문맥무관 언어

제 1 절 문맥무관 문법

정의 1 *문법(grammar) G는 네 가지 요소 (V, Σ, S, P)로 구성된다.*

1. *V는 변수들의 유한 집합이고,*
2. *Σ는 알파벳이고,*
3. *$S \in V$는 시작변수이고,*
4. *P는 생성규칙들의 유한 집합이다. 각 생성규칙은 $x \to y$의 형태를 가지는데, 여기에서 $x \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$ 이고 $y \in (V \cup \Sigma)^*$ 이다.*

예를 들면 $A, B \in V$와 $a, b \in \Sigma$에 대하여

$$aAb \to bBa$$

는 생성규칙이다.

정의 2 *문법 $G = (V, \Sigma, S, P)$의 모든 생성규칙이 $A \in V$와*

$x \in (V \cup \Sigma)^*$*에 대하여*

$$A \to x$$

형태이면 G*는 문맥무관 문법(context-free grammar)이라고 부른다.*

$A \to x$와 $A \to y$를 줄여서

$$A \to x \mid y$$

로 표시한다.

문법은 문장을 표현하는 데 광범위하게 사용된다. 변수는 일반적으로 문장 성분을 나타낸다. 예를 들어, "구름이 산을 넘는다"는 문장을 다음 문법을 이용하여 만들 수 있다.

$$\begin{aligned}
\langle \text{문장} \rangle &\to \langle \text{주어} \rangle \langle \text{목적어} \rangle \langle \text{서술어} \rangle \\
\langle \text{주어} \rangle &\to \langle \text{명사} \rangle \langle \text{주격조사} \rangle \\
\langle \text{목적어} \rangle &\to \langle \text{명사} \rangle \langle \text{목적격조사} \rangle \\
\langle \text{서술어} \rangle &\to \langle \text{동사} \rangle
\end{aligned}$$

$$\begin{aligned}
\langle \text{명사} \rangle &\to \text{구름} \mid \text{산} \\
\langle \text{주격조사} \rangle &\to \text{이} \\
\langle \text{목적격조사} \rangle &\to \text{을} \\
\langle \text{동사} \rangle &\to \text{넘는다}
\end{aligned}$$

이제 문장을 만들기 위해서는 시작변수 S에서부터 생성규칙을 차례로 적용한다. 일반적으로 $u, v \in (V \cup \Sigma)^*$와 $A \in V$에 대하여 uAv에 생성규칙 $A \to w$를 적용하면 uwv를 얻는데, 이 과정을 유도(derivation)라고 부르고

$$uAv \Rightarrow uwv$$

로 표시한다. 따라서

$$\begin{aligned}
\langle \text{문장} \rangle &\Rightarrow \langle \text{주어} \rangle \langle \text{목적어} \rangle \langle \text{서술어} \rangle \\
&\Rightarrow \langle \text{명사} \rangle \langle \text{주격조사} \rangle \langle \text{목적어} \rangle \langle \text{서술어} \rangle \\
&\Rightarrow \text{구름} \langle \text{주격조사} \rangle \langle \text{목적어} \rangle \langle \text{서술어} \rangle
\end{aligned}$$

$\Rightarrow$ 구름이⟨목적어⟩⟨서술어⟩

$\Rightarrow$ 구름이⟨명사⟩⟨목적격조사⟩⟨서술어⟩

$\Rightarrow$ 구름이 산⟨목적격조사⟩⟨서술어⟩

$\Rightarrow$ 구름이 산을⟨서술어⟩

$\Rightarrow$ 구름이 산을⟨동사⟩

$\Rightarrow$ 구름이 산을 넘는다

x에 생성규칙을 0번 이상 적용하여 y를 얻으면, 이를 $x \stackrel{*}{\Rightarrow} y$로 표시한다.

- $S \stackrel{*}{\Rightarrow} x$ $(x \in (V \cup \Sigma)^*)$이면 x를 문법 G의 문장형태라고 부른다.
- $S \stackrel{*}{\Rightarrow} w$ $(w \in \Sigma^*)$이면 w를 문법 G의 문장(또는 G가 생성하는 스트링)이라고 부른다.

위의 예에서 "구름이⟨목적어⟩⟨서술어⟩"는 문장형태이고 "구름이 산을 넘는다"는 문장이다. 그러나 이 문법이 "산이 구름을 넘는다"도 만들 수 있음을 유의하라.

문법 G의 언어 $L(G)$는 G가 생성하는 문장의 집합이다. 즉

$$L(G) = \{w \in \Sigma^* : S \stackrel{*}{\Rightarrow} w\}.$$

예제 1 $G = (\{S\}, \{0,1\}, S, P)$ *이고* P*가*

$$S \to 0S1 \mid \epsilon$$

이면 G*는 문맥무관 문법이다.*

$$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow 0011$$

에서 보는 것처럼 $L(G) = \{0^n 1^n : n \geq 0\}$*임을 알 수 있다.*

정의 3 *문맥무관 문법이 생성하는 언어를 문맥무관 언어(context-free language)라고 부른다.*

예제 2 $G = (\{S\}, \{(,)\}, S, P)$*이고* P*가*

$$S \to SS \mid (S) \mid \epsilon$$

이면,

$$S \Rightarrow SS \stackrel{*}{\Rightarrow} (S)(S) \stackrel{*}{\Rightarrow} ()()$$
$$S \Rightarrow SS \stackrel{*}{\Rightarrow} ((S))(S) \stackrel{*}{\Rightarrow} (())()$$

이다. 즉 G는 여는 괄호와 닫는 괄호가 올바로 쌍을 이루는 스트링을 생성한다.

예제 3 다음 문맥무관 문법은 수식을 생성한다.

$$\begin{aligned} E &\rightarrow E + T \mid T \\ T &\rightarrow T * F \mid F \\ F &\rightarrow (E) \mid \texttt{id} \end{aligned}$$

즉

$$\begin{aligned} E &\Rightarrow E + T \\ &\Rightarrow E + T * F \\ &\Rightarrow E + T * (E) \end{aligned}$$

$$\begin{aligned} &\Rightarrow E + T * (E + T) \\ &\stackrel{*}{\Rightarrow} F + F * (F + F) \\ &\stackrel{*}{\Rightarrow} \texttt{id} + \texttt{id} * (\texttt{id} + \texttt{id}) \end{aligned}$$

예제 4 $L = \{0^m 1^n : m \neq n\}$을 생성하는 문맥무관 문법을 구하라.

$m > n$과 $m < n$의 경우로 나누어서 생각한다. 문맥무관 문법을 만들기 위해서는 어떤 변수가 무슨 스트링을 만들 것인지를 정해야 한다. $m > n$인 경우, 변수 A가 0과 1의 개수가 같은 스트링을 만들도록 하고, 변수 B가 1개 이상의 0을 만들면 된다.

$$\begin{aligned} S &\rightarrow BA \\ A &\rightarrow 0A1 \mid \epsilon \\ B &\rightarrow 0B \mid 0 \end{aligned}$$

$m < n$인 경우, 변수 A는 그대로 이용하면 되고 변수 C가 1개 이상

의 1을 만들면 된다.

$$\begin{aligned} S &\rightarrow AC \\ C &\rightarrow 1C \mid 1 \end{aligned}$$

예제 5 $L = \{a^n b^m c^k : n + m = k\}$를 생성하는 문맥무관 문법을 구하라.

예제 6 $L = \{0^n 1^m : n \leq m \leq 2n\}$을 생성하는 문맥무관 문법을 구하라.

예제 7 $L = \{a^m b^n c^u d^v : m + n = u + v\}$를 생성하는 문맥무관 문법을 구하라.

예제 8 $L_{ab} = \{w \in \{a, b\}^* : N_a(w) = N_b(w)\}$를 생성하는 문맥무관 문법을 구하라.

간단한 답은

$$S \rightarrow aSb \mid bSa \mid SS \mid \epsilon$$

이다. 또 다른 답은

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

이다. 여기에서 A는 a가 b보다 하나 많은 스트링을 만들어내고 B는 b가 a보다 하나 많은 스트링을 만들어낸다.

문맥무관 문법이 어떤 언어를 만드는지 쉽게 확인할 수 없을 때 증명을 의해 이를 밝힐 수 있다.

예제 9 예제 8의 간단한 문법을 G라고 할 때, $L(G) = L_{ab}$임을 증명하라.

증명. $L(G) \subseteq L_{ab}$: G에서 알파벳 글자를 만드는 생성규칙은 $S \rightarrow aSb$와 $S \rightarrow bSa$이므로, G가 생성하는 모든 스트링은 같은 수의 a와 b를 갖는다.

$L_{ab} \subseteq L(G)$: $w \in L_{ab}$의 길이에 대한 귀납법으로 증명하자.

$|w| = 0$일 때, G가 ϵ을 생성한다. $|w| < k$일 때, G가 w를 생성한다고 (즉 $S \stackrel{*}{\Rightarrow} w$) 가정하자.

