Chapter 9

不可判定性

9.1 不可判定性

非形式的, 我们使用问题来表示诸如“一个给定的 CFG 是否歧义? ”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.

不可判定问题

- 递归可枚举语言 — 图灵机所识别
- 递归语言 — 保证停机的图灵机所识别

定义. *一个问题, 如果它的语言是递归的, 就称为可判定 (decidable) 问题, 否则称为不可判定 (undecidable) 问题.*

不可判定的问题

- 不存在保证停机的图灵机识别该问题的语言
- 不存在解决该问题的算法

9.2 非递归可枚举的语言

可判定吗?

“图灵机 M 接受输入 w 吗? ”

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

定义. 将全部 $(\mathbf{0}+\mathbf{1})^*$ 中的字符串按长度和字典序排序, 那么第 i 个串就是w_i. 且刚好有

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

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

图灵机编码

将 $\Sigma = \{0, 1\}$ 上的全部图灵机, 用二进制字符串编码

$$M = (Q,\, \Sigma,\, \Gamma,\, \delta,\, q_1,\, B,\, F)$$

1. $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$, 开始状态为 q_1, 终态为 q_2 且停机;
2. $\Gamma = \{X_1, X_2, \cdots, X_{|\Gamma|}\}$, 总有 $X_1 = 0$, $X_2 = 1$, $X_3 = B$;
3. 设带头移动方向 $D_1 = L,\ D_2 = R$;
4. 任意的转移 $\delta(q_i, X_j) = (q_k, X_l, D_m)$ 可用一条编码 ($Code$) 表示为

$$C = 0^i 1 0^j 1 0^k 1 0^l 1 0^m;$$

5. 则全部 n 个转移的编码合并在一起, 作为图灵机 M 的编码:

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

第 i 个图灵机 M_i

定义. 如果图灵机 M 的编码为第 i 个串 w_i, 则称 M 是第 i 个图灵机 M_i.

- 任意图灵机 M 都对应一个字符串 w
- 任意的字符串 w 都可以看作图灵机的编码
- 如果编码不合法, 将其看作接受 $\varnothing$ 且立即停机的图灵机

9.2.3 对角化语言 L_d

非递归可枚举的语言

定义. 使第 i 个串 w_i 不属于第 i 个图灵机 M_i 的语言 $\mathbf{L}(M_i)$ 的所有 w_i 的集合, 称为对角化语言 L_d, 即

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

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

对角化语言 L_d 可以由上图的矩阵给出. 矩阵的每列上顺序排列每个字符串 w_j, 矩阵的每行前顺序排列每个图灵机 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, 刚好表示图灵机 M_i 是否接受第 i 个串 w_i. 那么只需将对角线的值取补, 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处不同, 因此也不可能是任何图灵机 (的语言) 的特征向量.

定理 43. *L_d 不是递归可枚举语言, 即不存在图灵机接受 L_d.*

证明: 反证法.

假设存在识别 L_d 的图灵机 M, 那么 M 也可被编码, 不妨设第 i 个图灵机 $M_i = M$, 即 $\mathbf{L}(M_i) = L_d$.

那么, 考虑第 i 个串 w_i 是否会被 M_i 识别:

1. 如果 $w_i \in \mathbf{L}(M_i) = L_d$, 那么由 L_d 的定义, 又有 $w_i \notin \mathbf{L}(M_i)$;
2. 如果 $w_i \notin \mathbf{L}(M_i)$, 那么由 L_d 的定义, 又有 $w_i \in L_d = \mathbf{L}(M_i)$.

无论如何都会矛盾, 因此假设不成立, 不存在接受 L_d 的图灵机. □

9.3 递归可枚举但非递归的语言

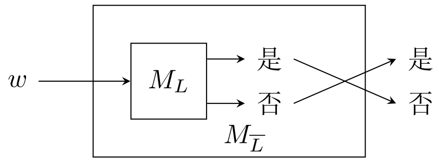
对角化语言 L_d 不存在图灵机, 那么肯定不存在算法解决语言为 L_d 的问题, 显然这样的问题都是不可判定的. 但即使存在图灵机, 如果无法保证停机, 对于问题的解决也没有实质的贡献, 因此将“问题”区分为可判定的和不可判定的, 要比区分问题是否具有图灵机更有意义. 这里将给出一个语言的实例—通用语言 L_u, 属于递归可枚举语言但不属于递归语言.

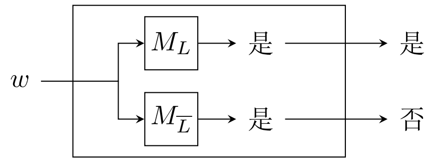
9.3.1 递归语言的封闭性

首先, 给出递归语言的两个封闭性定理. 递归语言中"递归"的含义是, 可以通过递归函数来解决, 而递归函数总会结束.

定理 44. *如果 L 是递归的, 那么 $\overline{L}$ 也是递归的.*

定理 45. *如果语言 L 和 $\overline{L}$ 都是递归可枚举的, 那么 L 是递归的.*





9.3.2 通用语言与通用图灵机

定义 (有序对 (M, w))**.** *一个图灵机 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.

定义. *如果图灵机 M 接受串 w, 那么由 $M111w$ 表示的有序对 (M, w) 构成的语言L_u, 称为通用语言 (universal language)*

$$L_u = \{M111w \mid w \in \mathbf{L}(M)\}.$$

定义. *构造图灵机 U, 当输入 $M111w$ 时, 利用多带技术模拟 M 处理串 w 的过程. 因为 M 接受 w 时会停机, 因此 U 可以识别 L_u, 图灵机 U 称为通用图灵机 (univerasl Turing machine).*

递归可枚举但非递归的语言

定理 46. *通用语言 L_u 是递归可枚举的, 但不是递归的.*

证明: L_u 是递归可枚举的. 用反证法证明 L_u 不是递归的.

通用图灵机 U 使用 3 条带分别: (1) 装载 M 的编码; (2) 放置 w, 模拟 M 的带; (3) 存储 M 的状态.

假设存在算法 A 识别 L_u, 那么可如下得到识别对角化语言 L_d 的算法 B.

将 B 的输入 $w = w_i$ 转换为 M_i111w_i 交给 A 判断:

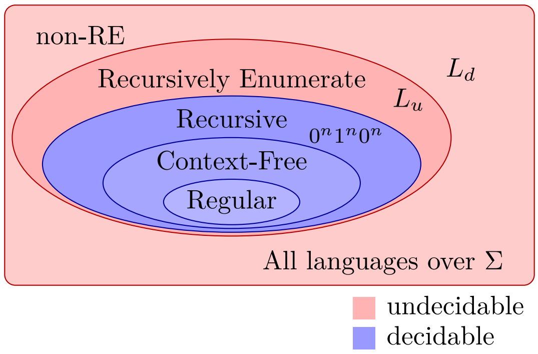
- 当 A 接受, 表示 $w_i \in \mathbf{L}(M_i)$, 则 B 拒绝;
- 当 A 拒绝, 表示 $w_i \notin \mathbf{L}(M_i)$, 则 B 接受.

而由于 L_d 不是递归的, 所以 B 不可能存在, 所以 L_u 不可能是递归的. □

通用图灵机的重要意义

- 识别 L_u 的通用图灵机 U, 可以模拟任意图灵机
- 冯 • 诺伊曼通用数字电子计算机体系结构设计思想的灵感来源
- 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


non-RE
Recursively Enumerate
L_d
L_u
Recursive
$0^n1^n0^n$
Context-Free
Regular
All languages over Σ
undecidable
decidable