Chomsky Normal Form

(개요) Context-free 문법 중 가장 간략한(normal) 형태(form)를 가진 문법[1)]

(정의) **쓸모 있게 줄여진(reduced)** context-free 문법 $G = (N, T, P, S)$의 문법규칙 P가,
$S \to \epsilon$이외에는 모두
$A \to BC$ 또는 $A \to a$인 형태(단 $A, B, C \in N$이고 $a \in T$이다)만 가지는
문법 G를 Chomsky Normal Form(CNF)이라 부른다.

(정의 1) $A \to \epsilon$인 형태의 규칙을 ϵ-production이라고 부른다.

문법규칙에서 ϵ-production은 우변의 길이가 0이므로, $S \to \epsilon$이외에는 CNF에는 맞지 않는다. 따라서 ϵ-production을 없애는 작업이 필요하다.

(정의 1.1) $A \Rightarrow^* \epsilon$인 넌 터미널 A를 nullable 문자(symbol)이라고 부른다. 이를 구하는 알고리즘이 쓸모 있는 문자에서 terminate를 구하는 알고리즘을 약간 변형하면 된다.

(귀납정의 1.1[2)]) Nullable 넌 터미널
(기본) $\forall A \to \epsilon \in P$: $A \subseteq$ Nullable
(귀납) $\forall A \to B_1 B_2 \cdots B_n \in P$:
$[\forall i \in \{1, \cdots, n\}: B_i \in N \wedge B_i \in$ Nullable$] \Rightarrow [A \subseteq$ Nullable$]$

(정의 1.2) $S \to \epsilon$이외에는 $A \to \epsilon$ 형태의 규칙이 없는 문법 G를 ϵ-free 문법이라 부른다.

(알고리즘 1.2) ϵ-free로 만들기
$\forall A \to X_1 X_2 \cdots X_n \in P$:
$\forall i \in \{1, \cdots, n\}$:
if $X_i \in$ Nullable $\to Z_i = (X_i \mid \epsilon)$
| $X_i \notin$ Nullable $\to Z_i = X_i$
fi
Replace $A \to X_1 X_2 \cdots X_n$ **to** $A \to Z_1 Z_2 \cdots Z_n$ **in** P.
/* **Replace** $A \to X_1 X_2 \cdots X_n \in P$ **to** $A \to Z_1 Z_2 \cdots Z_n \in P'$. */[3)]
$\forall A \to \epsilon \in P$ **where** $A \neq S$: **Remove** $A \to \epsilon$.

Chomsky's Normal Form에서 $S \to \epsilon$를 허용하는 것은 CNF으로 바꾸기 전의 문법 G가 ϵ를 **문장**으로 가지는 경우도 CNF가 허용하기 위함이다.

(알고리즘 1.3) ϵ-free로 만드는 이해하기 쉬운 알고리즘(2)

---

1) 다음 절에서 설명할 Pumping Lemma와 CYK 알고리즘에 증명에 긴요한 형태의 CFG이다.
2) 귀납정의 1.1은 Nullable을 귀하는 귀납 알고리즘이기도 하다.
3) 위와 아래 중 어느 것이 더 나을까?

Repeat

$\forall A \to \epsilon \in P$:

$\forall B \to \alpha A \gamma \in P$ **where** $\alpha, \gamma \in (N \cup T)^*$:

Replace A **to** $(A \mid \epsilon)$ **in** $B \to \alpha A \gamma \in P$

until no more **new** $B \to \epsilon$ is added

$\forall A \to \epsilon$ **where** $A \neq S$ **: Remove** $A \to \epsilon$.

위 알고리즘 1.3은 귀납정의 1.1의 Nullable을 구하는 과정이 없어졌음에 주의하라.

(정의 2) Context-free 문법 $G = (N, T, P, S)$에서 $A, B \in N$일 때, $A \to B$인 형태의 문법 규칙을 unit production이라 부른다.

Unit production은 넌 터미널 A의 우변이 다른 넌 터미널 B의 우변을 포함할 때, 규칙의 길이를 줄이기 위하여 사용되는 방법이다.

(예1: $G_{식_1}$)
$$E \to E + T \mid T \times F \mid (E) \mid a$$
$$T \to T \times F \mid (E) \mid a$$
$$F \to (E) \mid a$$

(예2: $G_{식_2}$)
$$E \to E + T \mid T$$
$$T \to T \times F \mid F$$
$$F \to (E) \mid a$$

$G_{식_1}$과 $G_{식_2}$는 같은 일을 하는, $L(G_{식_1}) = L(G_{식_2})$, 문법이지만, $G_{식_2}$는 $G_{식_1}$보다 **문법 길이가** $|P_{식_1}| = 16+12+8 = 36$에서 $|P_{식_2}| = 8+8+8 = 24$로 작아졌다. 그러나 파스나무(parse tree)나 유도과정(derivation sequence)를 살펴보면, $E \Rightarrow T$, $T \Rightarrow F$의 unit production의 유도가 더해져서, **나무의 높이가 더 높아지고**, **유도과정이 더 길어진**다. Unit production은 **장점**과 **단점**을 동시에 가지고 있는 표현 방식이다. 또한 unit production은 CNF에 맞지 않다 그래서 Unit production이 있는 문법에서 unit production을 제거하고 같은 일을 하는 문법으로 바꾸는 변환이 필요하다.

(정의 2.1) $A \Rightarrow^* B \in N$일 때 넌 터미널 (A, B)쌍을 unit pair라고 부른다.

(귀납정의 2.1) Unit pair 구하기
(기본) $\forall A \to B \in P$: $(A, B) \subseteq$ UnitPair
(귀납) $(A, B) \in$ UnitPair $\wedge$ $B \to C \in P$: $(A, C) \subseteq$ UnitPair

(정의 2.2) $A \to B$인 형태의 규칙이 없는 문법 G를 unit production free라 부른다.

(알고리즘 2.2) Unit production free로 만드는 알고리즘
$\forall (A, B) \in$ UnitPair:

$\forall B \to \beta \in P$ **where** $\beta \notin N$**:**
$A \to \beta \subseteq P$ /* Add $A \to \beta$ to P */
$\forall A \to B \in P$**: Remove** $A \to B$ **from** P.

(알고리즘 2.3) Unit production free로 만드는 이해하기 쉬운 다른 알고리즘
Repeat
$\forall A \to B \in P$**:**
$\forall B \to \alpha A \gamma \in P$ **where** $\alpha, \gamma \in (N \cup T)^*$**:**
Replace A **to** $(A \mid \epsilon)$ **in** $B \to \alpha A \gamma \in P$
until no more **new** $B \to \epsilon$ is added

$\forall \to \alpha A \gamma \in P$ **where** $\alpha, \gamma \in (N \cup T)^*$**:**
if $\forall A \to \epsilon \in P \to$
$\forall A \to \beta \in P$**: Replace** A **to** β **in** $B \to \alpha A \gamma \in P$
fi
$\forall A \to B \in P$**: Remove** $A \to B$ **from** P.

(알고리즘 3) 우변 길이가 2이상인 규칙에서 터미널 문자를 없애기
(1) 길이가 0인 경우는 $S \to \epsilon$뿐이다 따라서 변환이 필요 없다.
(2) 길이가 1인 경우,
$A \to a$, $a \in T$뿐이다.
넌 터미널 A의 이름을 A_a로 고쳐 $A_a \to a \in P$로 한다.
만일 $b \in T$에 대하여 $B \to b$인 규칙이 없다면, $A_b \to b \in P$를 더한다.
(3) 길이가 2이상인 경우,
규칙의 우변에 있는 모든 $a \in T$를 A_a로 바꾼다.

알고리즘 3이 끝나면 규칙 $\forall A \to \alpha \in P$(단 $\alpha \in \{\epsilon\} \cup T \cup N^*$)는 아래 세 가지 경우이다.
(1) $|\alpha| = 0$일 때, $S \to \epsilon$이다.
(2) $|\alpha| = 1$일 때, $\forall a \in T$: $A_a \to a \in P$.
(3) $|\alpha| \geq 2$일 때, $\forall A \to \alpha \in P$, $\alpha \in N^*$.
$|\alpha| = 0, 1, 2$는 CNF형태에 맞는다.

(알고리즘 4) 길이가 3이상인 규칙을 모두 길이 2인 규칙으로 바꾸기
$\forall A \to A_1 A_2 \cdots A_n \in P$, $n \geq 3$, $1 \leq \forall i \leq n$: $A_i \in N$.
새로운 넌 터미널 $B_1, B_2, \cdots, B_{n-2} \notin N$을 N'에 추가한 후에,
$A \to A_1 B_1$, $B_1 \to A_2 B_2$, $\cdots$, $B_i \to A_{i+1} B_{i+1}$, $\cdots$, $B_{n-3} \to A_{n-2} B_{n-2}$,
$B_{n-2} \to A_{n-1} A_n \in P'$.
$1 \leq \forall i < n-2$: $B_i \notin N$, $B_i \in N'$, $B_i \to A_{i+1} B_{i+1} \in P'$,
$B_{n-2} \to A_{n-1} A_n \in P'$.

규칙 1개 (크기 $n+1$)가 (n-3) + 1 = n-2 개로(크기 3(n-2) = 3n-6)

알고리즘 4가 끝나고 나면 규칙 $\forall A \rightarrow \alpha \in P$는 아래 세 가지 경우 뿐 이다.

(1) $|\alpha| = 0$일 때, $S \rightarrow \epsilon$이다.

(2) $|\alpha| = 1$일 때, $\forall a \in T$: $A_a \rightarrow a \in P$.

(3) $|\alpha| = 2$일 때, $\forall A \rightarrow BC \in P$, $A, B \in N$.

즉 Chomsky Normal Form이다.

알고리즘을 정리하면 다음과 같다. 단 아래 모든 경우, $N \subseteq N'$, $P \subseteq P'$이다.

(1) 알고리즘 1 Context-free 문법 $G = (N, T, P, S)$을 $L(G) = L(G')$이고 ϵ-free인 문법 $G' = (N', T, P', S)$으로 바꾼다.

(2) 알고리즘 2 Context-free 이고 ϵ-free인 문법 $G = (N, T, P, S)$을 $L(G) = L(G')$이고 unit production free인 문법 $G' = (N', T, P', S)$으로 바꾼다.

(3) 알고리즘 3 Context-free 이고 ϵ-free, unit production free인 문법 $G = (N, T, P, S)$을 $L(G) = L(G')$이고 우변의 길이가 1이면 터미널 문자이고, 우변의 길이가 2이상이면 넌 터미널 문자열인 문법 $G' = (N', T, P', S)$으로 바꾼다.

(4) 알고리즘 4 위 알고리즘 3의 결과 에 맞는 context-free 문법 $G = (N, T, P, S)$을 $L(G) = L(G')$ (1) $S \rightarrow \epsilon$ (2) $A \rightarrow a$, $a \in T$ (3) $A \rightarrow BC$, $A, B \in N$에 세 가지 형태만 가지는 Chomsky Normal Form 문법 $G' = (N', T, P', S)$으로 바꾼다.

(결론) 임의의 context-free 문법 $G = (N, T, P, S)$을 $L(G) = L(G')$이고

(1) $S \rightarrow \epsilon$,

(2) $A \rightarrow a$, $a \in T$,

(3) $A \rightarrow BC$, $A, B \in N$,

인 3가지 형태만 가지는 Chomsky Normal Form 문법 $G' = (N', T, P', S)$로 만든다.