计算理论导论——期末复习

陈志朋

Contents

1 总览

所有语言的关系如图1。

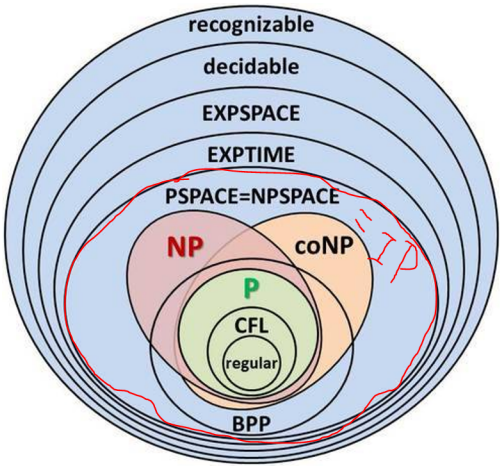

Figure 1: Overview

2 正则语言

2.1 有穷自动机

2.1.1 有穷自动机的形式定义

定义 2.1 有穷自动机是一个 5 元组 $(Q, \Sigma, \delta, q_0, F)$：

1) Q 是一个有穷集合，叫做**状态集**。
2) Σ 是一个有穷集合，叫做**字母集**。
3) $\delta : Q \times \Sigma \to Q$ 是**转移函数**。
4) $q_0 \in Q$ 是**起始状态**。
5) $F \subseteq Q$ 是接受状态集。

2.1.2 有穷自动机举例

例 2.3 接受语言 $L(M) = \{w | w \ is \ the \ empty \ string \ \varepsilon \ or \ ends \ in \ a \ 0\}$ 的 DFA，示意图 2

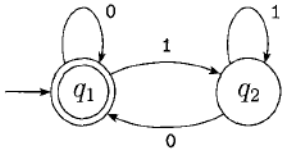

Figure 2: A DFA of language M

2.1.3 计算的形式定义

当一个字符串 w 从有穷自动机的起始状态开始，按照转移函数从一个状态到一个状态，并且机器结束在接受状态，则称 M **接受** w。如果 $A = \{w | Macceptsw\}$，则称 M **识别语言** A。

定义 2.7 如果一个语言被一台**有穷自动机**识别，则称它是**正则语言**。

2.1.4 设计有穷自动机

例 2.9 有穷自动机 E_2 识别含有 001 作为子串的所有字符串组成的正则语言，示意图 3

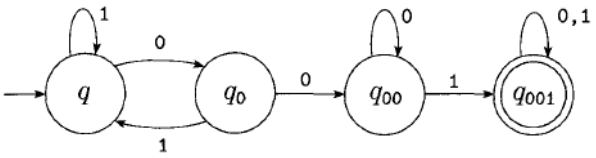

Figure 3: DFA E_2

2.1.5 正则运算

定义 2.10 设 A 和 B 是两个语言，定义正则运算**并**、**连接**和**星号**如下：

并：$A \cup B = \{x | x \in A \ or \ x \in B\}$

连结： $A \circ B = \{xy | x \in A \ and \ y \in B\}$

星号： $A^* = \{x_1 \dot{x}_k | k \geq 0 \ and \ \forall x_i \in A\}$

定理 2.12 正则语言类在并运算下封闭。

证明思路 设 $M_1 = Q_1, \Sigma, \delta_1, q_1, F_1$ 可以识别 A，设 $M_2 = Q_2, \Sigma, \delta_2, q_2, F_2$ 可以识别 B。 构造 $M = Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), (F_1 \times Q_2) \cup (F_2 \times Q_1)$，其中 $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$，可以识别 $A \cup B$。

定理 2.13 正则语言类在连结运算下封闭。

证明思路 引入非确定性的新技术，即使用非确定型有穷自动机来证明。

2.2 非确定性

2.2.1 非确定型有穷自动机的形式定义

定义 2.17 **非确定型有穷自动机**是一个 5 元组 $(Q, \Sigma, \delta, q_0, F)$，其中

1) Q 是有穷的状态集。
2) Σ 是有穷的字母表。
3) $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ 是转移函数。
4) $q_0 \in Q$ 是起始状态。
5) $F \subseteq Q$ 是接受状态集。

2.2.2 NFA 与 DFA 的等价性

定理 2.19 每一台非确定型有穷自动机都有一条等价的确定型有穷自动机。

证明思路 把能到达的子集合并，构造新的状态。

推论 2.20 一个语言是正则的，当且仅当有一台非确定型的有穷自动机识别它。

证明思路 NFA 和 DFA 等价。

2.2.3 在正则运算下的封闭性

定理 2.22 正则语言在并运算下封闭。

证明思路 示意图 4。

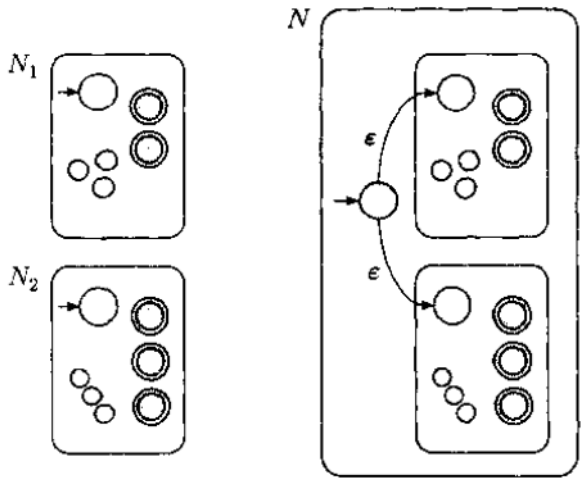

图 2-24 识别 $A_1 \cup A_2$ 的 NFA N 的构造

Figure 4: NFA recognizes $A = A_1 \cup A_2$

定理 2.23 正则语言在连结运算下封闭。

证明思路 示意图 5。

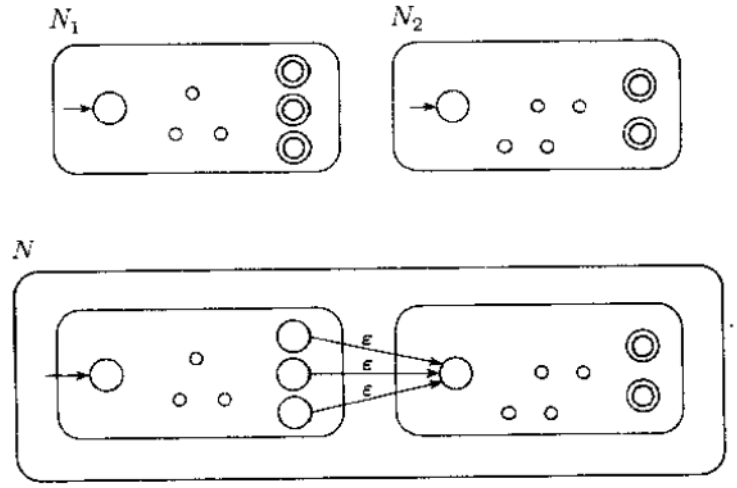

图 2-25 识别 $A_1 \circ A_2$ 的 N 的构造

Figure 5: NFA recognizes $A = A_1 \circ A_2$

定理 2.22 正则语言在星号运算下封闭。

证明思路 示意图 6。

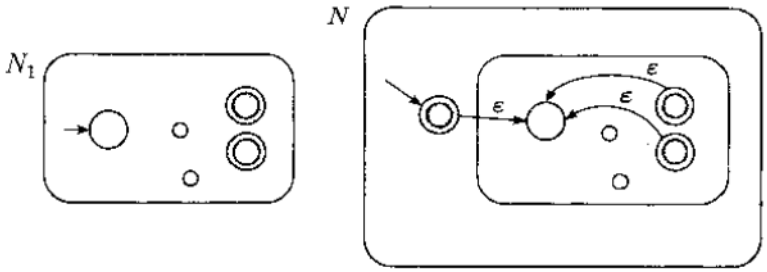

图 2-26 识别 A_1^* 的 N 的构造

Figure 6: NFA recognizes $A = A_1^*$

2.3 正则表达式

2.3.1 正则表达式的形式定义

定义 2.26 称 R 是一个正则表达式，如果 R 是：

1) a，这里的 a 是字母表 Σ 中的一个元素。
2) ε 。
3) $\emptyset$ 。
4) $(R_1 \cup R_2)$，这里的 R_1 和 R_2 是正则表达式。
5) $(R_1 \circ R_2)$，这里的 R_1 和 R_2 是正则表达式。
6) (R_1^*)，这里的 R_1 是正则表达式。

连接符号和括号可以省略。如果没有括号，优先级从高到低为：星号、连结、并。

2.3.2 与有穷自动机的等价性

定理 2.28 一个语言是正则的，当且仅当可以用正则表达式描述它。

证明思路 使用引理 2.29 和引理 2.30。

引理 2.29 如果一个语言可以用正则表达式描述，则它是正则的。

证明思路 对于每种正则运算有对应的 NFA 构造方法。

引理 2.30 如果一个语言是正则的，则它可以用正则表达式描述。

证明思路 把 DFA 转化为 GNFA，然后把 GNFA 转化为正则表达式。

定义 2.33 **广义非确定型有穷自动机** $(Q, \Sigma, \delta, q_{start}, q_{accept})$ 是一个 5 元组，其中：

