形式语言与自动机理论

图灵机

王春宇
chunyu@hit.edu.cn

计算学部
哈尔滨工业大学

2022 年 2 月

图灵机

- 不可判定的问题
- 图灵机
- 图灵机的变形

不可判定问题

典型问题

给定语言 $L \subseteq \Sigma^*$ 和字符串 $w \in \Sigma^*$, 判断是否 $w \in L$ 的问题,
称为语言 L 上的一个判定性问题.

(非形式) 定义

如果一个问题, 不存在能解决它的程序, 则称为不可判定问题.

不可判定问题

典型问题

给定语言 $L \subseteq \Sigma^*$ 和字符串 $w \in \Sigma^*$, 判断是否 $w \in L$ 的问题,
称为语言 L 上的一个判定性问题.

(非形式) 定义

如果一个问题, 不存在能解决它的程序, 则称为不可判定问题.

是否存在不可判定的问题?

1. $\{L \mid L \subseteq \Sigma^*\}$ 是不可数的;
2. $\{P \mid P\text{是一个程序}\}$ 是可数的;
3. 问题显然比程序多, 必然存在不可判定问题.

hello-world 问题

判断任意给定的程序 P 在任意给定的输入 x 时,
是否会以 `hello, world` 为其输出的前 12 个字符.

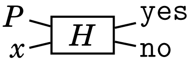
定理

hello-world 问题是不可判定的.

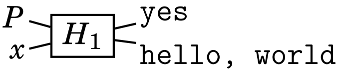
- “具有这个输入的这个程序是否显示 `hello, world`?”
- 解决这样问题的通用程序是不存在的.

(非形式) 证明: 反证法. 假设这样的程序 H 存在.

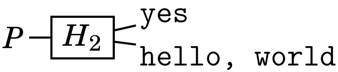
1. H 检查程序 P 在输入 x 时的输出, 并回答 `yes` 或 `no`:



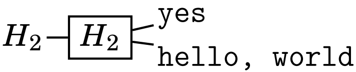
2. 修改 H, 在回答 `no` 时, 输出 `hello, world`:



3. 修改 H_1, 将程序 P 作为 P 自己的输入:



4. 那么, 当程序 H_2 以 H_2 为输入时:



5. H_2 的输出会出现矛盾 (悖论), 所以 H 不可能存在. □

问题的归约

如何证明问题是不可判定的?

1. 归谬法 (反证法)
2. 问题的归约

calls-foo 问题

程序 Q 在输入 y 时, 是否会调用函数 `foo` ?

例 1. 利用归约证明 calls-foo 问题是不可判定的.

calls-foo 问题

程序 Q 在输入 y 时, 是否会调用函数 `foo` ?

例 1. 利用归约证明 calls-foo 问题是不可判定的.

证明: 将 hello-world 问题归约到 calls-foo 问题.

$$P \text{ 输入 } x \text{ 时会输出 } \texttt{hello, world.} \mapsto Q \text{ 输入 } y \text{ 时会调用 } \texttt{foo}.$$

calls-foo 问题

程序 Q 在输入 y 时, 是否会调用函数 `foo`？

例 1. 利用归约证明 calls-foo 问题是不可判定的.

证明: 将 hello-world 问题归约到 calls-foo 问题.

$$P \text{ 输入 } x \text{ 时会输出 } \texttt{hello, world.} \mapsto Q \text{ 输入 } y \text{ 时会调用 } \texttt{foo}.$$

1. P_1: 如果 P 中有 `foo` 函数, 将其重命名 (重构);
2. P_2: 给 P_1 增加函数 `foo`;
3. P_3: 保存 P_2 输出的前 12 个字符;
4. P_4: 当 P_3 输出为 `hello, world` 时, 调用 `foo`.
5. 令 $Q = P_4$, $y = x$.

□

例 2. [Exercise 8.1.1a] 判断程序 R 在给定输入 z 时是否会运行结束 (halt)?

证明: 将 hello-world 问题归约到该问题.

$$P \text{ 输入 } x \text{ 时会输出 } \texttt{hello, world}. \mapsto R \text{ 输入 } z \text{ 时会运行结束}.$$

例 2. [Exercise 8.1.1a] 判断程序 R 在给定输入 z 时是否会运行结束 (halt)?

