Chapter 8

图灵机

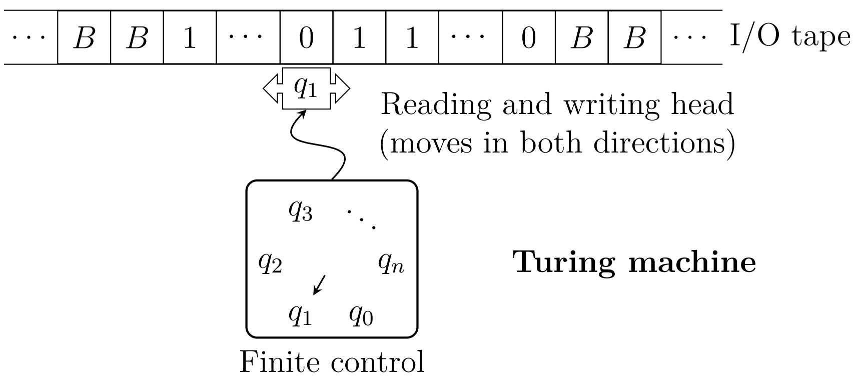
8.1 图灵机

在研究可计算的整数函数的过程中, 数理逻辑学家给出了几种不同的定义. 20 世纪 30 年代, 美国的丘奇 (A. Church) 提出了 λ-演算, 哥德尔和克林 (S. Kleene) 给出了递归演算系统, 并由此定义了递归函数类. 同时, 在英国的图灵 (A. M. Turing) 也在研究可计算的本质. 图灵分析了人类进行算法演算的过程, 并定义了用来模拟这一过程的机器, 即图灵机. 当人们在用纸和笔进行计算时, 要在纸的一定部位写上或擦去所用的符号, 人的眼光在纸上移动以进行计算, 并需要根据一定的规则进行每一步的计算. 图灵最初提出的机器并不是用来进行识别语言的, 而是进行函数计算用的. 尽管这个机器很简单, 但是图灵断言它在功能上等价于一个进行数学运算的人.

图灵机既可以作为*语言的识别器*, 也可以作为*整数函数的计算器*. 它的运算过程真正体现了机械而有效的特点, 虽然与其等价的递归演算系统和 λ-演算也都反映了计算的本质, 但和图灵机相比, 似乎都需要更多的智慧. 也是因为图灵和丘奇的工作, 算法的概念才首次被形式化的定义.

这里我们学习可计算性理论的一些基本知识. 可计算性理论是由递归函数的产生而开始发展的, 在数理逻辑和计算机科学中, 递归函数是一类从自然数到自然数的函数, 它在某种直觉意义上是“可计算的”, 因此递归函数也是“算法”这个直观概念的精确定义.

8.1.1 形式定义



图灵机具有一个有穷控制器, 一条两端无穷的输入输出带和一个带头. 带划分为单元格, 每个单元格可以放置一个符号, 带头每次根据当前状态和带头处单元格的符号内容, 根据转移规则选择下一个动作, 每个动作都包括下一个状态, 修改带头处单元格的符号以及带头向左或

向右移动一个单元格.

图灵机的形式定义

定义. 图灵机 *(TM, Turing Machine)* M 为七元组

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

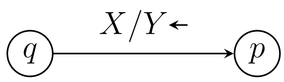
1. Q: 有穷状态集;
2. Σ: 有穷输入符号集;
3. Γ: 有穷带符号集, 且总有 $\Sigma \subset \Gamma$;
4. δ: $Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ 转移函数;
5. $q_0 \in Q$: 开始状态;
6. $B \in \Gamma - \Sigma$: 空格符号;
7. $F \subseteq Q$: 终态集或接受状态集.

图灵机的动作及状态转移图

图灵机的有穷控制器处于状态 q, 读写头所在单元格为符号 X, 动作为

$$\delta(q, X) = (p, Y, L),$$

表示当前状态转到 p, 读写头处当前的单元格改为 Y, 然后读写头向左移动一个单元格.



因为每个动作都是确定的, 因此是“确定的图灵机”.

