z_{\text{prev}(i)}, (x \circ u)_{\text{inputpos}(i)})$, 其中 $c$ 是表示一个 snapshot 所需的比特数.

可以构造一个有 $n + p(n) + cT(n)$ 个变量的 CNF $\varphi_x$, 其中 $p, T$ 是 $M$ 的参数 (分别是 verifier 长度与运行时间关于 $n$ 的函数). 把这些变量分别记为 $y$ 以及 $z_1, \cdots, z_{T(n)}$, 它是下述这些条件的 AND:

1. $y$ 的前 $n$ 位和 $x$ 相同.
2. $z_1$ 表示初始状态 $(\triangleright, \square, q_{\text{start}})$.
3. $z_i = F(z_{i-1}, z_{\text{prev}(i)}, y_{\text{inputpos}(i)})$ $(\forall i \in \{2, \cdots, T(n)\})$.
4. $z_{T(n)}$ 表示到达 $q_{\text{halt}}$ 的终止状态.

这些条件用 CNF 表示出来, 其长度是 $d(n + T(n))$, 其中 $d$ 是与 $n$ 无关的常数. 从而我们实现了多项式时间规约, 完成了 $L \leqslant_p \mathsf{SAT}$ 的证明.

然后考虑证明 $\mathsf{SAT} \leqslant_p \mathsf{3SAT}$. 只需要注意到

$$
\bigvee_{i=1}^{k} u_i \cong \left(z \vee \bigvee_{i=1}^{k-2} u_i\right) \wedge (\overline{z} \vee u_{k-1} \vee u_k)
$$

说明任意 CNF 都可以多项式时间规约到 3CNF. 所以 $\mathsf{SAT} \leqslant_p \mathsf{3SAT}$. $\square$

**例 5.2 (独立集).** $\mathsf{INDSET} = \{\langle G, k\rangle | G \text{ has an independent set of size } k\} \in \mathbf{NP}$-complete.

*证明.* 显然 $\mathsf{INDSET} \in \mathbf{NP}$. 考虑通过 $\mathsf{3SAT} \leqslant_p \mathsf{INDSET}$ 来证明 $\mathsf{INDSET} \in \mathbf{NP}$-hard.

对于一个有 $m$ 个 clause 的 3CNF $\varphi$, 对每个有 $l$ 项的 clause 建不超过 $2^l$ 个点, 表示能使这个 clause 为 True 的部分变量赋值 (互不相同). 在所有会产生冲突的点对间连边, 得到一张不超过 $7m$ 个点的图. 不难验证这张图存在大小为 $m$ 的独立集当且仅当 $\varphi$ 可满足. $\square$

**例 5.3 (点覆盖).** Vertex-Cover $= \{\langle G, k\rangle | G \text{ has a subset of } k \text{ vertices that covers all edges}\} \in \mathbf{NP}$-complete.

*证明.* 独立集很容易规约到点覆盖, 懂的都懂. □

**例 5.4 (01 整数规划).** 给出一列有理系数线性不等式, 判断是否存在一组 $\{0,1\}$ 赋值满足所有不等式. 把可满足的不等式组的集合记作 01IPROG. 01IPROG $\in \mathbf{NP}$-complete.

*证明.* 考虑证明 SAT $\leqslant_p$ 01IPROG. 只需要对每个 clause 构造一个不等式即可.

$$u_1 \vee \cdots \vee u_n \vee \overline{u_{n+1}} \vee \cdots \vee \overline{u_{n+m}} \Rightarrow \sum_{i=1}^{n} u_i + m - \sum_{i=1}^{m} u_{n+i} \geqslant 1$$

□

**注 5.2.** 如果把变量取值限制从 $\{0,1\}$ 改为全体有理数, 问题就变成了线性规划, 从而 $\in \mathbf{P}$.

**例 5.5.** DSAT $= \{\varphi \in \text{DNF} | \varphi \text{ is satisfiable}\} \in \mathbf{P}$. 只要线性地检查一下有没有某个 clause 都满足就行了.

**例 5.6 (有向图哈密顿路径).** dHAMPATH $= \{\llcorner G \lrcorner | G \text{ is directed and has a Hamiltonian path}\} \in \mathbf{NP}$-complete.