1) Q 是有穷的状态集。

2) Σ 是输入字母表。

3) $\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\})$。

4) q_{start} 是起始状态。

5) q_{accept} 是接受状态。

如果字符串 w 可写成 $w = w_1, w_2 \dots w_k$，这里每一个 $w_i \in \Sigma^*$，并且存在状态序列 $q_0, q_1, \dots, q_k$，使得：

1) $q_0 = q_{start}$ 是起始状态。

2) $q_0 = q_{accept}$ 是接受状态。

3) 对于每一个 i，$w_i \in L(R_i)$，其中 $R_i = \delta(q_{i-1}, q_i)$，换句话说，$R$ 是从 q_{i-1} 到 q_i 的箭头上的表达式。

则称这台 GNFA 接受字符串 w。

GNFA 的特殊形式：

1) 起始状态有射到其他每一个状态的箭头，但是没有从任何其他状态射入的箭头。

2) 有唯一的一个接受状态，并且他有从每一个状态射入的箭头，但是没有射到任何其他状态的箭头。此外，这个接受状态与其实状态不同。

3) 除起始状态和接受状态之外，每一个状态到自身和其他每一个状态都有一个箭头。

DFA 到 GNFA 的转化：添加新的起始状态和新的接受状态，从新起始状态到老起始状态有一个 ε 箭头，从每一个老接受状态到新接受状态有一个 ε 箭头。对于其他状态，没有边的连一个 $\emptyset$ 的箭头，一条边上有多个的字母的取并集。

GNFA 到正则表达式的转化：当状态数 $k > 2$ 时，选取一个状态（非起始状态和接受状态），删除这个状态，然后对其他状态的路径进行修改，示意图 7。

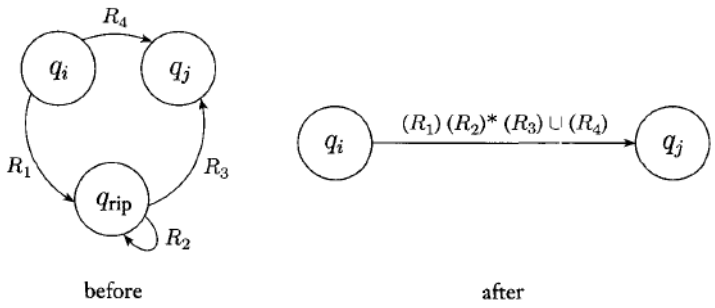

Figure 7: Delete q_{rip}

2.4 非正则语言

定理 2.37 泵引理（Pumping Lemma）

设 A 是一个正则语言，则存在一个数 p（泵长度 Pumping Length）使得，如果 s 是 A 中任一长度不小于 p 的字符串，那么 s 可以被分成 3 段，$s = xyz$，满足下述条件：

1) 对**每一个** $i \geq 0$，$xyz \in A$

2) $|y| > 0$

2) $|xy| \leq p$

证明思路 设 M 是识别 A 的 DFA，令泵长度（Pumping Length）p 等于 M 的状态数，起始节点也算一个状态，所以一共经过了 $p+1$ 个状态。根据鸽笼原理一定会经过重复结点，这个重复状态（即 y）可以进行扩展，也可以把这段删除。

例 2.38 证明语言 $B = \{0^n 1^n | n \geq 0\}$ 不是正则的。

证明

假设语言 B 是正则的，那么存在 Pumping Length，设为 p。

取 $s = 0^p 1^p$，那么 s 可以被分解为 3 部分，即 $s = xyz$，其中 $|y| > 0$，且 $|xy| \leq p$。那么，$xy = 0^{t_1+t_2}$，其中 $t_1 = |x|, t_2 = |y|$ 且 $t_2 > 0$ $t_1 + t_2 \leq p$。

取 $i = 0$，则 $xy^0 z = xz = 0^{n-t_2} 1^n$，显然 $n - t_2 \neq n$，所以 $xy^0 z \notin B$，矛盾！

综上，语言 B 不是正则的。

3 上下文无关语言

3.1 上下文无关文法

上下文无关文法：一个文法有一组**替换规则**组成，替换规则又叫做**产生式**。在上述文法中每一个规则占一行，由一个符号和一个字符串构成，符号和字符串之间同一个箭头隔开。这个符号叫做**变元**。字符串由边缘和另一种叫做终结符的符号组成。变元经常用大写字母表示。终结符类似于输入符号，经常用小写字母、数字或特殊符号表示。一个变元被指定为**起始变元**，通常它出现在第一条规则的左边。

3.1.1 上下文无关文法的形式定义

定义 3.1 **上下文无关文法**是一个 4 元组 (V, Σ, R, S)，这里

1) V 是一个有穷集合，称作**变元集**。

2) Σ 是一个与 V 不相交的有穷集合，称作**终结符集**。

3) R 是一个有穷的**规则集**，每一条规则是一个变元和一个由变元和终结符组成的字符串。

4) $S \in V$ 是**起始变元**。

3.1.2 上下文无关文法举例

例 3.2 考虑文法 $G_3 = (\{S\}, \{a, b\}, R, S)$，其中规则集 R 为：

$$S \to aSb \mid SS \mid \varepsilon$$

3.1.3 设计上下文无关文法

略

3.1.4 歧义性

定义 3.4 如果字符串 w 在上下文无关文法 G 中有两个或两个以上不同的最左派生，则称在 G 中**歧义地**产生字符串 w。如果文法 G 歧义地产生某个字符串，则称 G 是**歧义的**。

一个示例 8。

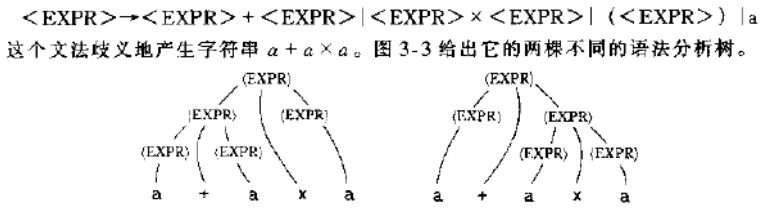

Figure 8: Ambitious grammar

3.1.5 乔姆斯基范式

定义 3.5 一个上下文无关文法为**乔姆斯基范式**，如果它的每一个规则具有如下形式：

$$A \to BC$$

$$A \to a$$

这里，a 是任意的终结符，A、B 和 C 是任意的变元，但 B 和 C 不能是起始变元。此外，允许规则 $S \to \varepsilon$，其中 S 是起始变元。

定理 3.6 任一上下文无关语言都可以用乔姆斯基范式的上下文无关文法产生。

证明思路 分四阶段将上下文无关语法转化为乔姆斯基范式：

1) 增加新的起始变元 S_0，然后增加规则 $S_0 \to S$。

2) 如果存在 $A \to \varepsilon$，删除这条规则，然后修改其他和 A 有关的规则。

3) 处理所有单一规则。对于单一规则 $A \to B$，删除这条，然后增加 A 生成 B 能产生的字符串的规则。举例：对于 $A \to B$ 和 $B \to u$，删除第一条，增加一条 $A \to u$。

4) 把所有留下的规则转换称适当的形式。

3.2 下推自动机

3.2.1 下推自动机的形式定义

定义 3.8 **下推自动机**是一个 6 元组 $(Q, \Sigma, \Gamma, \delta, q_0, F)$，这里 Q，Σ，Γ 和 F 都是有穷集合，并且

1) Q 是状态集。

2) Σ 是输入字母表。

3) Γ 是栈字母表。

4) $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to \mathcal{P}(Q \times \Gamma_\varepsilon)$

5) $q_0 \in Q$ 是起始状态。

6) $F \subseteq Q$ 是接受状态。

3.2.2 下推自动机举例

例 3.9 下面是识别语言 $\{0^n 1^n | n \geq 0\}$ 的 PDA，示意图 9。

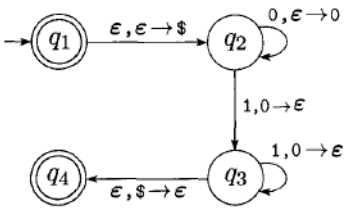

Figure 9: PDA recognizes $\{0^n1^n|n \geq 0\}$

3.2.3 与上下文无关语法的等价性

定理 3.12 一个语言是上下文无关的，当且仅当存在一台下推自动机识别它。

证明思路 使用引理 3.13 和引理 3.15。

引理 3.13 如果一个语言是上下文无关的，则存在一台下推自动机识别它。

证明思路 非确定性的枚举，然后压到栈中，和输入进行比较，如果不成立就拒绝该计算分支。如果有一个计算分支接受则接受。

引理 3.15 如果一个语言被一台下推自动机识别，则它是上下文无关的。

证明思路 粗略的证明：使用分治的思想，A_{pq} 可以由 A_{pr} 和 A_{rq} 产生，如图所示10。

PROOF Say that $P = (Q, \Sigma, \Gamma, \delta, q_0, \{q_{\text{accept}}\})$ and construct G. The variables of G are $\{A_{pq} | p, q \in Q\}$. The start variable is $A_{q_0, q_{\text{accept}}}$. Now we describe G's rules.