길이가 k인 $w \in L_{ab}$의 양쪽 끝 글자를 보자. 다음 네 가지 경우가 있다: $w = axb$, $w = bxa$, $w = axa$, $w = bxb$.

- $w = axb$인 경우: x는 같은 수의 a와 b를 가지고 있고 그 길이가 k 미만이므로, 귀납 가설에 의해 $S \stackrel{*}{\Rightarrow} x$이다. 따라서 $S \Rightarrow aSb \stackrel{*}{\Rightarrow} axb$이므로, G가 w를 생성한다. ($w = bxa$인 경우도 마찬가지)
- $w = axa$인 경우: 그림 1과 같이 w의 왼쪽 끝에 0을 놓고 오른쪽으로 진행하면서 a가 나오면 $+1$을 하고 b가 나오면 -1을 하자. w가 같은 수의 a와 b를 가지고 있으므로 오른쪽 끝의 값은 0이 되고, 그 바로 앞의 값은 -1이 된다. x의 왼쪽 끝은 $+1$이고, 오른 쪽 끝은 -1이므로 x의 중간에 0이 반드시 나타나야 한다. w에서 이 0의 왼쪽 부분을 u, 오른쪽 부분을 v라고 하면 (즉 $w = uv$), u와 v는 각각 같은 수의 a와 b를 갖고, 그 길이는 k 미만이다. 귀납 가설에 의해 $S \stackrel{*}{\Rightarrow} u$와 $S \stackrel{*}{\Rightarrow} v$가 성립한다. 따

라서 $S \Rightarrow SS \stackrel{*}{\Rightarrow} uv$이므로, G가 w를 생성한다. ($w = bxb$인 경우도 마찬가지)

□

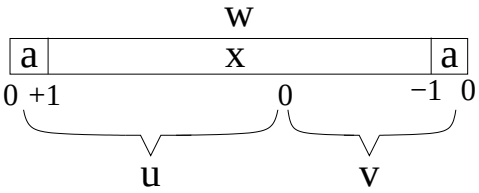

그림 1:

제 2 절 정규문법

정의 4 문법 $G = (V, \Sigma, S, P)$의 모든 생성규칙이 $A, B \in V$와

$w \in \Sigma^*$ *에 대하여*

$$\begin{aligned} A &\rightarrow wB \\ A &\rightarrow w \end{aligned}$$

형태이면 G*는 정규문법이라고 부른다.*

정규문법의 유도과정에서 문장형태는 항상 오른쪽 끝에 한 개의 변수만을 갖게 된다.

$$\begin{aligned} S &\Rightarrow w_1 A \\ &\Rightarrow w_1 w_2 B \\ &\Rightarrow w_1 w_2 w_3 C \\ &\Rightarrow w_1 w_2 w_3 w_4 \end{aligned}$$

정규문법이 생성하는 언어의 집합이 정규언어 종류임을 보이고자 한다.

정리 1 L*이 정규언어이면,* $L = L(G)$*인 정규문법* G*가 존재한다.*

증명. L이 정규언어이므로 L을 받아들이는 DFA $M = (Q, \Sigma, \delta, q_0, F)$이 존재한다. M의 전이과정을 흉내내는 정규문법 $G = (Q, \Sigma, q_0, P)$를 만들고자 한다. (DFA의 상태가 문법의 변수가 되는 것에 유의) 즉 그림 2처럼 DFA가 w를 읽고 상태 q에 도달하면, 문법은 $q_0 \stackrel{*}{\Rightarrow} wq$에 이르도록 문법 G를 만든다.

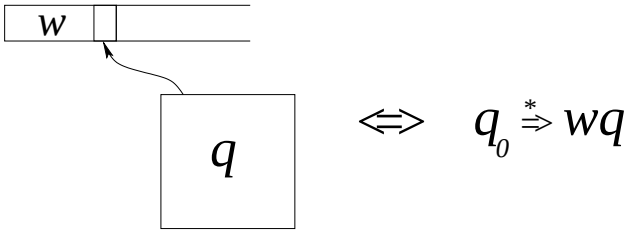

그림 2:

P는 다음과 같이 정의된다.

1. 각 전이 $\delta(p, a) = q$에 대하여 $p \rightarrow aq$를 P에 더한다.
2. 각 최종 상태 $q \in F$에 대하여 $q \rightarrow \epsilon$를 P에 더한다.

$\delta^*(p, w) = q$인 경우에만 $p \stackrel{*}{\Rightarrow} wq$임을 보일 수 있다. 따라서 $\delta^*(q_0, w) \in F$인 경우에만 $q_0 \stackrel{*}{\Rightarrow} w$이다. 즉 $L(M) = L(G)$. □

예제 10 *예제 ??(그림 ??)의 DFA와 동등한 정규문법을 구하라. (죽은 상태는 생략)*

$$\begin{aligned} q_1 &\rightarrow 0q_2 \mid \epsilon \\ q_2 &\rightarrow 1q_3 \\ q_3 &\rightarrow 0q_4 \mid \epsilon \\ q_4 &\rightarrow 0q_2 \mid 1q_3 \mid \epsilon \end{aligned}$$

정리 2 *정규문법 G가 생성하는 언어 $L(G)$는 정규언어이다.*

증명. $G = (V, \Sigma, S, P)$라고 하자. G의 유도과정을 흉내내는 NFA $M = (Q, \Sigma, \Delta, S, \{q_f\})$를 만들고자 한다. 정규문법에 의해 만들어지는 문장형태는 항상 wA $(w \in \Sigma^*, A \in V)$이므로, 이 때 NFA는 w를 읽고 상태 A에 도달해 있도록 만든다.

상태 집합 Q는 $V \cup \{q_f\}$와 중간 상태들로 구성된다. Δ는 다음과 같이 정의된다.

1. 각 생성규칙 $A \rightarrow wB$에 대하여, $\Delta^*(A, w)$에 B를 더해 준다.
2. 각 생성규칙 $A \rightarrow w$에 대하여, $\Delta^*(A, w)$에 q_f를 더해 준다.

이 때, w의 길이가 2 이상이면 $\Delta^*(A, w)$를 표시하기 위하여 중간 상태를 필요로 한다.

$S \stackrel{*}{\Rightarrow} w$인 경우에만 $q_f \in \Delta^*(S, w)$임을 귀납법으로 증명할 수 있다. 즉 $L(G) = L(M)$. □

예제 11 *정규문법*

$$\begin{aligned} S &\rightarrow aA \\ A &\rightarrow abS \mid a \end{aligned}$$

와 동등한 NFA를 구하라. 그림 3를 보라.

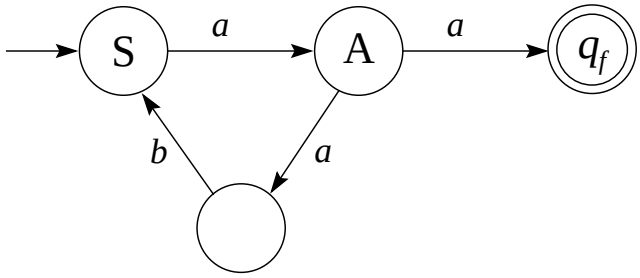

그림 3:

정규문법은 문맥무관 문법의 일종이므로, 모든 정규언어는 문맥무관언어이다. 즉, 정규언어 종류는 문맥무관언어 종류의 부분집합이다.

언어는 스트링의 집합이고, 언어종류는 언어의 집합임에 유의하라. 그림 4에서 언어 $\{0^n1^n\}$은 언어 $L(0^*1^*)$의 부분집합이지만, $\{0^n1^n\}$은 정규언어가 아니고 문맥무관언어이다.

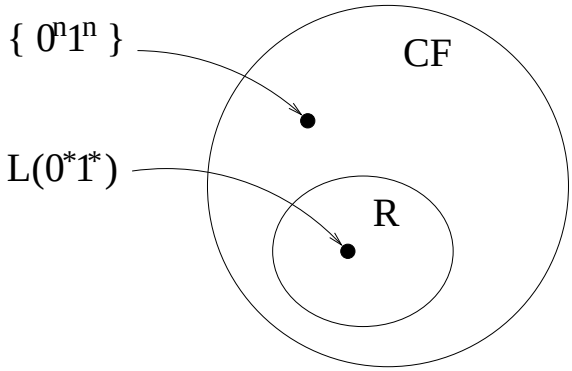

그림 4:

제 3 절 파스 트리

$G = (V, \Sigma, S, P)$를 문맥무관 문법이라고 하자. G에서의 유도과정을 파스 트리로 표시할 수 있다. G에 대한 파스 트리(parse tree)는

다음과 같다.

1. 루트 노드의 이름은 S이다.
2. 내부 노드의 이름은 V의 원소이고, A 이름의 내부 노드가 $u_1, \ldots, u_r$ 이름의 자식 노드들을 가지고 있으면 $A \to u_1 \cdots u_r$은 P의 생성규칙이어야 한다.
3. 단말 노드의 이름은 $\Sigma \cup \{\epsilon\}$의 원소이고, 단말 노드의 이름이 ϵ이면 이 노드는 부모 노드의 유일한 자식 노드이어야 한다.