*证明.* 考虑证明 SAT $\leqslant_p$ dHAMPATH. 对于一个有 $n$ 个变量和 $m$ 个 clause 的 CNF $\varphi$:

- 对每个变量建 $2m+1$ 个点, 第 $i$ 个变量对应的点记作 $a_{i,1}, \cdots, a_{i,2m+1}$. 对每个 clause 建一个点, 第 $i$ 个 clause 对应的点记作 $b_i$. 再建立额外源汇 $v_{\text{start}}, v_{\text{end}}$, 总共是 $(2m+1)n+m+2$ 个点.
- 对于每个 $i \in [1,n]$, $a_{i,1}, \cdots, a_{i,2m+1}$ 首尾相接连成一个 $2m+1$ 大小的**双向**环.
- $v_{\text{start}}$ 向 $\{a_{1,1}, a_{1,2m+1}\}$ 连边, $\{a_{i,1}, a_{i,2m+1}\}$ 向 $\{a_{i+1,1}, a_{i+1,2m+1}\}$ 连边, $\{a_{n,1}, a_{n,2m+1}\}$ 向 $v_{\text{end}}$ 连边, 均为单向.
- 如果 $u_i$ 出现在了第 $j$ 个从句中, 连边 $a_{i,j} \to b_j \to a_{i,j+1}$. 如果 $\overline{u_i}$ 出现在了第 $j$ 个从句中, 连边 $a_{i,j+1} \to b_j \to a_{i,j}$.

对于每个双向环, 它可以也应该被顺时针或者逆时针地单向遍历, 即要么首先访问 $a_{i,1}$ 然后从小到大走到 $a_{i,2m+1}$, 要么首先访问 $a_{i,2m+1}$ 然后从大到小走到 $a_{i,1}$. 顺/逆时针的遍历方向对应着相应变量的 0/1 取值, 当取值符合期望时, 遍历该变量对应的环时可以顺带覆盖掉一些 $b_j$.

可以证明 $\varphi \in$ SAT 当且仅当上述构造出的图存在 $v_{\text{start}}$ 到 $v_{\text{end}}$ 的哈密顿路径. □

**例 5.7 (有向图哈密顿回路).** dHAMCYCLE $= \{\llcorner G \lrcorner | G \text{ is directed and has a Hamiltonian cycle}\} \in \mathbf{NP}$-complete.

*证明.* 考虑证明 dHAMPATH $\leqslant_p$ dHAMCYCLE. 对于图 $G$, 添加额外源汇 $s, t$, $s$ 向所有点连边, 所有点向 $t$ 连边, $t$ 向 $s$ 连边, 得到图 $G'$. $G$ 存在哈密顿路径当且仅当 $G'$ 存在哈密顿回路. □

**例 5.8 (无向图哈密顿路径).** uHAMPATH $= \{\llcorner G \lrcorner | G \text{ is undirected and has a Hamiltonian path}\} \in \mathbf{NP}$-complete.

*证明.* 考虑证明 dHAMPATH $\leqslant_p$ uHAMPATH.

对于有向图 $G$, 添加额外源汇 $s, t$(要连边) 后把每个点拆成三个点: 入点, 中间点, 出点, 适当连边得到无向图 $G'$. $G$ 存在哈密顿路径 $\Rightarrow$ $G'$ 存在哈密顿路径是显然的, 而如果 $G'$ 存在哈密顿路径, 则任意一个入点/出点都会与其中间点相邻, 因为否则会导致中间点不得不成为路径的起止点, 而显然 $s$ 的入点与 $t$ 的出点是必须作为起止点的 (度数是 1). 于是 $G'$ 的哈密顿路径不会破坏 $G$ 的有向图结构, 从而可以构造出 $G$ 的哈密顿路径. □

**例 5.9 (无向图哈密顿回路).** uHAMCYCLE $= \{\llcorner G \lrcorner | G \text{ is undirected and has a Hamiltonian cycle}\} \in \mathbf{NP}$-complete.

*证明.* 考虑证明 dHAMCYCLE $\leqslant_p$ uHAMCYCLE.