- For each $p, q, r, s \in Q$, $t \in \Gamma$, and $a, b \in \Sigma_\varepsilon$, if $\delta(p, a, \varepsilon)$ contains (r, t) and $\delta(s, b, t)$ contains (q, ε), put the rule $A_{pq} \to aA_{rs}b$ in G.
- For each $p, q, r \in Q$, put the rule $A_{pq} \to A_{pr}A_{rq}$ in G.
- Finally, for each $p \in Q$, put the rule $A_{pp} \to \varepsilon$ in G.

You may gain some insight for this construction from the following figures.

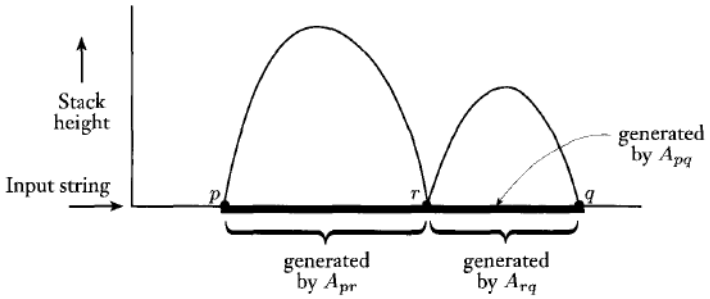

Figure 10: PDA computation corresponding to the rule $A_{pq} \to A_{pr}A_{rq}$

3.3 非上下文无关语言

定理 3.19 （**关于上下文无关语言的泵引理（Pumping Lemma）**）如果 A 是上下文无关语言，则存在数 p（泵长度 Pumping Length），使得 A 中热河一个长度不小于 p 的字符串 s 能够划分成 5 段，$s = uvxyz$，满足下述条件：

1) 对于每一个 i，$uv^ixy^iz \in A$。

2) $|vy| > 0$。

3) $|vxy| \leq p$。

证明思路 鸽笼原理。取 $p = b^{|V|+2}$，其中 b 为一个结点最多的儿子数，$|V|$ 为 G 中变元的数目（由于叶子节点为终结符，所以这里要加 2）。示意图 11。

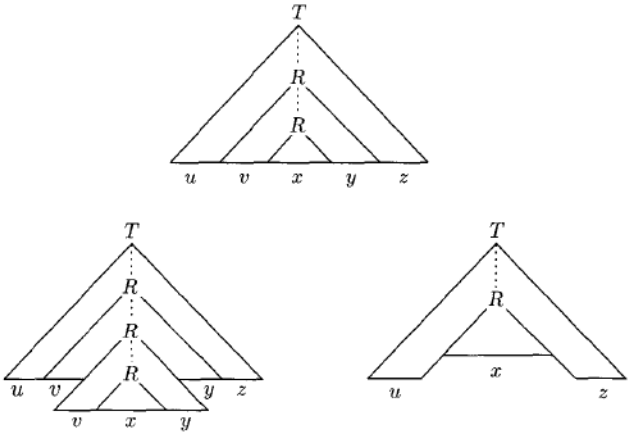

Figure 11: Surgery on parse trees

例 3.20 用 Pumping Lemma 证明语言 $B = \{a^n b^n c^n | n \geq 0\}$ 不是上下文无关的。

证明

假设语言 B 是上下文无关的，设 Pumping Length 为 p。

令 $s = a^p b^p c^p$，则 s 可以被分解为 5 部分，$s = uvxyz$。其中，$|vxy| \leq p$，$|vy| > 0$。显然，vy 中至多包含两种字符。

取 $i = 0$，即 $uv^0xy^0z = uxz$，至少由一种字符的数量减少了，同时至少由一种字符的数量没变。显然，$uxz \notin B$，矛盾！

综上，语言 B 不是上下文无关的。

4 丘奇—图灵论题

4.1 图灵机

4.1.1 图灵机的形式定义

定义 4.1 一个图灵机是一个 7 元组 $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$，其中：$Q, \Sigma, \Gamma$ 是有穷集合，并且：

1) Q 是状态集
2) Σ 是输入字母表，不包括特殊空白符号。
3) Γ 是带字母表，其中：空白符号属于 Γ，$\Sigma \subseteq \Gamma$。
4) $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ 是转移函数。
5) $q_0 \in Q$ 是起始状态。
6) $q_{accept} \in Q$ 是接受状态。
7) $q_{reject} \in Q$ 是拒绝状态。

图灵机计算时，当前状态、当前带内容和读写头位置，这三项构成的整体叫做图灵机的**格局**。

定义 4.2 如果有图灵机识别一个语言，则称该语言是**图灵可识别**的。

在输入上运行一个 TM 是，可能出现三种结果：接受、拒绝和循环。**这里的循环仅仅指机器不停机**。

定义 4.3 称一个语言是**图灵可判定**的，或简单地讲，是**可判定**的，如果有图灵机判定它。

称对所有输入都停机的图灵机为**判定器**，他们永不循环，总能决定接受还是拒绝。也称识别某个语言的判定器**判定**该语言。

图灵可判定语言都是图灵可识别语言，但某些图灵可识别语言不是可判定的。

4.1.2 图灵机的例子

例 4.4 现在描述 TM M_2，它识别的语言是所有由 0 组成，长度为 2 的放米的字符串，即他判定的语言是 $A = \{0^{2^n} | n \geq 0\}$。

M_2= “对于输入字符串 w：

1) 从左往右扫描整个带子，隔一个消去一个 0。
2) 如果在第一步之后，带子上只剩下唯一的一个 0，则接受。
3) 如果在第一步之后，带子上包含不只一个 0，并且 0 的个数是奇数，则拒绝。
4) 让读写头返回带子的最左端。
5) 转到第一步。”

下面给出 $M_2 = (Q, \Sigma, \Gamma, \delta, q_1, q_{accept}, q_{reject})$ 的形式描述：

- $Q = \{q_1, q_2, q_3, q_4, q_5, q_{accept}, q_{reject}\}$
- $\Sigma = \{0\}$
- $\Gamma = \{0, x, BLANK\}$
- 将 δ 描述成状态图 12
- 开始、接受和拒绝状态分别是 q_1、q_{accept} 和 q_{reject}。

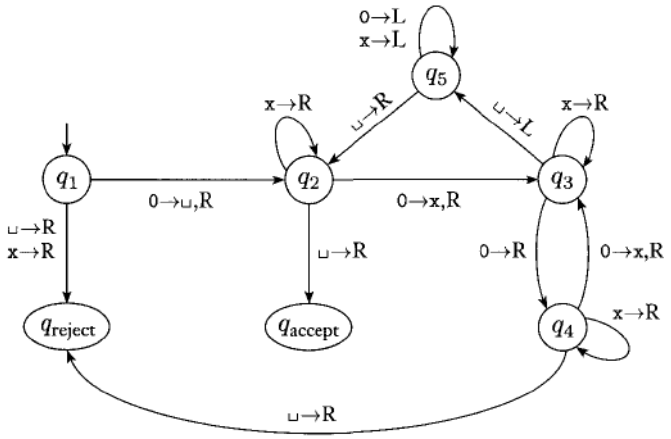

Figure 12: State diagram for Turing machine M_2

4.2 图灵机的变形

4.2.1 多带图灵机

多带图灵机，在普通图灵机的基础上，有多个带子，每个带子都有子集的读写头，用于读和写。开始时，输入出现在第一个带子上，其余的带子都是空白的。转移函数该为允许同时进行读、写和移动读写头，其形式为：

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

定理 4.8 每个多带图灵机都有一个与之等价的单带图灵机。

证明思路 把多条纸带的内容写道一条纸带上，用特殊符号分隔。

推论 4.9 一个语言是图灵可识别的，当且仅当有多带图灵机识别它。

证明思路 多带图灵机可以转化为单带图灵机。

4.2.2 非确定型图灵机

非确定型图灵机，在计算的任何时刻，机器可以在多种可能性中选择一种继续进行，它的转移函数具有如下形式：

$$\delta : Q \times \Gamma \to \mathcal{P}_3(Q \times \Gamma \times \{L, R, S\})$$

定理 4.10 每个非确定型图灵机都有一个与之等价的确定型图灵机。

证明思路 使用宽度优先搜索的方法（BFS）模拟非确定型的计算分支，一共使用三条纸带。

1）第一条纸带用于储存输入，只读不写。

2）第二条纸带用于进行与原来 NTM 类似的计算。

3）第三条纸带用与存储可能转移到的下一组格局。

推论 4.11 一个语言是图灵可识别的，当且仅当有非确定型图灵机识别它。

证明思路 非确定型图灵机能转化为多带图灵机，多带图灵机可以转化为单带图灵机。

推论 4.12 一个语言是可判定的，当且仅当有非确定型图灵机判定它。

4.2.3 枚举器

定理 4.13 一个语言是图灵可识别的，当且仅当有枚举器枚举它。

证明思路

如果有枚举器 E 枚举语言 A，则有 TMM 识别 A。TMM 如下运行：

M=“对与输入 w：

1）运行 E，每当 E 输出一个串时，将之与 w 比较。

2）如果 w 曾经在 E 的输出中出现过，则接受。”