단말 노드의 이름을 왼쪽부터 오른쪽으로 접합시킨 것을 파스 트리의 열매 스트링이라고 부른다. $S \stackrel{*}{\Rightarrow} w$를 파스 트리로 나타내면 그 열매 스트링이 w가 된다.

예제 12 *예제 2에서 유도과정*

$$S \Rightarrow SS \Rightarrow (S)S \Rightarrow ()S \Rightarrow ()(S) \Rightarrow ()()$$

는 다음 파스 트리로 나타내어진다.

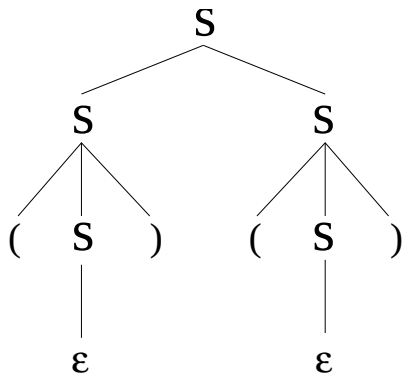

그림 5:

유도과정에서 항상 맨 왼쪽의 변수에 생성규칙을 적용하는 것을 좌측 유도(leftmost derivation)라고 부른다. 한 개의 파스 트리에 대하여 여러 가지 유도 과정이 있을 수 있지만 좌측 유도는 한 가지만 존재한다. 즉 파스 트리와 좌측 유도는 1대1 대응관계를 갖는다.

예제 13 *앞 예제에서 유도과정*

$$S \Rightarrow SS \Rightarrow S(S) \Rightarrow S() \Rightarrow (S)() \Rightarrow ()()$$

도 파스 트리로 나타내면 위의 그림과 같아진다.

정의 5 *문맥무관 문법 G가 어떤 $w \in L(G)$에 대하여 두 개 이상의 파스 트리를 가지면, G는 애매하다(ambiguous)라고 말한다.*

예제 14 *다음 문법 G를 고려하자.*

$$\begin{aligned} E &\rightarrow E + E \\ E &\rightarrow E * E \\ E &\rightarrow a \mid b \mid c \end{aligned}$$

$a + b * c$는 그림 6과 같이 두 개의 파스 트리를 가지므로 G는 애매하다. 그러나 이 문법은 예제 3처럼 애매하지 않은 문법으로 바꿀 수 있다.

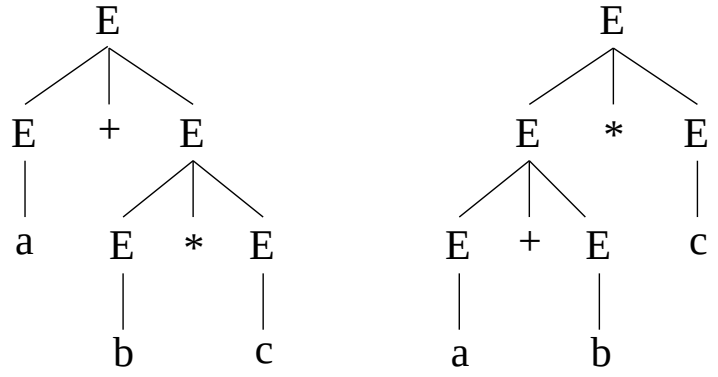

그림 6:

예제 15

$$\begin{aligned} S &\rightarrow \texttt{if } C \texttt{ then } S \texttt{ else } S \\ S &\rightarrow \texttt{if } C \texttt{ then } S \\ S &\rightarrow x \mid y \end{aligned}$$

$$C \quad \rightarrow \quad a \mid b$$

`if` a `then if` b `then` x `else` y는 두 개의 파스 트리를 가진다. 이 경우 일반적으로 문법은 그대로 둔 채 "`else`는 가장 가까운 `if`에 붙는다"는 의미(*semantics*)를 부여하여 애매성을 없앤다.

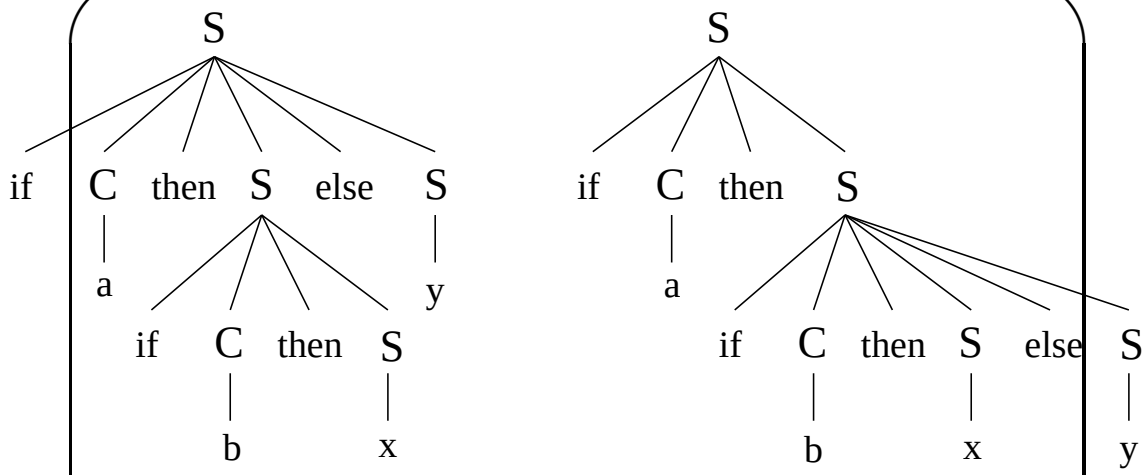

그림 7:

정의 6 *문맥무관 언어 L을 생성하는 애매하지 않은 문법 G가 존재하면, L은 "애매하지 않다"라고 한다. L을 생성하는 모든 문법이 애매하면, L은 "본질적으로 애매하다"라고 한다.*

예제 16 $L = \{a^n b^n c^m\} \cup \{a^m b^n c^n\}$*은 본질적으로 애매하다.*

(증명이 아니고 설명) L*을 생성하는 어떤 문법이든지* $\{a^n b^n c^m\}$*과* $\{a^m b^n c^n\}$*을 서로 다른 방법으로 만들어야 하므로* $a^n b^n c^n$*을 만드는 파스 트리가 2개 이상 존재하게 된다.*

제 4 절 표준형

문맥무관 언어 L이 ϵ을 포함하면, L을 생성하는 문법은 반드시 $A \to \epsilon$ 형태의 생성규칙을 가져야 한다. 이 경우 시작변수 S_0에서

$$S_0 \to \epsilon \mid S$$

의 생성규칙을 만든 후, S에서 $L - \{\epsilon\}$을 생성하면 된다. 따라서 이후의 문맥무관 언어는 ϵ을 포함하지 않는다고 가정한다.

정의 7 (Chomsky 표준형) *문맥무관 문법* $G = (V, \Sigma, S, P)$*의 모든 생성규칙이* $A, B, C \in V$*와* $a \in \Sigma$*에 대하여*

$$\begin{aligned} A &\to BC \\ A &\to a \end{aligned}$$

형태이면, G*를 Chomsky 표준형이라고 부른다.*

정의 8 (Greibach 표준형) *문맥무관 문법* $G = (V, \Sigma, S, P)$*의 모든 생성규칙이* $A \in V$*,* $a \in \Sigma$*,* $x \in V^*$*에 대하여*

$$A \to ax$$

형태이면, G*를 Greibach 표준형이라고 부른다.*

예제 8의 두 번째 답은 Greibach 표준형이다.

정리 3 *모든 문맥무관 언어는 Chomsky 표준형의 문법에 의해 생성된다.*

증명. (**) 임의의 문맥무관 문법 $G = (V, \Sigma, S, P)$를 다음 과정에 의해 Chomsky 표준형으로 바꾼다.