证明: 将 hello-world 问题归约到该问题.

$$P \text{ 输入 } x \text{ 时会输出 } \texttt{hello, world.} \mapsto R \text{ 输入 } z \text{ 时会运行结束.}$$

1. P_1: 在 P 主函数结束前增加死循环, 如 `while(1);`;
2. P_2: 保存 P_1 输出的前 12 个字符;
3. P_3: 当 P_2 输出为 `hello, world` 时, 结束程序;
4. 令 $R = P_3$, $z = x$. $\square$

计算模型

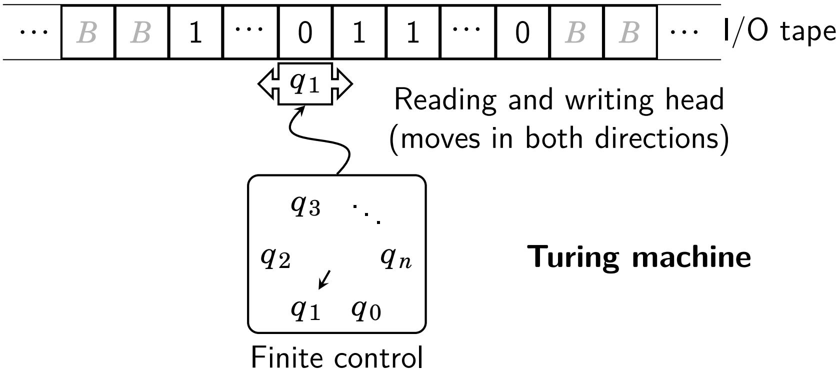
研究计算或可计算性, 需要 "计算" 的模型和形式定义:

- *Simple*: 描述/理解/使用. 易于形式化推理, 与直觉相符.
- *Powerful*: 计算能力/表示能力. 可表示任意算法.

图灵机

- 不可判定的问题
- 图灵机
  - 形式定义
  - 瞬时描述及其转移
  - 语言与停机
  - 整数函数计算器
- 图灵机的变形

图灵机



⋯
B
B
1
⋯
0
1
1
⋯
0
B
B
⋯
I/O tape
q_1
Reading and writing head
(moves in both directions)
q_3
q_2
q_n
q_1
q_0
Finite control
Turing machine

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. $q_0 \in Q$: 初始状态;
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

有穷控制器处于状态 q, 带头所在单元格为符号 X, 如果动作的定义为

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

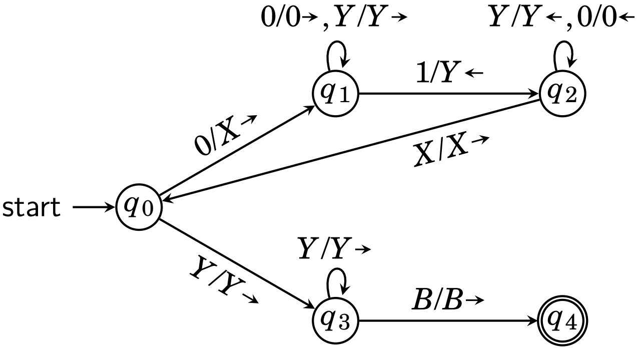
表示状态转移到 p, 单元格改为 Y, 然后带头向左移动一个单元格.

$$q \xrightarrow{X/Y\leftarrow} p$$

因为每个动作都是确定的, 因此是“确定的图灵机”.

例 3. 设计识别 $\left\{0^n1^n \,\middle|\, n \geq 1\right\}$ 的图灵机.

例 3. 设计识别 $\{0^n1^n \mid n \geq 1\}$ 的图灵机.



续例 3. 设计识别 $\left\{0^n 1^n \,\middle|\, n \geq 1\right\}$ 的图灵机.

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

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	$-$	$-$	$-$	$-$	$-$

瞬时描述

定义

图灵机虽有无穷长的带，但非空内容总是有限的. 因此用带上全部的非空符号、当前状态和带头位置，同时定义**瞬时描述**(*ID*) 为

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

1. 图灵机的当前状态 q;
2. 带头在左起第 i 个非空格符上;
3. $X_1X_2\cdots X_n$ 是最左到最右非空格内容.
4. 一般假定 Q 和 Γ 不相交以避免混淆.

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

$$X_1\cdots X_{i-1}qX_i\cdots X_n \vdash_M X_1\cdots X_{i-2}pX_{i-1}YX_{i+1}\cdots X_n$$

如果 $\delta(q,X_i)=(p,Y,R)$ 那么

$$X_1\cdots X_{i-1}qX_i\cdots X_n \vdash_M X_1\cdots X_{i-1}YpX_{i+1}\cdots X_n$$

若某 *ID* 是从另一个经有限步 *(*包括零步*)* 转移而得到的, 记为 $\vdash_M^*$.
若 M 已知, 简记为 $\vdash$ 和 $\vdash^*$.

