形式语言与自动机理论

课程简介与基础知识

王春宇

计算机科学与技术学院
哈尔滨工业大学

课程简介与基础知识

- 课程简介
- 基础知识

计算理论

核心问题

计算机的基本能力和限制是什么?

❶ 究竟哪些问题, 可通过计算解决? — 可计算性理论

❷ 解决可计算的问题, 究竟需要多少资源? — 计算复杂性理论

❸ 为了研究计算, 要使用哪些计算模型? — 形式语言与自动机理论

什么是自动机理论?

自动机理论: 研究抽象机器及其所能解决问题的理论.

- 图灵机
- 有限状态机
- 文法, 下推自动机

什么是形式语言?

形式语言: 经数学定义的语言.

		自然语言		形式语言	
		English	中文	化学分子式	C 语言
语言	字符	A,a,B,b,...	天, 地,...	A-Z,a-z,0-9 ...	`A-Z,a-z,0-9` ...
	单词	apple	苹果	H_2O	`char`
	句子	How're you?	早上好!	$2H_2+O_2=2H_2O$	`char a=10;`
	语法	Grammar	语法规则	精确定义的规则	

课程内容

- 正则语言
  - 有穷自动机
  - 正则表达式
  - 正则语言的性质
- 上下文无关语言
  - 上下文无关文法
  - 下推自动机
  - 上下文无关语言的性质
- 计算导论
  - 图灵机及其扩展
  - 不可判定性

参考书

- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* 《自动机理论、语言和计算导论》机械工业出版社
- Michael Sipser. *Introduction to the Theory of Computation.* 《计算理论导引》机械工业出版社

课程简介与基础知识

- 课程简介
- 基础知识
  - 基本概念
  - 语言和问题
  - 形式化证明

基本概念

1. 字母表: 符号 (或字符) 的非空有穷集.

$$\begin{aligned}
\Sigma_1 &= \{0, 1\}, \\
\Sigma_2 &= \{a, b, \ldots, z\}, \\
\Sigma_3 &= \{x \mid x \text{ 是一个汉字}\}.
\end{aligned}$$

2. 字符串: 由某字母表中符号组成的有穷序列.

若 $\Sigma_1 = \{0, 1\}$, 那么 $0, 1, 00, 111001$ 为 Σ_1 上的字符串;
若 $\Sigma_2 = \{a, b, \ldots, z\}$, 那么 $ab, xkcd$ 为 Σ_2 上的字符串.

3. 空串: 记为 ε, 有 0 个字符的串.

字母表 Σ 可以是任意的, 但都有 $\varepsilon \notin \Sigma$.

4. 字符串的长度: 字符串中符号所占位置的个数, 记为 $|\sqcup|$.
若字母表为 Σ, 可递归定义为:

$$|w| = \begin{cases} 0 & w = \varepsilon \\ |x| + 1 & w = xa \end{cases},$$

其中 $a \in \Sigma$, w 和 x 是 Σ 中字符组成的字符串.

$\star$. 符号使用的一般约定:

- 字母表: $\Sigma, \Gamma, \Delta, \ldots$
- 字符: $a, b, c, \ldots$
- 字符串: $\ldots, w, x, y, z$
- 集合: $A, B, C, \ldots$

5. 字符串 x 和 y 的连接: 将首尾相接得到新字符串的运算, 记为 $x \cdot y$ 或 xy. 同样, 可递归定义为

$$x \cdot y = \begin{cases} x & y = \varepsilon \\ (x \cdot z)a & y = za \end{cases},$$

其中 $a \in \Sigma$, 且 x, y, z 都是字符串.

对任何字符串 x, 有 $\varepsilon \cdot x = x \cdot \varepsilon = x$.

连接运算的符号 “$\cdot$” 一般省略.

6. 字符串 x 的 n 次幂($n \geq 0$), 递归定义为

$$x^n = \begin{cases} \varepsilon & n = 0 \\ x^{n-1}x & n > 0 \end{cases} .$$

例如,

$$\begin{aligned} (ba)^2 &= (ba)^1 ba \\ &= (ba)^0 baba \\ &= \varepsilon baba \\ &= baba \end{aligned} \qquad \begin{aligned} ba^2 &= ba^1 a \\ &= ba^0 aa \\ &= b\varepsilon aa \\ &= baa \end{aligned}$$

7. 集合 A 和 B 的连接, 记为 $A \cdot B$ 或 AB, 定义为

$$A \cdot B = \{w \mid w = x \cdot y,\ x \in A \text{ 且 } y \in B\}.$$

8. 集合 A 的 n 次幂$(n \geq 0)$, 递归定义为

$$A^n = \begin{cases} \{\varepsilon\} & n = 0 \\ A^{n-1}A & n \geq 1 \end{cases}.$$

那么, 若 Σ 为字母表, 则 Σ^n 为 Σ 上长度为 n 的字符串集合.
如果 $\Sigma = \{0, 1\}$, 有

$$\begin{aligned} \Sigma^0 &= \{\varepsilon\} \\ \Sigma^1 &= \{0, 1\} \\ \Sigma^2 &= \{00, 01, 10, 11\} \\ \Sigma^3 &= \{000, 001, 010, 011, 100, 101, 110, 111\} \\ &\vdots \end{aligned}$$

9. 克林闭包($Kleene\ Closure$):

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i.$$

10. 正闭包($Positive\ Closure$):

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i.$$

显然,

$$\Sigma^* = \Sigma^+ \cup \{\varepsilon\}.$$

其他的概念如有向图, 树, 字符串的前缀, 后缀等定义这里省略.

语言

定义

若 Σ 为字母表且 $\forall L \subseteq \Sigma^*$, 则 L 称为字母表 Σ 上的语言.

- 自然语言, 程序设计语言等
- $\{0^n 1^n \mid n \geq 0\}$
- The set of strings of 0's and 1's with an equal number of each:

$$\{\varepsilon, 01, 10, 0011, 0101, 1100, \ldots\}$$

- $\varnothing$, $\{\varepsilon\}$ 和 Σ^* 分别都是任意字母表 Σ 上的语言, 但注意 $\varnothing \neq \{\varepsilon\}$

关于语言

唯一重要的约束就是所有字母表都是有穷的.

问题

典型问题

判断给定的字符串 w 是否属于某个具体的语言 L,

$$w \in L?$$

- 任何所谓问题, 都可以转为语言成员性的问题
- 语言和问题其实是相同的东西

形式化证明: 演绎法, 归纳法和反证法

例 1. 若 x 和 y 是 Σ 上的字符串, 请证明 $|xy| = |x| + |y|$.

证明: 通过对 $|y|$ 的归纳来证明

❶ 基础: 当 $|y| = 0$, 即 $y = \varepsilon$

$$\begin{aligned} |x\varepsilon| &= |x| && \text{连接的定义} \\ &= |x| + |\varepsilon| && \text{长度的定义} \end{aligned}$$

❷ 递推: 假设 $|y| = n$ $(n \geq 0)$ 时命题成立,
那么当 $|y| = n + 1$, 即 $y = wa$

$$\begin{aligned} |x(wa)| &= |(xw)a| && \text{连接的定义} \\ &= |xw| + 1 && \text{长度的定义} \\ &= |x| + |w| + 1 && \text{归纳假设} \\ &= |x| + |wa| && \text{长度的定义} \end{aligned}$$

□

形式化证明: 演绎法, 归纳法和反证法

例 1. 若 x 和 y 是 Σ 上的字符串, 请证明 $|xy| = |x| + |y|$.

证明: 通过对 y 的结构归纳来证明

❶ 基础: $y = \varepsilon$ 时

$$\begin{aligned} |x\varepsilon| &= |x| && \text{连接的定义} \\ &= |x| + |\varepsilon| && \text{长度的定义} \end{aligned}$$

❷ 递推: 假设 $y = w$ $(w \in \Sigma^*)$ 时命题成立,
那么当 $y = wa$ 时

$$\begin{aligned} |x(wa)| &= |(xw)a| && \text{连接的定义} \\ &= |xw| + 1 && \text{长度的定义} \\ &= |x| + |w| + 1 && \text{归纳假设} \\ &= |x| + |wa| && \text{长度的定义} \end{aligned}$$

□

形式语言与自动机理论

有穷自动机

王春宇

计算机科学与技术学院
哈尔滨工业大学

有穷自动机

- 有穷状态系统
- 确定的有穷自动机
- 非确定有穷自动机
- 带有空转移的非确定有穷自动机

有穷状态系统

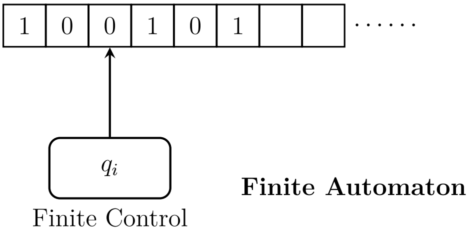
- 有限状态机: Moore Machine, Mealy Machine
- 数字电路设计
- 电脑游戏的 AI 设计
- 各种通讯协议: TCP, HTTP, Bluetooth, Wifi
- 文本搜索, 词法分析

有穷自动机

- 有穷状态系统
- 确定的有穷自动机
  - 形式定义
  - DFA 的设计举例
  - 扩展转移函数
  - DFA 的语言与正则语言
- 非确定有穷自动机
- 带有空转移的非确定有穷自动机

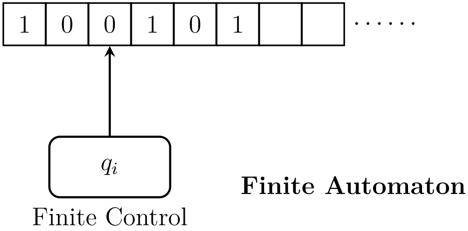
确定的有穷自动机

- 一条输入带
- 一个读头
- 一个有穷控制器



确定的有穷自动机

- 一条输入带
- 一个读头
- 一个有穷控制器



例 1. 用有穷自动机识别 $\{w \in \{0,1\}^* \mid w$ 的长度 $|w|$ 是偶数.$\}$

确定的有穷自动机的形式定义

定义

确定的有穷自动机(*DFA*, *Deterministic Finite Automaton*) A 为五元组

$$A = (Q, \Sigma, \delta, q_0, F)$$

1. Q : 有穷状态集;
2. Σ : 有穷输入符号集或字母表;
3. $\delta : Q \times \Sigma \to Q$, 状态转移函数;
4. $q_0 \in Q$: 初始状态;
5. $F \subseteq Q$: 终结状态集或接受状态集.

例 2. 请设计 DFA, 在任何由 0 和 1 构成的串中, 接受含有 01 子串的全部串.

例 2. 请设计 DFA, 在任何由 0 和 1 构成的串中, 接受含有 01 子串的全部串.

❶ 未发现 01, 即使 0 都还没出现过;

❷ 未发现 01, 但刚刚读入字符是 0;

❸ 已经发现了 01.

例 2. 请设计 DFA, 在任何由 0 和 1 构成的串中, 接受含有 01 子串的全部串.

❶ 未发现 01, 即使 0 都还没出现过;
❷ 未发现 01, 但刚刚读入字符是 0;
❸ 已经发现了 01.

因此 DFA A 的可定义为:

$$A = (\{q_1, q_2, q_3\}, \{0, 1\}, \delta, q_1, \{q_3\})$$

其中 δ 为:

$$\begin{aligned} \delta(q_1, 1) &= q_1 & \delta(q_2, 1) &= q_3 & \delta(q_3, 1) &= q_3 \\ \delta(q_1, 0) &= q_2 & \delta(q_2, 0) &= q_2 & \delta(q_3, 0) &= q_3 \end{aligned}$$

状态转移图

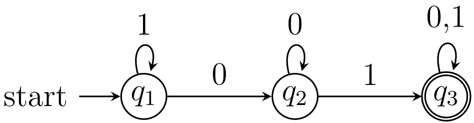
❶ 每个状态 q 对应一个节点, 用圆圈表示;

❷ 状态转移 $\delta(q,a)=p$ 为一条从 q 到 p 且标记为字符 a 的有向边;

❸ 开始状态 q_0 用一个标有 start 的箭头表示;

❹ 接受状态的节点, 用双圆圈表示.

续例 2. 含有 01 子串的全部串的状态转移图



状态转移表

1. 每个状态 q 对应一行, 每个字符 a 对应一列;
2. 若有 $\delta(q, a) = p$, 用第 q 行第 a 列中填入的 p 表示;
3. 开始状态 q_0 前, 标记箭头 $\rightarrow$ 表示;
4. 接受状态 $q \in F$ 前, 标记星号 $*$ 表示.

续例 2. 含有 01 子串的全部串的状态转移表

	0	1
$\rightarrow q_1$	q_2	q_1
q_2	q_2	q_3
$*q_3$	q_3	q_3

典型问题

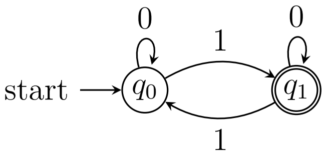
设计 DFA 使其接受且仅接受给定的语言 L.

例 3. 若 $\Sigma = \{0, 1\}$, 给出接受全部含有奇数个 1 的串 DFA.

典型问题

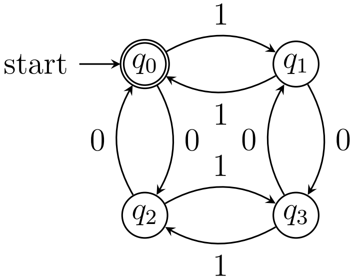
设计 DFA 使其接受且仅接受给定的语言 L.

例 3. 若 $\Sigma = \{0, 1\}$, 给出接受全部含有奇数个 1 的串 DFA.



例 4. 若 $\Sigma = \{0, 1\}$, 给出接受全部含有偶数个 0 和偶数个 1 的串 DFA.

例 4. 若 $\Sigma = \{0, 1\}$, 给出接受全部含有偶数个 0 和偶数个 1 的串 DFA.



思考题

若 $\Sigma = \{0, 1\}$

❶ 如何设计接受 $\varnothing$ 的 DFA?

❷ 如何设计接受 Σ^* 的 DFA?

❸ 如何设计接受 $\{\varepsilon\}$ 的 DFA?

扩展转移函数

定义

扩展 δ 到字符串，定义**扩展转移函数** $\hat{\delta} : Q \times \Sigma^* \to Q$ 为

$$\hat{\delta}(q, w) = \begin{cases} q & w = \varepsilon \\ \delta(\hat{\delta}(q, x), a) & w = xa \end{cases}$$

其中 $a \in \Sigma$，$w, x \in \Sigma^*$.

扩展转移函数

定义

扩展 δ 到字符串，定义扩展转移函数 $\hat{\delta}: Q \times \Sigma^* \to Q$ 为

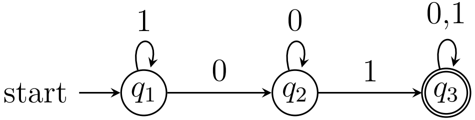
$$\hat{\delta}(q, w) = \begin{cases} q & w = \varepsilon \\ \delta(\hat{\delta}(q, x), a) & w = xa \end{cases}$$

其中 $a \in \Sigma$，$w, x \in \Sigma^*$.

那么，当 $w = a_0 a_1 \cdots a_n$，则有

$$\begin{aligned} \hat{\delta}(q, w) &= \delta(\hat{\delta}(q, a_0 a_1 \cdots a_{n-1}), a_n) \\ &= \delta(\delta(\hat{\delta}(q, a_0 a_1 \cdots a_{n-2}), a_{n-1}), a_n) = \cdots \\ &= \delta(\delta(\cdots \delta(\hat{\delta}(q, \varepsilon), a_0) \cdots, a_{n-1}), a_n) \end{aligned}$$

续例 2. 接受全部含有 01 子串的 DFA, $\hat{\delta}$ 处理串 0101 的过程.



$$
\begin{aligned}
\hat{\delta}(q_0, 0101) &= \delta(\hat{\delta}(q_0, 010), 1) \\
&= \delta(\delta(\hat{\delta}(q_0, 01), 0), 1) \\
&= \delta(\delta(\delta(\hat{\delta}(q_0, 0), 1), 0), 1) \\
&= \delta(\delta(\delta(\delta(\hat{\delta}(q_0, \varepsilon), 0), 1), 0), 1) \\
&= \delta(\delta(\delta(\delta(q_0, 0), 1), 0), 1) \\
&= \delta(\delta(\delta(q_1, 1), 0), 1) \\
&= \delta(\delta(q_2, 0), 1) = \delta(q_2, 1) = q_2
\end{aligned}
$$

思考题

1. 扩展转移函数 $\hat{\delta}$ 必须从开始状态 q_0 处理字符串吗?
2. 对任意的串 w, $\hat{\delta}$ 能保证一定会跳转到某个状态吗?

例 5. 对任何状态 q 及字符串 x 和 y, 证明 $\hat{\delta}(q,xy)=\hat{\delta}(\hat{\delta}(q,x),y)$.

证明: 对 y 使用归纳法.

❶ 当 $y=\varepsilon$ 时

$$\begin{aligned}\hat{\delta}(\hat{\delta}(q,x),\varepsilon) &= \hat{\delta}(q,x) && \hat{\delta}\text{的定义}\\ &= \hat{\delta}(q,x\varepsilon)\end{aligned}$$

❷ 假设 $y=w$ $(w\in\Sigma^*)$ 时命题成立, 当 $y=wa$ $(a\in\Sigma)$ 时

$$\begin{aligned}\hat{\delta}(q,xwa) &= \delta(\hat{\delta}(q,xw),a) && \hat{\delta}\text{和连接的定义}\\ &= \delta(\hat{\delta}(\hat{\delta}(q,x),w),a) && \text{归纳假设}\\ &= \hat{\delta}(\hat{\delta}(q,x),wa) && \hat{\delta}\text{的定义}\end{aligned}$$

$\square$

DFA 的语言与正则语言

定义

若 $D=(Q,\Sigma,\delta,q_0,F)$ 是一个*DFA*, 则 D **接受的语言**为

$$\mathbf{L}(D)=\{w\in\Sigma^*\mid\hat{\delta}(q_0,w)\in F\}.$$

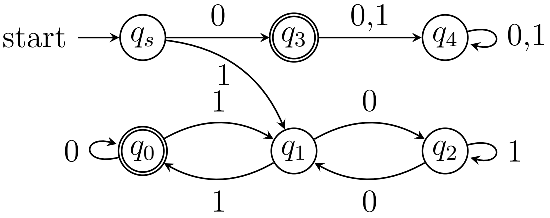
定义

如果语言 L 是某个 *DFA* D 的语言, 即 $L=\mathbf{L}(D)$, 则称 L 是**正则语言**.

- $\varnothing$, $\{\varepsilon\}$ 都是正则语言
- 若 Σ 是字母表, Σ^*, Σ^n 都是 Σ 上的正则语言

例 6. 设计 DFA 接受 $\{0,1\}$ 上的字符串 w, 且 w 是 3 的倍数的二进制表示.

例 6. 设计 DFA 接受 $\{0,1\}$ 上的字符串 w, 且 w 是 3 的倍数的二进制表示.

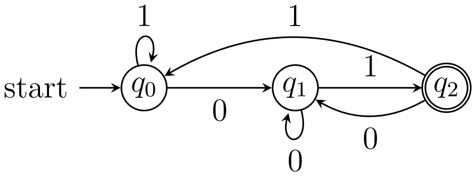


有穷自动机

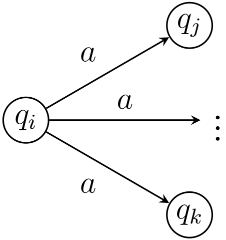
- 有穷状态系统
- 确定的有穷自动机
- 非确定有穷自动机
  - 形式定义
  - 扩展转移函数
  - NFA 的语言
  - DFA 与 NFA 的等价性
- 带有空转移的非确定有穷自动机

例 7. 由 0 和 1 构成的串中, 接受全部以 01 结尾的串, 如何设计 DFA?

例 7. 由 0 和 1 构成的串中, 接受全部以 01 结尾的串, 如何设计 DFA?

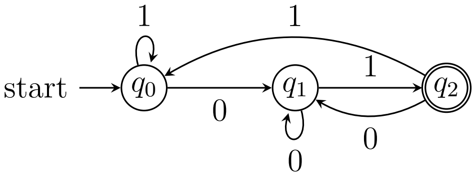


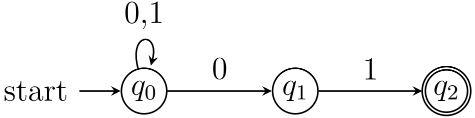
状态的非确定转移



- 同一个状态在相同的输入下, 可以有多个转移状态
- 自动机可以处在多个当前状态
- 使自动机的设计更容易

续例 7. 由 0 和 1 构成的串中, 接受全部以 01 结尾的串.





思考题

有穷自动机有了非确定性，能否增加它识别语言的能力？

非确定有穷自动机的形式定义

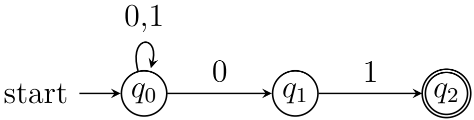
定义

非确定有穷自动机(*NFA*, *Nondeterministic Finite Automaton*) A 为五元组

$$A = (Q, \Sigma, \delta, q_0, F)$$

1. Q : 有穷状态集;
2. Σ : 有穷输入符号集或字母表;
3. $\delta : Q \times \Sigma \to 2^Q$ 状态转移函数;
4. $q_0 \in Q$: 为初始状态;
5. $F \subseteq Q$: 为终结状态集或接受状态集.

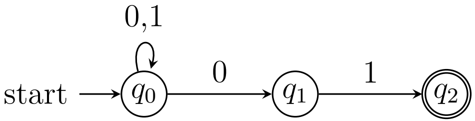
续例 7. 接受全部以 01 结尾的串的 NFA.



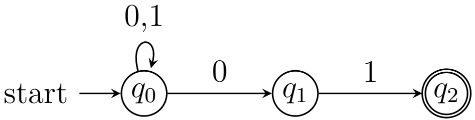
五元组为 $A = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\})$, 转移函数 δ:

$$\begin{array}{lll} \delta(q_0, 0) = \{q_0, q_1\} & \delta(q_1, 0) = \varnothing & \delta(q_2, 0) = \varnothing \\ \delta(q_0, 1) = \{q_0\} & \delta(q_1, 1) = \{q_2\} & \delta(q_2, 1) = \varnothing \end{array}$$

续例 7. 接受全部以 01 结尾的串的 NFA. 识别字符串 00101 的过程.



续例 7. 接受全部以 01 结尾的串的 NFA.



状态转移表:

	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$*q_2$	$\varnothing$	$\varnothing$

扩展转移函数

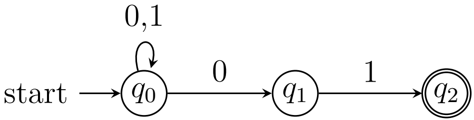
定义

扩展 δ 到字符串，定义扩展转移函数 $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ 为

$$\hat{\delta}(q, w) = \begin{cases} \{q\} & w = \varepsilon \\ \bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a) & w = xa \end{cases}$$

其中 $a \in \Sigma$，$w, x \in \Sigma^*$.

续例 7. 接受 01 结尾的串的 NFA, $\hat{\delta}$ 处理 00101 时每步的状态转移.



❶ $\hat{\delta}(q_0, \varepsilon) = \{q_0\}$

❷ $\hat{\delta}(q_0, 0) = \delta(q_0, 0) = \{q_0, q_1\}$

❸ $\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$

❹ $\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

❺ $\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_2, 0) = \{q_0, q_1\} \cup \varnothing = \{q_0, q_1\}$

❻ $\hat{\delta}(q_0, 00101) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} = \{q_0, q_2\}$

因为 q_2 是接受状态, 所以 NFA 接受 00101.

NFA 的语言

回顾

若 $D = (Q, \Sigma, \delta, q_0, F)$ 是一个 DFA, 则 D 接受的语言为

$$\mathbf{L}(D) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}.$$

定义

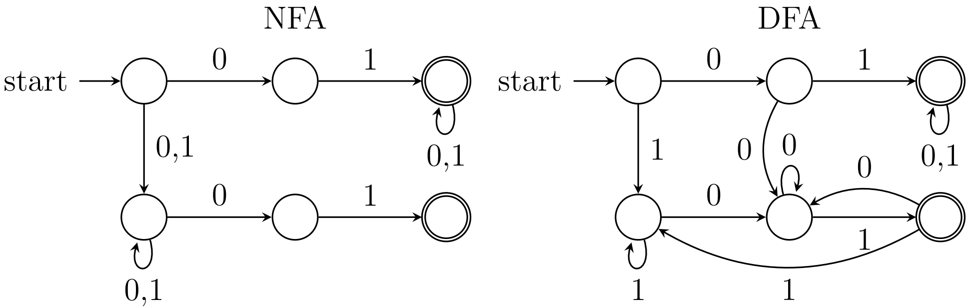
若 $N = (Q, \Sigma, \delta, q_0, F)$ 是一个*NFA*, 则 N 接受的语言为

$$\mathbf{L}(N) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \varnothing\}.$$

例 8. 设计 $L = \{w \in \{0,1\}^* \mid w$ 的首尾字符相同.$\}$ 的 NFA.

例 9. 设计 $L = \{w \in \{0,1\}^* \mid w \text{ either begin or ends with 01.}\}$ 的 NFA.

例 9. 设计 $L = \{w \in \{0,1\}^* \mid w \text{ either begin or ends with 01.}\}$ 的 NFA.



DFA 与 NFA 的等价性

定理 1

如果语言 L 被 *NFA* 接受，当且仅当 L 被 *DFA* 接受.

子集构造法

如果 NFA $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ 构造 DFA

$$D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$$

1. $Q_D = 2^{Q_N}$;
2. $F_D = \{S \mid S \subseteq Q_N, S \cap F_N \neq \varnothing\}$;
3. $\forall S \subseteq Q_N$, $\forall a \in \Sigma$:

$$\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a).$$

那么有 $\mathbf{L}(D) = \mathbf{L}(N)$.

证明: 为证明 $\mathbf{L}(D)=\mathbf{L}(N)$, 对 $|w|$ 用归纳法, 往证

$$\hat{\delta}_D(\{q_0\},w)=\hat{\delta}_N(q_0,w).$$

❶ 归纳基础: 当 $w=\varepsilon$ 时,

$$\hat{\delta}_D(\{q_0\},\varepsilon)=\{q_0\}=\hat{\delta}_N(q_0,\varepsilon)$$

❷ 归纳递推: 假设 $w=x$ $(x\in\Sigma^*)$ 时成立, 当 $w=xa$ $(a\in\Sigma)$ 时

$$\begin{aligned}
\hat{\delta}_N(q_0,xa) &= \cup_{p\in\hat{\delta}_N(q_0,x)}\delta_N(p,a) && \text{NFA 的 } \hat{\delta} \text{ 定义}\\
&= \cup_{p\in\hat{\delta}_D(\{q_0\},x)}\delta_N(p,a) && \text{归纳假设}\\
&= \delta_D(\hat{\delta}_D(\{q_0\},x),a) && D \text{ 的构造}\\
&= \hat{\delta}_D(\{q_0\},xa) && \text{DFA 的 } \hat{\delta} \text{ 定义}
\end{aligned}$$

因此上式成立.

因为

$$\hat{\delta}_D(\{q_0\}, w) = \hat{\delta}_N(q_0, w)$$

所以, 对 $\forall w \in \Sigma^*$ 有

$$\begin{aligned}
w \in \mathbf{L}(N) &\iff \hat{\delta}_N(q_0, w) \cap F_N \neq \varnothing && \text{NFA 的语言} \\
&\iff \hat{\delta}_D(\{q_0\}, w) \cap F_N \neq \varnothing && \text{刚证明的} \\
&\iff \hat{\delta}_D(\{q_0\}, w) \in F_D && D \text{ 的构造} \\
&\iff w \in \mathbf{L}(D) && \text{DFA 的语言}
\end{aligned}$$

所以

$$\mathbf{L}(D) = \mathbf{L}(N).$$

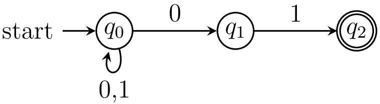
□

思考题

非确定性没能增加有穷自动机识别语言的能力，原因是什么呢？

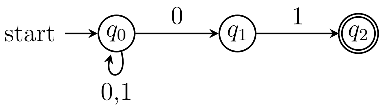
子集构造法: 构造与 NFA 等价的 DFA

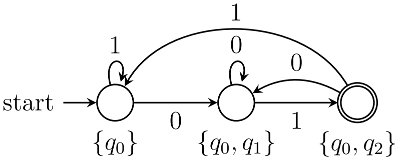
续例 7. 将接受全部以 01 结尾的串的 NFA 转换为 DFA.



	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$*q_2$	$\varnothing$	$\varnothing$

续例 7. 将接受全部以 01 结尾的串的 NFA 转换为 DFA.

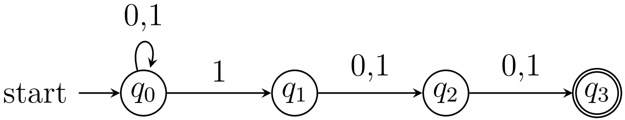




	0	1
$\rightarrow q_0$	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
$*q_2$	$\varnothing$	$\varnothing$
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\cancel{\{q_1\}}$	$\varnothing$	$\{q_2\}$
$\cancel{*\{q_2\}}$	$\varnothing$	$\varnothing$
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\cancel{\varnothing}$	$\varnothing$	$\varnothing$
$\cancel{*\{q_1, q_2\}}$	$\varnothing$	$\{q_2\}$
$\cancel{*\{q_0, q_1, q_2\}}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

例 10. 设计 NFA 识别 $L = \{w \in \{0,1\}^* \mid w$ 倒数第 3 个字符是 $1\}$.

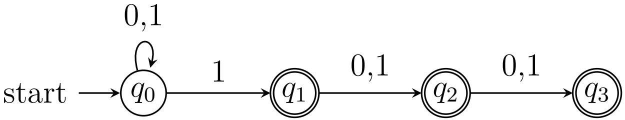
例 10. 设计 NFA 识别 $L = \{w \in \{0,1\}^* \mid w$ 倒数第 3 个字符是 $1\}$.



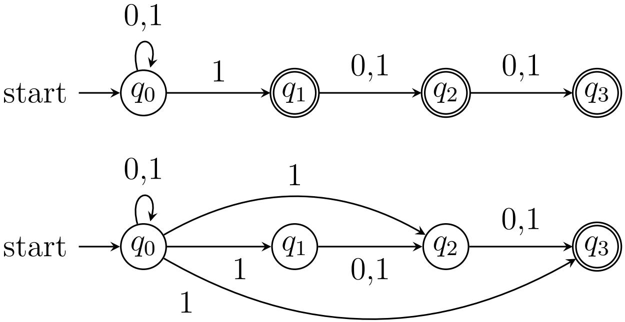
- 有穷状态系统
- 确定的有穷自动机
- 非确定有穷自动机
- 带有空转移的非确定有穷自动机
  - 形式定义
  - ε-闭包
  - 扩展转移函数
  - ε-NFA 的语言
  - ε-NFA 与 DFA 等价性

例 11. 设计 $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 NFA.

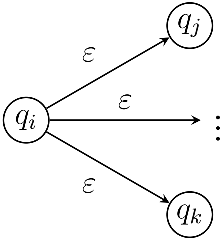
例 11. 设计 $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 NFA.



例 11. 设计 $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 NFA.

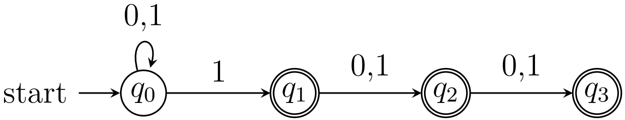


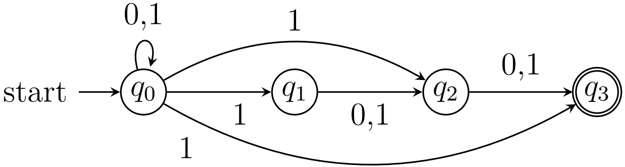
状态的 ε 转移

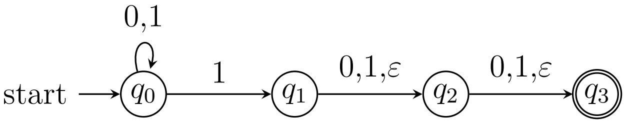


- 允许状态因空串 ε 而转移, 即不消耗输入字符就发生状态的改变
- 使自动机的设计更容易

续例 11. 设计 $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 NFA.







带空转移非确定有穷自动机的形式定义

定义

带空转移非确定有穷自动机(ε-NFA) A 为五元组

$$A = (Q, \Sigma, \delta, q_0, F)$$

1. Q : 有穷状态集;
2. Σ : 有穷输入符号集或字母表;
3. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow 2^Q$, 转移函数;
4. $q_0 \in Q$: 初始状态;
5. $F \subseteq Q$: 终结状态集或接受状态集.

ε-NFA, NFA, DFA 之间的主要区别

❶ 自动机在某状态, 读入某个字符时, 可能有多个转移;

❷ 自动机在某状态, 读入某个字符时, 可能没有转移;

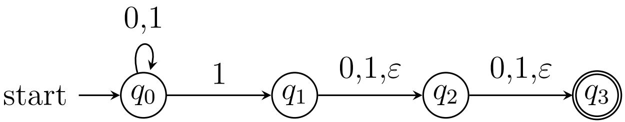
❸ 自动机在某状态, 可能不读入字符, 就进行转移.

注意

此后, 不再明确区分 ε-NFA 和 NFA, 而认为它们都是 NFA.

续例 11. $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 ε-NFA.

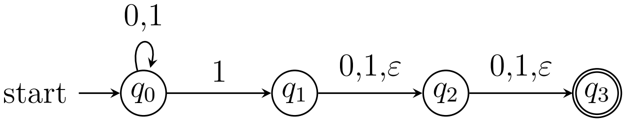
利用 ε 转移设计的有穷自动机:



状态转移表:

续例 11. $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 ε-NFA.

