Reduced grammar

(개요) 쓸모 있는(**useful**) 규칙(rule)과 문자(symbol)만으로 줄인(**reduced**) 문법

문법은 언어를 정의하는 것이므로 문법규칙이 언어(**문장**)을 **정의**하는데 **사용**되어야 **쓸모 있는**(**useful**)규칙이다.

(정의 1) 문법 $G = (N, T, P, S)$에서 임의의 규칙 $A \to \beta \in P$가 아래조건을 만족하면 **쓸모 있는**(**useful**) **규칙**이라고 한다.

$$S \Rightarrow^* xBz \Rightarrow^{B\to\beta} x\beta z \Rightarrow^* xyz,\ x, y, z \in T^*.$$

즉 문법규칙 $B\to\beta$가 **쓸모 있으려면**, (1) 시작문자 S로부터 문법규칙의 좌변 B를 유도(derive)할 수 있어야하고(**accessible**), (2) 문법규칙 $B\to\beta$의 좌변 B가 우변 β로 바뀐 뒤 ($\Rightarrow^{B\to\beta}$) 우변 β가 터미널 문자열($\beta\Rightarrow^* y,\ y\in T^*$)로 끝이 나야한다(**terminate**).

규칙의 우변 $\beta = Y_1 Y_2 \cdots Y_n (1 \le \forall i \le n:\ Y_i \in N \cup T)$가 끝나려면 규칙의 우변에 있는 문자, Y_1, Y_2, $\cdots$, Y_n이 **모두** 끝나야한다.$(1 \le \forall i \le n:\ Y_i \in N \cup T:\ Y_i \Rightarrow^* y_i \in T^*)$. 문법규칙은 좌변과 우변 모두 **문자**로 이루어져 있으므로, 규칙의 **쓸모**를 문자의 **쓸모**로 바꾸어 아래와 같이 정의한다. 모든 문자가 끝난다면, 유도조건 $S \Rightarrow^* xBz,\ x, z \in T^*$는 $S \Rightarrow^* \alpha A\beta,\ \alpha, \beta \in (N \cup T)^*$로 **쉬워**(weaker)진다.

(정의 2) 문법 $G = (N, T, P, S)$에서 임의의 문자 $X \in N \cup T$가
(조건 1) Terminate하고: $X \Rightarrow^* x,\ x \in T^*$,
(조건 2) Accessible하면: $S \Rightarrow^* \alpha X\gamma,\ \alpha,\gamma \in (N\cup T)^*$,
문자 X는 **쓸모 있다**고 정의한다.

(정의 2)에서 (2) 문자 X의 accessible 검사에서 좌 문맥(left context) α나 우 문맥(right context) γ가 일반적인 문자열($(N \cup T)^*$)로 **쉬워**졌으므로, terminate하지 않는 α나 γ가 끝나지 않는 문자를 가지고 있으면, 문자 X는 accessible하여도 terminate하지 않으므로, (1)에 terminate 검사가 먼저 이루어져 terminate하지 않는 문자를 제거한 후에 (2)에 accessible 검사가 이루어져야 함에 주의하여야한다[1].

이제는 문법을 쓸모 있게 줄이기 위하여, (정의 2)의 (조건 1) terminate와 (조건 2) accessible을 구할 차례이다.

(조건 1) 끝나는(terminating) 문자($N \cup T$)

$$X \Rightarrow^* x,\ x \in T^*.$$

(사실) 모든 터미널문자(T)는 이미 끝나있다.

1) 이 순서를 거꾸로 (2)를 먼저하고 (1)을 할 경우. 쓸모없는 문자가 쓸모 있는 문자로 잘못 판단되는 경우를 시험문제에서 종종 출제하니 주의하여야 한다.

(조건 1') 끝나는(terminating) 넌 터미널,

$\forall A \in N,\ A \Rightarrow^* x,\ x \in T^*,$

(귀납정의 1) 끝나는(terminating) 넌 터미널

(기본) $\forall a \in T$: $a \subseteq$ Terminate[2]

(귀납) $\forall A \to X_1 X_2 \cdots X_n \in P$:

$(A \to \epsilon) \vee (\forall i \in \{1, \cdots, n\}: X_i \in \text{Terminate})$

$\Rightarrow A \subseteq \text{Terminate}$

(알고리즘 1) 끝나는(terminating) 넌 터미널 구하기

Terminate := {};

$\forall a \in T$: **do** $a \subseteq$ Terminate **od;**

repeat

$\forall A \to X_1 X_2 \cdots X_n \in P$ **do**

if ($(A \to \epsilon) \vee (\forall i \in \{1, \cdots, n\}: X_i \in \text{Terminate})$**)**

$\to A \subseteq$ Terminate

| **otherwise** → **skip**

fi

od

until No more symbols[3] are added to Terminate

$O(k|P||N|)$, 단 k 문법규칙 우변 길이의 최대값.