8.1.2 瞬时描述

瞬时描述

定义. 图灵机虽有无穷长的带, 但是在有限步移动之后, 带上的非空内容总是有限的. 因此用带上最左边到最右边全部的非空符号、当前状态和读写头位置, 同时定义瞬时描述 *(ID, Instantaneous Description)* 或格局 *(Configuration)* 为

$$X_1X_2 \cdots X_{i-1}qX_iX_{i+1} \cdots X_n$$

其中的信息包括:

1. 图灵机的当前状态 q;
2. 带头在左起第 i 个非空格符号上;
3. $X_1X_2 \cdots X_n$ 是最左到最右非空格内容, 虽然中间可能有空格符.

转移符号

定义. 图灵机 M 中, 如果 $\delta(q, X_i) = (p, Y, L)$, 定义 ID 转移为

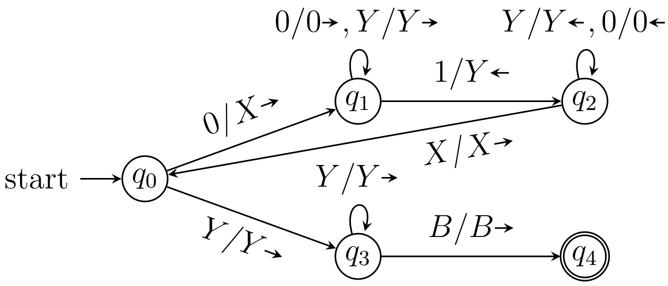
$$X_1 \dots X_{i-1} q X_i \dots X_n \vdash_M X_1 \dots X_{i-2} p X_{i-1} Y X_{i+1} \dots X_n$$

如果 $\delta(q, X_i) = (p, Y, R)$ 那么

$$X_1 \dots X_{i-1} q X_i \dots X_n \vdash_M X_1 \dots X_{i-1} Y p X_{i+1} \dots X_n$$

若 M 已知, 记为 $\vdash$ 和 $\vdash^*$.

例 1. 设计识别 $\{0^n 1^n \mid n \geq 1\}$ 的图灵机.



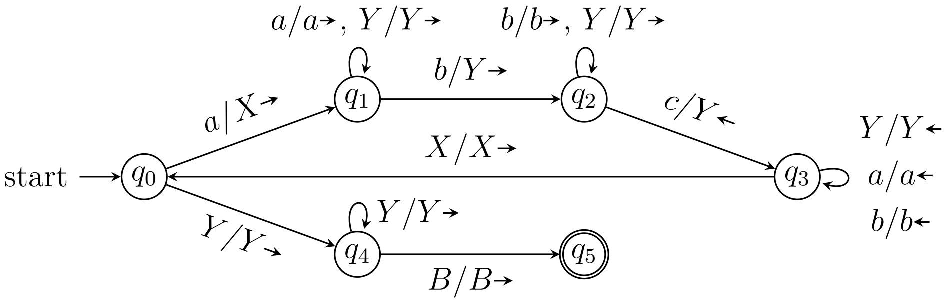
$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, B, \{q_4\})$$

δ	0	1	X	Y	B
q_0	(q_1, X, R)	$-$	$-$	(q_3, Y, R)	$-$
q_1	$(q_1, 0, R)$	(q_2, Y, L)	$-$	(q_1, Y, R)	$-$
q_2	$(q_2, 0, L)$	$-$	(q_0, X, R)	(q_2, Y, L)	$-$
q_3	$-$	$-$	$-$	(q_3, Y, R)	(q_4, B, R)
q_4	$-$	$-$	$-$	$-$	$-$

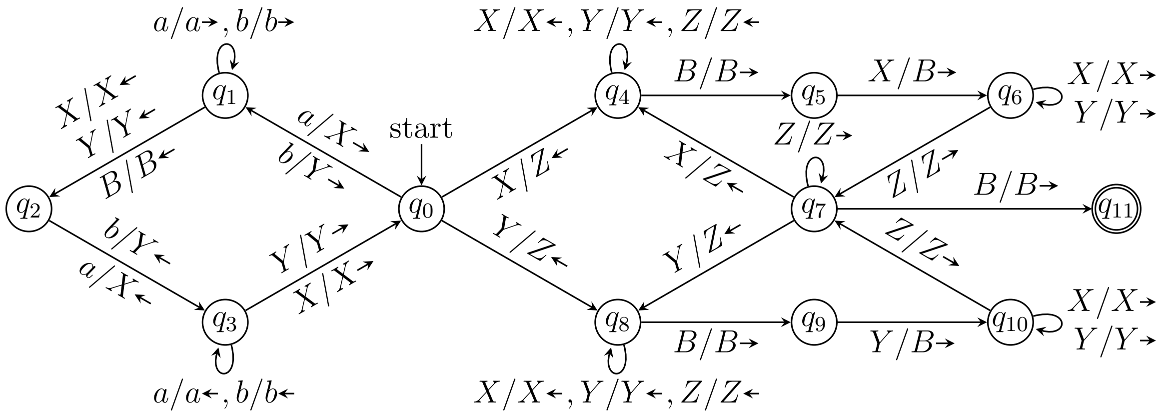
接受 0011 的 ID 序列为

$$\begin{aligned} q_0 0011 &\vdash X q_1 011 & &\vdash X 0 q_1 11 & &\vdash X q_2 0 Y 1 \\ &\vdash q_2 X 0 Y 1 & &\vdash X q_0 0 Y 1 & &\vdash X X q_1 Y 1 \\ &\vdash X X Y q_1 1 & &\vdash X X q_2 Y Y & &\vdash X q_2 X Y Y \\ &\vdash X X q_0 Y Y & &\vdash X X Y q_3 Y & &\vdash X X Y Y q_3 B \\ &\vdash X X Y Y B q_4 B \end{aligned}$$

例 2. 设计接受 $L = \{a^n b^n c^n \mid n \geq 1\}$ 的图灵机.



例 3. 设计接受 $L = \{ww \mid w \in \{a, b\}^+\}$ 的图灵机.



思考

1. DFA 和 TM 的主要区别?
2. 计算机, 究竟是 TM 还是 DFA?

8.1.3 语言与停机

图灵机的语言

定义. 如果 $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ 是一个图灵机, 则 M 接受的语言为

$$\mathbf{L}(M) = \{w \mid w \in \Sigma^*,\ q_0 w \vdash^* \alpha p \beta,\ p \in F,\ \alpha, \beta \in \Gamma^*\}.$$

输入串 w 放在输入带上, M 处于 q_0, 带头位于输入串的第一个字符上, 输入串最终会导致 M 进入某个终结状态.

定义. 如果语言 L 是某个图灵机 M 的语言, 即 $L = \mathbf{L}(M)$, 则称 L 是递归可枚举语言.

一般假定当输入串 w 被接受时 M 总会停机 (*halt*), 即没有下一个动作的定义. 而对于不接受的输入, 图灵机可能永远不停止. 我们永远也不会知道, 到底是因为运行的时间不够长而没有接受呢, 还是根本就不会停机. 能够被图灵机接受的语言类, 称为递归可枚举的 (*recursively enumerable*, RE). “可枚举”的意思是这些语言中的串可以被某个图灵机枚举出来. 这个语言类中包含某些语言 $\mathbf{L}(M)$, 在不属于 $\mathbf{L}(M)$ 的某些输入上 M 停不下来.

- 一般假定, 当输入串 w 被接受时图灵机 M 总会停机
- 而对于不接受的输入, 图灵机可能永远不停止

定义. 对接受和不接受的输入, 都能保证停机的图灵机, 所接受的语言称为递归语言.

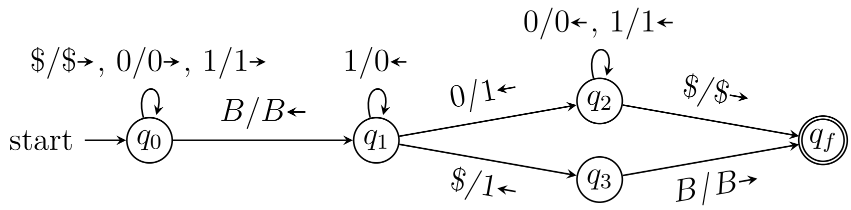
算法的形式化

保证停机的图灵机, 正是算法的好模型, 这也是算法概念的首次形式化.

8.1.4 整数计算器

图灵机可以作为语言的识别器, 也可以用作整数函数计算器和语言的枚举器.

