네 가지 유형의 형식언어와 자연언어 불어의 연관관계 연구

송 도 규
(한국외국어대학교)

차 례

I. 서 론

자연언어의 현상들을 유한한 개수의 규칙으로 설명하려는 즉 자연언어를 이들 규칙으로 분석하고 생성하려는 시도는 언어의 규칙성을 이루는 근본 메카니즘을 규명하여 언어학을 하나의 과학으로 정립하려는 취지로 오래 전부터 있어 왔고 근자에는 자연언어 자동처리 분야의 대두로 그 중요성이 더욱 부각되고 있다. 자연언어를 다루는 규칙 중에서 일반적으로 널리 쓰이고 유용한 수단을 제공하는 것이 다시쓰기 규칙(règles de réécriture)이다. 다시쓰기 규칙은 문장의 통사구조를 일목요연하게 보여주는 통사분석법 중 직접 구성 성분 분석[1]으로 문장을 분석한 것을 규칙의 형태로 나타낸 것이다. 직접 구성 성분 분석은 언어학을 획기적으로 발전시킨 분석 방법으로 문장을 문장이라는 범주와 각 단어나 형태소의 어휘범주로만 나누던 전통문법과는 달리 구(syntagmes)라는 개념을 도입함으로써 문장 구조에 대한 보다 체계적인 연구를 가능하게 했다. 이 방법의 근본 취지는 문장을 분석하는 데에 문장 구성 성분들을 보다 더 밀접한 관련이 있는 것들끼리 나누어 가는 것이다. 가령 주어, 직접 목적 보어와 결합하여 문장을 구성하는 이항 술어 즉 타동사의 경우만 보더라도 이 타동사는 주어보다는 직접 목적 보어와 더 밀접히 결합되어 있음을 알 수 있다.

1) 문장의 통사 분석법에는 직접 구성 성분 분석외에도 핵심어와 수식어 분석, 역할 구조 분석 등이 있으며 핵심어와 수식어 분석에는 다시 문장 다이어그램 분석과 상관관계 구조 분석이 있다. 자세한 것은 송 도규 (1996:20-50)를 참조하기 바란다.

(1) L'homme a des doutes.
(2) L'homme doute.

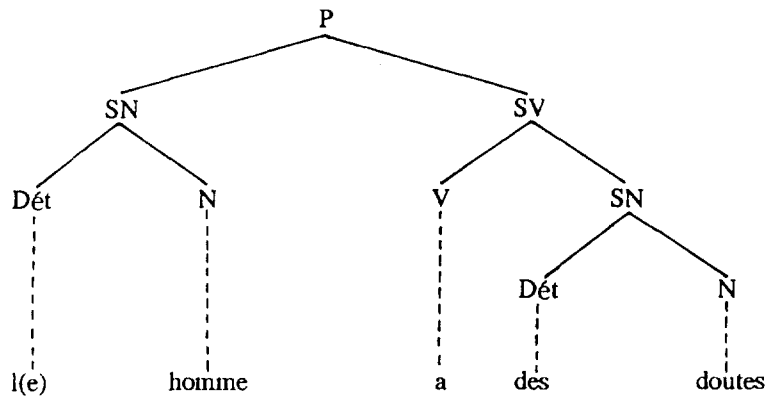
문장 (1)의 동사 "a"는 주어 "l'homme"보다는 목적 보어 "des doutes"와 더 밀접히 결합되어 있어 주어 "l'homme"은 그대로 두고 목적 보어와 먼저 결합하여 문장 (2)와 같이 자동사 "doute"로 대치될 수 있다. 반면에 동사 "a"가 목적 보어 "des doutes"를 그대로 놔두고 주어 "l'homme"과 우선적으로 결합하여 다른 성분으로 대체될 수는 없다. 이것은 다만 불어에 한정된 현상이 아니라 여러 언어에서 공통적으로 확인할 수 있는 보편적인 언어 현상이라 볼 수 있다.

영어: (3) Paul eats dinner.
(4) Paul dines.

한국어: (5) 나는 밥을 먹는다.
(6) 나는 식사한다.

따라서 문장 (1)을 직접 구성 성분 분석법으로 분석하여 수형도로 나타내면 다음과 같다.

(7)



우리는 이 수형도를 다음 규칙으로 나타낼 수 있다.

(8) a. P → SN SV
b. SN → Dét N
c. SV → V SN
d. Dét → l(e), des
e. N → homme, doutes
f. V → a

이 (8)의 규칙과 (7)의 수형도는 표기 방법상의 차이가 있을 뿐 뜻하는 바는 엄밀히 동일하다. 우리는 이 규칙들을 다시쓰기 규칙[2]이라 한다.

다시쓰기 규칙은 그 기능과 성격 상 두 가지로 구분되는데 하나는 통사적 다시쓰기 규칙이고 다른 하나는 어휘삽입 다시쓰기 규칙이다. 통사적 다시쓰기 규칙은 어휘범주와 통사범주 혹은 통사범주[3] 자신들 간의 관계를 규명하는 규칙으로 위 (8)의 a, b, c를 가리키고 어휘삽입 다시쓰기 규칙은 각 어휘범주에 해당하는 실제 단어나 형태소를 삽입시키는 규칙으로 (8)의 d, e, f가 이에 해당한다. 통사적 다시쓰기 규칙은 화살표 왼편에 있는 통사 단위가 화살표 오른편에 있는 구성 성분들로 직접 이루어진다는 뜻이고 어휘삽입 다시쓰기 규칙은 화살표 왼편에 있는 어휘범주 자리에 화살표 오른편에 있는 단어나 형태소가 삽입된다는 의미이다.

이 규칙들은 앞에서도 언급 한대로 유한한 개수의 규칙들로 무한한 수의 자연언어 문장들을 분석하고 생성할 수 있다는 점에서 그 시사하는 바가 크다. 그저 몇 쪽에 모조리 나열할 수 있는 유한한 규칙들로 무한한 자연언어의 문장들을 다룰 수 있다는 가능성이 언어를 체계적으로 설명할 수 있는 발판을 마련해 주었고 그 자동처리의 모태가 되었다 해도 과언이 아닐 것이다.

그런데 여기서 이 다시쓰기 규칙들에 대해 좀 더 자세히 생각해 볼 필요가 있다. 왜냐하면 다시쓰기 규칙의 형태에 따라 이것이 구성할 수 있는 문법과 이 문법이 생성할 수 있는 언어의 성격이 매우 달라지기 때문이다. 다시쓰기 규칙의 형태에 제한을 두지 않으면 다시 말해서 다시쓰기 규칙이 모든 형태를 자유로이 가질 수 있으면 그것이 분석, 생성할 수 있는 대상 언어의 양은 많아지고 보다 복잡한 유형의 언어를 다룰 수 있게 되는 반면에 규칙 자체에 제한이 없으므로 일정하게 체계화할 수 없을 뿐만 아니라 주먹구구식의 규칙 운용이 되기 쉽다. 반대로 규칙의 형태에 많은 제약을 가하면 형태가 단순 명료해져서 규칙을 일관성 있고 체계적으로 다룰 수 있게 될 뿐만 아니라 자연언어 자동처리를 위한 시스템 구축에도 용이하게 활용할 수 있다는 장점은 있으나 자연언어의 모든 형태를 분석, 생성할 수 없어진다는 치명적 결함이 남는다. 따라서 우리는 여기서 자연언어의 모든 문장 구조를 설명할 수 있으면서 동시에 가장 단순한 형태의 다시쓰기 규칙을 찾아내는 것이 관건이라는 것에 쉽게 생각이 미친다.

II. 본 론

촘스키는 다시쓰기 규칙의 여러 가능한 형태를 네 가지로 정리하여 다음과 같이 분류하였다.

1. $\alpha \rightarrow \beta$ $(\alpha, \beta \in (S_T \cup S_N)^*$, α는 하나 이상의 비단말기호 포함)

2. $A\alpha B \rightarrow A\beta B$ $(A,B,\beta \in (S_T \cup S_N)^*, \alpha \in S_N, \beta \neq \varepsilon)$

2) 이 규칙들은 구 구조 규칙(règles syntagmatiques), 생산 규칙(règles de production), 생성 규칙(règles de génération), 형성 규칙(règles de formation)이라 불리기도 하는데 이 규칙들이 이렇게도 불려지는 이유는 이들을 문장을 분석할 때와 역순으로 적용하여 원래의 문장 "l'homme a des doutes"를 생성할 수 있기 때문이다.

3) 여기에 쓰인 어휘범주, 통사범주는 송 도규(1994, 1996:91-98)에 정의된 대로이다.

3. $A \rightarrow \alpha$ ($\alpha \in (S_T \cup S_N)^*$, $A \in S_N$, 왼편에 오직 하나의 비단말기호)

4. $A \rightarrow \alpha B$, $A \rightarrow \alpha$, $A \rightarrow \varepsilon$

혹은 ($A,B \in S_N$, $\alpha \in S_T$)

$A \rightarrow B\alpha$, $A \rightarrow \alpha$, $A \rightarrow \varepsilon$

여기서 'ST'는 단말기호(symboles terminaux)를, 'SN'은 비단말기호(symboles non-terminaux)를 뜻한다. 단말기호라는 것은 '실제 언어를 구성하는 어휘들의 집합'을 가리키고 비단말기호라는 것은 '실제 언어에는 나타나지 않지만 그들의 생성에 필요한 단위들의 집합'으로 정의된다. 사실 단말기호나 비단말기호라는 명칭은 수형도의 말단마디에 나타날 수 있는가의 여부에 따라 말단마디에 나타나는 단어와 형태소는 단말기호에 속하고 말단마디에 나타날 수 없는 어휘범주와 통사범주는 비단말기호에 해당한다. 또 '*'는 길이가 0인 문자열부터 길이가 무한대인 문자열까지의 가능한 모든 조합을 뜻하는 '클린 스타(étoile de Kleene)'이다.

촘스키는 이 서로 다른 형태의 규칙들이 구성하는 문법을 각각 구별하여 모두 네 가지 유형의 문법을 상정하였다. 1번 류의 다시쓰기 규칙을 포함하는 문법은 유형 0문법(grammaires du type 0)으로, 2번 류의 규칙들로 구성될 수 있는 문법을 유형 1문법(grammaires du type 1), 3번 류의 규칙들이 구성하는 문법을 유형 2문법(grammaires du type 2), 4번 류의 규칙들로 이루어지는 문법을 유형 3문법(grammaires du type 3)으로 분류하였다. 그는 또 이 모든 유형의 문법들을 형식문법(grammaires formelles)이라 이름하고 다음과 같은 네 가지 요소로 구성하였다.