1. P에서 $A \to \epsilon$ 형태의 ϵ 생성규칙을 없앤다.
   (a) $A \stackrel{*}{\Rightarrow} \epsilon$을 만족하는 모든 변수의 집합 V_ϵ을 구한다.
      i. $A \to \epsilon$인 A를 V_ϵ에 넣는다.
      ii. $A \to B_1 \cdots B_r$에서 모든 B_i $(1 \le i \le r)$이 V_ϵ 의 원소이면 A를 V_ϵ에 넣는다.
   (b) 새로운 생성규칙의 집합 P_1을 만든다. P의 각 생성규칙 $A \to x_1 \cdots x_t$ $(x_i \in V \cup \Sigma)$에 대하여 x_i가 V_ϵ의 원소이면 그 자리에 x_i와 ϵ을 번갈아 넣어 만들어지는 모든 생성규칙을 P_1에 넣는다. 단 $A \to \epsilon$이 만들어지는 경우 이것은 P_1에 넣지 않는다.
   (c) $G_1 = (V, \Sigma, S, P_1)$는 G와 동등하다.
2. P_1에서 $A \to B$ 형태의 단위 생성규칙을 없앤다.
   (a) $A \stackrel{*}{\Rightarrow} B$를 만족하는 변수들을 구한다. 단위 생성규칙

$A \to B$에 대하여 A 정점에서 B 정점으로 가는 간선을 가지는 그래프를 만들면 $A \stackrel{*}{\Rightarrow} B$ 관계를 파악할 수 있다.
   (b) P_1에서 단위 생성규칙이 아닌 것을 P_2에 넣는다.
   (c) $A \stackrel{*}{\Rightarrow} B$이고 P_1의 원소 $B \to x$이 단위 생성규칙이 아니면 $A \to x$를 P_2에 넣는다.
   (d) $G_2 = (V, \Sigma, S, P_2)$은 G_1과 동등하다.
3. P_2의 모든 생성규칙을 Chomsky 표준형으로 바꾼다.
   (a) $A \to a$인 생성규칙은 이미 Chomsky 표준형이므로 P_3에 넣는다.
   (b) $r \ge 2$인 각 생성규칙 $A \to x_1 \cdots x_r$ $(x_i \in V \cup \Sigma)$에서 x_i가 알파벳 글자 a이면 새로운 변수 C_a를 만들고 $C_a \to a$를 P_3에 넣고 x_i를 C_a로 대치한다. 그러면 생성규칙의 우변에 변수만이 나타난다.
   (c) 생성규칙 $A \to B_1 \cdots B_r$에 대하여 새로운 변수

$D_1, \ldots, D_{r-2}$을 도입하고

$$\begin{aligned} A &\rightarrow B_1 D_1 \\ D_1 &\rightarrow B_2 D_2 \\ &\cdots \\ D_{r-3} &\rightarrow B_{r-2} D_{r-2} \\ D_{r-2} &\rightarrow B_{r-1} B_r \end{aligned}$$

을 P_3에 넣는다.

(d) V와 새 변수들을 합하여 V'라 하면, $G_3 = (V', \Sigma, S, P_3)$은 G_2와 동등하다.

□

예제 17 *다음 문맥무관 문법을 Chomsky 표준형으로 바꾸라.*

$$S \rightarrow aB \mid AAa$$

$$\begin{aligned} A &\rightarrow B \mid S \\ B &\rightarrow b \mid \epsilon \end{aligned}$$

$V_\epsilon = \{B, A\}$이므로 *1* 단계 후 G_1은 다음과 같다

$$\begin{aligned} S &\rightarrow a \mid aB \mid AAa \mid Aa \\ A &\rightarrow B \mid S \\ B &\rightarrow b \end{aligned}$$

$A \stackrel{*}{\Rightarrow} B$, $A \stackrel{*}{\Rightarrow} S$이므로 *2* 단계 후 G_2는 다음과 같다

$$\begin{aligned} S &\rightarrow a \mid aB \mid AAa \mid Aa \\ A &\rightarrow b \mid a \mid aB \mid AAa \mid Aa \\ B &\rightarrow b \end{aligned}$$

3 단계 후 *Chomsky* 표준형은 다음과 같다.

$$S \rightarrow a \mid CB \mid AD \mid AC$$

$$
\begin{aligned}
A &\rightarrow b \mid a \mid CB \mid AD \mid AC \\
B &\rightarrow b \\
C &\rightarrow a \\
D &\rightarrow AC
\end{aligned}
$$

정리 4 *모든 문맥무관 언어는 Greibach 표준형의 문법에 의해 생성된다.*

제 5 절 내리누름 오토마타

내리누름 오토마타(pushdown automata)를 줄여서 PA로 부르기로 하자. PA는 입력 테입과 제어장치외에 내리누름 저장장치인 스택을 가지고 있다. 내리누름 오토마타를 스택 오토마타라고 부르기도 한다. 그림 8를 보라.

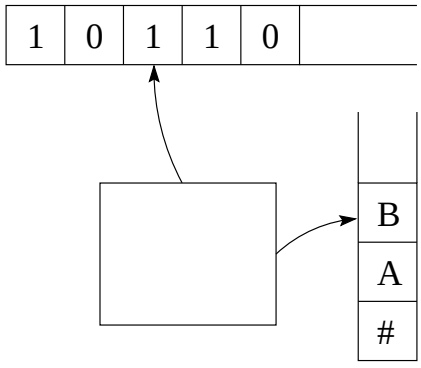

그림 8:

정의 9 *내리누름 오토마타 M은 여섯 가지 요소 $(Q, \Sigma, \Gamma, \Delta, q_0, F)$로 구성된다. 여기에서*

1. *Q는 상태들의 유한 집합이고*
2. *Σ는 입력 알파벳이고*
3. *Γ는 스택 알파벳이고 시작 글자 #를 포함한다.*

4. Δ는 $(Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\})) \times (Q \times \Gamma^*)$*의 부분집합인 전이 관계이고*
5. $q_0 \in Q$*는 초기 상태이고*
6. $F \subseteq Q$*는 최종상태들의 집합이다.*

PA는 Δ가 전이 관계이므로 비결정 오토마타이다. 따라서 유한 오토마타 때와 마찬가지로 최종상태는 한 개만 있다고 가정할 수 있다. $((p, a, A), (q, u)) \in \Delta$는 PA의 상태가 p이고 입력 글자가 a이고 스택의 맨위 원소가 A일 때, a를 읽고 스택의 A를 u로 바꾸고 q 상태가 되는 전이를 나타낸다. 따라서 $((p, 0, 0), (q, 10))$은 스택에 1을 넣는(push) 연산이고, $((p, 0, 0), (q, \epsilon))$는 스택의 맨위 원소 0을 빼는(pop) 연산이다. 전이 관계에서 $\Sigma \cup \{\epsilon\}$의 ϵ는 입력을 읽지 않고 전이가 일어날 수 있음을 나타내고, $\Gamma \cup \{\epsilon\}$의 ϵ는 스택을 보지 않고 전이가 일어날 수 있음을 나타낸다.

PA는 시작할 때, 스택에 #만을 가지고 있다. #는 스택의 밑바닥을 나타내는 기능을 한다. PA의 전이 도중 #를 스택에서 빼는 연산은 하지 않는다고 가정한다.

PA의 상황(configuration)은 현재 상태, 입력 스트링의 읽지 않은 부분과 현재 스택의 내용에 의해 결정된다. 입력 스트링이 w일 때, 시작 상황은 $(q_0, w, \#)$이다. 한 번의 전이에 의해서 PA M의 상황이 변화하는 과정을 $\vdash_M$으로 표시한다. 현재 상황이 $(q, 01101, 100\#)$이고 $((q, 0, \epsilon), (p, 1)) \in \Delta$이면,

$$(q, 01101, 100\#) \vdash_M (p, 1101, 1100\#).$$

PA는 입력 스트링 w를 읽은 후의 상태가 최종상태이면 (스택의 내용에 관계없이) w를 받아들인다. 즉,

$$L(M) = \{w \in \Sigma^* : (q_0, w, \#) \vdash_M^* (p, \epsilon, u), p \in F, u \in \Gamma^*\}.$$

예제 18 $M = (\{q_1, q_2, q_3\}, \{0, 1\}, \Gamma = \{0, \#\}, \Delta, q_1, \{q_3\})$*이고* Δ*는 아래의 전이를 가지고 있다.*

$$((q_1, 0, \epsilon), (q_1, 0))$$

$((q_1, \epsilon, \#), (q_3, \#))$

$((q_1, 1, 0), (q_2, \epsilon))$

$((q_2, 1, 0), (q_2, \epsilon))$

$((q_2, \epsilon, \#), (q_3, \#))$

M*을 그림으로 나타내면 그림 9와 같다.* $L(M) = \{0^n 1^n : n \geq 0\}$. *입력 스트링이* 01*이면* M*의 전이과정은 다음과 같다.*

$$(q_1, 01, \#) \vdash (q_1, 1, 0\#) \vdash (q_2, \epsilon, \#) \vdash (q_3, \epsilon, \#)$$

$(q_1, 01, \#) \vdash (q_3, 01, \#)$ *전이는 더 이상 진행하지 못한다.*

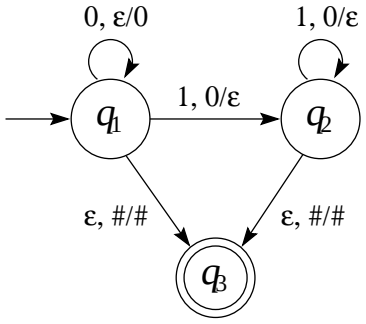

그림 9:

예제 19 $\{wcw^R : w \in \{a, b\}^*\}$*을 받아들이는 PA를 구하라. 그림 10.*

예제 20 $\{ww^R : w \in \{a, b\}^*\}$*을 받아들이는 PA를 구하라. 그림 10에서* $((q_1, c, \epsilon), (q_2, \epsilon))$*을* $((q_1, \epsilon, \epsilon), (q_2, \epsilon))$*으로 바꾼다.*

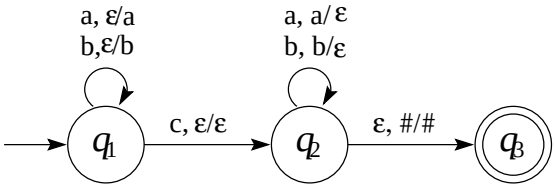

그림 10:

예제 21 $\{a^n b^m c^k : n + m = k\}$*를 받아들이는 PA를 구하라.*

예제 22 $\{0^n 1^m : n \leq m \leq 2n\}$*를 받아들이는 PA를 구하라.*

예제 23 $\{vw : v, w \in \{0,1\}^*, w \neq v^R, |v| = |w|\}$*를 받아들이는 PA를 구하라.*

제 6 절 내리누름 오토마타와 문맥무관 언어

정리 5 *문맥무관 언어 L에 대하여, $L = L(M)$인 내리누름 오토마타 M이 존재한다.*

증명. L을 생성하는 Greibach 표준형의 문맥무관 문법을 $G = (V, \Sigma, S, P)$라 하자. G의 좌측 유도를 흉내내는 PA M을 만들고자 한다. G가 Greibach 표준형이므로 좌측 유도에서 문장 형태는 xu ($x \in \Sigma^*$, $u \in V^*$) 이다. 이 순간 M은 입력 스트링에서 x를 읽고, 스택에 $u\#$를 가지고자 한다.