귀납(recursion)정의를 **반복**(iteration)계산으로 표현하는 좋은 예이다.

(조건 2) 유도되는(accessible) 문자($N \cup T$)[4]

$\forall X \in N \cup T,\ S \Rightarrow^* \alpha X \beta,\ \alpha, \beta \in (N \cup T)^*.$

(정의) 관계(dependency) 그래프 $D = (N \cup T,\ d)$

$A\ d\ X$, if $A \to \alpha X \beta \in P$, 단 $X \in N \cup T,\ \alpha, \beta \in (N \cup T)^*.$

(알고리즘 2) 유도되는(accessible) 문자($N \cup T$) 구하기

관계그래프에서 시작문자 S로부터 경로(path)가 있는($S\, d^*\, X$) 문자(X)는 유도되는 문자이다.

$O(n^2)$[5], 단 n은 문자수.

---

2) Imperative 문장 Terminate := Terminate ∪ {a}의 declarative 표현이다.

3) 이렇게 구하는 방법을 loop의 (고정점)fixed point이러고 부르고, 이것이 끝나는 이유는, 문자(symbol)가 유한하고, 같은 문자가 반복하여 더해지는 것은 집합의 성질 때문에 새로운 문자의 추가로 보지 않기 때문이다.

4) 쓸모없는 규칙을 쓸모없는 문자로부터 정의하기 위하여 유도되는 터미널을 추가하여 정의한다.

5) 유명한 depth-first search 알고리즘이 경로를 찾는 optimal 알고리즘이다.

(정의 3) 문법 $G = (N, T, P, S)$에서 임의의 규칙 $A \to Y_1 Y_2 \cdots Y_n \in P$에서 좌변 A와 우변 Y_1, Y_2, $\cdots$, Y_n의 모든 **문자** $1 \le \forall i \le n$: $Y_i \in N \cup T$가 **쓸모 있으면**, **규칙** $A \to \beta \in P$는 **쓸모 있다**고 정의한다.

문법 $G = (N, T, P, S)$을 쓸모 있게 줄이는 방법은

(1) Terminate하는 문자 $N_{Ter}(\subseteq N)$[6]과 $T_{Ter}(\subseteq T)$을 찾아,
$G_{Ter} = (N_{Ter}, T_{Ter}, P, S)$을 구한다.

(2) Accessible한 문자를 $N_{Usf}(\subseteq N_{Ter} \subseteq N)$과 $T_{Usf}(\subseteq T_{Ter} \subseteq T)$을 찾아,
$G_{Usf} = (N_{Usf}, T_{Usf}, P, S)$을 구한다.

(3) 쓸모없는 문자를 포함하는 규칙을 제거하여($P_{Usf} \subseteq P$) 문법을 줄여서(reduced),
$G_{Red} = (N_{Usf}, T_{Usf}, P_{Usf}, S)$을 구한다.

(정의) 쓸모없는 문자와 규칙을 제거한 문법은 **쓸모 있는(reduced)** 문법이라고 한다.

(정의) 문법 $G = (N, T, P, S)$의 규칙 P의 크기 $|P| \stackrel{\text{def}}{=} \sum_{\alpha \to \beta \in P} |\alpha| + |\beta|$로 정의하고,
문자의 크기 $|V|$는 $|V| \stackrel{\text{def}}{=} |N| + |T|$로 정의하고,
문법 G의 크기 $|G|$는 $|G| \stackrel{\text{def}}{=} \max(|V|, |P|)$로 정의한다.

(사실) 크기가 n인 context-free 문법 $G = (N, T, P, S)$는 $O(n^2)$에 쓸모없는 문자와 규칙을 찾을 수 있고 이를 제거한 **쓸모 있는(reduced)** 문법 $G' = (N', T', P', S')$으로 $O(n^2)$에 바꾼다.

앞으로 모든 문법은 특별한 말이 없으면 쓸모없는 문자와 규칙을 모두 **제거하여(reduced) 쓸모 있는** 문법이다.

6) $S \notin N_{Ter}$이면 $N_{Usf} = T_{Usf} = P_{Usf} = \varnothing$이므로 $G_{Red} = (\varnothing, \varnothing, \varnothing, S)$이다.