$G = \{S_T, S_N, P, R\}$

ST : 단말기호 (실제 언어를 구성하는 어휘들의 집합)

SN : 비단말기호 (실제 언어에는 나타나지 않지만 그것들의 생성에 필요한 단위들의 집합)

P : 비단말기호의 하나로 특수한 상태에 해당하며 초기 상태(état initial)라 불린다.

R : 다시쓰기 규칙들의 집합.

이 형식문법을 한 마디로 정의하면 모든 문법적인 문자열 즉 정문(正文)을 분석, 생성할 수 있는 동시에 비문법적인 문자열 즉 비문(非文)은 전혀 분석, 생성하지 않는 체계적 장치를 말한다. 따라서 형식문법의 기능은 크게 두 가지로 볼 수 있다. 그 하나는 정문을 비문으로부터 구분하는 것이고 다른 하나는 정문의 구조를 규명하는 것이다.

본고에서는 각 유형의 형식 문법의 구조와 각 형식문법들이 분석, 생성할 수 있는 언어들간에 나타나는 차이를 따져 보고 나름대로의 언어 특이성(typologie)를 가지고 있는 하나의 자연언어인 불어는 위 네 가지 중 어느 유형의 형식문법으로 설명될 수 있으며 어느 유형의 형식언어에 해당하는지를 가늠해 보겠다.

1. 형식언어의 정의

각 유형에 해당하는 형식문법과 형식언어를 개별적으로 논의하기 전에 먼저 일반적으로 언어[4]란 무엇인가 생각해 보고 넘어가는 것이 좋을 듯 싶다. 이해를 쉽게 하기 위하여 단말기호가 a와 b로만 이루어지는 언어를 가정해 보자.

(9) S = { a, b}

이 단말기호 a와 b로 조합될 수 있는 모든 문자열의 집합은 다음과 같다.

(10) S* = {ε, a, b, aa, ab, ba, bb, aaa, aaab, abab, abbaab, ...}

여기서 '*'는 앞에서 설명한 길이가 0인 문자열부터 길이가 무한대인 문자열까지의 가능한 모든 조합을 뜻하는 '클린 스타(Kleene star)'이고 S*는 S에 대한 '모노이드 monoïde'라 한다. 이와 같은 가정 하에서 언어(L)를 다음과 같이 정의할 수 있다.

(11) $L \subseteq S^*$

즉 언어는 S 모노이드의 부분집합이다.

그러나 여기서 언어를 구성하는 문자열의 길이를 제한하면 단말기호로부터 생성될 수 있는 문자열의 수가 단말기호의 개수에 규칙적으로 비례한다. 즉 (9)로부터 생성될 수 있는 길이가 단말기호 하나인 문자열의 집합은 다음과 같다.

(12) L_1 = { a, b}

또 (9)로부터 생성될 수 있는 길이가 단말기호 두 개인 문자열의 집합은 아래와 같고

(13) L_2 = { aa, ab, ba, bb}

길이가 기호 세 개인 문자열의 집합은 다음처럼 나타난다.

(14) L_3 = { aaa, aab, aba, baa, abb, bab, bba, bbb}

4) 일반적으로 형식문법에 의해 생성되는 언어를 형식언어라 한다. 따라서 이 논의에서 다루어지고 있는 언어는 자연언어를 제외하고는 모두 형식언어에 속한다.

여기서 우리는 단말기호의 개수에 따라 생성될 수 있는 제한된 길이의 문자열의 개수를 아래 규칙으로 일반화할 수 있다.

> 임의의 언어의 단말기호의 개수를 m이라 하고 가능한 문자열의 길이를 n이라 하면, 또 이 단말기호로부터 생성될 수 있는 문자열의 길이가 n인 총 문자열의 개수를 p라고 하면 이들 간에는 다음과 같은 규칙이 성립한다.

(15) $p = m^n$

여기서 이 규칙을 불어에 적용하여 보면 어떻게 될까. 불어는 모두 26개의 문자를 가지고 있으므로 이 문자들이 조합해서 생성할 수 있는 단어나 형태소의 수는 이들 문자열이 포함할 수 있는 문자의 수에 따라 다음과 같다.

(16) 문자의 수가 1개인 모든 조합 가능한 단어의 개수 = 26^1 = 26
문자의 수가 2개인 모든 조합 가능한 단어의 개수 = 26^2 = 676
2문자의 수가 3개인 모든 조합 가능한 단어의 개수 = 26^3 = 17,576
문자의 수가 4개인 모든 조합 가능한 단어의 개수 = 26^4 = 456,976
문자의 수가 5개인 모든 조합 가능한 단어의 개수 = 26^5 = 11,881,376
⋮

이렇게 보면 문자의 수가 1개인 단어부터 문자의 수가 5개인 단어까지만 보더라도 불어의 26개 문자가 구성할 수 있는 가능한 모든 단어의 수는 12,356,630개에 이른다. 이는 문자의 수가 5개 이하인 단어의 수만 해도 천만 개를 훨씬 넘을 수 있다는 말이다. 게다가 불어에는 문자의 수가 6개 이상인 단어도 많다.

이로부터 우리는 불어의 단어는 불어 알파벳이 조합해서 생성할 수 있는 모든 가능한 문자열로 이루어지지 않았다는 것을 알 수 있다. 그러니까 가능한 문자열에서 어떤 것은 단어로 쓰이고 어떤 것은 단어로 선택되지 않았다는 것이다.

자 이제는 이 메카니즘을 단어와 문장의 차원으로 확대 적용해 보자. 여기서는 단어를 단말기호로 보고 이들이 즉 단어가 조합해서 문장을 구성한다. 논의의 편의를 위해 단말기호가 다음과 같이 5개의 단어로 구성된 언어를 상정해 보자.

(17) S = { je, tu, il, chante, chantes}

이것으로부터 얻을 수 있는 S 모노이드는 다음과 같다.

(18) S^* = { ε, je, tu, il,..., je je, je tu, tu il,..., chante, chante chante, chante chanes, ..., je chante, tu chante, il chante,..., il il il ,..., il chante chante, chante je tu, chantes chante il je,...}

이 S 모노이드는 그 원소가 가히 무한하다. 따라서 우리의 논의를 위해 단어 수가 2개인 문자열로만 이루어진 언어만을 대상으로 제한하겠다. (17)로부터 생성 가능한 단어 수가 2개인 문자열의 집합은 위 규칙 (15)에 따라 그 원소의 개수가 52 즉 25개이다. 이를 나열해 보자.

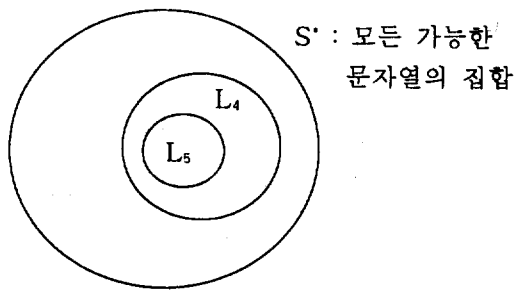
(19) L_4 = { je je, je tu, je il, tu tu, tu je, tu il, il il, il je, il tu, je chante, je chantes, tu chante, tu chantes, il chante, il chatnes, chante je, chantes je, chante tu, chantes tu, chante il, chantes il, chante chante, chante chantes, chantes chante, chantes chantes}

즉 (17)의 단말기호를 적용해서 얻을 수 있는 단어의 수가 2개인 가능한 모든 문자열로 이루어진 언어는 위 유한 집합 L4로 나타난다. 그러나 불어에서는 단어의 경우와 마찬가지로 이 모든 문자열을 모두 정문으로 인정하지 않고 다음 문자열만 정문으로 받아들인다.

(20) L_5 = { je chante, tu chantes, il chante}[5)]

다시 말해서 위 (17)의 S로부터 얻을 수 있는 가능한 수많은 언어들 중 L5만이 정문으로만 구성된 불어에 합치되는 언어임을 알 수 있다. 즉 불어는 불어의 단말기호인 불어 단어나 형태소로 이루어진 모노이드의 진부분집합이다. 이들 언어들간의 집합 관계를 그림으로 나타내면 다음과 같다.

(21)



<그림 1 언어 L_4, L_5 간의 관계>

여기서 우리는 L_5를 L_4로부터 구분해 주는 규칙이 필요하고 이 규칙의 형태가 이를 수행하는 데에 충분히 강력하면서 동시에 가능한 가장 제한된 형태를 갖는 규칙이 앞에서 논의한 4가지 유형의 형식문법 중 어는 문법을 구성하는 지를 알아보는 것이 본고의 목적이다.

이제부터 모노이드의 상이한 부분집합으로 정의되는 여러 가지 유형의 언어들과 이 언어들을 생성할 수 있는 문법들을 차례로 설명하겠다.

5) "chantes tu"와 "chante il"은 "chantes-tu"와 "chante-t-il"과 같이 두 단어 사이에 "-"와 "-t-"가 삽입되어야 하므로 여기에서는 정문으로 간주하지 않는다.

2. 네 가지 유형의 형식문법과 형식언어

2.1 정규 문법과 정규 언어

정규 문법(grammaires régulières)은 클린 문법(grammaires de Kleene)이라고도 하는데 정규 언어를 기술하는 하나의 방법이다. 정의에 의해 정규 문법의 각 규칙의 화살표 오른편에 비단말기호(symboles non-terminaux)는 오직 하나만 존재해야 하며 단말기호(symboles terminaux)는 비단말기호의 오른편이나 왼편 중 한쪽에만 위치해야 한다. 다음은 정규 문법의 정의이다.

정규 문법의 정의

문법 $G = \{ S_T, S_N, P, R \}$의 모든 규칙의 형태가

$A \rightarrow \alpha B \qquad A, B \in S_N$

$A \rightarrow \alpha \qquad \alpha \in S_T$

인 경우 문법 G를 우선형 정규 문법이라 한다.

만약 모든 규칙의 형태가

$A \rightarrow B \alpha$

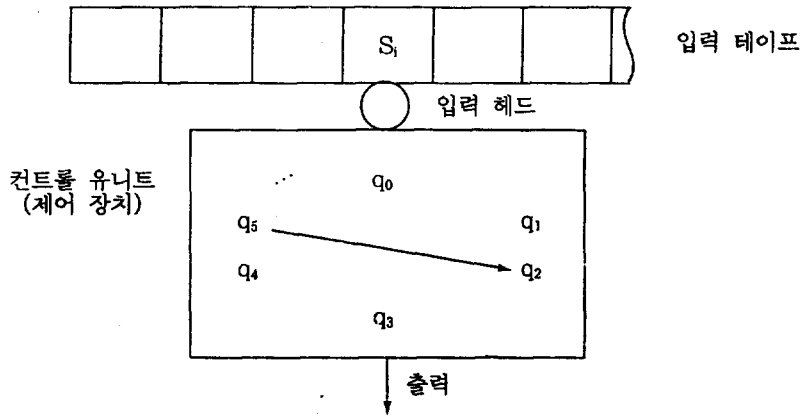
$A \rightarrow \alpha$

인 경우 문법 G를 좌선형 정규 문법이라 한다.