G와 동등한 PA M은 다음과 같다 (그림 11).

$$M = (\{p, q, r\}, \Sigma, V \cup \{\#\}, \Delta, p, \{r\})$$

1. $((p, \epsilon, \#), (q, S\#)) \in \Delta$
2. 모든 $A \to ay$에 대하여 $((q, a, A), (q, y)) \in \Delta$
3. $((q, \epsilon, \#), (r, \#)) \in \Delta$.

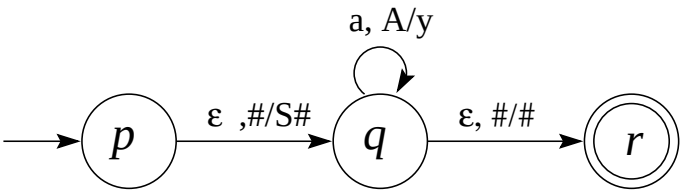

그림 11:

$$S \stackrel{*}{\Rightarrow} xu\text{인 경우에만 } (q, x, S\#) \vdash_M^* (q, \epsilon, u\#) \qquad (1)$$

임을 귀납법으로 보이고자 한다.

(1)의 $\Rightarrow$ 경우를 증명하자. 유도 횟수가 0일 때는 $x = \epsilon, u = S$이므로 당연히 성립한다. 유도 횟수가 i 미만일 때는 (1)가 성립한다고 가정한다. 유도 횟수가 i일 때 $x = x_1 a$라고 하면

$$S \stackrel{*}{\Rightarrow} x_1 A u_1 \Rightarrow x_1 a u_2 u_1$$

이고 $u = u_2 u_1$이다. 귀납법 가정에 의해
$(q, x_1 a, S\#) \vdash_M^* (q, a, Au_1\#)$이다. $A \to au_2$가 생성 규칙이므로
$((q, a, A), (q, u_2)) \in \Delta$이다. 따라서,

$$(q, x_1 a, S\#) \vdash_M^* (q, a, Au_1\#) \vdash_M (q, \epsilon, u_2 u_1\#).$$

마찬가지로 (1)의 $\Leftarrow$ 경우를 증명할 수 있다. (1)에 의해서
$S \stackrel{*}{\Rightarrow} w$인 경우에만 $(q, w, S\#) \vdash_M^* (q, \epsilon, \#)$이고 그래서

$$(p, w, \#) \vdash_M (q, w, S\#) \vdash_M^* (q, \epsilon, \#) \vdash_M (r, \epsilon, \#)$$

이다. 즉 $w \in L(G)$인 경우에만 $w \in L(M)$이다.

또 다른 증명: L을 생성하는 문맥무관 문법을 $G = (V, \Sigma, S, P)$라 하자. G를 흉내내는 PA는 다음과 같다.

$$(\{p, q, r\}, \Sigma, V \cup \Sigma \cup \{\#\}, \Delta, p, \{r\})$$

1. $((p, \epsilon, \#), (q, S\#)) \in \Delta$
2. 모든 $A \to x$에 대하여 $((q, \epsilon, A), (q, x)) \in \Delta$

3. 모든 $a \in \Sigma$에 대하여 $((q, a, a), (q, \epsilon)) \in \Delta$
4. $((q, \epsilon, \#), (r, \#)) \in \Delta$.

이 경우 문법 G가 Greibach 표준형일 필요가 없다. □

예제 24 *Greibach 표준형인 문법*

$$\begin{aligned} S &\rightarrow 0SA \mid 0A \\ A &\rightarrow 1 \end{aligned}$$

과 동등한 내리누름 오토마타를 구하라. (그림 12)

$$S \Rightarrow 0SA \Rightarrow 00AA \Rightarrow 001A \Rightarrow 0011$$

유도과정에 해당되는 PA의 전이를 설명한다.

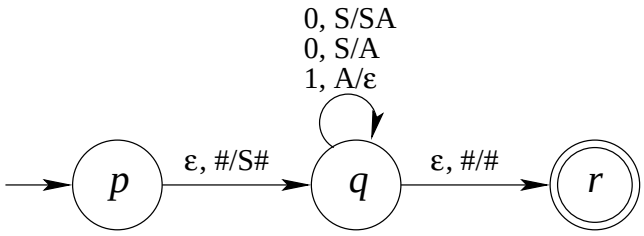

그림 12:

예제 25 *문법*

$$S \rightarrow aSb \mid bSa \mid SS \mid \epsilon$$

과 동등한 내리누름 오토마타를 구하라. (그림 13)

$$S \Rightarrow aSb \Rightarrow abSab \Rightarrow abab$$

유도과정에 해당되는 PA의 전이를 설명한다.

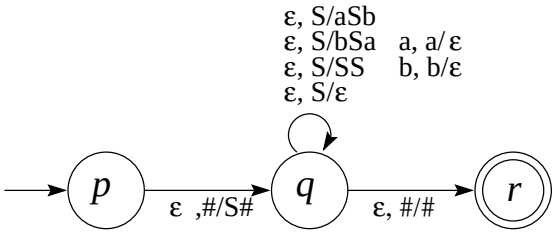

그림 13:

정리 6 *내리누름 오토마타 M에 대하여, $L(M)$은 문맥무관 언어이다.*

증명. 일반성을 잃지 않고 $M = (Q, \Sigma, \Gamma, \Delta, q_0, F)$에 대하여 다음과 같이 가정한다. $a, b \in \Sigma \cup \{\epsilon\}$, $A \in \Gamma$라고 하자.

- M은 최종상태가 q_f 한 개만 있고, 끝날 때에 스택에 #만 남기고 q_f로 간다. (그렇지 않은 경우에는 스택에서 pop하는 전이를 더해주면 된다.)

- M의 전이는 $((p, a, \epsilon), (q, A))$ (push) 또는 $((p, a, A), (q, \epsilon))$ (pop) 형태만을 갖는다. (일반적인 전이는 push와 pop의 연산들로 바꿀 수 있다.)

주요 개념은 M이 p 상태에서 시작하여 동일한 스택으로 (스택에 push한 후 pop할 수는 있지만 아래쪽으로는 내려가지 않고) q 상태에 도달하는 동안 입력에서 x를 읽으면, 문법에서는 $\langle pq \rangle \stackrel{*}{\Rightarrow} x$이 되도록 하는 것이다. (그림 14)

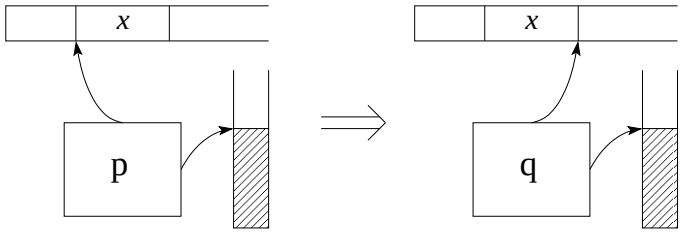

그림 14:

M이 x를 읽는 동안 스택의 변화는 다음 두 가지중 하나이다.

(a) 처음에 push된 글자가 마지막에 pop되는 경우: 이 경우를 $\langle pq\rangle \to a\langle rs\rangle b$로 흉내내려고 한다. 여기에서 a와 r은 처음 전이 때 읽은 글자와 다음 상태이고, b와 s는 마지막 전이 때 읽은 글자와 이전 상태이다.

(b) 처음에 push된 글자가 중간에 pop되는 경우: 이 경우를 $\langle pq\rangle \to \langle pr\rangle\langle rq\rangle$로 흉내내려고 한다. 여기에서 r은 처음에 push된 글자가 pop되었을 때의 상태이다.

M과 동등한 문맥무관 문법 $G = (V, \Sigma, S, P)$는 다음과 같다.

