Chapter 6

下推自动机

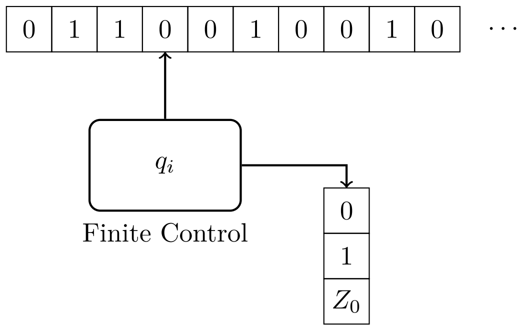
6.1 介绍

下推自动机是可以看做带有堆栈的 ε-NFA. 工作方式类似 ε-NFA, 有一个有穷控制器, 并能够以非确定的方式进行状态转移, 并读入输入字符; 增加的堆栈, 用来存储无限的信息, 但只能以后进先出的方式使用.

$$\varepsilon\text{-NFA} + 栈 = \text{PDA}$$

ε-NFA: 有限状态, 非确定, ε 转移

栈: 后进先出, 只用栈顶, 长度无限



6.2 下推自动机的定义

6.2.1 形式定义

下推自动机 (*Pushdown Automata*, PDA) P 的形式定义, 为七元组 $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$:

(1) Q, 有穷状态集;

(2) Σ, 有穷输入字母表;

(3) Γ, 有穷栈字母表, 或栈符号集;

(4) $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \mapsto 2^{Q\times\Gamma^*}$, 状态转移函数;

(5) $q_0 \in Q$, 初始状态;

(6) $Z_0 \in \Gamma - \Sigma$, 初始符号, PDA 开始时, 栈中包含这个符号的一个实例, 用来表示栈底, 栈底符号之下无任何内容;

(7) $F \subseteq Q$, 接收状态集或终态集.

PDA 的动作

如果 q 和 p_i 是状态 $(1 \le i \le m)$, 输入符号 $a \in \Sigma$, 栈符号 $Z \in \Gamma$, 栈符号串 $\beta_i \in \Gamma^*$, 那么映射

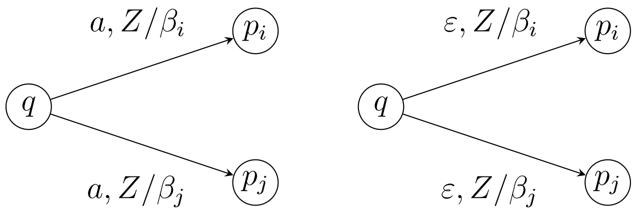
$$\delta(q, a, Z) = \{(p_1, \beta_1), (p_2, \beta_2), \cdots, (p_m, \beta_m)\}$$

的解释是: 输入符号是 a, 栈顶符号 Z 的情况下, 处于状态 q 的 PDA 能够进入状态 p_i, 且用符号串 β_i 替换栈顶的符号 Z, 这里的 i 是任意的, 然后输入头前进一个符号. (约定 β_i 的最左符号在栈最上.) 但是若 $i \neq j$, 不能同时选择 p_i 和 β_j. 而

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

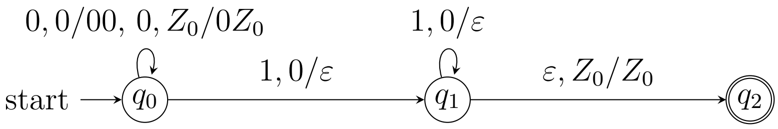
的解释是: 与扫描的输入符号无关, 只要 Z 是栈符号, 处于状态 q 的 PDA, 就可以进行上面的动作, 但输入头不向前移动.

PDA 的图形表示



示例

设计识别 $L_{0n1n} = \{0^n1^n \mid n \geq 1\}$ 的 PDA P.