对于有向图 $G$, 直接把每个点拆成三个点: 入点, 中间点, 出点, 适当连边得到无向图 $G'$. $G$ 存在哈密顿回路 $\Rightarrow$ $G'$ 存在哈密顿回路是显然的, 而如果 $G'$ 存在哈密顿回路, 则这条回路必然是 "入-中间-出-入-中间-出" 循环的, 因此也可以构造出 $G$ 的哈密顿回路. □

**例 5.10 (旅行商问题).** $\mathsf{TSP} = \{\langle G, d_{ij}, k\rangle | G \text{ has a Hamiltonian cycle with distance measured by } d_{ij} \text{ at most } k\} \in$ **NP**-complete.

证明. 考虑证明 dHAMCYCLE $\leqslant_p$ TSP. 根据 $G$ 到底有没有这条边把边权设为 0 或者 1, $k$ 取 0 就行. □

**定义 5.5 (coNP).** $\mathbf{coNP} = \{L | \overline{L} \in \mathbf{NP}\}$.

**例 5.11.** $\overline{\mathsf{SAT}} \in \mathbf{coNP}$, 但在证明 $\overline{\mathsf{SAT}} \in \mathbf{NP}$ 时遇到了困难: 难以高效地验证 $\varphi$ 对所有的 assignment 都为 False.

**定义 5.6 (coNP 的另一种定义).** 称语言 $L \subseteq \{0,1\}^*$ 属于 **coNP**, 如果存在一个多项式 $p: \mathbb{N} \to \mathbb{N}$ 和一个多项式时间图灵机 $M$ 使得对于任意的 $x \in \{0,1\}^*$, 都有

$$x \in L \Leftrightarrow \forall u \in \{0,1\}^{p(|x|)}, M(x,u) = 0$$

**定义 5.7 (coNP-hard, coNP-complete).** 称 $L' \in$ **coNP**-hard, 如果任意 $L \in \mathbf{coNP}, L \leqslant_p L'$.

**coNP**-complete $=$ **coNP** $\cap$ **coNP**-hard.

**例 5.12.** $\overline{\mathsf{SAT}} \in$ **coNP**-complete. 这是因为 $\forall L \in \mathbf{coNP}$, 我们有 $\overline{L} \in \mathbf{NP}$. 由 Cook-Levin Thm. 知 $\overline{L} \leqslant_p \mathsf{SAT}$, 于是便有 $L \leqslant_p \overline{\mathsf{SAT}}$. ($A \leqslant_p B \Leftrightarrow \overline{A} \leqslant_p \overline{B}$.)

**命题 5.3.** 如果 $\mathbf{P} = \mathbf{NP}$, 则 $\mathbf{NP} = \mathbf{coNP} = \mathbf{P}$. 反过来说, 只要证明了 $\mathbf{NP} \neq \mathbf{coNP}$, 就能说明 $\mathbf{P} \neq \mathbf{NP}$.

**定义 5.8 (NEXP).** $\mathbf{NEXP} = \bigcup_{c \geqslant 0} \mathbf{NTIME}(2^{n^c})$.

**定理 5.5.** 如果 $\mathbf{EXP} \neq \mathbf{NEXP}$, 则 $\mathbf{P} \neq \mathbf{NP}$.

证明. 考虑证明逆否命题: 如果 $\mathbf{P} = \mathbf{NP}$, 则 $\mathbf{EXP} = \mathbf{NEXP}$. 使用一种叫做**填充 (padding)** 的技术: 考虑 $L \in \mathbf{NTIME}(2^{n^c})$, 构造 $L_{\text{pad}} = \{\langle x, 1^{2^{|x|^c}}\rangle | x \in L\}$, 则可验证 $L_{\text{pad}} \in \mathbf{NP}$(因为问题几乎没变, 而输入规模变大了). 而如果 $L_{\text{pad}} \in \mathbf{P}$, 则也可以验证 $L \in \mathbf{EXP}$. $\mathbf{EXP} \subseteq \mathbf{NEXP}$ 是显然的, 故证明了 $\mathbf{EXP} = \mathbf{NEXP}$. □

迄今为止, 我们研究的所有问题都是 Decision Problems, 即输出信息量为一比特的问题. 如果考虑把设定延伸到 Search Problems, 例如对给定的 CNF $\varphi$ 求出一组 satisfying assignment, 这种问题会不会与 Decision Problems 有所不同呢?