利用 ε 转移设计的有穷自动机:

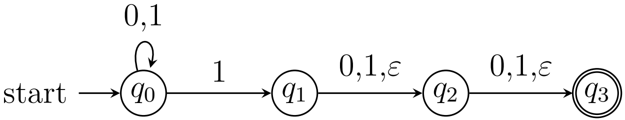


状态转移表:

	0	1	ε
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$	$\varnothing$
q_1	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$
q_2	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$
$*q_3$	$\varnothing$	$\varnothing$	$\varnothing$

续例 11. $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 ε-NFA.

利用 ε 转移设计的有穷自动机:



当输入字符串是 011 时, ε-NFA 的状态变化.

思考题

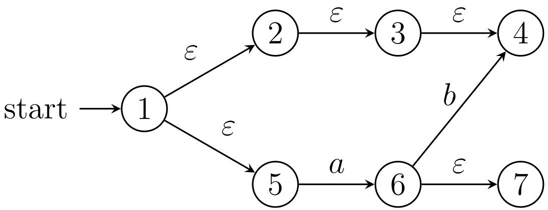
1. 如果初始状态有 ε 转移，第 1 个字符该如何处理？
2. 如果最后的字符所到的状态有 ε 转移呢？

状态的 ε-闭包

定义

状态 q 的ε-闭包(ε-*Closure*)，记为 $\textsc{Eclose}(q)$，表示从 q 经过 ε 序列可达的全部状态集合，递归定义为：

1. $q \in \textsc{Eclose}(q)$；
2. $\forall p \in \textsc{Eclose}(q)$，若 $r \in \delta(p, \varepsilon)$，则 $r \in \textsc{Eclose}(q)$.



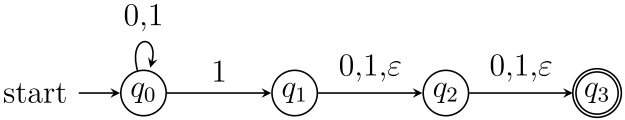
状态集合的 ε-闭包

定义

状态集 S 的 ε-闭包为

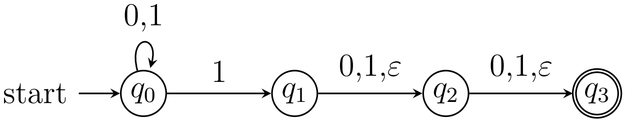
$$\textsc{Eclose}(S) = \bigcup_{q \in S} \textsc{Eclose}(q).$$

续例 11. $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 NFA.



状态转移表及每个状态的闭包:

续例 11. $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 NFA.



状态转移表及每个状态的闭包:

	0	1	ε	$\textsc{Eclose}(\textvisiblespace)$
$\rightarrow q_0$	$\{q_0\}$	$\{q_0, q_1\}$	$\varnothing$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$	$\{q_1, q_2, q_3\}$
q_2	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$	$\{q_2, q_3\}$
$*q_3$	$\varnothing$	$\varnothing$	$\varnothing$	$\{q_3\}$

扩展转移函数

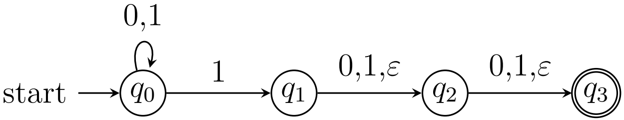
定义

扩展 δ 到字符串，定义**扩展转移函数** $\hat{\delta} : Q \times \Sigma^* \to 2^Q$ 为

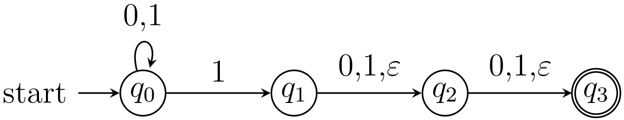
$$\hat{\delta}(q, w) = \begin{cases} \textsc{Eclose}(q) & w = \varepsilon \\ \textsc{Eclose}\left(\bigcup_{p \in \hat{\delta}(q,x)} \delta(p, a)\right) & w = xa \end{cases}$$

其中 $a \in \Sigma$，$w, x \in \Sigma^*$.

续例 11. 若 $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 ε-NFA 如下, 求 $\hat{\delta}(q_0, 10)$.



续例 11. 若 $L = \{w \in \{0,1\}^* \mid w$ 倒数 3 个字符至少有一个是 $1\}$ 的 ε-NFA 如下, 求 $\hat{\delta}(q_0, 10)$.



$$
\begin{aligned}
\hat{\delta}(q_0, \varepsilon) &= \textsc{Eclose}(q_0) = \{q_0\} \\
\hat{\delta}(q_0, 1) &= \textsc{Eclose}\big(\cup_{p \in \hat{\delta}(q_0, \varepsilon)} \delta(p, 1)\big) \\
&= \textsc{Eclose}(\hat{\delta}(q_0, 1)) = \textsc{Eclose}(\{q_0, q_1\}) = \{q_0, q_1, q_2, q_3\} \\
\hat{\delta}(q_0, 10) &= \textsc{Eclose}\big(\cup_{p \in \hat{\delta}(q_0, 1)} \delta(p, 0)\big) \\
&= \textsc{Eclose}(\delta(q_0, 0) \cup \delta(q_1, 0) \cup \delta(q_2, 0) \cup \delta(q_3, 0)) \\
&= \textsc{Eclose}(\{q_0, q_2, q_3\}) = \{q_0, q_2, q_3\}
\end{aligned}
$$

ε-NFA 的语言

回顾

DFA $D = (Q, \Sigma, \delta, q_0, F)$ 和 NFA $N = (Q, \Sigma, \delta, q_0, F)$ 的语言分别为

$$\mathbf{L}(D) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\},$$
$$\mathbf{L}(N) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \varnothing\}.$$

定义

若 $E = (Q, \Sigma, \delta, q_0, F)$ 是一个*ε-NFA*, 则 E 接受的语言为

$$\mathbf{L}(E) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \cap F \neq \varnothing\}.$$

消除空转移的子集构造法

构造方法

如果 ε-NFA $E = (Q_E, \Sigma, \delta_E, q_E, F_E)$, 构造 DFA

$$D = (Q_D, \Sigma, \delta_D, q_D, F_D)$$

❶ $Q_D = 2^{Q_E}$, 或 $Q_D = \{S \subseteq Q_E \mid S = \textsc{Eclose}(S)\}$;

❷ $q_D = \textsc{Eclose}(q_E)$;

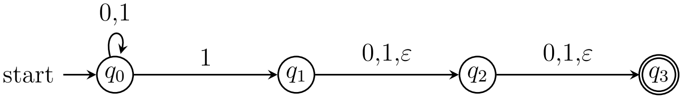
❸ $F_D = \{S \mid S \in Q_D, S \cap F_E \neq \varnothing\}$;

❹ $\forall S \in Q_D, \forall a \in \Sigma$,

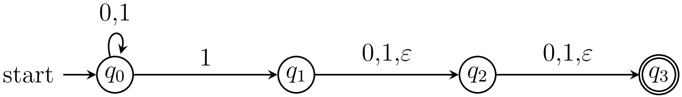
$$\delta_D(S, a) = \textsc{Eclose}\left(\bigcup_{p \in S} \delta_E(p, a)\right).$$

那么有 $\mathbf{L}(D) = \mathbf{L}(E)$.

续例 11. 将下图 L 的 ε-NFA, 转为等价的 DFA.

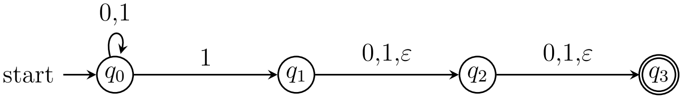


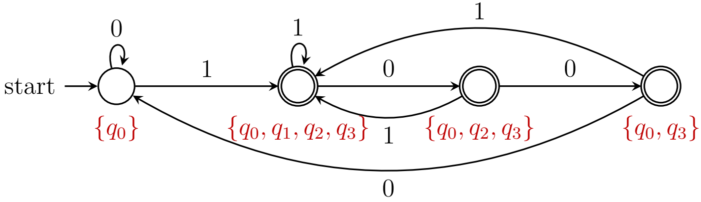
续例 11. 将下图 L 的 ε-NFA, 转为等价的 DFA.



	0	1	ε	ECLOSE()
$\to q_0$	$\{q_0\}$	$\{q_0, q_1\}$	$\varnothing$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$	$\{q_2\}$	$\{q_1, q_2, q_3\}$
q_2	$\{q_3\}$	$\{q_3\}$	$\{q_3\}$	$\{q_2, q_3\}$
$*q_3$	$\varnothing$	$\varnothing$	$\varnothing$	$\{q_3\}$

续例 11. 将下图 L 的 ε-NFA, 转为等价的 DFA.





	0	1
$\rightarrow \{q_0\}$	$\{q_0\}$	$\{q_0, q_1, q_2, q_3\}$
$*\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$*\{q_0, q_2, q_3\}$	$\{q_0, q_3\}$	$\{q_0, q_1, q_2, q_3\}$
$*\{q_0, q_3\}$	$\{q_0\}$	$\{q_0, q_1, q_2, q_3\}$

ε-NFA 与 DFA 等价性

定理 2

如果语言 L 被 ε-*NFA* 接受，当且仅当 L 被 *DFA* 接受.

证明：必要性显然成立，因为任何 DFA 都是 ε-NFA.
为证明充分性，对 w 归纳，往证 $\hat{\delta}_E(q_E, w) = \hat{\delta}_D(q_D, w)$.

❶ 当 $w = \varepsilon$ 时

$$\hat{\delta}_E(q_E, \varepsilon) = \textsc{Eclose}(q_E) = q_D = \hat{\delta}_D(q_D, \varepsilon).$$

❷ 当 $w = xa$ 时

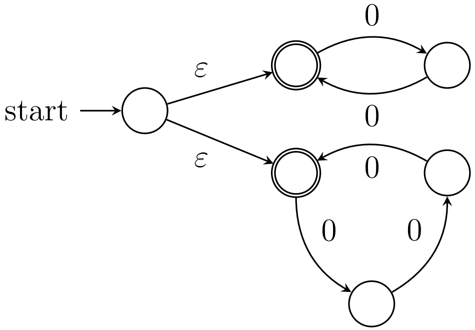
$$\hat{\delta}_E(q_E, xa) = \textsc{Eclose}\Big(\bigcup_{p \in \hat{\delta}_E(q_E, x)} \delta_E(p, a)\Big) = \textsc{Eclose}\Big(\bigcup_{p \in \hat{\delta}_D(q_D, x)} \delta_E(p, a)\Big)$$

$$= \delta_D(\hat{\delta}_D(q_D, x), a) = \hat{\delta}_D(q_D, xa)$$

$\square$

例 12. Design ε-NFA for language: $\{0^k \mid k \text{ is a multiple of 2 or 3}\}$.

例 12. Design ε-NFA for language: $\{0^k \mid k \text{ is a multiple of 2 or 3}\}$.



形式语言与自动机理论

正则表达式

王春宇

计算机科学与技术学院
哈尔滨工业大学

正则表达式

- 正则表达式
  - 语言的运算
  - 正则表达式的递归定义
  - 运算符的优先级
  - 正则表达式示例
- 有穷自动机和正则表达式
- 正则表达式的代数定律

正则表达式

- 有穷自动机
  - 通过机器装置描述正则语言
  - 用计算机编写相应算法，易于实现
- 正则表达式
  - 通过表达式描述正则语言，代数表示方法，使用方便
  - 应用广泛
    - `grep` 工具 (Global Regular Expression and Print)
    - Emacs / Vim 文本编辑器
    - `lex` / `flex` 词法分析器
    - 各种程序设计语言 Python / Perl / Haskull / $\cdots$

语言的运算

设 L 和 M 是两个语言，那么

并 $$L \cup M = \{w \mid w \in L \text{ 或 } w \in M\}$$

连接 $$L \cdot M = \{w \mid w = xy,\ x \in L \text{ 且 } y \in M\}$$

幂
$$L^0 = \{\varepsilon\}$$
$$L^1 = L$$
$$L^n = L^{n-1} \cdot L$$

克林闭包 $$L^* = \bigcup_{i=0}^{\infty} L^i$$

例 1. 若有语言 $L = \{0, 11\}$ 和 $M = \{\varepsilon, 001\}$, 那么

$$L \cup M = \qquad\qquad L^0 =$$

$$LM = \qquad\qquad L^1 =$$

$$ML = \qquad\qquad L^2 =$$

例 2. 对于空语言 $\varnothing$

$$\varnothing^0 =$$

$$\forall n \geq 1, \quad \varnothing^n =$$

$$\varnothing^* =$$

四则运算表达式的递归定义:

1. 任何数都是四则运算表达式;
2. 如果 a 和 b 是四则运算表达式, 那么

$$a+b,\ a-b,\ a\times b,\ a\div b \text{和} (a)$$

   都是四则运算表达式.

正则表达式的递归定义

定义

如果 Σ 为字母表，则 Σ 上的正则表达式递归定义为：

1. $\emptyset$ 是一个正则表达式，表示空语言；
   $\boldsymbol{\varepsilon}$ 是一个正则表达式，表示语言 $\{\varepsilon\}$；
   $\forall a \in \Sigma$，$\mathbf{a}$ 是一个正则表达式，表示语言 $\{a\}$；
2. 如果正则表达式 $\mathbf{r}$ 和 $\mathbf{s}$ 分别表示语言 R 和 S，那么

$$\mathbf{r}+\mathbf{s},\ \ \mathbf{rs},\ \ \mathbf{r}^*\ \text{和}\ (\mathbf{r})$$

   都是正则表达式，分别表示语言

$$R \cup S,\ \ R \cdot S,\ \ R^*\ \text{和}\ R.$$

运算符的优先级

正则表达式中三种运算以及括号的优先级:

❶ 首先,“括号”优先级最高;

❷ 其次,“星”运算: $\mathbf{r}^*$;

❸ 然后,“连接”运算: $\mathbf{rs}$, $\mathbf{r}\cdot\mathbf{s}$;

❹ 最后,“加”最低: $\mathbf{r}+\mathbf{s}$, $\mathbf{r}\cup\mathbf{s}$;

例 3.

$$\begin{aligned}\mathbf{1}+\mathbf{0}\mathbf{1}^* &= \mathbf{1}+(\mathbf{0}(\mathbf{1}^*)) \\ &\neq \mathbf{1}+(\mathbf{0}\mathbf{1})^* \\ &\neq (\mathbf{1}+\mathbf{0}\mathbf{1})^* \\ &\neq (\mathbf{1}+\mathbf{0})\mathbf{1}^*\end{aligned}$$

正则表达式示例

例 4.

E	$\mathbf{L}(E)$
$\mathbf{a}+\mathbf{b}$	$\mathbf{L}(\mathbf{a})\cup\mathbf{L}(\mathbf{b})=\{a\}\cup\{b\}=\{a,b\}$
$\mathbf{bb}$	$\mathbf{L}(\mathbf{b})\cdot\mathbf{L}(\mathbf{b})=\{b\}\cdot\{b\}=\{bb\}$
$(\mathbf{a}+\mathbf{b})(\mathbf{a}+\mathbf{b})$	$\{a,b\}\{a,b\}=\{aa,ab,ba,bb\}$
$(\mathbf{a}+\mathbf{b})^*(\mathbf{a}+\mathbf{bb})$	$\{a,b\}^*\{a,bb\}=\{a,b\}^*\{a\}\cup\{a,b\}^*\{bb\}=$ $\{w\in\{a,b\}^*\mid w$ 仅以 a 或 bb 结尾.$\}$
$\mathbf{1}+(\mathbf{01})^*$	$\{1,\varepsilon,01,0101,010101,\ldots\}$
$(\mathbf{0}+\mathbf{1})^*\mathbf{01}(\mathbf{0}+\mathbf{1})^*$	$\{x01y\mid x,y\in\{0,1\}^*\}$

例 5. 给出正则表达式 $(\mathbf{aa})^*(\mathbf{bb})^*\mathbf{b}$ 定义的语言.

$$
\begin{aligned}
\mathbf{L}((\mathbf{aa})^*(\mathbf{bb})^*\mathbf{b}) &= \mathbf{L}((\mathbf{aa})^*) \cdot \mathbf{L}((\mathbf{bb})^*) \cdot \mathbf{L}(\mathbf{b}) \\
&= (\{a\}\{a\})^*(\{b\}\{b\})^*\{b\} \\
&= \{a^2\}^*\{b^2\}^*\{b\} \\
&= \{a^{2n}b^{2m+1} \mid n \geq 0, m \geq 0\}
\end{aligned}
$$

例 6. Design regular expression for $L = \{w \mid w$ consists of 0's and 1's, and the third symbol from the right end is 1.$\}$

$$(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})(\mathbf{0}+\mathbf{1})$$

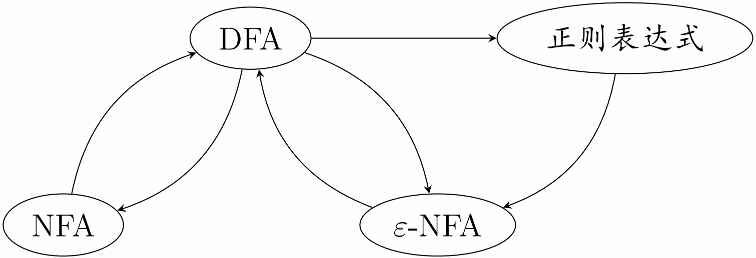
例 7. Design regular expression for
$L = \{w \mid w \in \{0,1\}^* \text{ and } w \text{ has no pair of consecutive 0's.}\}$

$$\mathbf{1}^*(\mathbf{011}^*)^*(\mathbf{0}+\varepsilon) \text{ 或 } (\mathbf{1}+\mathbf{01})^*(\mathbf{0}+\varepsilon)$$

例. Design regular expression for $L = \{w \mid w \in \{0,1\}^* \text{ and } w \text{ contains } 01\}$.
例. Write a regular expression for $L = \{w \in \{0,1\}^* \mid 0 \text{ and } 1 \text{ alternate in } w\}$.
例. Find a regular expression for the set $\{a^n b^m \mid (n+m) \text{ is odd }\}$.
例. Give regular expression for the complement of
$L = \{a^n b^m \mid n \geq 3, m \leq 4\}$.
例. Write a regular expression for the set of all C real numbers.

- 正则表达式
- 有穷自动机和正则表达式
  - 由 DFA 到正则表达式, 递归表达式法
  - 由 DFA 到正则表达式, 状态消除法
  - 由正则表达式到 ε-NFA
- 正则表达式的代数定律

DFA, NFA, ε-NFA 和正则表达式的等价性



由 DFA 到正则表达式, 递归表达式法

定理 3

若 $L = \mathbf{L}(A)$ 是某 DFA A 的语言, 那么存在正则表达式 R 满足 $L = \mathbf{L}(R)$.

由 DFA 到正则表达式, 递归表达式法

定理 3

若 $L = \mathbf{L}(A)$ 是某 *DFA* A 的语言, 那么存在正则表达式 R 满足 $L = \mathbf{L}(R)$.

证明: 对 DFA A 的状态编号, 令 1 为开始状态, 即

$$A = (\{1, 2, \dots, n\}, \Sigma, \delta, 1, F),$$

设正则表达式 $R_{ij}^{(k)}$ 表示从 i 到 j 但中间节点不超过 k 全部路径的字符串集:

$$R_{ij}^{(k)} = \{x \mid \hat{\delta}(i, x) = j, x \text{ 经过的状态除两端外都不超过 } k\}.$$

那么与 $A=(\{1,2,\ldots,n\},\Sigma,\delta,1,F)$ 等价的正则表达式为

$$\bigcup_{j\in F} R_{1j}^{(n)}$$

且递归式为

$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$$

$$R_{ij}^{(0)} = \begin{cases} \{a \mid \delta(q_i,a)=q_j\} & i\neq j \\ \{a \mid \delta(q_i,a)=q_j\}\cup\{\varepsilon\} & i=j \end{cases}$$

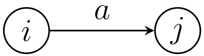
下面对 k 归纳, 证明可用以上递归式求得 $R_{ij}^{(k)}$.

归纳基础: 当 $i \neq j, k = 0$ 时, 即 i 到 j 没经过任何中间节点

- 没有 i 到 j 的状态转移

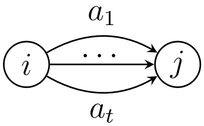
$$R_{ij}^{(0)} = \varnothing$$

- 有一个 i 到 j 的状态转移



$$R_{ij}^{(0)} = \mathbf{a}$$

- 有多个 i 到 j 的状态转移



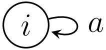
$$R_{ij}^{(0)} = \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_t$$

归纳基础 (续): 当 $i=j, k=0$ 时, 即从 i 到自身没经过任何中间节点

- 状态 i 没有到自己的转移

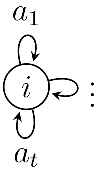
$$R_{ii}^{(0)} = \boldsymbol{\varepsilon}$$

- 状态 i 有一个到自身的转移



$$R_{ii}^{(0)} = \mathbf{a} + \boldsymbol{\varepsilon}$$

- 状态 i 有多个到自身的转移

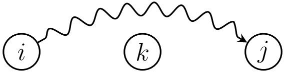


$$R_{ii}^{(0)} = \mathbf{a}_1 + \mathbf{a}_2 + \cdots + \mathbf{a}_t + \boldsymbol{\varepsilon}$$

归纳假设: 假设已知 $R_{ij}^{(k-1)}$, $R_{ik}^{(k-1)}$, $R_{kk}^{(k-1)}$ 和 $R_{kj}^{(k-1)}$.

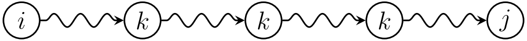
归纳递推: 那么 $R_{ij}^{(k)}$ 中全部路径, 可用节点 k 分为两部分

- 从 i 到 j 不经过 k 的



$$R_{ij}^{(k)} = R_{ij}^{(k-1)}$$

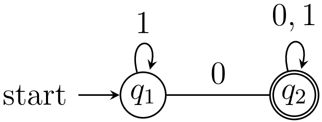
- 从 i 到 j 经过 k 的



$$R_{ij}^{(k)} = R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$$

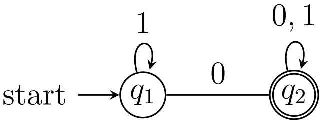
因此 $R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^* R_{kj}^{(k-1)}$. □

例 8. 将如图 DFA 转换为正则表达式.



- 计算 $R_{ij}^{(0)}$

例 8. 将如图 DFA 转换为正则表达式.



- 计算 $R_{ij}^{(0)}$

$R_{ij}^{(k)}$	$k=0$
$R_{11}^{(0)}$	$\boldsymbol{\varepsilon}+\mathbf{1}$
$R_{12}^{(0)}$	$\mathbf{0}$
$R_{21}^{(0)}$	$\varnothing$
$R_{22}^{(0)}$	$\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1}$

- 计算 $R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^* R_{1j}^{(0)}$

$R_{ij}^{(k)}$	$k=0$
$R_{11}^{(0)}$	$\boldsymbol{\varepsilon} + \mathbf{1}$
$R_{12}^{(0)}$	$\mathbf{0}$
$R_{21}^{(0)}$	$\varnothing$
$R_{22}^{(0)}$	$\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1}$

- 计算 $R_{ij}^{(1)} = R_{ij}^{(0)} + R_{i1}^{(0)}(R_{11}^{(0)})^* R_{1j}^{(0)}$

$R_{ij}^{(k)}$	$k=0$
$R_{11}^{(0)}$	$\boldsymbol{\varepsilon}+\mathbf{1}$
$R_{12}^{(0)}$	$\mathbf{0}$
$R_{21}^{(0)}$	$\varnothing$
$R_{22}^{(0)}$	$\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1}$

$R_{ij}^{(k)}$	$k=1$
$R_{11}^{(1)}$	$(\boldsymbol{\varepsilon}+\mathbf{1})+(\boldsymbol{\varepsilon}+\mathbf{1})(\boldsymbol{\varepsilon}+\mathbf{1})^*(\boldsymbol{\varepsilon}+\mathbf{1})$
$R_{12}^{(1)}$	$\mathbf{0}+(\boldsymbol{\varepsilon}+\mathbf{1})(\boldsymbol{\varepsilon}+\mathbf{1})^*\mathbf{0}$
$R_{21}^{(1)}$	$\varnothing+\varnothing(\boldsymbol{\varepsilon}+\mathbf{1})^*(\boldsymbol{\varepsilon}+\mathbf{1})$
$R_{22}^{(1)}$	$\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1}+\varnothing(\boldsymbol{\varepsilon}+\mathbf{1})^*\mathbf{0}$

- 几个基本的化简规则

  如果 $\mathbf{r}$ 和 $\mathbf{s}$ 是两个正则表达式

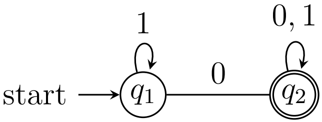
$$\begin{aligned}
(\boldsymbol{\varepsilon}+\mathbf{r})^* &= \mathbf{r}^* \\
(\boldsymbol{\varepsilon}+\mathbf{r})\mathbf{r}^* &= \mathbf{r}^* \\
\mathbf{r}+\mathbf{r}\mathbf{s}^* &= \mathbf{r}\mathbf{s}^* \\
\varnothing\mathbf{r} &= \mathbf{r}\varnothing = \varnothing && \text{(零元)} \\
\varnothing+\mathbf{r} &= \mathbf{r}+\varnothing = \mathbf{r} && \text{(单位元)}
\end{aligned}$$

- 化简 $R_{ij}^{(1)}$

$R_{ij}^{(k)}$	$k=1$	化简
$R_{11}^{(1)}$	$(\boldsymbol{\varepsilon}+\mathbf{1})+(\boldsymbol{\varepsilon}+\mathbf{1})(\boldsymbol{\varepsilon}+\mathbf{1})^*(\boldsymbol{\varepsilon}+\mathbf{1})$	$\mathbf{1}^*$
$R_{12}^{(1)}$	$\mathbf{0}+(\boldsymbol{\varepsilon}+\mathbf{1})(\boldsymbol{\varepsilon}+\mathbf{1})^*\mathbf{0}$	$\mathbf{1}^*\mathbf{0}$
$R_{21}^{(1)}$	$\varnothing+\varnothing(\boldsymbol{\varepsilon}+\mathbf{1})^*(\boldsymbol{\varepsilon}+\mathbf{1})$	$\varnothing$
$R_{22}^{(1)}$	$\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1}+\varnothing(\boldsymbol{\varepsilon}+\mathbf{1})^*\mathbf{0}$	$\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1}$

- 计算 $R_{ij}^{(2)} = R_{ij}^{(1)} + R_{i2}^{(1)}(R_{22}^{(1)})^* R_{2j}^{(1)}$

$R_{ij}^{(k)}$	$k=1$	$R_{ij}^{(k)}$	$k=2$
$R_{11}^{(1)}$	$\mathbf{1}^*$	$R_{11}^{(2)}$	$\mathbf{1}^* + \mathbf{1}^*\mathbf{0}(\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})^*\varnothing$
$R_{12}^{(1)}$	$\mathbf{1}^*\mathbf{0}$	$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0} + \mathbf{1}^*\mathbf{0}(\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})^*(\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})$
$R_{21}^{(1)}$	$\varnothing$	$R_{21}^{(2)}$	$\varnothing + (\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})(\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})^*\varnothing$
$R_{22}^{(1)}$	$\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1}$	$R_{22}^{(2)}$	$\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1} + (\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})(\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})^*(\boldsymbol{\varepsilon}+\mathbf{0}+\mathbf{1})$



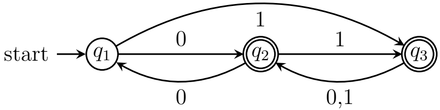
- 化简 $R_{ij}^{(2)}$

$R_{ij}^{(k)}$	$k=2$	化简
$R_{11}^{(2)}$	$\mathbf{1}^* + \mathbf{1}^*\mathbf{0}(\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})^*\varnothing$	$\mathbf{1}^*$
$R_{12}^{(2)}$	$\mathbf{1}^*\mathbf{0} + \mathbf{1}^*\mathbf{0}(\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})^*(\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})$	$\mathbf{1}^*\mathbf{0}(\mathbf{0} + \mathbf{1})^*$
$R_{21}^{(2)}$	$\varnothing + (\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})(\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})^*\varnothing$	$\varnothing$
$R_{22}^{(2)}$	$\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1} + (\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})(\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})^*(\boldsymbol{\varepsilon} + \mathbf{0} + \mathbf{1})$	$(\mathbf{0} + \mathbf{1})^*$

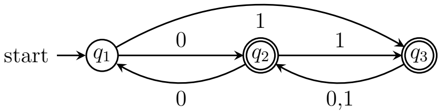
- 因只有 q_2 是接受状态, 所以该 DFA 正则表达式为

$$R_{12}^{(2)} = \mathbf{1}^*\mathbf{0}(\mathbf{0} + \mathbf{1})^*.$$

例 9. 将如图 DFA 转换为正则表达式.

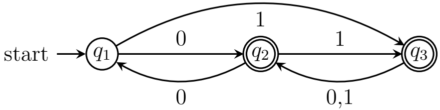


例 9. 将如图 DFA 转换为正则表达式.



	$k=0$	$k=1$	$k=2$
$R_{11}^{(k)}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	$(\mathbf{00})^*$
$R_{12}^{(k)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}(\mathbf{00})^*$
$R_{13}^{(k)}$	$\mathbf{1}$	$\mathbf{1}$	$\mathbf{0}^*\mathbf{1}$
$R_{21}^{(k)}$	$\mathbf{0}$	$\mathbf{0}$	$\mathbf{0}(\mathbf{00})^*$
$R_{22}^{(k)}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}+\mathbf{00}$	$(\mathbf{00})^*$
$R_{23}^{(k)}$	$\mathbf{1}$	$\mathbf{1}+\mathbf{01}$	$\mathbf{0}^*\mathbf{1}$
$R_{31}^{(k)}$	$\varnothing$	$\varnothing$	$(\mathbf{0}+\mathbf{1})(\mathbf{00})^*\mathbf{0}$
$R_{32}^{(k)}$	$\mathbf{0}+\mathbf{1}$	$\mathbf{0}+\mathbf{1}$	$(\mathbf{0}+\mathbf{1})(\mathbf{00})^*$
$R_{33}^{(k)}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}+(\mathbf{0}+\mathbf{1})\mathbf{0}^*\mathbf{1}$

例 9. 将如图 DFA 转换为正则表达式.



仅状态 2 和 3 是接受状态:

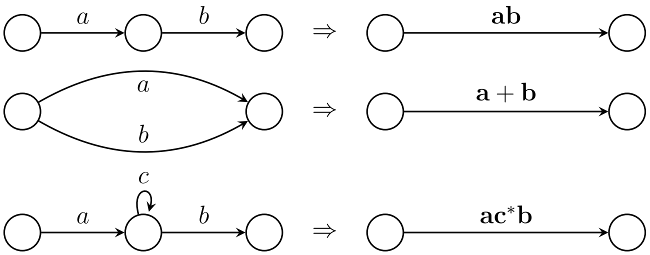
$$\begin{aligned}
R_{12}^{(3)} &= R_{12}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^* R_{32}^{(2)} \\
&= \mathbf{0}(\mathbf{00})^* + \mathbf{0}^*\mathbf{1}(\boldsymbol{\varepsilon} + (\mathbf{0}+\mathbf{1})\mathbf{0}^*\mathbf{1})^*(\mathbf{0}+\mathbf{1})(\mathbf{00})^* \\
&= \mathbf{0}(\mathbf{00})^* + \mathbf{0}^*\mathbf{1}((\mathbf{0}+\mathbf{1})\mathbf{0}^*\mathbf{1})^*(\mathbf{0}+\mathbf{1})(\mathbf{00})^*
\end{aligned}$$

$$\begin{aligned}
R_{13}^{(3)} &= R_{13}^{(2)} + R_{13}^{(2)}(R_{33}^{(2)})^* R_{33}^{(2)} \\
&= \mathbf{0}^*\mathbf{1} + \mathbf{0}^*\mathbf{1}(\boldsymbol{\varepsilon} + (\mathbf{0}+\mathbf{1})\mathbf{0}^*\mathbf{1})^*(\boldsymbol{\varepsilon} + (\mathbf{0}+\mathbf{1})\mathbf{0}^*\mathbf{1}) \\
&= \mathbf{0}^*\mathbf{1}((\mathbf{0}+\mathbf{1})\mathbf{0}^*\mathbf{1})^*
\end{aligned}$$

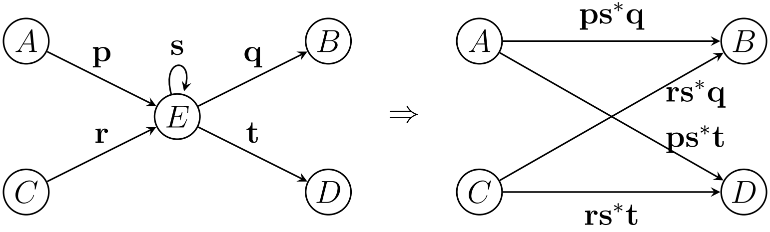
$$R_{12}^{(3)} + R_{13}^{(3)} = \mathbf{0}^*\mathbf{1}((\mathbf{0}+\mathbf{1})\mathbf{0}^*\mathbf{1})^*(\boldsymbol{\varepsilon} + (\mathbf{0}+\mathbf{1})(\mathbf{00})^*) + \mathbf{0}(\mathbf{00})^*.$$

由 DFA 到正则表达式, 状态消除法

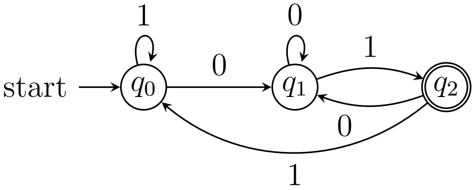
- 从 DFA 中逐个删除状态
- 用标记了正则表达式的新路径替换被删掉的路径
- 保持“自动机”等价.



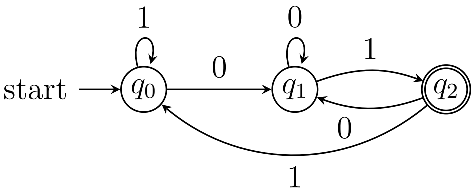
- 更一般的情况如图
- 若要删除状态 E, 需添加相应路径



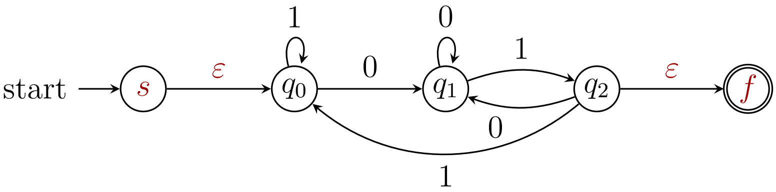
例 10. 利用状态消除法, 设计下图自动机的正则表达式.



例 10. 利用状态消除法, 设计下图自动机的正则表达式.

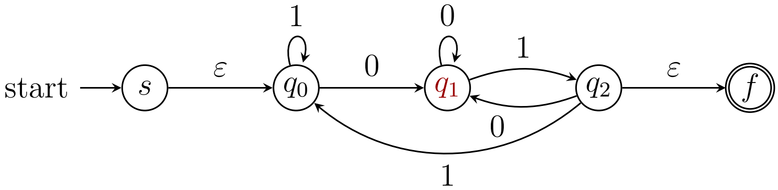


❶ 利用空转移, 添加新的开始 s 和结束状态 f:

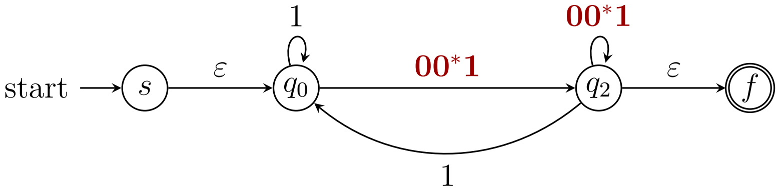


例 10. 利用状态消除法, 设计下图自动机的正则表达式.

❶ 利用空转移, 添加新的开始 s 和结束状态 f:

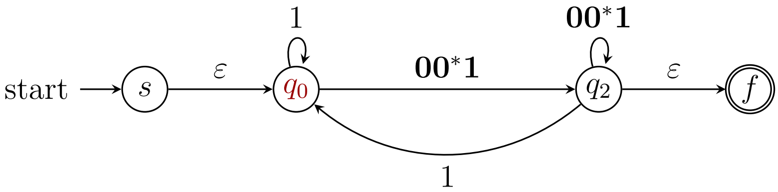


❷ 消除状态 q_1, 添加路径 $q_0 \to q_2$ 和 $q_2 \to q_2$:

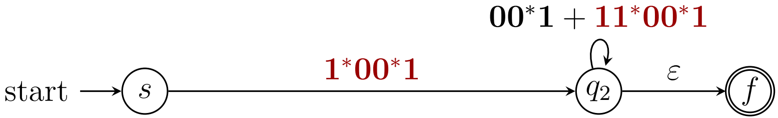


例 10. 利用状态消除法, 设计下图自动机的正则表达式.

❷ 消除状态 q_1, 添加路径 $q_0 \to q_2$ 和 $q_2 \to q_2$:

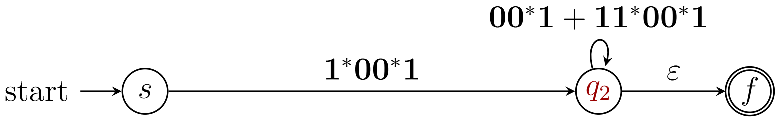


❸ 消除状态 q_0, 添加路径 $s \to q_2$ 和 $q_2 \to q_2$:

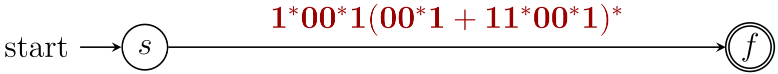


例 10. 利用状态消除法, 设计下图自动机的正则表达式.

❸ 消除状态 q_0, 添加路径 $s \to q_2$ 和 $q_2 \to q_2$:

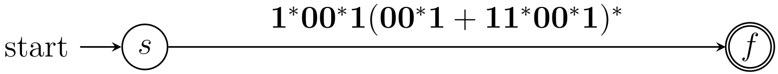


❹ 消除状态 q_2, 添加路径 $s \to f$:



例 10. 利用状态消除法, 设计下图自动机的正则表达式.

❹ 消除状态 q_2, 添加路径 $s \to f$:



❺ 因此该自动机的正则表达式为

$$\mathbf{1^*00^*1(00^*1+11^*00^*1)^*}.$$

由正则表达式到有穷自动机

定理 4

正则表达式定义的语言，都可被有穷自动机识别.

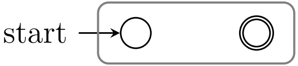
由正则表达式构造 ε-NFA

任何正则表达式 $\mathbf{r}$, 都存在等价的 ε-NFA A, 即 $\mathbf{L}(A) = \mathbf{L}(\mathbf{r})$, 并且 A 满足:

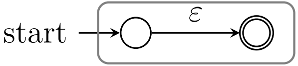
1. 仅有一个接收状态;
2. 没有进入开始状态的边;
3. 没有离开接受状态的边.

证明: 归纳基础:

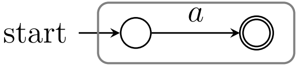
❶ 对于 $\emptyset$, 有 ε-NFA:



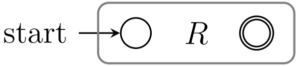
❷ 对于 $\boldsymbol{\varepsilon}$, 有 ε-NFA:



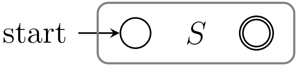
❸ $\forall a \in \Sigma$, 对于 $\mathbf{a}$, 有 ε-NFA:



归纳递推: 假设正则表达式 $\mathbf{r}$ 和 $\mathbf{s}$ 的 ε-NFA 分别为 R 和 S

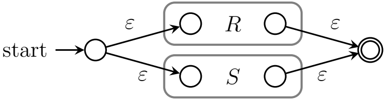


和

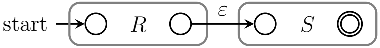


那么正则表达式 $\mathbf{r}+\mathbf{s}$, $\mathbf{rs}$ 和 $\mathbf{r}^*$, 可由 R 和 S 分别构造如下:

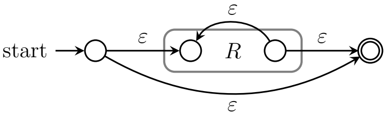
❶ 对于 $\mathbf{r}+\mathbf{s}$, 有 ε-NFA:



❷ 对于 $\mathbf{rs}$, 有 ε-NFA:

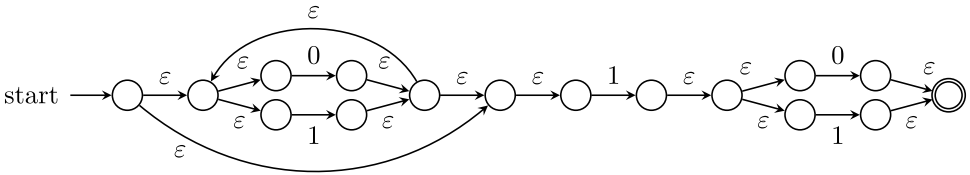


❸ 对于 $\mathbf{r}^*$, 有 ε-NFA:



因此任何结构的正则表达式, 都可递归构造出等价的 ε-NFA. □

例 11. 正则表达式 $(\mathbf{0}+\mathbf{1})^*\mathbf{1}(\mathbf{0}+\mathbf{1})$ 构造为 ε-NFA.



思考题

正则表达式到 ε-NFA 构造方法中的 3 个限制条件, 都有必要吗?

正则表达式

- 正则表达式
- 有穷自动机和正则表达式
- 正则表达式的代数定律
  - 基本的代数定律
  - 发现与验证代数定律

正则表达式的代数定律

定义

含有变量的两个正则表达式，如果以任意语言替换其变量，二者所表示的语言仍然相同，则称这两个正则表达式等价. 在这样的意义下，正则表达式满足一些代数定律.

- 并运算

$$
\begin{aligned}
(L+M)+N &= L+(M+N) && \text{(结合律)} \\
L+M &= M+L && \text{(交换律)} \\
L+L &= L && \text{(幂等律)} \\
\varnothing+L &= L+\varnothing = L && \text{(单位元 } \varnothing\text{)}
\end{aligned}
$$

- 连接运算

$$
\begin{aligned}
(LM)N &= L(MN) && \text{(结合律)} \\
\varepsilon L &= L\varepsilon = L && \text{(单位元 }\varepsilon\text{)} \\
\varnothing L &= L\varnothing = \varnothing && \text{(零元 }\varnothing\text{)} \\
LM &\neq ML
\end{aligned}
$$

- 分配率

$$
\begin{aligned}
L(M+N) &= LM + LN && \text{(左分配律)} \\
(M+N)L &= ML + NL && \text{(右分配律)}
\end{aligned}
$$

- 闭包运算

$$
\begin{aligned}
(L^*)^* &= L^* \\
\varnothing^* &= \boldsymbol{\varepsilon} \\
\boldsymbol{\varepsilon}^* &= \boldsymbol{\varepsilon} \\
L^* &= L^+ + \boldsymbol{\varepsilon} \\
(\boldsymbol{\varepsilon} + L)^* &= L^*
\end{aligned}
$$

发现与验证正则表达式的代数定律

检验方法

要判断表达式 E 和 F 是否等价，其中变量为 $L_1, \ldots, L_n$:

1. 将变量替换为具体表达式，得正则表达式 $\mathbf{r}$ 和 $\mathbf{s}$，
   例如，替换 L_i 为 $\mathbf{a}_i$;
2. 判断 $\mathbf{L}(\mathbf{r}) \stackrel{?}{=} \mathbf{L}(\mathbf{s})$，如果相等则 $E = F$，否则 $E \neq F$.

例 12. 判断 $(L+M)^* = (L^*M^*)^*$.

❶ 将 L 和 M 替换为 $\mathbf{a}$ 和 $\mathbf{b}$;

❷ $(\mathbf{a}+\mathbf{b})^* \stackrel{?}{=} (\mathbf{a}^*\mathbf{b}^*)^*$;

❸ 因为 $\mathbf{L}((\mathbf{a}+\mathbf{b})^*) = \mathbf{L}((\mathbf{a}^*\mathbf{b}^*)^*)$;

❹ 所以 $(L+M)^* = (L^*M^*)^*$.

例 13. 判断 $L + ML = (L + M)L$.

❶ 将 L 和 M 替换为 $\mathbf{a}$ 和 $\mathbf{b}$;

❷ 判断 $\mathbf{a} + \mathbf{ba} \stackrel{?}{=} (\mathbf{a} + \mathbf{b})\mathbf{a}$;

❸ 因为 $aa \notin \mathbf{a} + \mathbf{ba}$ 而 $aa \in (\mathbf{a} + \mathbf{b})\mathbf{a}$;

❹ 所以 $\mathbf{a} + \mathbf{ba} \neq (\mathbf{a} + \mathbf{b})\mathbf{a}$;

❺ 即 $L + ML \neq (L + M)L$.

注意

这种方法仅限于判断正则表达式，否则可能会发生错误.

例 14. 若用此方法判断 $L \cap M \cap N \stackrel{?}{=} L \cap M$, 以 $\mathbf{a}, \mathbf{b}, \mathbf{c}$ 替换 L, M, N, 有

$$\{a\} \cap \{b\} \cap \{c\} = \varnothing = \{a\} \cap \{b\},$$

而显然

$$L \cap M \cap N \neq L \cap M.$$

形式语言与自动机理论

正则语言的性质

王春宇

计算机科学与技术学院
哈尔滨工业大学

正则语言的性质

- 证明语言的非正则性
  - 正则语言的泵引理
  - 泵引理的应用
  - 泵引理只是必要条件
- 正则语言的封闭性
- 正则语言的判定性质
- 自动机的最小化

例 1. $L = \{0^m 1^n \mid m, n \geq 0\}$ 是否是正则语言?

例 2. $L = \{0^m 1^n \mid m \geq 2, n \geq 4\}$ 是否是正则语言?

例 3. $L_{01} = \{0^n 1^n \mid n \geq 0\}$ 是否是正则语言?

正则语言的泵引理

定理 5 (正则语言的泵引理)

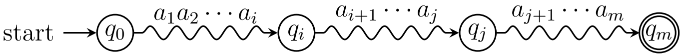
如果语言 L 是正则的，那么存在正整数 N，它只依赖于 L，对 $\forall w \in L$，只要 $|w| \geq N$，就可以将 w 分为三部分 $w = xyz$ 满足:

1. $y \neq \varepsilon$ $(|y| > 0)$;
2. $|xy| \leq N$;
3. $\forall k \geq 0,\ xy^k z \in L$.

证明:

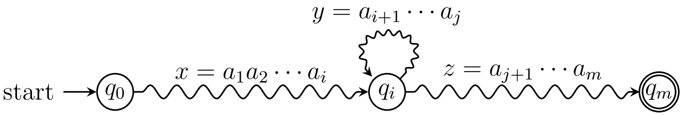
❶ 如果 L 正则, 那么存在有 n 个状态 DFA A 使 $\mathbf{L}(A) = L$;

❷ 取 $w = a_1 \dots a_m \in L$ $(m \geq n)$, 定义 $q_i = \hat{\delta}(q_0, a_1 \dots a_i)$;



❸ 由鸽巢原理, 必有两状态相同 $q_i = q_j$ $(0 \leq i < j \leq n)$;

❹ 那么 $w = xyz$ 如图, 且有 $\forall k > 0,\ xy^k z \in L$;



❺ 而因为 $i < j$ 所以 $y \neq \varepsilon$ (即 $|y| > 0$), 因为 $j \leq n$ 所以 $|xy| \leq n$. $\square$

泵引理的应用

续例 3. 证明 $L_{01} = \{0^n 1^n \mid n \geq 0\}$ 不是正则语言.

证明:

❶ 假设 L_{01} 是正则的.

❷ 那么, 存在 $N \in \mathbb{Z}^+$, 对 $\forall w \in L_{01}\,(|w| \geq N)$ 满足泵引理.

❸ 从 L_{01} 中取 $w = 0^N 1^N$, 显然 $w \in L_{01}$ 且 $|w| = 2N \geq N$.

❹ 那么, w 可被分为 $w = xyz$, 且 $|xy| \leq N$ 和 $y \neq \varepsilon$.

❺ 因此 y 只能是 0^m 且 $m > 0$.

❻ 那么 $xy^2z = 0^{N+m}1^N \notin L_{01}$, 而由泵引理 $xy^2z \in L_{01}$, 矛盾.

❼ 所以假设不成立, L_{01} 不是正则的. □

例 4. 证明 $L_{\text{eq}} = \{w \in \{0,1\}^* \mid w \text{ 由数量相等的 } 0 \text{ 和 } 1 \text{ 构成}\}$ 不是正则的.

思考题

刚刚已经证明了

$$L_{01} = \{0^n 1^n \mid n \geq 0\}$$

不是正则语言, 那么能否使用

$$L_{01} \subseteq L_{\text{eq}}$$

来说明 L_{eq} 也不是正则的呢?

续例 4. 证明 $L_{\text{eq}} = \{w \in \{0,1\}^* \mid w$ 由数量相等的 0 和 1 构成$\}$ 不是正则的.

证明:

❶ 假设 L_{eq} 是正则的.

❷ 那么, 存在 $N \in \mathbb{Z}^+$, 对 $\forall w \in L_{\text{eq}}\,(|w| \geq N)$ 满足泵引理.

❸ 从 L_{eq} 中取 $w = 0^N 1^N$, 显然 $w \in L_{\text{eq}}$ 且 $|w| = 2N \geq N$.

❹ 那么, w 可被分为 $w = xyz$, 且 $|xy| \leq N$ 和 $y \neq \varepsilon$.

❺ 因此 y 只能是 0^m 且 $m > 0$.

❻ 那么 $xy^2z = 0^{N+m}1^N \notin L_{\text{eq}}$, 而由泵引理 $xy^2z \in L_{\text{eq}}$, 矛盾.

❼ 所以假设不成立, L_{eq} 不是正则的. □

例 5. 证明 $L = \{0^i1^j \mid i > j\}$ 不是正则的.

证明:

❶ 假设 L 是正则的.

❷ 那么, 存在 $N \in \mathbb{Z}^+$, 对 $\forall w \in L\,(|w| \geq N)$ 满足泵引理.

❸ 从 L 中取 $w = 0^{N+1}1^N$, 则 $w \in L$ 且 $|w| = 2N + 1 \geq N$.

❹ 由泵引理, w 可被分为 $w = xyz$, 且 $|xy| \leq N$ 和 $y \neq \varepsilon$.

❺ 那么, y 只能是 0^m 且 $m \geq 1$.

❻ 那么, $xz = xy^0z = 0^{N+1-m}1^N \notin L$, 因为 $N + 1 - m \leq N$,
而由泵引理 $xy^0z \in L$, 矛盾.

❼ 所以假设不成立, L 不是正则的. $\square$

例 6. Prove $L = \{a^3 b^n c^{n-3} \mid n \geq 3\}$ is not regular.

证明:

❶ 假设 L 是正则的.

❷ 那么, 存在 $N \in \mathbb{Z}^+$, 对 $\forall w \in L\,(|w| \geq N)$ 满足泵引理.

❸ 从 L 中取 $w = a^3 b^N c^{N-3}$, 则 $w \in L$ 且 $|w| = 2N \geq N$.

❹ 由泵引理, w 可被分为 $w = xyz$, 且 $|xy| \leq N$ 和 $y \neq \varepsilon$.

❺ 那么, 则 y 只可能有 3 种情况 $(m > 0, r > 0, s > 0)$:

- ❶ $y = a^m$, 则 $xy^2z = a^{3+m} b^N c^{N-3} \notin L$;
- ❷ $y = b^m$, 则 $xy^2z = a^3 b^{N+m} c^{N-3} \notin L$;
- ❸ $y = a^r b^s$, 则 $xy^2z = a^3 b^s a^r b^N c^{N-3} \notin L$.

❻ 无论 y 为何种情况, xy^2z 都不可能在 L 中, 与泵引理矛盾.

❼ 所以假设不成立, L 不是正则的. $\square$

思考题

- $L = \{0^n 1^n \mid 0 \leq n \leq 100\}$ 是否是正则语言?
- 有限的语言，是否符合泵引理呢?
  - $\varnothing$
  - $\{\varepsilon\}$
  - $\{0, 00\}$

泵引理只是必要条件

- 泵引理只是正则语言的必要条件
- 只能用来证明某个语言不是正则的
- 与正则语言等价的定理 — Myhill-Nerode Theorem

例 7. 语言 L 不是正则的, 但每个串都可以应用泵引理

$$L = \{ca^n b^n \mid n \geq 1\} \cup \{c^k w \mid k \neq 1, w \in \{a,b\}^*\}$$

- 其中 $A = \{ca^n b^n \mid n \geq 1\}$ 部分不是正则的
- 而 $B = \{c^k w \mid k \neq 1, w \in \{a,b\}^*\}$ 部分是正则的
- 而 A 的任何串 $w = ca^i b^i$, 都可应用泵引理, 因为

$$w = (\varepsilon)(c)(a^i b^i)$$

重复字符 c 生成的新串都会落入 B 中

思考题

对任何正则语言 L, 在泵引理中, 与 L 相关联的正整数 N

- 与识别 L 的 DFA 的状态数 n 之间有何关系?
- 与识别 L 的 NFA 的状态数之间呢?

思考题

语言

$$L = \{0^n x 1^n \mid n \geq 1, x \in \{0,1\}^*\}$$

是否是正则语言?

正则语言的性质

- 证明语言的非正则性
- 正则语言的封闭性
- 正则语言的判定性质
- 自动机的最小化

正则语言的封闭性

定义

正则语言经某些运算后得到的新语言仍保持正则，称为在这些运算下封闭.

正则语言 L 和 M，在这些运算下封闭

- 并：$L \cup M$
- 连接：LM
- 闭包：L^*
- 补：$\overline{L}$
- 差：$L - M$
- 交：$L \cap M$
- 反转：$L^R = \{w^R \mid w \in L\}$
- 同态：$h(L) = \{h(w) \mid w \in L, \text{同态 } h : \Sigma \to \Gamma^*\}$
- 逆同态：
  $h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L \subseteq \Gamma^*, \text{同态 } h : \Sigma \to \Gamma^*\}$

定理 6 (并/连接/闭包的封闭性)

正则语言在并, 连接和闭包运算下保持封闭.

证明: 由正则表达式的定义得证. $\square$

定理 7 (补运算封闭性)

如果 L 是 Σ 上的正则语言, 那么 $\overline{L} = \Sigma^* - L$ 也是正则的.

证明: 设接受语言 L 的 DFA

$$A = (Q, \Sigma, \delta, q_0, F)$$

即 $\mathbf{L}(A) = L$. 构造 DFA

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

则有 $\overline{L} = \mathbf{L}(B)$, 因为 $\forall w \in \Sigma^*$

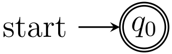
$$w \in \overline{L} \iff \hat{\delta}(q_0, w) \notin F \iff \hat{\delta}(q_0, w) \in Q - F \iff w \in \mathbf{L}(B).$$

□

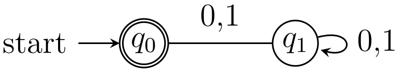
注意

使用这种方法求正则语言的补时, DFA 不能有缺失状态.

例 8. 若 $\Sigma = \{0,1\}$, $L = \{\varepsilon\}$ 的 DFA 如图, 请给出 $\overline{L}$ 的 DFA.



应使用完整的 DFA 去求补:



思考题

如何求正则表达式的补？

例 9. 证明 $L_{\text{neq}} = \{w \mid w$ 由数量不相等的 0 和 1 构成 $\}$ 不是正则的.

证明:

- 由泵引理不易直接证明 L_{neq} 不是正则的;
- 因为无论如何取 w, 将其分为 $w = xyz$ 时, 都不易产生 L_{neq} 之外的串;
- 而证明 L_{eq} 非正则很容易;
- 由补运算的封闭性, 所以 $L_{\text{neq}} = \overline{L_{\text{eq}}}$ 也不是正则的. □

定理 8

若 DFA A_L, A_M 和 A 的定义如下

$$
\begin{aligned}
A_L &= (Q_L, \Sigma, \delta_L, q_L, F_L) \\
A_M &= (Q_M, \Sigma, \delta_M, q_M, F_M) \\
A &= (Q_L \times Q_M, \Sigma, \delta, (q_L, q_M), F_L \times F_M)
\end{aligned}
$$

其中

$$
\begin{aligned}
\delta : (Q_L \times Q_M) \times \Sigma \to Q_L \times Q_M \\
\delta((p, q), a) = (\delta_L(p, a), \delta_M(q, a)).
\end{aligned}
$$

则对任意 $w \in \Sigma^*$,

$$
\hat{\delta}((q_L, q_M), w) = (\hat{\delta}(q_L, w), \hat{\delta}(q_M, w)).
$$

证明: 对 w 的结构归纳.

归纳基础：当 $w = \varepsilon$ 时

$$
\begin{aligned}
\hat{\delta}((q_L, q_M), \varepsilon) &= (q_L, q_M) && \hat{\delta}\text{ 的定义} \\
&= (\hat{\delta}_L(q_L, \varepsilon), \hat{\delta}_M(q_M, \varepsilon)) && \text{同理}
\end{aligned}
$$

归纳递推：当 $w = xa$ 时

$$
\begin{aligned}
\hat{\delta}((q_L, q_M), xa) &= \delta(\hat{\delta}((q_L, q_M), x), a) && \hat{\delta}\text{ 的定义} \\
&= \delta((\hat{\delta}(q_L, x), \hat{\delta}(q_M, x)), a) && \text{归纳假设} \\
&= (\delta_L(\hat{\delta}_L(q_L, x), a), \delta_M(\hat{\delta}_M(q_M, x), a)) && \delta\text{ 的构造} \\
&= (\hat{\delta}_L(q_L, xa), \hat{\delta}_M(q_M, xa)) && \hat{\delta}\text{ 的定义}
\end{aligned}
$$

$\square$

定理 9 (交运算封闭性)

如果 L 和 M 是正则语言，那么 $L \cap M$ 也是正则语言.

证明 1: 由 $L \cap M = \overline{\overline{L} \cup \overline{M}}$ 得证. $\square$

证明 2: 由定理 8 构造识别 $L \cap M$ 的 DFA A, 则 $\forall w \in \Sigma^*$,

$$\begin{aligned} w \in L \cap M &\iff \hat{\delta}_L(q_L, w) \in F_L \wedge \hat{\delta}_M(q_M, w) \in F_M \\ &\iff (\hat{\delta}_L(q_L, w), \hat{\delta}_M(q_M, w)) \in F_L \times F_M \\ &\iff \hat{\delta}((q_L, q_M), w) \in F_L \times F_M \\ &\iff w \in \mathbf{L}(A). \end{aligned}$$

因此 $\mathbf{L}(A) = L \cap M$, 所以 $L \cap M$ 也是正则的. $\square$

例 10. 如果已知语言

$$L_{01} = \{0^n1^n \mid n \geq 0\}$$

不是正则的, 请用封闭性证明语言

$$L_{\text{eq}} = \{w \in \{0,1\}^* \mid w \text{ 由数量相等的 0 和 1 构成}\}$$

也不是正则的.

证明:

❶ 首先, 因为 $\mathbf{0}^*\mathbf{1}^*$ 是正则语言;

❷ 而 $L_{01} = \mathbf{L}(\mathbf{0}^*\mathbf{1}^*) \cap L_{\text{eq}}$;

❸ 如果 L_{eq} 是正则的, L_{01} 必然也是正则的;

❹ 因为已知 L_{01} 不是正则的, 所以 L_{eq} 一定不是正则的. □

思考题

为什么又能用 L_{eq} 的子集 L_{01} 是非正则的，来证明 L_{eq} 是非正则的呢？

例 11. 如果 L_1 和 L_2 都不是正则的, 那么 $L_1 \cap L_2$ 一定不是正则的吗?

不一定. 因为, 如果令

$$L_1 = \{0^n 1^n \mid n \geq 0\}$$

$$L_2 = \{a^n b^n \mid n \geq 0\}$$

显然两者都不是正则语言, 但

$$L_1 \cap L_2 = \{\varepsilon\}$$

是正则语言.

定理 10 (差运算封闭性)

如果 L 和 M 都是正则语言, 那么 $L-M$ 也是正则的.

证明: $L-M=L\cap\overline{M}$. □

例 12. 证明正则语言在以下运算下封闭

$$\min(L) = \{w \mid w \text{ is in } L, \text{but no proper prefix of } w \text{ is in } L\}$$

证明 1: 设 L 的 DFA 为 $A = (Q, \Sigma, \delta, q_0, F)$, 构造 $\min(L)$ 的 DFA $B = (Q, \Sigma, \delta', q_0, F)$ 其中 δ' 如下, 往证 $L(B) = \min(L)$:

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{if } q \notin F \\ \varnothing & \text{if } q \in F \end{cases}$$

❶ $\forall w \in L(B)$, 存在转移序列 $q_0 q_1 \cdots q_n \in F$ 使 B 接受 w, 其中 $q_i \notin F \, (0 \le i \le n-1)$. $\therefore w \in \min(L)$.

❷ $\forall w \in \min(L)$, 有 $w \in L$, A 接受 w 的状态序列为如果 $q_0 q_1 \cdots q_n \in F$, 则显然 $q_i \notin F \, (0 \le i \le n-1)$, 否则 w 会有 L 可接受的前缀. $\therefore w \in L(B)$ □

例 12. 证明正则语言在以下运算下封闭

$$\min(L) = \{w \mid w \text{ is in } L, \text{but no proper prefix of } w \text{ is in } L\}$$

证明 2:

由封闭性

$$\min(L) = L - L\Sigma^+,$$

得证. $\square$

定义

字符串 $w = a_1a_2\ldots a_n$ 的反转，记为 w^R，定义为

$$w^R = a_na_{n-1}\ldots a_1.$$

定义

语言 L 的反转，记为 L^R，定义为

$$L^R = \{w^R \in \Sigma^* \mid w \in L\}.$$

定理 11 (反转的封闭性)

如果 L 是正则语言, 那么 L^R 也是正则的.

两种证明方法:

- 对正则表达式 E 的结构归纳, 往证
$$\mathbf{L}(E^R) = (\mathbf{L}(E))^R.$$
- 构造识别 L 的 NFA $A = (Q, \Sigma, \delta_A, q_0, F)$, 将其转换为识别 L^R 的 NFA
$$B = (Q, \Sigma, \delta_B, q_s, \{q_0\})$$
  1. 将 A 的边调转方向;
  2. 将 A 的初始状态 q_0, 改为唯一的接受状态;
  3. 新增初始状态 q_s, 且令 $\delta_B(q_s, \varepsilon) = F$.

例 13. 语言 L 及其反转 L^R 分别为

$$L = \{w \in \{0,1\}^* \mid w \text{ ends in 01.}\}$$

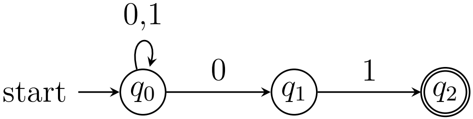
$$L^R = \{w \in \{0,1\}^* \mid w \text{ starts with 10.}\}$$

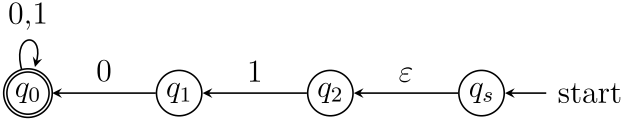
正则表达式分别为

$$L = (\mathbf{0}+\mathbf{1})^*\mathbf{01}$$

$$L^R = \mathbf{10}(\mathbf{0}+\mathbf{1})^*.$$

自动机分别为





证明: 往证如果有正则表达式 E, 则存在正则表达式 E^R 使

$$\mathbf{L}(E^R) = (\mathbf{L}(E))^R.$$

归纳基础:

❶ 当 $E = \varnothing$ 时, 有 $\varnothing^R = \varnothing$;

❷ 当 $E = \boldsymbol{\varepsilon}$ 时, 有 $\boldsymbol{\varepsilon}^R = \boldsymbol{\varepsilon}$;

❸ $\forall a \in \Sigma$, 当 $E = \mathbf{a}$ 时, 有 $\mathbf{a}^R = \mathbf{a}$;

都满足 $\mathbf{L}(E^R) = (\mathbf{L}(E))^R$, 因此命题成立.

归纳递推:

❶ 当 $E = E_1 + E_2$ 时, 有 $(E_1 + E_2)^R = E_1^R + E_2^R$

$$
\begin{aligned}
& (\mathbf{L}(E_1 + E_2))^R && \\
&= (\mathbf{L}(E_1) \cup \mathbf{L}(E_2))^R && \text{正则表达式的加} \\
&= \{w^R \mid w \in \mathbf{L}(E_1) \cup w \in \mathbf{L}(E_2)\} && \text{语言的反转} \\
&= (\mathbf{L}(E_1))^R \cup (\mathbf{L}(E_2))^R && \text{同上} \\
&= \mathbf{L}(E_1^R) \cup \mathbf{L}(E_2^R) && \text{归纳假设} \\
&= \mathbf{L}(E_1^R + E_2^R) && \text{正则表达式的加}
\end{aligned}
$$

归纳递推:

❶ 当 $E = E_1 + E_2$ 时, 有 $(E_1 + E_2)^R = E_1^R + E_2^R$

❷ 当 $E = E_1E_2$ 时, 有 $(E_1E_2)^R = E_2^R E_1^R$

$$
\begin{aligned}
(\mathbf{L}(E_1E_2))^R &= (\mathbf{L}(E_1)\mathbf{L}(E_2))^R && \text{正则表达式的连接} \\
&= \{w_1w_2 \mid w_1 \in \mathbf{L}(E_1), w_2 \in \mathbf{L}(E_2)\}^R && \text{语言的连接} \\
&= \{(w_1w_2)^R \mid w_1 \in \mathbf{L}(E_1), w_2 \in \mathbf{L}(E_2)\} && \text{语言的反转} \\
&= \{w_2^R w_1^R \mid w_1 \in \mathbf{L}(E_1), w_2 \in \mathbf{L}(E_2)\} && \text{字符串的反转} \\
&= \{w_2^R \mid w_2 \in \mathbf{L}(E_2)\}\{w_1^R \mid w_1 \in \mathbf{L}(E_1)\} && \text{语言的连接} \\
&= (\mathbf{L}(E_2))^R(\mathbf{L}(E_1))^R && \text{语言的反转} \\
&= \mathbf{L}(E_2^R)\mathbf{L}(E_1^R) = \mathbf{L}(E_2^R E_1^R) && \text{正则表达式的连接}
\end{aligned}
$$

归纳递推:

❶ 当 $E = E_1 + E_2$ 时, 有 $(E_1 + E_2)^R = E_1^R + E_2^R$

❷ 当 $E = E_1E_2$ 时, 有 $(E_1E_2)^R = E_2^R E_1^R$

❸ 当 $E = E_1^*$ 时, 有 $(E_1^*)^R = (E_1^R)^*$

$$
\begin{aligned}
&(\mathbf{L}(E_1^*))^R \\
&= \{w_1w_2\dots w_n \mid n \geq 0, w_i \in \mathbf{L}(E_1)\}^R && \text{正则表达式的闭包} \\
&= \{(w_1w_2\dots w_n)^R \mid n \geq 0, w_i \in \mathbf{L}(E_1)\} && \text{语言的反转} \\
&= \{w_n^R w_{n-1}^R \dots w_1^R \mid n \geq 0, w_i \in \mathbf{L}(E_1)\} && \text{字符串的反转} \\
&= \{w_n^R w_{n-1}^R \dots w_1^R \mid n \geq 0, w_i^R \in \mathbf{L}(E_1^R)\} && \text{归纳假设} \\
&= \{w_1w_2\dots w_n \mid n \geq 0, w_i \in \mathbf{L}(E_1^R)\} && \text{变量重命名} \\
&= \mathbf{L}((E_1^R)^*) && \text{正则表达式的闭包}
\end{aligned}
$$

都满足 $(\mathbf{L}(E))^R = \mathbf{L}(E^R)$, 因此命题成立, 所以 L^R 也是正则语言. □

同态

定义

若 Σ 和 Γ 是两个字母表，**同态**定义为函数 $h : \Sigma \to \Gamma^*$

$$\forall a \in \Sigma,\ \ h(a) \in \Gamma^*.$$

扩展 h 的定义到字符串，

$$\begin{aligned} &(1) \quad h(\varepsilon) = \varepsilon \\ &(2) \quad h(xa) = h(x)h(a) \end{aligned}$$

再扩展 h 到语言，对 $\forall L \subseteq \Sigma^*$，

$$h(L) = \{h(w) \mid w \in L\}.$$

例 14. 若由 $\Sigma=\{0,1\}$ 到 $\Gamma=\{a,b\}$ 的同态函数 h 为

$$h(0)=ab,\quad h(1)=\varepsilon.$$

则 Σ 上的字符串 0011, 在 h 的作用下

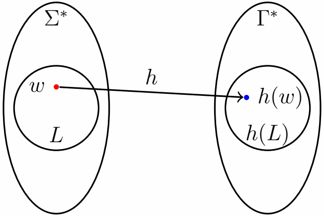
$$\begin{aligned}h(0011)&=h(\varepsilon)h(0)h(0)h(1)h(1)\\&=\varepsilon\cdot ab\cdot ab\cdot\varepsilon\cdot\varepsilon\\&=abab.\end{aligned}$$

语言 $L=\mathbf{1}^*\mathbf{0}+\mathbf{0}^*\mathbf{1}$, 在 h 的作用下, $h(L)$ 为:

$$\begin{aligned}h(\mathbf{1}^*\mathbf{0}+\mathbf{0}^*\mathbf{1})&=(h(\mathbf{1}))^*h(\mathbf{0})+(h(\mathbf{0}))^*h(\mathbf{1})\\&=(\boldsymbol{\varepsilon})^*(\mathbf{ab})+(\mathbf{ab})^*(\boldsymbol{\varepsilon})\\&=(\mathbf{ab})^*\end{aligned}$$

定理 12 (同态的封闭性)

若 L 是字母表 Σ 上的正则语言, h 是 Σ 上的同态, 则 $h(L)$ 也是正则的.



- 若 L 的正则表达式为 E, 即 $L = \mathbf{L}(E)$, 按如下规则构造表达式 $h(E)$

$$
\begin{aligned}
h(\varnothing) &= \varnothing & h(\mathbf{r}+\mathbf{s}) &= h(\mathbf{r}) + h(\mathbf{s}) \\
h(\boldsymbol{\varepsilon}) &= \boldsymbol{\varepsilon} & h(\mathbf{rs}) &= h(\mathbf{r})h(\mathbf{s}) \\
\forall a \in \Sigma,\ h(\mathbf{a}) &= h(a) & h(\mathbf{r}^*) &= (h(\mathbf{r}))^*
\end{aligned}
$$

- 往证 $\mathbf{L}(h(E)) = h(\mathbf{L}(E))$, 而 $h(E)$ 显然也是正则表达式, 因此 $h(L)$ 正则

证明: 对 E 的结构归纳, 往证 $\mathbf{L}(h(E)) = h(\mathbf{L}(E))$.

归纳基础:

- 当 $E = \boldsymbol{\varepsilon}$ 时

$$h(\mathbf{L}(\boldsymbol{\varepsilon})) = h(\{\varepsilon\}) = \{\varepsilon\} = \mathbf{L}(\boldsymbol{\varepsilon}) = \mathbf{L}(h(\boldsymbol{\varepsilon}))$$

- 当 $E = \varnothing$ 时

$$h(\mathbf{L}(\varnothing)) = h(\varnothing) = \varnothing = \mathbf{L}(\varnothing) = \mathbf{L}(h(\varnothing))$$

- $\forall a \in \Sigma$, 当 $E = \mathbf{a}$ 时

$$h(\mathbf{L}(\mathbf{a})) = h(\{a\}) = \{h(a)\} = \mathbf{L}(h(a)) = \mathbf{L}(h(\mathbf{a}))$$

所以命题成立.

归纳递推: 假设对正则表达式 F, G 分别有

$$\mathbf{L}(h(F)) = h(\mathbf{L}(F)), \quad \mathbf{L}(h(G)) = h(\mathbf{L}(G))$$

- 当 $E = F + G$ 时:

$$\begin{aligned}
h(\mathbf{L}(F+G)) &= h(\mathbf{L}(F) \cup \mathbf{L}(G)) && \text{正则表达式的加} \\
&= h(\mathbf{L}(F)) \cup h(\mathbf{L}(G)) && h\ \text{作用在每个集合的串上} \\
&= \mathbf{L}(h(F)) \cup \mathbf{L}(h(G)) && \text{归纳假设} \\
&= \mathbf{L}(h(F) + h(G)) && \text{正则表达式的加} \\
&= \mathbf{L}(h(F+G)) && h(F+G)\ \text{的定义}
\end{aligned}$$

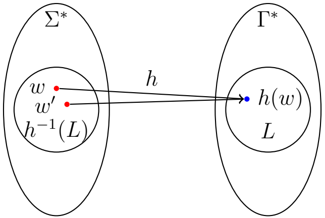
- 当 $E = FG$ 时: 略
- 当 $E = F^*$ 时: 略 $\square$

逆同态

定义

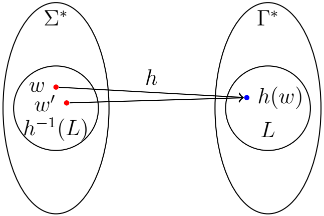
若 h 是字母表 Σ 到 Γ 的同态，且 L 是 Γ 上的语言，那么使 $h(w) \in L$ 的 w $(w \in \Sigma^*)$ 的集合，称为**语言 L 的 h 逆**，记为 $h^{-1}(L)$，即

$$h^{-1}(L) = \{w \in \Sigma^* \mid h(w) \in L\}.$$



定理 13 (逆同态的封闭性)

如果 h 是字母表 Σ 到 Γ 的同态，L 是 Γ 上的正则语言，那么 $h^{-1}(L)$ 也是正则语言.



证明: 由 L 的 DFA $A=(Q,\Gamma,\delta,q_0,F)$, 构造识别 $h^{-1}(L)$ 的 DFA

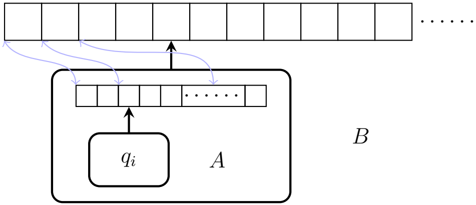
$$B=(Q,\Sigma,\delta',q_0,F),$$

证明: 由 L 的 DFA $A = (Q, \Gamma, \delta, q_0, F)$, 构造识别 $h^{-1}(L)$ 的 DFA

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

其中

$$\delta'(q, a) = \hat{\delta}(q, h(a)).$$



为证明 $\mathbf{L}(B) = h^{-1}(L)$, 先证明 $\widehat{\delta'}(q, w) = \hat{\delta}(q, h(w))$.

对 $|w|$ 归纳, 往证 $\widehat{\delta'}(q,w)=\hat{\delta}(q,h(w))$.

❶ 归纳基础: 若 $w=\varepsilon$

$$\hat{\delta}(q,h(\varepsilon))=\hat{\delta}(q,\varepsilon)=q=\widehat{\delta'}(q,\varepsilon),$$

❷ 归纳递推: 若 $w=xa$

$$\begin{aligned}
\widehat{\delta'}(q,xa) &= \delta'(\widehat{\delta'}(q,x),a) && \widehat{\delta'}\ \text{定义}\\
&= \delta'(\hat{\delta}(q,h(x)),a) && \text{归纳假设}\\
&= \hat{\delta}(\hat{\delta}(q,h(x)),h(a)) && \delta'\ \text{构造}\\
&= \hat{\delta}(q,h(x)h(a)) && \text{DFA 节例 5}\\
&= \hat{\delta}(q,h(xa)).
\end{aligned}$$

所以 $\forall w\in\Sigma^*$, $\widehat{\delta'}(q_0,w)=\hat{\delta}(q_0,h(w))\in F$, 即 w 被 B 接受当且仅当 $h(w)$ 被 A 接受, B 是识别 $h^{-1}(L)$ 的 DFA, 因此 $h^{-1}(L)$ 是正则的. □

例 15. Prove that $L = \{0^n 1^{2n} \mid n \geq 0\}$ is a language not regular.

证明: 设同态 $h : \{0, 1\} \to \{0, 1\}^*$ 为

$$h(0) = 0,$$

$$h(1) = 11,$$

那么

$$h^{-1}(L) = \{0^n 1^n \mid n \geq 0\} = L_{01},$$

我们已知 L_{01} 非正则, 由封闭性, L 不是正则的. $\square$

例 16. 若语言 $L = (\mathbf{00} + \mathbf{1})^*$, 同态 $h : \{a, b\} \to \{0, 1\}^*$ 为

$$h(a) = 01, \quad h(b) = 10,$$

请证明 $h^{-1}(L) = (\mathbf{ba})^*$.

证明: 往证 $h(w) \in L \iff w = (ba)^n$.

$(\Leftarrow)$ 若 $w = (ba)^n$, 而 $h(ba) = 1001$, 因此 $h(w) = (1001)^n \in L$.

$(\Rightarrow)$ 若 $h(w) \in L$, 假设 $w \notin (\mathbf{ba})^*$, 则只能有四种情况:

❶ w 以 a 开头, 则 $h(w)$ 以 01 开头, 显然 $h(w) \notin (\mathbf{00} + \mathbf{1})^*$;

❷ w 以 b 结尾, 则 $h(w)$ 以 10 结尾, 显然 $h(w) \notin (\mathbf{00} + \mathbf{1})^*$;

❸ w 有连续的 a, 即 $w = xaay$, 则 $h(w) = z1010v$, 显然 $h(w) \notin (\mathbf{00} + \mathbf{1})^*$;

❹ w 有连续的 b, 即 $w = xbby$, 则 $h(w) = z0101v$, 显然 $h(w) \notin (\mathbf{00} + \mathbf{1})^*$;

因此 w 只能是 $(ba)^n, n \geq 0$ 的形式. $\square$

例 17. For a language L, define head(L) to be the set of all prefixes of strings in L. Prove that if L is regular, so is head(L).

证明. 设 L 是 Σ 上的正则语言且 $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, a, b\}$.
定义同态 $h : \Gamma \to \Sigma^*$ 和 $g : \Gamma \to \Sigma^*$ 分别为:

$$\begin{array}{llll} h(0) = 0 & h(a) = 0 & g(0) = 0 & g(a) = \varepsilon \\ h(1) = 1 & h(b) = 1 & g(1) = 1 & g(b) = \varepsilon \end{array}$$

则因为 $(\mathbf{0} + \mathbf{1})^*(\mathbf{a} + \mathbf{b})^*$ 是 Γ 上的正则语言, 所以

$$(\mathbf{0} + \mathbf{1})^*(\mathbf{a} + \mathbf{b})^* \cap h^{-1}(L)$$

是 Γ 上的正则语言, 所以

$$\text{head}(L) = g((\mathbf{0} + \mathbf{1})^*(\mathbf{a} + \mathbf{b})^* \cap h^{-1}(L))$$

是 Σ 上的正则语言, 因此 head(L) 是正则的. $\square$

正则语言的性质

- 证明语言的非正则性
- 正则语言的封闭性
- 正则语言的判定性质
  - 空性，有穷性和无穷性
  - 等价性
- 自动机的最小化

正则语言的判定性质

正则语言, 或任何语言, 典型的 3 个判定问题:

1. 以某种形式化模型描述的语言是否为空? 是否无穷?
2. 某个特定的串 w 是否属于所描述的语言?
3. 以两种方式描述的语言, 是否是相同的? — 语言的等价性

我们想知道, 要回答这类问题的具体算法, 是否存在.

空性，有穷性和无穷性

定理 14

具有 n 个状态的有穷自动机 M 接受的集合 S:

1. S 是非空的，当且仅当 M 接受某个长度小于 n 的串；
2. S 是无穷的，当且仅当 M 接受某个长度为 m 的串，$n \leq m < 2n$.

所以，对于正则语言:

- 存在算法，判断其是否为空，只需检查全部长度小于 n 的串；
- 存在算法，判断其是否无穷，只需检查全部长度由 n 到 $2n-1$ 的串.

证明: 设接受正则语言 S 的 DFA 为 A.

1. 必要性: 显然成立. 充分性:
   - i. 如果 S 非空, 设 w 是 A 接受的串中长度最小者之一;
   - ii. 必然 $|w| < n$, 否则由泵引理 $w = xyz$, 接受 xz 更短.
2. 必要性: 由泵引理, 显然成立. 充分性:
   - i. 如果 S 无穷, 假设没有长度 n 到 $2n-1$ 之间的串;
   - ii. 那么取 $w \in \mathbf{L}(A)$ 是长度 $\geq 2n$ 中最小者之一;
   - iii. 由泵引理 $w = xyz$, 且 A 会接受更短的串 xz;
   - iv. 于是, 或者 w 不是长度最小的, 或者长度 n 到 $2n-1$ 之间有被接受的串, 因此假设不成立. □

定理 15

存在算法, 判定两个有穷自动机是否等价(接受语言相同).

证明:

1. 设 M_1 和 M_2 是分别接受 L_1 和 L_2 的有穷自动机;
2. 则 $(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$ 是正则的, 所以可被某个有穷自动机 M_3 接受;
3. 而 M_3 接受某个串, 当且仅当 $L_1 \neq L_2$;
4. 由于存在算法判断 $\mathbf{L}(M_3)$ 是否为空, 因此得证. □

正则语言的性质

- 证明语言的非正则性
- 正则语言的封闭性
- 正则语言的判定性质
- 自动机的最小化
  - DFA 状态的等价性
  - 填表算法与 DFA 最小化

DFA 状态的等价性

定义

DFA $A = (Q, \Sigma, \delta, q_0, F)$ 中两个状态 p 和 q, 对 $\forall w \in \Sigma^*$:

$$\hat{\delta}(p, w) \in F \Leftrightarrow \hat{\delta}(q, w) \in F,$$

则称这两个状态是等价的, 否则称为可区分的.

- 等价性只要求 $\hat{\delta}(p, w)$ 和 $\hat{\delta}(q, w)$ 同时在或不在 F 中, 而不必是相同状态.

填表算法

递归寻找 DFA 中全部的可区分状态对:

❶ 如果 $p \in F$ 而 $q \notin F$, 则 $[p, q]$ 是可区分的;

❷ $\exists a \in \Sigma$, 如果

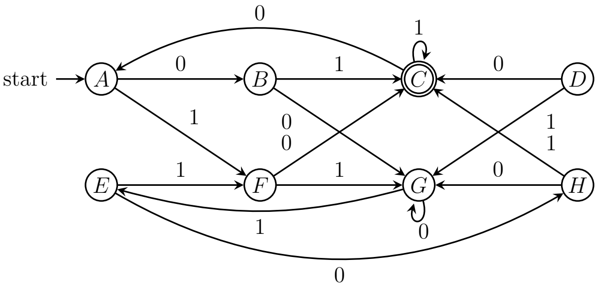
$$[r = \delta(p, a), s = \delta(q, a)]$$

是可区分的, 则 $[p, q]$ 是可区分的.

定理 16

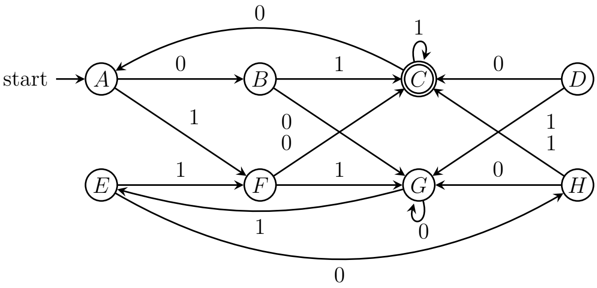
如果填表算法不能区分两个状态, 则这两个状态是等价的.

例 18. 用填表算法找到如图 DFA 中全部可区分状态对.



B							
C							
D							
E							
F							
G							
H							
	A	B	C	D	E	F	G

例 18. 用填表算法找到如图 DFA 中全部可区分状态对.

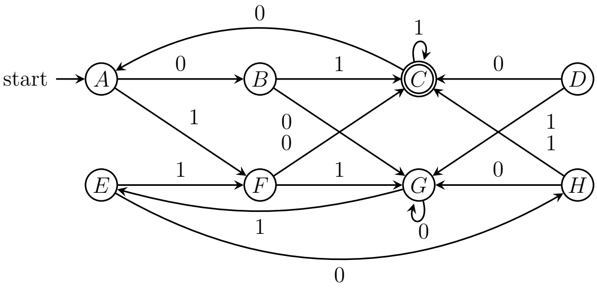


	A	B	C	D	E	F	G
B							
C	×	×					
D			×				
E			×				
F			×				
G			×				
H			×				

❶ 直接标记终态和非终态之间的状态对:

$$\{C\}\times\{A, B, D, E, F, G, H\}.$$

例 18. 用填表算法找到如图 DFA 中全部可区分状态对.

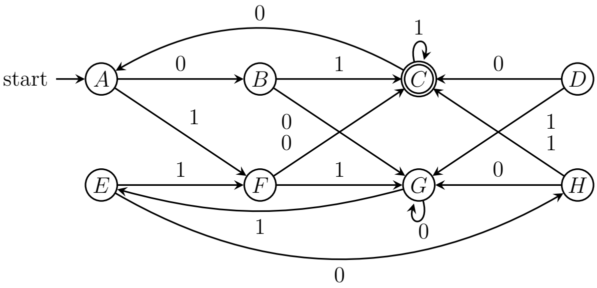


	A	B	C	D	E	F	G
B							
C	×	×					
D	×	×	×				
E			×	×			
F	×	×	×		×		
G			×	×		×	
H			×	×		×	

❷ 标记所有经过字符 0 到达终态和非终态的状态对:

$$\{D, F\} \times \{A, B, C, E, G, H\}.$$

例 18. 用填表算法找到如图 DFA 中全部可区分状态对.

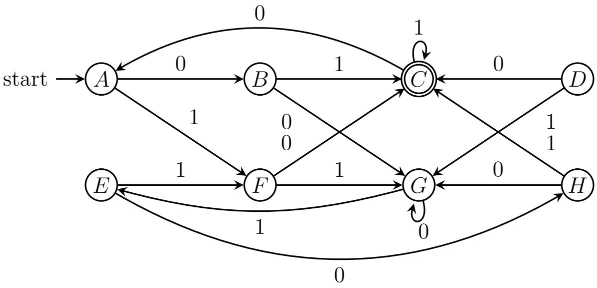


B	×						
C	×	×					
D	×	×	×				
E		×	×	×			
F	×	×	×		×		
G		×	×	×		×	
H	×		×	×	×	×	×
	A	B	C	D	E	F	G

❸ 标记所有经过字符 1 到达终态和非终态的状态对:

$$\{B, H\} \times \{A, C, D, E, F, G\}.$$

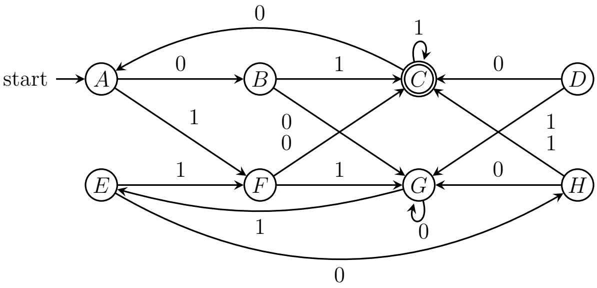
例 18. 用填表算法找到如图 DFA 中全部可区分状态对.



	A	B	C	D	E	F	G
B	×						
C	×	×					
D	×	×	×				
E		×	×	×			
F	×	×	×		×		
G		×	×	×		×	
H	×		×	×	×	×	×

❹ 此时还有 $[A, E]$, $[A, G]$, $[B, H]$, $[D, F]$, $[E, G]$ 未标记, 只需逐个检查.

例 18. 用填表算法找到如图 DFA 中全部可区分状态对.

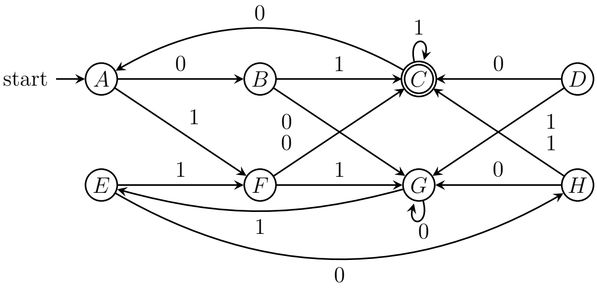


	A	B	C	D	E	F	G
B	×						
C	×	×					
D	×	×	×				
E		×	×	×			
F	×	×	×		×		
G	×	×	×	×	×	×	
H	×		×	×	×	×	×

❹ 此时还有 $[A, E]$, $[A, G]$, $[B, H]$, $[D, F]$, $[E, G]$ 未标记, 只需逐个检查.

- × $[A, G]$ 是可区分的, 因为经串 01 到可区分的 $[C, E]$;
- × $[E, G]$ 是可区分的, 因为经串 10 到可区分的 $[C, H]$.

例 18. 用填表算法找到如图 DFA 中全部可区分状态对.



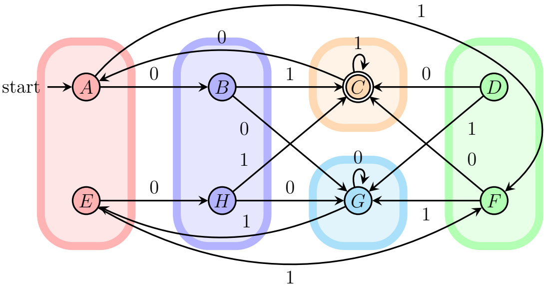
B	×						
C	×	×					
D	×	×	×				
E		×	×	×			
F	×	×	×		×		
G	×	×	×	×	×	×	
H	×		×	×	×	×	×
	A	B	C	D	E	F	G

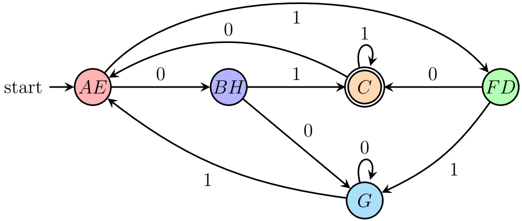
❺ 而 $[A, E]$, $[B, H]$ 和 $[D, F]$ 在经过很短的字符串后, 都会到达相同状态, 因此都是等价的.

DFA 最小化

根据等价状态, 将状态集划分成块, 构造等价的最小化 DFA.

续例 18. 构造其最小化的 DFA.





思考题

NFA 能否最小化?

形式语言与自动机理论

上下文无关文法

王春宇

计算机科学与技术学院
哈尔滨工业大学

上下文无关文法

- 上下文无关文法
  - 形式定义
  - 归约和派生
  - 最左派生和最右派生
  - 文法的语言
- 语法分析树
- 文法和语言的歧义性
- 文法的化简与范式

自然语言的文法

$$
\begin{aligned}
\langle \textit{sentence} \rangle &\to \langle \textit{noun-phrase} \rangle \langle \textit{verb-phrase} \rangle \\
\langle \textit{noun-phrase} \rangle &\to \langle \textit{article} \rangle \langle \textit{noun} \rangle \mid \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
\langle \textit{verb-phrase} \rangle &\to \langle \textit{verb} \rangle \mid \langle \textit{verb} \rangle \langle \textit{noun-phrase} \rangle \\
\langle \textit{article} \rangle &\to \texttt{a} \mid \texttt{the} \\
\langle \textit{noun} \rangle &\to \texttt{boy} \mid \texttt{girl} \mid \texttt{cat} \\
\langle \textit{adjective} \rangle &\to \texttt{big} \mid \texttt{small} \mid \texttt{blue} \\
\langle \textit{verb} \rangle &\to \texttt{sees} \mid \texttt{likes} \\
&\ldots
\end{aligned}
$$

自然语言的文法

使用文法规则产生句子:

$$
\begin{aligned}
\langle \textit{sentence} \rangle &\Rightarrow \langle \textit{noun-phrase} \rangle \langle \textit{verb-phrase} \rangle \\
&\Rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle \langle \textit{verb-phrase} \rangle \\
&\Rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle \langle \textit{verb} \rangle \langle \textit{noun-phrase} \rangle \\
&\Rightarrow \langle \textit{article} \rangle \langle \textit{noun} \rangle \langle \textit{verb} \rangle \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
&\Rightarrow \texttt{the } \langle \textit{noun} \rangle \langle \textit{verb} \rangle \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
&\Rightarrow \texttt{the girl } \langle \textit{verb} \rangle \langle \textit{article} \rangle \langle \textit{adjective} \rangle \langle \textit{noun} \rangle \\
&\Rightarrow \cdots \\
&\Rightarrow \texttt{the girl sees a blue cat}
\end{aligned}
$$

定义

如果字符串 $w \in \Sigma^*$ 满足

$$w = w^R,$$

则称字符串 w 为回文 *(palindrome)*.

定义

如果语言 L 中的字符串都是回文，则称 L 为回文语言

$$L = \{w \in \Sigma^* \mid w = w^R\}.$$

- ε, 010, 0000, radar, racecar, drawkward
- A man, a plan, a canal — Panama
- 僧游云隐寺，寺隐云游僧

例 1. 字母表 $\Sigma = \{0, 1\}$ 上的回文语言

$$L_{\text{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}.$$

- 很容易证明是 L_{pal} 是非正则的. 但如何表示呢?
- 可使用递归的方式来定义:
  1. 首先 ε, 0, 1 都是回文
  2. 如果 w 是回文, $0w0$ 和 $1w1$ 也是回文
- 使用嵌套定义表示这种递归结构:

$$\begin{array}{ll} A \to \varepsilon & \qquad A \to 0A0 \\ A \to 0 & \qquad A \to 1A1 \\ A \to 1 & \end{array}$$

上下文无关文法的形式定义

定义

上下文无关文法(*CFG*, *Context-Free Grammar*, 简称文法) G 是一个四元组

$$G = (V, T, P, S),$$

1. V: 变元的有穷集, 变元也称为非终结符或语法范畴;
2. T: 终结符的有穷集, 且 $V \cap T = \varnothing$;
3. P: 产生式的有穷集, 每个产生式包括:
   1. 一个变元, 称为产生式的头或左部;
   2. 一个产生式符号 $\rightarrow$, 读作定义为;
   3. 一个 $(V \cup T)^*$ 中的符号串, 称为体或右部;
4. $S \in V$: 初始符号, 文法开始的地方.

- 产生式 $A \to \alpha$, 读作 A 定义为 α
- 如果有多个 A 的产生式

$$A \to \alpha_1, \ A \to \alpha_2, \ \cdots, \ A \to \alpha_n$$

  可简写为

$$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$$

- 文法中变元 A 的全体产生式, 称为 A 产生式

续例 1. 回文语言 $L_{\text{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}$ 的文法可设计为

$$G = (\{A\}, \{0,1\}, \{A \to \varepsilon \mid 0 \mid 1 \mid 0A0 \mid 1A1\}, A).$$

字符使用的一般约定

- 终结符: $0, 1, \ldots,\ a, b, \ldots$
- 终结符串: $\ldots, w, x, y, z$
- 非终结符: $S, A, B, \ldots$
- 终结符或非终结符: $\ldots, X, Y, Z$
- 终结符或非终结符组成的串: $\alpha, \beta, \gamma, \ldots$

例 2. 简化版的算数表达式:

- 运算只有“加”和“乘” $(+, *)$, 参数仅为标识符;
- 标识符: 以 $\{a, b\}$ 开头由 $\{a, b, 0, 1\}$ 组成的字符串.

这样的表达式集合可用文法 G_{exp} 表示

$$G_{\mathrm{exp}} = (\{E, I\}, \{a, b, 0, 1, +, *, (,)\}, P, E),$$

其中产生式集 P 中有 10 条产生式

$$\begin{array}{lll} 1.\, E \to I & 5.\, I \to a & 9.\, I \to I0 \\ 2.\, E \to E + E & 6.\, I \to b & 10.\, I \to I1 \\ 3.\, E \to E * E & 7.\, I \to Ia & \\ 4.\, E \to (E) & 8.\, I \to Ib & \end{array}$$

注意, 变元 I 所定义的标识符集合, 刚好是 $(\mathbf{a}+\mathbf{b})(\mathbf{a}+\mathbf{b}+\mathbf{0}+\mathbf{1})^*$.

归约和派生

非形式定义

从字符串到文法变元的分析过程，称为**递归推理**或**归约**；
从文法变元到字符串的分析过程，称为**推导**或**派生**.

$$\begin{tikzcd} L \arrow[dd, bend right=60, "\text{派生}"'] \\ \text{文法 } G \\ w \arrow[uu, bend right=60, "\text{归约}"'] \end{tikzcd}$$

- 归约：自底向上，由产生式的体向头的分析
- 派生：自顶向下，由产生式的头向体分析

续例 2. 用算数表达式文法 G_{exp}, 将 $a * (a + b00)$ 归约的过程.

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

续例 2. 用算数表达式文法 G_{exp}, 将 $a*(a+b00)$ 归约的过程.

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

	串归约到变元		应用产生式	重用结果
(1)	a	I	5. $I \to a$	–
(2)	b	I	5. $I \to b$	–
(3)	$b0$	I	9. $I \to I0$	(2)
(4)	$b00$	I	9. $I \to I0$	(3)
(5)	a	E	1. $E \to I$	(1)
(6)	$b00$	E	1. $E \to I$	(4)
(7)	$a + b00$	E	2. $E \to E + E$	(5), (6)
(8)	$(a + b00)$	E	4. $E \to (E)$	(7)
(9)	$a * (a + b00)$	E	3. $E \to E * E$	(5), (8)

派生和归约的形式定义

定义

若 CFG $G = (V, T, P, S)$, 设 $\alpha, \beta, \gamma \in (V \cup T)^*$, $A \in V$, $A \to \gamma \in P$, 那么称在 G 中由 $\alpha A \beta$ 可派生出 $\alpha\gamma\beta$, 记为

$$\alpha A \beta \underset{G}{\Rightarrow} \alpha\gamma\beta.$$

相应的, 称 $\alpha\gamma\beta$ 可归约为 $\alpha A \beta$.

- $\alpha A \beta \underset{G}{\Rightarrow} \alpha\gamma\beta$, 即用 $A \to \gamma$ 的右部 γ 替换串 $\alpha A \beta$ 中变元 A 得到串 $\alpha\gamma\beta$
- 如果语境中 G 是已知的, 可省略, 记为 $\alpha A \beta \Rightarrow \alpha\gamma\beta$

- 设 $\alpha_1, \ldots, \alpha_m \in (V \cup T)^*$, $m \geq 1$, 对 $i = 1, \ldots, m-1$ 如果有

  $$\alpha_i \underset{G}{\Rightarrow} \alpha_{i+1}$$

  成立, 即 α_1 经过零步或多步派生可得到 α_m

  $$\alpha_1 \underset{G}{\Rightarrow} \alpha_2 \underset{G}{\Rightarrow} \cdots \underset{G}{\Rightarrow} \alpha_{m-1} \underset{G}{\Rightarrow} \alpha_m,$$

  那么, 记为

  $$\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \alpha_m.$$

- 若 α 派生出 β 刚好经过了 i 步, 可记为

  $$\alpha \overset{i}{\underset{G}{\Rightarrow}} \beta.$$

续例 2. 算数表达式 $a * (a + b00)$ 在文法 G_{exp} 中的派生过程.

续例 2. 算数表达式 $a*(a+b00)$ 在文法 G_{exp} 中的派生过程.

$$
\begin{aligned}
E &\Rightarrow E*E \Rightarrow E*(E) \Rightarrow I*(E) \\
&\Rightarrow I*(E+E) \Rightarrow I*(E+I) \Rightarrow I*(I+I) \\
&\Rightarrow I*(a+I) \Rightarrow a*(a+I) \Rightarrow a*(a+I0) \\
&\Rightarrow a*(a+I00) \Rightarrow a*(a+b00)
\end{aligned}
$$

最左派生和最右派生

定义

为限制派生的随意性，要求只替换符号串中最左边变元的派生过程，称为最左派生，记为

$$\underset{\text{lm}}{\Rightarrow}, \quad \overset{*}{\underset{\text{lm}}{\Rightarrow}},$$

只替换最右的，称为最右派生，记为

$$\underset{\text{rm}}{\Rightarrow}, \quad \overset{*}{\underset{\text{rm}}{\Rightarrow}} .$$

- 任何派生都有等价的最左派生和最右派生

$$A \overset{*}{\Rightarrow} w \quad \text{当且仅当} \quad A \overset{*}{\underset{\text{lm}}{\Rightarrow}} w \quad \text{当且仅当} \quad A \overset{*}{\underset{\text{rm}}{\Rightarrow}} w.$$

续例 2. 表达式 $a*(a+a)$ 在 $G_{\exp}$ 中的最左派生和最右派生分别为:

1. $E \to I$
2. $E \to E+E$
3. $E \to E*E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

$$\begin{aligned}
E &\underset{\text{lm}}{\Rightarrow} E*E \\
&\underset{\text{lm}}{\Rightarrow} I*E \\
&\underset{\text{lm}}{\Rightarrow} a*E \\
&\underset{\text{lm}}{\Rightarrow} a*(E) \\
&\underset{\text{lm}}{\Rightarrow} a*(E+E) \\
&\underset{\text{lm}}{\Rightarrow} a*(I+E) \\
&\underset{\text{lm}}{\Rightarrow} a*(a+E) \\
&\underset{\text{lm}}{\Rightarrow} a*(a+I) \\
&\underset{\text{lm}}{\Rightarrow} a*(a+a)
\end{aligned}$$

$$\begin{aligned}
E &\underset{\text{rm}}{\Rightarrow} E*E \\
&\underset{\text{rm}}{\Rightarrow} E*(E) \\
&\underset{\text{rm}}{\Rightarrow} E*(E+E) \\
&\underset{\text{rm}}{\Rightarrow} E*(E+I) \\
&\underset{\text{rm}}{\Rightarrow} E*(E+a) \\
&\underset{\text{rm}}{\Rightarrow} E*(I+a) \\
&\underset{\text{rm}}{\Rightarrow} E*(a+a) \\
&\underset{\text{rm}}{\Rightarrow} I*(a+a) \\
&\underset{\text{rm}}{\Rightarrow} a*(a+a)
\end{aligned}$$

定义

CFG $G = (V, T, P, S)$ 的语言定义为

$$\mathbf{L}(G) = \{w \mid w \in T^*,\ S \overset{*}{\underset{G}{\Rightarrow}} w\}.$$

那么符号串 w 在 $\mathbf{L}(G)$ 中, 要满足:

1. w 仅由终结符组成;
2. 初始符号 S 能派生出 w.

定义

语言 L 是某个 *CFG* G 定义的语言，即 $L = \mathbf{L}(G)$，则称 L 为上下文无关语言 (*CFL*, *Context-Free Language*).

- 上下文无关是指在文法派生的每一步

$$\alpha A\beta \Rightarrow \alpha\gamma\beta,$$

符号串 γ 仅根据 A 的产生式派生，而无需依赖 A 的上下文 α 和 β.

定义

如果有两个文法 CFG G_1 和 CFG G_2, 满足

$$\mathbf{L}(G_1) = \mathbf{L}(G_2),$$

则称 G_1 和 G_2 是等价的.

句型

定义

若 CFG $G = (V, T, P, S)$, 初始符号 S 派生出来的符号串, 称为 G 的句型, 即

$$\alpha \in (V \cup T)^* \text{ 且 } S \overset{*}{\Rightarrow} \alpha.$$

如果 $S \underset{\text{lm}}{\overset{*}{\Rightarrow}} \alpha$, 称 α 为左句型. 如果 $S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha$, 称 α 为右句型.

- 只含有终结符的句型, 也称为 G 的句子
- 而 $\mathbf{L}(G)$ 就是文法 G 全部的句子

例 3. 给出语言 $L = \{w \in \{0,1\}^* \mid w \text{ contains at least three 1s}\}$ 的文法.

解: $S \to A1A1A1A,\ A \to 0A \mid 1A \mid \varepsilon$

例 4. 描述 CFG $G = (\{S\}, \{a, b\}, \{S \to aSb, S \to ab\}, S)$ 定义的语言?

解: $\mathbf{L}(G) = \{a^n b^n \mid n \geq 1\}$, 因为 $S \Rightarrow aSb \Rightarrow \cdots \Rightarrow a^{n-1}Sb^{n-1} \Rightarrow a^n b^n$.

例 5. 请为语言 $L = \{0^n 1^m \mid n \neq m\}$ 设计文法.

解:

$$
\begin{array}{ll}
S \to AC \mid CB & A \to A0 \mid 0 \\
C \to 0C1 \mid \varepsilon & B \to 1B \mid 1
\end{array}
$$

例 6. 设计 $L_{\text{eq}} = \{w \in \{0,1\}^* \mid w \text{ 中 } 0 \text{ 和 } 1 \text{ 个数相等}\}$ 的文法.

解 1: $S \to 0S1 \mid 1S0 \mid SS \mid \varepsilon$, 寻找递归结构, 用变量构造递归结构;
解 2: $S \to S0S1S \mid S1S0S \mid \varepsilon$, “目标串”这样构成, 由变量定义变量.

例 7. 设计 $L_{j\geq 2i} = \{a^i b^j \mid j \geq 2i\}$ 的文法.

解:

$$
\begin{aligned}
S &\to AB \\
A &\to aAbb \mid \varepsilon \\
B &\to bB \mid \varepsilon
\end{aligned}
$$

或

$$
\begin{aligned}
S &\to aSbb \mid B \\
B &\to \varepsilon \mid bB
\end{aligned}
$$

程序设计语言的文法定义

- C — ISO C 1999 definition

  . . .

  selection-statement:

  `if` (*expression*) *statement*

  `if` (*expression*) *statement* `else` *statement*

  `switch` (*expression*) *statement*

  . . .

- Python — Full Grammar specification

  . . .

```
try_stmt: ('try' ':' suite
           ((except_clause ':' suite)+
            ['else' ':' suite]
            ['finally' ':' suite] |
           'finally' ':' suite))
```

  . . .

例 8. [Exe. 5.1.3] Show that every regular laugnage is a context-free laugnage.

例 8. [Exe. 5.1.3] Show that every regular laugnage is a context-free laugnage.

证明：对正则表达式 R 中运算符的个数 n 进行归纳.
归纳基础: 当 $n=0$ 时, R 只能是 $\boldsymbol{\varepsilon}$, $\varnothing$ 或 $\mathbf{a}$ $(a \in \Sigma)$, 可以构造仅有一条产生式的文法 $S \to \varepsilon$, $S \to \varnothing$ 或 $S \to a$ 得到.
归纳递推: 假设当 $n \le m$ 时成立. 当 $n = m+1$ 时, R 的形式只能由表达式 R_1 和 R_2 由连接、并或闭包形成：

- 若 $R = R_1 + R_2$, 则 R_1 和 R_2 中运算符都不超过 m, 所以都存在文法 G_1 和 G_2, 分别开始于 S_1 和 S_2, 只需构造新产生式和开始符号 $S \to S_1|S_2$, 连同 G_1 和 G_2 的产生式, 构成 R 的文法；
- 若 $R = R_1R_2$, 则同理构造 $S \to S_1S_2$ 即可；
- 若 $R = R_1^*$, 则构造 $S \to SS_1 \mid \varepsilon$ 即可.

且每种构造, 文法的语言与该表达式的语言等价. □

例 9. [Exe. 5.1.5] Let $T = \{0, 1, (,), +, *, \varnothing, e\}$. We may think of T as the set of symbols used by regular expressions over the alphabet $\{0, 1\}$; the only difference is that we use e for symbol ε, to avoid potential confusion in what follows. Your task is to design a CFG with set of terminals T that generates exactly the regular expressions with alphabet $\{0, 1\}$.

例 9. [Exe. 5.1.5] Let $T = \{0, 1, (,), +, *, \varnothing, e\}$. We may think of T as the set of symbols used by regular expressions over the alphabet $\{0, 1\}$; the only difference is that we use e for symbol ε, to avoid potential confusion in what follows. Your task is to design a CFG with set of terminals T that generates exactly the regular expressions with alphabet $\{0, 1\}$.

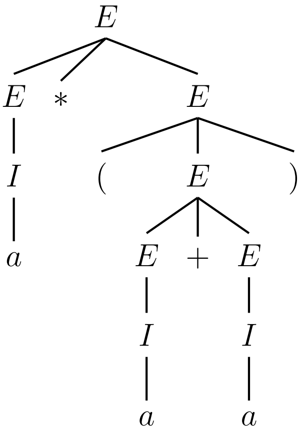
解: $S \to S + S \mid SS \mid S^* \mid (S) \mid 0 \mid 1 \mid \varnothing \mid e.$

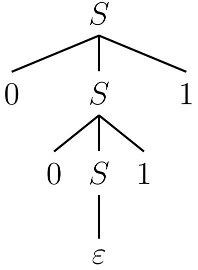
上下文无关文法

- 上下文无关文法
- 语法分析树
  - 形式定义
  - 语法树和派生的等价性
- 文法和语言的歧义性
- 文法的化简与范式

派生或归约的过程可以表示成树形结构.

- 例 2 文法 G_{exp} 中推导算数表达式 $a*(a+a)$ 的过程
- 例 6 中语言 L_{eq} 的文法中推导 0011 的过程





语法分析树的形式定义

定义

CFG $G=(V,T,P,S)$ 的**语法分析树**(语法树或派生树)为:

1. 每个内节点标记为 V 中的变元符号;
2. 每个叶节点标记为 $V\cup T\cup\{\varepsilon\}$ 中的符号;
3. 如果某内节点标记是 A, 其子节点从左至右分别为

$$X_1,\ X_2,\ \cdots,\ X_n$$

那么

$$A\to X_1X_2\cdots X_n\in P,$$

若有 $X_i=\varepsilon$, 则 ε 是 A 唯一子节点, 且 $A\to\varepsilon\in P$.

定义

语法树的全部叶节点从左到右连接起来，称为该树的**产物**或结果.

- 如果树根节点是初始符号 S，叶节点是终结符或 ε，那么该树的产物属于 $\mathbf{L}(G)$.

定义

语法树中标记为 A 的内节点及其全部子孙节点构成的子树，称为 **A 子树**.

语法分析树和派生的等价性

定理 17

CFG $G = (V, T, P, S)$ 且 $A \in V$, 那么文法 G 中

$$A \stackrel{*}{\Rightarrow} \alpha$$

当且仅当 G 中存在以 A 为根节点产物为 α 的语法树.

证明: [充分性] 对 $A \overset{j}{\Rightarrow} \alpha$ 的步骤数 j 归纳证明.

证明: [充分性] 对 $A \overset{j}{\Rightarrow} \alpha$ 的步骤数 j 归纳证明.

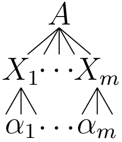
归纳基础: 当 $j=1$ 时, 即 $A \Rightarrow \alpha$, 那么有 $A \to \alpha \in P$, 可构造 $\begin{array}{c} A \\ \wedge \\ \alpha \end{array}$.

归纳递推: 假设 $j \le n$ 时命题成立. 当 $j = n+1$ 时, $A \xRightarrow{n+1} \alpha$ 的派生过程为

$$A \Rightarrow X_1 \cdots X_m \overset{n}{\Rightarrow} \alpha_1 \cdots \alpha_m = \alpha,$$

即第 1 步一定由某产生式 $A \to X_1 X_2 \cdots X_m \in P$ 派生.

而 X_i 若非终结符, 一定有 $X_i \overset{*}{\Rightarrow} \alpha_i$ 且不超过 n 步, 由归纳假设存在语法树 $\begin{array}{c} X_i \\ \wedge \\ \alpha_i \end{array}$. 因此可以构造以 A 为根, 以 X_i 为子树 (或叶子) 的语法树, 其产物刚好为 α.



[必要性] 对语法分析树的内节点数 j 归纳证明.

[必要性] 对语法分析树的内节点数 j 归纳证明.

归纳基础: 当 $j = 1$ 时, 即 $\begin{matrix} A \\ /|\backslash \\ \alpha \end{matrix}$, A 必为根, 则 $A \to \alpha \in P$, 所以 $A \stackrel{*}{\Rightarrow} \alpha$.

归纳递推: 假设 $j \leq n$ 时命题成立. 当 $j = n + 1$ 时, 根节点 A 的儿子依次为 $X_1, X_2, \ldots, X_m$, 则

$$A \to X_1 \cdots X_m \in P,\ \text{且}\ A \Rightarrow X_1 \cdots X_m.$$

而 X_i 子树 (或叶子) 内节点数都不超过 n, 由归纳假设有

$$X_i \stackrel{*}{\Rightarrow} \alpha_i,$$

从左至右连接 α_i, 刚好为树的产物 α, 所以有

$$X_1 X_2 \cdots X_m \stackrel{*}{\Rightarrow} \alpha_1 X_2 \cdots X_m \stackrel{*}{\Rightarrow} \cdots \stackrel{*}{\Rightarrow} \alpha_1 \alpha_2 \cdots \alpha_m = \alpha.$$

因此 $A \stackrel{*}{\Rightarrow} \alpha$ 命题成立. $\square$

语法树唯一确定最左 (右) 派生

- 每棵语法分析树都有唯一的最左 (右) 派生
- 给定 CFG $G = (V, T, P, S)$, $A \in V$, 以下命题等价:

  ❶ 通过递归推理, 确定串 w 在变元 A 的语言中.

  ❷ 存在以 A 为根节点, 产物为 w 的语法分析树.

  ❸ $A \overset{*}{\Rightarrow} w$.

  ❹ $A \overset{*}{\underset{\text{lm}}{\Rightarrow}} w$.

  ❺ $A \overset{*}{\underset{\text{rm}}{\Rightarrow}} w$.

例 10. [Exe. 5.2.2] Suppose that G is a CFG without any productions that have ε as the right side. If w is in $L(G)$, the length of w is n, and w has a derivation of m steps, show that w has a parse tree with $n + m$ nodes.

例 10. [Exe. 5.2.2] Suppose that G is a CFG without any productions that have ε as the right side. If w is in $L(G)$, the length of w is n, and w has a derivation of m steps, show that w has a parse tree with $n+m$ nodes.

证明:

❶ 派生 w 的每一步推导都对应语法树的一个内节点, 所以 w 语法树中共有 m 个内节点;

❷ 每个 w 的终结符都构成一个叶节点, 所以至少有 n 个叶节点, 而由于 G 中没有空产生式, 因此不会有标记为 ε 的叶节点, 所以只能有 n 个叶节点. 所以 w 的语法树有 $n+m$ 个节点. □

例 11. [Exe. 5.2.3] Suppose all is as in Exercise 5.2.2, but G may have some productions with ε as the right side. Show that a parse tree for a string w other than ε may have as many as $n + 2m - 1$ nodes, but no more.

例 11. [Exe. 5.2.3] Suppose all is as in Exercise 5.2.2, but G may have some productions with ε as the right side. Show that a parse tree for a string w other than ε may have as many as $n+2m-1$ nodes, but no more.

证明:

❶ 派生 w 的每一步推导都对应语法树的一个内节点, 所以 w 语法树中共有 m 个内节点.

❷ 每个 w 的终结符都构成一个叶节点, 所以至少有 n 个叶节点.

❸ 推导过程中, 每次空产生式的应用, 都会增加一个标记 ε 的叶节点, 但显然不能全部的 m 步都使用空产生式, 所以最多增加 $m-1$ 个 ε 叶节点. 因此 w 的语法树有最多 $m+n+m-1=n+2m-1$ 个节点. □

上下文无关文法

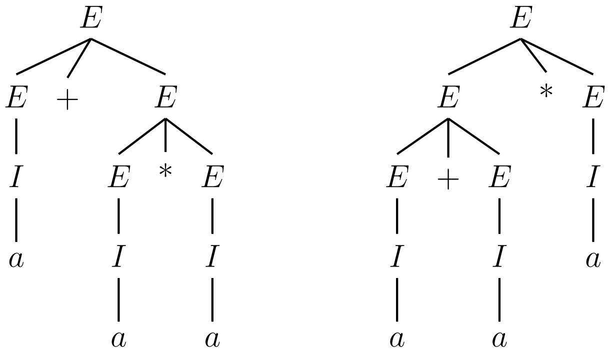
- 上下文无关文法
- 语法分析树
- 文法和语言的歧义性
  - 文法歧义性的消除
  - 语言的固有歧义性
- 文法的化简与范式

文法的歧义性

定义

如果 CFG G 使某些符号串有两棵不同的语法分析树，称文法 G 是**歧义的**.

续例 2. 算数表达式的文法 G_{exp} 中, 对句型 $a+a*a$ 有下面两棵语法分析树:



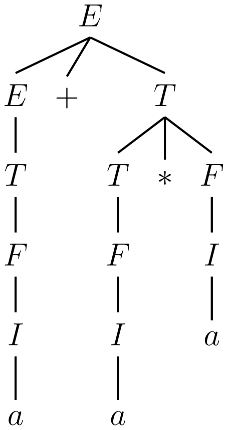
$$
\begin{aligned}
(1)\ E &\Rightarrow E+E \\
&\Rightarrow E+E*E \\
&\overset{*}{\Rightarrow} a+a*a
\end{aligned}
\qquad\qquad
\begin{aligned}
(2)\ E &\Rightarrow E*E \\
&\Rightarrow E+E*E \\
&\overset{*}{\Rightarrow} a+a*a
\end{aligned}
$$

文法歧义性的消除

有些文法的歧义性, 可以通过重新设计文法来消除.

续例 2. 文法 G_{exp} 重新设计为文法 $G_{\text{exp}'}$ 可消除歧义.

$$
\begin{aligned}
E &\to E + T \mid T \\
T &\to T * F \mid F \\
F &\to (E) \mid I \\
I &\to a \\
I &\to b \\
I &\to Ia \\
I &\to Ib \\
I &\to I0 \\
I &\to I1
\end{aligned}
$$



语言的固有歧义性

定义

定义同样的语言可以有多个文法，如果 CFL L 的所有文法都是歧义的，那么称语言 L 是**固有歧义的**.

- 固有歧义的语言确实存在，如语言

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

  中任何形为 $a^n b^n c^n$ 的串，总会有两棵语法树.

- “判定任何给定 CFG G 是否歧义” 是一个不可判定问题.

上下文无关文法

- 上下文无关文法
- 语法分析树
- 文法和语言的歧义性
- 文法的化简与范式
  - 消除无用符号
  - 消除 ε-产生式
  - 消除单元产生式
  - 乔姆斯基范式
  - 格雷巴赫范式

为什么要化简

- 典型问题: 给定 CFG G 和串 w, 判断 $w \in \mathbf{L}(G)$?
- 编译器设计和自然语言处理的基本问题之一
- 但文法的形式非常自由, 过于复杂不易于自动处理
- 以不改变语言为前提, 化简文法和限制文法的格式

例 12. 如下文法中, 有无意义的变元和产生式

$$
\begin{aligned}
S &\to 0DS1D \mid B \mid \varepsilon \\
B &\to BC1 \mid 0CBC \\
A &\to A0 \mid A1 \mid \varepsilon \\
C &\to D \\
D &\to \varepsilon
\end{aligned}
$$

文法的化简

1. 消除无用符号: 对文法定义语言没有贡献的符号
2. 消除 ε 产生式: $A \to \varepsilon$ (得到语言 $L - \{\varepsilon\}$)
3. 消除单元产生式: $A \to B$

无用符号

定义

CFG $G=(V,T,P,S)$, 符号 $X\in(V\cup T)$:

❶ 如果 $S\overset{*}{\Rightarrow}\alpha X\beta$, 称 X 是可达的;

❷ 如果 $\alpha X\beta\overset{*}{\Rightarrow}w\ (w\in T^*)$, 称 X 是产生的;

❸ 如果 X 同时是产生的和可达的, 即

$$S\overset{*}{\Rightarrow}\alpha X\beta\overset{*}{\Rightarrow}w\quad(w\in T^*),$$

则称 X 是有用的, 否则称 X 为无用符号.

消除无用符号

消除无用符号：删除全部含有“非产生的”和“非可达的”符号的产生式

计算“产生的”符号集

1. 每个 T 中的符号都是产生的；
2. $A \to \alpha \in P$ 且 α 中符号都是产生的，则 A 是产生的.

计算“可达的”符号集

1. 符号 S 是可达的；
2. $A \to \alpha \in P$ 且 A 是可达的，则 α 中符号都是可达的.

定理 18

每个非空的 CFL 都能被一个不带无用符号的 CFG 定义.

注意

- 先寻找并消除全部非“产生的”符号
- 再寻找并消除全部非“可达的”符号
- 否则可能消除不完整

例 13. 消除如下文法无用符号

$$S \to AB \mid a$$
$$A \to b$$

解：先消除非产生的

$$S \to a$$
$$A \to b$$

再消除非可达的

$$S \to a$$

消除 ε-产生式

定义

文法中形如 $A \to \varepsilon$ 的产生式称为ε-产生式.
如果变元 $A \overset{*}{\Rightarrow} \varepsilon$, 称 A 是可空的.

- ε-产生式在文法定义语言时, 除产生空串外没有其他帮助
- 对于 CFL L, 消除其文法中全部的 ε-产生式后, 得到语言 $L - \{\varepsilon\}$

确定“可空变元”

1. 如果 $A \to \varepsilon$, 则 A 是可空的;
2. 如果 $B \to \alpha$ 且 α 中的每个符号都是可空的, 则 B 是可空的.

替换产生式

将含有可空变元的一条产生式 $A \to X_1X_2\cdots X_n$,
用一组产生式 $A \to Y_1Y_2\cdots Y_n$ 代替, 其中:

1. 若 X_i 不是可空的, Y_i 为 X_i;
2. 若 X_i 是可空的, Y_i 为 X_i 或 ε;
3. 但 Y_i 不能全为 ε.

定理 19

任何 CFG G, 都存在一个不带无用符号和 ε-产生式的 CFG G', 使 $\mathbf{L}(G') = \mathbf{L}(G) - \{\varepsilon\}$.

例 14. 消除 CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$ 的 ε-产生式.

$$
\begin{aligned}
S &\to AB \\
A &\to AaA \mid \varepsilon \\
B &\to BbB \mid \varepsilon
\end{aligned}
$$

解: CFG G' 为

$$
\begin{aligned}
S &\to AB \mid A \mid B \\
A &\to AaA \mid Aa \mid aA \mid a \\
B &\to BbB \mid Bb \mid bB \mid b
\end{aligned}
$$

消除单元产生式

确定“单元对”

如果有 $A \overset{*}{\Rightarrow} B$, 则称 $[A, B]$ 为单元对.

1. $A \to B \in P$, 则 $[A, B]$ 是单元对;
2. 若 $[A, B]$ 和 $[B, C]$ 都是单元对, 则 $[A, C]$ 是单元对.

消除单元产生式

1. 删除全部形为 $A \to B$ 的单元产生式;
2. 对每个单元对 $[A, B]$, 将 B 的产生式复制给 A.

定理 20

每个不带 ε 的 CFL 都可由一个不带无用符号, ε-产生式和单元产生式的文法定义.

例 15. 消除文法的单元产生式

$$
\begin{aligned}
S &\to A \mid B \mid 0S1 \\
A &\to 0A \mid 0 \\
B &\to 1B \mid 1
\end{aligned}
$$

解: 单位对为 $[S, A]$ 和 $[S, B]$, 带入得:

$$
\begin{aligned}
S &\to 0S1 & \qquad A &\to 0A \mid 0 \\
S &\to 0A \mid 0 & \qquad B &\to 1B \mid 1 \\
S &\to 1B \mid 1
\end{aligned}
$$

文法化简的可靠顺序

1. 消除 ε-产生式;
2. 消除单元产生式;
3. 消除非产生的无用符号;
4. 消除非可达的无用符号.

例 16. [Exe. 7.1.2] Begin with the grammar:

$$S \to ASB \mid \varepsilon$$
$$A \to aAS \mid a$$
$$B \to SbS \mid A \mid bb$$

❶ Eliminate $\varepsilon-$productions.

❷ Eliminate any unit productions in the resulting grammar.

❸ Eliminate any useless symbols in the resulting grammar.

限制文法格式

将任意形式的文法转换为:

❶ 乔姆斯基范式 (CNF, Chomsky Normal Form)

❷ 格雷巴赫范式 (GNF, Greibach Normal Form)

乔姆斯基范式

定理 21 (乔姆斯基范式 CNF)

每个不带 ε 的 CFL 都可由这样的 CFG G 定义，G 中每个产生式都形为

$$A \to BC \quad \text{或} \quad A \to a$$

其中 A, B 和 C 都是变元，a 是终结符.

- 利用 CNF 派生长度为 n 的串，刚好需要 $2n-1$ 步
- 因此存在算法判断任意字符串 w 是否在给定的 CFL 中
- 利用 CNF 的 CYK 算法 — $O(n^3)$ 时间复杂度的解析算法

CFG 转为 CNF 的方法

❶ 将产生式

$$A \to X_1 X_2 \cdots X_m \quad (m \geq 2)$$

中每个终结符 a 替换为新变元 C_a, 并增加新产生式

$$C_a \to a$$

❷ 引入新变元 $D_1, D_2, \cdots, D_{m-2}$, 将产生式

$$A \to B_1 B_2 \cdots B_m \quad (m > 2)$$

替换为一组级联的产生式

$$\begin{aligned} A &\to B_1 D_1 \\ D_1 &\to B_2 D_2 \\ &\cdots \\ D_{m-2} &\to B_{m-1} B_m \end{aligned}$$

例 17. CFG $G = (\{S, A, B\}, \{a, b\}, P, S)$, 产生式集合 P 为:

$$
\begin{aligned}
S &\to bA \mid aB \\
A &\to bAA \mid aS \mid a \\
B &\to aBB \mid bS \mid b
\end{aligned}
$$

请设计等价的 CNF 文法.

解: CNF 为

$$
\begin{array}{lll}
S \to C_bA \mid C_aB & & \\
A \to C_aS \mid C_bD_1 \mid a & D_1 \to AA & C_a \to a \\
B \to C_bS \mid C_aD_2 \mid b & D_2 \to BB & C_b \to b
\end{array}
$$

证明:

证明: 设 CFL L 不含 ε, 由定理 20, 存在不含 ε-产生式和单元产生式的等价文法 $G_1=(V,T,P,S)$. 考虑 P 中一条产生式 $A\to X_1X_2\dots X_m$ $(m\geq 2)$.

❶ 若某个 X_i 是终结符 a, 则引入新变元 C_a 和新产生式 $C_a\to a$, 并用 C_a 替换 X_i, 得文法 $G_2=(V',T,P',S)$.

❷ 显然 $\mathbf{L}(G_1)\subseteq\mathbf{L}(G_2)$, 因为如果 $\alpha\underset{G_1}{\Rightarrow}\beta$, 那么 $\alpha\overset{*}{\underset{G_2}{\Rightarrow}}\beta$.

❸ 用归纳法证明 $A\overset{i}{\underset{G_2}{\Rightarrow}}w\Longrightarrow A\overset{*}{\underset{G_1}{\Rightarrow}}w$, 这里的 $A\in V$, $w\in T^*$.

  ⅰ 当 $i=1$ 时是显然的, 或者用了 P 中未修改的产生式, 或者用了被修改的产生式, 而二者都有 $A\overset{*}{\underset{G_1}{\Rightarrow}}w$.

  ⅱ 假设当 $i\leq n$ 时命题成立. 当 $i=n+1$ 时, $A\overset{i}{\underset{G_2}{\Rightarrow}}w$ 的第 1 步, 必然使用了某个产生式 $A\to B_1B_2\cdots B_m$, 即 $A\underset{G_2}{\Rightarrow}B_1B_2\cdots B_m\overset{n}{\underset{G_2}{\Rightarrow}}w=w_1w_2\cdots w_m$ 那么, 如果 $B_i\in V'-V$, B_i 一定是对应某个终结符 a_i 的 C_{a_i}, w_i 必然是 a_i, 令 $X_i=a_i$; 如果 $B_i\in V$, $B_i\overset{*}{\underset{G_2}{\Rightarrow}}w_i$ 一定不超过 n 步, 由归纳假设, $B_i\overset{*}{\underset{G_1}{\Rightarrow}}w_i$, 那么令 $X_i=B_i$. 由 P' 的结构, $A\to X_1X_2\cdots X_m$ 是 P 的一条产生式, 所以 $A\underset{G_1}{\Rightarrow}X_1X_2\cdots X_m\overset{*}{\underset{G_1}{\Rightarrow}}w_1w_2\cdots w_m=w$.

所以 $\mathbf{L}(G_2)\subseteq\mathbf{L}(G_1)$. □

格雷巴赫范式

定理 22 (格雷巴赫范式 GNF)

每个不带 ε 的 CFL 都可由这样的 CFG G 定义，G 中每个产生式都形为

$$A \to a\alpha$$

其中 A 是变元，a 是终结符，α 是零或多个变元的串.

- GNF 每个产生式都会引入一个终结符
- 长度为 n 的串的派生恰好是 n 步

例 18. 将以下文法转换为 GNF.

$$
\begin{aligned}
S &\to AB \\
A &\to aA \mid bB \mid b \\
B &\to b
\end{aligned}
$$

解: GNF 为

$$
\begin{aligned}
S &\to aAB \mid bBB \mid bB \\
A &\to aA \mid bB \mid b \\
B &\to b
\end{aligned}
$$

直接左递归

定义

文法中形式为 $A \to A\alpha$ 的产生式, 称为直接左递归.

消除直接左递归

❶ 若 A 产生式

$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m$$

其中 $\alpha_i \neq \varepsilon$, β_j 不以 A 开始;

❷ 引入新变元 B, 并用如下产生式替换

$$\begin{aligned} A &\to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m \mid \beta_1 B \mid \beta_2 B \mid \cdots \mid \beta_m B \\ B &\to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n \mid \alpha_1 B \mid \alpha_2 B \mid \cdots \mid \alpha_n B \end{aligned}$$

间接左递归

定义

文法中如果有形式为

$$A \to B\alpha \mid \dots$$
$$B \to A\beta \mid \dots$$

的产生式，称为间接左递归.

- 会有 $A \Rightarrow B\alpha \Rightarrow A\beta\alpha$，无法通过代换消除递归

消除间接左递归

❶ 将文法中变元重命名为 $A_1, A_2, \cdots, A_n$;

❷ 通过代入, 使产生式都形如

$$\begin{aligned} A_i &\to A_j\alpha \\ A_i &\to a\alpha \end{aligned}$$

但要求 $i \le j$;

❸ 消除直接左递归 $A_i \to A_i\beta$, 再代入其他产生式.

例 19. Convert the following grammar to GNF.

$$
\begin{aligned}
S &\to AB \\
A &\to BS \mid b \\
B &\to SA \mid a
\end{aligned}
$$

解:

1. 重命名变元, 代换 $i > j$ 的 A_j

$$
\begin{aligned}
A_1 &\to A_2A_3 \\
A_2 &\to A_3A_1 \mid b \\
A_3 &\to a \mid \cancel{A_1A_2} \mid \cancel{A_2A_3A_2} \mid \\
&\quad A_3A_1A_3A_2 \mid bA_3A_2
\end{aligned}
$$

2. 消除直接左递归

$$
\begin{aligned}
A_1 &\to A_2A_3 \\
A_2 &\to A_3A_1 \mid b \\
A_3 &\to bA_3A_2 \mid a \mid bA_3A_2B_1 \mid aB_1 \\
B_1 &\to A_1A_3A_2 \mid A_1A_3A_2B_1
\end{aligned}
$$

3. A_3 产生式代入到 A_2, A_2 产生式代入到 A_1, A_1 产生式代入 B_1

$$
\begin{aligned}
A_3 &\to bA_3A_2 \mid a \mid bA_3A_2B_1 \mid aB_1 \\
A_2 &\to bA_3A_2A_1 \mid aA_1 \mid bA_3A_2B_1A_1 \mid aB_1A_1 \mid b \\
A_1 &\to bA_3A_2A_1A_3 \mid aA_1A_3 \mid bA_3A_2B_1A_1A_3 \mid aB_1A_1A_3 \mid bA_3 \\
B_1 &\to bA_3A_2A_1A_3A_3A_2 \mid aA_1A_3A_3A_2 \mid bA_3A_2B_1A_1A_3A_3A_2 \mid \\
&\quad aB_1A_1A_3A_3A_2 \mid bA_3A_3A_2 \mid bA_3A_2A_1A_3A_3A_2B_1 \mid aA_1A_3A_3A_2B_1 \mid \\
&\quad bA_3A_2B_1A_1A_3A_3A_2B_1 \mid aB_1A_1A_3A_3A_2B_1 \mid bA_3A_3A_2B_1
\end{aligned}
$$

GNF 引理 1

如果有文法 $G = (V, T, P, S)$, 设 $A \to \alpha_1 B \alpha_2$ 是 P 中的一个产生式, 且 $B \to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_n$ 是 P 中的全部 B 产生式. 将产生式 $A \to \alpha_1 B \alpha_2$ 从 P 中删除, 并增加

$$A \to \alpha_1 \beta_1 \alpha_2 \mid \alpha_1 \beta_2 \alpha_2 \mid \cdots \mid \alpha_1 \beta_n \alpha_2$$

一组产生式, 得到文法 $G_1 = (V, T, P', S)$, 那么 $\mathbf{L}(G) = \mathbf{L}(G_1)$.

GNF 引理 1

如果有文法 $G=(V,T,P,S)$, 设 $A\to\alpha_1B\alpha_2$ 是 P 中的一个产生式, 且 $B\to\beta_1\mid\beta_2\mid\cdots\mid\beta_n$ 是 P 中的全部 B 产生式. 将产生式 $A\to\alpha_1B\alpha_2$ 从 P 中删除, 并增加

$$A\to\alpha_1\beta_1\alpha_2\mid\alpha_1\beta_2\alpha_2\mid\cdots\mid\alpha_1\beta_n\alpha_2$$

一组产生式, 得到文法 $G_1=(V,T,P',S)$, 那么 $\mathbf{L}(G)=\mathbf{L}(G_1)$.

证明:

❶ 显然 $\mathbf{L}(G_1)\subseteq\mathbf{L}(G)$, 因为 G_1 的派生中, 如果用到了 $A\to\alpha_1\beta_i\alpha_2$, 在 G 中可以使用 $A\underset{G}{\Rightarrow}\alpha_1B\alpha_2\underset{G}{\Rightarrow}\alpha_1\beta_i\alpha_2$.