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

(1) 初始状态 (q_0, Z_0) 输入 0 或 1, 状态不变, 则直接压栈:

$$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\},\ \delta(q_0, 1, Z_0) = \{(q_0, 1Z_0)\};$$

(2) 继续输入, 则对不同的栈顶, 仍然压栈:

$$\delta(q_0, 0, 0) = \{(q_0, 00)\},\ \delta(q_0, 1, 0) = \{(q_0, 10)\},$$
$$\delta(q_0, 0, 1) = \{(q_0, 01)\},\ \delta(q_0, 1, 1) = \{(q_0, 11)\};$$

(3) 不论栈顶是 Z_0, 0, 或 1, 开始匹配后半部分, 非确定的转移到弹栈状态:

$$\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\},\ \delta(q_0, \varepsilon, 0) = \{(q_1, 0)\},\ \delta(q_0, \varepsilon, 1) = \{(q_1, 1)\};$$

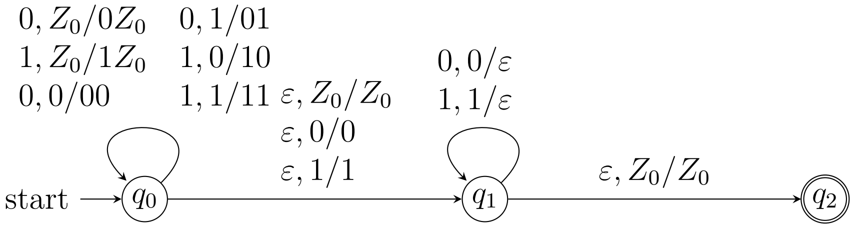
(4) 处于弹栈状态, 弹出的符号必须和输入符号一致:

$$\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\},\ \delta(q_1, 1, 1) = \{(q_1, \varepsilon)\};$$

(5) 只有看到栈底符号了, 才允许非确定的转移到接受状态:

$$\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}.$$

状态转移图



6.2.2 瞬时描述和转移符号

瞬时描述 为了形式描述 PDA 在一个给定瞬间的格局, 定义瞬时描述 (*Instantaneous Description*, ID) 为三元组 (q, w, γ), 是 $Q \times \Sigma^* \times \Gamma^*$ 中的元素, q 表示状态, w 表示剩余的输入串, γ 表示栈中的符号串.

ID 转移符号 $\underset{P}{\vdash}$ 和 $\overset{*}{\underset{P}{\vdash}}$ 在 PDA P 中，如果 $(p, \beta) \in \delta(q, a, Z)$, 那么, 定义 ID 转移符号 $\underset{P}{\vdash}$ 为

$$(q, aw, Z\alpha) \underset{P}{\vdash} (p, w, \beta\alpha)$$

其中 $w \in \Sigma^*$, $\alpha \in \Gamma^*$. 并递归的定义 $\overset{*}{\underset{P}{\vdash}}$ 为

(1) 对每个 ID I, 有 $I \overset{*}{\underset{P}{\vdash}} I$;

(2) 对 ID I, J 和 K, 若 $I \underset{P}{\vdash} J$, $J \overset{*}{\underset{P}{\vdash}} K$, 则 $I \overset{*}{\underset{P}{\vdash}} K$.

若 P 已知, 则可以省略, 记为 $\vdash$ 和 $\overset{*}{\vdash}$.

定理 1. 如果 $(q, x, \alpha) \overset{*}{\underset{P}{\vdash}} (p, y, \beta)$, 则任意 $w \in \Sigma^*$ 和任意 $\gamma \in \Gamma^*$, 有

$$(q, xw, \alpha\gamma) \overset{*}{\underset{P}{\vdash}} (p, yw, \beta\gamma).$$

定理 2. 如果 $(q, xw, \alpha) \overset{*}{\underset{P}{\vdash}} (p, yw, \beta)$, 则 $(q, x, \alpha) \overset{*}{\underset{P}{\vdash}} (p, y, \beta)$.

6.3 PDA 接受的语言

设 PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, 则分别定义两种接受方式下的语言如下.