**定理 5.6.** 假如 $\mathbf{P} = \mathbf{NP}$, 则对于任意 $L \in \mathbf{NP}$ 和它的 verifier $M$, 都存在多项式时间图灵机 $B$, 能够对输入 $x \in L$ 输出 $x$ 的一个 certificate.

证明. 首先证明 $L = \mathsf{SAT}$ 时是正确的. 只需要依次对每个变量做 0/1 代入, 并调用 $M$ 判断一下当前已进行的代入下是否仍存在解就行了.

对于任意的 $L \in \mathbf{NP}$, 由 Cook-Levin Thm. 我们知道 $L \leqslant_p \mathsf{SAT}$. 事实上这种规约有一个更好的性质: 不仅 $x \in L \Leftrightarrow f(x) \in \mathsf{SAT}$, 而且可以把 $f(x)$ 的一个 satisfying assignment 映射到 $x$ 的一个 certificate. 这样的规约被称为 **Levin 规约 (Levin reduction)**. □

**定理 5.7 (Ladners Theorem, "NP-intermediate").** 假设 $\mathbf{P} \neq \mathbf{NP}$, 则存在既非 **P** 亦非 **NP**-complete 的 **NP** 语言.

证明. 对于函数 $H: \mathbb{N} \to \mathbb{N}$, 定义语言 $\mathsf{SAT}_H = \left\{\psi 01^{n^{H(n)}} | \psi \in \mathsf{SAT}, |\psi| = n\right\}$.

用如下奇怪的方式构造 $H$, $H(n)$ 的值为: 最小的 $i$, 满足对于任意长度不超过 $\log n$ 的输入 $x$, $M_i$(以 $i$ 作为 binary representation 得到的图灵机) 都可以在 $i|x|^i$ 步内输出 $\mathsf{SAT}_H(x)$, 然后把这个 $i$ 对 $\log\log n$ 取 min.

*可以通过给出一个计算 $H$ 的具体算法来说明 $H$ 是良定的: 想要计算 $H(n)$, 我们需要 **(1)** 在不超过 $\log\log n$ 台图灵机上运行 $2^{\log n} = n$ 个输入 $x$ 至多 $\log\log n(\log n)^{\log\log n}$ 步, 以得到所有 $M_i$ 的输出, **(2)** 对每个 $i \leqslant \log n$ 计算 $H(i)$, 以及检查所有长度不超过 $\log n$ 的输入 $x$ 是否 $\in SAT$, 以得到所有 $SAT_H(x)$ 的正确值. 因此有 $T(n) \leqslant \log n T(\log n) + O(n^2)$, 得到了一个时间复杂度为 $T(n) = O(n^2)$ 的算法.*

从定义不难看出 $H(n)$ 是关于 $n$ 单调的, 即要么 $H(n) = O(1)$, 要么 $\lim\limits_{n\to\infty} H(n) = \infty$. 我们证明如下引理.

**引理 5.1.** $\mathsf{SAT}_H \in \mathbf{P}$ 当且仅当 $H(n) = O(1)$.

证明. $\Rightarrow$: 假设存在一台图灵机 $M$ 可以在 $cn^c$ 步内对任意长度为 $n$ 的输入 $x$ 计算 $\mathsf{SAT}_H(x)$. 由于 $M$ 可以被无限多个串表示, 故总存在 $i > c$ 使 $M_i = M$, 此时根据定义, 对于任意的 $n > 2^{2^i}$ $(\log\log n > i)$ 都有 $H(n) \leqslant i$, 因此 $H(n) = O(1)$.

$\Leftarrow$: $H(n) = O(1)$ 说明 $H$ 的取值数量有限, 从而总会存在某个 $i$ 使 $H(n) = i$ 对无限多个 $n$ 成立. 这就说明了 $M_i$ 可以在 $in^i$ 时间内计算 $\mathsf{SAT}_H$ (否则如果 $M_i$ 在输入 $x \in \{0,1\}^*$ 上寄了, 那么对于任意 $n > 2^{|x|}$ 都应该有 $H(n) \neq i$) 注意到这里的 $i$ 是常数, 从而说明了 $\mathsf{SAT}_H \in \mathbf{P}$. $\square$

根据以上引理, 我们利用归谬证明如果 $\mathbf{P} \neq \mathbf{NP}$, 则 $\mathsf{SAT}_H$ 既不属于 $\mathbf{P}$, 也不属于 $\mathbf{NP}$-complete.