각 정규 문법은 우선형이거나 좌선형 중 하나로 정의된다.

좌선형과 우선형이 혼합된 문법을 단순히 선형 정규 문법이라 한다.

이 정규 문법으로 생성할 수 있는 언어의 집합이 정규 언어이다. 정규 언어는 정규 문법 외에도 유한상태 오토마타와 정규표현에 의해서도 구현될 수 있다. 정규 언어는 무한할 수 있다. 그러나, 모든 정규 언어는 유한개수의 상태를 갖는 유한상태 오토마타(automates à états finis)에 의해 인식되기 때문에 정규 언어의 구조에는 어떤 제약이 따른다. 그러면 여기서 어떻게 유한상태 오토마타가 정규 언어를 인식하는지를 살펴보자. 먼저 오토마타는 '인간의 두뇌를 흉내낸 기계에서의 정보처리 구조'를 말한다. 이 오토마타는 추상적 개념으로 그 구조를 그림으로 나타내면 다음과 같다.



<그림 2 유한상태 오토마타의 구조>

이제는 앞에서 예로 다룬 자연언어의 한 집합 L_5를 인식하는 정규 문법을 만들어 보자.

S_T = { je, tu, il, chante, chantes}

S_N = {q_0, q_1, q_2, q_3}

P = q_0

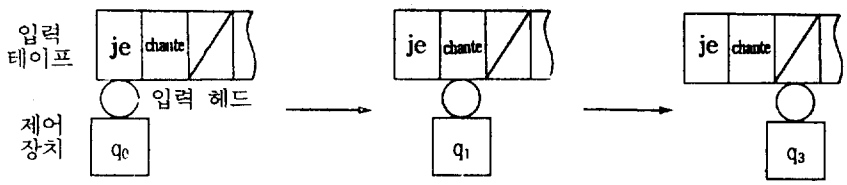
R : $q_0 \rightarrow$ je q_1
$q_0 \rightarrow$ tu q_2
$q_0 \rightarrow$ il q_1
$q_1 \rightarrow$ chante q_3
$q_2 \rightarrow$ chantes q_3

이 정규 문법은 언어 L_5의 원소인 "je chante", "tu chantes", "il chante"를 정문으로 인식한다. 반면에 비문인 "je chantes", "tu chante"등은 인식하지 못한다. 이제 그 과정을 자세히 관찰하자. 위 정규 문법의 다시쓰기 규칙을 표로 나타내면 다음과 같다.

상태 \ 입력	je	tu	il	chante	chantes
q_0	q_1	q_2	q_1	/	/
q_1	/	/	/	q_3	/
q_2	/	/	/	/	q_3
q_3	/	/	/	/	/

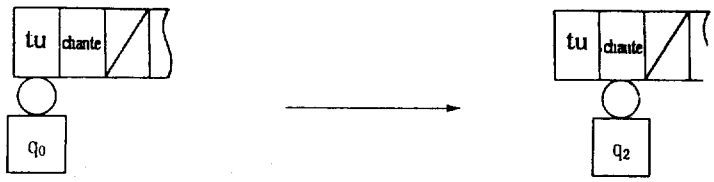
<그림 3 표로 나타낸 위 정규 문법의 다시쓰기 규칙>

자 이제 이 정규 문법의 다시쓰기 규칙을 그대로 수행하는 오토마타가 "je chante"를 인식하는 과정을 보자.



<그림 4 위 유한상태 오토마타가 문자열 "je chante"를 인식하는 과정>

더 이상의 입력이 없는 상황에서 제어장치의 상태가 최종상태인 'q_3'이므로 이 문자열은 이 오토마타에 의해서 인식된 정문이다. 이번에는 비문인 "tu chante"의 인식과정을 검토하자.



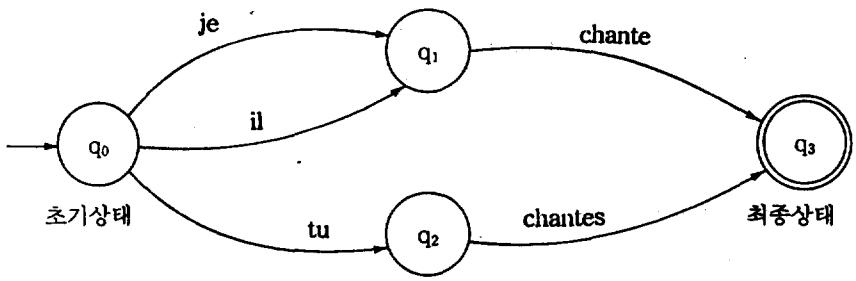
<그림 5 위 유한상태 오토마타가 문자열 "tu chante"의 인식을 실패하는 과정>

여기서는 더 이상의 입력이 남아 있지 않은 동시에 제어장치의 상태가 'q_2'이므로 최종상태가 아니다. 따라서 이 문자열은 위 오토마타에 의해서 성공적으로 인식되지 않는 비문인 것이다.

앞에서도 밝힌 바대로 오토마타는 실제 존재하는 기계가 아닌 추상적 개념이므로 상태 그래프로 나

타내면 편리하다. 이 상태 그래프는 상태 다이어그램, 전이 그래프 혹은 전이 다이어그램이라 하기도 한다. 상태 그래프를 이용하여 앞에서 다룬 언어의 예 L_5을 나타내기로 하자.

L_5 = {je chante, tu chantes, il chante}

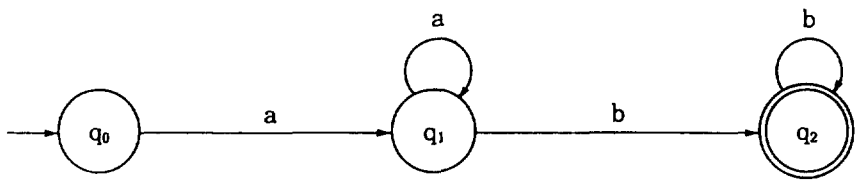


<그림 6 언어 L_5를 표상한 상태 그래프>

이 상태 그래프는, 쉽게 확인할 수 있듯이, "je chante", "tu chantes", "il chante"를 인식한다. 연습을 위해 다른 타입의 언어를 인식하는 상태 그래프를 만들어 보자. 단말기호가 a, b 두개이며 적어도 하나씩의 a와 b가 임의의 수로 반복하여 이루어지는 다음과 같은 언어를 가정하자.

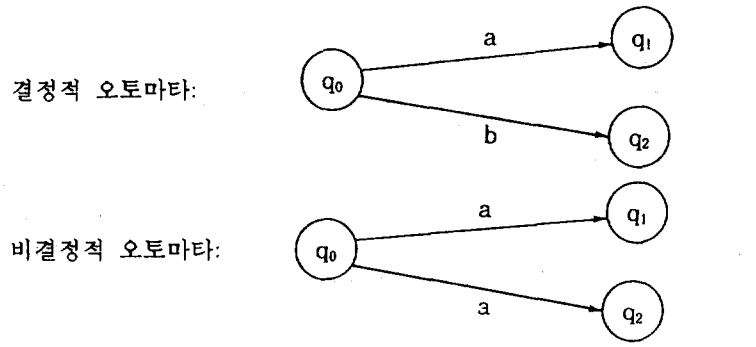
$L_i = a^m b^n$= {ab, aab, abb, aaaab, aabbb, ...} (m,n≥1)

이 언어를 인식하는 상태 그래프는 다음과 같다.



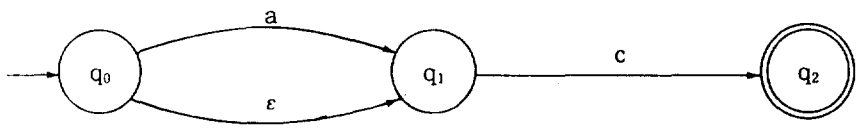
<그림 7 언어 L_i를 인식하는 상태 그래프>

오토마타는 한 상태에서 다른 상태로 전이하는 과정에서 취할 수 있는 상태의 선택 폭에 따라 두 종류로 구분된다. 그 하나는 결정적 오토마타(automates déterministes)이고 다른 하나는 비결정적 오토마타(automates non-déterministes)이다. 결정적 오토마타와 비결정적 오토마타를 상태 그래프로 표상하면 아래와 같다.



<그림 8 결정적 오토마타와 비결정적 오토마타>

즉 결정적 오토마타에서는 입력 자료에 따라 다음에 취할 수 있는 상태가 오직 하나인 반면에 비결정적 오토마타는 하나의 같은 입력 자료에 따라 취할 수 있는 상태가 두 개 이상이다. 그러나 모든 비결정적 오토마타는 결정적 오토마타로 전환이 가능하다. 다음 예를 보자.

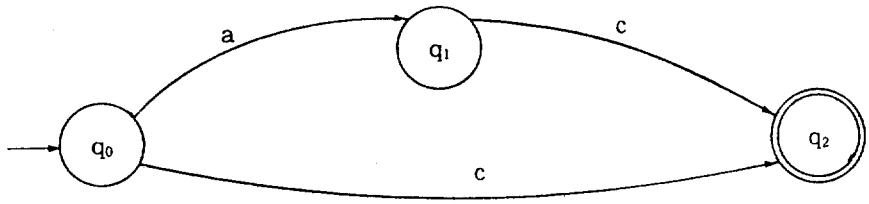


<그림 9 비결정적 오토마타의 한 예>

이것은 비결정적 오토마타이다. 왜냐하면 입력 자료가 a일때 q_0에서 출발하여 q_1으로 전이될 수도 있고 a를 $(\varepsilon \cdot a)$로 보아 먼저 q_1으로 전이하여 입력 자료에 해당하는 a를 찾을 수도 있기 때문이다. 이 비결정적 오토마타는 다음 계산에 의해 결정적 오토마타로 전환될 수 있다.

(22) $(a + \varepsilon)\cdot c = a\cdot c + \varepsilon\cdot c = a\cdot c + c$

즉



<그림 10 그림 9의 비결정적 오토마타에 해당하는 결정적 오토마타>

이리하여 같은 인식력을 가지는 위의 결정적 오토마타로 표상할 수 있는 것이다.

