

# 计算理论导论 课程笔记

酥雨

zusuyu@stu.pku.edu.cn

April 25, 2022

## 目录

<b>1 正则语言</b>	<b>2</b>
1.1 有限自动机	2
1.2 正则表达式	3
1.3 Pumping Lemma	4
<b>2 上下文无关语言</b>	<b>5</b>
2.1 上下文无关文法	5
2.2 下推自动机	5
2.3 Pumping Lemma	7
<b>3 图灵机</b>	<b>8</b>
<b>4 不可判定语言</b>	<b>10</b>
<b>5 时间复杂性, P 与 NP</b>	<b>11</b>
5.1 P versus NP	11
5.2 NP 完全性	12
5.3 常见 NP-complete 问题的规约	13
5.4 coNP, NP-intermediate 以及更多	14
<b>6 空间复杂性</b>	<b>16</b>
6.1 空间受限的计算	16
6.2 PSPACE 完全性	17
6.3 NL, coNL 与 NL 完全性	18
<b>7 Polynomial Hierarchy</b>	<b>20</b>
<b>8 布尔线路</b>	<b>21</b>
8.1 线路可满足性	21
8.2 线路一致生成	21
8.3 线路规模	22
8.4 接受建议的图灵机	23
8.5 Karp-Lipton Theorem	23
8.6 线路深度	23

# 1 正则语言

## 1.1 有限自动机

**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.
- $\delta: Q \times \Sigma \rightarrow Q$  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  $w = w_1w_2 \cdots w_m (w_i \in \Sigma)$  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:**  $A^* = \{x_1x_2 \cdots x_k | k \geq 0 \text{ and each } x_i \in A\}$ .

**注 1.1.** 补集  $\bar{A} = \Sigma^* - A$  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  $Q \times \Sigma \rightarrow Q$  的函数, 而是  $Q \times \Sigma \rightarrow \mathcal{P}(Q)$  的, 其中  $\mathcal{P}$  表示幂集,  $\Sigma_\epsilon$  表示  $\Sigma \cup \{\epsilon\}$ .

相应的, 称字符串  $w = w_1w_2 \cdots w_m (w_i \in \Sigma)$  可以被 NFA  $N = (Q, \Sigma, \delta, q_0, F)$  接受, 如果  $w$  可以写成  $w = y_1y_2 \cdots y_{m'} (y_i \in \Sigma_\epsilon)$ , 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  $\epsilon$  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

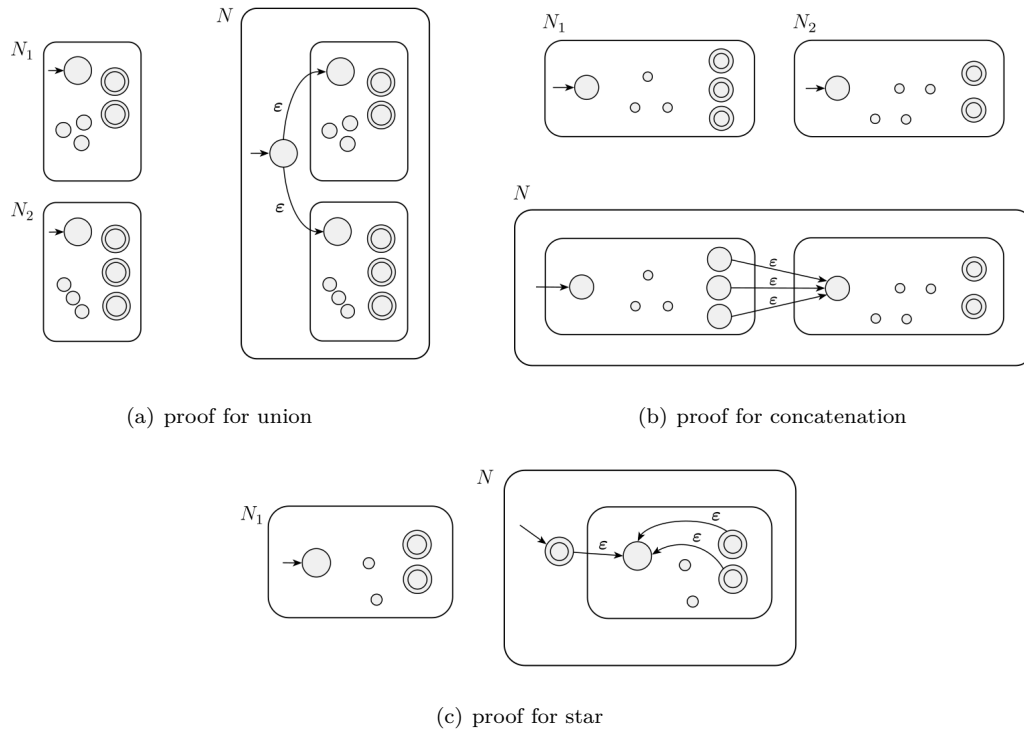
- $Q' = \mathcal{P}(Q)$ .
- $\forall R \in Q', a \in \Sigma, \delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$ .
- $q'_0 = \{q_0\}$ .
- $F' = \{R \in Q' | R \cap F \neq \emptyset\}$ .

□

**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.



□

## 1.2 正则表达式

**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
- $\{\epsilon\}$ , 其中  $\epsilon$  表示空串
- $\emptyset$
- $(R_1 \cup R_2)$ , 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  $R$ ,  $R \cup \emptyset = R \circ \epsilon = R$ ,  $R \circ \emptyset = \emptyset$ ,  $\emptyset^* = \{\epsilon\}$ .

**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  $k \geq 2$  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  $(Q \setminus \{q_{\text{accept}}\}) \times (Q \setminus \{q_{\text{start}}\}) \rightarrow \mathcal{R}$  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  $q_{\text{start}} \neq q_{\text{accept}}$ 。

### 1.3 Pumping Lemma

**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  $i \geq 0, xy^iz \in A$ ,
- $|y| > 0$ ,
- $|xy| \leq p$ .

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  $|xy| \leq p$  有时也是有用的。

**例 1.2.**  $B = \{0^n 1^n | n \geq 0\}$  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  $0^p 1^p$ , 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  $0^p 1^p$ , 注意到我们要求了  $|xy| \leq p$ , 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  $0^p 10^p 1$ , 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.**  $D = \{1^{n^2} | n \geq 0\}$  不是正则语言。

证明. 考虑串  $1^{p^2}$ . 由于  $|y| \leq p$ , 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.**  $E = \{0^i 1^j | i > j\}$  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 上下文无关语言

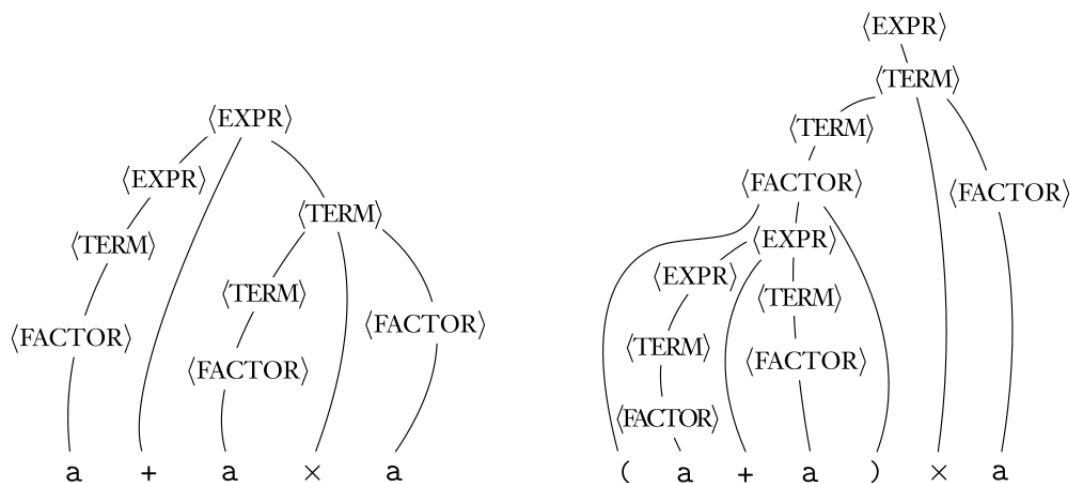
### 2.1 上下文无关文法

**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  $\{w \in \Sigma^* | S \xRightarrow{*} w\}$ .

**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  $R_i \rightarrow aR_j$ , 而如果  $q_i$  是接受态, 就添加规则  $R_i \rightarrow \varepsilon$ .