以终态方式接受

P 以终态方式接受的语言 $\mathbf{L}(P)$ 是

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

以空栈方式接受

P 以空栈方式接受的语言 $\mathbf{N}(P)$ 是

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

示例

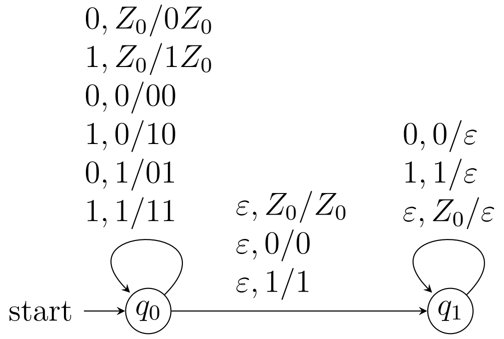
识别 L_{wwr} 的 PDA P, 从终态方式接受, 改为空栈方式接受, 只需用

$$\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$$

代替

$$\delta(q_1, \varepsilon, Z_0) = \{(q_2, Z_0)\}$$

即可.

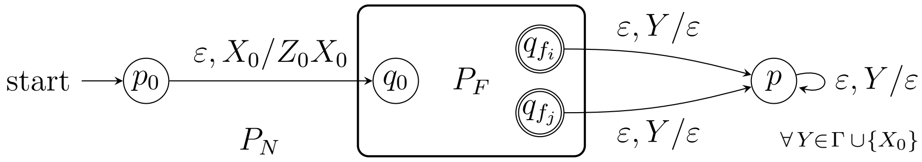


6.3.1 从终态方式到空栈方式

定理 3. 如果以终态方式接受的 PDA P_F 接受的语言 $L = \mathbf{L}(P_F)$, 那么一定存在以空栈方式接受的 PDA P_N 使 $L = \mathbf{N}(P_N)$.

证明. 设 $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. 构造 P_N, 增加新的初始状态 p_0 和新的状态 p, 使用新的栈底符号 X_0, 并定义新的转移函数 δ_N, 即

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



其中 δ_N 定义如下:

(1) P_N 开始时, 将 P_F 栈底符号压入栈, 并准备开始模拟 P_F:

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

(2) P_N 模拟 P_F, 即 $\forall q \in Q,\ \forall a \in \Sigma \cup \{\varepsilon\},\ \forall Y \in \Gamma$:

$$\delta_N(q, a, Y) \text{ 包含 } \delta_F(q, a, Y) \text{ 的诸元素}$$

(3) 从 $q_f \in F$ 开始弹出栈符号, 即 $\forall q_f \in F, \forall Y \in \Gamma \cup \{X_0\}$:

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

(4) 在状态 p 时, 弹出全部栈符号, 即 $\forall Y \in \Gamma \cup \{X_0\}$:

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

需要证明 $w \in \mathbf{L}(P_F) \Leftrightarrow w \in \mathbf{N}(P_N)$.

$(\Rightarrow)$ 如果 $w \in \mathbf{L}(P_F)$, 则有到 $q_f \in F$ 的 ID 序列

$$(q_0, w, Z_0) \overset{*}{\vdash}_{P_N} (q_f, \varepsilon, \gamma)$$

这些动作也是 P_F 的合法动作, 因此

$$(q_0, w, Z_0) \overset{*}{\vdash}_{P_F} (q_f, \varepsilon, \gamma)$$

又因为, 栈底之下增加符号不会影响这些动作, 因此

$$(q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_F} (q_f, \varepsilon, \gamma X_0)$$

以及 P_N 在开始状态的空转移

$$(p_0, w, X_0) \vdash_{P_N} (q_0, w, Z_0X_0)$$

和 $q_f \in F$ 时, 会清空栈

$$(q_f, \varepsilon, \gamma X_0) \overset{*}{\vdash}_{P_N} (p, \varepsilon, \varepsilon)$$

所以

$$(p_0, w, X_0) \vdash_{P_N} (q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_N} (q_f, \varepsilon, \gamma X_0) \overset{*}{\vdash}_{P_N} (p, \varepsilon, \varepsilon)$$

因此 $w \in \mathbf{N}(P_N)$. $(\Leftarrow)$ 反之, 类似, 略.