- 假如 $\mathsf{SAT}_H \in \mathbf{P}$, 那么根据引理 $H(n) = O(1)$, 这说明 $\mathsf{SAT}_H$ 只是 $\mathsf{SAT}$ “填充”了多项式个数个 1 得到的, 从而 $\mathsf{SAT} \leqslant_p \mathsf{SAT}_H$, 导致了 $\mathsf{SAT} \in \mathbf{P}$, 与 $\mathbf{P} \neq \mathbf{NP}$ 矛盾.
- 假如 $\mathsf{SAT}_H \in \mathbf{NP}$-complete, 则存在一种运行时间为 $O(n^i)$ 的从 $\mathsf{SAT}$ 到 $\mathsf{SAT}_H$ 的规约 $f$. 注意到由上一条归谬我们已经知道了 $\lim\limits_{n\to\infty} H(n) = \infty$, 故存在 $N$ 使 $H(n) > 3i$ 对 $\forall n > N$ 成立. 考虑构造一种多项式时间算法来实现 $\mathsf{SAT}$ 的判定:
  - 对于输入 $\varphi$, 若 $|\varphi| < N$ 则暴力判定, 否则 $O(|\varphi|^i)$ 地计算 $f(\varphi)$ 并检查其是否形如 $\psi 0 1^{|\psi|^{H(\psi)}}$, 若不形如则直接返回 False, 否则有 $\varphi \in \mathsf{SAT} \Leftrightarrow f(\varphi) \in \mathsf{SAT}_H \Leftrightarrow \psi \in \mathsf{SAT}$, 可以递归判定 $\psi$.

  考虑限制一下 $|\psi|$. 注意到 $|\psi 0 1^{|\psi|^{H(\psi)}}| = |\psi| + 1 + |\psi|^{H(\psi)} \leqslant |\varphi| + |\varphi|^i$, 随意缩放一下有 $|\psi|^{3i} < |\psi|^{H(\psi)} \leqslant |\varphi|^{i+1}$, 从而 $|\psi| \leqslant \sqrt{|\varphi|}$. 那么算法的时间复杂度就是 $T(n) \leqslant O(n^i) + T(\sqrt{n}) \Rightarrow T(n) = O(n^i)$. 这也导致 $\mathsf{SAT} \in \mathbf{P}$, 与 $\mathbf{P} \neq \mathbf{NP}$ 矛盾.

$\square$

**定理 5.8 (Nondeterministic Time Hierarchy Theorem).** $f, g$ 是满足 $f(n+1) = o(g(n))$ 的 time constructible 的函数, 则

$$\mathbf{NTIME}(f(n)) \subsetneq \mathbf{NTIME}(g(n))$$

证明. 摆烂了不证了. $\square$

# 6 空间复杂性

**定义 6.1 (运行空间, SPACE 与 NSPACE).** 对于 $S:\mathbb{N}\to\mathbb{N}$ 和 $L\subseteq\{0,1\}^*$, 称 $L\in\mathbf{SPACE}(S(n))$, 如果存在常数 $c$ 以及可以决定 $L$ 的图灵机 $M$, 满足在对任意长度为 $n$ 的输入的计算中, $M$ 只会访问到至多 $c\cdot S(n)$ 个 work tapes 上 (不包含 input) 的位置, 称 $M$ 的运行空间为 $O(S(n))$.

类似地可以定义 **NSPACE**, 这里要求在任何一种决策下用到的位置数量都不超过 $c\cdot S(n)$.

**定义 6.2 (Space constructible functions).** 称一个函数 $S:\mathbb{N}\to\mathbb{N}$ 是 space-constructible 的, 如果 $S(n)\geqslant\log n$ 且存在运行空间为 $S(n)$ 的计算函数 $x\to\llcorner S(x)\lrcorner$ 的图灵机.