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 leftmost 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.

### 2.2 下推自动机

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,
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \rightarrow \mathcal{P}(Q \times \Gamma_\epsilon)$  是转移函数,
- $q_0 \in Q$  是起始态,
- $F \subseteq Q$  是接受态集合.

其中  $\Sigma_\epsilon, \Gamma_\epsilon$  分别表示  $\Sigma \cup \{\epsilon\}, \Gamma \cup \{\epsilon\}$ .  $(q', b) \in \delta(q, c, a)$  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.

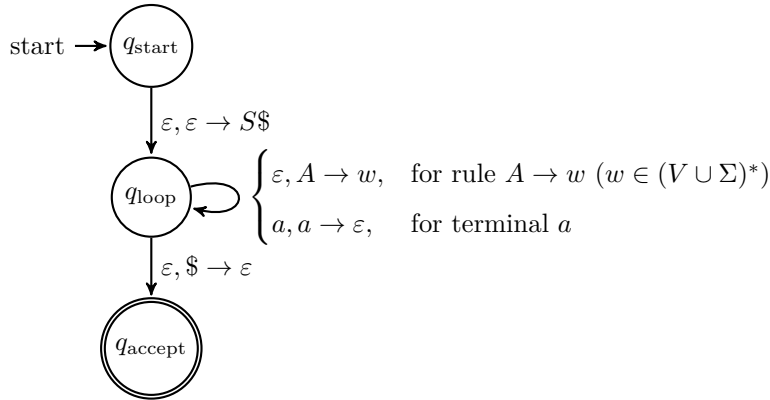
称字符串  $w = w_1 w_2 \cdots w_m (w_i \in \Sigma_\epsilon)$  可以被 PDA  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  接受, 如果存在状态序列  $r_0, r_1, \dots, r_m \in Q$  和字符串 (栈) 序列  $s_0, s_1, \dots, s_m \in \Gamma^*$ , 满足

- $r_0 = q_0, s_0 = \epsilon$ ,
- for  $i = 0, 1, \dots, m-1, (r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ , where  $s_i = at, s_{i+1} = bt$  for some  $a, b \in \Gamma_\epsilon$  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<sup>1</sup>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  $p, q, r, s \in Q, u \in \Gamma, a, b \in \Sigma_\epsilon$ , 如果  $(r, u) \in \delta(p, a, \epsilon), (q, \epsilon) \in \delta(s, b, u)$ , 就添加规则  $A_{pq} \rightarrow a A_{rs} b$ ,
- 对于任意  $p, q, r \in Q$ , 添加规则  $A_{pq} \rightarrow A_{pr} A_{rq}$ ,
- 对于任意  $p \in Q$ , 添加规则  $A_{pp} \rightarrow \epsilon$ .

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.  $\square$

<sup>1</sup>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.

## 2.3 Pumping Lemma

**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

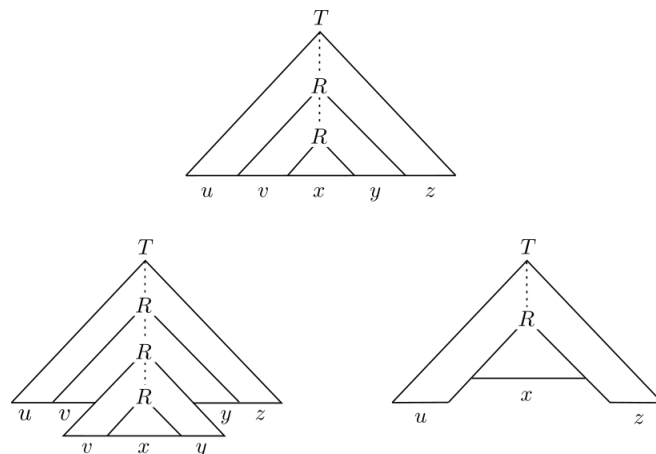
- for each  $i \geq 0, uv^i xy^i z \in A$ ,
- $|vy| > 0$ ,
- $|vxy| \leq p$ .

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.**  $B = \{a^n b^n c^n | n \geq 0\}$  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  $a^p b^p c^p$ , 注意到  $|vxy| \leq p$  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.**  $C = \{a^i b^j c^k | 0 \leq i \leq j \leq k\}$  不是上下文无关语言.

证明. 考虑串  $a^p b^p b^p$ , 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  $s = 0^p 1^p 0^p 1^p$ .

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  $uv^2 xy^2 z$  中, 中点右侧的字符一定是 1 (因为  $|vy| \leq p$ , 只会把  $\frac{|vy|}{2} \leq \frac{p}{2}$  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  $0^p 1^i 0^j 1^p$ , 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.  $\Sigma \subseteq \Gamma$ .
- $\delta: Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{\text{L(ef)}, \text{S(tay)}, \text{R(ight)}\}^k$  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  $\Sigma^* \rightarrow \{0, 1\}$  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  $L(M) = \{x \in \{0, 1\}^* \mid M(x) = 1\}$  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  $\Sigma^* \rightarrow \Sigma^*$  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  $f: \{0, 1\}^* \rightarrow \{0, 1\}^*$  以及  $T: \mathbb{N} \rightarrow \mathbb{N}$ , 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  $T: \mathbb{N} \rightarrow \mathbb{N}$  是 time-constructible 的, 如果  $T(n) \geq n$  且存在运行时间为  $T(n)$  的计算函数  $x \mapsto \lfloor T(|x|) \rfloor$  的图灵机  $M$ , 其中  $\lfloor x \rfloor$  表示  $x$  的 binary representation.

**例 3.1.**  $A = \{w\#w \mid w \in \{0, 1\}^*\}$  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.**  $B = \{ww \mid w \in \{0, 1\}^*\}$  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  $f : \{0, 1\}^* \rightarrow \{0, 1\}$  以及 time-constructible function  $T : \mathbb{N} \rightarrow \mathbb{N}$ , 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.  $\square$

**命题 3.2 (单纸带模拟多纸带).** 对于任意  $f : \{0, 1\}^* \rightarrow \{0, 1\}$  以及 time-constructible function  $T : \mathbb{N} \rightarrow \mathbb{N}$ , 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<sup>2</sup>, 运行时间为  $O(kT^2(n))$ .  $\square$

**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  $f : \{0, 1\}^* \rightarrow \{0, 1\}$  以及 time-constructible function  $T : \mathbb{N} \rightarrow \mathbb{N}$ , 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、纸带条数、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  $O(T \log T)$ .  $\square$

<sup>2</sup>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.  $\square$

**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  $f: \{0, 1\}^* \rightarrow \{0, 1\}$ , 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. 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  $\alpha \rightarrow M_\alpha$  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  $UC(\alpha) = 1 - M_\alpha(\alpha)$ , 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  $UC(\ulcorner M \urcorner)$ , 由于  $M$  计算了  $UC$ , 我们知道  $UC(\ulcorner M \urcorner) = M(\ulcorner M \urcorner)$ , 但根据  $UC$  的定义, 又有  $UC(\ulcorner M \urcorner) = 1 - M(\ulcorner M \urcorner)$ , 产生了矛盾.  $\square$

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

证明. 假设存在  $M_{HALT}$  可以判定  $HALT$ .