例 4. 二进制数的加 1 函数, 使用符号 \$ 作为数字前的占位标记. 例如 $q_0\$10011 \vdash^* \$q_f10100$, $q_0\$111 \vdash^* q_f1000$.



8.2 图灵机的变形

TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.

状态中有存储的图灵机

设计有限控制器中可以存储有限个符号的图灵机:

$$M' = (Q', \Sigma, \Gamma, \delta, q_0', B, F')$$

其中 $Q' = Q \times \Gamma \times \cdots \times \Gamma$, $q_0' = [q_0, B, \cdots, B]$.

TM $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$.

多道图灵机

设计多道图灵机:

$$M' = (Q, \Sigma, \Gamma', \delta, q_0, B', F)$$

其中 $\Gamma' = \Gamma \times \Gamma \times \cdots \times \Gamma$.

8.2.1 扩展的图灵机

多带图灵机

由有限控制器, k 个带头和 k 条带组成. 在一个动作中, 根据有限控制器的状态和每个带头扫视的符号, 机器能够:

1. 改变状态;
2. 在带头所在单元, 打印一个符号;
3. 独立的向左或向右移动每个单元, 或保持不动.

开始时, 输入在第 1 条带上, 其他都是空的, 其形式定义非常繁琐.

定理 39. *如果语言 L 被一个多带图灵机接受, 那么 L 能够被某个单带图灵机接受.*

证明方法:

1. 用 $2k$ 道的单带图灵机 N 模拟 k 带图灵机 M;
2. N 用两道模拟 M 一带, 一道放置内容, 另一道标记带头;
3. 模拟 M 的一个动作, N 需要从左至右, 再从右至左扫描一次;
4. 第一次扫描搜集当前格局, 第二次扫描更新带头和位置.

非确定图灵机 (NTM)

在每个状态 q 和每个带符号 X 的转移, 可以有有限个选择的图灵机, 即

$$\delta(q, X) = \{(q_1, Y_1, D_1), (q_2, Y_2, D_2), \cdots, (q_k, Y_k, D_k)\}.$$

图灵机增加了非确定性, 并未改变图灵机接受语言的能力.

定理 40. *如果 L 被非确定图灵机接受, 那么 L 被图灵机接受.*

证明方法:

1. 同样用多带技术, 用确定的 TM M 模拟 NTM N;
2. M 用第 1 条带存储 N 未处理的 ID, 用第 2 条带模拟 N;
3. M 从第 1 条带取 N 的当前 ID 放到第 2 带;
4. 若不接受, 把当前 ID 可能的 k 个转移 ID 复制到第 1 条带的最末端;
5. 然后循环, 继续从第 1 带取下一个 ID 去模拟.

思考题

为什么非确定性没有改变图灵机识别语言的能力?

多维图灵机

1. 这种装置具有通常的有穷控制器;
2. k 维阵列组成的带, 在 $2k$ 个方向上都是无限的;
3. 根据状态和读入符号改变状态, 并沿着 k 个轴的正和负向移动;
4. 开始时，输入沿着某一个轴排列, 带头在输入的左端.

同样, 这样的扩展也没有增加额外的能力, 仍然等价于基本的图灵机.

8.2.2 受限的图灵机

半无穷带图灵机

图灵机的输入输出带只有一侧是无穷的.

定理 41. *半无穷带图灵机, 与图灵机等价.*

证明方法:

一侧无穷的图灵机带, 可使用多道技术, 模拟双侧无穷的图灵机带.

多栈机

基于下推自动机的扩展, k 栈机器是具有 k 个栈的确定型下推自动机.

定理 42. *如果图灵机接受 L, 那么双栈机接受 L.*

证明方法:

1. 一个堆栈保存带头左边内容, 一个堆栈保存带头右边内容;
2. 带头的移动用两个栈分别弹栈和压栈模拟;
3. 带头修改字符 A 为 B, 用一个栈弹出 A 而另一个压入 B 来模拟;
4. 开始时输入在双栈机的输入带, 但先将输入扫描并压入一个栈, 再依次弹出并压入另一个栈, 然后开始模拟图灵机.

例 5. 利用双栈机器接受 $L = \{a^n b^n c^n \mid n \geq 0\}$ 和 $L = \{a^n b^n c^n d^n e^n \mid n \geq 0\}$.