**注 6.1.** 相比于 time constructible functions, 我们不要求 space constructible functions 满足 $S(n)\geqslant n$, 但为了能够"记住在输入纸带上的位置", 我们一般会要求 $S(n)\geqslant\log n$.

**定理 6.1.** 对于任何 space-constructible 的函数 $S:\mathbb{N}\to\mathbb{N}$, 有

$$\mathbf{DTIME}(S(n))\subseteq\mathbf{SPACE}(S(n))\subseteq\mathbf{NSPACE}(S(n))\subseteq\mathbf{DTIME}(2^{O(S(n))})$$

*证明.* 前两个 $\subseteq$ 都是平凡的, 只考虑证明最后一个.

我们称一台 (确定或非确定) 图灵机 $M$ 的一个 **configuration** 包含 (i) work tape 上的所有非空字符; (ii) 所有纸带的 head 位置; (iii) $M$ 所处的状态, 则对于确定的输入 $x\in\{0,1\}^*$, 一个 configuration 的后继 configuration 是 (a) 对于图灵机来说, 唯一确定的; (b) 对于非确定图灵机来说, 至多唯二确定的. 把 configuration 之间的转移看成一张有向图, 记作 $G_{M,x}$. 不失一般性假设 $M$ 只有一种 configuration $C_{\text{accept}}$ 满足"输出 1 后停机" (可以让图灵机在停机前擦除所有中间记录), 这样 $M(x)=1$ 就等价于 $G_{M,x}$ 中存在一条 $C_{\text{start}}$ 到 $C_{\text{accept}}$ 的路径.

陈述两个事实:

- 给定 $M,x$, $G_{M,x}$ 中的每个节点用 $O(S(n))$ 个比特来表示, 也即, $G_{M,x}$ 只有 $2^{O(S(n))}$ 个节点.
- 对于任意两个 configuration $C,C'$, 存在 $O(S(n))$ 大小的 CNF $\varphi_{M,x}$ 满足 $\varphi_{M,x}(C,C')=1$ 当且仅当 $G_{M,x}$ 中 $C$ 有边连向 $C'$.

因此用 $2^{O(S(n))}$ 的时间把整张 $G_{M,x}$ 建出来, 再 BFS 一下即可验证 $C_{\text{start}}$ 到 $C_{\text{accept}}$ 是否连通. $\square$

**定义 6.3 (PSPACE, NPSPACE, L 与 NL).**

$$\mathbf{PSPACE}=\bigcup_{c\geqslant 1}\mathbf{SPACE}(n^c)$$
$$\mathbf{NPSPACE}=\bigcup_{c\geqslant 1}\mathbf{NSPACE}(n^c)$$
$$\mathbf{L}=\mathbf{SPACE}(\log n)$$
$$\mathbf{NL}=\mathbf{NSPACE}(\log n)$$

**推论 6.1.** $\mathbf{NP}\subseteq\mathbf{PSPACE}$, 因为都可以暴力枚举答案, 用多项式空间存下来然后验证.

**推论 6.2.** 在定理 6.1 中分别代入 $S(n)=\log n, S(n)=n^c$, 可以得到

$$\mathbf{L}\subseteq\mathbf{NL}\subseteq\mathbf{P}\subseteq\mathbf{NP}\subseteq\mathbf{PSPACE}\subseteq\mathbf{NSPACE}\subseteq\mathbf{EXP}$$

**例 6.1.**

$$\mathsf{PATH}=\{\langle G,s,t\rangle : G \text{ is a direct graph in which there is a path from } s \text{ to } t\}$$

即判断图中两点之间是否存在一条路径. 显然 $\mathsf{PATH}\in\mathbf{NL}$, 但其是否属于 $\mathbf{L}$ 仍是一个 open problem. 而事实上, $\mathsf{PATH}\in\mathbf{L}$ 等价于 $\mathbf{L}=\mathbf{NL}$, 即 $\mathsf{PATH}$ 是 $\mathbf{NL}$-complete.