利用  $M_{HALT}$  可以构造判定语言  $L = \{\alpha | UC(\alpha) = 1\}$  的 decider: 计算  $M_{HALT}(\langle \alpha, \alpha \rangle)$ , 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.  $\square$

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

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

**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  $f: \Sigma^* \rightarrow \Sigma^*$ , 使得  $w \in A$  当且仅当  $f(w) \in B$ , 记作  $A \leq_m B$ .

**命题 4.1.** 如果  $A \leq_m B$ , 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.**  $NAC = \{\ulcorner M \urcorner | M \text{ accept nothing}\}$  是不可判定语言.

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

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

证明. 考虑构造映射  $g$ . 取  $g(\ulcorner M \urcorner) = \langle \ulcorner M \urcorner, \ulcorner M' \urcorner \rangle$  其中  $M'$  是拒绝一切输入的图灵机. 于是  $\ulcorner M \urcorner \in NAC \Leftrightarrow \langle \ulcorner M \urcorner, \ulcorner M' \urcorner \rangle \in EQU$ ,  $NAC \leq_m EQU$ , 从而  $EQU$  是不可判定语言.  $\square$

**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, 考虑并行运行  $M_1$  和  $M_2$ , 总有一个会给出结果.  $\square$

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

## 5 时间复杂性, P 与 NP

### 5.1 P versus NP

**定义 5.1 (DTIME 与 P).** 对于函数  $T : \mathbb{N} \rightarrow \mathbb{N}$ , 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  $L$ . 定义  $\mathbf{P} = \bigcup_{c \geq 0} \mathbf{DTIME}(n^c)$ .

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

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

**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  $f(n) \log f(n) = o(g(n))$ , 故存在充分大的  $n_0$  使得  $g(n) > c'cf(n) \log f(n)$  对于任意  $n \geq n_0$  均成立. 令  $x' = \lfloor M \rfloor$  满足  $|x'| \geq n_0$ , 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  $p : \mathbb{N} \rightarrow \mathbb{N}$  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  $\mathbf{EXP} = \bigcup_{c \geq 0} \mathbf{DTIME}(2^{n^c})$ , 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  $T : \mathbb{N} \rightarrow \mathbb{N}$  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.**  $\mathbf{NP} = \bigcup_{c \geq 0} \mathbf{NTIME}(n^c)$ .

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. □

## 5.2 NP 完全性

**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  $L \leq_p L'$ ), 如果存在一个多项式时间可计算函数  $f : \{0,1\}^* \rightarrow \{0,1\}^*$  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  $L \in \mathbf{NP}$ ,  $L \leq_p L'$ .  $\mathbf{NP-complete} = \mathbf{NP} \cap \mathbf{NP-hard}$ .

**定理 5.3 ( $\leq_p$  的传递性).** • 若  $L \leq_p L'$  且  $L' \leq_p L''$ , 则  $L \leq_p L''$ .

• 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).** SAT, 3SAT  $\in \mathbf{NP-complete}$ .

其中  $\text{SAT} = \{\varphi \in \text{CNF} \mid \varphi \text{ is satisfiable}\}$ ,  $\text{3SAT} = \{\varphi \in \text{3CNF} \mid \varphi \text{ is satisfiable}\}$ . 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. SAT, 3SAT  $\in \mathbf{NP}$  是平凡的. 先考虑证明 SAT  $\in \mathbf{NP-hard}$ , 即  $\forall L \in \mathbf{NP}, L \leq_p \text{SAT}$ . 回顾两者定义

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

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  $F : \{0,1\}^{2c+1} \rightarrow \{0,1\}^c$  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  $z_1, \dots, z_{T(n)}$ , 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.  $z_i = F(z_{i-1}, z_{\text{prev}(i)}, y_{\text{inputpos}(i)}) \ (\forall i \in \{2, \dots, T(n)\})$ .
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  $L \leq_p \text{SAT}$  的证明.

然后考虑证明 SAT  $\leq_p$  3SAT. 只需要注意到

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

说明任意 CNF 都可以多项式时间规约到 3CNF. 所以 SAT  $\leq_p$  3SAT. □

### 5.3 常见 NP-complete 问题的规约

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

证明. 显然  $\text{INDSET} \in \mathbf{NP}$ . 考虑通过  $3\text{SAT} \leq_p \text{INDSET}$  来证明  $\text{INDSET} \in \mathbf{NP-hard}$ .

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

证明. 独立集很容易规约到点覆盖, 懂的都懂.  $\square$

**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  $01\text{IPROG}$ .  $01\text{IPROG} \in \mathbf{NP-complete}$ .

证明. 考虑证明  $\text{SAT} \leq_p 01\text{IPROG}$ . 只需要对每个 clause 构造一个不等式即可.

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

$\square$

**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.**  $\text{DSAT} = \{\varphi \in \text{DNF} \mid \varphi \text{ is satisfiable}\} \in \mathbf{P}$ . 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).**  $\text{dHAMPATH} = \{\langle G \rangle \mid G \text{ is directed and has a Hamiltonian path}\} \in \mathbf{NP-complete}$ .

证明. 考虑证明  $\text{SAT} \leq_p \text{dHAMPATH}$ . 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  $a_{i,1}, \dots, a_{i,2m+1}$ . 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  $i \in [1, n]$ ,  $a_{i,1}, \dots, a_{i,2m+1}$  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  $a_{i,j} \rightarrow b_j \rightarrow a_{i,j+1}$ . 如果  $\overline{u_i}$  出现在了第  $j$  个从句中, 连边  $a_{i,j+1} \rightarrow b_j \rightarrow a_{i,j}$ .

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  $\varphi \in \text{SAT}$  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.  $\square$

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

证明. 考虑证明  $\text{dHAMPATH} \leq_p \text{dHAMCYCLE}$ . 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.  $\square$

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



证明. 考虑证明  $\text{dHAMPATH} \leq_p \text{uHAMPATH}$ .

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.  $\square$

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

证明. 考虑证明  $\text{dHAMCYCLE} \leq_p \text{uHAMCYCLE}$ .

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.  $\square$

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

证明. 考虑证明  $\text{dHAMCYCLE} \leq_p \text{TSP}$ . 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.  $\square$

## 5.4 coNP, NP-intermediate 以及更多

**定义 5.5 (coNP).**  $\text{coNP} = \{L \mid \bar{L} \in \text{NP}\}$ .

**例 5.11.**  $\overline{\text{SAT}} \in \text{coNP}$ , 但在证明  $\overline{\text{SAT}} \in \text{NP}$  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  $\text{coNP}$ , 如果存在一个多项式  $p: \mathbb{N} \rightarrow \mathbb{N}$  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  $L \in \text{coNP-hard}$ , 如果任意  $L' \in \text{coNP}$  都满足  $L' \leq_p L$ . 定义  $\text{coNP-complete} = \text{coNP} \cap \text{coNP-hard}$ .

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

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

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

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

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

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題. 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  $\text{P} = \text{NP}$ , 则对于任意  $L \in \text{NP}$  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  $L = \text{SAT}$  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  $L \in \text{NP}$ , 由 Cook-Levin Thm. 我们知道  $L \leq_p \text{SAT}$ . 事实上这种规约有更好的性质: 不仅  $x \in L \Leftrightarrow f(x) \in \text{SAT}$ , 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)**.  $\square$

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

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

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  $\text{SAT}_H(x)$ , 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  $i \leq \log n$  计算  $H(i)$ , 以及检查所有长度不超过  $\log n$  的输入  $x$  是否  $\in \text{SAT}$ , 以得到所有  $\text{SAT}_H(x)$  的正确值. 因此有  $T(n) \leq \log n T(\log n) + O(n^2)$ , 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  $\lim_{n \rightarrow \infty} H(n) = \infty$ . 我们证明如下引理.

**引理 5.1.**  $\text{SAT}_H \in \text{P}$  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  $\text{SAT}_H(x)$ . 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  $H(n) \leq i$ , 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  $\text{SAT}_H$  (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  $\text{SAT}_H \in \text{P}$ .  $\square$

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

- 假如  $\text{SAT}_H \in \text{P}$ , 那么根据引理  $H(n) = O(1)$ , 这说明  $\text{SAT}_H$  只是  $\text{SAT}$  “填充”了多项式个数个 1 得到的, 从而  $\text{SAT} \leq_p \text{SAT}_H$ , 导致了  $\text{SAT} \in \text{P}$ , 与  $\text{P} \neq \text{NP}$  矛盾.
- 假如  $\text{SAT}_H \in \text{NP-complete}$ , 则存在一种运行时间为  $O(n^i)$  的从  $\text{SAT}$  到  $\text{SAT}_H$  的规约  $f$ . 注意到由上一条归谬我们已经知道了  $\lim_{n \rightarrow \infty} H(n) = \infty$ , 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  $\text{SAT}$  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  $\psi 01^{H(\psi)}$ , 若不形如则直接返回 False, 否则有  $\varphi \in \text{SAT} \Leftrightarrow f(\varphi) \in \text{SAT}_H \Leftrightarrow \psi \in \text{SAT}$ , 可以递归判定  $\psi$ .

考虑限制一下  $|\psi|$ . 注意到  $|\psi 01^{H(\psi)}| = |\psi| + 1 + |\psi|^{H(\psi)} \leq |\varphi| + |\varphi|^i$ , 随意缩放一下有  $|\psi|^{3i} < |\psi|^{H(\psi)} \leq |\varphi|^{i+1}$ , 从而  $|\psi| \leq \sqrt[i+1]{|\varphi|}$ . 那么算法的时间复杂度就是  $T(n) \leq O(n^i) + T(\sqrt[n]{n}) \Rightarrow T(n) = O(n^i)$ . 这也导致  $\text{SAT} \in \text{P}$ , 与  $\text{P} \neq \text{NP}$  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

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

证明. 摆烂了不证了.  $\square$



## 6 空间复杂性

### 6.1 空间受限的计算

**定义 6.1 (运行空间, SPACE 与 NSPACE).** 对于  $S : \mathbb{N} \rightarrow \mathbb{N}$  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**, 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  $S : \mathbb{N} \rightarrow \mathbb{N}$  是 space-constructible 的, 如果  $S(n) \geq \log n$  且存在运行空间为  $S(n)$  的计算函数  $x \rightarrow \lfloor S(x) \rfloor$  的图灵机.

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

**定理 6.1.** 对于任何 space-constructible 的函数  $S : \mathbb{N} \rightarrow \mathbb{N}$ , 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, NSPACE, L 与 NL).**

$$\begin{aligned} \mathbf{PSPACE} &= \bigcup_{c \geq 1} \mathbf{SPACE}(n^c) \\ \mathbf{NSPACE} &= \bigcup_{c \geq 1} \mathbf{NSPACE}(n^c) \\ \mathbf{L} &= \mathbf{SPACE}(\log n) \\ \mathbf{NL} &= \mathbf{NSPACE}(\log n) \end{aligned}$$

**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.**

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

即判断图中两点之间是否存在一条路径. 显然  $\mathbf{PATH} \in \mathbf{NL}$ , 但其是否属于  $\mathbf{L}$  仍是一个 open problem.

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

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

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.  $\square$

## 6.2 PSPACE 完全性

**定义 6.4 (PSPACE-hard, PSPACE-complete).** 称  $L \in \text{PSPACE-hard}$ , 如果对于任意  $L' \in \text{PSPACE}$ ,  $L' \leq_p L$ .  $\text{PSPACE-complete} = \text{PSPACE} \cap \text{PSPACE-hard}$ .

**例 6.2.**

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

这是一个  $\text{PSPACE-complete}$  语言.

**定义 6.5 (Quantified Boolean formula, QBF).** 一个 QBF 是形如  $Q_1x_1Q_2x_2\cdots Q_nx_n\varphi(x_1, x_2, \dots, x_n)$  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 TQBF 表示所有为真的 QBF 的集合.

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

证明. 先证明  $\text{TQBF} \in \text{PSPACE}$ . 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  $L \in \text{PSPACE}$  都满足  $L \leq_p \text{TQBF}$ . 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  $C \rightarrow C'$  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  $i \geq 1$ ,  $\psi_i(C, C') = \exists C'' \psi_{i-1}(C, C'') \wedge \psi_{i-1}(C'', C')$ .

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  $\leq_p$  成立.  $\square$

**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  $\text{TQBF} \in \text{NPSPACE-complete}$ , 从而说明  $\text{PSPACE} = \text{NPSPACE}$ .

**定理 6.4 (Savitch's Theorem).** 对于任意 space constructible 的函数  $S: \mathbb{N} \rightarrow \mathbb{N}$  满足  $S(n) \geq \log n$ , 有

$$\text{NSPACE}(S(n)) \subseteq \text{SPACE}(S(n)^2)$$

证明. 这个证明类似于定理 6.3 的证明. 对于  $L \in \text{NSPACE}(S(n))$ , 考虑识别其的图灵机  $M$  与输入  $x \in \{0, 1\}^*$ , 则 configuration graph  $G_{M,x}$  有至多  $2^{m=O(S(n))}$  个节点. 设计递归算法  $R(u, v, i)$  判断是否存在  $G_{M,x}$  中从  $u$  到  $v$  长度不超过  $2^m$  的路径, 其通过枚举中间节点  $w$  递归到  $R(u, w, i-1)$  与  $R(w, v, i-1)$ , 则  $R(*, *, m)$  的空间复杂度是  $s(m) = s(m-1) + O(m) = O(S(n)^2)$ , 因此  $x \in L$  可以由一台使用  $O(S(n)^2)$  空间的 DTM  $M'$  来计算, 从而  $L \in \text{SPACE}(S(n)^2)$ .  $\square$

### 6.3 NL, coNL 与 NL 完全性

**定义 6.6 (NL 的另一种定义).** 称语言  $L \in \text{NL}$ , 如果存在一台 DTM  $M$  和多项式  $p: \mathbb{N} \rightarrow \mathbb{N}$ , 满足对于任意  $x \in \{0,1\}^*$ ,  $x \in L \Leftrightarrow \exists u \in \{0,1\}^{p(|x|)}$ , s.t.  $M(x,u) = 1$ , 其中  $u$  被放在  $M$  一条只能读一次的纸带上, 且  $M$  只能使用到  $O(\log |x|)$  个工作纸带空间.

**注 6.3.** 这种定义与  $\text{NL} = \text{NSPACE}(\log n)$  的定义是等价的, 因为多项式长度的“只能读一次”的 certificate 可以对应于 NDTM 的选择序列.

**定理 6.5 (Immerman-Szelepcsényi Theorem).**  $\overline{\text{PATH}} \in \text{NL}$ .

证明. 按照 NL 的另一种定义方式, 只需要给出一种  $O(\log n)$  空间验证算法  $A$  (后称 verifier), 对任意一张  $n$  个节点的图  $G$  以及节点  $s, t$ , 都存在一个多项式规模的 certificate  $u$  满足  $A(\langle G, s, t \rangle, u) = 1$  当且仅当  $G$  中不存在从  $s$  到  $t$  的路径.

对于  $i \in [n]$ , 记  $C_i$  表示在  $G$  中从  $s$  出发在  $i$  步之内能到达的点的集合. 很容易证明  $v \in C_i$  是可验证的: 只需要给出序列  $v_0 v_1 \cdots v_k$ , 这个序列显然是关于  $n$  多项式规模的, 同时 verifier 只需要进行如下验证: (1)  $v_0 = s$ , (2)  $G$  中有一条边从  $v_{j-1}$  到  $v_j$ , (3)  $v_k = v$ , (4)  $k \leq i$ .