1. V는 $p, q \in Q$에 대하여 $\langle pq\rangle$형태의 변수들로 구성된다.
2. S는 $\langle q_0 q_f\rangle$이다.
3. $((p, a, \epsilon), (r, A)), ((s, b, A), (q, \epsilon)) \in \Delta$에 대하여 $\langle pq\rangle \to a\langle rs\rangle b$를 만든다.
4. 모든 $p, q, r \in Q$에 대하여 $\langle pq\rangle \to \langle pr\rangle\langle rq\rangle$를 만든다.
5. 모든 $q \in Q$에 대하여 $\langle qq\rangle \to \epsilon$을 만든다.

$$(p, x, \epsilon) \vdash_M^* (q, \epsilon, \epsilon)\text{인 경우에만 } \langle pq\rangle \stackrel{*}{\Rightarrow} x \qquad (2)$$

임을 귀납법으로 보이고자 한다.

(2)의 $\Rightarrow$ 경우를 증명하자. 전이 횟수가 0인 경우에는 $p = q$, $x = \epsilon$이고, 규칙 5에 의해 $\langle pp\rangle \to \epsilon$이므로 (2)이 성립한다. 전이 횟수가 i 미만일 때 (2)이 성립한다고 가정하자.

전이 횟수가 $i \geq 1$일 때 스택에서 (a)와 (b)중 하나가 발생한다. (a)가 발생하는 경우, A를 처음에 스택에 push된 글자, a와 r은 처음 전이 때 읽은 글자와 다음 상태, b와 s는 마지막 전이 때 읽은 글자와 이전 상태라고 하자. 즉 처음 전이가 $((p, a, \epsilon), (r, A))$, 마지막 전이가 $((s, b, A), (q, \epsilon))$이다. 규칙 3에 의해 G는 $\langle pq\rangle \to a\langle rs\rangle b$를 갖는다. $x = ayb$라고 하자. 귀납법 가정에 의해 $(r, y, \epsilon) \vdash^* (s, \epsilon, \epsilon)$ 이므로 $\langle rs\rangle \stackrel{*}{\Rightarrow} y$이다. 따라서 $\langle pq\rangle \Rightarrow a\langle rs\rangle b \stackrel{*}{\Rightarrow} x$.

(b)가 발생하는 경우, r을 처음에 push된 글자가 pop되었을 때의 상태라고 하고, y와 z를 각각 r 이전과 이후에 읽은 입력 스트링이라고 하자. 규칙 4에 의해 G는 $\langle pq\rangle \to \langle pr\rangle\langle rq\rangle$를 갖는다. 귀납법 가정

에 의해 $\langle pr \rangle \stackrel{*}{\Rightarrow} y$이고 $\langle rq \rangle \stackrel{*}{\Rightarrow} z$이다. 따라서 $\langle pq \rangle \Rightarrow \langle pr \rangle \langle rq \rangle \stackrel{*}{\Rightarrow} x$.

마찬가지로 (2)의 $\Leftarrow$ 경우를 증명할 수 있다. (2)에 의해서

$$\langle q_0 q_f \rangle \stackrel{*}{\Rightarrow} w\text{인 경우에만 } (q_0, w, \#) \vdash^* (q_f, \epsilon, \#).$$

즉 $w \in L(G)$인 경우에만 $w \in L(M)$이다. $\square$

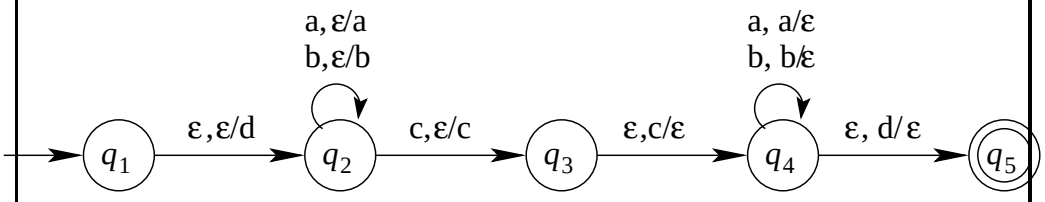

그림 15:

예제 26 *그림 15의 PA와 동등한 문맥무관 문법을 구하라.*

$$\begin{aligned}
\langle q_1 q_1 \rangle &\rightarrow \epsilon \\
&\cdots \\
\langle q_5 q_5 \rangle &\rightarrow \epsilon \\
\\
\langle q_1 q_3 \rangle &\rightarrow \langle q_1 q_2 \rangle \langle q_2 q_3 \rangle \\
&\cdots \\
\langle q_1 q_5 \rangle &\rightarrow \langle q_1 q_3 \rangle \langle q_3 q_5 \rangle \\
\\
\langle q_1 q_5 \rangle &\rightarrow \langle q_2 q_4 \rangle \\
\langle q_2 q_4 \rangle &\rightarrow c \langle q_3 q_3 \rangle \\
\langle q_2 q_4 \rangle &\rightarrow a \langle q_2 q_4 \rangle a \\
\langle q_2 q_4 \rangle &\rightarrow b \langle q_2 q_4 \rangle b
\end{aligned}$$

이 문법으로 스트링 $abcba$를 생성하는 과정:

$$\begin{aligned}\langle q_1 q_5\rangle &\Rightarrow \langle q_2 q_4\rangle \\ &\overset{*}{\Rightarrow} ab\langle q_2 q_4\rangle ba \\ &\Rightarrow abc\langle q_3 q_3\rangle ba \\ &\Rightarrow abcba.\end{aligned}$$

제 7 절 문맥무관 언어의 성질

어떤 언어가 문맥무관 언어가 아님을 보이기 위해서는 다음 정리를 사용한다.

정리 7 (펌프 정리) *L을 문맥무관 언어라고 하면, 다음을 만족하는 양의 정수 t가 존재한다. 길이가 t 이상인 임의의 스트링 $w \in L$는 $w = uvxyz$로 표현되며 여기서*

1. $|vxy| \le t$,

2. $|vy| \ge 1$,
3. *모든 $i \ge 0$에 대하여 $uv^i xy^i z \in L$ 이다.*

증명. $L - \{\epsilon\}$를 생성하는 Chomsky 표준형의 문법을 $G = (V, \Sigma, S, P)$라고 하자. G가 Chomsky 표준형이므로 $w \in L(G)$의 파스 트리의 높이가 i이면 $|w| \le 2^{i-1}$임을 귀납법으로 보일 수 있다.

$|V| = k$라고 하면, t는 2^k이다. $w \in L(G)$이고 $|w| \ge t$이면, w의 파스 트리의 높이가 $k+1$ 이상이다. 루트에서 가장 긴 경로를 P라고 하면, P의 길이는 $k+1$ 이상이고 그 경로 상에 $k+1$ 이상의 변수가 나타난다. G는 k 개의 변수만을 가지므로 P를 리프에서부터 올라가면, 반복되는 변수가 존재한다. 처음 반복되는 변수를 A라 하고 A 이름의 노드를 p, q (p가 q의 아래쪽) 라고 하면, 리프부터 q까지의 길이는 $k+1$ 이하이다. 그림 16를 보라. P가 가장 긴 경로이므로 q의 높이는 $k+1$ 이하이다. 따라서 q를 루트로 갖는 부분 트리의 열매 스트링 vxy의 길이는 2^k 이하이다 (즉 $|vxy| \le t$). 여기에서 x는 p를 루트로 갖는 부분 트리의 열매 스트링이다. 즉 w는

$uvxyz$로 표시된다. 또한 q에서 사용된 생성규칙은 $A \to BC$ 형태이므로 v, y 둘 중의 하나는 ϵ가 될 수 없다 (즉 $|vy| \geq 1$).

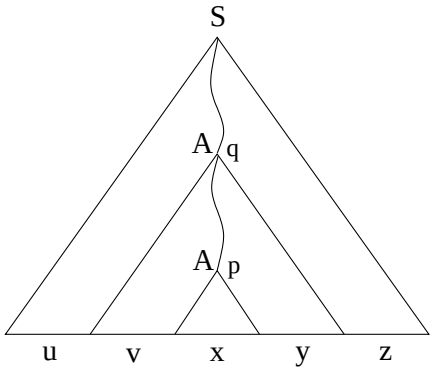

그림 16:

q와 p에서 시작하는 유도 과정은

$$A \stackrel{*}{\Rightarrow} vAy \text{와 } A \stackrel{*}{\Rightarrow} x$$

이므로 모든 $i \geq 0$에 대하여 $A \stackrel{*}{\Rightarrow} v^i x y^i$이다. □

예제 27 $L = \{a^n b^n c^n : n \geq 0\}$는 문맥무관 언어가 아님을 증명하라.

L이 문맥무관 언어라고 가정하자. 그러면 정리 7를 만족하는 t가 존재한다. 스트링 $w = a^t b^t c^t$를 고려하자. $|vxy| \leq t$이므로 vy는 a, b, c 세 글자를 모두 포함할 수 없다. 따라서 uv^2xy^2z는 같은 수의 a, b, c를 가질 수 없다.

예제 28 $L = \{ww : w \in \{a, b\}^*\}$는 문맥무관 언어가 아니다.