**定理 6.2 (Space Hierarchy Theorem).** $f, g$ 是满足 $f(n) = o(g(n))$ 的 space constructible 的函数, 则

$$\mathbf{SPACE}(f(n)) \subsetneq \mathbf{SPACE}(g(n))$$

证明. 技术细节在于通用图灵机 $\mathcal{U}$ 模拟图灵机 $M$ 只需要常数倍的空间, 所以相比于 Time Hierarchy Theorem 没有了对数项. 其余部分跟 Time Hierarchy Theorem 的证明类似, 就不再赘述了. □

**定义 6.4 (PSPACE-hard, PSPACE-complete).** 称 $L'$ 是 **PSPACE**-hard, 如果对于任意 $L \in \mathbf{PSPACE}$, $L \leqslant_p L'$. **PSPACE**-complete = **PSPACE** ∩ **PSPACE**-hard.

**例 6.2.**

$$\mathsf{SPACE\ TMSAT} = \{\langle M, w, 1^n \rangle : \text{DTM } M \text{ accepts } w \text{ in space } n\}$$

这是一个 **PSPACE**-complete 语言.

**定义 6.5 (Quantified Boolean formula, QBF).** 一个 QBF 是形如 $Q_1x_1Q_2x_2 \cdots Q_nx_n\varphi(x_1, x_2, \cdots, x_n)$ 的公式, 其中 $Q_i \in \{\forall, \exists\}$, $x_i$ 的取值是 $\{0, 1\}$, $\varphi$ 是一个 plain(unquantified) boolean formula .

上述定义专注于讨论**前束范式**的 QBF , 因为非前束范式都可以转化成等价的前束范式. 一个 QBF 有真值 True 或 False .

用 $\mathsf{TQBF}$ 表示所有为真的 QBF 的集合.

**定理 6.3.** $\mathsf{TQBF} \in \mathbf{PSPACE}$-complete.

证明. 先证明 $\mathsf{TQBF} \in \mathbf{PSPACE}$. 这个是简单的, 因为判定可以通过 DFS 实现, 而 DFS 只需要 $O(n+m)$ 的空间, 其中 $n$ 是变量数, $m$ 是 QBF 的长度.

再证明任意 $L \in \mathbf{PSPACE}$ 都满足 $L \leqslant_p \mathsf{TQBF}$. 假设 $M$ 是在 $S(n)$ 空间内计算 $L$ 的图灵机, 对于输入 $x \in \{0, 1\}^*$, 考虑 configuration graph $G_{M,x}$, 我们陈述过图中每个点可以用 $m = O(S(n))$ 个比特来表示, 以及存在一个 CNF $\varphi_{M,x}$ 满足 $\varphi_{M,x}(C, C') = \text{True}$ 当且仅当 $G_{M,x}$ 中有 $C \to C'$ 的边.

考虑根据 $\varphi_{M,x}$ 来构造我们想要的 QBF $\psi$ . 用 $\psi_i$ 表示一个 QBF , $\psi_i(C, C') = \text{True}$ 当且仅当 $G_{M,x}$ 中存在一条长度不超过 $2^i$ 从 $C$ 到 $C'$ 的路径, 那么显然 $\psi = \psi_m(C_{\text{start}}, C_{\text{accept}}), \psi_0(C, C') = \varphi_{M,x}(C, C') \vee (C = C')$. $\psi_i$ 可以递归定义: 对于 $i \geqslant 1$, $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$.

一个技术细节是需要改进递归定义的具体方式以保证 $\psi$ 的长度是多项式级别的. 可以用一种看上去有点奇怪, 但与前述定义等价的形式:

$$\psi_i(C, C') = \exists C'' \forall D_1 \forall D_2 ((D_1, D_2) = (C, C'') \wedge (D_1, D_2) = (C'', C')) \Rightarrow \psi_{i-1}(D_1, D_2)$$

这样构造出的 QBF $\psi$ 的长度是 $O(m^2) = O(S^2(n))$ 的, 从而 $\leqslant_p$ 成立. □

**注 6.2.** 上述证明只利用了 configuration graph 的性质, 而这个性质是无关 determinism 的. 因此我们可以类似证明 $\mathsf{TQBF} \in \mathbf{NSPACE}$-complete, 从而说明 $\mathbf{PSPACE} = \mathbf{NSPACE}$.