紧接着我们陈述可以进行如下三种验证, 从而实现验证  $t \notin C_n$ , 即不存在从  $s$  到  $t$  的路径.

- 给定  $|C_i|$ , 可以验证  $v \notin C_i$ . 只需要按照**严格升序**给出  $C_i$  所有元素的 certificate, 此时 verifier 验证 (1) 每个 certificate 是合法, (2) 相邻两个 certificate 所对应节点编号是严格增加的, (3) certificate 共有  $|C_i|$  个, (4)  $v$  不在其中.
- 给定  $|C_{i-1}|$ , 可以验证  $v \notin C_i$ . 类似地, 只需要升序给出  $C_{i-1}$  所有元素的 certificate, 并验证没有任何一个能一步走到  $v$  即可.
- 给定  $|C_{i-1}|$ , 可以验证  $|C_i| = c$ . 只需要按照升序给出每个点  $v$  的  $v \in C_i$  或者  $v \notin C_i$  的 certificate 即可.

□

**定义 6.7 (coNL).**  $\text{coNL} = \{L | \overline{L} \in \text{NL}\}$ .

**定义 6.8 (对数空间规约, NL-complete).** 称函数  $f: \{0,1\}^* \rightarrow \{0,1\}^*$  是 隐对数空间可计算的 (implicitly logspace computable), 如果  $f$  的输出长度是多项式的 (存在  $c$  使得  $|f(x)| \leq |x|^c$  对任意  $x \in \{0,1\}^*$  成立), 且语言  $L_f = \{\langle x, i \rangle | \text{the } i\text{-th bit of } f(x) \text{ is } 1\}$ ,  $L'_f = \{\langle x, i \rangle | i \leq |f(x)|\}$  均属于  $\mathbf{L}$ .

称语言  $L$  可**对数空间规约**到语言  $L'$  (记作  $L \leq_l L'$ ), 如果存在一个隐对数空间可计算函数  $f$  使得对于任意  $x \in \{0,1\}^*$ ,  $x \in L \Leftrightarrow f(x) \in L'$ .

称语言  $L \in \text{NL-complete}$ , 如果  $L \in \text{NL}$  且  $L' \leq_l L$  对于任意  $L' \in \text{NL}$  成立.

**引理 6.1.** • 如果  $B \leq_l C, C \leq_l D$ , 则  $B \leq_l D$ .

- 如果  $B \leq_l C, C \in \mathbf{L}$ , 则  $B \in \mathbf{L}$ .

证明. 需要证明隐对数空间可计算函数在复合运算下封闭. 如果这个性质成立, 则引理的第一条自然成立, 第二条则考虑  $C \in \mathbf{L}$  等价于存在隐对数空间可计算函数  $g$  满足  $g(x) = [x \in C]$ , 故也成立.

假设  $M_f, M_g$  分别对隐对数空间可计算函数  $f, g$  计算  $x, i \rightarrow f(x)_i$  和  $y, j \rightarrow g(y)_j$ , 考虑对  $h = f \circ g$  构造  $M_h$  计算  $x, k \rightarrow g(f(x))_k$ . 大体上,  $M_h$  只需要仿照  $M_g$  的计算, 并在每次需要查询输入纸带内容时, 临时调用  $M_f$  计算  $f(x)$  的某一位即可.

□

**定理 6.6.**  $\text{PATH} \in \text{NL-complete}$ .

证明. 考虑任意  $L \in \mathbf{NL}$ , 记  $M$  是在  $O(\log n)$  空间内判定  $L$  的图灵机. 构造隐对数空间可计算函数  $f$  满足对于输入  $x \in \{0,1\}^*$ ,  $f(x) = \langle G_{M,x}, C_{\text{start}}, C_{\text{accept}} \rangle$ . 由于  $G_{M,x}$  的节点数量是  $2^{O(\log |x|)}$ , 故  $|f(x)|$  是关于  $|x|$  多项式的. 显然  $G_{M,x}$  的邻接矩阵的每一位都是对数空间可计算的, 因此整个  $f$  是隐对数空间可计算的, 又有  $x \in L \Leftrightarrow f(x) \in \text{PATH}$ , 从而说明  $L \leq_l \text{PATH}$ .  $\square$

**注 6.4.** 结合  $\text{PATH} \in \mathbf{NL}$ -complete 以及  $\overline{\text{PATH}} \in \mathbf{NL}$ , 两者说明了  $\mathbf{NL} = \mathbf{coNL}$ . 这是因为:

- $\forall L \in \mathbf{NL}, L \leq_l \text{PATH} \Rightarrow \overline{L} \leq_l \overline{\text{PATH}} \Rightarrow \overline{L} \in \mathbf{NL} \Rightarrow L \in \mathbf{coNL}$ , 说明  $\mathbf{NL} \subseteq \mathbf{coNL}$ .
- $\forall L \in \mathbf{coNL}, \overline{L} \in \mathbf{NL} \Rightarrow \overline{L} \leq_l \text{PATH} \Rightarrow L \leq_l \overline{\text{PATH}} \Rightarrow L \in \mathbf{NL}$ , 说明  $\mathbf{coNL} \subseteq \mathbf{NL}$ .

这样的结论也可以一般化, 即对于任意 space constructible 的函数  $S : \mathbb{N} \rightarrow \mathbb{N}$  满足  $S(n) \geq \log n$ , 都有  $\mathbf{NSPACE}(S(n)) = \mathbf{coNSPACE}(S(n))$ .

## 7 Polynomial Hierarchy

我们知道  $\text{INDSET} = \{\langle G, k \rangle \mid G \text{ has an independent set of size } k\} \in \mathbf{NP}$ .

考虑  $\text{EXACT-INDSET} = \{\langle G, k \rangle \mid \text{the largest independent set of } G \text{ is of size } k\}$ , 这个语言还属于  $\mathbf{NP}$  吗?

我们可以写出  $\langle G, k \rangle \in \text{EXACT-INDSET}$  的等价条件: (i) 存在大小为  $k$  的独立集, 且 (ii) 任意独立集大小不超过  $k$ . 证明  $\text{EXACT-INDSET} \in \mathbf{NP}$  的难点在于无法简单地对 (ii) 给出多项式规模的 certificate.

类似的, 考虑  $\text{MIN-EQ-CNF} = \{\langle \phi, k \rangle \mid \exists \text{ CNF formula } \psi \text{ of size } \leq k \text{ s.t. } \psi = \phi\}$ , 此时  $\langle \phi, k \rangle \in \text{MIN-EQ-CNF}$  的等价条件为: 存在 CNF formula  $\psi$  满足  $|\psi| \leq k$ , 使得对于任意赋值  $x$ , 都有  $\psi(x) = \phi(x)$ .

上述的两种语言在定义中都包含了  $\exists$  与  $\forall$  两种量词, 我们尝试对这样的语言构建出一种新的语言类.

**定义 7.1** ( $\Sigma_2^P$ ). 称语言  $L \in \Sigma_2^P$ , 如果存在多项式时间图灵机  $M$  以及多项式  $q$ , 满足对于任意  $x \in \{0, 1\}^*$  有

$$x \in L \Leftrightarrow \exists u \in \{0, 1\}^{q(|x|)}, \forall v \in \{0, 1\}^{q(|x|)}, \text{ s.t. } M(x, u, v) = 1$$

**注 7.1.** 显然  $\text{EXACT-INDSET}, \text{MIN-EQ-CNF} \in \Sigma_2^P$ , 此外也有  $\mathbf{NP} \subseteq \Sigma_2^P$  (忽略  $v$ ) 以及  $\mathbf{coNP} \subseteq \Sigma_2^P$  (忽略  $u$ ).