❷ 而因为 $A\to\alpha_1B\alpha_2$ 是唯一在 G 中而不再 G_1 中的产生式, 每当 G 的派生中用到了 $\alpha_1B\alpha_2$ 时, 一定会在后面某一步中用到形如 $B\to\beta_i$ 的产生式来派生 B, 这两步在 G_1 中可以使用一步 $A\underset{G_1}{\Rightarrow}\alpha_1\beta_i\alpha_2$ 来代替, 所以 $\mathbf{L}(G)\subseteq\mathbf{L}(G_1)$. □

GNF 引理 2

如果有文法 $G=(V,T,P,S)$, 设带有直接左递归的 A 产生式为

$$A \to A\alpha_1 \mid A\alpha_2 \mid \cdots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m$$

其中 β_i 不以 A 开头. 在 V 中引入新的变元 B 并用以下产生式

$$\begin{aligned} A &\to \beta_1 \mid \beta_2 \mid \cdots \mid \beta_m \mid \beta_1 B \mid \beta_2 B \mid \cdots \mid \beta_m B \\ B &\to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n \mid \alpha_1 B \mid \alpha_2 B \mid \cdots \mid \alpha_n B \end{aligned}$$

替换全部 A 产生式, 得到文法 $G_1=(V\cup\{B\},T,P',S)$, 那么 $\mathbf{L}(G)=\mathbf{L}(G_1)$.

证明: 在文法 G 中一系列使用 $A \to A\alpha_i$ 的最左派生, 最后必以 $A \to \beta_j$ 结束, 而这样的最左派生

$$
\begin{aligned}
A \underset{\text{lm}}{\Rightarrow} A\alpha_{i_1} \underset{\text{lm}}{\Rightarrow} A\alpha_{i_2}\alpha_{i_1} \underset{\text{lm}}{\Rightarrow} \cdots \\
\underset{\text{lm}}{\Rightarrow} A\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_1} \\
\underset{\text{lm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_1},
\end{aligned}
$$

在 G_1 中可以使用一系列最右派生来代替

$$
\begin{aligned}
A \underset{\text{rm}}{\Rightarrow} \beta_j B \underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p} B \underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}} B \underset{\text{rm}}{\Rightarrow} \cdots \\
\underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_2} B \\
\underset{\text{rm}}{\Rightarrow} \beta_j\alpha_{i_p}\alpha_{i_{p-1}}\ldots\alpha_{i_2}\alpha_{i_1}.
\end{aligned}
$$

而且, 相反的转换也成立, 因此 $\mathbf{L}(G) = \mathbf{L}(G_1)$. $\square$

形式语言与自动机理论

下推自动机

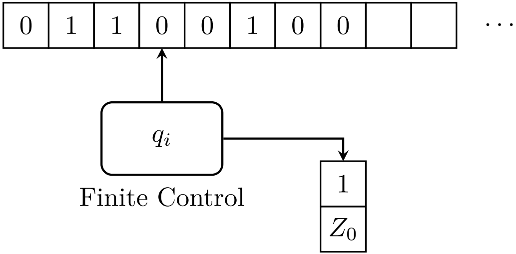
王春宇

计算机科学与技术学院
哈尔滨工业大学

下推自动机

- 下推自动机
  - 形式定义
  - 瞬时描述和转移符号
- 下推自动机接受的语言
- 下推自动机与文法的等价性
- 确定型下推自动机

下推自动机



下推自动机的形式定义

定义

下推自动机(*PDA*, *Pushdown Automata*) P 为七元组

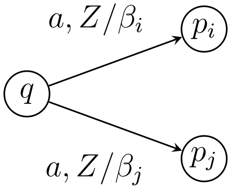
$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

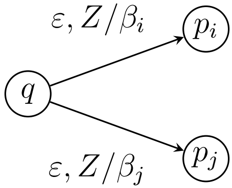
1. Q, 有穷状态集;
2. Σ, 有穷输入符号集;
3. Γ, 有穷栈符号集;
4. $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to 2^{Q \times \Gamma^*}$, 状态转移函数;
5. $q_0 \in Q$, 初始状态;
6. $Z_0 \in \Gamma - \Sigma$, 栈底符号;
7. $F \subseteq Q$, 接收状态集或终态集.

PDA 的动作和状态转移图

如果 $q, p_i \in Q \ (1 \le i \le m)$, $a \in \Sigma$, $Z \in \Gamma$, $\beta_i \in \Gamma^*$, 可以有动作:

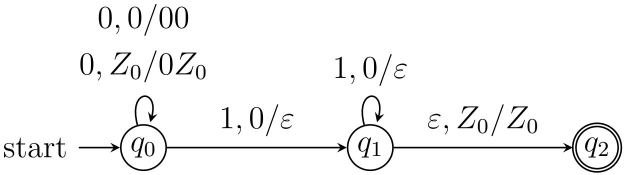
$$\delta(q, a, Z) = \{(p_1, \beta_1), (p_2, \beta_2), \cdots, (p_m, \beta_m)\}, \text{ 或}$$
$$\delta(q, \varepsilon, Z) = \{(p_1, \beta_1), (p_2, \beta_2), \cdots, (p_m, \beta_m)\}.$$





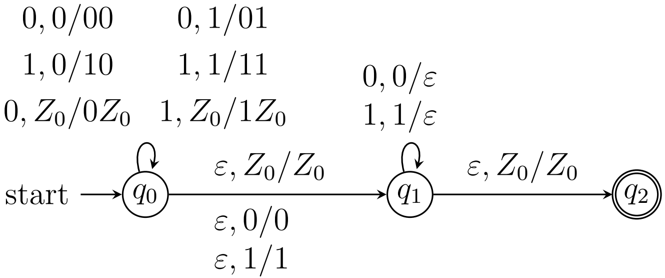
例 1. 设计识别 $L_{01} = \{0^n 1^n \mid n \geq 1\}$ 的 PDA.

例 1. 设计识别 $L_{01} = \{0^n 1^n \mid n \geq 1\}$ 的 PDA.

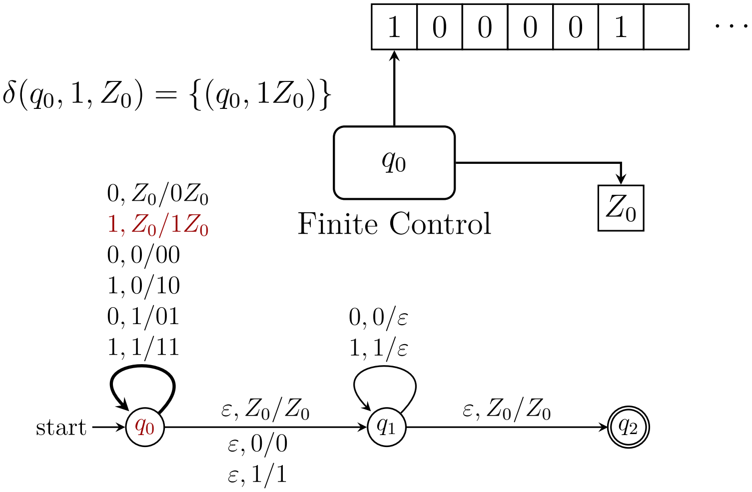


例 2. 设计识别 $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA.

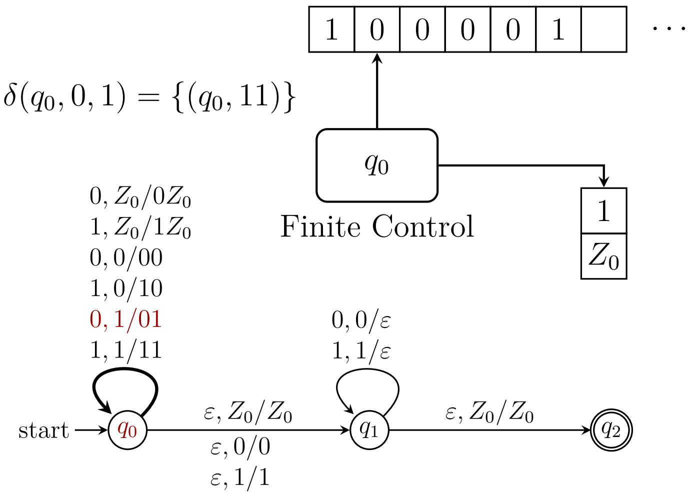
例 2. 设计识别 $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA.



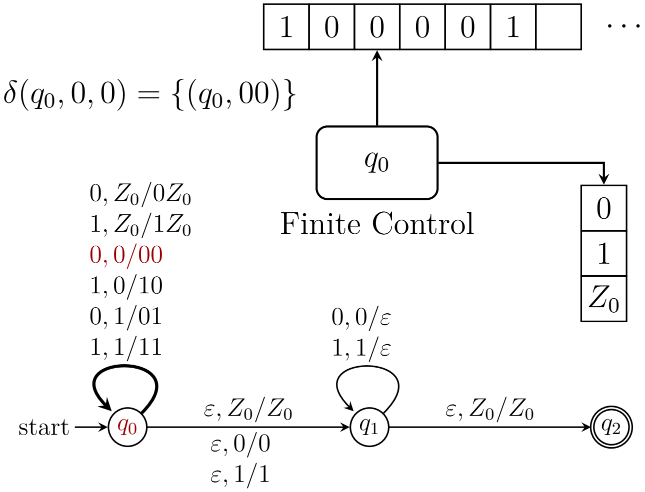
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



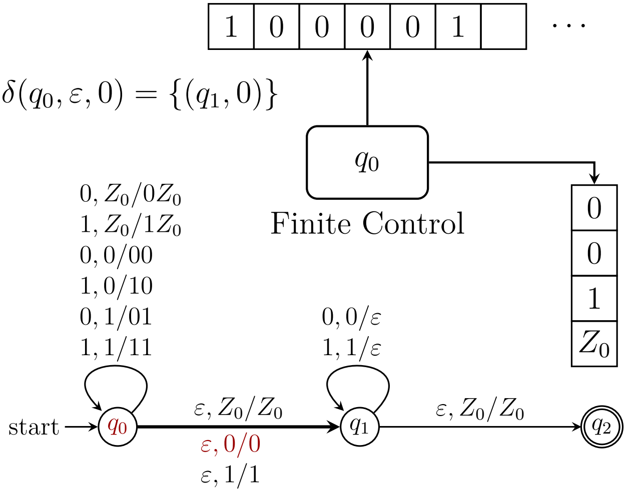
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



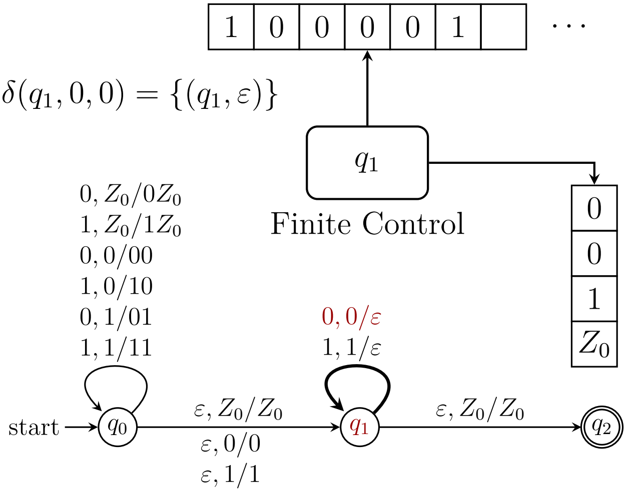
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



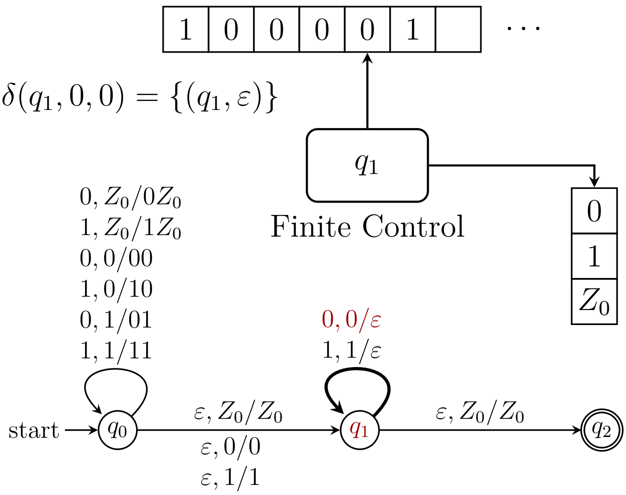
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



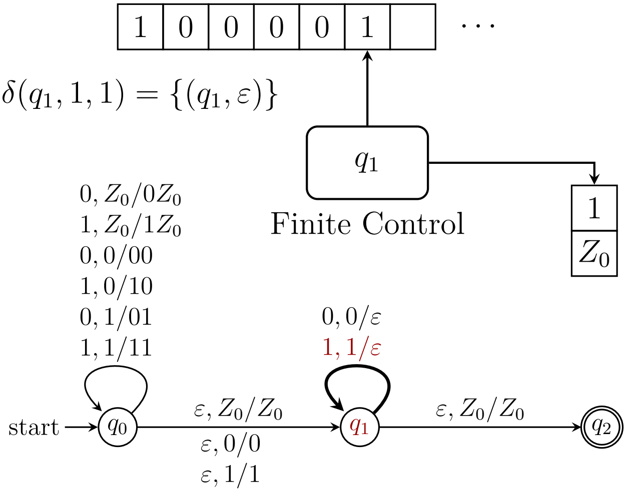
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



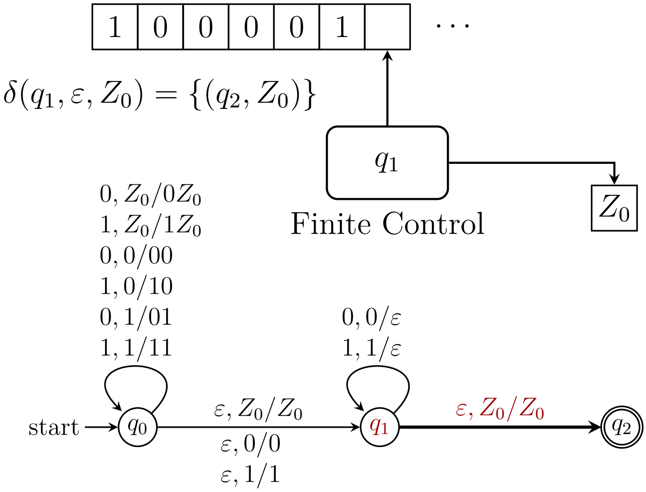
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



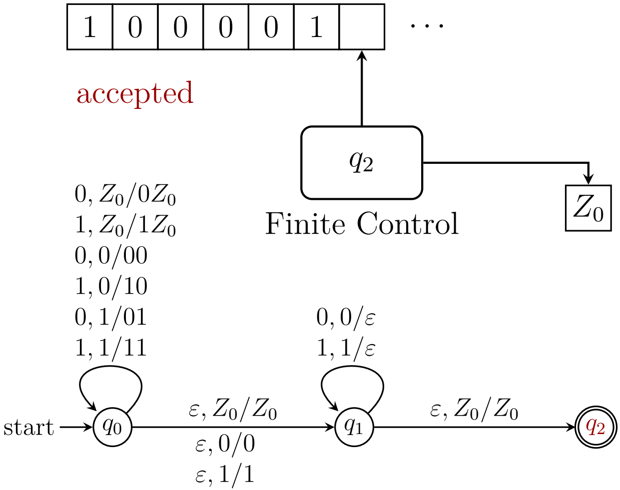
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



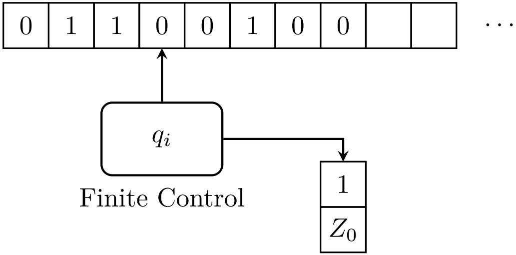
瞬时描述

定义

为描述 PDA 瞬间的格局, 定义 $Q \times \Sigma^* \times \Gamma^*$ 中三元组

$$(q, w, \gamma)$$

为瞬时描述(*ID, Instantaneous Description*), 表示此时 PDA 处于状态 q, 剩余输入串 w, 栈为 γ.



转移符号

定义

在 PDA P 中如果 $(p, \beta) \in \delta(q, a, Z)$, 由 $(q, aw, Z\alpha)$ 到 $(p, w, \beta\alpha)$ 的变化, 称为 ID 的**转移** $\vdash_P$, 记为

$$(q, aw, Z\alpha) \vdash_P (p, w, \beta\alpha)$$

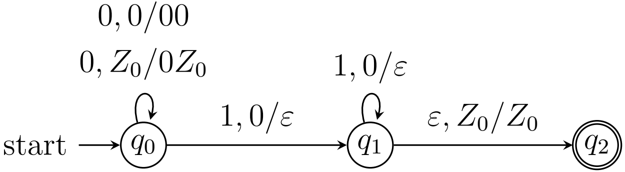
其中 $w \in \Sigma^*$, $\alpha \in \Gamma^*$.

若有 ID I, J 和 K, 递归定义 $\vdash_P^*$ 为:

1. $I \vdash_P^* I$;
2. 若 $I \vdash_P J$, $J \vdash_P^* K$, 则 $I \vdash_P^* K$.

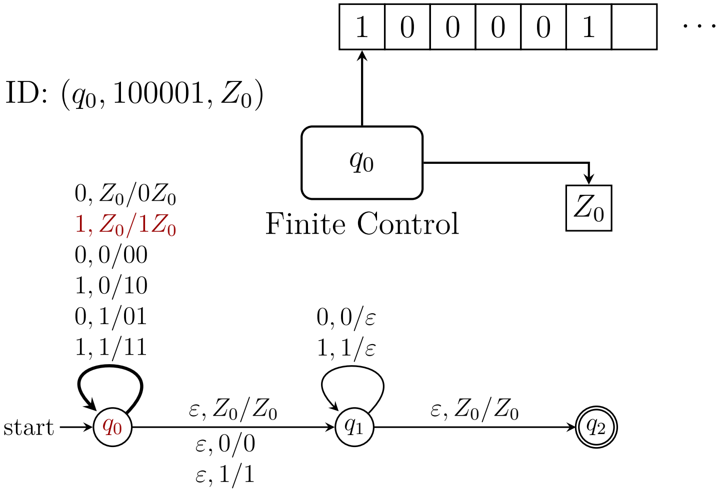
若 P 已知, 可省略, 记为 $\vdash$ 和 $\vdash^*$.

续例 1. 语言 $L_{01} = \{0^n1^n \mid n \geq 1\}$ 的 PDA, 识别 0011 时的 ID 序列.



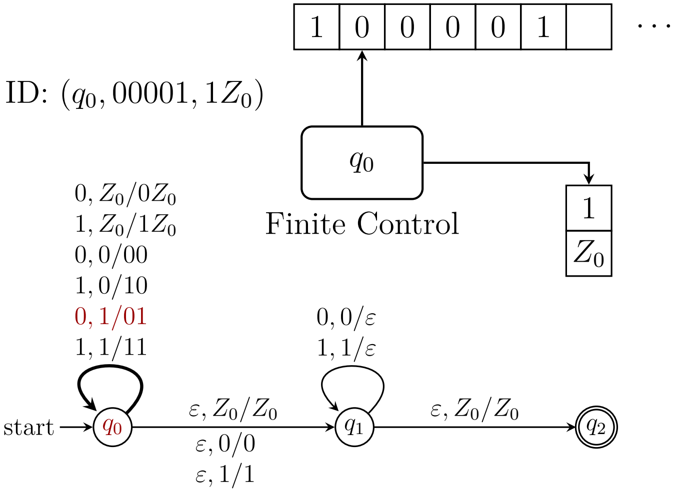
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.

ID: $(q_0, 100001, Z_0)$

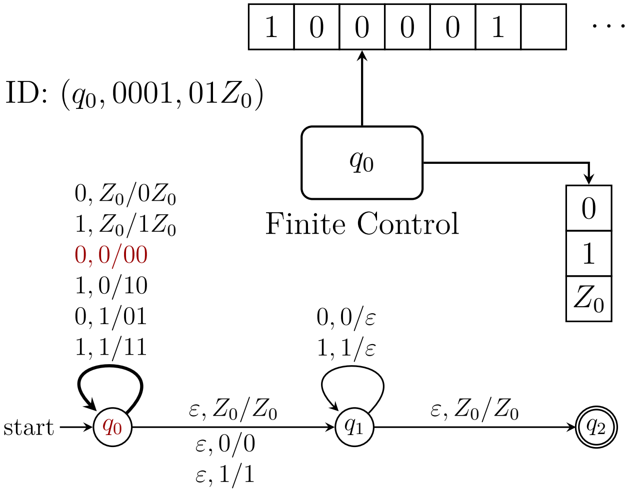


续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.

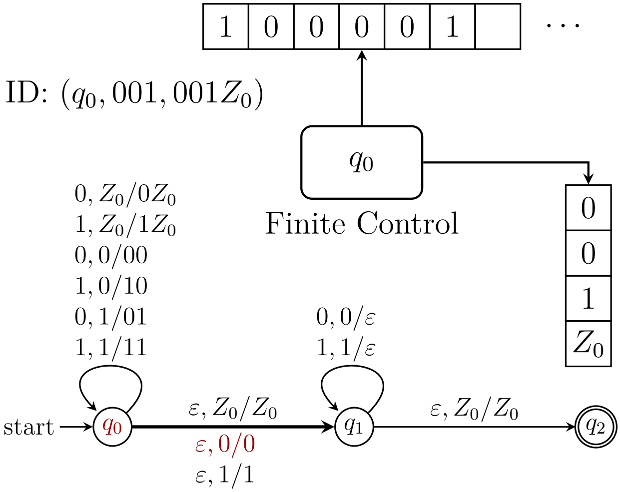
ID: $(q_0, 00001, 1Z_0)$



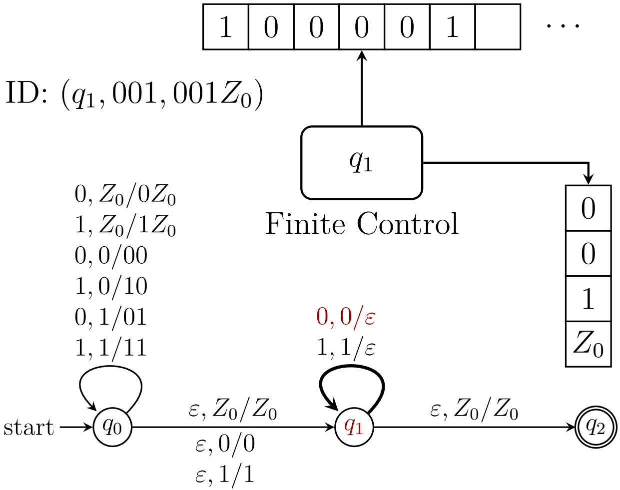
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



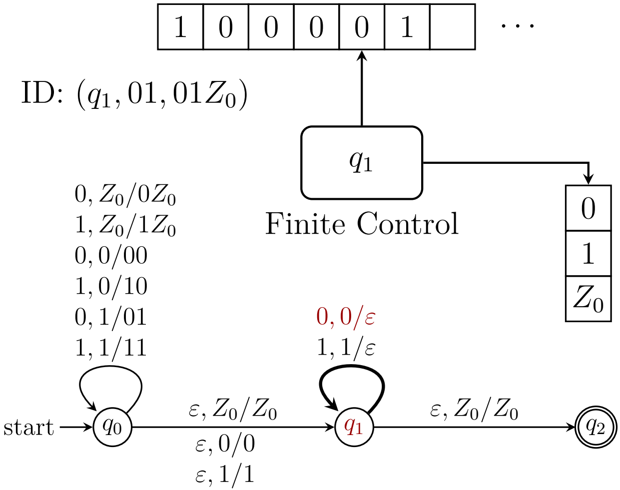
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



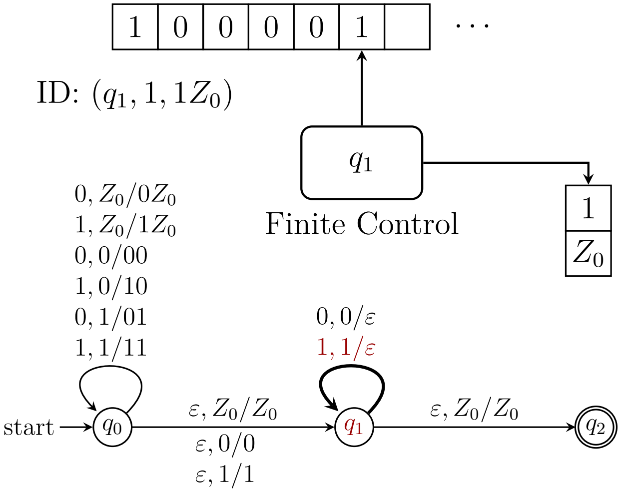
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



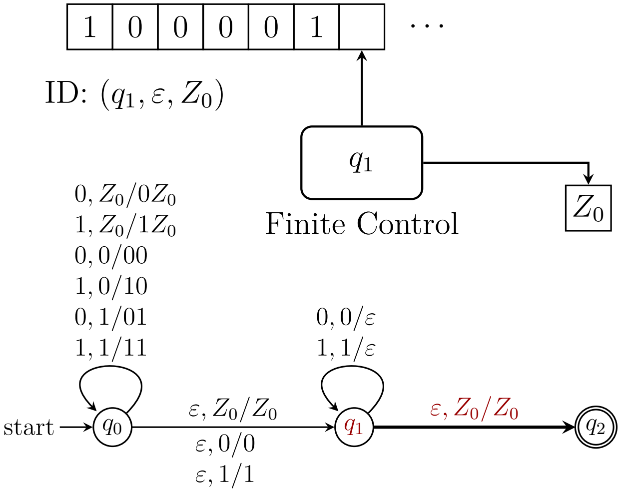
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



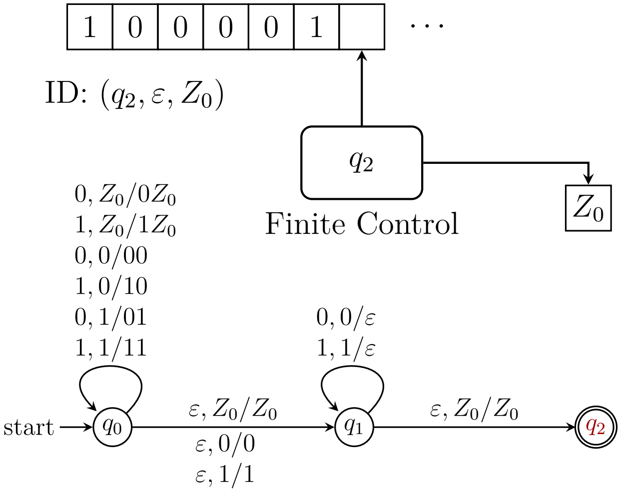
续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



续例 2. $L_{wwr} = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$ 的 PDA 识别串 100001 的过程.



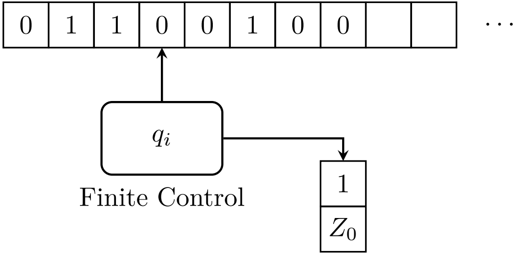
有关 ID 的序列

对 PDA P 的一个合法 ID 序列 (计算):

❶ 把相同的字符串加到所有 ID 的输入串末尾, 所得到的计算合法;

❷ 把相同的栈符号串加到所有 ID 的栈底之下, 所得到的计算合法;

❸ 把所有 ID 中都未消耗的部分输入串去掉, 所得到的计算合法.



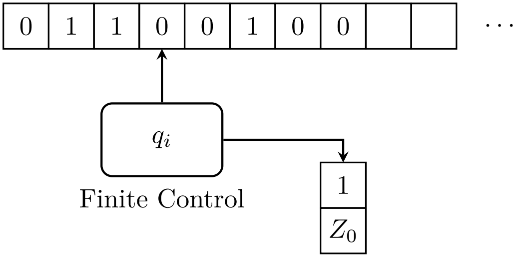
定理 23

对 $\forall w \in \Sigma^*, \forall \gamma \in \Gamma^*$, 如果

$$(q, x, \alpha) \vdash_P^* (p, y, \beta),$$

那么

$$(q, xw, \alpha\gamma) \vdash_P^* (p, yw, \beta\gamma).$$



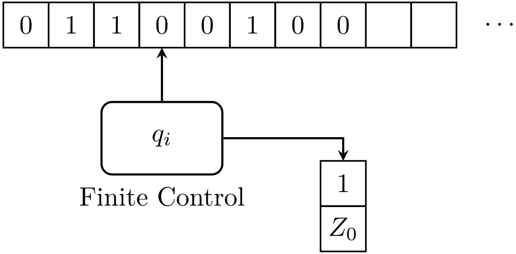
定理 24

对 $\forall w \in \Sigma^*$, 如果

$$(q, xw, \alpha) \vdash_P^* (p, yw, \beta),$$

那么

$$(q, x, \alpha) \vdash_P^* (p, y, \beta).$$



下推自动机

- 下推自动机
- 下推自动机接受的语言
  - 从终态方式到空栈方式
  - 从空栈方式到终态方式
- 下推自动机与文法的等价性
- 确定型下推自动机

下推自动机接受的语言

定义

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, 以两种方式接受语言:

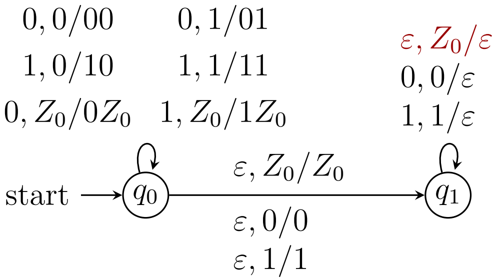
- P 以终态方式接受的语言, 记为$\mathbf{L}(P)$, 定义为

$$\mathbf{L}(P) = \{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \gamma),\ p \in F\}.$$

- P 以空栈方式接受的语言, 记为$\mathbf{N}(P)$, 定义为

$$\mathbf{N}(P) = \{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)\}.$$

续例 2. 识别 L_{wwr} 的 PDA P, 从终态方式接受, 改为空栈方式接受.
用 $\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$ 代替 $\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$ 即可.

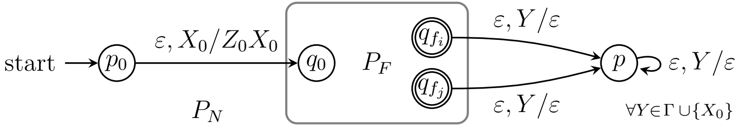


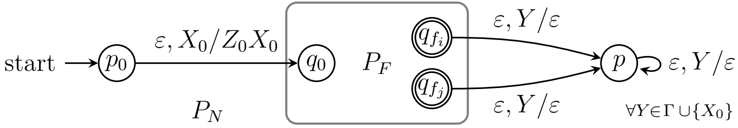
定理 25

如果 PDA P_F 以终态方式接受语言 L, 则存在 PDA P_N 以空栈方式接受 L.

证明: 设 $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$, 构造 PDA P_N,

$$P_N = (Q \cup \{p_0, p\},\ \Sigma,\ \Gamma \cup \{X_0\},\ \delta_N,\ p_0,\ X_0,\ \varnothing).$$





其中 δ_N 定义如下:

❶ P_N 首先将 P_F 的栈底符号压栈, 开始模拟 P_F:

$$\delta_N(p_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\};$$

❷ P_N 模拟 P_F 的动作: $\forall q \in Q$, $\forall a \in \Sigma \cup \{\varepsilon\}$, $\forall Y \in \Gamma$:

$$\delta_N(q, a, Y) \text{ 包含 } \delta_F(q, a, Y) \text{ 的全部元素};$$

❸ 从 $q_f \in F$ 开始弹出栈中符号, 即 $\forall q_f \in F, \forall Y \in \Gamma \cup \{X_0\}$:

$$\delta_N(q_f, \varepsilon, Y) \text{ 包含 } (p, \varepsilon);$$

❹ 在状态 p 时, 弹出全部栈中符号, 即 $\forall Y \in \Gamma \cup \{X_0\}$:

$$\delta_N(p, \varepsilon, Y) = \{(p, \varepsilon)\}.$$

对 $\forall w \in \Sigma^*$ 有

$$
\begin{aligned}
w \in \mathbf{L}(P_F) &\Rightarrow (q_0, w, Z_0) \vdash_{P_F}^{*} (q_f, \varepsilon, \gamma) \\
&\Rightarrow (q_0, w, Z_0X_0) \vdash_{P_F}^{*} (q_f, \varepsilon, \gamma X_0) && \text{定理23} \\
&\Rightarrow (q_0, w, Z_0X_0) \vdash_{P_N}^{*} (q_f, \varepsilon, \gamma X_0) && P_N\text{模拟}P_F \\
&\Rightarrow (p_0, w, X_0) \vdash_{P_N} (q_0, w, Z_0X_0) \vdash_{P_N}^{*} (q_f, \varepsilon, \gamma X_0) && \delta_N\text{构造}\, p_0\,\text{部分} \\
&\Rightarrow (p_0, w, X_0) \vdash_{P_N}^{*} (q_f, \varepsilon, \gamma X_0) \vdash_{P_N}^{*} (p, \varepsilon, \varepsilon) && \delta_N\text{构造}\, q_f\,\text{和}\,p\,\text{部分} \\
&\Rightarrow w \in \mathbf{N}(P_N)
\end{aligned}
$$

即 $\mathbf{L}(P_F) \subseteq \mathbf{N}(P_N)$.

对 $\forall w \in \Sigma^*$ 有

$$
\begin{aligned}
w \in \mathbf{N}(P_N) &\Rightarrow (p_0, w, X_0) \vdash^*_{P_N} (p, \varepsilon, \varepsilon) && \text{其他状态不可能空栈} \\
&\Rightarrow (p_0, w, X_0) \vdash_{P_N} (q_0, w, Z_0X_0) \vdash^*_{P_N} (p, \varepsilon, \varepsilon) && \text{第一个动作必然到 } q_0 \\
&\Rightarrow (q_0, w, Z_0X_0) \vdash^*_{P_N} (q_f, \varepsilon, \gamma X_0) \vdash^*_{P_N} (p, \varepsilon, \varepsilon) && \text{必经 } q_f \in F \text{ 消耗完 } w \\
&\Rightarrow (q_0, w, Z_0) \vdash^*_{P_N} (q_f, \varepsilon, \gamma) && P_N \text{ 中未用过栈底的 } X_0 \\
&\Rightarrow (q_0, w, Z_0) \vdash^*_{P_F} (q_f, \varepsilon, \gamma) && \text{均为模拟 } P_F \\
&\Rightarrow w \in \mathbf{L}(P_F)
\end{aligned}
$$

即 $\mathbf{N}(P_N) \subseteq \mathbf{L}(P_F)$. 所以 $\mathbf{N}(P_N) = \mathbf{L}(P_F)$. $\square$

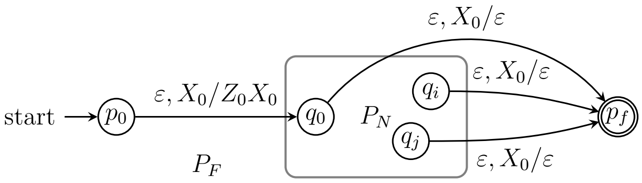
从空栈方式到终态方式

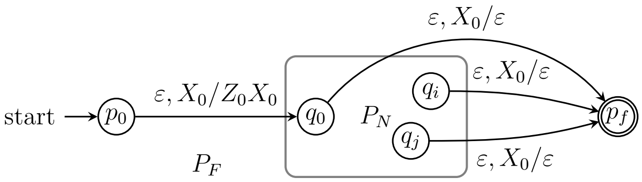
定理 26

如果 *PDA* P_N 以空栈方式接受语言 L, 则存在 *PDA* P_F 以终态方式接受 L.

证明: 设 $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0, \varnothing)$. 构造 PDA P_F,

$$P_F = (Q \cup \{p_0, p_f\},\ \Sigma,\ \Gamma \cup \{X_0\},\ \delta_F,\ p_0,\ X_0,\ \{p_f\})$$





其中 δ_F 定义如下:

❶ P_F 开始时, 将 P_N 栈底符号压入栈, 并开始模拟 P_N,

$$\delta_F(p_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\};$$

❷ P_F 模拟 P_N, $\forall q \in Q$, $\forall a \in \Sigma \cup \{\varepsilon\}$, $\forall Y \in \Gamma$:

$$\delta_F(q, a, Y) = \delta_N(q, a, Y);$$

❸ 在 $\forall q \in Q$ 时, 看到 P_F 的栈底 X_0, 则转移到新终态 p_f:

$$\delta_F(q, \varepsilon, X_0) = \{(p_f, \varepsilon)\}.$$

对 $\forall w \in \Sigma^*$ 有

$$
\begin{aligned}
w \in \mathbf{N}(P_N) &\Rightarrow (q_0, w, Z_0) \vdash^*_{P_N} (q, \varepsilon, \varepsilon) & \\
&\Rightarrow (q_0, w, Z_0X_0) \vdash^*_{P_N} (q, \varepsilon, X_0) & \text{定理23} \\
&\Rightarrow (q_0, w, Z_0X_0) \vdash^*_{P_F} (q, \varepsilon, X_0) & P_F \text{ 模拟 } P_N \\
&\Rightarrow (p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \vdash^*_{P_F} (q, \varepsilon, X_0) & \delta_F \text{ 构造}, p_0 \text{ 部分} \\
&\Rightarrow (p_0, w, X_0) \vdash^*_{P_F} (q, \varepsilon, X_0) \vdash_{P_F} (p_f, \varepsilon, \varepsilon) & \delta_F \text{ 构造}, p_f \text{ 部分} \\
&\Rightarrow (p_0, w, X_0) \vdash^*_{P_F} (p_f, \varepsilon, \varepsilon) & \\
&\Rightarrow w \in \mathbf{L}(P_F) &
\end{aligned}
$$

即 $\mathbf{N}(P_N) \subseteq \mathbf{L}(P_F)$.

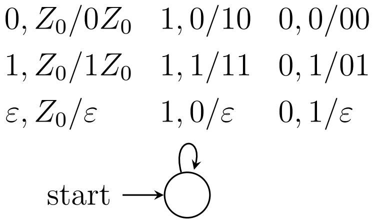
对 $\forall w \in \Sigma^*$ 有

$$
\begin{aligned}
w \in \mathbf{L}(P_F) &\Rightarrow (p_0, w, X_0) \vdash_{P_F}^{*} (p_f, \varepsilon, \varepsilon) \\
&\Rightarrow (p_0, w, X_0) \vdash_{P_F}^{*} (q, \varepsilon, X_0) \vdash_{P_F} (p_f, \varepsilon, \varepsilon) && \text{经 } q \text{ 才可达 } p_f \\
&\Rightarrow (p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \vdash_{P_F}^{*} (q, \varepsilon, X_0) && P_F \text{ 第一个动作} \\
&\Rightarrow (q_0, w, Z_0X_0) \vdash_{P_F}^{*} (q, \varepsilon, X_0) && \text{即上式} \\
&\Rightarrow (q_0, w, Z_0) \vdash_{P_N}^{*} (q, \varepsilon, \varepsilon) && P_N \text{ 与 } X_0 \text{ 无关} \\
&\Rightarrow w \in \mathbf{N}(P_N)
\end{aligned}
$$

即 $\mathbf{N}(P_F) \subseteq \mathbf{L}(P_N)$. 所以 $\mathbf{L}(P_F) = \mathbf{N}(P_N)$. □

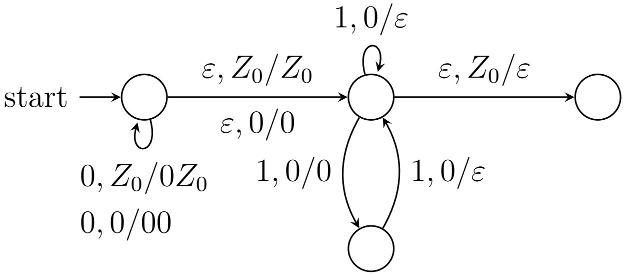
例 3. 接受 $L_{\mathrm{eq}} = \{w \in \{0,1\}^* \mid w$ 中字符 0 和 1 的数量相同$\}$ 的 PDA.

例3. 接受 $L_{\mathrm{eq}} = \{w \in \{0,1\}^* \mid w$ 中字符 0 和 1 的数量相同$\}$ 的 PDA.



例 4. 接受 $L = \{0^n 1^m \mid 0 \le n \le m \le 2n\}$ 的 PDA.

例 4. 接受 $L = \{0^n 1^m \mid 0 \le n \le m \le 2n\}$ 的 PDA.



下推自动机

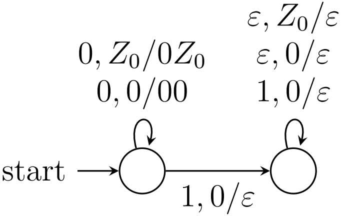
- 下推自动机
- 下推自动机接受的语言
- 下推自动机与文法的等价性
  - 由 CFG 到 PDA
  - 由 PDA 到 CFG
- 确定型下推自动机

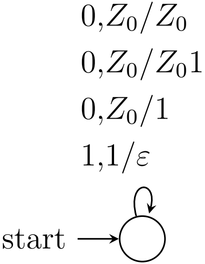
由 CFG 到 PDA

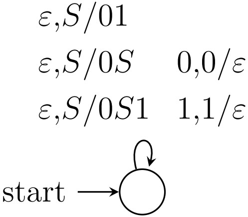
例 5. 设计语言 $L = \{0^n 1^m \mid 1 \leq m \leq n\}$ 的 PDA.

由 CFG 到 PDA

例 5. 设计语言 $L = \{0^n 1^m \mid 1 \le m \le n\}$ 的 PDA.







续例 5. 设计语言 $L = \{0^n 1^m \mid 1 \le m \le n\}$ 的 CFG.

CFG G:

$$
\begin{aligned}
S &\to AB \\
A &\to 0A \mid \varepsilon \\
B &\to 0B1 \mid 01
\end{aligned}
$$

字符串 00011 的最左派生:

$$S \underset{\text{lm}}{\Rightarrow} AB \underset{\text{lm}}{\Rightarrow} 0AB \underset{\text{lm}}{\Rightarrow} 0B \underset{\text{lm}}{\Rightarrow} 00B1 \underset{\text{lm}}{\Rightarrow} 00011$$

续例 5. 语言 $L = \{0^n1^m \mid 1 \le m \le n\}$.

用 PDA 栈顶符号的替换, 模拟文法的最左派生:

PDA		CFG	
PDA 的 ID 转移	PDA 的动作	产生式	最左派生
$(q_0,\ 00011,\ S)$			S
$\vdash (q_0,\ 00011,\ AB)$	$\varepsilon, S/AB$	$S \to AB$	$\underset{\text{lm}}{\Rightarrow} AB$
$\vdash (q_0,\ 00011,\ 0AB)$	$\varepsilon, A/0A$	$A \to 0A$	$\underset{\text{lm}}{\Rightarrow} 0AB$
$\vdash (q_0,\ 0011,\ AB)$	$0, 0/\varepsilon$		
$\vdash (q_0,\ 0011,\ B)$	$\varepsilon, A/\varepsilon$	$A \to \varepsilon$	$\underset{\text{lm}}{\Rightarrow} 0B$
$\vdash (q_0,\ 0011,\ 0B1)$	$\varepsilon, B/0B1$	$B \to 0B1$	$\underset{\text{lm}}{\Rightarrow} 00B1$
$\vdash (q_0,\ 011,\ B1)$	$0, 0/\varepsilon$		
$\vdash (q_0,\ 011,\ 011)$	$\varepsilon, B/01$	$B \to 01$	$\underset{\text{lm}}{\Rightarrow} 00011$
$\vdash (q_0,\ 11,\ 11)$	$0, 0/\varepsilon$		
$\vdash (q_0,\ 1,\ 1)$	$1, 1/\varepsilon$		
$\vdash (q_0,\ \varepsilon,\ \varepsilon)$	$1, 1/\varepsilon$		

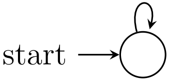
续例 5. 语言 $L = \{0^n 1^m \mid 1 \le m \le n\}$.

$$
\begin{aligned}
S &\to AB \\
A &\to 0A \mid \varepsilon \\
B &\to 0B1 \mid 01
\end{aligned}
$$

$\varepsilon, S/AB$

$\varepsilon, A/0A \quad \varepsilon, A/\varepsilon \quad 0, 0/\varepsilon$

$\varepsilon, B/0B1 \quad \varepsilon, B/01 \quad 1, 1/\varepsilon$



定理 27

任何 *CFL* L, 一定存在 *PDA* P, 使 $L = \mathbf{N}(P)$.

构造与文法等价的 PDA

如果 CFG $G = (V, T, P', S)$, 构造 PDA

$$P = (\{q\}, T, V \cup T, \delta, q, S, \varnothing),$$

其中 δ 为:

❶ $\forall A \in V$:

$$\delta(q, \varepsilon, A) = \{(q, \beta) \mid A \to \beta \in P'\}$$

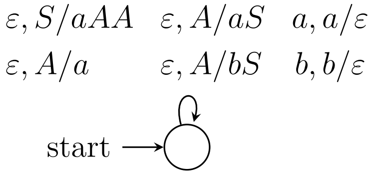
❷ $\forall a \in T$:

$$\delta(q, a, a) = \{(q, \varepsilon)\}$$

那么 $\mathbf{L}(G) = \mathbf{N}(P)$.

例 6. 为文法 $S \to aAA$, $A \to aS \mid bS \mid a$ 构造 PDA.

例 6. 为文法 $S \to aAA,\ A \to aS \mid bS \mid a$ 构造 PDA.



证明: 往证

$$S \overset{*}{\Rightarrow} w \Longleftrightarrow (q, w, S) \vdash_{\mathrm{P}}^{*} (q, \varepsilon, \varepsilon).$$

[充分性] 往证

$$S \underset{\mathrm{lm}}{\overset{*}{\Rightarrow}} w \implies (q, w, S) \vdash^{*} (q, \varepsilon, \varepsilon).$$

设 $S \underset{\mathrm{lm}}{\overset{*}{\Rightarrow}} w$ 中第 i 个左句型为 $x_i A_i \alpha_i$, 其中 $x_i \in \Sigma^*$, $A_i \in V$, $\alpha_i \in (V \cup T)^*$.
并将 S 看作第 0 个左句型 $x_0 A_0 \alpha_0 = S$, 那么

$$x_0 = \varepsilon, A_0 = S, \alpha_0 = \varepsilon.$$

将 w 看作为第 n 个左句型 $x_n A_n \alpha_n = w$, 那么

$$x_n = w, A_n = \varepsilon, \alpha_n = \varepsilon.$$

再对派生步骤 i 归纳, 往证

$$S \underset{\mathrm{lm}}{\overset{i}{\Rightarrow}} x_i A_i \alpha_i \wedge w = x_i y_i \Longrightarrow (q, w, S) \vdash^{*} (q, y_i, A_i \alpha_i).$$

归纳基础: 最左派生在第 0 步时, 显然成立

$$(q, w, S) \vdash^* (q, y_0, A_0\alpha_0) = (q, w, S).$$

归纳递推: 假设第 i 步时成立, 当第 $i+1$ 步时, 一定是 $A_i \to \beta$ 应用到 $x_iA_i\alpha_i$

$$S \overset{i}{\underset{\text{lm}}{\Rightarrow}} x_iA_i\alpha_i \underset{\text{lm}}{\Rightarrow} x_i\beta\alpha_i = x_{i+1}A_{i+1}\alpha_{i+1}.$$

即最左变元 A_{i+1} 一定在 $\beta\alpha_i$ 中, 设 A_{i+1} 之前的终结符为 x', 那么由

$$\begin{aligned} x_i\beta\alpha_i &= x_ix'A_{i+1}\alpha_{i+1} &&= x_{i+1}A_{i+1}\alpha_{i+1} \\ x_iy_i &= x_ix'y_{i+1} &&= x_{i+1}y_{i+1} = w \end{aligned}$$

则有

$$\begin{aligned} \beta\alpha_i &= x'A_{i+1}\alpha_{i+1}, \\ y_i &= x'y_{i+1}. \end{aligned}$$

那么, 在 PDA 中从 ID $(q, y_i, A_i\alpha_i)$ 模拟最左派生, 用产生式 $A_i \to \beta$ 替换栈顶 A_i 后, 有

$$
\begin{aligned}
(q, w, S) &\vdash^* (q, y_i, A_i\alpha_i) && \text{归纳假设} \\
&\vdash (q, y_i, \beta\alpha_i) && A_i \to \beta \\
&= (q, x'y_{i+1}, x'A_{i+1}\alpha_{i+1}) && \\
&\vdash^* (q, y_{i+1}, A_{i+1}\alpha_{i+1}) && \text{弹出栈顶终结符}
\end{aligned}
$$

因此 $S \underset{\text{lm}}{\overset{n}{\Rightarrow}} w \Longrightarrow (q, w, S) \vdash^* (q, y_n, A_n\alpha_n) = (q, \varepsilon, \varepsilon)$, 即充分性得证.

[必要性] 往证更一般的, 对任何变元 A, 都有:

$$(q, x, A) \vdash^* (q, \varepsilon, \varepsilon) \Longrightarrow A \overset{*}{\Rightarrow} x.$$

对 ID 转移 $(q, x, A) \vdash^i (q, \varepsilon, \varepsilon)$ 的次数 i 归纳证明.

归纳基础: 当 $i=1$ 步时, 只能是 $x=\varepsilon$ 且 $A\to\varepsilon$ 为产生式, 所以 $A\overset{*}{\Rightarrow}\varepsilon$.

归纳递推: 假设 $i\leq n$ $(n\geq 1)$ 步时上式成立. 当 $i=n+1$ 时, 因为 A 是变元, 其第 1 步转移一定是应用某产生式 $A\to Y_1Y_2\cdots Y_m$

$$(q,x,A)\vdash(q,x,Y_1Y_2\cdots Y_m)$$

其中 Y_i 是变元或终结符. 而其余的 n 步转移

$$(q,x,Y_1Y_2\cdots Y_m)\overset{*}{\vdash}(q,\varepsilon,\varepsilon)$$

中每个 Y_i 从栈中被完全弹出时, 将消耗掉的那部分 x 记为 x_i, 那么显然有

$$x=x_1x_2\cdots x_m.$$

而每个 Y_i 从栈中被完全弹出时, 都不超过 n 步, 所以由归纳假设,

$$(q,x_i,Y_i)\overset{*}{\vdash}(q,\varepsilon,\varepsilon)\Longrightarrow Y_i\overset{*}{\Rightarrow}x_i.$$

再由产生式 $A \to Y_1 Y_2 \cdots Y_m$, 有

$$\begin{aligned} A &\Rightarrow Y_1 Y_2 \cdots Y_m \\ &\stackrel{*}{\Rightarrow} x_1 Y_2 \cdots Y_m \\ &\stackrel{*}{\Rightarrow} x_1 x_2 \cdots Y_m \\ &\stackrel{*}{\Rightarrow} x_1 x_2 \cdots x_m = x. \end{aligned}$$

因此当 $A = S$, $x = w$ 时,

$$(q, w, S) \vdash^* (q, \varepsilon, \varepsilon) \Longrightarrow S \stackrel{*}{\Rightarrow} w$$

成立, 即必要性得证.

所以, 任何 CFL 都可由 PDA 识别. $\square$

构造与 GNF 格式文法等价的 PDA

如果 GNF 格式的 CFG $G = (V, T, P', S)$, 那么构造 PDA

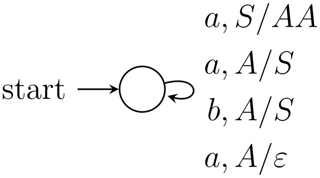
$$P = (\{q\},\ T,\ V,\ \delta,\ q,\ S,\ \varnothing),$$

为每个产生式, 定义 δ 为:

$$\delta(q, a, A) = \{(q, \beta) \mid A \to a\beta \in P'\}.$$

续例6. 文法 $S \to aAA$, $A \to aS \mid bS \mid a$ 为 GNF 格式, 构造等价的 PDA.

续例 6. 文法 $S \to aAA$, $A \to aS \mid bS \mid a$ 为 GNF 格式, 构造等价的 PDA.



由 PDA 到 CFG

定理 28

如果 PDA P, 有 $L = \mathbf{N}(P)$, 那么 L 是上下文无关语言.

构造与 PDA 等价的 CFG

如果 PDA $P=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,\varnothing)$, 那么构造 CFG $G=(V,\Sigma,P',S)$, 其中 V 和 P' 为

❶ $V=\{[qXp]\mid p,q\in Q,\ X\in\Gamma\}\cup\{S\}$;

❷ 对 $\forall p\in Q$, 构造产生式 $S\to[q_0Z_0p]$;

❸ 对 $\forall(p,Y_1Y_2\cdots Y_n)\in\delta(q,a,X)$, 构造 $|Q|^n$ 个产生式

$$[qXr_n]\to a[pY_1r_1][r_1Y_2r_2]\cdots[r_{n-1}Y_nr_n]$$

其中 $a\in\Sigma\cup\{\varepsilon\}$, $X,Y_i\in\Gamma$, 而 $r_i\in Q$ 是 n 次 $|Q|$ 种状态的组合; 若 $i=0$, 为 $[qXp]\to a$.

证明: 只需证明

$$(q, w, X) \vdash^* (p, \varepsilon, \varepsilon) \iff [qXp] \overset{*}{\Rightarrow} w.$$

并令 $X = Z_0$, $q = q_0$, 与开始符号 S 的产生式一起, 即可完成定理的证明.

[充分性] 对 PDA 中 $(q, w, X) \vdash^* (p, \varepsilon, \varepsilon)$ 的转移次数 i 归纳证明.

归纳基础: 当 $i = 1$ 时, P 只能消耗不超过一个的字符, 即 $w = a$

$$(q, w, X) = (q, a, X) \vdash (p, \varepsilon, \varepsilon),$$

其中 $a \in \Sigma \cup \{\varepsilon\}$ 且 $(p, \varepsilon) \in \delta(q, a, X)$, 则由文法的构造会有

$$[qXp] \to a,$$

因此 $[qXp] \overset{*}{\Rightarrow} a = w$.

归纳递推: 假设当 $i \leq m$ $(m \geq 1)$ 时命题成立. 当 $i = m+1$ 时, 转移的第 1 步, 一定由某个 $(r_0, Y_1Y_2\cdots Y_n) \in \delta(q, a, X)$ 开始

$$(q, ax, X) \vdash (r_0, x, Y_1Y_2\cdots Y_n),$$

其中 $a \in \Sigma \cup \{\varepsilon\}, w = ax$. 而其余的 m 步为

$$(r_0, x, Y_1Y_2\cdots Y_n) \overset{*}{\vdash} (p, \varepsilon, \varepsilon).$$

而这些转移, 会从栈中依次弹出 Y_i 并消耗掉部分 x. 若分别记为 x_i, 则有

$$w = ax = ax_1x_2\cdots x_n.$$

若设弹出 Y_i 之前和之后的状态分别是 r_{i-1} 和 r_i, 这里 $i = 1, 2, \cdots n$, 那么有

$$(r_{i-1}, x_i, Y_i) \overset{*}{\vdash} (r_i, \varepsilon, \varepsilon),$$

且转移步数都不会超过 m. 那么, 由归纳假设有

$$(r_{i-1}, x_i, Y_i) \overset{*}{\vdash} (r_i, \varepsilon, \varepsilon) \implies [r_{i-1}Y_ir_i] \overset{*}{\Rightarrow} x_i.$$

而由动作 $(r_0, Y_1Y_2\cdots Y_n) \in \delta(q, a, X)$ 所构造的产生式会包含

$$[qXr_n] \to a[r_0Y_1r_1][r_1Y_2r_2]\cdots[r_{n-1}Y_nr_n].$$

而显然弹出 X 后的状态 p 与弹出 Y_n 后的状态 r_n 是同一个, 所以

$$[qXp] = [qXr_n] \Rightarrow a[r_0Y_1r_1][r_1Y_2r_2]\cdots[r_{n-1}Y_nr_n] \overset{*}{\Rightarrow} ax_1x_2\cdots x_n = w$$

因此充分性得证. 那么当 $X = Z_0, q = q_0$ 时有

$$(q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon) \Longrightarrow [q_0Z_0p] \overset{*}{\Rightarrow} w,$$

以及产生式 $S \to [q_0Z_0p]$ 有 $S \overset{*}{\Rightarrow} w$, 即 PDA 接受的串可由文法派生得到.

[必要性]: 略. $\square$

例 7. 将 PDA $P=(\{p,q\},(0,1),\{X,Z\},\delta,q,Z)$ 转为 CFG, 其中 δ 如下:

(1) $\delta(q,1,Z)=\{(q,XZ)\}$

(2) $\delta(q,1,X)=\{(q,XX)\}$

(3) $\delta(q,0,X)=\{(p,X)\}$

(4) $\delta(q,\varepsilon,Z)=\{(q,\varepsilon)\}$

(5) $\delta(p,1,X)=\{(p,\varepsilon)\}$

(6) $\delta(p,0,Z)=\{(q,Z)\}$

δ	产生式
(0)	$S \to [qZq]$
	$S \to [qZp]$
(1)	$[qZq] \to 1[qXq][qZq]$
	$[qZq] \to 1[qXp][pZq]$
	$[qZp] \to 1[qXq][qZp]$
	$[qZp] \to 1[qXp][pZp]$
(2)	$[qXq] \to 1[qXq][qXq]$
	$[qXq] \to 1[qXp][pXq]$
	$[qXp] \to 1[qXq][qXp]$
	$[qXp] \to 1[qXp][pXp]$
(3)	$[qXq] \to 0[pXq]$
	$[qXp] \to 0[pXp]$
(4)	$[qZq] \to \varepsilon$
(5)	$[pXp] \to 1$
(6)	$[pZp] \to 0[qZp]$
	$[pZq] \to 0[qZq]$

消除无用符号	重命名 (可选)
$S \to [qZq]$	$S \to A$
$[qZq] \to 1[qXp][pZq]$	$A \to 1BC$
$[qXp] \to 1[qXp][pXp]$	$B \to 1BD$
$[qXp] \to 0[pXp]$	$B \to 0D$
$[qZq] \to \varepsilon$	$A \to \varepsilon$
$[pXp] \to 1$	$D \to 1$
$[pZq] \to 0[qZq]$	$C \to 0A$

- 下推自动机
- 下推自动机接受的语言
- 下推自动机与文法的等价性
- 确定型下推自动机
  - 正则语言与 DPDA
  - DPDA 与无歧义文法

确定型下推自动机

定义

如果 $PDA\ P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ 满足

❶ $\forall a \in \Sigma \cup \{\varepsilon\}$, $\delta(q, a, X)$ 至多有一个动作;

❷ $\exists a \in \Sigma$, 如果 $\delta(q, a, X) \neq \varnothing$, 那么 $\delta(q, \varepsilon, X) = \varnothing$.

则称 P 为确定型下推自动机(*DPDA*).

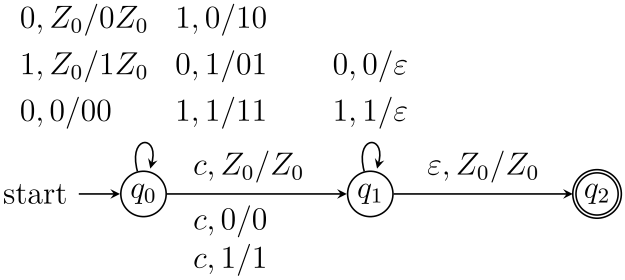
DPDA P 以终态方式接受的语言 $\mathbf{L}(P)$ 称为 *DCFL*.

- DPDA 中 $\forall(q, a, Z) \in Q \times \Sigma \times \Gamma$ 满足 $|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leq 1$

DPDA 与 PDA 不等价

例 8. 任何 DPDA 都无法接受 L_{wwr}, 但是可以接受

$$L_{wcwr} = \{wcw^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}.$$



正则语言与 DPDA

定理 29

如果 L 是正则语言，那么存在 $DPDA\ P$ 以终态方式接受 L，即 $L = \mathbf{L}(P)$.

证明：显然，因为 DPDA P 可以不用栈而模拟任何 DFA. $\square$

- L_{wcwr} 显然是 CFL，所以 DCFL 语言类真包含正则语言
- DPDA 无法识别 L_{wwr}，所以 DCFL 语言类真包含于 CFL

定义

如果语言 L 中不存在两个不同的字符串 x 和 y, 使 x 是 y 的前缀, 称语言 L 满足**前缀性质**.

定理 30

如果有 $DPDA$ P 且 $L = \mathbf{N}(P)$, 当且仅当 L 有前缀性质且存在 $DPDA$ P' 使 $L = \mathbf{L}(P')$.

证明: $[\Rightarrow]$ $\forall x \in \mathbf{N}(P)$ 会弹空 P 的栈, 所以不会接受以 x 为前缀的其他串; 而转换为终态方式不改变确定性. $[\Leftarrow]$ 到达终态则弹空栈, 即可. $\square$

- DPDA P 的 $\mathbf{N}(P)$ 更有限, 即使正则语言 $\mathbf{0}^*$ 也无法接受

定理 31

DPDA P, 语言 $L = \mathbf{N}(P)$, 那么 L 有无歧义的 *CFG*.

证明: 利用定理 28 由 P 构造的文法 G 一定无歧义, 因为:

1. P 是确定的, 那么它接受 w 的 ID 序列也是确定的;
2. 而由 $\delta(q, a, X) = \{(p, Y_1 \cdots Y_n)\}$ 继续弹出 Y_i 后的状态 r_i 也是确定的;
3. 那么由每个动作构造的一组产生式

$$[qXr_n] \to a[pY_1r_1][r_1Y_2r_2] \cdots [r_{n-1}Y_nr_n]$$

   中, 仅会有一个是有效的;
4. 那么, G 中最左派生 $S \overset{*}{\underset{\text{lm}}{\Rightarrow}} w$ 就是唯一的, 所以是无歧义的. □

定理 32

DPDA P, 语言 $L = \mathbf{L}(P)$, 那么 L 有无歧义的 *CFG*.

证明:

❶ 设符号 \$ 不在 L 中出现, 令 $L' = \{w\$ \mid w \in L\}$, 则 L' 具有前缀性质;

❷ 可修改 P 接受 L', 则由定理 30, 存在 DPDA P' 使 $\mathbf{N}(P') = L'$;

❸ 由定理 31, 存在无歧义文法 G' 使 $\mathbf{L}(G') = L'$;

❹ 将 \$ 看作变元, 增加产生式 $\$ \to \varepsilon$, 修改 G' 为文法 G;

❺ 则文法 G 和 G' 一样无歧义, 且 $\mathbf{L}(G) = L$. □

DCFL/DPDA 的重要应用

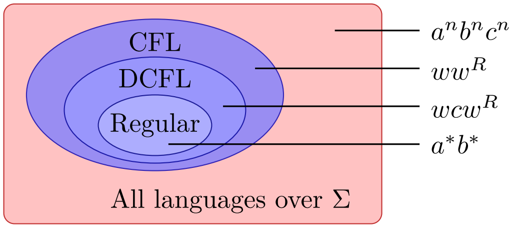
- 程序设计语言的语法分析器

  如 LR(k) 文法, Yacc 的基础, 解析的时间复杂度为 $O(n)$ 的算法

- 非固有歧义语言的真子集

  如 L_{wwr} 有无歧义文法 $S \to 0S0 \mid 1S1 \mid \varepsilon$

语言类之间的关系


CFL
DCFL
Regular
All languages over Σ
$a^n b^n c^n$
ww^R
wcw^R
$a^* b^*$

形式语言与自动机理论

上下文无关语言的性质

王春宇

计算机科学与技术学院
哈尔滨工业大学

上下文无关语言的性质

- 上下文无关语言的泵引理
  - 上下文无关语言的泵引理
  - 泵引理的应用
- 上下文无关语言的封闭性
- 上下文无关语言的判定性质
- 乔姆斯基文法体系

任何 Σ 上的所有语言是不可数的

不妨设 $\Sigma = \{a\}$, 对任何 $0 \le x < 1$ 的实数 x, 定义语言

$$L_x = \{a^n \mid x \cdot 2^n \bmod 1 \ge 1/2\},$$

即 $a^n \in L_x$ 当且仅当 x 二进制表示的第 $n+1$ 位为 1.

1. 如果 $x \neq y$, 则 x 和 y 一定有某些位不同, 所以 $L_x \neq L_y$;
2. 所以 Σ 上的所有语言, 至少与 0 和 1 之间的实数一样多;
3. 因此, Σ 上的所有语言是不可数的.

任何 Σ 上的所有 CFL 是可数的

任何 CFG $G = (V, \Sigma, P, S)$ 可由符号集 $V \cup \Sigma \cup \{\varepsilon, \boldsymbol{\rightarrow}, |, \Diamond\}$ 编码.

- 如文法 $S \to A \mid B,\ A \to aA \mid aC,\ B \to Bb \mid Cb,\ C \to \varepsilon \mid aCb$ 可编码为

$$S{\boldsymbol{\rightarrow}}A|B\Diamond A{\boldsymbol{\rightarrow}}aA|aC\Diamond B{\boldsymbol{\rightarrow}}Bb|Cb\Diamond C{\boldsymbol{\rightarrow}}\varepsilon|aCb;$$

- 用 0/1 编码这些符号

$\varepsilon \mapsto 10$	$a \mapsto 110$	$S \mapsto 1110$
$\boldsymbol{\rightarrow} \mapsto 100$	$b \mapsto 1100$	$A \mapsto 11100$
$\mid\ \mapsto 1000$		$B \mapsto 111000$
$\Diamond \mapsto 10000$		$C \mapsto 1110000.$

- 文法编码再转换为 0/1 字符串

$$\begin{gathered}11101001110010001110001000011100100110111001000110\\111000001000011100010011100011001000111000011001000\\01110000010010100011011100001100;\end{gathered}$$

- 当作二进制表示则为整数

$$2486025347845581444133243339142670726924.$$

- 而 Σ 上两个文法如果不同, 这样编码会得到不同的整数;
- 因此 Σ 上所有 CFL 至多与正整数一样多, 是可数的.
- 因此, 并非所有的语言都是 CFL.

语法分析树的大小

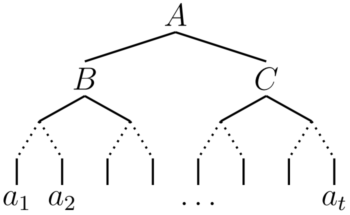
定理 33

对于乔姆斯基范式文法 $G=(V,T,P,S)$ 的语法树，如果产物为终结符串 w，且树中最长路径的长度是 n，那么 $|w|\le 2^{n-1}$.

证明：对最长路径的长度归纳.

基础：为 1 时，只能是 $\begin{array}{c}A\\ |\\ a\end{array}$，显然成立.

递推：为 n 时根节点一定是 $A\to BC$，而 B 和 C 子树最长路径最多为 $n-1$，由归纳假设，产物最长都为 2^{n-2}. 因此整棵树产物最长 $2^{n-2}+2^{n-2}=2^{n-1}$. $\square$



定理 34

如果语言 L 是 CFL, 那么存在正整数 N, 它只依赖于 L, 对 $\forall z \in L$, 只要 $|z| \geq N$, 就可以将 z 分为五部分 $z = uvwxy$ 满足:

1. $vx \neq \varepsilon \quad$ (或$|vx| > 0$);
2. $|vwx| \leq N$;
3. $\forall i \geq 0,\ uv^iwx^iy \in L$.

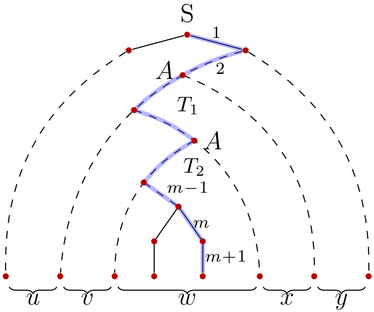
证明:

❶ 设 CNF 格式 CFG G 中变元数 $|V| = m$, 令 $N = 2^m$, 若有 $z \in L(G)$, 且 $|z| \geq N$.

❷ 则 z 的派生树内节点是二叉树, 最长路径长度至少 $m+1$, 节点至少 $m+2$ 个.

❸ 该路径由下至上 $m+1$ 个内节点中, 必有两个 T_2 和 T_1 标记了相同的变元 A.

❹ 若记 T_2 产物为 w, 且是 T_1 的子树, T_1 的产物可记为 vwx, 则有 $A \overset{*}{\Rightarrow} vAx$ 和 $A \overset{*}{\Rightarrow} w$.



❺ 那么 $\forall i \geq 0$, $A \overset{*}{\Rightarrow} v^i w x^i$. 不妨设 $z = uvwxy$, 则 $S \overset{*}{\Rightarrow} uAy \overset{*}{\Rightarrow} uv^i w x^i y$.

❻ T_1 路径长不超过 $m+1$, 那么 T_1 产物长不超过 2^m, 所以 $|vwx| \leq 2^m$.

❼ T_2 必在 T_1 的左/右儿子中, 所以 v 和 x 不可能同时为空, 即 $vx \neq \varepsilon$.

$\square$

泵引理的应用

例 1. 证明 $L = \{0^n 1^n 2^n \mid n \geq 1\}$ 不是上下文无关语言.

泵引理的应用

例 1. 证明 $L = \{0^n1^n2^n \mid n \geq 1\}$ 不是上下文无关语言.

证明:

1. 假设 L 是 CFL, 那么存在整数 N, 对 $\forall z \in L\,(|z| \geq N)$ 满足泵引理.
2. 从 L 中取 $z = 0^N1^N2^N$, 则显然 $z \in L$ 且 $|z| = 3N \geq N$.
3. 由泵引理, z 可被分为 $z = uvwxy$, 且有 $|vwx| \leq N$ 和 $vx \neq \varepsilon$.
4. 那么 vwx 只能包含一种或两种字符:
   1. 一种字符, 或为 0, 或为 1, 或为 2, 那么 $uwy \notin L$;
   2. 两种字符, 或为 0 和 1, 或为 1 和 2, 那么也有 $uwy \notin L$;
5. 与泵引理 $uwy = uv^0wx^0y \in L$ 矛盾, 假设不成立.
6. L 不是上下文无关的. □

例 2. 证明 $L=\{ww \mid w\in\{0,1\}^*\}$ 不是上下文无关的.

(错误的) 证明: 假设 L 是 CFL. 取 $z=0^N10^N1$, 那么 $z=uvwxy$ 为

$$z=\overbrace{\underbrace{00\cdots00}_{u}\ \underbrace{0}_{v}\ \underbrace{1}_{w}}^{0^N1}\ \overbrace{\underbrace{0}_{x}\ \underbrace{00\cdots01}_{y}}^{0^N1}$$

则对任意 $i\geq 0$, 有 $uv^iwx^iy\in L$, 满足泵引理. □

(正确的) 证明: 假设 L 是 CFL. 取 $z=0^N1^N0^N1^N$, 将 z 分为 $z=uvwxy$ 时

❶ 若 vwx 在 z 中点的一侧, uv^0wx^0y 显然不可能属于 L;

❷ 若 vwx 包括 z 中点, 那么 uv^0wx^0y 为 $0^N1^i0^j1^N$, 也不可能属于 L.

所以假设不成立, L 不是 CFL. □

CFL 的泵引理同样只是必要条件

有些非 CFL, 泵引理对它们没有什么作用. 例如

$$L = \{a^i b^j c^k d^l \mid i = 0 \text{ 或 } j = k = l\}$$

不是上下文无关的.

- 如果选 $z = b^j c^k d^l$, 则可以让 $z = uvwxy$ 的 vwx 只含有 b, 那么对任何 m, 都有 $uv^m wx^m y \in L$;
- 如果选 $z = a^i b^j c^j d^j$, 则可以让 v 和 x 只包含 a, 那么对任何 m, 都有 $uv^m wx^m y \in L$.

所以无法使用泵引理证明 L 非 CFL.

Ogden 引理 (的较弱形式)

如果语言 L 是 CFL, 那么存在正整数 N, 它只依赖于 L, 对 $\forall z \in L$, 在 z 中至少 N 个任意位置作标记后, 就可以将 z 分为五部分 $z = uvwxy$ 满足:

1. v 和 x 一起至少含有一个标记位置;
2. vwx 中至多有 N 个标记位置;
3. $\forall i \geq 0, uv^iwx^iy \in L$.

上下文无关语言的性质

- 上下文无关语言的泵引理
- 上下文无关语言的封闭性
  - 代换的封闭性
  - 并/连接/闭包/同态/逆同态/反转的封闭性
  - 交和补运算不封闭
  - 封闭性的应用
- 上下文无关语言的判定性质
- 乔姆斯基文法体系

代换

定义

两个字母表 Σ 到 Γ 的函数 $s : \Sigma \to 2^{\Gamma^*}$ 称为代换. Σ 中的一个字符 a 在 s 的作用下为 Γ 上的一个语言 L_a, 即

$$s(a) = L_a.$$

扩展 s 的定义到字符串,

$$s(\varepsilon) = \{\varepsilon\}$$

$$s(xa) = s(x)s(a)$$

再扩展 s 到语言, 对 $\forall L \subseteq \Sigma^*$,

$$s(L) = \bigcup_{x \in L} s(x).$$

定理 35

如果有 Σ 上的 *CFL* L 和代换 s, 且每个 $a \in \Sigma$ 的 $s(a)$ 都是 *CFL*, 那么 $s(L)$ 也是 *CFL*.

构造方法

设 CFL L 的文法 $G = (V,\, T,\, P,\, S)$, 每个 $s(a)$ 的文法 $G_a = (V_a, T_a, P_a, S_a)$. 那么 $s(L)$ 的文法可以构造为

$$G' = (V',\, T',\, P',\, S):$$

❶ $V' = V \cup \left(\bigcup_{a\in T} V_a\right)$

❷ $T' = \bigcup_{a\in T} T_a$

❸ P' 包括每个 P_a 和 P 中产生式, 但是要将 P 的产生式中每个终结符 a 均替换为文法 G_a 的开始符号 S_a.

证明: 对 $\forall w \in s(L)$, 那么一定存在某个 $x = a_1a_2\cdots a_n \in L$ 使

$$w \in s(x) = s(a_1)s(a_2)\cdots s(a_n).$$

那么 w 可以分为 $w = w_1w_2\cdots w_n$ 且 $w_i \in s(a_i)$, 即

$$S_{a_i} \overset{*}{\underset{G_{a_i}}{\Rightarrow}} w_i.$$

由于 $S \overset{*}{\underset{G}{\Rightarrow}} x = a_1a_2\cdots a_n$, 所以

$$S \overset{*}{\underset{G'}{\Rightarrow}} S_{a_1}S_{a_2}\cdots S_{a_n} \overset{*}{\underset{G'}{\Rightarrow}} w_1w_2\cdots w_n = w,$$

所以 $w \in \mathbf{L}(G')$, 即 $s(L) \subseteq \mathbf{L}(G')$.

因为 G' 的终结符仅能由每个 S_a 派生, 因此对 $\forall w \in \mathbf{L}(G')$ 有

$$S \underset{G'}{\overset{*}{\Rightarrow}} \alpha = S_{a_1} S_{a_2} \cdots S_{a_n} \underset{G'}{\overset{*}{\Rightarrow}} w.$$

因为 G' 中的每个 S_a 在 G 中是终结符 a, 所以

$$S \underset{G}{\overset{*}{\Rightarrow}} a_1 a_2 \cdots a_n = x \in L$$

又因为 $\alpha = S_{a_1} \cdots S_{a_n} \underset{G'}{\overset{*}{\Rightarrow}} w = w_1 \cdots w_n$, 所以 $S_{a_i} \underset{G'}{\overset{*}{\Rightarrow}} w_i$, 即 $w_i \in s(a_i)$. 那么

$$w = w_1 w_2 \cdots w_n \in s(a_1) s(a_2) \cdots s(a_n) = s(a_1 a_2 \cdots a_n) = s(x) \subseteq s(L),$$

所以 $w \in s(L)$, 即 $\mathbf{L}(G') \subseteq s(L)$. 因此 $\mathbf{L}(G') = s(L)$. $\square$

例 3. 设 $L = \{w \in \{a,b\}^* \mid w$ 有相等个数的 a 和 $b\}$, 代换

$$s(a) = L_a = \{0^n 1^n \mid n \geq 1\}$$
$$s(b) = L_b = \{ww^R \mid w \in (\mathbf{0}+\mathbf{1})^*\}$$

求 $s(L)$ 的文法.

解: 设计 L 的文法为: $S \to aSbS \mid bSaS \mid \varepsilon$

L_a 的文法为: $S_a \to 0S_a1 \mid 01$

L_b 的文法为: $S_b \to 0S_b0 \mid 1S_b1 \mid \varepsilon$

那么 $s(L)$ 的文法为:

$$S \to S_aSS_bS \mid S_bSS_aS \mid \varepsilon$$
$$S_a \to 0S_a1 \mid 01$$
$$S_b \to 0S_b0 \mid 1S_b1 \mid \varepsilon$$

CFL 对并/连接/闭包/同态封闭

定理 36

上下文无关语言在并，连接，闭包，正闭包，同态下封闭.

CFL 对并/连接/闭包/同态封闭

定理 36

上下文无关语言在并, 连接, 闭包, 正闭包, 同态下封闭.

证明 1: 设 $\Sigma = \{1, 2\}$, L_1, L_2 是任意 CFL. 定义代换

$$s(1) = L_1, \quad s(2) = L_2.$$

语言 $\{1, 2\}$, $\{12\}$, $\{1\}^*$ 和 $\{1\}^+$ 显然都是 CFL, 那么

❶ 由 $s(\{1, 2\}) = s(1) \cup s(2) = L_1 \cup L_2$, 所以并运算封闭;

❷ 由 $s(\{12\}) = s(12) = s(\varepsilon)s(1)s(2) = L_1L_2$, 所以连接运算封闭;

❸ 闭包和正比包运算封闭, 因为

$$\begin{aligned} s(\{1\}^*) &= s(\{\varepsilon,\ 1,\ 11,\ \cdots\}) \\ &= s(\varepsilon) \cup s(1) \cup s(11) \cup \cdots \\ &= \{\varepsilon\} \cup s(1) \cup s(1)s(1) \cup \cdots \\ &= L_1^*. \end{aligned}$$

若 h 是 Σ 上的同态, L 是 Σ 上的 CFL, 对 $\forall a \in \Sigma$ 令代换 $s'(a) = \{h(a)\}$, 则

$$h(L) = \{h(w) \mid w \in L\} = \bigcup_{w\in L} \{h(w)\} = \bigcup_{w\in L} s'(w) = s'(L),$$

所以同态运算封闭. $\square$

证明 2: 用文法证明并/连接/闭包的封闭性. 设 CFL L_1 和 L_2 的文法分别为

$$G_1 = (V_1,\, T_1,\, P_1,\, S_1),\ G_2 = (V_2,\, T_2,\, P_2,\, S_2)$$

那么, 分别构造

❶ $L_1 \cup L_2$ 的文法为

$$G_{\text{union}} = (V_1 \cup V_2 \cup \{S\},\ T_1 \cup T_2,\ P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\},\ S);$$

❷ $L_1 L_2$ 的文法为

$$G_{\text{concat}} = (V_1 \cup V_2 \cup \{S\},\ T_1 \cup T_2,\ P_1 \cup P_2 \cup \{S \to S_1 S_2\},\ S);$$

❸ L_1^* 的文法为

$$G_{\text{closure}} = (V_1 \cup \{S\},\ T_1,\ P_1 \cup \{S \to S_1 S \mid \varepsilon\},\ S).$$

再证明所构造文法的正确性, 略. $\square$

CFL 对反转封闭

定理 37

如果 L 是 CFL, 那么 L^R 也是 CFL.

证明:
设 L 的文法 $G = (V, T, P, S)$, 构造文法

$$G' = (V,\ T,\ \{A \to \alpha^R \mid A \to \alpha \in P\},\ S),$$

则 $L(G') = L^R$. 证明略. $\square$

CFL 对逆同态封闭

定理 38

如果 L 是字母表 Δ 上的 CFL, h 是字母表 Σ 到 Δ^* 的同态, 那么 $h^{-1}(L)$ 也是 CFL.

CFL 对逆同态封闭

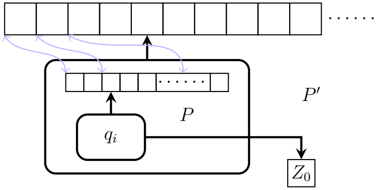
定理 38

如果 L 是字母表 Δ 上的 CFL, h 是字母表 Σ 到 Δ^* 的同态,
那么 $h^{-1}(L)$ 也是 CFL.

证明:
设 PDA $P = (Q, \Delta, \Gamma, \delta, q_0, Z_0, F)$, $\mathbf{L}(P) = L$. 构造 $\mathbf{L}(P') = h^{-1}(L)$ 的 PDA

$$P' = (Q',\ \Sigma,\ \Gamma,\ \delta',\ [q_0, \bar{\varepsilon}],\ Z_0,\ F \times \{\bar{\varepsilon}\}).$$

在 P' 的状态中, 使用缓冲, 暂存字符 $a \in \Sigma$ 的同态串 $h(a)$ 的后缀.



❶ $Q' \subset Q \times \Delta^*$: 状态 $[q, \overline{x}]$ 中的 $\overline{x}$ 为缓冲;

❷ 设 $q \in Q$, 那么 δ' 定义如下:

ⓘ $\forall [q, \overline{\varepsilon}] \in Q \times \{\overline{\varepsilon}\}$, $\forall a \in \Sigma$, $\forall X \in \Gamma$

$$\delta'([q, \overline{\varepsilon}], a, X) = \{([q, h(a)], X)\}$$

ⓘⓘ 若 $\delta(q, \overline{a}, X) = \{(p_1, \beta_1), (p_2, \beta_2), \cdots, (p_k, \beta_k)\}$, 则

$$\delta'([q, \overline{ax}], \varepsilon, X) = \{([p_1, \overline{x}], \beta_1), ([p_2, \overline{x}], \beta_2), \cdots, ([p_k, \overline{x}], \beta_k)\}$$

这里 $\overline{a} \in \Delta \cup \{\overline{\varepsilon}\}$, $\overline{x}$ 是某个 $h(a)$ 的后缀. $\square$

CFL 对交/补运算不封闭

CFL 对交运算不封闭

因为语言

$$L_1 = \{0^n 1^n 2^i \mid n \geq 1, i \geq 1\}$$
$$L_2 = \{0^i 1^n 2^n \mid n \geq 1, i \geq 1\}$$

都是 CFL, 而

$$L_1 \cap L_2 = \{0^n 1^n 2^n \mid n \geq 1\}$$

不是 CFL.

CFL 对补运算不封闭

因为

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}.$$

定理 39

若 L 是 CFL 且 R 是正则语言, 则 $L \cap R$ 是 CFL.

证明: 设 DFA $D = (Q_1,\ \Sigma,\ \delta_1,\ q_1,\ F_1)$ 且 $\mathbf{L}(D) = R$,
PDA $P = (Q_2,\ \Sigma,\ \Gamma,\ \delta_2,\ q_2,\ Z_0,\ F_2)$ 且 $\mathbf{L}(P) = L$, 构造 PDA

$$P' = (Q_1 \times Q_2,\ \Sigma,\ \Gamma,\ \delta,\ [q_1, q_2],\ F_1 \times F_2)$$

其中 δ 为:

定理 39

若 L 是 CFL 且 R 是正则语言, 则 $L \cap R$ 是 CFL.

证明: 设 DFA $D = (Q_1,\ \Sigma,\ \delta_1,\ q_1,\ F_1)$ 且 $\mathbf{L}(D) = R$,
PDA $P = (Q_2,\ \Sigma,\ \Gamma,\ \delta_2,\ q_2,\ Z_0,\ F_2)$ 且 $\mathbf{L}(P) = L$, 构造 PDA

$$P' = (Q_1 \times Q_2,\ \Sigma,\ \Gamma,\ \delta,\ [q_1, q_2],\ F_1 \times F_2)$$

其中 δ 为:

$$\delta([p,q],a,Z) = \begin{cases} \{([p,s],\beta) \mid (s,\beta) \in \delta_2(q,a,Z)\} & a = \varepsilon \\ \{([r,s],\beta) \mid r = \delta_1(p,a) \wedge (s,\beta) \in \delta_2(q,a,Z)\} & a \neq \varepsilon \end{cases}$$

再往证 $\mathbf{L}(P') = L \cap R$, 略. $\square$

封闭性的应用

例 4. 请证明语言 L 不是 CFL

$$L = \{w \in \{a,b,c\}^* \mid \mathrm{n}_a(w) = \mathrm{n}_b(w) = \mathrm{n}_c(w)\},$$

其中 $\mathrm{n}_a(w)$ 表示 w 中 a 的个数.

证明:

❶ 因为 $\mathbf{a}^*\mathbf{b}^*\mathbf{c}^*$ 是正则语言,

❷ 而 $L \cap \mathbf{a}^*\mathbf{b}^*\mathbf{c}^* = \{a^n b^n c^n \mid n \geq 0\}$ 不是 CFL,

❸ 由 CFL 与正则语言的交还是 CFL, 所以 L 不可能是 CFL. □

上下文无关语言的性质

- 上下文无关语言的泵引理
- 上下文无关语言的封闭性
- 上下文无关语言的判定性质
- 乔姆斯基文法体系

上下文无关语言的判定性质

可判定的 CFL 问题

- 空性: 只需判断文法的开始符号 S 是否为非产生的.
- 有穷性和无穷性:
  1. 用不带无用符号的 CNF 的产生式画有向图;
  2. 变元为顶点, 若有 $A \to BC$, 则 A 到 B 和 C 各画一条有向边;
  3. 检查图中是否有循环.
- 成员性: 利用 CNF 范式, 有 CYK 算法检查串 w 是否属于 L.

CYK[1]算法

- CNF $G = (V, T, P, S)$, 以 $O(n^3)$ 时间检查 "$w = a_1 a_2 \cdots a_n \in \mathbf{L}(G)$?"
- 以动态规划方式, 在表中由下至上逐行计算 X_{ij}, 再检查 "$S \in X_{1n}$?"

$$X_{ij} = \{A \in V \mid A \overset{*}{\Rightarrow} a_i a_{i+1} \cdots a_j,\ 1 \leq i \leq j \leq n\},$$

- 计算首行

$$X_{ii} = \{A \mid A \to a_i \in P\}$$

- 计算其他

$$X_{ij} = \left\{ A \;\middle|\; \begin{array}{l} i \leq k < j, \\ BC \in X_{ik} X_{k+1,j}, \\ A \to BC \in P \end{array} \right\}$$

$$\begin{array}{ccccccccc}
 & & & & X_{15} & & & & \\
 & & & X_{14} & & X_{25} & & & \\
 & & X_{13} & & X_{24} & & X_{35} & & \\
 & X_{12} & & X_{23} & & X_{34} & & X_{45} & \\
X_{11} & & X_{22} & & X_{33} & & X_{44} & & X_{55} \\
\hline
a_1 & & a_2 & & a_3 & & a_4 & & a_5
\end{array}$$

[1]J. Cocke, D. Younger, T. Kasami 分别独立发现了算法的基本思想

例 5. CNF G 如下, 用 CYK 算法判断 $bbabaa \in \mathbf{L}(G)$?

$$
\begin{aligned}
S &\to AB \mid BC \\
A &\to BA \mid a \\
B &\to CC \mid b \\
C &\to AB \mid a
\end{aligned}
$$

例 5. CNF G 如下, 用 CYK 算法判断 $bbabaa \in \mathbf{L}(G)$?

$$
\begin{aligned}
S &\to AB \mid BC \\
A &\to BA \mid a \\
B &\to CC \mid b \\
C &\to AB \mid a
\end{aligned}
$$

$\{S, A\}$					
$\{B\}$	$\{S, A\}$				
$\{S, C\}$	$\{B\}$	$\{S, A\}$			
$\{A\}$	$\{S, C\}$	$\{B\}$	$\{\}$		
$\{\}$	$\{S, A\}$	$\{S, C\}$	$\{S, A\}$	$\{B\}$	
$\{B\}$	$\{B\}$	$\{A, C\}$	$\{B\}$	$\{A, C\}$	$\{A, C\}$
b	b	a	b	a	a

因为 $S \in X_{16} = \{S, A\}$, 所以 $bbabaa \in \mathbf{L}(G)$.

上下文无关语言的判定性质

不可判定的 CFL 问题

1. 判断 CFG G 是否歧义的?
2. 判断 CFL 是否固有歧义的?
3. 两个 CFL 的交是否为空?
4. 两个 CFL 是否相同?
5. 判断 CFL 的补是否为空? 尽管有算法判断 CFL 是否为空
6. 判断 CFL 是否等于 Σ^*?

上下文无关语言的性质

- 上下文无关语言的泵引理
- 上下文无关语言的封闭性
- 上下文无关语言的判定性质
- 乔姆斯基文法体系

乔姆斯基文法体系

如果文法 $G=(V,T,P,S)$, 符号串 $\alpha \in (V\cup T)^*V(V\cup T)^*$, $\beta \in (V\cup T)^*$, 产生式都形如

$$\alpha \to \beta$$

即每个产生式的左部 α 中至少要有一个变元, 那么:

1. 称 G 为 0 型文法或短语结构文法;
2. 若 $|\beta| \geq |\alpha|$, 称 G 为 1 型文法或上下文有关文法;
3. 若 $\alpha \in V$, 称 G 为 2 型文法或上下文无关文法;
4. 若 $\alpha \to \beta$ 都形如 $A \to aB$ 或 $A \to a$, 称 G 为 3 型文法或正则文法.

形式语言与自动机理论

图灵机

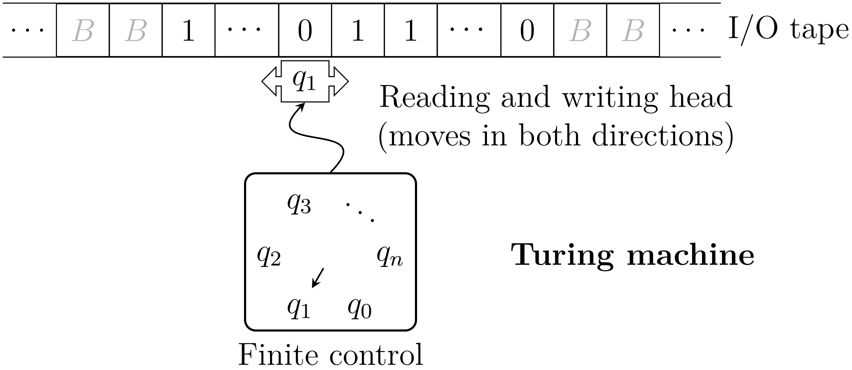
王春宇

计算机科学与技术学院
哈尔滨工业大学

图灵机

- 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

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

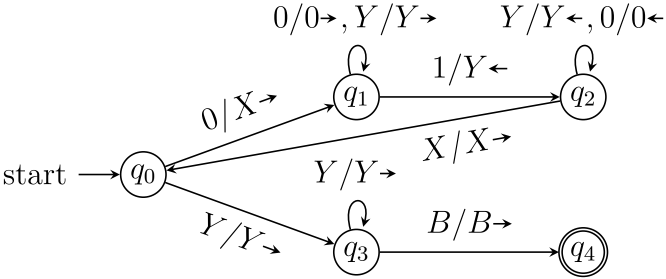
表示状态转移到 p, 单元格改为 Y, 然后带头向左移动一个单元格.

$$\begin{tikzcd} q \arrow[rr, "X/Y\leftarrow"] & & p \end{tikzcd}$$

因为每个动作都是确定的, 因此是“确定的图灵机”.

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

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



续例 1. 设计识别 $\{0^n1^n \mid n \geq 1\}$ 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	$-$	$-$	$-$	$-$	$-$

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. 为避免混淆，一般假定 Q 和 Γ 不相交.

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 1. 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

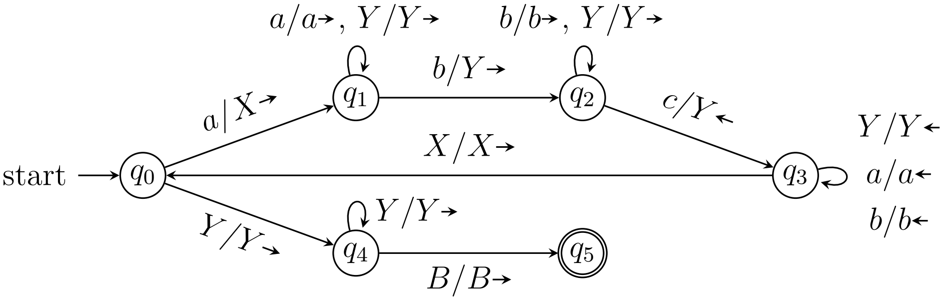
续例 1. 设计识别 $\{0^n1^n \mid n \geq 1\}$ 的图灵机.

接受 0011 的 ID 序列 (M 的一个计算) 为

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

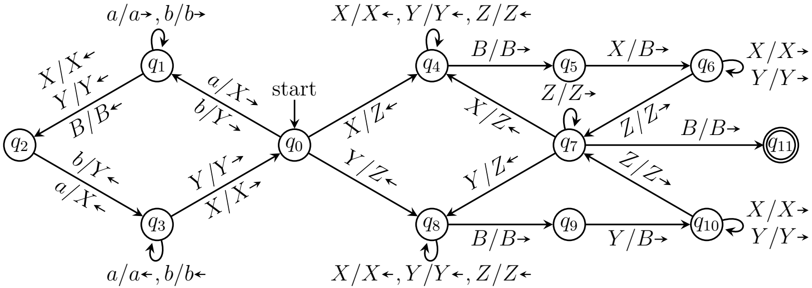
例 2. 设计接受 $L = \{a^n b^n c^n \mid n \geq 1\}$ 的图灵机.

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.

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



思考

❶ DFA 和 TM 的主要区别?

❷ 计算机, 究竟是 TM 还是 DFA?

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

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

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.

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 $i_1, i_2, \ldots, i_k$ 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 $f(i_1, i_2, \ldots, i_k) = m$.
- M 计算的 f，不必对所有不同的 $i_1, i_2, \ldots, i_k$ 都有值.

定义

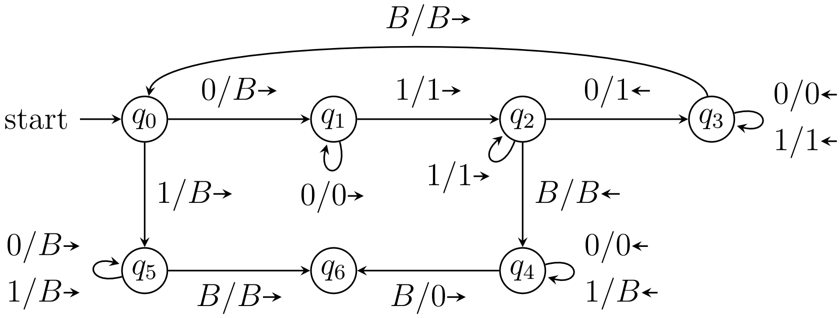
如果 $f(i_1, i_2, \ldots, i_k)$ 对所有$i_1, i_2, \ldots, i_k$ 都有定义，称 f 为全递归函数.
被图灵机计算的函数 $f(i_1, i_2, \ldots, i_k)$ 称作部分递归函数.

例 4. 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 4. 给出计算整数真减法 ($\dot{-}$) 的图灵机, 其定义为

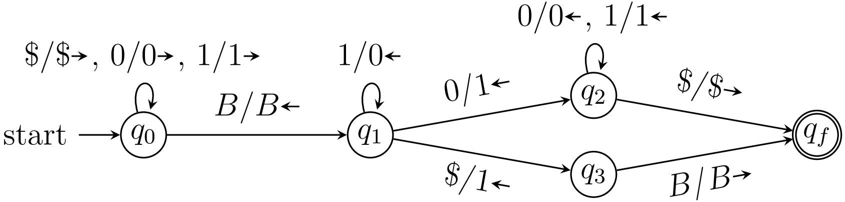
$$m \dot{-} n = \begin{cases} m - n & m \geq n \\ 0 & m < n \end{cases}.$$



例 5. 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 $q_0\$10011 \vdash^* \$q_f10100$, $q_0\$111 \vdash^* q_f1000$.

例 5. 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 $q_0\$10011 \vdash^* \$q_f10100$, $q_0\$111 \vdash^* q_f1000$.



图灵机

- 图灵机
- 图灵机的变形
  - 扩展的图灵机
  - 受限的图灵机

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

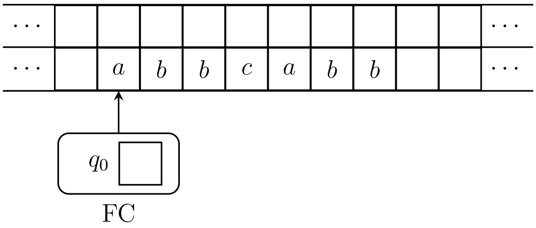
多道

多道图灵机:

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

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

例 6. 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 7. 设计 TM 实现全递归函数 “乘法”.

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.

证明方法:

❶ 用 $2k$ 道的单带图灵机 N 模拟 k 带图灵机 M;

❷ N 用两道模拟 M 一带, 一道放置内容, 另一道标记带头;

❸ 模拟 M 的一个动作, N 需要从左至右, 再从右至左扫描一次;

❹ 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 41

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.

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 42

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 43

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

形式语言与自动机理论

不可判定性

王春宇

计算机科学与技术学院
哈尔滨工业大学

不可判定性

- 不可判定性
- 非递归可枚举的语言
- 递归可枚举但非递归的语言
- 语言类的关系

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, 则称为不可判定的.

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, 则称为不可判定的.

是否存在不可判定的问题?

1. $\{L \mid L \in \Sigma^*\}$ 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, 必然存在不可判定的问题.

hello-world 问题

判断带有给定输入的任意给定的程序, 是否以 `hello, world` 为其输出的前 12 个字符.

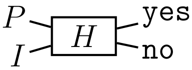
定理

hello-world 问题是不可判定的.

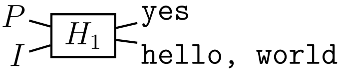
- "具有这个输入的这个程序是否显示 `hello, world`?"
- 解决这样问题的通用程序是不存在的.

(非形式) 证明: 反证法. 假设这样的程序 H 存在.

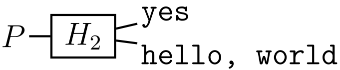
❶ H 检查程序 P 在输入 I 时的输出, 并回答 `yes` 或 `no`:



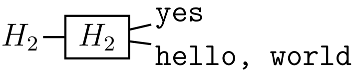
❷ 修改 H, 在回答 `no` 时, 输出 `hello, world`:



❸ 再修改 H_1, 将程序 P 作为 P 的输入:



❹ 那么, 当程序 H_2 以 H_2 为输入时:



❺ H_2 的输出会出现矛盾 (悖论), 所以 H 不可能存在. $\square$

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

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，就称为可判定问题，否则称为不可判定问题.

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

不可判定性

- 不可判定性
- 非递归可枚举的语言
  - 第 i 个串
  - 图灵机编码与第 i 个图灵机
  - 对角化语言 L_d
- 递归可枚举但非递归的语言
- 语言类的关系

可判定吗?

“图灵机 M 接受输入 w 吗? ”

第 i 个串 w_i

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	$\cdots$

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

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

❶ $Q = \{q_1, q_2, \cdots, q_{|Q|}\}$，开始状态为 q_1，终态为 q_2 且停机；

❷ $\Gamma = \{X_1, X_2, \cdots, X_{|\Gamma|}\}$，总有 $X_1 = 0,\ X_2 = 1,\ X_3 = B$；

❸ 设带头移动方向 $D_1 = L,\ D_2 = R$；

❹ 任意的转移 $\delta(q_i, X_j) = (q_k, X_l, D_m)$ 编码为

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

❺ 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

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

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

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

定理 44

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. $\square$

不可判定性

- 不可判定性
- 非递归可枚举的语言
- 递归可枚举但非递归的语言
  - 递归语言的封闭性
  - 通用语言与通用图灵机
- 语言类的关系

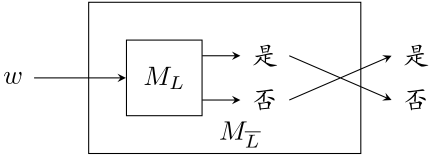
递归语言的封闭性

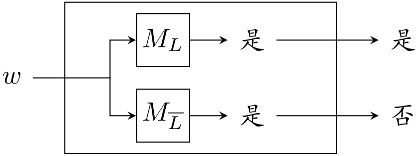
定理 45

如果 L 是递归的，那么 $\overline{L}$ 也是递归的.

定理 46

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





通用语言与通用图灵机

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

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

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.

递归可枚举但非递归的语言

定理 47

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 $M_i 111 w_i$ 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. □

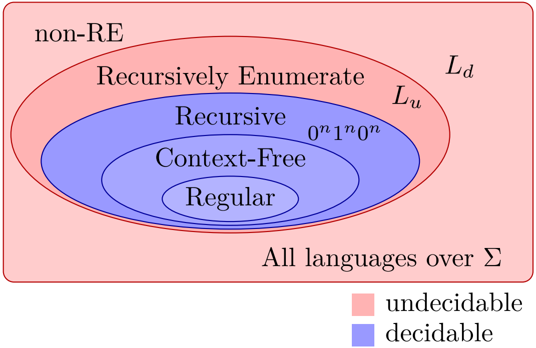
通用图灵机的重要意义

- 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

不可判定性

- 不可判定性
- 非递归可枚举的语言
- 递归可枚举但非递归的语言
- 语言类的关系

语言类的关系


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