Chapter 9

不可判定性

9.1 问题

非形式的, 我们使用问题来表示诸如“一个给定的 CFG 是否是歧义？”这样的询问. 那么一个具体的 CFG 就是一个问题的实例, 一般来说, 问题的一个实例就是一个自变量表, 每个自变量都表示问题的一个参数. 用某个字母表, 可以将问题的实例进行编码, 我们就能将是否存在解决某一问题的算法这一问题, 转化为一个特定的语言是否是递归的问题.

可判定问题和不可判定问题

一个问题, 如果它的语言是递归的, 就称为*可判定* (*decidable*) 的问题, 否则, 该问题是*不可判定的* (*undecidable*). 也就是说, 对于不可判定的问题, 不存在能够保证停机的图灵机, 识别该问题的语言, 或者说不存在解决该问题的算法. 下面就给出两个不可判定的问题.

9.2 非递归可枚举的语言

我们将使用对角线法证明一个特定的问题是不可判定的, 这个问题是“图灵机 M 接受输入 w 吗?”. 这里的 M 和 w 都是该问题参数, 并且限制 w 是 $\{0,1\}$ 上的串而 M 是仅接受 $\{0,1\}$ 上的串的图灵机. 这个受限的问题是不可判定的, 那么较一般的问题也肯定是不可判定的. 首先我们需要将问题实例编码为字符串, 将问题转化为语言.

9.2.1 第 i 个串 w_i

将全部 $(0+1)^*$ 中的串按长度和字典序排序, 那么第 i 个串就是 w_i, 即

$$\text{binary}(i) = 1w_i$$

比如:

i	1	2	3	4	5	6	7	8	9	$\cdots$
binary(i)	1ε	10	11	100	101	110	111	1000	1001	$\cdots$
w_i	ε	0	1	00	01	10	11	000	001	$\cdots$

9.2.2 图灵机编码与第 i 个图灵机

将字母表为 $\{0,1\}$ 的任意 TM 用二进制串进行编码. 设 TM M 为

$$M = (Q,\ \{0,1\},\ \Gamma,\ \delta,\ q_1,\ B,\ F)$$

再为状态, 栈符号和移动方向指派整数编码:

(1) $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$, 指派开始状态为 q_1, 终态为 q_2, 且终态 (一定停机) 只需一个;

(2) $\Gamma = \{X_1, X_2, \cdots, X_{|\Gamma|}\}$, 这里总有 $X_1 = 0$, $X_2 = 1$, $X_3 = B$;

(3) 方向 L 为 D_1, 方向 R 为 D_2.

那么任意的转移函数

$$\delta(q_i, X_j) = (q_k, X_l, D_m)$$

可用一条编码 (C) 表示:

$$0^i10^j10^k10^l10^m$$

而 M 全部的 n 个转移动作的编码合在一起, 就可以作为整个 TM 的编码:

$$C_1\,11\,C_2\,11\,\cdots\,11\,C_{n-1}\,11\,C_n.$$

第 i 个图灵机

那么如果 TM M 编码为第 i 个串 w_i, 则称 M 是“第 i 个图灵机”, 记为 M_i. 而任意的串 w_i 也都可以看作 TM 编码, 如果不合法, 则可以认为是没有任何动作的 TM, 只有一个状态, 在任何输入上立即停机并拒绝输入, 其接受的语言是 $\emptyset$.

有序对 (M, w)

一个 TM M 和一个输入串 w, 组成的有序对 (M, w), 可以表示为一个串即

$$M111w$$

这里的 M 不含任何连续 3 个的 1, 所以可以将 M 和 w 区分开.

9.2.3 对角化语言 L_d

定义对角化语言 L_d:

$$L_d = \{w_i \mid w_i \notin \mathbf{L}(M_i), i \geq 1\}$$

即, 使第 i 个串 w_i 不属于第 i 个图灵机的语言 $\mathbf{L}(M_i)$ 的所有 w_i 的集合.

L_d 可以由下图的矩阵给出. 矩阵的上边顺序排列每个 w_j, 矩阵的左边顺序排列每个 M_i, 如果 M_i 接受 w_j, 则矩阵中对应的位置为 1, 否则为 0. 矩阵的每行可以看做语言 $\mathbf{L}(M_i)$ 的特征向量 (*characteristic vector*), 处于对角线位置的值, 刚好表示 M_i 是否接受 w_i. 那么只需将对角线的值取补 (*complementary*), 就是 L_d 的特征向量, 即给出了语言 L_d. 这里的对角化技术使 L_d 的特征向量与表中每行都在某列不同, 因此也不可能是任何图灵机 (的语言) 的特征向量.

$$\begin{array}{cc|ccccccc}
 & & & w_j \longrightarrow & & & & & \\
 & & 1 & 2 & 3 & 4 & 5 & 6 & \cdots \\
\hline
 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & \cdots \\
 & 2 & 1 & 0 & 0 & 1 & 0 & 0 & \cdots \\
M_i & 3 & 0 & 1 & 1 & 0 & 0 & 1 & \cdots \\
\downarrow & 4 & 0 & 0 & 1 & 1 & 1 & 1 & \cdots \\
 & 5 & 1 & 1 & 0 & 0 & 0 & 1 & \cdots \\
 & 6 & 0 & 1 & 0 & 1 & 1 & 1 & \cdots \\
 & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

9.2.4 L_d 不是递归可枚举的

定理 1. *L_d 不是递归可枚举语言, 即不存在 TM 接受 L_d.*

证明. 假设存在 TM M 使 $\mathbf{L}(M) = L_d$, 由于 L_d 是 $\{0,1\}$ 上的语言, 因此 M 可以被编码成二进制串, 不妨设为 w_i, 则 $M = M_i$. 那么 w_i 是否在 L_d 中呢?

(1) 若 $w_i \in L_d$, 根据假设, 则 $w_i \in \mathbf{L}(M_i)$; 而如果 $w_i \in \mathbf{L}(M_i)$, 根据 L_d 定义，则 $w_i \notin L_d$;

(2) 若 $w_i \notin L_d$, 根据假设, 则 $w_i \notin \mathbf{L}(M_i)$; 而如果 $w_i \notin \mathbf{L}(M_i)$, 根据 L_d 定义，则 $w_i \in L_d$.

因此假设不成立, 不存在 TM 能够接受 L_d. □

因此图灵机所能接受的语言也不是任意的, 至少有一个语言 L_d 是无法被图灵机接受的.

9.3 递归可枚举但非递归的语言

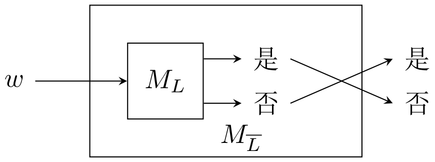
语言 L_d 是不能被图灵机接受的, 那么肯定不存在算法解决语言为 L_d 的问题, 显然这样的问题都是不可判定的. 但是即使存在图灵机, 却无法保证停机, 对于问题的解决也没有实质的贡献, 因此将 “问题” 区分为可判定的和不可判定的, 要比区分问题是否具有 TM 更有意义. 所以这里给出一个语言的实例 L_u, 属于递归可枚举语言但不属于递归语言.

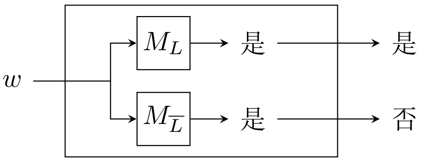
9.3.1 递归语言的封闭性

这里我们只给出递归语言封闭性的两个定理.

定理 2. *如果 L 是递归的, 那么 $\overline{L}$ 也是递归的.*

定理 3. *如果语言 L 和它的补 $\overline{L}$ 都是递归可枚举的, 那么 L 是递归的.*





9.3.2 通用图灵机

如果 TM M 接受串 w, 那么由有序对 (M, w) 构成的语言, 称为**通用语言** (*universal language*), 记为 L_u.

定理 4. *L_u 是递归可枚举的, 但不是递归的.*

证明. 识别 L_u 的图灵机 U 可以这样构造, 利用多带技术让 U 模拟输入 (M, w) 中 M 识别 w 的动作, U 使用 3 条带, 第 1 带放置 (M, w), 即存储 M 动作的定义, 第 2 带模拟 M 的带, 第 3 带存储 M 的状态. 因此 L_u 是递归可枚举的.

利用反证法证明 L_u 不是递归的. 假设 L_u 是递归的, 则存在识别 L_u 的算法 A, 那么可以这样得到识别 L_d 的算法 B: 将输入 $w = w_i$ 转换为 (M_i, w_i), 并交给 A 判断; 当 A 接受 (M_i, w_i), 表示 $w_i \in \mathbf{L}(M_i)$, 则 B 拒绝; 当 A 拒绝 (M_i, w_i), 表示 $w_i \notin \mathbf{L}(M_i)$, 则 B 接受. 而由于 L_d 不是递归的, 所以这样的算法 B 不可能存在, 所以算法 A 并不存在, 所以 L_u 不可能是递归的. □

因为识别 L_u 的图灵机 U, 可以模拟任意图灵机, 因此也称为**通用图灵机** (*univerasl Turing machine*). 正是因为通用图灵机的概念, 帮助冯 • 诺伊曼产生了通用电子计算机体系的设计思想, 这也可以看出抽象理论的先期发展可以对实际问题有很大帮助.

9.4 语言间的关系