**定义 7.2** (**Polynomial Hierarchy**). 对于  $i \geq 1$ , 称语言  $L \in \Sigma_i^P$ , 如果存在多项式时间图灵机  $M$  以及多项式  $q$ , 满足对于任意  $x \in \{0, 1\}^*$  有

$$x \in L \Leftrightarrow \exists u_1 \in \{0, 1\}^{q(|x|)}, \forall u_2 \in \{0, 1\}^{q(|x|)}, \dots, Q_i u_i \in \{0, 1\}^{q(|x|)}, \text{ s.t. } M(x, u_1, u_2, \dots, u_i) = 1$$

其中  $Q_i = \begin{cases} \exists, & i \text{ is odd} \\ \forall, & i \text{ is even} \end{cases}$ . 定义 polynomial hierarchy  $\mathbf{PH} = \bigcup_{i \geq 1} \Sigma_i^P$ .

对每个  $i \geq 1$ , 定义  $\Pi_i^P = \mathbf{co}\Sigma_i^P = \{L \mid \bar{L} \in \Sigma_i^P\}$ , 则有  $\Sigma_i^P \subseteq \Pi_{i+1}^P \subseteq \Sigma_{i+2}^P$ , 从而  $\mathbf{PH} = \bigcup_{i \geq 1} \Pi_i^P$  也成立.

我们猜测  $\mathbf{P} \neq \mathbf{NP}, \mathbf{NP} \neq \mathbf{coNP}$ , 而放到 polynomial hierarchy 中, 类似的猜想是  $\Sigma_i^P \neq \Sigma_{i+1}^P, \Sigma_i^P \neq \Pi_i^P$ .

**定理 7.1.** 对于  $i \geq 1$ , 如果  $\Sigma_i^P = \Pi_i^P$ , 则  $\mathbf{PH} = \Sigma_i^P$ , 即  $\mathbf{PH}$  坍塌到了第  $i$  级. 特别的, 如果  $\mathbf{P} = \mathbf{NP}$ , 则  $\mathbf{PH}$  会坍塌到  $\mathbf{P}$ .

证明. 如果  $\Sigma_i^P = \Pi_i^P$ , 考虑证明  $\Sigma_j^P = \Pi_j^P = \Sigma_i^P$  对所有  $j \geq i$  成立. 考虑对  $j$  归纳.

已知  $\Sigma_j^P = \Pi_j^P = \Sigma_i^P$ , 任取语言  $L \in \Sigma_j^P$ , 则根据定义, 存在多项式时间图灵机  $M$  满足对于任意  $x \in \{0, 1\}^*$  有

$$x \in L \Leftrightarrow \exists u_1, \forall u_2, \dots, Q_{j+1} u_{j+1}, \text{ s.t. } M(x, u_1, u_2, \dots, u_{j+1}) = 1$$

定义语言  $L'$  满足  $\langle x, u_1 \rangle \in L' \Leftrightarrow \forall u_2, \dots, Q_{j+1} u_{j+1}, \text{ s.t. } M(x, u_1, u_2, \dots, u_{j+1}) = 1$ , 根据定义  $L' \in \Pi_j^P = \Sigma_j^P$ , 故存在多项式时间图灵机  $M'$  满足

$$\langle x, u_1 \rangle \in L' \Leftrightarrow \exists u_2, \forall u_3, \dots, Q_j u_{j+1}, \text{ s.t. } M'(x, u_1, u_2, \dots, u_{j+1}) = 1$$

从而我们得到

$$\begin{aligned} x \in L &\Leftrightarrow \exists u_1, \langle x, u_1 \rangle \in L' \\ &\Leftrightarrow \exists u_1, \exists u_2, \forall u_3, \dots, Q_j u_{j+1}, \text{ s.t. } M'(x, u_1, u_2, \dots, u_{j+1}) = 1 \end{aligned}$$

故  $L \in \Sigma_j^P$ , 于是  $\Sigma_{j+1}^P \subseteq \Sigma_j^P$ , 类似地可证明  $\Pi_{j+1}^P \subseteq \Pi_j^P$ , 从而说明  $\Sigma_{j+1}^P = \Pi_{j+1}^P = \Sigma_i^P$ .  $\square$

类似之前完全性的定义, 我们称语言  $L \in \Sigma_i^P$ -complete, 如果  $L \in \Sigma_i^P$  且  $L' \leq_p L$  对任意  $L' \in \Sigma_i^P$  成立. 类似可定义  $\Pi_i^P$ -complete 与  $\mathbf{PH}$ -complete.

**命题 7.1.** 如果存在  $\mathbf{PH}$ -complete 语言, 则  $\mathbf{PH}$  必然坍塌到了某个  $\Sigma_i^P$ .

**例 7.1.** 记  $\Sigma_i \text{SAT}$  为包含所有形如  $\exists u_1 \forall u_2 \dots Q_i u_i, \phi(u_1, \dots, u_i)$  的真值为 True 的 QBF 的语言, 可以证明  $\Sigma_i \text{SAT} \in \Sigma_i^P$ -complete. ( $i = 1$  时就是 Cook Levin Theorem,  $i > 1$  时也可采用类似方法证明.)

**推论 7.1.** 显然  $\mathbf{PH} \subseteq \mathbf{PSPACE}$ . 如果  $\mathbf{PH}$  不坍塌, 则  $\mathbf{PH}$  是  $\mathbf{PSPACE}$  的真子集. 因为如果  $\mathbf{PH} = \mathbf{PSPACE}$ , 那么 TQBF 是  $\mathbf{PH}$ -complete, 导致  $\mathbf{PH}$  坍塌.

## 8 布尔线路

**定义 8.1 (布尔线路).** 对于任意  $n \in \mathbb{N}$ , 一个  $n$  源单汇布尔的线路是一张有  $n$  个源点和 1 个汇点的 DAG, 其中每个非源节点标有逻辑运算符  $\{\wedge, \vee, \neg\}$  中的一者, 且如果标有  $\wedge$  或者  $\vee$ , 则该节点在图中的入度为 2, 如果标有  $\neg$ , 则该节点在图中的入度为 1. 记布尔线路  $C$  的规模  $|C|$  为 DAG 中的节点数.

对于  $n$  源单汇的布尔线路  $C$  和  $x \in \{0, 1\}^n$ , 可以自然地定义  $C(x) \in \{0, 1\}$  表示  $C$  对输入  $x$  得到的输出.

**定义 8.2 (线路族, SIZE).** 对于函数  $T: \mathbb{N} \rightarrow \mathbb{N}$ , 一个  $T(n)$  规模线路族是一个布尔线路序列  $\{C_n\}_{n \in \mathbb{N}}$ , 其中  $C_n$  是  $n$  源单汇布尔线路, 且  $|C_n| \leq T(n)$ .

称语言  $L \in \mathbf{SIZE}(T(n))$ , 如果存在一个  $T(n)$  规模线路族  $\{C_n\}_{n \in \mathbb{N}}$ , 满足对任意  $n \in \mathbb{N}$ , 任意  $x \in \{0, 1\}^n$ ,  $x \in L \Leftrightarrow C_n(x) = 1$ .

**定义 8.3 ( $\mathbf{P/poly}$ ).**  $\mathbf{P/poly}$  是所有多项式规模线路族可判定的语言类, 即  $\mathbf{P/poly} = \bigcup_{c \geq 0} \mathbf{SIZE}(n^c)$ .

**定理 8.1.**  $\mathbf{P} \subseteq \mathbf{P/poly}$ .

证明. 类似于 Cook Levin Theorem 的证明.

由注 3.2, 对于任意  $L \in \mathbf{P}$ , 都存在一台健忘的图灵机  $M$  在多项式时间  $T(n)$  内判定  $L$ . 利用  $M$  的健忘性来构造多项式规模线路族  $\{C_n\}_{n \in \mathbb{N}}$ : 考虑  $M$  运行时的 snapshot  $z_1, z_2, \dots, z_{T(n)}$ , 每个  $z_i$  都可以用常数个比特表达, 且确定  $z_i$  也只需要用到之前的常数个  $z_j$ , 因此构造布尔线路  $C_n$  依次计算每个  $z_i$ , 再把  $C_n$  的输出设置为“ $z_{T(n)}$  是否到达  $M$  的接受态”即可.  $\square$

### 8.1 线路可满足性

**定义 8.4 (CKT-SAT).** 语言 CKT-SAT 包含了所有可满足的布尔线路 (的二进制表示). 即, 对于  $n$  源布尔线路  $C$ ,  $\lfloor C \rfloor \in \text{CKT-SAT}$  当且仅当存在  $u \in \{0, 1\}^n$  使得  $C(u) = 1$ .