지금까지 살펴본 정규 문법은 자연언어의 모든 문자열을 다 인식하지는 못한다. 대표적인 예가 a^nb^n 형태의 언어이다. 이 언어는 같은 수의 a와 b가 반복하여 나타나는 문자열의 집합이다.

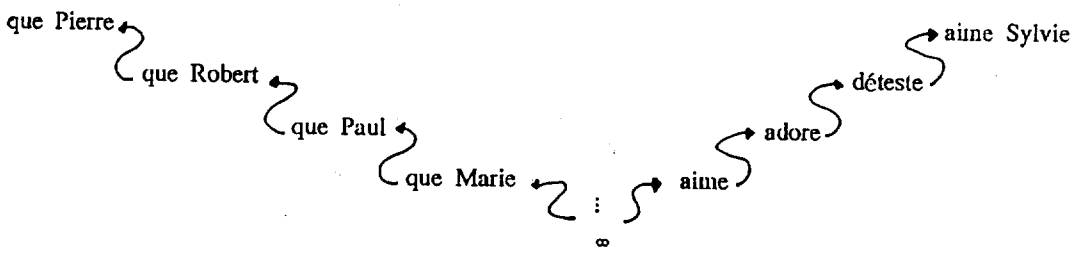
$L_j = a^nb^n = \{\varepsilon, ab, aabb, aaabbb, aaaabbbb, ..., aaaaaaabbbbbbb, ...\}$ $(n \geq 0)$

이 형태에 해당하는 불어 문장의 예에는 다음과 같은 것들이 있다.

(23) a. Que Pierre que Robert que Paul que Marie aime adore déteste aime Sylvie.

b. Jean, Paul et Jacques aiment respectivement Martine, Colette et Sylvie.

관계절을 무한대로 가운데에 삽입할 수 있는 (23.a)와 같은 형태의 문장은 전반부의 'que 명사'와 후반부의 '동사'의 개수가 똑같아야 하는 (명사구)^n(동사구)^{n}형의 문장이다.



<그림 11 a^nb^n형 불어 문장의 한 예>

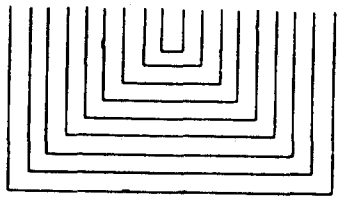
관계절이 무한대는커녕 몇 개 이상이 삽입된 문장도 실제 일상 언어에서는 찾아보기 힘들다 할지라도 이는 만들어지고 이해될 수 있는 정문이다. 따라서 이를 인식하지 못하는 정규 문법은 자연언어의 모든 구조를 다루는데 부적절하다고 판단되고 있다. 또 (23.b)에서도 주어를 구성하는 명사의 개수와 목적어를 이루는 명사의 개수가 정확히 일치하여야 한다. 이런 구조의 문장도 정규 문법으로는 다룰 수 없다.

정규 문법으로는 다음과 같은 언어도 생성할 수 없다. 그것은 문자열이 가운데를 중심으로 양측이 대칭적으로 같은 거울 영상 언어(langages de miroir)이다. 예를 몇 개 들어보자.

aabbbbaa

abaabbaabbaaba

aabbababbbbababbaa



<그림 12 거울 영상 언어의 예들>

지금까지 언급한 a^nb^n형의 언어와 거울 영상 언어는 정규 문법으로는 생성할 수 없다. 왜냐하면 사실 이 언어들은 정규 문법의 생성력을 넘어서는 문맥 무관 언어에 속하기 때문이다. 다음에는 이 언어들을 자유롭게 생성할 수 있는 문맥 무관 문법에 대해서 논의해 보겠다.

2.2 문맥 무관 문법과 문맥 무관 언어

앞 절에서 정규 문법으로는 생성할 수 없는 언어들에 대해서 논의하였다. 이 언어들은 문맥 무관 언어(langages indépendants du contexte)들로서 문맥 무관 문법(grammaires indépendants du contexte)[6]

6) 우리는 흔히 문맥 무관 문법(영어로는 context-free grammars라고 한다)을 그 이름만으로 짐작하여 문장의 전후 관계를 고려하지 않는, 텍스트 전체의 정보를 무시하는 문법으로 오해하는 경향이 있다. 또 이 문법에 상대적인 문맥 연관 문법(grammaires dépendantes du contexte, 영어로는 context-sensitive grammars)을 문맥 무관 문법과는 반대로 문장의 전후 관계를 고려하는, 텍스트 전체의 정보를 참조하는 화용론적인 혹은 텍스트 언어학적인 문법 이론으로 잘 못 이해하는 경우가 많다. 그러나 문맥 무관 문법과 문맥 연관 문법은 문장의 전후 관계, 즉 텍스트 차원에서의 문맥과는 전혀 관계없는, 각 문법을 구성하는 다시쓰기 규칙의 형태에 따른 문법 유형의 분류들이다. 즉 다시쓰기 규칙의 적용이 화살표 왼편에 있는 기호에 따라 다른 요소를 고려하지 않고 이루어지는 문법을 문맥 무관 문법이라 하고 반대로 규칙의 왼편에 있는 기호가 둘 이상의 통사 환경을 만들어 이를 만족할 때에만 규칙의 적용이 수행되는 문법을 문맥 연관 문법이라 한다. 문맥 무관 문법을 가리키는 불어의 'grammaires indépendantes du contexte'는 영어의 'context-free grammars'를 불어

이상의 생성력을 가진 문법으로서만 생성할 수 있다. 그러면 문맥 무관 문법을 정의하여 보자.

문맥 무관 문법의 정의

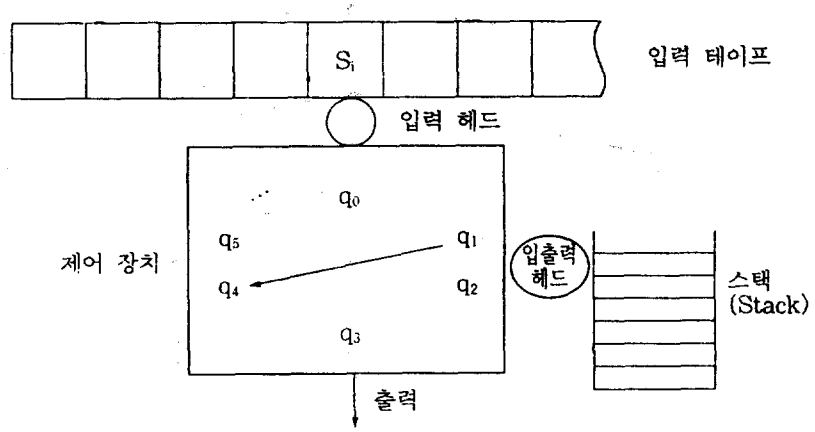
문법 $G = \{ S_T, S_N, P, R \}$의 모든 규칙의 형태가

$$A \rightarrow \alpha \qquad A \in S_N, \quad \alpha \in (S_N \cup S_T)^*$$

인 경우 문법 G를 문맥 무관 문법이라 한다.

이 문맥 무관 문법으로 생성되는 언어의 집합이 문맥 무관 언어이다. 문맥 무관 문법이라는 명칭은 규칙의 왼편에 있는 비단말기호가 문장을 구성하는 다른 요소들 즉 문맥과는 무관하게 치환될 수 있다는 가정으로부터 생겨났다. 이는 규칙의 왼편에 오직 하나의 비단말기호만이 존재하기 때문에 가능하다.

문맥 무관 언어는 푸시다운 오토마타(automates à 1 pile de mémoire)에 의해 인식된다. 다음 그림은 푸시다운 오토마타의 구조를 나타낸 것이다.



<그림 13 푸시다운 오토마타의 구조>

로 옮긴 것이고 문맥 연관 문법에 해당하는 'grammaires dépendantes du contexte'는 영어의 'context-sensitive grammars'를 불어로 번역한 것이다. 학자에 따라서는 이 문법들을 각각 'grammaires non-contextuelles'과 'grammaires contextuelles'이라 부르기도 한다. 그런데 여기서 한 가지 특기할 것은 문장의 전후 관계를 고려하는, 텍스트 전체 혹은 담화 상황 전체의 관점에서 언어를 다루려는 문법을 영어로 'context-dependent grammars'라 하고 이것들을 고려치 않는 문법을 'context-independent grammars'라 한다는 것이다.

이 푸시다운 오토마타는 입력 테이프에서 입력 자료를 입력 헤드를 통해 받아들여 그에 따라 스택의 내용을 변경시킨다. 제어 장치의 작동은 입력 자료와 스택의 맨 위에 저장되어 있는 자료에 의해 결정된다. 이 작동의 결과로 제어 장치 내의 상태가 변경되고 스택의 맨 위의 자료가 바뀐다.

이제 문맥 무관 문법으로 앞에서 정규 문법으로는 생성할 수 없었던 문장형들을 생성해 보자. 먼저 a^nb^n형의 문장을 생성할 수 있는 문맥 무관 문법을 만들어 보자.

$S_T = \{a, b\}$

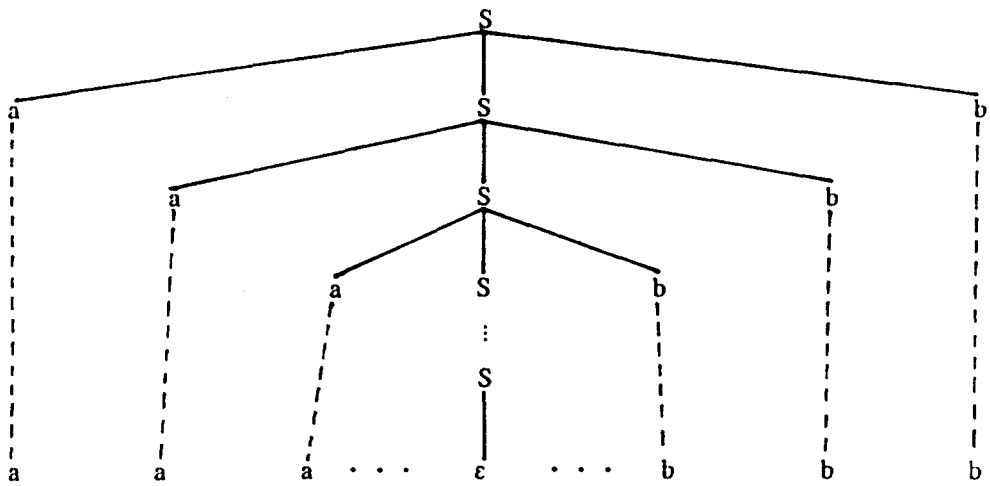
$S_N = \{S\}$

$P = S$

$R : S \rightarrow aSb$

$S \rightarrow \varepsilon$

위에서 설정된 하나의 문맥 무관 문법은 다음과 같은 과정으로 a^nb^n형의 문자열을 생성해 낼 수 있다.



그리하여 문자열 "aaaa...$a_n$bbbb...b_n"을 생성함을 알 수 있다.

따라서 앞의 정규 문법으로는 생성할 수 없었던 불어 문장 "Que Pierre que Robert que Paul que Marie aime adore déteste aime Sylvie"를 생성할 수 있는 문맥 무관 문법을 다음과 같이 설정할 수 있

S_T = {SN, SV}

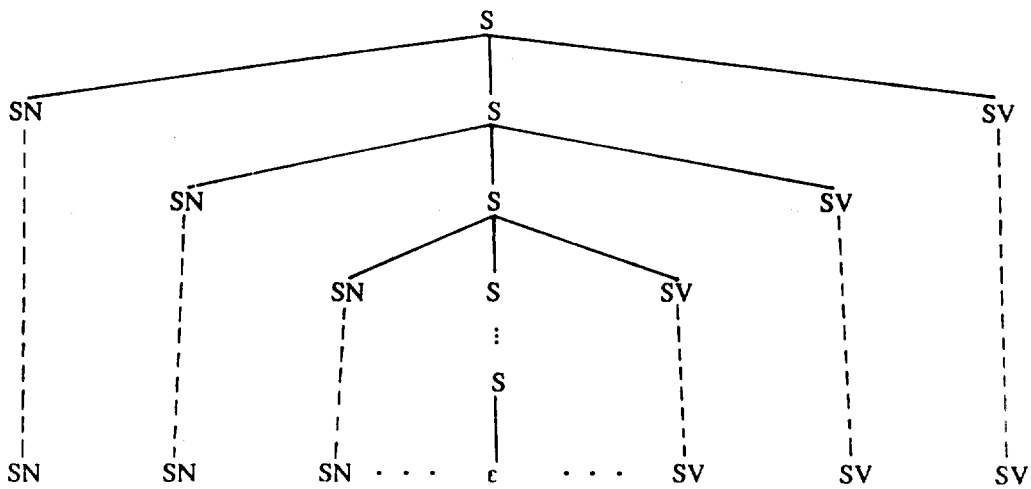
S_N = {S}

P = S

R : S → SN S SV

S → ε

그리하여 다음과 같이 (명사구)^n (동사구)^n 구조를 만들어 낼 수 있다.



이번에는 거울 영상 언어를 생성할 수 있는 문맥 무관 문법을 설정해 보자.

S_T = {a, b}

S_N = {S}

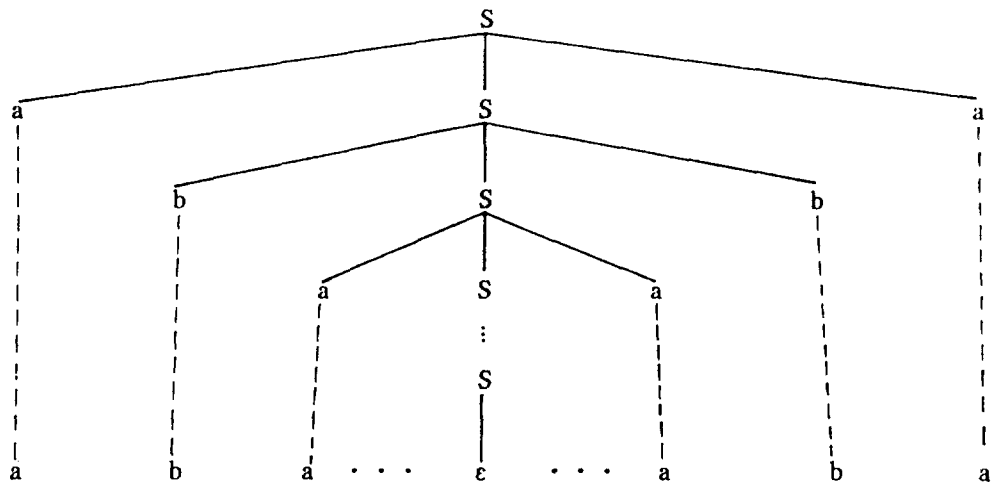
P = S

R : S → aSa

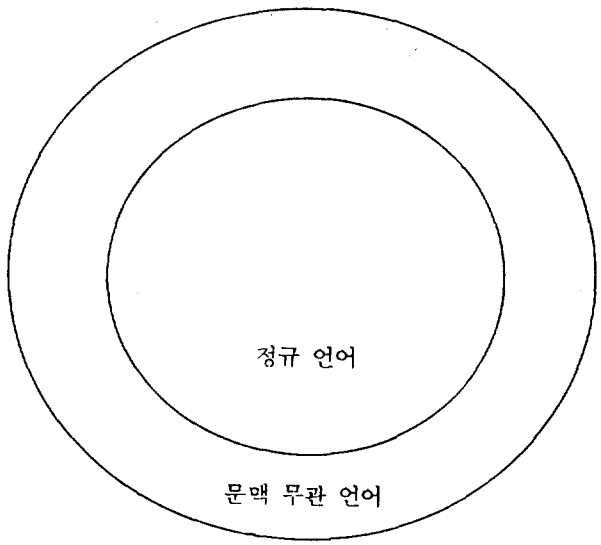
S → bSb

S → ε

이 문맥 무관 문법은 다음과 같이 거울 영상 언어를 생성한다.



따라서 정규 언어와 문맥 무관 언어 사이에는 다음과 같은 관계가 성립된다.



<그림 14 정규 언어와 문맥 무관 언어의 관계>

그러므로 모든 정규 언어는 문맥 무관 언어로 전환될 수 있는 반면에 모든 문맥 무관 언어가 정규 언어로 전환될 수 있는 것은 아니다.

그러나 정규 문법보다 강력한 생성력을 가지고 있는 문맥 무관 문법으로도 생성할 수 없는 문장의 형태가 불어에는 존재한다. 그것은 $a^nb^nc^n$ $(n\geq 0)$형태의 문자열에 속하는 것으로서 다음 문장이 불어에

나타나는 대표적인 예이다.

(24) a. Paul, Simon, Pierre offrent un livre, une fleur, une carte, à Marie, à Sylvie, à Martine.

b. Paul, Jacques, Louis sont médecin, dentiste, coiffeur à Paris, à Lyon, à Marseille.

이 문장은 n=3인 $a^nb^nc^n$의 형태로서 (고유명사)^3(명사구)^3(전치사구)^{3}에 해당한다. 우리는 여기서 n의 수를 무한히 증가시킬 수 있다.

(25) Paul, Simon, Pierre... offrent un livre, une fleur, une carte,... à Marie, à Sylvie, à Martine...

그러나 실제 일상 불어 사용에서 이처럼 각 요소가 무한히 반복되는 문장을 접하기란 불가능하다. 하지만 이와 같은 문장도 자연언어에 속하는 엄연한 정문이기 때문에 이런 류의 문장도 생성할 수 있는 체계적 장치 즉 문법이 필요하다. 이 문법이 문맥 연관 문법이다. $a^nb^nc^n$형태의 언어는 문맥 무관 문법의 생성력을 벗어나는 문맥 연관 언어로서 문맥 연관 문법 이상의 생성력을 가지는 문법으로서만 생성될 수 있다.

2.3 문맥 연관 문법과 문맥 연관 언어

앞 절에서 문맥 무관 문법과 이 문법이 생성하지 못하는 문맥 연관 언어에 대해서 논의하였다. 이 문맥 연관 언어(langages dépendants du contexte)는 문맥 연관 문법(grammaires dépendantes du contexte) 이상의 생성력을 가진 문법으로서만 생성될 수 있음도 밝혔다. 그러면 여기서 문맥 연관 문법이란 무엇인가 정의하여 보자.

문맥 연관 문법의 정의

문법 $G = \{ S_T, S_N, P, R \}$의 모든 규칙의 형태가

$$u \rightarrow v \qquad u, v \in (S_N \cup S_T)^+ \quad |u| \leq |v|$$

을 만족하는 경우 문법 G를 문맥 연관 문법이라 한다.

여기서 S^+는 $S^* - \{\varepsilon\}$ 을 뜻한다.

이 문맥 연관 문법에서는 규칙의 왼편에는 아무런 제한이 없지만 오른편에는 적어도 왼편의 문자열 보다 같거나 긴 문자열이 있어야 한다는 제약이 있다. 이제 앞절에서 예로 든 문맥 연관 언어 $a^nb^nc^n$을 생성할 수 있는 하나의 문맥 연관 문법을 설정해 보자.

S_T = {a, b, c}

S_N = {S, B, C}

P = S

R : S → aSBC
S → aBC
CB → BC
B → b
C → c

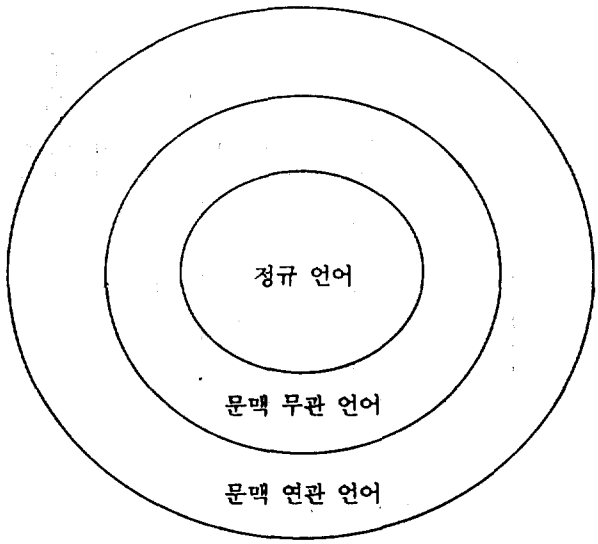
이 문법이 $a^nb^nc^n$을 생성하는 과정은 다음과 같다.

			S					
a			S				B	C
a	a		S		B	C	B	C
a	a	a	B	C	B	C	B	C
a	a	a	B	B	C	C	B	C
a	a	a	B	B	C	B	C	C
a	a	a	B	B	B	C	C	C
a	a	a	b	b	b	c	c	c

<그림 15 언어 $a^nb^nc^n$의 생성 과정>

이 생성은 위에서 설정한 문맥 연관 문법의 규칙들중 세 번째 규칙인 'CB→BC'에 의해서 가능한데 이것이 바로 문맥 연관 규칙이다. 즉 'CB'가 기호 'C'와 'B'가 연이어 나타나는 문맥에서만 'BC'로 치환될 수 있다는 것이다.

지금까지 살펴본 정규 언어, 문맥 무관 언어, 문맥 연관 언어에는 다음과 같은 관계가 성립함을 알 수 있다.

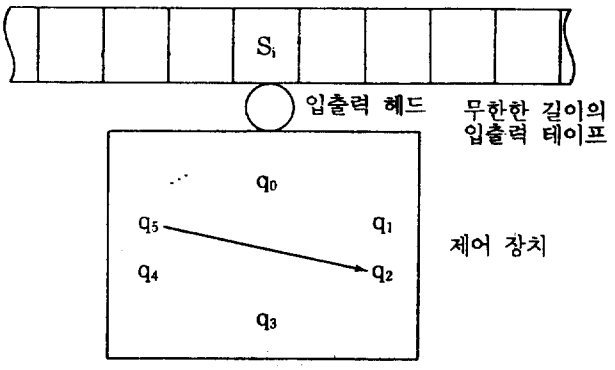


<그림 16 정규 언어, 문맥 무관 언어, 문맥 연관 언어 간의 관계>

이제 다음 절에서는 형식문법의 네 가지 유형 가운데 가장 강력한 언어 인식, 생성력을 가졌다는 무제한 문법에 대해 논의하겠다.

2.4 무제한 문법과 순환적 열거 가능 언어

촘스키의 형식문법 분류에 따라 타입 0에 해당하는 이 문법은 원래 정식 명칭이 없다. 하지만 일반적으로 무제한 문법(grammaires non-restreintes)이라 불린다. 이 무제한 문법은 이름 그대로, 앞에서 언급한 문법들과는 달리, 규칙에 아무런 제한이 없다. 무제한 문법의 언어 인식, 생성력은 영국의 수학자 튜링(Alan Turing: 1912-1954)이 고안한 튜링 기계(machine de Turing)와 동일하다. 이 튜링 기계는 독일의 수학자 힐베르트(David Hilbert(1928))가 제기한 의문 "수학적인 모든 문제를 순차적으로 해결할 수 있는 일반적인 기계적 절차가 존재하는가?"에 해답을 제시하기 위해 튜링에 의해 창안되었다. 튜링 기계는 무한한 기억용량을 가진 추상적인 계산기로서 가능한 계산은 모두 수행할 수 있다. 튜링 기계가 앞에서 살펴본 오토마타들과 다른 점은 무한한 용량의 기억장치가 있다는 점이다. 이처럼 튜링 기계는 무한한 기억용량을 전제로 하기 때문에 실재할 수 없는 추상적인 장치로서 그 구조를 그림으로 나타내면 다음과 같다.



<그림 17 튜링 기계의 구조>

다음은 튜링 기계의 정의이다.

튜링 기계의 정의

튜링 기계 T는

$T = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

로 정의되며, 각 기호는 다음을 뜻한다.

Q : 내부 상태의 유한 집합

Σ : 공백을 제외한 입력 기호 ($\Sigma \subseteq \Gamma - \{B\}$)

Γ : 기호들의 유한 집합

δ : 전이 함수

$q_0 \in Q$: 초기 상태

B : 공백(blank)을 표시하는 기호

$F \subseteq Q$: 최종 상태의 집합

자 이제 튜링 기계의 작동 과정을 보자. 다음과 같은 임의의 튜링 기계를 설정하자.

$Q = \{q_0, q_1\}$

$\Sigma = \{a, b\}$

$\Gamma = \{a, b, B\}$

$F = \{q_1\}$

$\delta(q_0,a) = (q_0, b, R)$

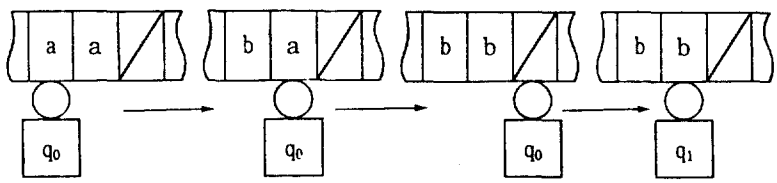
$\delta(q_0,b) = (q_0, b, R)$

$\delta(q_0,B) = (q_1, B, L)$

이 튜링 기계는 그 전이 함수 δ에 따라 다음과 같은 작동을 한다.

- $\delta(q_0,a) = (q_0, b, R)$: 제어 장치의 내부 상태가 q_0일때 입력 기호 a를 만나면 a를 b로 바꾸어 놓고 오른쪽(R)으로 한 칸 이동한다.
- $\delta(q_0,b) = (q_0, b, R)$: 제어 장치의 내부 상태가 q_0일때 입력 기호 b를 만나면 오른쪽(R)으로 한 칸 이동한다.
- $\delta(q_0,B) = (q_1, B, L)$: 제어 장치의 내부 상태가 q_0일때 공백(B)을 만나면 내부 상태를 q_1으로 바꾸어 왼쪽(L)으로 한 칸 이동한다.

따라서 이 튜링 기계에 예를 들어 aa라는 문자열이 입력되면 다음과 같은 작동 과정을 거쳐 이 문자열을 성공적으로 인식하고 bb라는 문자열을 출력한다.



<그림 18 튜링 기계의 작동 과정>

이처럼 튜링 기계가 최종 상태, 여기서는 q_1, 을 갖게 되면 입력 자료를 성공적으로 인식하는 동시에 적절한 출력을 내보내고 종료된다. 끝까지 최종 상태를 갖지 못하면 무한 루프에 있다고 하고 튜링 기계는 똑 같은 작동을 끝없이 되풀이한다.

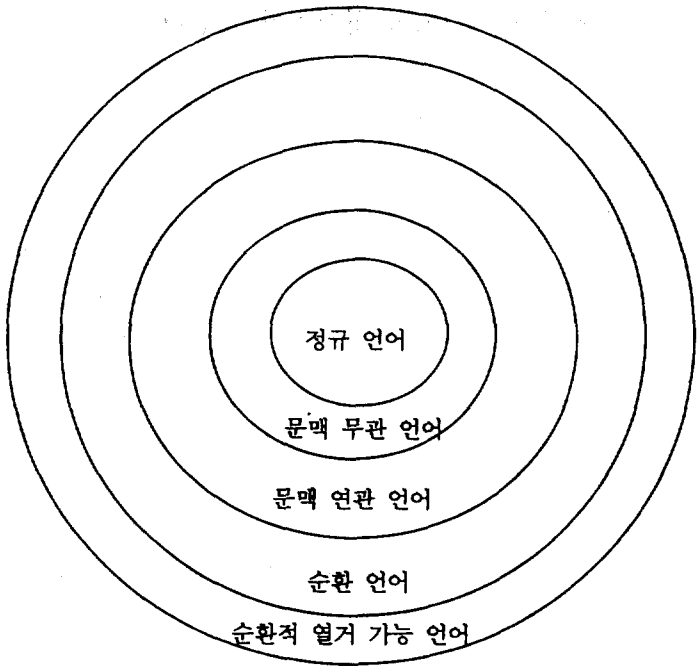
이 튜링 기계에 의해 인식, 생성되는 언어를 순환적 열거 가능 언어(langages récursivement énumérables)라 한다. 그러면 여기서 순환적 열거 가능 언어와 순환 언어(langages énumérables)에 대해 조금 더 자세히 살펴보자.

어떤 한 일련의 문자열의 집합 즉 한 언어에 대해 임의의 문자열이 이 언어에 속하는지 아닌지를 분명히 말해 줄 수 있는 알고리듬이 존재하면 우리는 이 언어를 순환 언어라고 한다. 이는 다시 말해 어떤 문자열이 소정의 언어에 속하는지 아닌지를 정확하게 나타내는 메시지를 출력하는 컴퓨터 프로그램이 존재할 수 있다면 이 언어는 순환적이라는 것이다. 반면에 어떤 한 언어에 대해 임의의 문자열이 이 언어에 속하면 속한다고 분명히 밝힐 수 있지만 이 언어에 속하지 않을 경우에는 속하지 않는다고 출력할 수도 있고 출력을 끌어내지 못한 채 알고리듬 수행을 중단할 수도 있고 혹은 아무런 대답도 없이 무한 루프에 빠져서 똑 같은 알고리듬 수행을 끝없이 되풀이 하는 그런 알고리듬이 존재하면 이 언어를 순환적 열거 가능 언어라 한다.

쉽게 풀어 얘기하면, "이 문장이 이 언어에 속하는 옳은 문장인가?"라는 질문에 그러면 그렇다, 아니면 아니다라고 분명히 밝힐 수 있는 알고리듬이 존재하면 이 언어는 순환 언어이고, 그럴 경우에는 그렇다라고 밝히지만 아닐 경우에는 꼭 아니다라는 출력을 내지 않는 그런 알고리듬이 존재하면 이 언어는 순환적 열거 가능 언어이다.

이런 이유로 순환적 열거 가능 언어를 설명해야 하는 알고리듬이 순환 언어를 설명해야 하는 알고리듬보다 부담이 적다고 할 수 있다. 또 모든 순환 언어는 순환적 열거 가능 언어인 반면에 모든 순환적 열거 가능 언어가 순환 언어인 것은 아니다. 왜냐하면 어떤 언어가 순환적이라고 확인할 수 있는 알고리듬은 자동적으로 이 언어가 순환적으로 열거 가능하다는 것을 보장할 수 있지만 그 역이 항상 성립하지는 않기 때문이다. 요컨대 순환 언어가 아니면서 순환적으로 열거 가능한 언어가 존재한다. 이는 다시 말해서 순환 언어는 순환적 열거 가능 언어의 진부분 집합이라는 것이다. 왜냐 하면 순환 언어는 이 언어에 속하는 문자열을 임의의 변별적인 기준에 따라 순서(이를 규범적 순서(ordre canonique)라 한다)대로 나열할 수 있기 때문이다.

지금까지 살펴 본 형식언어들 간의 상관관계는 다음과 같다.



<그림 19 형식언어들 간의 상관관계>

지금까지 살펴 본 네 가지 유형의 형식문법과 형식언어들을 표로 정리하면 다음과 같다.

	문법	언어	인식하는 기계	규칙의 형태
유형 0	무제한 문법	순환적 열거 가능 언어	튜링 기계	$\alpha \rightarrow \beta$ ($\alpha, \beta \in (S_T \cup S_N)^*$), α는 한개 이상의 비단말기호 포함
유형 1	문맥 연관 문법	문법 연관 언어	선형 한계 오토마타	$A\alpha B \rightarrow A\beta B$ ($A,B,\beta \in (S_T \cup S_N)^*$, $\alpha \in S_N$, $\beta \neq \varepsilon$)
유형 2	문맥 무관 문법	문맥 무관 언어	푸시다운 오토마타	$A \rightarrow \alpha$ ($\alpha \in (S_T \cup S_N)^*$, $A \in S_N$) 왼편에 오직 하나의 비단말 기호
유형 3	정규 문법	정규 언어	유한 상태 오토마타	$A \rightarrow \alpha B$, $A \rightarrow \alpha$, $A \rightarrow \varepsilon$ 혹은 $A \rightarrow B\alpha$, $A \rightarrow \alpha$, $A \rightarrow \varepsilon$ ($A,B \in S_N$, $\alpha \in S_T$)

S_N : 비단말기호 (symboles non-terminaux)
S_T : 단말기호 (symboles terminaux)

<표 1 네 가지 유형의 형식문법과 형식언어>

3. 여러 자연언어와 형식언어의 관계

지금까지 촘스키의 분류에 따른 네 가지 유형의 형식 문법과 형식언어 또 그들의 상관관계를 살펴보았다. 그러면 여기서 떠오르는 커다란 의문이 있다. 그것은 과연 불어는 어느 부류의 형식언어에 속하는가 하는 것이다. 정규 언어는 이를 인식, 생성할 수 있는 규칙의 형태가 가장 단순 명료하다는 장점은 있으나 앞에서 논의 한대로 불어의 모든 문장 구조를 인식, 생성하기에는 역부족인 것으로 판단되고 있으며 순환적 열거 가능 언어를 인식, 생성할 수 있는 무제한 문법은 가장 강력한 언어 인식, 생성력을 가지고 있음에도 불구하고 무한한 기억용량을 필요로 하기 때문에 제한된 기억용량밖에는 가지고 있지 않은 인간이 사용하는 자연언어를 다루는 데 있어서는 적당하지 않은 것으로 판단되고 있다. 따라서 불어를 일반적으로 문맥 무관 언어나 문맥 연관 언어로 보는 견해가 지배적이나 각각 상반된 장·단점이 있어서 학자들 간에 뜨거운 논쟁의 대상이 되고 있다. 형식언어와 불어의 관계에 대한 연구는 많이 행해지지 않았으나 불어와 같은 인구어에 속하는 다른 자연언어에 대해서는 상당한 논의가 진행되었다. 이들 자연언어가 문맥 무관 언어가 아니라는 주장은 Bar-Hillel과 Shamir(1964)와 Postal(1964)에서 비롯한다. 그 뒤를 이어 영어의 조동사로 시작되는 의문문을 전형적인 예로 간주하면서 이를 입증하려는 논의로는 Akmajian과 Heny(1975), Culicover(1976)가 있다. 특히 Culicover는 영어 의문사 의문문의 구조를 다루면서 "문맥 무관 구 구조 분석은 영어를 생성하기에는 충분하지 않다"고 결론지었다. Grinder와 Elgin(1973)도 주어와 동사의 일치조차도 문맥 무관 구 구조 문법(grammaires syntagmatiques indépendantes du contexte)의 규칙들로는 충분히 나타낼 수 없다고 하였으며 Bach(1974)와 Bresnan(1978)에서도 비슷한 논의를 찾아 볼 수 있다.

심지어 Selkirk(1977:285)는 "금세기의 언어학 이론의 발전에 있어서 가장 중요한 기여를 한 것을 하나만 고르라고 한다면 그것은 바로 자연언어 구조의 문법 모델로서 문맥 무관 구 구조 문법이 불충분하다는 것을 입증한 것이 될 것이다"라고 하였다. Winograd(1972), Langendoen(1975, 1977) 또 Pinker(1979)도 같은 견해를 표명하고 있다.

이와 같은 자연언어가 문맥 무관 언어가 아니라는 일반적이고도 뿌리 깊은 믿음에 근본적인 반론을 제기하면서 문맥 무관 문법으로는 설명할 수 없다고 간주되었던 $a^n b^n c^n$ 형의 문장 구조라든가 의문사, 관계절 등의 격리된 구조를 포함하는 문장들을 여러 예를 통해 문맥 무관 규칙들로 생성할 수 있다고 역설한 논의도 있는데, Gazdar(1981), Pullum과 Gazdar(1982), Pullum(1984)이 바로 그들이다. 이들은 모든 자연언어가 문맥 무관 언어라고 단정하는 것은 아니지만 자연언어가 문맥 무관 언어가 아니라고 주장했던 논의들이 근거 없는 착오라는 것을 보이는 데에 주력하였다.

이번에는 여기에 대해 만만치 않은 반론들이 속속 제기되었는데, 화란어의 문장 구조를 적용하여 자연언어가 문맥 무관 언어가 아니라는 논리를 전개한 것은 Bresnan, Kaplan, Peters와 Zaenen(1982)이다. 스위스 독일어의 예를 들어 이를 입증하려는 시도는 Shieber(1985)에서, 또 영어의 "such that" 구문을 적용하여 자연언어의 비문맥무관성을 규명하려는 논의는 Higginbotham(1984)에서 행해졌다.

자연언어가 문맥 무관 언어가 아니라는 것을 보이기 위해 지금까지의 논의에서 채택된 예들을 순서대로 정리해 보면 다음과 같다.

1. $a^n b^n c^n$ 형의 문장 구조 (Bar-Hillel과 Shamir(1964), Langendoen(1977))
2. 영어의 비교 구문 (Chomsky(1963))
3. 화란어의 부정형 동사 구문 (Huybregts(1976))
4. 영어의 "such that" 구문 (Higginbotham(1984))
5. 영어의 간접 의문 단축(間接 疑問 短縮, sluicing) 구문 (Langendoen과 Postal(1985))
6. 스위스 독일어의 예 (Shieber(1985)) 등

Savitch(1987)도 자연언어의 적절한 생성을 위해서는 문맥 연관 문법이 가장 적합하다는 견해를 문맥 무관 문법과 무제한 문법과의 비교를 통해 밝히고 있다.

이들이 이토록 단호하게 자연언어가 문맥 무관 언어가 아니라고 주장함에도 불구하고 그들이 예로 들은 구문이나 표현들은 문법적이기는 하지만 일상적인 언어사용에서 나타나는 평범하고 자연스러운 표현들이라기보다는 인위적으로 만들어진 듯한 논리 형식에 가까운 표현들임도 부인할 수 없다.

지금까지 논의 한대로 불어는 다른 인구어와 마찬가지로 문맥 무관 언어와 문맥 연관 언어 두 형식언어 중의 하나에 해당한다는 것이 일반적인 견해이다. 이 중 어느 유형의 형식언어에 해당하는지는 아직 밝혀지지 않은 채 언어학자들의 과제로 남겨져 있다.

III. 결 론

불어를 포함한 자연언어의 통사구조를 구성 성분 문법(grammaires des constituants)에 기반을 두고 규칙으로 나타낸 것이 다시쓰기 규칙이다. 이 규칙들은 유한한 개수의 규칙들로 무한한 수의 자연언어 문장들을 분석하고 생성할 수 있다는 점에서 그 시사하는 바가 크다.

그러나 다시쓰기 규칙은 그 형태가 다양해서 여러 상이한 유형의 문법을 구성함을 보았다. 이 문법들은 일반적으로 무제한 문법, 문맥 연관 문법, 문맥 무관 문법, 정규 문법으로 분류되고 각 문법을 구성하는 규칙의 형태와 각 문법들이 분석, 생성할 수 있는 언어의 형태가 다름을 보았다.

본고에서는 불어와 네 가지 유형의 형식문법으로 다루어질 수 있는 네 가지 유형의 형식언어간의 관계에 대하여 살펴보고자 하였다. 이를 위해 각 형식언어의 특성과 한계 그리고 이를 벗어나는 실제 불어 문장의 존재 유무를 따져 보았다.

먼저 정규 언어는 이를 인식, 생성할 수 있는 규칙의 형태가 가장 단순 명료하다는 장점은 있으나 앞에서 논의 한대로 불어의 모든 문장 구조를 인식, 생성하기에는 역부족인 것으로 판단되고 있으며 순환적 열거 가능 언어를 인식, 생성할 수 있는 무제한 문법은 자연언어를 다루는 데 있어서 필요 이상으

로 지나치게 강력하고 이 문법을 구성하는 규칙들에 아무런 제약이 없어서 이들을 일관성 있고 체계적으로 다루기가 어렵다. 뿐만 아니라 무제한 문법은 무한한 기억 용량을 필요로 하기 때문에 인간이 제한된 기억 용량 밖에는 가지고 있지 못하다는 점에서 가장 강력한 언어 인식, 생성력을 가지고 있음에도 불구하고 불어를 다루는 실제적인 적용에는 적당하지 않은 것으로 판단되고 있다. 따라서 불어를 일반적으로 문맥 무관 언어나 문맥 연관 언어로 보는 견해가 지배적이나 각각 상반된 장, 단점이 있어서 학자들 간에 뜨거운 논쟁의 대상이 되고 있다.

앞에서 이미 논의한 대로 불어에도 문맥 무관 언어를 벗어나는 $a^n b^n c^n$ 형의 문장이 엄연히 존재하는 것으로 미루어 볼 때 문맥 연관 언어에 속하는 것으로 간주하는 것이 합당해 보이나 n이 무한히 커지는 불어 문장을 접하기란 실제 불어 사용에서는 찾아보기 힘들다. 더욱이 문맥 연관 언어를 인식, 생성할 수 있는 문맥 연관 문법은 자연언어 자동처리 시스템의 프로그램 작성에 활용하기가 어려워서 문맥 무관 문법의 언어 인식, 생성력을 문맥 연관 문법의 수준으로 끌어올릴 수 있는 여러 가지 방법들이 고안되고 있다.

이런 이유로 불어는 문맥 무관 언어와 문맥 연관 언어 사이에서 어느 유형의 형식언어에 속하는지 아직 결정되지 못하고 있으며 이를 귀결 짓기 위해서는 앞으로도 많은 연구가 계속되어야 하며 설득력 있는 결론이 나오기까지는 아직도 상당한 시간이 필요하리라 생각된다.

참 고 문 헌

송 도규, 1994, 「범주에 대해서」, 『프랑스학연구』 제12권, 여 동찬교수 정년퇴임 기념논문집, 한국외국어대학교, 서울.

_____, 1996, 『인지언어학과 자연언어 자동처리』, 홍릉과학출판사, 서울.

신 영길 외, 1993, 『형식언어와 오토마타이론』, 상조사, 서울.

전 문석, 방 혜자, 1994, 『형식언어와 계산이론』, 홍릉과학출판사, 서울.

정 인정, 1993, 『오토마타와 계산이론』, 홍릉과학출판사, 서울, 다음 책의 역서, Hopcroft, John & Jeffrey D. Ullman, 1979, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Massachusetts.

Akmajian, Adrian & Frank Heny, 1975, *An Introduction to the Principles of Transformational Syntax*, MIT Press, Cambridge, Massachusetts.

Allen, James, 1987, *Natural Language Understanding*, The Benjamin/ Cummings Publishing Co.,Menlo Park, California.

Bach, Emmon, 1974, *Syntactic Theory*, Holt, Rinehart & Winston, New York.

BarHillel, Yehoshua & E. Shamir, 1964, 'Finite State Languages: Formal Representations and Adequacy

Problems', in Yehoshua Bar-Hillel, *Language and Information*, Addison Wesley, Massachusetts, pp. 87-98.

Berwick, Robert C., 1984, 'Strong Generative Capacity, Weak Generative Capacity, and Modern Linguistic Theories', in *Computational Linguistics, 10, 3-4*, Association for Computational Linguistics, New York, pp. 189-202.

Bresnan, Joan W., 1978, 'A Realistic Transformational Grammar', in M. Halle, J. W.Bresnan & G. A. Miller, *Linguistic Theory and Psychological Reality*, MIT Press, Cambridge, Massachusetts.

_____, 1982, *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, Massachusetts.

Bresnan, Joan W., Ronald M. Kaplan, Stanley Peters & Annie Zaenen, 1982, 'Cross-serial Dependencies in Dutch', *Linguistic Inquiry, 13, 4*, pp. 613-635.

Chomsky, Noam, 1959, 'On Certain Formal Properties of Grammars', *Information and Control, 2*, pp. 137-167.

_____, 1963, 'Formal properties of grammars', in R. D. Luce, R. R.. Bush & E. Galantier, *Handbook of Mathematical Psychology, II*, John Wiley & sons, New York, pp. 323-418.

Chomsky, Noam & George A. Miller, 1963, 'Introduction to the formal analysis of natural languages', in R. D. Luce, R. R.. Bush & E. Galantier, *Handbook of Mathematical Psychology, II*, John Wiley & sons, New York, pp. 269-321.

_____,1968, *L'analyse formelle des langues naturelles*, traduction de Chomsky(1963) et de Noam Chomsky & George A. Miller(1963), traduit par Ph. Richard & N. Ruwet, 2e tirage, 1971, Mouton/Gauthiers-Villars, Paris.

Culicover, Peter W., 1976, *Syntax*, Academic Press, New York.

Gazdar, Gerald, 1981, 'Unbounded Dependencies and Coordinate Structure', *Linguistic Inquiry, 12, 2*, pp. 155-184.

Gazdar, Gerald & Geoffrey K. Pullum, 1985, 'Computationally relevant properties of natural languages and their grammars', *New Generation Computing, 3*, pp. 273-306.

Grinder, J. T. & S. H. Elgin, 1973, *Guide to Transformational Grammar*, Holt, Rinehart & Winston, New York.

Gross, Maurice & André Lentin, 1967, *Notions sur les Grammaires Formelles*, Gauthier-Villars, Paris.

Higginbotham, James, 1984, 'English is not a context-free language', *Linguistic Inquiry, 15, 2*, pp. 225-234.

Hopcroft, John & Jeffrey D. Ullman, 1979, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Massachusetts, 정 인정에 의해 우리말로 번역, 1993, 「오토 마타와 계산이론」, 홍릉과학출판사, 서울.

Hughes, Michel, 1972, *Initiatin mathématique aux grammaires formelles*, Larousse, Paris.

Huybregts, M. A. C., 1976, 'Overlapping dependencies in Dutch' *Utrecht Working Papers in Linguistics, 1*, pp. 24-65.

Langendoen, D. Terence, 1975, 'Finite-state parsing of the phrase-structure languages and the status of readjustment rules in grammars', *Linguistic Inquiry*, 5, pp. 533-554.

_____,1977, 'On the inadequacy of type-2 and type-3 grammars for human languages', in P. J. Hopper, *Studies in descriptive and historical linguistics*, John Benjamin, Amsterdam, pp. 159-171.

_____,1981, 'The generative capacity of word-formation components', *Linguistic Inquiry, 12*, pp. 320-322.

Langendoen, D. Terence & Yedidyah Langsam, 1984, 'The representation of constituent structues for finite-state parsing', *COLING-84*, pp. 24-27.

Langendoen, D. Terence & Paul M. Postal, 1984, *The Vastness of Natural Languages*, Blackwell, Oxford.

_____,1985, 'English and the class of context-free languages', *Computational Linguistics, 10*, pp. 177-181.

Perrault, C. Raymond, 1984, 'On the Mathematical Properties of Linguistic Theories', *Computational Linguistics, 10, 3-4*, Association for Computational Linguistics, New York, pp. 165-176.

Pinker, Steven, 1979, 'Formal models of language learning', *Cognition, 7, 3*, pp.217-284.

Postal, Paul M., 1964, 'Limitations of Phrase Structure Grammars', in Jerry A. Fodor & Jerrold J. Katz, *The Structure of Langauge: Readings in the Philosophy of Language*, Prentice-Hall, Englewood Cliffs, pp. 137-151.

Postal, Paul M. & D. Terence Langendoen, 1984, 'English and the Class of Context-Free Languages', *Computational Linguistics, 10, 3-4*, Association for Computational Linguistics, New York, pp. 177-181.

Pullum, Geoffrey K., 1983, 'Context-freeness and the computer processing of human languages', *ACL Proceedings, 21st Annual Meeting*, pp. 1-6.

_____,1984, 'On Two Recent Attempts to Show that English is Not a CFL', *Computational Linguistics, 10, 3-4*, Association for Computational Linguistics, New York, pp. 182-186.

Pullum, Geoffrey K. & Gerald Gazdar, 1982, 'Natural Languages and Context-Free Languages', *Linguistics and Philosophy, 4, 4*, pp. 471-504.

Sabah, Gérard, 1988, *L'intelligence artificielle et le langage, volume 1, représentation des connaissances*, Hermès, Paris.

_____,1989, *L'intelligence artificielle et le langage, volume 2, processus de compréhension*, Hermès, Paris.

Savitch, Walter J., 1987, 'Context-Sensitive Grammar and Natural Language Syntax', in Walter J.

Savitch, Emmon Bach, William Marsh & Gila Safran-Naveh, *The Formal Complexity of Natural Language*, Reidel, Dordrecht, pp. 358-368.

Sebesta, Robert W., 1987, 'On Context-Free Programmed Grammars', in *Computer Languages, 14, 2*, pp. 99-108.

Selkirk, E. O., 1977, 'Some Remarks on Noun Phrase Structure', in Peter W. Culicover, Thomas Wasow & Adrian Akmajian, *Formal Syntax*, Academic Press, New York, pp. 285-316.

Shieber, Stuart M., 1985, 'Evidence against the context-freeness of natural language', *Linguistics and Philosophy, 8*, pp. 333-343.

Song, Do Gyu, 1993, *Formalisation et Simulation des Questions Partielles Directes ainsi que de Leurs Réponses en Français*, Thèse de Doctorat, Université de Provence, Aix-Marseille I, Aix-en-Provence.

Winograd, Terry, 1972, *Understanding natural language*, Academic Press, Edinburgh.

_____,1977, 'On some contested suppositions of generative linguistics about the scientific study of language', *Cognition, 5*, pp. 151-179.

_____,1983, *Language as a Cognitive Process, Volume 1: Syntax*, Addison-Wesley, Massachusetts.

≪ Résumé ≫

Etude sur la relation entre plusieurs types de langages formels et le français

SONG Do Gyu

C'est par des règles de réécriture que nous représentons la structure syntaxique des langues naturelles, y compris le français, ce sur la base des grammaires des constituants. Le fait que les règles, en nombre fini, peuvent analyser et générer des phrases naturelles au nombre infini revêt une grande importance.

Les formes de ces règles, cependant, sont si diverses qu'elles forment différents types de grammaires. Celles-ci sont des grammaires non-restrictives, des grammaires dépendantes du contexte, des grammaires indépendantes du contexte et des grammaires régulières.

Nous avons aussi vu que chaque type de grammaire est composé de différents types de règles de réécriture et que les langages qu'elles peuvent analyser et générer sont également différents.

L'objectif de cette étude a été de rechercher le type de langage formel auquel appartient le français et de vérifier quel type de grammaire formelle peut traiter cette langue naturelle sans aucune exception. Nous sommes allés vers cette même démarche dans le cas de l'anglais, du néérlandais et de l'allemand.

Examinons d'abord les langages réguliers. Tandis que les règles qui peuvent traiter ce type de langage sont les plus simples et les plus cohérentes, comme nous l'avons vu, il existe d'un autre côté des structures de phrases en français qui n'appartiennent pas à ce type de langage formel. Puisque les grammaires non-restrictives pouvant analyser et générer les langages récursivement énumérables nécessitent une mémoire illimitée, et que l'être humain n'en possède pas, ce type de grammaire n'est donc pas adéquat pour traiter le français malgré les grandes capacités de celui-ci quant à la reconnaissance et la génération des langages naturels. En conséquence, on considère, en général, le langage naturel soit comme un langage indépendant du contexte, soit comme un langage dépendant du contexte. Comme nous l'avons mentionné, les deux types de langage formel ont des qualités et des défauts. La préférence pour l'un ou l'autre fait actuellement l'objet d'une vive discussion.

Etant donné que la structure de phrase de la forme '$a^nb^nc^n$', qui dépasse la capacité de génération des grammaires indépendantes du contexte, existe toujours en français, il semble préférable de considérer ce dernier comme un type de langage dépendant du contexte. Mais il est presque impossible de trouver dans l'utilisation réelle du français des phrases correspondant à la formule '$a^nb^nc^n$'(le nombre de 'n' peut être infini). D'ailleurs, il est difficile d'exploiter les grammaires dépendantes du contexte qui peuvent traiter des langages dépendants du contexte dans le cadre du traitement automatique du langage naturel. Par conséquent, diverses alternatives sont proposées pour développer la capacité d'analyse et celle de génération des grammaires indépendantes du contexte par rapport à celle des grammaires dépendantes du contexte.

Conformément à ce qui a été dit précédemment, on peut affirmer ne pas être encore parvenu à conclure à quel type de langage formel, langage indépendant du contexte ou langage dépendant du contexte, appartient le français. Il s'en faut de beaucoup que nous arrivions à une conclusion sans controverses. Ceci exigerait des recherches beaucoup plus approfondies et beaucoup plus précises.