如果 TMM 识别语言 A，为 A 构造枚举器 E 如下：

E=“忽略输入。

1）对 $i = 1, 2.3, \ldots$ 重复下列步骤。

2）对 $s_1, s_2, \ldots, s_i$ 中的每一个，让 M 以其作为输入运行 i 步。

3）如果有计算接受，则打印出相应的 s_j。

4.2.4 与其他模型的等价性

略

4.3 算法的定义

4.3.1 希尔伯特问题

希尔伯特问题

$$D = \{p | p \ is \ a \ polynomial \ with \ an \ integral \ root\}$$

对于一元多项式，其根在 $\pm k\frac{c_{max}}{c_1}$。其中，k 是此多项式中项的个数，c_{max} 是绝对值最大的系数，c_1 是最高次项的系数。因此，一元多项式是否有整数根的问题是可判定的。

但是，多元多项式是否有整数根的问题是不可判定的。

4.3.2 描述图灵机的术语

略。

5 可判定性

5.1 可判定语言

5.1.1 与正则语言相关的可判定性

定理 5.1 $A_{DFA} = \{< B, w > | B \text{ is } DFA, \ w \text{ is string}, \ B \text{ accepts } w\}$ 是一个可判定语言。

证明思路 在输入 w 上模拟 B 即可。

定理 5.2 $A_{NFA} = \{< B, w > | B \text{ is } NFA, \ w \text{ is string}, \ B \text{ accepts } w\}$ 是一个可判定语言。

证明思路 先将 NFA 转化为 DFA，然后在输入上模拟即可。

定理 5.3 $A_{REX} = \{< B, w > | B \text{ is regular expression that generates string } w\}$ 是一个可判定语言。

证明思路 先将正则表达式转化为 DFA，然后在输入上模拟即可。

定理 5.4 $E_{DFA} = \{< A > | A \text{ is a } DFA \text{ and } L(A) = \emptyset\}$ 是一个可判定语言。

证明思路 从 DFA 的起始状态开始遍历，如果不能到达接受状态就接受，否则就拒绝。

定理 5.5 $EQ_{DFA} = \{< A, B > | A \text{ and } B \text{ are } DFAs \text{ and } L(A) = L(B)\}$ 是一个可判定语言。

证明思路 构造 DFAC，满足 $L(C) = (L(A) \cap \overline{L(B)}) \cup (\overline{L(A)} \cap L(B))$。判断 $L(C)$ 是否为空，为空则接受，否则拒绝。构造 $L(C)$ 的时候转化成正则表达式，正则语言类在补、并和交下是封闭的。

5.1.2 与上下文无关语言相关的可判定性问题

定理 5.6 $A_{CFG} = \{< G, w > | G \text{ is a } CFG \text{ that generates string } w\}$ 是一个可判定语言。

证明思路 转成乔姆斯基文法，列出 $2n-1$ 步的所有派生（n 为 w 的长度），判断 w 是否在其中。

定理 5.7 $E_{CFG} = \{< G > | G \text{ is a } CFG \text{ and } L(G) = \emptyset\}$ 是一个可判定语言。

证明思路 把终结符都打上标记，然后对于规则 $A \to U_1 \dots U_k$，如果 $U_1, \dots, U_k$ 都有标记，就把 A 也打上标记。如果起始状态没被标记则接受，否则拒绝。

定理 5.8 每个上下文无关语言是可判定的。

证明思路 使用定理 5.6 提到的图灵机。

5.2 停机问题

定理 5.9 $A_{TM} = \{< M, w > | M \text{ is a } TM \text{ that accepts } w\}$ 是不可判定的。

A_{TM} 有时被成为停机问题。

5.2.1 对角化方法

定义 5.10 设 A 和 B 两个集合，f 是从 A 到 B 的函数。如果 f 从步将两个不同元素映射到同一地方，即：只要 $a \neq b$ 就有 $f(a) \neq f(b)$，则称 f 是**一对一**的。对于 B 中的每个元素 b，都存在 $a \in A$，使得 $f(a) = b$，则称 f 是**到上的**。如果一个函数即使一对一又是到上的，那么称为**对应**。

例 5.11 自然数集合 $\mathcal{N} = \{1, 2, 3, \dots\}$ 和偶自然数集合 $\varepsilon = \{2, 4, 6, \dots\}$ 具有相同的规模。

定义 5.12 成一个集合是可数的，如果它是有限的，或者它与 $\mathcal{N}$ 有相同的规模。

例 5.13 正有理数集合 $\mathcal{Q}$ 与自然数集合 $\mathcal{N}$ 具有相同的规模，示意图 13。

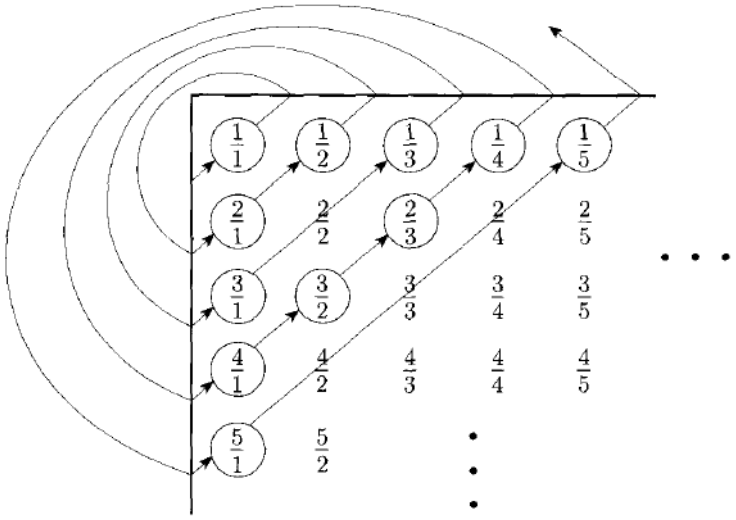

Figure 13: A correspondence of $\mathcal{N}$ and $\mathcal{Q}$

定理 5.14 $\mathcal{R}$ 是不可数的。

证明思路 假设每个自然数都对应了一个实数，那么取 $x \in \mathcal{R}$，其中 x 的整数位为 0，第 i 位小数与自然数 i 的第 i 位小数不同。因此，该实数没有被任意一个自然数映射。

推论 5.15 存在语言不是图灵可识别的。

证明思路 证明分为两部分：

1）所有图灵机构成的集合是可数的。

2）所有语言构成的集合是不可数的。

5.2.2 停机问题是不可判定的

证明

假设 A_{TM} 是可判定的。设 H 是 A_{TM} 的判定器。令 M 是一个 TM，w 是一个串。

$$H(< M, w >) = \begin{cases} accept & \text{M accepts w} \\ reject & \text{M rejects w} \end{cases}$$

构造一个新的图灵机 D，输入为一个图灵机 M。在输入 $< M, < M >>$ 上运行 H，并且得出与 H 相反的结论。

当以 D 的描述 $< D >$ 作为输入来运行 D 自身时，有

$$D(< D >) = \begin{cases} accept & \text{D rejects <D>} \\ reject & \text{D accepts <D>} \end{cases}$$

矛盾！

综上，图灵机 H 是不存在的。

5.2.3 一个图灵不可识别语言

定理 5.16 一个语言是可判定的，当且仅当它既是图灵可识别的，也是补图灵可识别的。

证明思路证明分两个部分：

1）如果 A 是可判定的，显然 A 和 $\overline{A}$ 都是图灵可识别的。因为任何可判定语言都是图灵可识别语言，任何可判定语言的补都是可判定语言。

2）构造两个图灵机：M_1 识别 A，M_2 识别 $\overline{A}$。构造图灵机 M：在 M_1 和 M_2 上同时运行 A（可以采用多带图灵机）。如果 M_1 接受就接受，M_2 接受就拒绝。

推论 5.17 $\overline{A_{TM}}$ 不是图灵可识别的。

证明思路 如果 $\overline{A_{TM}}$ 是图灵可识别的，那么可以推出 A_{TM} 是可判定的，矛盾！

6 可归约性

6.1 语言理论中的不可判定问题

定理 6.1 $HALT_{TM} = \{< M, w > | M \text{ is a TM and } M \text{ halts on input } s\}$ 是不可判定的。

证明思路 用反证法，A_{TM} 可以归约到 $HALT_{TM}$。

定理 6.2 $E_{TM} = \{< M > | M \text{ is a TM and } L(M) = \emptyset\}$ 是不可判定的。

证明思路 用反证法，把 A_{TM} 归约到 E_{TM}。对于一个 TM，先判断输入是否为 w，如果不是直接拒绝，否则执行图灵机。如果 E_{TM} 可判定，那么 A_{TM} 也是可判定的。

定理 6.3 $REGULAR_{TM} = \{< M > | M \text{ is a TM and } L(M) \text{ is a regular language}\}$ 是不可判定的。

证明思路 用反证法，如果 $REGULAR_{TM}$ 是可判定的，那么 A_{TM} 也是可判定的。

设判定 $REGULAR_{TM}$ 的图灵机为 R，判定 A_{TM} 的图灵机为 S。对于输入 $< M, w >$，构造图灵机 M_2，如果输入为 0^n1^n 的形式，则接受。否则，在输入 w 上运行 M，M 接受 M_2 就接受，否则拒绝。在输入 $< M_2 >$ 上运行 R，如果 R 接受则 S 接受，否则拒绝。

补充定理 **Race Theorem** Let P be any problem about TM that satisfies the following two properties. As usual, P is expressed as a language.