**引理 8.1.**  $\text{CKT-SAT} \in \mathbf{NP-hard}$ .

证明. 把定理 8.1 的证明抄过来就行了, 无非是在论证过程之前加了一个  $\exists u$  的量词.  $\square$

**引理 8.2.**  $\text{CKT-SAT} \leq_p 3\text{SAT}$ .

证明. 给定布尔线路  $C$ , 需要构造一个多项式规模的 3CNF  $\varphi$ , 使得  $C$  与  $\varphi$  可满足性相同.

对  $C$  的每个节点  $u$  构造变量  $z_u$ . 对于非源节点  $i$ , 需要限制  $z_i$  的取值是正确的, 当  $C$  中  $z_i = z_j \wedge z_k$  时, 可以在  $\varphi$  中加入以下从句:

$$(\neg z_i \vee z_j \vee z_k) \wedge (\neg z_i \vee z_j \vee \neg z_k) \wedge (\neg z_i \vee \neg z_j \vee z_k) \wedge (z_i \vee \neg z_j \vee \neg z_k)$$

$z_i = z_j \vee z_k, z_i = \neg z_j$  时也可类似构造. 此外, 在  $\varphi$  中加入  $C$  的汇点对应的变量  $v_{\text{sink}}$  作为一个从句.

如此构造出的  $\varphi$  与  $C$  可满足性相同, 且规模是多项式的, 因此得到了  $\text{CKT-SAT} \leq_p 3\text{SAT}$ .  $\square$

### 8.2 线路一致生成

定理 8.1 中的包含事实上是真包含, 因为  $\mathbf{P/poly}$  甚至包含不可判定语言. 比如说,  $\mathbf{P/poly}$  包含所有一进制语言, 即便这样的语言可能是不可判定的.

具体的, 对于任意语言  $L \subseteq \{1^n | n \in \mathbb{N}\}$ , 线路族  $\{C_n\}_{n \in \mathbb{N}}$  只需要对满足  $1^n \in L$  的  $n$ , 构造  $C_n$  使其检查输入的  $n$  位是否全是 1 并输出检查结果, 而对  $1^n \notin L$  的  $n$ , 构造  $C_n$  为直接输出 0.

另一方面, 考虑一进制语言  $\text{UHALT} = \{1^n | n = \lfloor M, x \rfloor \text{ such that } M \text{ halts on input } x\}$ , 不难验证  $\text{HALT} \leq_m \text{UHALT}$ , 因而  $\text{UHALT}$  是不可判定的.

这使得  $\mathbf{P/poly}$  语言类处于一种很奇怪的境地, 其根源在于线路族  $\{C_n\}$  是**不一致 (nonuniform)** 的. 一种直观想法是, 我们希望限制线路族  $\{C_n\}_{n \in \mathbb{N}}$  是“可高效构造的”, 从中引申出了一致生成 (uniformly generate) 的概念.



**定义 8.5 (P-一致线路族).** 一个线路族  $\{C_n\}_{n \in \mathbb{N}}$  被称为 **P-一致**的, 如果存在对输入  $1^n$  输出  $\lfloor C_n \rfloor$  的多项式时间图灵机.

**定理 8.2.** 语言  $L$  可以被 **P-一致线路族**判定, 当且仅当  $L \in \mathbf{P}$ .

证明.  $\Rightarrow$ : 根据定义, 可以先多项式时间地算出  $C_n$ , 再根据  $C_n$  多项式时间地算出结果.

$\Leftarrow$ : 照搬定理 8.1 的证明, 可以发现其中构造的线路族就是 **P-一致**的.  $\square$

**定义 8.6 (对数空间一致线路族).** 一个线路族  $\{C_n\}_{n \in \mathbb{N}}$  被称为对数空间一致 (logspace-uniform) 的, 如果存在将  $1^n$  映到  $\lfloor C_n \rfloor$  的隐对数空间可计算函数 (参见定义 6.8).

**定理 8.3.** 语言  $L$  可以被对数空间一致线路族判定, 当且仅当  $L \in \mathbf{P}$ .

证明.  $\Rightarrow$ :  $\mathbf{L} \subseteq \mathbf{P}$ , 照抄定理 8.2 的  $\Rightarrow$  证明即可.

$\Leftarrow$ : 照抄定理 8.2 的  $\Leftarrow$  证明, 这个线路族的构造方式是对数空间可计算的.  $\square$

### 8.3 线路规模

**定理 8.4.** 存在一个输入为  $n$  比特的布尔函数, 其只能被规模为  $\Omega\left(\frac{2^n}{n}\right)$  的布尔线路所计算.

证明. 考虑计数. 输入  $n$  比特的布尔函数一共有  $2^{2^n}$  个, 而规模为  $T$  的布尔线路只有不超过  $(3T^2)^T$  个. 一个布尔线路只能计算一个布尔函数, 所以  $(3T^2)^T \geq 2^{2^n} \Rightarrow T = \Omega\left(\frac{2^n}{n}\right)$ .  $\square$

**定理 8.5.** 任意一个输入为  $n$  比特的布尔函数, 都可以被规模为  $O\left(\frac{2^n}{n}\right)$  的布尔线路所计算.

证明. 列举一下三种不同量级的构造方式.

- $O(n2^n)$ :  $f: \{0,1\}^n \rightarrow \{0,1\}$  只有不超过  $2^n$  种满足  $f(x) = 1$  的取值  $x$ , 因此把这些  $x$  列举一遍写成一个 DNF 的形式即可.
- $O(2^n)$ : 考虑  $f(x_1, x_2, \dots, x_n) = (x_1 \wedge f(1, x_2, \dots, x_n)) \vee (\neg x_1 \wedge f(0, x_2, \dots, x_n))$ , 其中  $f(1, x_2, \dots, x_n), f(0, x_2, \dots, x_n)$  是另外两个输入  $n-1$  比特的布尔函数, 可以递归构造, 从而线路规模为  $S(n) = O(1) + 2S(n-1) = O(2^n)$ .
- $O\left(\frac{2^n}{n}\right)$ : 是一种**四毛子算法 (Method of Four Russians)**的简单应用. 考虑输入  $k$  比特的布尔函数只有  $2^{2^k}$  个, 可以使用前述方法将这些函数对应的线路全部预构造出来. 修改前述递归构造方法, 在递归到  $n = k$  时停止, 转而直接利用预构造得到的结果. 这种构造方法的电路规模为  $O(2^{n-k} + 2^{2^k+k})$ , 取  $k = \log n - \varepsilon$  即可得到  $O\left(\frac{2^n}{n}\right)$ .