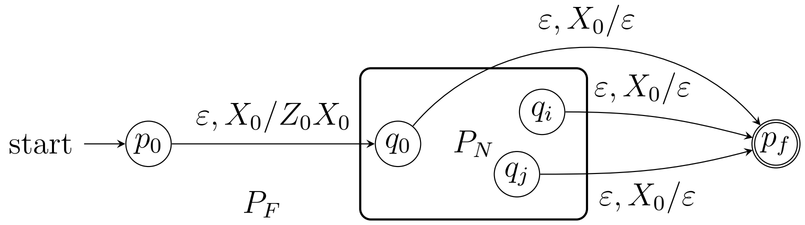
□

6.3.2 从空栈方式到终态方式

定理 4. *如果以空栈方式接受的 PDA P_N 接受的语言 $L = \mathbf{N}(P_N)$, 那么一定存在以终态方式接受的 PDA P_F 使 $L = \mathbf{L}(P_F)$.*

证明. 设 $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0, \emptyset)$. 构造 P_F, 增加新的初始状态 p_0 和新的终态 p_f, 使用新的栈底符号 X_0, 并定义新的转移函数 δ_F, 即

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



其中 δ_F 定义如下:

(1) P_F 开始时, 将 P_N 栈底符号压入栈, 并开始模拟 P_N,

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

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

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

(3) 在任何 $q \in Q$ 时, 看到 P_F 的栈底 X_0, 就可以转移到新终态 p_f:

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

需要证明 $w \in \mathbf{N}(P_N) \Leftrightarrow w \in \mathbf{L}(P_F)$.

$(\Rightarrow)$

$$(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_N} (q, \varepsilon, X_0) \vdash_{P_F} (p_f, \varepsilon, \varepsilon)$$

$(\Leftarrow)$ 如果 $w \in \mathbf{L}(P_F)$, 由于 (3) 接受 w 的时栈符号只能是 ε, 即 ID 是 $(p_f, \varepsilon, \varepsilon)$; 那么倒数第二个 ID 只能是 (q, ε, X_0); 而因为 (1) 开始时的 ID 只能由 (p_0, w, X_0) 得到 (q_0, w, Z_0X_0); 所以有

$$(p_0, w, X_0) \vdash_{P_F} (q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_F} (q, \varepsilon, X_0) \vdash_{P_F} (p_f, \varepsilon, \varepsilon)$$

而其中 $(q_0, w, Z_0X_0) \overset{*}{\vdash}_{P_F} (q, \varepsilon, X_0)$, 因为 (2) 是 P_F 模拟 P_N 所以与 X_0 无关, 因此

$$(q_0, w, Z_0) \overset{*}{\vdash}_{P_N} (q, \varepsilon, \varepsilon)$$

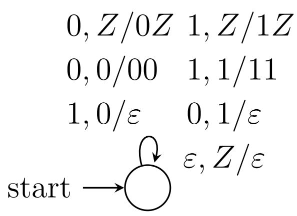
是 P_N 的合法 ID, 因此 $w \in \mathbf{N}(P_N)$.

□

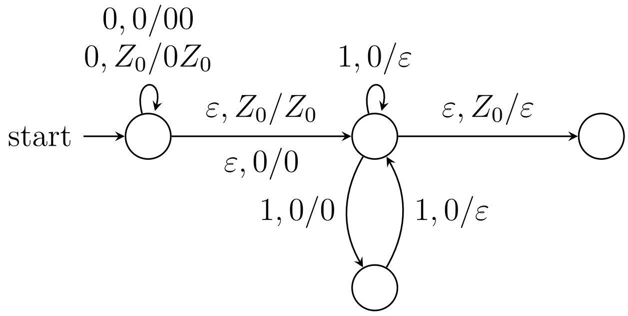
示例

所有 0 和 1 个数相同的 0 和 1 的串的集合.

以空栈方式接受:



接受 $\{0^n1^m|n \le m \le 2n\}$ 的 PDA.



6.4 PDA 与 CFG 的等价性

6.4.1 由 CFG 到 PDA

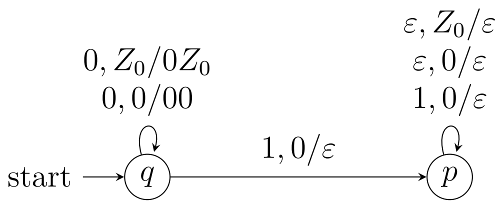
示例

识别 $L = \{0^n1^m \mid 1 \le m \le n\}$ 的 CFG 和 PDA 有

CFG $G, L = \mathbf{L}(G)$:

$$S \to AB \qquad A \to 0A \mid \varepsilon \qquad B \to 0B1 \mid 01$$

PDA $P, L = \mathbf{N}(P)$:



CFG G 有关串 00011 的文法派生过程如下:

$$S \Rightarrow AB \Rightarrow 0AB \Rightarrow 0B \Rightarrow 00B1 \Rightarrow 00011$$

PDA P 识别该串的 ID 转移序列如下:

$$\begin{aligned}(q, 00011, Z_0) &\vdash (q, 0011, 0Z_0) \vdash (q, 011, 00Z_0) \vdash (q, 11, 000Z_0)\\ &\vdash (p, 1, 00Z_0) \vdash (p, \varepsilon, 0Z_0) \vdash (p, \varepsilon, Z_0) \vdash (p, \varepsilon, \varepsilon)\end{aligned}$$

想要证明 CFG 和 PDA 的等价性, 需要思考如何使用 PDA 模拟文法的推导. 对任意属于某 CFL 的串 w, 其文法的推导过程，就是使用产生式去匹配 (产生)w, 如果 w 放在某 PDA 的输入带上, 我们的目的就是通过文法构造动作, 让 PDA 能从左到右的扫描输入串, 利用栈来模拟文法的派生过程即可.

定理 5. 如果 L 是上下文无关语言, 那么存在 PDA P, 使 $L = \mathbf{N}(P)$.

证明. **构造 PDA**: 设 CFG $G = (V, T, P', S)$ 且 $L(G) = L$, 构造 PDA

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

其中 δ 定义:

(1) 对每个变元 A:

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

(2) 对每个终结符 a:

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

那么 P 可以模拟 G 的最左派生, 每个动作只根据栈顶的符号: 如果是终结符则与输入串匹配, 如果是非终结符用产生式来替换.

充分性: 要证明 $S \overset{*}{\Rightarrow} w \implies (q, w, S) \vdash^* (q, \varepsilon, \varepsilon)$. 那么任意 $w \in \mathbf{L}(G)$, 则存在最左派生 $S \underset{\text{lm}}{\overset{*}{\Rightarrow}} w$, 并且除最后一步派生的 w 外, 每次派生的最左句型都有 $xA\alpha$ 的形式, 这里 $x \in T^*$, $A \in V$, $\alpha \in (V \cup T)^*$.

$$\begin{array}{ccccccccc}
S = x_1A_1\alpha_1 & \underset{\text{lm}}{\Rightarrow} & x_2A_2\alpha_2 & \underset{\text{lm}}{\Rightarrow} & \cdots & \underset{\text{lm}}{\Rightarrow} & x_{n-1}A_{n-1}\alpha_{n-1} & \underset{\text{lm}}{\Rightarrow} & x_n\alpha_n = w \\
w = x_1y_1 & = & x_2y_2 & = & \cdots & = & x_{n-1}y_{n-1} & = & x_ny_n = w \\
(q, w, S) = (q, y_1, A_1\alpha_1) & \vdash^* & (q, y_2, A_2\alpha_2) & \vdash^* & \cdots & \vdash^* & (q, y_{n-1}, A_{n-1}\alpha_{n-1}) & \vdash^* & (q, y_n, \alpha_n) \vdash^* (q, \varepsilon, \varepsilon)
\end{array}$$

当处于 $S \underset{\text{lm}}{\overset{*}{\Rightarrow}} w$ 的第 i 步时, 若 $x_iy_i = w$, 有:

$$x_iA_i\alpha_i \underset{\text{lm}}{\overset{*}{\Rightarrow}} x_{i+1}A_{i+1}\alpha_{i+1} \implies (q, y_i, A_i\alpha_i) \vdash^* (q, y_{i+1}, A_{i+1}\alpha_{i+1})$$

因为, 当最左派生处于左句型 $x_iA_i\alpha_i$ 时, ID 为 $(q, y_i, A_i\alpha_i)$, 如果有 $A_i \to \beta$ 的产生式, 则

$$x_iA_i\alpha_i \underset{\text{lm}}{\Rightarrow} x_i\beta\alpha_i$$

而 $x_i\beta\alpha_i$ 也是左句型, 所以最左的变量即为 A_{i+1}, 则 A_{i+1} 之前 x_i 之后的终结符记为 x', A_{i+1} 之后的记为 α_{i+1}, 那么就有

$$x_iA_i\alpha_i \underset{\text{lm}}{\Rightarrow} x_i\beta\alpha_i = x_ix'A_{i+1}\alpha_{i+1}$$