续例 3. 设计识别 $\left\{0^n 1^n \,\middle|\, n \geq 1\right\}$ 的图灵机.

接受 0011 的 ID 序列 (M 的一个计算) 为

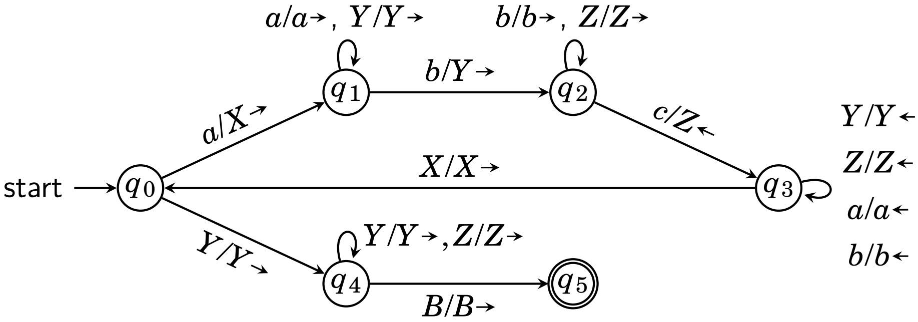
续例 3. 设计识别 $\{0^n1^n \mid n \geq 1\}$ 的图灵机.

接受 0011 的 ID 序列 (M 的一个计算) 为

$$
\begin{aligned}
q_0 0011 &\vdash Xq_1 011 && \vdash X0q_1 11 && \vdash Xq_2 0Y1 \\
&\vdash q_2 X0Y1 && \vdash Xq_0 0Y1 && \vdash XXq_1 Y1 \\
&\vdash XXYq_1 1 && \vdash XXq_2 YY && \vdash Xq_2 XYY \\
&\vdash XXq_0 YY && \vdash XXYq_3 Y && \vdash XXYYq_3 B \\
&\vdash XXYYBq_4 B
\end{aligned}
$$

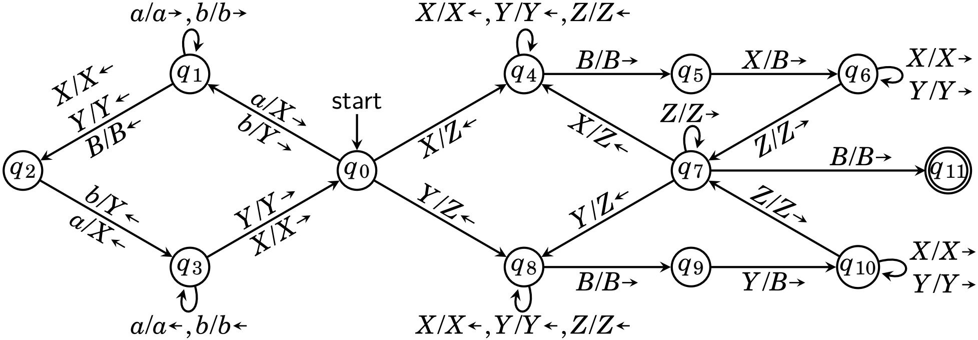
例 4. 设计接受 $L=\left\{a^n b^n c^n \mid n \geq 1\right\}$ 的图灵机.

例 4. 设计接受 $L = \{ a^n b^n c^n \mid n \geq 1 \}$ 的图灵机.



例 5. 设计接受 $L = \left\{ ww \,\middle|\, w \in \{a,b\}^+ \right\}$ 的图灵机.

例 5. 设计接受 $L = \left\{ ww \,\middle|\, w \in \{a,b\}^+ \right\}$ 的图灵机.



思考

DFA 和 TM 的主要区别? — 能够“写”是多么重要

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

$$\mathbf{L}(M)=\left\{\, w \;\middle|\; w\in\Sigma^*,\ q_0w \overset{*}{\vdash} \alpha p\beta,\ p\in F,\ \alpha,\beta\in\Gamma^* \,\right\}.$$

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

$$\mathbf{L}(M)=\left\{w \,\middle|\, w\in\Sigma^*,\ q_0w\overset{*}{\vdash}\alpha p\beta,\ p\in F,\ \alpha,\beta\in\Gamma^*\right\}.$$

定义

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

- 一般假定, 当输入串被接受时, 图灵机总会停机;
- 然而, 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，都保证停机的图灵机，所接受的语言称为**递归语言**.

定义

对接受和不接受的输入，都保证停机的图灵机，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，即算法概念的形式化.

- λ-caculus — Alonzo Church, Stephen Kleene
- Partial recursive functions — Kurt Gödel
- Post machines — Emil Post
- Turing machines — Alan Turing

整数函数计算器

- 传统的方法, 把整数 $i \geq 0$ 写为 1 进制, 用字符串 0^i 表示;
- 若计算 k 个自变量 $i_1, i_2, \dots, i_k$ 的函数 f, 用

$$0^{i_1}10^{i_2}1\cdots10^{i_k}$$

  作为 TM M 的输入;
- M 停机, 且输入带上为 0^m, 表示 $f(i_1, i_2, \dots, i_k) = m$.
- M 计算的 f, 不必对所有不同的 $i_1, i_2, \dots, i_k$ 都有结果.

定义

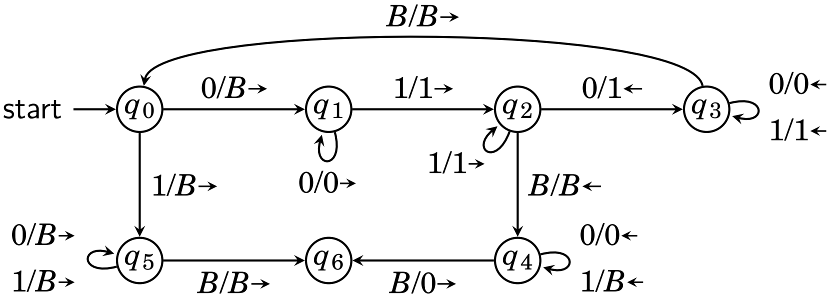
如果 $f(i_1, i_2, \dots, i_k)$ 对所有$i_1, i_2, \dots, i_k$ 都有定义, 称 f 为全递归函数.
被图灵机计算的函数 $f(i_1, i_2, \dots, i_k)$ 称作部分递归函数.

例 6. 给出计算整数真减法 ($\dot{-}$) 的图灵机, 其定义为

$$m \dot{-} n = \begin{cases} m - n & m \geq n \\ 0 & m < n \end{cases}.$$

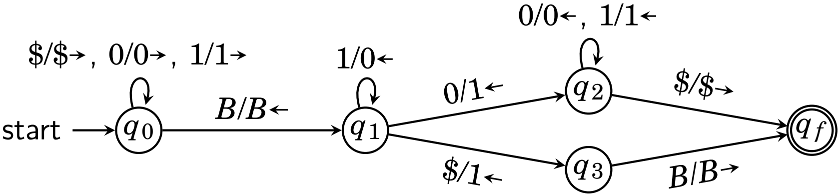
例 6. 给出计算整数真减法 ($\dot{-}$) 的图灵机, 其定义为

$$m \dot{-} n = \begin{cases} m-n & m \geq n \\ 0 & m < n \end{cases}.$$



例 7. 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 $q_0\$10011 \overset{*}{\vdash} \$q_f 10100$, $q_0\$111 \overset{*}{\vdash} q_f 1000$.

例 7. 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 $q_0\$10011 \overset{*}{\vdash} \$q_f10100$, $q_0\$111 \overset{*}{\vdash} q_f1000$.



图灵机

- 不可判定的问题
- 图灵机
- 图灵机的变形
  - 扩展的图灵机
  - 受限的图灵机
  - 图灵机的抽象描述

状态中存储

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 $Q' = Q \times \Gamma \times \cdots \times \Gamma$, $q'_0 = [q_0, B, \cdots, B]$.

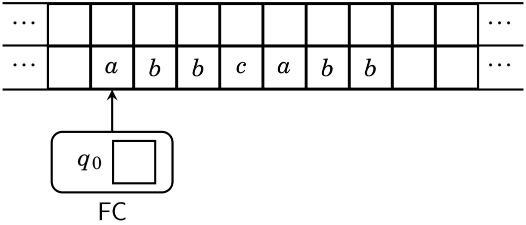
多道

多道图灵机:

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

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

例 8. 利用状态中存储与多道设计 TM 识别 $L = \{\, wcw \mid w \in \{a, b\}^* \,\}$.



子程序

设计 TM 的一部分作为一个子程序:

- 具有一个指定的初始状态;
- 具有一个指定的返回状态, 但暂时没有定义动作;
- 可以具有参数和返回值.

通过进入子程序的初始状态, 实现调用; 通过返回状态的动作, 实现返回.

例 9. 设计 TM 实现全递归函数 “乘法”.

扩展的图灵机

多带图灵机

有穷控制器、k 个带头和 k 条带组成. 每个动作, 根据状态和每个带头符号:

1. 改变控制器中的状态;
2. 修改带头单元格中的符号;
3. 每个带头独立的向左或右移动一个单元格, 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 40

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.

定理 40

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.

图灵机的运行时间

定义

图灵机 M 在输入 w 上的运行时间是 M 在停机之前移动的步数.
M 的时间复杂度是在所有长度为 n 的输入上, 运行时间最大值的函数 $T(n)$.

- 如果 M 在某些 w 上不停机, 则 $T(n)$ 是无穷的;
- 所以, 保证停机的图灵机, 其 $T(n)$ 才有意义;
- 但是, 只有多项式时间的 $T(n)$, 才是问题实际可解的边界.
- 然而, 对很多问题, 还没有比精确到多项式时间更好的结果.

定理 41

单带图灵机 N 模拟 k 带图灵机 M 的 n 步移动, 需要使用 $O(n^2)$ 的时间.

证明:

1. M 的 n 步移动, 带头相距不会超过 $2n$;
2. 而标记带头并调转方向至多需要 $2k$ 步;
3. 因此 N 模拟 M 的 1 步至多需要 $4n+2k$ 步, 即 $O(n)$ 时间;
4. 因此模拟 n 步需要 $O(n^2)$ 时间. □

实际可行性

使用单带 TM 或多带 TM, 不会改变问题是否实际可解.

非确定图灵机 (NTM)

图灵机在每组状态 q 和带符号 X 的转移 $\delta(q,X)$, 可以有有限个选择:

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

- NTM 接受语言的方式, 与 NFA 和 PDA 是类似的
- 存在从初始 ID 到某接受状态 ID 的转移, 其他选择可以忽略

定理 42

如果 L 被非确定图灵机接受, 那么 L 被图灵机接受.

定理 42

如果 L 被非确定图灵机接受, 那么 L 被图灵机接受.

证明:

1. TM M 用控制器保存并用两条带模拟 NTM N 的动作:
   第 1 条带存储 N 未处理的 ID, 第 2 条带模拟 N 的带.
2. M 将第 1 条带最前端的 ID 复制到第 2 带, 若接受则停止;
3. 把当前 ID 可能的 k 个转移 ID 复制到第 1 条带的最末端;
4. 将第 1 带上最前端的 ID 抹掉, 从第 2 步重复.

证明 (续):

- 只有 N 进入接受的 ID 时, M 才会且一定会接受;
- 因为若 N 每步最多 m 个选择, 那么从初始 ID 经过 n 步最多可到

$$1 + m + m^2 + \cdots + m^n$$

个 ID, 而 M 会以 "先广" 顺序检查这些最多为 nm^n 个的 ID. $\square$

TM 模拟 NTM 的时间

- NTM N 的 n 步计算, TM M 需要指数时间 $O(m^n)$ 才能完成.
- 但这里 “指数时间的增长” 是否必然, 仍是未知的.
- NTM 以多项式时间解决的问题, TM 是否也能以多项式时间解决呢?

$$\mathrm{P} \stackrel{?}{=} \mathrm{NP}$$

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. 具有通常的有穷控制器和一个带头;
2. 由 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.

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 43

半无穷带图灵机, 与图灵机等价.

证明方法: 一侧无穷的带上使用多道技术, 模拟双侧无穷的带.

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 44

如果图灵机接受 L, 那么双栈机接受 L.

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 44

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 10. 设计双栈机接受 $L = \left\{ a^n b^n c^n \;\middle|\; n \geq 0 \right\}$.

图灵机的描述

- 形式化表示 – 底层、准确
  - 精确的数学语言
  - 相当于实现算法的程序代码
- 实现细节说明 – 明确、规范
  - 准确的读写头动作和带内容管理
  - 相当于描述算法的伪代码
- 抽象叙述 – 精简、高效
  - 说明性的日常语言
  - 相当于概括算法的思想

哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

chunyu@hit.edu.cn
http://iilab.net/chunyu