$\square$

**定理 8.6 (Nonuniform Size Hierarchy Theorem).** 对于任意函数  $T, T': \mathbb{N} \rightarrow \mathbb{N}$  满足  $n \leq T(n), T'(n) \leq \frac{2^n}{n}$  且  $20T(n) \log^2 T(n) \leq T'(n)$ , 有

$$\mathbf{SIZE}(T(n)) \subsetneq \mathbf{SIZE}(T'(n))$$

证明. 考虑计数. 任意输入  $l$  比特的布尔函数都可以被规模为  $10l2^l$  的布尔线路所计算, 而也存在这样的布尔函数无法被规模为  $\frac{2^l}{10l}$  的布尔线路所计算, 记之为  $f_l$ . 令  $l = \log T(n) + \log \log T(n)$ , 考虑语言  $L \subseteq \{0,1\}^*$ , 任意长度为  $n$  的串  $x$  满足  $x \in L \Leftrightarrow f_l(x_1, \dots, x_l) = 1$ , 即对  $x$  的前  $l = \log T(n) + \log \log T(n)$  位计算布尔函数  $f_l$ , 其结果作为  $x$  是否属于  $L$  的判定标准. 此时有

$$L \in \mathbf{SIZE}(10l2^l) = \mathbf{SIZE}(10T(n) \log T(n) (\log T(n) + \log \log T(n))) \subseteq \mathbf{SIZE}(20T(n) \log^2 T(n)) \subseteq \mathbf{SIZE}(T'(n))$$

$$L \notin \mathbf{SIZE}\left(\frac{2^l}{10l}\right) = \mathbf{SIZE}\left(\frac{T(n) \log T(n)}{10(\log T(n) + \log \log T(n))}\right) \supseteq \mathbf{SIZE}(T(n))$$

从而  $\mathbf{SIZE}(T(n)) \subsetneq \mathbf{SIZE}(T'(n))$ .  $\square$



## 8.4 接受建议的图灵机

**定义 8.7 (接受建议的图灵机).** 考虑一个这样的语言类, 其包含所有这样的语言  $L$ : 存在一系列字符串 (建议)  $\{\alpha_n\}_{n \in \mathbb{N}}$  满足  $|\alpha(n)| = a(n)$ , 以及运行时间为  $T(n)$  的图灵机  $M$ , 满足  $x \in L \Leftrightarrow M(x, \alpha_{|x|}) = 1$ . 把这样的语言类记作  $\mathbf{DTIME}(T(n))/a(n)$ .

**定理 8.7.**  $\mathbf{P}_{/\text{poly}} = \bigcup_{c,d \geq 0} \mathbf{DTIME}(n^c)/n^d$ .

证明.  $\subseteq$ : 把线路族  $\{C_n\}_{n \in \mathbb{N}}$  编码为建议  $\{\alpha_n\}_{n \in \mathbb{N}}$  即可.

$\supseteq$ : 类似定理 8.1 中的构造, 可以得到多项式规模线路族  $\{D_n\}_{n \in \mathbb{N}}$  满足对于任意  $x \in \{0, 1\}^n, \alpha \in \{0, 1\}^{a(n)}$  都有  $D_{n+a(n)}(x, \alpha) = M(x, \alpha)$ . 将  $\alpha_n$  硬编码进  $D_n$  即可得到  $\{C_n(x) = D_{n+a(n)}(x, \alpha(n))\}_{n \in \mathbb{N}}$ .  $\square$

## 8.5 Karp-Lipton Theorem

有一个问题是,  $\text{SAT} \in \mathbf{P}_{/\text{poly}}$  吗? 或者等价的,  $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}}$  吗?

Karp, Lipton 二位给出了答案: 只要 polynomial hierarchy  $\mathbf{PH}$  不坍塌, 答案就是 “否”.

**定理 8.8 (Karp-Lipton Theorem).** 如果  $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}}$ , 那么  $\mathbf{PH} = \Sigma_2^P$ .

证明. 考虑证明  $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}} \Rightarrow \Pi_2\text{SAT} \in \Sigma_2^P \Rightarrow \Pi_2^P \subseteq \Sigma_2^P \Rightarrow \mathbf{PH} = \Sigma_2^P$ . 其中

$$\Pi_2\text{SAT} = \{\psi = \forall u \exists v \phi(u, v) \mid \psi \text{ is satisfiable}\}$$

可以验证  $\Pi_2\text{SAT} \in \Pi_2$ -complete.

考虑语言  $L = \{\langle \phi, u \rangle \mid \exists v, \phi(u, v) = 1\}$ , 不难看出  $L \leq_p \text{SAT}$ , 从而  $L \in \mathbf{NP}$ . 由于  $\mathbf{NP} \subseteq \mathbf{P}_{/\text{poly}}$ , 故存在多项式规模线路族  $\{C_n\}_{n \in \mathbb{N}}$  可以判定语言  $L$ , 即, 对于任意  $\langle \phi, u \rangle, \langle \phi, u \rangle \in L$  当且仅当  $C_n(\phi, u) = 1$ .

根据之前定理 5.6 的讨论, 可以把 search problems 多项式时间规约到 decision problems, 考虑其布尔线路版本, 也即存在另一个多项式规模线路族  $\{C'_n\}_{n \in \mathbb{N}}$ , 使得对于任意  $\langle \phi, u \rangle, \exists v, \phi(u, v) = 1$  当且仅当  $\phi(u, C'_n(\phi, u)) = 1$ , 即  $C'_n(\phi, u)$  输出了满足条件的  $v$ .

注意到  $C'_n$  的构造是与具体的  $\phi, u, v$  无关的, 因此其对应量词可交换. 同时注意到在给出  $\phi, u$  以及  $\{C'_n\}_{n \in \mathbb{N}}$  时, 可以构造一台多项式时间图灵机  $M$  来计算  $\phi(u, C'_n(\phi, u))$  的结果.

$$\begin{aligned} \psi &= \forall u \exists v \phi(u, v) \in \Pi_2\text{SAT} \\ &\Leftrightarrow \forall u, \langle \phi, u \rangle \in L \\ &\Leftrightarrow \forall u, \exists C_n, C_n(\phi, u) = 1 \\ &\Leftrightarrow \forall u, \exists C'_n, \phi(u, C'_n(\phi, u)) = 1 \\ &\Leftrightarrow \exists C'_n, \forall u, \phi(u, C'_n(\phi, u)) = 1 \\ &\Leftrightarrow \exists C'_n, \forall u, M(\psi, u, C'_n) = 1 \end{aligned}$$

从而说明  $\Pi_2\text{SAT} \in \Sigma_2^P$ . 接下来的  $\Pi_2\text{SAT} \in \Sigma_2^P \Rightarrow \Pi_2^P \subseteq \Sigma_2^P \Rightarrow \mathbf{PH} = \Sigma_2^P$  都比较简单, 故不再赘述.  $\square$

这说明了什么? 考虑逆否命题, 这说明了如果  $\mathbf{PH}$  没有坍塌到  $\Sigma_2^P$ , 则  $\mathbf{NP}$  语言, 或者更具体地,  $\text{SAT}$ , 不能被多项式规模线路族判定.

不过这个结论并没有对  $\mathbf{P} \neq \mathbf{NP}$  的研究做出任何进展性贡献, 因为  $\mathbf{PH} \neq \Sigma_2^P$  已经是比  $\mathbf{P} \neq \mathbf{NP}$  更强的结论了.

## 8.6 线路深度

线路深度与并行计算有很大的关系, 因此有很大的研究必要.

**定义 8.8 (NC, AC).** 称语言  $L \in \mathbf{NC}^c$ , 如果  $L$  可以被线路族  $\{C_n\}_{n \in \mathbb{N}}$  计算, 其中  $C_n$  的规模为  $\text{poly}(n)$ , 深度为  $O(\log^c n)$ . 记  $\mathbf{NC} = \bigcup_{c \geq 0} \mathbf{NC}^c$ .

称语言  $L \in \mathbf{AC}^c$ , 如果  $L$  可以被线路族  $\{C_n\}_{n \in \mathbb{N}}$  计算, 其中  $C_n$  的规模为  $\text{poly}(n)$ , 深度为  $O(\log^c n)$ , 同时  $C_n$  中对 AND, OR 门的输入数量不作限制. 记  $\mathbf{AC} = \bigcup_{c \geq 0} \mathbf{AC}^c$ .

**命题 8.1.**  $\mathbf{NC}^i \subseteq \mathbf{AC}^i \subseteq \mathbf{NC}^{i+1}$ , 对任意  $i \geq 0$  成立.

**定理 8.9.**  $\mathbf{NC}^0 \subsetneq \mathbf{AC}^0 \subsetneq \mathbf{NC}^1$ .