L이 문맥무관 언어라고 가정하고 $w = a^t b^t a^t b^t$를 선택한다. vy가 어디에 위치하든지 $uxz \notin L$이다.

예제 29 $L = \{a^n : n$은 소수$\}$은 문맥무관 언어가 아니다.

L이 문맥무관 언어라고 가정하면 정리 7를 만족하는 t가 존재한다. p를 t보다 큰 소수라고 하자. 즉 $a^p \in L$. $q = |vy|$, $r = |uxz|$라고 하

면, $q \geq 1$, $r \geq 1$이다. $i = q + r + 1$을 선택하면
$iq + r = (q + r)(q + 1)$이고 $q + r \geq 2$, $q + 1 \geq 2$이므로
$uv^i xy^i z \notin L$이다.

정리 8 *문맥무관 언어 종류는 (1) 합집합, (2) 접합, (3) Kleene 스타 연산에 대하여 닫혀 있다.*

증명. $G_1 = (V_1, \Sigma_1, S_1, P_1)$과 $G_2 = (V_2, \Sigma_2, S_2, P_2)$를 문맥무관 문법이라고 하자. 일반성을 잃지 않고 V_1와 V_2는 서로 교차하지 않는 집합이라고 하자.
(1) 새로운 변수 S를 사용하여

$$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, S, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\})$$

가 $L(G_1) \cup L(G_2)$를 생성한다.
(2) 합집합의 경우와 마찬가지이나, $S \to S_1 \mid S_2$ 대신 $S \to S_1 S_2$를 첨가한다.
(3)

$$G = (V_1 \cup \{S\}, \Sigma_1, S, P_1 \cup \{S \to SS_1 \mid \epsilon\})$$

이 $L(G_1)^*$를 생성한다. $\square$

정리 9 *문맥무관 언어 종류는 (1) 교집합, (2) 여집합 연산에 대하여 닫혀 있지 않다.*

증명. (1) $L_1 = \{a^n b^n c^m : n, m \geq 0\}$, $L_2 = \{a^n b^m c^m : n, m \geq 0\}$라고 하자. L_1과 L_2는 문맥무관 언어이다. 즉 L_1은 다음 문법에 의해 생성된다.

$$\begin{aligned} S &\to AB \\ A &\to aAb \mid \epsilon \\ B &\to cB \mid \epsilon \end{aligned}$$

그러나 $L_1 \cap L_2$는 문맥무관 언어가 아니다.
(2)

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

이므로 여집합에 대해 닫혀 있으면 교집합에 대해서도 닫혀 있어야 된다. □

정리 10 *L이 문맥무관 언어이고 R이 정규언어이면, $L \cap R$은 문맥무관 언어이다.*

증명. $M_1 = (Q_1, \Sigma, \Gamma, \Delta_1, q_1, F_1)$을 L을 받아들이는 PA,
$M_2 = (Q_2, \Sigma, \delta, q_2, F_2)$를 R을 받아들이는 DFA라고 하자. 주요 개념은 M_1과 M_2를 동시에 돌리는 PA M을 만드는 것이다. 구체적으로

$$M = (Q_1 \times Q_2, \Sigma, \Gamma, \Delta, (q_1, q_2), F_1 \times F_2)$$

인데, 전이 Δ는 다음과 같다.

- $((p, a, A), (p', u)) \in \Delta_1$이고 $\delta(q, a) = q'$이면, $(((p, q), a, A), ((p', q'), u) \in \Delta$이다.
- $((p, \epsilon, A), (p', u)) \in \Delta_1$이면, 모든 $q \in Q_2$에 대하여 $((p, q), \epsilon, A), ((p', q), u)) \in \Delta$이다.

$w \in L \cap R$인 경우에만 $w \in L(M)$임을 보일 수 있다. □

예제 30 (소속문제) *CFG $G = (V, \Sigma, S, P)$와 $w \in \Sigma^*$가 주어졌을 때, w가 $L(G)$의 원소인지 아닌지를 결정하라. 이 과정에서 $w \in L(G)$인 경우 파스 트리를 만들게 되므로 소속문제를 푸는 과정을 파싱(parsing)이라고 부른다.*

CYK (Cocke, Younger, Kasami) 알고리즘은 동적 계획법(dynamic programming)의 일종으로 다음과 같다.

G가 Chomsky 표준형이고 $w = a_1 a_2 \cdots a_n$이라고 하자.
$V_{ij} = \{A \in V : A \stackrel{*}{\Rightarrow} a_i \cdots a_j\}$라고 하자. 그러면, $S \in V_{1n}$인 경우에만 $w \in L(G)$이다. 따라서 V_{ij}를 $d = j - i$가 작은 값에서 큰 값으로 가면서 계산하면 된다. G에 생성 규칙이 상수 개 있으면, CYK 알고리즘은 $O(n^3)$ 시간이 걸린다.

```
for i = 1 to n do
    V_ii = {A : A → a_i}
od
for d = 1 to n − 1 do
    for i = 1 to n − d do
        j = i + d
        V_ij = ∅
        for k = i to j − 1 do
            V_ij = V_ij ∪ {A : A → BC, B ∈ V_ik, C ∈ V_{k+1,j}}
        od
    od
od
```

그림 17:

예제 31 *다음 문맥무관 문법이 baaba를 생성하는지 CYK 알고리즘을 사용하여 확인하라.*

$$\begin{aligned} S &\rightarrow AB \mid BC \\ A &\rightarrow BA \mid a \\ B &\rightarrow CC \mid b \\ C &\rightarrow AB \mid a \end{aligned}$$

CYK 알고리즘이 만드는 표는 다음과 같다.

$i\backslash j$	1	2	3	4	5
1	B	SA	$\emptyset$	$\emptyset$	SAC
2		AC	B	B	SAC
3			AC	SC	B
4				B	SA
5					AC

$S \in V_{15}$이므로 $baaba$는 이 문법에 의해 생성된다.

제 8 절 결정 내리누름 오토마타

내리누름 오토마타가 한 시점에 두 개 이상의 전이를 가질 수 없으면 이를 결정 내리누름 오토마타라고 부른다. 결정 내리누름 오토마타(deterministic pushdown automata)를 줄여서 DPA로 부르기로 하자. DPA가 입력이나 스택을 읽지 않고 전이할 수 있도록 하는 것이 편리하므로 전이함수 δ를 다음 조건을 만족하는 $Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\})$에서 $Q \times \Gamma^*$로의 부분함수(partial function)로 정의한다.

- $\delta(q, \epsilon, A)$가 정의되면, 모든 $a \in \Sigma$에 대하여 $\delta(q, a, A)$가 정의되지 않는다.
- $\delta(q, a, \epsilon)$이 정의되면, 모든 $A \in \Gamma$에 대하여 $\delta(q, a, A)$가 정의되지 않는다.

또한 DPA가 입력의 끝을 알 수 있도록 입력 다음 칸에 공백 글자 #가 들어 있다고 가정한다. DPA가 받아들이는 언어를 결정 문맥무관 언어라고 부른다.

예제 32 $\{0^n 1^n : n \geq 0\}$을 받아들이는 *DPA*는 그림 *18*과 같다.

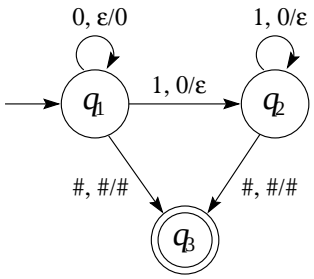

그림 18:

정리 11 *결정 문맥무관 언어는 여집합에 대하여 닫혀 있다.*

증명. (설명) DPA에서는 최종상태와 그렇지 않은 상태를 맞바꿈으로 여집합을 받아들이는 DPA를 만들 수 없다. 왜냐하면 계속 입력을 읽지 않고 무한루프에 들어가는 경우가 있기 때문이다. 따라서 이 증명을 위해서는 주어진 DPA를 매번 한 글자씩 입력을 읽는 DPA로 바꾼 다음에 최종상태와 그렇지 않은 상태를 바꿔야 한다. □

정리 12 *결정 문맥무관 언어 종류는 문맥무관 언어 종류의 진부분 집합이다.*

증명. $L = \{a^i b^j c^k : i \neq j \text{ 또는 } j \neq k\}$를 고려하자. L은 문맥무관 언어이다.

L이 결정 문맥무관 언어라고 가정하자. 그러면 $\overline{L}$도 결정 문맥무관이고 따라서 문맥무관이다. $\overline{L} \cap a^*b^*c^*$는 문맥무관이 되는데, 이것이 $\{a^n b^n c^n : n \geq 0\}$이므로 모순이다. □

제 9 절 하향 파싱

입력을 받아들여서 파스 트리를 만드는 과정을 파싱(parsing)이라고 한다. CYK 알고리즘을 사용하면 파싱을 할 수 있으나, 큰 프로그램을 파싱할 경우 너무 많은 시간이 걸린다. 일반적으로 컴파일러는 $O(n)$ 시간에 파싱하는 알고리즘을 필요로 한다.