1) For any TMs M_1 and M_2, where $L(M_1) = L(M_2)$, we have $< M_1 > \in P \Leftrightarrow < M_2 > \in P$

2) There exist TMs M_1 and M_2, where $< M_1 > \in P$, $< M_2 > \notin P$.

Then P is undecidable .

定理 6.4 $EQ_{TM} = \{< M_1, M_2 > | M_1 \text{ and } M_2 \text{ are TMs and } L(M_1) = L(M_2)\}$ 是不可判定的。

证明思路 用反证法，在输入 $< M, M_1 >$ 上运行 EQ_{TM}，其中 M_1 拒绝所有输入，即可判定 E_{TM} 问题，矛盾。

定义 6.5 设 M 是一个图灵机，w 是一个串。M 在 w 上的一个**接受计算历史**是一个格局序列 $C_1, C_2, \ldots, C_l$，其中：C_1 是 M 在 w 上的其实格局，C_l 是 M 的一个接受格局，且每个 C_i 都是 C_{i-1} 的合法结果，即符合 M 的规则。同理可以定义**拒绝计算历史**。

定义 6.6 **线性界限自动机（LBA）**是一种受到限制的图灵机，它不允许其读写头离开包含输入的带区域。

引理 6.7 设 M 是有 q 个状态和 g 个带符号的 LBA。对于长度为 n 的带子，M 汽油 qng^n 个不同的格局。

证明思路 每个状态、读写头位置、纸带内容。

定理 6.8 $A_{LBA} = \{< M, w > | M \text{ is a LBA that accepts } w\}$ 是可判定的。

证明思路 因为 LBA 的格局数有限，当模拟超过一定步数后，会出现循环，直接拒绝。这样保证了一定能停机。

定理 6.9 $E_{LBA} = \{< M > | M \text{ is a LBA and } L(M) = \emptyset\}$ 是不可判定的。

证明思路 构造 A_{TM} 的判定机 R。对于输入 $< M, w >$，构造一个接受 $< M, w >$ 的接受计算历史的 LBAB。如果 B 接受的语言为空，R 则拒绝，否则接受。

定理 6.10 $ALL_{CFG} = \{< G > | G \ is \ a \ CFG \ and \ L(G) = \Sigma^*\}$ 是不可判定的。

证明思路 对于 $< M, w >$，构造 CFG 派生所有计算历史（**除了接受计算历史**）。如果 CFG 能接受所有串，那么 M 就不能接受 w，否则 M 就能接受 w。

可以用定理 6.10 来证明 EQ_{CFG} **是不可判定的**。

6.2 一个简单的不可判定问题

一个关于串操作的不可判定问题，称为**波斯特对应问题**，或 **PCP**。

一族骨牌：

$$\{[\frac{b}{ca}], [\frac{a}{ab}], [\frac{ca}{a}], [\frac{abc}{c}]\}$$

上述骨牌的一个匹配：

$$[\frac{a}{ab}], [\frac{b}{ca}], [\frac{ca}{a}], [\frac{a}{ab}], [\frac{abc}{c}]$$

骨牌的上部和下部组成的字符串都是 $abcaaabc$。

定理 6.11 $PCP = \{< P > | P \ is \ an \ instance \ of \ the \ PCP \ with \ a \ match\}$ 是不可判定的。

证明思路 用反证法，利用 PCP 问题，构造判定 A_{TM} 问题的图灵机。

先将 PCP 问题转化为 $MPCP$ 问题（第一个骨牌是固定的），利用接受历史计算来构造骨牌的上部（上一个历史计算）和下部（下一个历史理算）。如果有一个匹配，就是有一个接受历史计算，示意图 14。

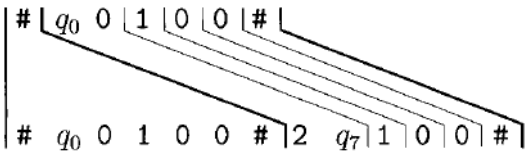

Figure 14: Example of $MPCP$

通过添加特殊符号，将 $MPCP$ 问题转化为 PCP 问题，示意图 15。

$$\left\{\left[\frac{\star t_1}{\star b_1 \star}\right], \left[\frac{\star t_1}{b_1 \star}\right], \left[\frac{\star t_2}{b_2 \star}\right], \left[\frac{\star t_3}{b_3 \star}\right], \cdots, \left[\frac{\star t_k}{b_k \star}\right], \left[\frac{*\diamond}{\diamond}\right]\right\}.$$

Figure 15: $MPCP$ to PCP

6.3 映射可归约性

用映射可归约性将问题 A 归约为问题 B。

6.3.1 可计算函数

定义 6.12 函数 $f : \Sigma^* \to \Sigma^*$ 是一个可计算函数，如果有某个图灵机 M，使得在每个输入 w 上，M 停机，且此时只有 $f(w)$ 出现在带上。

6.3.2 映射可归约的形式定义

定义 6.15 语言 A 是**映射可归约**到语言 B 的，如果存在可计算函数 $f:\Sigma^*\to\Sigma^*$ 使得对每个 w，

$$w\in A\Leftrightarrow f(w)\in B$$

记作 $A\leq_m B$。称函数 f 为 A 到 B 的归约。

定理 6.16 如果 $A\leq_m B$ 且 B 是可判定的，则 A 也是可判定的。

推论 6.17 如果 $A\leq_m B$ 且 A 不是可判定的，则 B 也不是可判定的。

定理 6.22 如果 $A\leq_m B$ 且 B 是图灵可识别的的，则 A 也是图灵可识别的的。

推论 6.23 如果 $A\leq_m B$ 且 A 不是图灵可识别的，则 B 也不是图灵可识别的。

定理 6.24 EQ_{TM} 既不是图灵可识别的，也不是不图灵可识别的。

证明思路 分为两步

1）证明 EQ_{TM} 不是图灵可识别的，只要证明 A_{TM} 可归约到 $\overline{EQ_{TM}}$ 即可。

2）证明 $\overline{EQ_{TM}}$ 不是图灵可识别的，只要证明 A_{TM} 可归约到 $\overline{\overline{EQ_{TM}}}$ 即可。

7 可计算理论的高级专题

7.1 递归定理

7.1.1 自引用

引理 7.1 存在可计算函数 $q:\Sigma^*\to\Sigma^*$，对任意串 w，q(w) 是图灵机 P_w 的描述，P_w 打印出 w，然后停机。

输出图灵机自己的编码，可以将这个编码分为两部分：$<SELF>=<AB>$。首先，令 $<A>=q(<B>)$，然后 A 输出 $<B>$，接着 B 根据 $<B>$ 输出 $q(<B>)$，最后拼接得到 $<AB>=<SELF>$。

定理 7.2 **递归定理** 设 T 是计算函数 $t:\Sigma^*\to\Sigma^*$ 的一个图灵机。则存在计算函数 $r:\Sigma^*\to\Sigma^*$ 的一个图灵机 R，使得对每个 w，有：

$$r(w)=t(<R>,w)$$

7.1.2 应用递归定理的术语

略

7.1.3 应用

定理 7.3 A_{TM} 是不可判定的。

证明思路 反证法。假设图灵机 H 判定 A_{TM}，构造图灵机 B：对于输入 w，通过递归定理构造 $<B>$，在输入 $<B,w>$ 上运行 H。如果 H 接受，则 B 拒绝；如果 H 拒绝，则 B 接受。

定义 7.4 如果 M 是一个图灵机，则 M 的描述 $<M>$ 的**长度**是描述 M 的串中所含符号的个数。如果没有与 M 等价的图灵机有更短的描述，则称 M 是**极小的**。令

$$MIN_{TM}=\{<M>|M\ is\ a\ minimal\ TM\}$$

定理 7.5 MIN_{TM} 是不可判定的。

证明思路 反证法。假设 TME 枚举 MIN_{TM}，构造 TMC：对于任意输入 w，通过递归定理得到自己的描述 $< C >$，然后通过枚举器 E 找到一个更长描述的 TMD，在 w 上模拟 D。如果 D 接受，C 就接受；如果 D 拒绝，C 就拒绝。

定理 7.6 设 $t : \Sigma^* \to \Sigma^*$ 是一个可计算函数，则存在一个图灵机 F，使得 $t(< F >)$ 描述一个与 F 等价的图灵机。

证明思路 构造图灵机 F：对于任意输入 w，通过递归定理获得自己的描述 $< F >$，然后计算 $t(< F >)$ 得到一个 TMG 的描述，在输入 w 上模拟 G。如果 G 接受，F 就接受；如果 G 拒绝，F 就拒绝。

7.2 逻辑理论的可判定性

对于模型 $\mathcal{M}$，令 $\mathcal{M}$ 的理论是这个模型语言中所有真聚集的集合，记作 $Th(\mathcal{M})$。

7.2.1 一个可判定的理论

定理 7.10 $Th(\mathcal{N}, +)$ 是可判定的。

7.2.2 一个不可判定的理论

定理 7.11 $Th(\mathcal{N}, +, \times)$ 是不可判定的。

引理 7.12 设 M 是一个图灵机，w 是一个串。从 M 和 w 能构造 $Th(\mathcal{N}, +, \times)$ 的语言中的公式 $\phi_{M,w}$，使得它只包含单个自由变元 x，且句子 $\exists x \phi_{M,w}$ 为真当且仅当 M 接受 w。

定理 7.13 $Th(\mathcal{N}, +, \times)$ 中可证命题的集合是图灵可识别的。

定理 7.14 $Th(\mathcal{N}, +, \times)$ 中有真命题是不可证的。

7.3 图灵可归约性

定义 7.16 语言 B 的一个**谕示**是一个能够报告某个串是否为 B 的成员的外部装置。一个**谕示图灵机**是一种修改过的图灵机，它有询问一个谕示的额外能力。记 M^B 为对语言 B 有谕示的谕示图灵机。

定义 7.19 语言 A **图灵可归纳**到 B，如果 A 相对于 B 是可判定的，基座 $A \leq_T B$。

定理 7.19 如果 $A \leq_T B$ 且 B 是可判定的，则 A 也是可判定的。

7.4 信息的定义

7.4.1 极小长度的描述

定义 7.20 设 x 是二进制数的串。x 的**极小描述**是最短的串 $< M, w >$，其中：TMM 在输入 w 上停机时，x 在带上。且如果有多个这样的串存在，则在其中选择字典序下的第一个串。记 x 的极小描述为 $d(x)$。x 的**描述复杂性** $K(x)$ 是

$$K(x) = |d(x)|$$

定理 7.21 $\exists c \forall x [K(x) \leq |x| + c]$

定理 7.22 $\exists c \forall x [K(xx) \leq K(x) + c]$

定理 7.23 $\exists c \forall x, y [K(xy) \leq 2K(x) + K(y) + c]$

7.4.2 定义的优化

定理 7.24 对任何描述语言 p，存在一个只与 p 有关的常量 c，使得：

$$\forall x[K(x) \leq K_p(x) + c]$$

7.4.3 不可压缩的串和随机性

定义 7.25 设 x 是一个串。如果

$$K(x) \leq |x| - c$$

则称 x 是 c—**可压缩的**。如果 x 不是 c—**可压缩的**，则称 x 是**不可压缩** c **的**。如果 x 是不可压缩 1 的，则称 x 是**不可压缩的**。

定理 7.26 对于每个长度，都存在不可压缩的串。

推论 7.27 至少有 $2^n - 2^{n-c+1} + 1$ 个长度为 n 的串是不可压缩 c 的。

定理 7.28 设 f 是一个对几乎所有串成立的性质，则对任意 $b > 0$，性质 f 只在有限多个不可压缩 b 的串上的值是 FALSE。

定理 7.29 存在常量 b，使得对每个串 x，x 的极小描述 $d(x)$ 都不可压缩 b。

8 时间复杂性

8.1 度量复杂性

定义 8.1 时间复杂度是一个函数：$f : \mathcal{N} \to \mathcal{N}$。

8.1.1 大 O 和小 o 记法

定义 8.2 设 f 和 g 是两个函数 $f, g : \mathcal{N} \to \mathcal{R}^+$。称 $f(n) = O(g(n))$，若存在正整数 c 和 n_0，使得对所有 $n \geq n_0$，有

$$f(n) \leq cg(n)$$

当 $f(n) = O(g(n))$ 时，称 $g(n)$ 是 $f(n)$ 的**上界**，更准确说是**渐进上界**。

例 8.3 设 $f_1(n) = 5n^3 + 2n^2 + 22n + 6$，则 $f_1(n) = O(n^3)$

例 8.4 写 $f(n) = O(\log n)$ 时，不必再指明基数。

定义 8.5 设 f 和 g 是两个函数 $f, g : \mathcal{N} \to \mathcal{R}^+$。称 $f(n) = o(g(n))$，若存在正整数 n_0，对于任何实数 $c > 0$，使得对所有 $n \geq n_0$，有

$$f(n) < cg(n)$$

或

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

8.1.2 分析算法

对于语言 $A = \{0^k 1^k | k \geq 0\}$，算法如图所示 16。该算法的时间复杂度为 $O(n^2)$

定义 8.7 令 $t : \mathcal{N} \to \mathcal{N}$ 是一个函数。定义时间复杂性类 $TIME(t(n))$ 为

$$TIME(t(n)) = \{L | L \text{ is a language that can be decided in } O(t(n))\}$$

对于语言 A 存在一个 $O(n \log n)$ 的算法：每次间隔一个删一个，直至删完。

当图灵机有两条纸带时，语言 A 能在 $O(n)$ 的时间内被判定。

M_1 = "对输入串 w：

1）扫描带，如果在 1 的右边发现 0，就拒绝。

2）如果 0 和 1 都在带上，就重复下一步。

3）扫描带，删除一个 0 和一个 1。

4）如果所有 1 都被删除以后还有 0，或者所有 0 都被删除以后还有 1，就拒绝。否则，如果在带上没有留下 0 和 1，就接受。"

Figure 16: Algorithm for language $A = \{0^k 1^k | k \geq 0\}$

8.1.3 模型间的复杂度关系

定理 8.8 设 $t(n)$ 是一个函数，$t(n) \geq n$。则每一个 $t(n)$ 时间的多带图灵机都和某一个 $O(t(n)^2)$ 时间的单带图灵机等价。

证明思路 模拟多带图灵机的每一步最多需要单带机的 $O(t(n))$ 步，因此总共需要时间为 $O(t^2(n))$ 步。

定义 8.9 N 的运行时间是函数 $f : \mathcal{N} \to \mathcal{N}$，$f(n)$ 是在任何长度为 n 的输入上，N 的所有计算分支的最大步数。

定理 8.10 设 $t(n)$ 是一个函数，$t(n) \geq n$。则每一个 $t(n)$ 时间的非确定型单带图灵机都和某一个 $2^{O(t(n)^2)}$ 时间的确定型单带图灵机等价。

证明思路 对递归模拟计算树。计算树的最大深度为 $O(t(n))$，最后一层有 b 个儿子，则 $O(t(n)b^{t(n)}) = 2^{O(t(n))}$

8.2 P 类

8.2.1 多项式时间

定义 8.11 P 是确定型单带图灵机在多项式时间内可判定的语言类。换句话说，

$$P = \cup_k (n^k)$$

8.2.2 P 中的问题举例

$PATH = \{< G, s, t > | G \ is \ a \ directed \ graph \ that \ has \ a \ directed \ path \ from \ s \ to \ t\}$

定理 8.12 $PATH \in P$。

$RELPRIME = \{< x, y > | x \ and \ y \ are \ relatively \ prime\}$

定理 8.13 $RELPRIME \in P$。

定理 8.14 每一个上下文无关语言都是 P 的成员。

证明思路 对于一个 CFL，存在对应的乔姆斯基范式 G。对于长度为 n 的串，需要 $2n - 1$ 来生成。使用动态规划算法进行判定，运行时间为 $O(n^3)$。

8.3 NP 类

定义 8.15 语言 A 的验证机是一个算法 V，这里

$$A = \{w | V \ accepts \ < w, c > \ for \ some \ string \ c\}$$

定义 8.16 NP 是具有多项式时间验证机的语言类。

定理 8.17 一个语言在 NP 中，当且仅当它能被某个非确定型多项式图灵机判定。

证明思路 非确定性地枚举证书进行验证。

定义 8.18

$$NTIME(t(n)) = \{L | L \ is \ a \ language \ decided \ by \ a \ O(t(n)) \ time \ nondeterministic \ Turing \ machine\}$$

推论 8.19 $NP = \cup_k NTIME(n^k)$

8.3.1 NP 中的问题举例

$$CLIQUE = \{< G, k > | G \ is \ an \ undirected \ graph \ with \ a \ k - clique\}$$

定理 8.20 $CLIQUE$ 属于 NP。

证明思路

思路一： 团即是证书。

思路二： 非确定性地选择一个点集，验证这个点集是否满足条件。

$$SUBSET - SUM = \{< S, t > | S = \{x_1, \ldots, x_k\} \ and \ for \ some \ \{y_1, \ldots, y_l\} \subseteq S, \ we \ have \ \Sigma y_i = t\} \quad (1)$$

定理 8.21 $SUBSET - SUM$ 属于 NP。

证明思路子集即是证书。

8.3.2 P 与 NP 问题

求解 NP 语言的已知的最好方式确定性地使用指数时间。换言之，可以证明。

$$NP \subseteq EXPTIME = \cup_k TIME(2^K)$$

8.4 NP 完全性

$$SAT = \{< \phi > | \phi \ is \ a \ satisfiable \ Boolean \ formula\}$$

定理 8.22 库克—列文定理

$$SAT \in P \ if \ and \ only \ if \ P = NP$$

8.4.1 多项式时间可归约性

定义 8.23 若存在多项式时间图灵机 M，使得在任何输入 w 上，M 停机是 $f(w)$ 恰好在带上，则称函数 $f : \Sigma^* \to \Sigma^*$ 为**多项式时间可计算函数**。

定义 8.24 语言 A 成为多项式时间映射可归约到语言 B，或简称为多项式时间可归约到 B，记为 $A \leq_P B$，若存在多项式时间可计算函数：$f : \Sigma^* \to \Sigma^*$，对于每一个 w，有

$$w \in A \Leftrightarrow f(w) \in B$$

函数 f 称为 A 到 B 的**多项式归约**。

定理 8.25 若 $A \leq_P B$ 且 $B \in P$，则 $A \in P$。

证明思路 A 多项式时间归约到 B，然后 B 在多项式时间内求解。

定理 8.26 $3SAT$ 多项式时间可规约到 $CLIQUE$。

证明思路 可以同时成真的结点连边，需要找一个大小为 k 的团（k 为合取式的数量），示意图 17

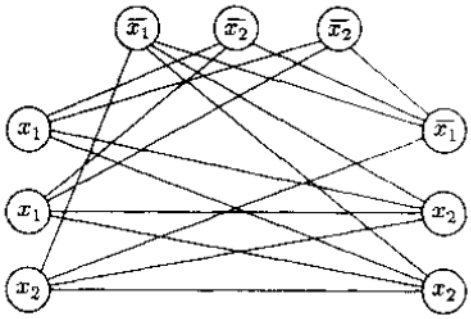

图 8-7 归约从 $\phi=(x_1\vee x_1\vee x_2)\wedge(x_1\vee x_2\vee x_2)\wedge(x_1\vee x_2\vee x_2)$ 生成的图

Figure 17: $3SAT$ to $CLIQUE$

8.4.2 NP 完全性的定义

定义 8.27 语言 B 是 NP **完全的**，若它满足两个条件：

1) B 属于 NP。

2) NP 中的每一个语言 A 多项式时间可归约到 B。

定理 8.28 若 B 是 NP 完全的，且 $B\in P$，则 $P=NP$。

定理 8.29 若 B 是 NP 完全的，且 $B\leq_P C$，C 属于 NP，则 C 是 NP 完全的。

8.4.3 库克—列文定理

定理 8.30 SAT 是 NP 完全的。

证明思路 首先，证明 SAT 是 NP 的（容易）。然后证明所有 NP 的问题都能规约到 SAT（极其困难，略）。

推论 8.32 $3SAT$ 是 NP 完全的。

要证明一个问题是 NP 完全的，只需证 $3SAT$ 能归约到这个问题，且这个问题是 NP 的即可。

8.5 几个 NP 完全问题

推论 8.33 $CLIQUE$ 是 NP 完全的。

8.5.1 顶点覆盖问题

$$VECTOR-COVER=\{<G,k>|G\ is\ an\ undirected\ graph\ that\ has\ a\ k-node\ vertex\ cover\}\quad(2)$$

定理 8.34 $VECTOR-COVER$ 是 NP 完全的。

证明思路 从 $3SAT$ 归约到 $VECTOR-COVER$，示意图 18。

8.5.2 哈密顿路径问题

定理 8.35 $HAMPATH$ 是 NP 完全的。

证明思路 从 $3SAT$ 归约到 $HAMPATH$，示意图 19。

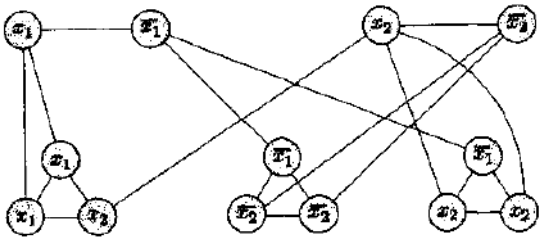

图 8-11　归约从 $\phi=(x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$ 产生的图

Figure 18: $3SAT$ to $VECTOR-COVER$

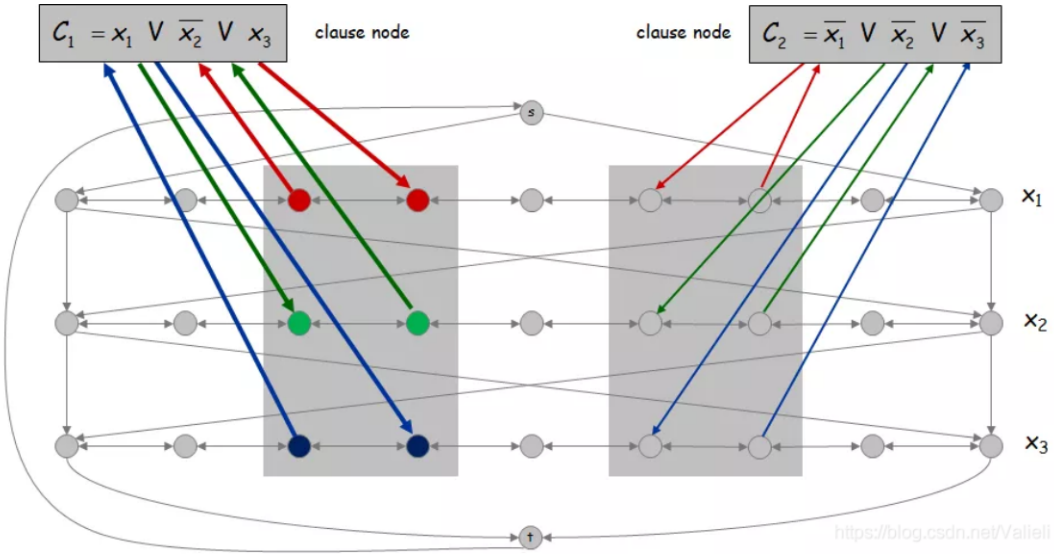

Figure 19: $3SAT$ to $HAMPATH$

定理 8.36　$UHAMPATH$（无向图的哈密顿回路）是 NP 完全的。

证明思路　从 $HAMPATH$ 归约到 $UHAMPATH$。有向图的一个点 u，拆成无向图的三个点 u^{in}, u^{mid}, u^{out}。

8.5.3　子集和问题

定理 8.34　$SUBSET-SUM$ 是 NP 完全的，示意图 20。

证明思路　从 $3SAT$ 归约到 $SUBSET-SUM$

8.5.4　补充

补充定理　最小旅行商问题 TSP 是 NP 完全的。

证明思路　从哈密顿回路问题归约到 TSP 问题，每条路径的长度都为 1。

补充定理　$k-$ 独立集 是 NP 完全的。

证明思路　从 $k-CLIQUE$ 归约到 $k-$ 独立集。

更多 NP 完全问题的归约关系图 21

9　空间复杂性

定义 9.1　空间复杂度：$O(f(n))$。其中，$f:\mathcal{N}\to\mathcal{N}$。

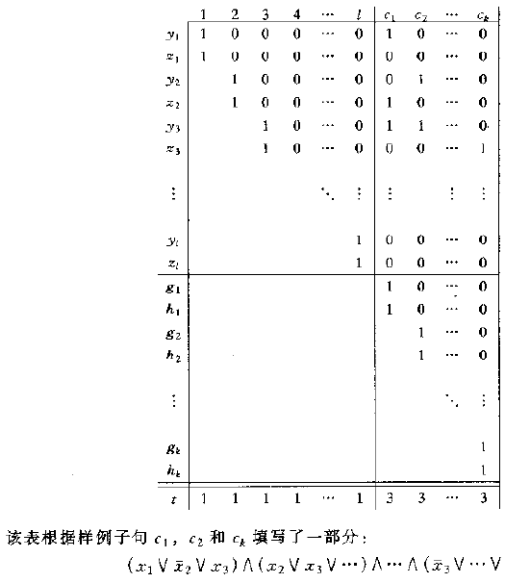

Figure 20: $3SAT$ to $SUBSET - SUM$

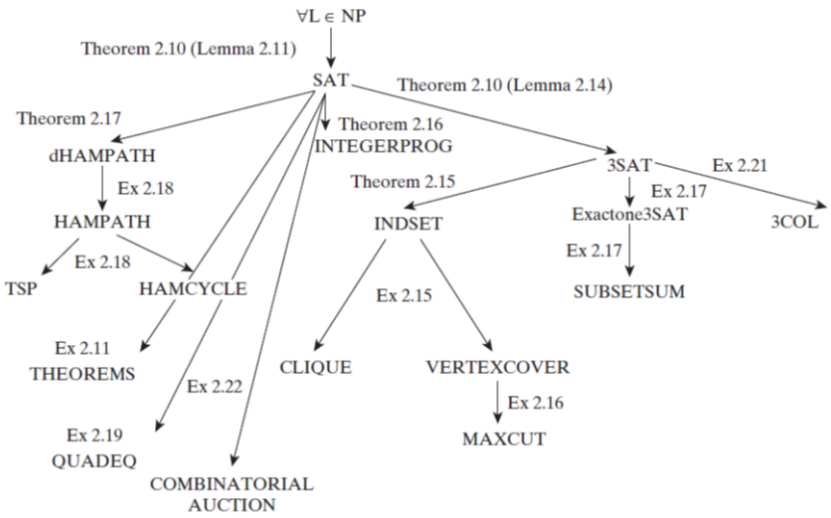

Figure 21: More NP Problems

定义 9.2 空间复杂性类：$SPACE(f(n))$ 和 $NSPACE(f(n))$。前者指确定型图灵机，后者指非确定型图灵机。

例 9.3 SAT 问题可以在线性空间内运行。

例 9.4 对于问题

$$ALL_{NFA} = \{< A > | A \ is \ NFA \ and \ L(A) = \Sigma^*\}$$

该问题在非确定的空间 $O(n)$ 内运行。

9.1 萨维奇定理

定理 9.5 萨维奇定理

对于任何函数：$f : \mathcal{N} \to \mathcal{N}$，其中 $f(n) \geq n$，

$$NSPACE(f(n)) \subseteq SPACE(f(n)^2)$$

证明思路 使用分治的方式，总空间复杂度为 $O(\log |\Sigma|^{f(n)} \times f(n)) = O(f(n)^2)$。

9.2 $PSPACE$ 类

定义 9.6 $PSPACE$ 是在确定型图灵机上、在多项式**空间**内可判定的语言类。换言之，

$$PSPACE = \cup_k SPACE(n^k)$$

同理，可以定义 $NPSPACE$。

需要注意的是，$PSPACE$ **对于一个语言的补集是封闭的**，由于空间有上限，所以格局有上限（设为 q），计算次数超过 2^q 即出现循环，可以判定此时不会停机。

通过**萨维奇定理**，可以推到得到 $NPSPACE = PSPACE$。

总结所定义的复杂性类之间的关系：

$$P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq EXPTIME$$

9.3 $PSPACE$ 完全性

定义 9.7 语言 B 是 $PSPACE$ 完全的，若它满足两个条件：

1) B 属于 $PSPACE$。

2) $PSPACE$ 中的每一个语言 A 多项式时间可归约到 B。

若只满足条件 2，则称它为 $PSPACE$ **难的**。

9.3.1 问题 $TQBF$

TQBF 问题就是要判定一个全量词化的布尔公式是真还是假。定义语言

$$TQBF = \{< \phi > | \phi \ is \ a \ true \ fully \ quantified \ Boolean \ formula\}$$

定理 9.8 $TQBF$ 是 $PSPACE$ 完全的

证明思路

条件 1：构造算法，说明空间复杂度为多项式的。

条件 2：利用分治的思想，将一个问题拆成两部分，使用 $\exists$（存在一种拆法）和 $\forall$（对于所有的递归步骤）两种量词进行构造：

$$\phi_{c_1,c_2,t} = \exists m_1 \forall (c_3, c_4) \in \{(c_1, m1), (m_1, c_2\}[\phi_{c_3,c_4,\lceil \frac{t}{2} \rceil}]$$

9.3.2 博弈的必胜策略

公式博弈游戏：玩家按照顺序给 $x_1, \ldots, x_n$ 赋值。最后公式真值为 TRUE 则玩家 E 胜利，否则玩家 A 胜利。

例 9.9 公式博弈：$\exists x_1 \forall x_2 \exists x_3 [(x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\overline{x_2} \vee \overline{x_3})]$

公式博弈的语言如下所示：

$$FORMULA - GAME = \{< \phi > | Player \ E \ has \ a \ winning \ strategy \ in \ the \ formula \ game \ associated \ with \ \phi\} \quad (3)$$

定理 9.10 $FORMULA - GAME$ 是 $PSPACE$ 完全的

证明思路 由于公式中存在 $\exists$ 和 $\forall$ 量词，所以 $FORMULA - GAME = TQBF$。

9.3.3 广义地理学

广义地理学游戏：对于一个图，玩家轮流在图的结点上填写地理名词，对于有向边 $A \to B$，要求 B 以 A 的结尾字母开头。

广义地理学的语言如下所示：

$$GG = \{< G, b > | Player\ I\ has\ a\ winning\ strategy\ for\ the\ generalized\ geography\ game\ played\ on\ graph\ G\ starting\ at\ node\ b\} \quad (4)$$

定理 9.11 GG 是 $PSPACE$ 完全的

证明思路

条件 1：构造算法，说明空间复杂度为多项式的。

条件 2：将 $FORMULA - GAME$ 转化为 GG，示意图22。玩家 E 想使公式为真，玩家 A 想使公式为假。最后一个钻石是 E 来选择，那么 A 走完这个钻石，E 走到结点 c 上。此时，如果想要公式成假，那么只要有一个子式为假即可，即这个子式中的任意一个结点之前都没走过。那么 E 和 A 再各走一步之后，E 无路可走，此时原式为假，A 获胜。否则，E 走那条连接之前走过的结点的边，A 无路可走，此时原式为真，E 获胜。

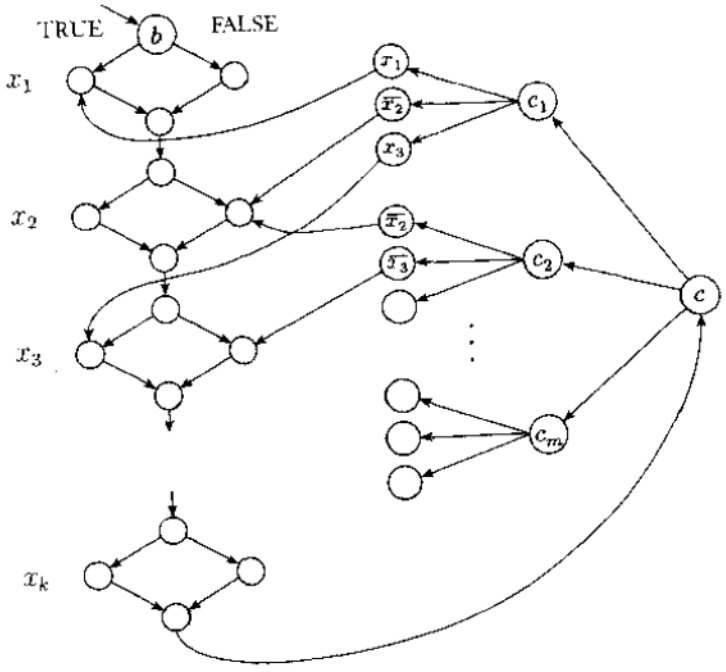

图 9-5 模拟公式游戏的地理学游戏的全部结构，其中

$\phi = \exists x_1 \forall x_2 \cdots Q x_k [(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\ \ )]$

Figure 22: $FORMULA - GAME$ to GG

9.4 L 类和 NL 类

定义 9.12 L 和 NL

L 是确定型图灵机在对数空间内可判定的语言类，即 $L = SPACE(\log n)$。

NL 是确定型图灵机在对数空间内可判定的语言类，即 $L = NSPACE(\log n)$。

需要注意的是，在亚线性空间界限下，图灵机有两条纸带，一条只读，一条可写，模仿内存和外存。

例 9.13 语言 $A = \{0^k 1^k | k \geq 0\}$ 是 L 的成员。

例 9.14 语言 $PATH = \{< G, s, t > | G\ is\ a\ directed\ graph\ that\ has\ a\ directed\ path\ from\ s\ to\ t\}$ 是 NL 的成员

定义 9.15 若 M 是一个有单独的只读输入带的图灵机，ω 是输入，则 M **在** ω **上的格局**是状态、工作带和两个读写头位置的一种设置。输入 ω 不最为 M 在 ω 上的格局的一部分。

9.5 NL 完全性

定义 9.16

对数空间转换器是有一条只读输入带、一条只写输出带和一条读写工作带的图灵机。

对数空间可计算函数： $f : \Sigma^* \to \Sigma^*$

如果语言 A 通过对数空间可计算函数 f 映射可规约到语言 B，则称 A **对数空间可规约到** B，记为 $A \leq_L B$。

定义 9.17 语言 B 是 NL **完全的**，如果

1）$B \in NL$，并且

2）NL 中的每个 A 对数空间可规约到 B。

定理 9.18 若 $A \leq_L B$ 且 $B \in L$，则 $A \in L$。

推论 9.19 若有一个 NL 完全语言属于 L，则 $L = NL$。

定理 9.20 $PATH$ 是 NL 完全的。

证明思路 把字符串 ω 映射成一个图，图中结点对应的是 NTM 在输入 ω 上的格局。

推论 9.21 $NL \subseteq P$

证明思路 一个结论：消耗空间为 $O(f(n))$ 的图灵机在 $n2^{O(f(n))}$ 下运行。

9.6 NL 等于 $coNL$

$L \in coNL$ 当且仅当 $\overline{L} \in NL$

定理 9.22 $NL = coNL$

证明思路 证明 $\overline{PATH} \in NL$。由于 $PATH$ 是 NL 完全的，所以当语言 A 可以归约到 $PATH$，那么 $\overline{A}$ 可以归约到 $\overline{PATH}$，因此 $A \in coNL$，即可证明 $NL = coNL$。

当前已知的几个复杂性类之间相互关系的认识总结如下：

$$L \subseteq NL = coNL \subseteq P \subseteq PSPACE$$