那么由 (1) P 模拟 $A_i \to \beta$ 的动作得到

$$(q, y_i, A_i\alpha_i) \vdash (q, y_i, \beta\alpha_i) = (q, y_i, x'A_{i+1}\alpha_{i+1})$$

而 $w = x_iy_i = x_ix'y_{i+1}$, 所以 $y_i = x'y_{i+1}$, 由 (2) P 会弹出 x', 那么

$$(q, y_i, x'A_{i+1}\alpha_{i+1}) = (q, x'y_{i+1}, x'A_{i+1}\alpha_{i+1}) \vdash^* (q, y_{i+1}, A_{i+1}\alpha_{i+1})$$

因此当 $S \underset{\text{lm}}{\overset{*}{\Rightarrow}} w$ 有 $(q, w, S) \vdash^* (q, \varepsilon, \varepsilon)$, 即 $\mathbf{L}(G) \subseteq \mathbf{N}(P)$.

必要性: 要证明 $(q, w, S) \vdash^* (q, \varepsilon, \varepsilon) \implies S \overset{*}{\Rightarrow} w$. 我们证明更一般的结论

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

通过对 ID 转移的次数进行归纳证明. *归纳基础*: 当仅需要 1 次时, 只能是 $x = \varepsilon$ 且 $A \to \varepsilon$ 为产生式. (因为即使 $x = a$ 和产生式 $A \to a$, 也需要 2 步才能清空栈: 替换栈顶 A 为 a, 再弹出 a.) 所以 $A \overset{*}{\Rightarrow} \varepsilon$ 成立.

归纳递推: 假设转移次数不大于 $n\,(n \geq 0)$ 步时结论成立. 当需要 $n+1$ 步时, 因为 A 是变元, 其第 1 步转移一定是 $(q, x, A) \vdash (q, x, Y_1Y_2\cdots Y_m)$ 且 $A \to Y_1Y_2\cdots Y_m$ 是产生式, 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 不论是变元或终结符, 从栈中被完全弹出时, 都会消耗掉部分的 x, 记为 x_i, 那么显然有 $x = x_1x_2\cdots x_m$. 而且为了清空每个 Y_i, 需要的转移次数都不超过 n, 所以对 $i = 1, 2, \cdots, m$ 有

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

再由 A 的产生式有

$$A \underset{\text{lm}}{\Rightarrow} Y_1Y_2\cdots Y_m \underset{\text{lm}}{\overset{*}{\Rightarrow}} x_1Y_2\cdots Y_m \underset{\text{lm}}{\overset{*}{\Rightarrow}} x_1x_2\cdots Y_m \underset{\text{lm}}{\overset{*}{\Rightarrow}} x_1x_2\cdots x_m = x.$$

因此 $(q, w, S) \overset{*}{\vdash} (q, \varepsilon, \varepsilon) \Longrightarrow S \overset{*}{\Rightarrow} w$, 即 $\mathbf{N}(G) \subseteq \mathbf{L}(G)$. □

示例

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

构造 PDA $P = (\{q\}, \{a, b\}, \{a, b, A, S\}, \delta, q, S, \emptyset)$, 其中 δ 为

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

$\delta(q, \varepsilon, A) = \{(q, aS), (q, bS), (q, a)\}$ $\qquad\delta(q, b, b) = \{(q, \varepsilon)\}$.

利用 GNF 的构造方法

将文法转换为 GNF 格式的 CFG $G = (V, T, P', S)$, 那么 PDA P 的另一种构造方式为:

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

为每个产生式, 定义 δ:

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

示例

上例中的文法是 GNF, 构造 PDA $P' = (\{q\}, \{a, b\}, \{S, A\}, \delta, q, S, \emptyset)$, 其中 δ 为

$\delta(q, a, S) = \{(q, AA)\}$

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

$\delta(q, b, A) = \{(q, S)\}$.

6.4.2 由 PDA 到 CFG

定理 6. *如果 PDA P, 有 $L = \mathbf{N}(P)$, 那么 L 是上下文无关语言.*

证明. **文法构造**: 设 PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \emptyset)$. 那么构造 CFG $G = (V, T, P, S)$, 其中 V 是形如 $[qXp]$ 的对象和符号 S 的集合, 其中 $p, q \in Q$, $X \in \Gamma$, 产生式集合 P 包括:

(1) 为 Q 中的每个 p, 构造一个产生式:

$$S \to [q_0Z_0p]$$

(2) 如果 $\delta(q, a, X)$ 包括 $(p, Y_1Y_2\cdots Y_n)$, 构造一组产生式:

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

这里 $a \in \Sigma \cup \{\varepsilon\}$; $X, Y_i \in \Gamma$; $p, q \in Q$; 而 $r_1, r_2, \cdots r_n$ 是 Q 中各种可能的 n 个状态; 若 $i = 0$ 则构造产生式 $[qXp] \to a$.

那么

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

充分性: ID 序列 $(q,w,X)\overset{*}{\vdash}(p,\varepsilon,\varepsilon)$ 表示栈弹出 X 而消耗了串 w (状态从 q 到了 p); 而 $[qXp]\overset{*}{\Rightarrow} w$ 表示在 P 中经过栈符号 X 可以产生出 w (状态从 q 到了 p); 设 $w=ax$, $a\in\Sigma\cup\{\varepsilon\}$.

(1) 我们要证明

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

(2) 左边部分 ID 变化如果需多步完成, 那么第 1 步时, 一定有 $\delta(q,a,X)$ 包含 $(p,Y_1Y_2\cdots Y_n)$,

(i) 则第 1 步为

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

而其余步骤中, 为弹出 Y_i 会消耗 x 中的一部分 x_i, 显然 $w=ax=ax_1x_2\cdots x_n$;

(ii) 设弹出 Y_i 之前和之后 (弹出 Y_{i+1} 之前) 的状态分别是 r_{i-1} 和 r_i, 消耗的串是 x_i, 这里 $i=1,2,\cdots n$ 且 $r_0=p$, 那么其他步骤就是

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

(iii) 而根据文法的构造规则, 有

$$[qXp]\Rightarrow a[pY_1r_1][r_1Y_2r_2]\cdots[r_{n-1}Y_nr_n]$$

(iv) 因此, 只要 $i=1,2,\cdots,n$ 有

$$(r_{i-1},x_i,Y_i)\overset{*}{\vdash}(r_i,\varepsilon,\varepsilon) \implies [r_{i-1}Y_ir_i]\overset{*}{\Rightarrow} x_i$$

成立, 就有

$$[qXp]\Rightarrow a[pY_1r_1][r_1Y_2r_2]\cdots[r_{n-1}Y_nr_n]\overset{*}{\Rightarrow} ax_1x_2\cdots x_n=w$$

成立.

(3) 而左边部分 ID 动作变化如果仅需 1 步完成 (或者说 i=0 时), 由于 $(q,a,X)\vdash(p,\varepsilon,\varepsilon)$, P 只能消耗不超过一个的字符, 即 $w=a$ ($a\in\Sigma\cup\{\varepsilon\}$), 且 (p,ε) 在 $\delta(q,a,X)$, 所以, 由文法构造规则 $[qXp]\to a$, 即

$$(q,a,X)\vdash(p,\varepsilon,\varepsilon)\Longrightarrow[qXp]\Rightarrow a$$

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

必要性: 略. □

示例

将 PDA $P=(\{p,q\},(0,1),\{X,Z\},\delta,q,Z)$ 转为 CFG, 其中 δ 如下:

(1) $\delta(q,1,Z)=\{(q,XZ)\}$	(4) $\delta(q,\varepsilon,Z)=\{(q,\varepsilon)\}$
(2) $\delta(q,1,X)=\{(q,XX)\}$	(5) $\delta(p,1,X)=\{(p,\varepsilon)\}$
(3) $\delta(q,0,X)=\{(p,X)\}$	(6) $\delta(p,0,Z)=\{(q,Z)\}$

0	$S\to[qZq]$	
	$S\to[qZp]$	消掉 $[qZp]$, 因与自己循环
1	$[qZq]\to 1[qXq][qZq]$	
	$[qZq]\to 1[qXp][pZq]$	
	$[qZp]\to 1[qXq][qZp]$	...
	$[qZp]\to 1[qXp][pZp]$	因生成 step 2 中的 $[qZp]$
...	...	

6.5 确定型下推自动机 (DPDA)

PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ 是确定型下推自动机 (*DPDA*), 当且仅当:

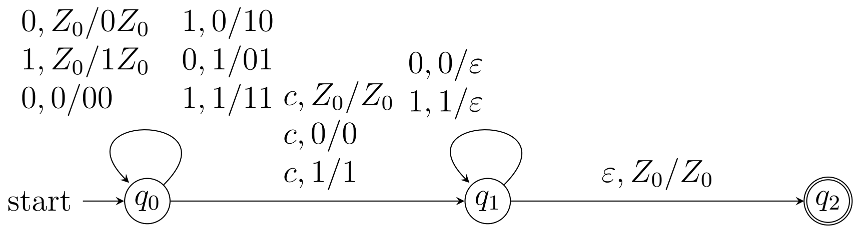
(1) $\delta(q, a, X)$ 至多有一个动作, 这里 $a \in \Sigma \cup \{\varepsilon\}$;

(2) $\forall a \in \Sigma$, 如果 $\delta(q, a, X) \neq \emptyset$, 那么 $\delta(q, \varepsilon, X) = \emptyset$.

即 $\forall (q, a, Z) \in Q \times \Sigma \times \Gamma$, $|\delta(q, a, Z)| + |\delta(q, \varepsilon, Z)| \leq 1$. 下面给出关于这种装置的一些事实.

在任何情况下都不需要去选择可能的移动就是 DPDA, 以终态方式接受的语言也称为 DCFL. 虽然与 PDA 不等价, 但也有意义, 例如语法分析器通常都是 DPDA, DPDA 接受的语言是非固有歧义语言的真子集, Knuth 提出 LR(k) 文法的语言也恰好是 DPDA 接受语言的一个子集, 解析的时间复杂度为 $O(n)$, LR(k) 文法也是 YACC 的基础.

任何 DPDA 都无法接受 L_{wwr}, 但是可以接受 $L_{wcwr} = \{wcw^R \mid w \in (0+1)^*\}$.



6.5.1 RL 与 DPDA

定理 7. *如果语言 L 是正则的, 那么有 DPDA P 以终态方式接受 L, 即 $L = \mathbf{L}(P)$.*

DPDA P 可以不使用栈, 而仅模拟 DFA 即可. 又因为 L_{wcwr} 显然是 CFL 不是正则语言, 所以 $\mathbf{L}(P)$ 语言类真包含正则语言.

定理 8. *DPDA P 且 $L = \mathbf{N}(P)$, 当且仅当 L 有前缀性质, 且存在 DPDA P' 使 $L = \mathbf{L}(P')$.*

DPDA P 若以空栈方式接受, 能够接受的语言更有限, 仅能接受具有前缀性质的语言. 前缀性质是指, 这个语言中不存在不同的串 x 和 y 使 x 是 y 的前缀. 即使正则语言 0^* 也无法接受, 因为其任何两个串中都有一个是前缀. 但以空栈方式接受的语言, 却可以被另一个 DPDA 以终态方式接受.

6.5.2 DPDA 与 CFL

DPDA P 无法识别 L_{wwr}. 所以 $\mathbf{L}(P)$ 语言类真包含于上下文无关语言.

6.5.3 DPDA 与歧义文法

定理 9. *如果有 DPDA P, 语言 $L = \mathbf{L}(P)$, 那么 L 有无歧义的 CFG.*

定理 10. *如果有 DPDA P, 语言 $L = \mathbf{N}(P)$, 那么 L 有无歧义的 CFG.*

证明略. DPDA 也因此在语法分析中占重要地位. 但是并非所有非固有歧义 CFL 都会被 DPDA 识别. 例如 L_{wwr} 有无歧义文法 $S \to 0S0 \mid 1S1 \mid \varepsilon$.

6.5.4 语言间的关系