예제 33 *예제 24의 문법*

$$\begin{aligned} S &\rightarrow 0SA \mid 0A \\ A &\rightarrow 1 \end{aligned}$$

과 스트링 0011이 주어졌을 때, 파싱을 해보자.

스트링의 첫 번째 글자가 0이므로 시작변수에서 $S \rightarrow 0SA$ 또는 $S \rightarrow 0A$를 적용해야 한다. 어느 생성규칙을 적용해야 할지 알 수 없지만 $S \rightarrow 0SA$를 적용해보면, 유도과정

$$S \Rightarrow 0SA$$

을 얻는다. 변수가 S이고 두 번째 글자도 0이므로 앞의 경우와 마찬가지인데, $S \to 0A$를 적용해보면

$$S \stackrel{*}{\Rightarrow} 00AA$$

을 얻는다. 변수가 A이고 세 번째 글자가 1이므로 $A \to 1$을 적용해야 한다. 네 번째 글자의 경우도 마찬가지이다. (참고로, 입력 글자를 두 개 읽으면 어느 생성규칙을 적용해야 할지 바로 알 수 있다. 현재 변수가 S일 때, 00을 읽으면 $S \to 0SA$를, 01을 읽으면 $S \to 0A$를 적용하면 된다.)

현재 변수 A와 글자 a에 대해서 $A \to a\ldots$ 형태의 생성규칙이 하나만 있으면 유도과정을 $O(n)$에 완성할 수 있다. 즉, $O(n)$ 시간에 파싱할 수 있다.

하향 파싱(topdown parsing)은 위의 개념을 일반화시킨 것이다. 하향 파싱은 입력을 왼쪽에서 오른쪽으로 읽으면서 파스 트리를 위에서 아래로 만들어 가는 파싱이다. 파스 트리를 위에서 아래로 만들어 가는 과정은 좌측 유도를 만드는 것과 동일하다.

다음 문법 G에 대하여 $w \in L(G)$를 파싱하는 DPA를 구하라.

$$\begin{aligned} E &\to E + T \mid T \\ T &\to T * F \mid F \\ F &\to \texttt{id} \end{aligned}$$

여기에서 `id`는 하나의 글자로 간주한다.

하향 파싱이 용이하도록 다음 두 규칙을 사용하여 문법을 바꾼다.

1. $A \to Ax$ 형태의 생성규칙을 없앤다. A가 좌변에 나타나는 모든 생성규칙이

   $$\begin{aligned} A &\to Ax_1 \mid \cdots \mid Ax_r \\ A &\to y_1 \mid \cdots \mid y_t \end{aligned}$$

   이면, 이를

   $$\begin{aligned} A &\to y_1A' \mid \cdots \mid y_tA' \\ A' &\to x_1A' \mid \cdots \mid x_rA' \mid \epsilon \end{aligned}$$

로 바꾼다.

2. 생성규칙의 우변이 같은 스트링으로 시작되지 않도록 고친다.

$$A \to xy_1 \mid \cdots \mid xy_r$$

$(x \neq \epsilon, r \geq 2)$이면, 이를

$$\begin{aligned} A &\to xA' \\ A' &\to y_1 \mid \cdots \mid y_r \end{aligned}$$

로 바꾼다.

문법 G에 규칙 1을 적용하면

$$\begin{aligned} E &\to TE' \\ E' &\to +TE' \mid \epsilon \\ T &\to FT' \\ T' &\to *FT' \mid \epsilon \\ F &\to \texttt{id} \end{aligned}$$

예제 34 *위의 문법으로* $\texttt{id} + \texttt{id}$*를 생성하는 과정은 다음과 같다.*

$$\begin{aligned} E &\Rightarrow TE' \\ &\Rightarrow T + TE' \\ &\Rightarrow FT' + TE' \\ &\overset{*}{\Rightarrow} F + T \\ &\overset{*}{\Rightarrow} F + F \\ &\overset{*}{\Rightarrow} \texttt{id} + \texttt{id} \end{aligned}$$

DPA가 입력 글자를 하나 읽어서 현재 변수에서 어떤 생성규칙을 적용할지 결정하도록 만들려고 한다. 이를 위해서 변수 A와 입력 글자 a가 주어지면 적용할 생성규칙을 파싱 표 $M(A, a)$에 저장한다. 각 생성규칙 $A \to x$에 대하여

1. x에서 유도되는 첫 번째 글자가 a이면 (즉 $x \overset{*}{\Rightarrow} ay$) $M(A, a)$에 $A \to x$를 넣는다.
2. $x \overset{*}{\Rightarrow} \epsilon$인 경우, A 다음에 나올 수 있는 글자가 a이면 $M(A, a)$에

$A \to x$를 넣는다.

G의 생성규칙을 고려하면

- $E \to TE'$: T는 F를 거쳐 id를 생성하므로 $M(E, \texttt{id})$에 이 생성규칙을 넣는다.
- $E' \to \epsilon$: $E \to TE'$에 의해 E'은 입력의 끝에 위치하므로 그 다음에 #이 올 수 있어서 $M(E', \#)$에 넣는다.
- $T' \to \epsilon$: T'도 입력의 끝에 위치할 수 있으므로 $M(T', \#)$에 넣고 또한 T' 다음에 +이 올 수 있으므로 $M(T', +)$에 넣는다.

이와 같이 파싱 표를 만들면 그림 19과 같다. 파싱 표를 만드는 자세한 방법은 [**?**]를 참고하라.

	id	+	*	#
E	$E \to TE'$			
E'		$E' \to +TE'$		$E' \to \epsilon$
T	$T \to FT'$			
T'		$T' \to \epsilon$	$T' \to *FT'$	$T' \to \epsilon$
F	$F \to \texttt{id}$			

그림 19:

이제 DPA는 입력 글자를 읽은 후 위의 파싱 표에 의해 생성규칙을 적용한다 (그림 20).

Figure omitted

그림 20:

입력 id + id * id#에 대한 DPA의 동작과정이 그림 21에 있다. 상태나 입력, 스택이 변하지 않은 경우에는 –로 표시하였다. 이와 같이 만들어진 파스 트리는 그림 22에 있다.

Table omitted

그림 21:

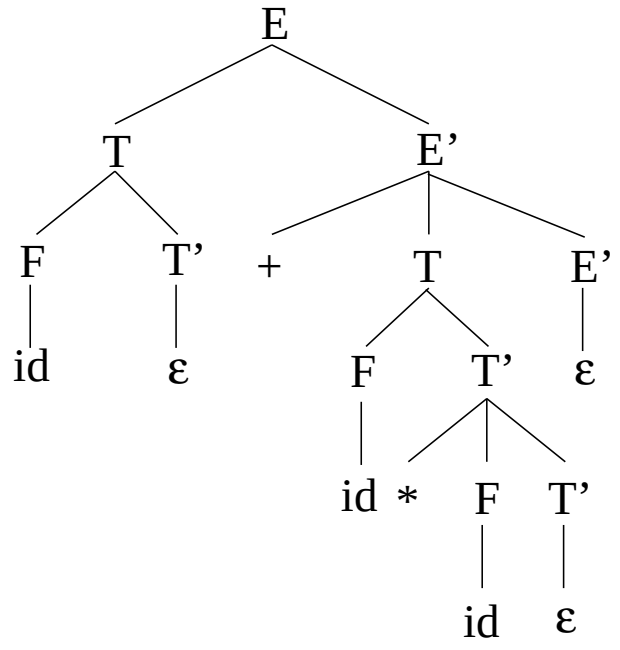

그림 22:

`if`문을 나타내는 문법을 간략히 하면 다음과 같다.

$$\begin{aligned} S &\rightarrow iCtSeS \mid iCtS \mid x \\ C &\rightarrow a \end{aligned}$$

여기에서 i, t, e는 각각 `if`, `then`, `else`를 나타낸다.

규칙 2를 적용하면

$$\begin{aligned} S &\rightarrow iCtSS' \mid x \\ S' &\rightarrow eS \mid \epsilon \\ C &\rightarrow a \end{aligned}$$

를 얻는다. 이제 파싱 표를 구하자.

- $S' \rightarrow eS$는 $M(S', e)$에 들어가야 된다.
- $S' \rightarrow \epsilon$을 고려하면, $iCtSS'$에서 $S \rightarrow iCtSS'$와 $S' \rightarrow eS$를 사용하면 S' 다음에 e가 나오므로 $M(S', e)$에 $S' \rightarrow \epsilon$이 들어가야 된다.

원래 문법이 애매하기 때문에 $M(S', e)$에 두 개의 생성규칙이 들어가게 된다. 이 경우 두 생성규칙 중 $S' \rightarrow eS$를 선택하면 `else`를 가장 가까운 `if`에 붙이는 결과가 된다. 파싱 표를 완성하면 그림 23과 같다.

	i	e	x	a	$\#$
S	$S \rightarrow iCtSS'$		$S \rightarrow x$		
S'		$S' \rightarrow eS$			$S' \rightarrow \epsilon$
C				$C \rightarrow a$	

그림 23: