(정의) 비결정적(Nondeterministic) Pushdown Automata(PDA) P는 요소 일곱 개로 이루어져있다. 즉 $P = (Q, \Sigma, \Gamma, \delta, q_o, Z_0, F)$로 쓰고

(1) Q은 **상태**(state)의 **유한** 집합(state vocabulary)이다.

(2) Σ은 **입력문자**의 **유한** 집합(input vocabulary)이다.

(3) Γ은 **스택문자**의 **유한** 집합(stack vocabulary)이다.

(4) δ는 **상태변화함수**(state transition function)로서 상태집합 Q와 입력문자, 스택문자 $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma$를 정의역으로, 상태 Q와 스택문자열의 집합[1]을 함수의 치역으로 가진다. 즉

$$\delta : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$$로 정의하고,

현 상태 $q \in Q$에서 **입력문자** $a \in \Sigma \cup \{\epsilon\}$와, **스택 top 문자** $X \in \Gamma$를 보고, **상태**가 $p \in Q$로 바뀌고, 스택 top 문자 X를 pop한 다음에 스택문자열 $\alpha \in \Gamma^*$를 push하여 스택 top이 α로 바뀔 때[2], $(p, \alpha) \in \delta(q, a, X)$이다.

(5) $q_0 \in Q$는 **처음상태**(initial state)라고 불리는 특별한(distinguished) 상태이다.

(6) $Z_0 \in \Gamma^*$는 **처음스택내용**(initial stack contents)이라고 불리는 특별한(distinguished) 스택문자열이다.

(7) $F \subseteq Q$는 **끝나는 상**태(final state)라고 불리는 특별한 상태들의 집합이다.

(정의) PDA $P = (Q, \Sigma, \Gamma, \delta, q_o, Z_0, F)$에서 상황(configuration, instantaneous description)
(상태, 남은 입력문자열, 스택 내용)
$(q, x, \gamma) \in (Q, \Sigma^*, \Gamma^*)$

(정의) PDA $P = (Q, \Sigma, \Gamma, \delta, q_o, Z_0, F)$에서 상황변화, $\vdash_P$.
$(q, ax, X\gamma) \vdash_P (p, x, \beta\gamma)$, if $(p, \beta) \in \delta(q, a, X)$.
$(q, x, X\gamma) \vdash_P (p, x, \beta\gamma)$, if $(p, \beta) \in \delta(q, \epsilon, X)$.
PDA P가 뻔하면, $\vdash_P$에서 P를 생략하고 $\vdash$로 쓴다.

(정의) 상황변화 $\vdash$의 반복횟수(recursive definition) $\vdash^n$, $n \geq 0$.
$\vdash^0 \stackrel{\text{def}}{=}_B id_{Q \times \Sigma^* \times \Gamma^*}$.
$\vdash^n \stackrel{\text{def}}{=}_R \vdash \circ \vdash^{n-1}$.

(정의) 상황변화 $\vdash$의 반복합(closure) $\vdash^+$와 $\vdash^*$.
$$\vdash^+ \stackrel{\text{def}}{=} \bigcup_{i \in N_1} \vdash^i = \vdash^1 \cup \vdash^2 \cup \vdash^3 \cup \cdots$$
$$\vdash^* \stackrel{\text{def}}{=} \bigcup_{i \in N_0} \vdash^i = \vdash^0 \cup \vdash^1 \cup \vdash^2 \cup \cdots$$

(정의) PDA $P = (Q, \Sigma, \Gamma, \delta, q_o, Z_0, F)$가 정의하는 언어, $L(P)$.
$$L(P) = \{x \in \Sigma^* \mid (q_0, x, Z_0) \vdash_P^* (f, \epsilon, \gamma),\ f \in F\}.$$
$$N(P) = \{x \in \Sigma^* \mid (q_0, x, Z_0) \vdash_P^* (q, \epsilon, \epsilon),\ q \in Q\}.$$

---

1) 상태와 스택문자열의 집합을 치역으로 허용하고 ϵ-move도 허용하므로 nondeterministic하다.

2) 스택 문자열 $\alpha = X_1 X_2 \cdots X_n$일 때, 오른쪽 끝 문자 X_n부터 push하여 α의 가장 왼쪽 문자열 X_1이 스택 top에 들어난다.

(정의) 제한된 PDA $P = (Q, \Sigma, \Gamma, \delta, q_o, Z_0, F)$에서

...

(3) δ: $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow 2^{Q \times \Gamma^*}$의 $(p, \beta) \in \delta(q, a, X)$에서 $\beta \in \Gamma^*$를 세 가지로 제한

1) $(p, YX) \in \delta(q, a, X)$, $(q, ax, X\gamma) \vdash_P (p, x, YX\gamma)$ push Y

2) $(p, X) \in \delta(q, a, X)$, $(q, ax, X\gamma) \vdash_P (p, x, X\gamma)$ 스택변화 없음

3) $(p, \epsilon) \in \delta(q, a, X)$, $(q, ax, X\gamma) \vdash_P (p, x, \gamma)$ pop X

...

(사실) 제한된 PDA $P = (Q, \Sigma, \Gamma, \delta, q_o, Z_0, F)$로 $L(P) = L(P')$인 보통 PDA $P' = (Q', \Sigma, \Gamma, \delta', q_o, Z_0, F)$를 흉내 낼 수 있다.

(3) δ는 **상태변화함수**(state transition function)로서 입력문자열 Σ^*, 스택문자Γ^*, 즉 $\Sigma^* \times \Gamma^*$를 정의역과 치역으로 가진다.

δ: $\Sigma^* \times \Gamma^* \rightarrow \Sigma^* \times \Gamma^*$로 정의한다. 단 입력문자열은 읽을 수 만 있으므로(read only) $\delta(xy, \alpha) = (y, \beta)$ (단 $q, p \in Q$, $x, y \in \Sigma^*$, $\alpha, \beta \in \Gamma^*$)의 형태만 허용한다.

XXX **현 상태**가 $q \in Q$에서 **입력문자** $a \in \Sigma \cup \{\epsilon\}$[3]와, **스택 top 문자** $X \in \Gamma$를 보고, **상태**가 $p \in Q$로 바뀌고, 스택 top 문자 X를 pop한 다음에 스택문자열 $\alpha \in \Gamma^*$를 push하여 스택 top이 α로 바뀔 때[4]. $\delta(q, a, X) = (p, \alpha)$라 쓴다.

(4) $Z_0 \in \Gamma^*$는 **처음상태**(initial stack contents)라고 불리는 특별한 스택문자열이다.

(5) $F \subseteq \Gamma^*$는 **끝나는 상**태(final state)라고 불리는 특별한 스택문자열의 집합이다.

확장된 Pushdown Automata(PDA) P는 요소 일곱 개로 이루어져있다. 즉 $P = (Q, \Sigma, \Gamma, \delta, q_o, Z_0, F)$로 쓰고

(1) Q은 **상태**(state)의 **유한** 집합(state vocabulary)이다.

(2) Σ은 **입력문자**의 **유한** 집합(input vocabulary)이다.

(3) Γ은 **스택문자**의 **유한** 집합(stack vocabulary)이다.

(4) δ는 **상태변화함수**(state transition function)로서 상태집합 Q와 입력문자열 Σ^*, 스택문자Γ^*, 즉 $Q \times \Sigma^* \times \Gamma^*$를 정의역과 치역으로 가진다.

δ: $Q \times \Sigma^* \times \Gamma^* \rightarrow Q \times \Sigma^* \times \Gamma^*$로 정의한다. 단 입력문자열은 읽을 수 만

3) ϵ-move를 허용한다.

4) 스택 문자열 $\alpha = X_1 X_2 \cdots X_n$일 때, 오른쪽 끝 문자 X_n부터 push하여 α의 가장 왼쪽 문자열 X_1이 스택 top에 들어난다.

있으므로(read only) $\delta(q, xy, \alpha) = (p, y, \beta)$ (단 $q, p \in Q$, $x, y \in \Sigma^*$, $\alpha, \beta \in \Gamma^*$)의 형태만 허용한다.

XXX **현 상태**가 $q \in Q$에서 **입력문자** $a \in \Sigma \cup \{\epsilon\}$[5]와, **스택 top 문자** $X \in \Gamma$를 보고, **상태**가 $p \in Q$로 바뀌고, 스택 top 문자 X를 pop한 다음에 스택문자열 $\alpha \in \Gamma^*$를 push하여 스택 top이 α로 바뀔 때[6]. $\delta(q, a, X) = (p, \alpha)$라 쓴다.

(5) $q_0 \in Q$는 **처음상태**(initial state)라고 불리는 특별한 상태이다.

(6) $Z_0 \in \Gamma^*$는 **처음상태**(initial stack contents)라고 불리는 특별한 스택문자열이다.

(7) $F \subseteq Q$는 **끝나는 상**태(final state)라고 불리는 특별한 상태들의 집합이다.

(**정의** 5.2) **고쳐 쓰기**(유도; rewriting, derivation)

고쳐 쓰기 틀 $R = (A, P)$에서 규칙 $\alpha \to \beta \in P$에 대하여, $\gamma, \delta \in A^*$가 임의의 문자열일 때, $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$로 쓰고 문자열 $\gamma\alpha\delta$를 규칙 $\alpha \to \beta \in P$를 써서 $\gamma\beta\delta$로 **다시 썼다**(rewrite; derive)고 하고, 특별히 **다시 쓴** 규칙 $r = \alpha \to \beta \in P$를 표현하고 싶을 때는 $\gamma\alpha\delta \Rightarrow^r \gamma\beta\delta$로 표현한다.

임의의 문자열의 부분문자열(substring)이 규칙 $\alpha \to \beta \in P$의 좌변 α과 같으면 나머지 부분(prefix γ과 suffix δ)을 제외한 α만 규칙의 우변 β로 바꾸어서 **다시 쓸**($\gamma\alpha\delta \Rightarrow^{\alpha\to\beta} \gamma\beta\delta$)수 있다.

다시 쓰는 관계가 가진 기본문자집합을 일부 분류하여 다음과 같이 **문법**을 정의한다.

(**정의** 5.3) 다시 쓰기 틀 $R = (N \cup T, P)$에서 **문법** $G = (N, T, P, S)$을 아래로 정의한다.

i) N은 넌 터미널(nonterminal; variable) 문자에 집합,

ii) T는 터미널(terminal) 문자에 집합, 단 $N \cap T = \varnothing$,

iii) 규칙 P는 $(N \cup T)^*$에서 정의된 관계이고, 특히 문법규칙이라고도 부른다.

iv) $S \in N$은 시작문자(start symbol).

문법은 다시 쓰기 틀에서 문장(sentence)에 구분을 위하여 기본문자를 **터미널**과 **넌 터미널**로 나누고, 문장 다시 쓰기의 **시작**을 위하여 시작문자 S를 별도로 정의한 것이다.

(**정의** 5.4) 문법 $G = (N, T, P, S)$에 **언어** $L(G)$를 아래로 정의한다.

$$L(G) = \{x \in T^* \mid S \Rightarrow^* x\}.$$

문법에 새로 쓰기 중간과정에서 나타나는 문자열을 **문장형태**(sentential form)라고 부르고 문장형태가 터미널 문자만으로 이루어 졌을 때 **문장**(sentence) 된다.

---

5) ϵ-move를 허용한다.

6) 스택 문자열 $\alpha = X_1 X_2 \cdots X_n$일 때, 오른쪽 끝 문자 X_n부터 push하여 α의 가장 왼쪽 문자열 X_1이 스택 top에 들어난다.

5.2. 문맥자유 문법

(**정의** 5.5) **문맥자유(context-free) 문법(grammar)**

(정의 5.4)에서 문법규칙에 **좌변**이 **넌 터미널 문자**로 제한된다. 즉 $A \to \alpha \in P$에서 $A \in N$이고 $\alpha \in (N \cup T)^*$이다.

(**정의** 5.6) 문맥자유(context-free) 언어(language)

임의의 언어 L을 만들어내는 **문맥자유문법** G가 있을 때, $L = L(G)$, 언어 L을 **문맥자유언어**라 부른다.

5.3 문맥자유문법과 파스트리 그리고 고쳐 쓰기순서

$S \Rightarrow^{\pi} x \in T^*$이고 $\pi \in P^*$일 때, 사용된 문법규칙으로 subtree를 만들면 전체 문장 $x \in T^*$에 대한 트리를 만들 수 있고, 이를 파스트리라 부른다.

문맥자유문법의 파싱(parsing)은 임의의 터미널 문자열 $x \in T^*$가 주어진 문법 G에 문장이면 해당하는 파스트리를 만들어주고, 문장이 아니면 이를 알려주는 과정이다.

문맥자유문법의 **문장형태**는 일반적으로 여러 개에 넌 터미널을 가지는데, 어떤 넌 터미널이 다시 쓰기에 먼저 참여할까 하는 것은 파스트리에 최종모양과는 관계가 없다. 따라서 문장형태에 가장 왼쪽에 있는 넌 터미널부터 다시 쓰기를 하는 왼쪽부터 고쳐쓰기(leftmost derivation)과 오른쪽부터 고쳐쓰기(rightmost derivation)에 두 가지 극단적인 경우를 생각할 수 있다. 이를 각각 $\Rightarrow_{lm}$와 $\Rightarrow_{\mathrm{rm}}$로 표시한다.

$$S \Rightarrow_{lm}^{\pi_1^L} xA\gamma \Rightarrow_{lm} x\beta\gamma \Rightarrow_{lm}^{\pi_2^L} xy\gamma \Rightarrow_{lm}^{\pi_3^L} xyz, \qquad \pi_1^L, \pi_2^L, \pi_3^L \in P^*,$$
$$\text{단 } x \in T^*,\ \gamma \in (N \cup T)^*,\ A \to \beta \in P,\ y, z \in T^*,\ \beta \Rightarrow^* y,\ \gamma \Rightarrow^* z.$$
$$S \Rightarrow_{\mathrm{rm}}^{\pi_1^R} \alpha Az \Rightarrow_{\mathrm{rm}} \alpha\beta z \Rightarrow_{\mathrm{rm}}^{\pi_2^R} \alpha yz \Rightarrow_{\mathrm{rm}}^{\pi_3^R} xyz, \qquad \pi_1^R, \pi_2^R, \pi_3^R \in P^*,$$
$$\text{단 } z \in T^*,\ \alpha \in (N \cup T)^*,\ A \to \beta \in P,\ y, x \in T^*,\ \beta \Rightarrow^* y,\ \alpha \Rightarrow^* x.$$

이 때 $\pi_L = \pi_1^L(A \to \beta)\pi_2^L\pi_3^L \in P^*$라 하고 $\pi_R = \pi_1^R(A \to \beta)\pi_2^R\pi_3^R \in P^*$라 하면 π_L과 π_R를 각각 문장 x에 대한 left parse와 right parse라고 부른다. 문장 x에 대한 left parse나 right parse가 정해지면 해당하는 파스트리도 정해진다.