한국어 문장분석과 어휘정보의 연결에 관한 연구

최병진*†
목포대학교

Byung-Jin Choi. 2000. A Study of the Interface between Korean Sentence Parsing and Lexical Information. *Language and Information 4.2*, 55–68. The efficiency and stability of an NLP system depends crucially on how its lexicon is organized. The lexicon ought to encode linguistic generalizations and exceptions thereof. Nowadays many computational linguists tend to construct such lexical information in an inheritance hierarchy. DATR is good for this purpose. In this research I will construct a DATR-lexicon in order to parse sentences in Korean using QPATR. QPATR is implemented on the basis of a unification-based grammar developed in Düsseldorf. In this paper I want to show the interface between a syntactic parser(QPATR) and DATR-formalism representing lexical information. The QPATR parser can extract the lexical information from the DATR lexicon which is organised hierarchically. **(Mokpo National University)**

1. 서론

컴퓨터로 하여금 인간의 언어를 분석하여 이해하도록 하기 위하여 여러 문법이론들이 등장하였다. 그 중에서도 80년대 초부터 등장한 통합 기반 문법은 언어의 전산처리를 위하여 계속 발전해 나가고 있다. 이러한 발전 과정 속에서 주목할 만한 사실은 언어 분석에 필요한 문법이 어휘정보를 포함하는 사전에 포함되면서, 어휘정보의 구성과 관련된 사전의 역할이 점차로 중요시되고 있다는 것이다.

사전은 어휘정보와 언어학적인 일반 규칙성, 그리고 예외적인 현상들을 효율적으로 표상하고 있어야 하며, 이와 같이 표상된 사전을 중심으로 문법이론이 자연언어처리시스템에 구현될 때 그 시스템의 성능은 향상된다. 또한 최근 인지과학의 발달과 함께 인간의 뇌 속에서 지식이 어떠한 형태로 저장되어 있는지에 대한 관심과 이를 모델화 하고자 하는 노력 속에서 사전에 대한 연구는 인지과학, 인공지능의 지식 표상과도 밀접한 관련을 맺고 있다.

본 논문에서는 이러한 맥락에서 어휘정보를 계층구조 형태의 사전으로 구성하고, 각각 독립적으로 구축된 어휘정보가 실제로 문장분석을 하기 위해서 구문분석기와 어떻게 연결(interface)되어 사용될 수 있는지를 살펴본다. 이를 위해 PATR-II 문법형식을 바탕으로 하는 QPATR를 이용하여 한국어 문법 기술의 가능성을 살펴보고, 문장분석에 필요한 어휘정보를 DATR 형식으로 표상함으로써, 문법과 사전을 각각 독립된 형태로 기술한다. 그리고 각각 개별적으로 기술된 문법정보와 어휘정보가 컴퓨터로 문장을 분석하는 과정에서 서로 연결되어 사용될 수 있음을 구체적으로 제시한다.

* 534-729 전남 무안군 청계면 도림리 61 국립목포대학교 독일언어문화학과,
E-mail: bjchoi@apollo.mokpo.ac.kr
† 본 논문을 면밀히 검토하여 주시고 문제점을 지적해 주신 두 분 심사위원께 감사드린다.

2. 본론

2.1 QPATR의 구문분석 알고리즘

QPATR는 Shieber (1986)의 PATR-II 문법형식을 바탕으로 독일의 Düsseldorf대학에서 통합기반문법(unification-based grammars)을 위해 개발된 도구로, 언어학자가 문장분석을 위해서 프로그래밍을 하는 것이 아니라, 선언적으로 문법을 기술함으로써 문장분석을 가능하게 하는 도구이다. QPATR는 문장을 분석하는 방식에 있어서 상향식(bottom-up) 구문분석 방법과 하향식(top-down) 구문분석방법을 결합한 left-corner 구문분석방법을 채택하고 있다. 상향식 구문분석방법은 모든 범주 n_i $(1 \leq i \leq k)$[1]에 대해 부분 구조가 모두 발견된 경우에만 $n_0 \rightarrow n_1 \ldots n_k$이라는 규칙이 적용될 수 있는 특징이 있다. 그에 반해 left-corner 구문분석방법은 이미 n_1에 의해 지배를 받는 하나의 구조만 인식되기만 하면 n_1을 지배하는 범주와 연속되는 구성성분(자매마디)에 관한 가정을 형성하기 위하여 규칙이 적용된다. 이 때, 규칙의 왼쪽 구석(n_1)이 상향식으로, 규칙의 나머지 부분($\ldots n_k$)은 하향식 처리방식으로 이루어진다.

구체적으로 left-corner 구문분석 알고리즘(Naumann and Langer, 1994)을 살펴보면 다음과 같다.

우선 G= $< V_N, V_T, S, R >$[2]이 문맥자유 문법이라면, 임의의 문장 w가 L(G)의 문장인지를 분별하게 하기 위해서 우리는 3개의 STACK과 3가지 절차(Procedure)가 필요하다.[3]

STACK1: 분석할 문장에서 아직 처리되지 않은 부분
STACK2: 하향식(top-down)으로 인식될 범주($n_2 \ldots n_k$)
STACK3: 인식된 구성성분(n_0)

1. *procedure* REDUCE_{LC}

입력: first(STACK1)=n_1 인 STACK1
STACK2
STACK3
출력: pop(STACK1)
push($n_2 \ldots n_k$ t, STACK2)
push(n_0, STACK3)
수행조건:
1. 임의의 범주 n_0에 대해서 $n_0 \rightarrow n_1 \ldots n_k \in R$ 이거나 또는 $n_1 \in n_0$ 인 규칙이 하나 존재한다.
2. first(STACK2)= A, $[A \in (V_N \bigcup V_T)]$ 인 경우

STACK1에서의 첫 번째 기호가 $n_0 \rightarrow n_1 \ldots n_k$ 형태의 규칙에서 하부구조의 왼쪽 구석에 자리잡고 있는 기호와 일치하는 기호(n_1)가 삭제된다. 이를 제외한 오른쪽의 나머지 기호들은 V_T 이나 V_N에 속하지 않는 't'라는 기호와 함께 STACK2에 삽입된다. 't'라는 기호는 한 규칙의 끝을 의미하며 규칙에서 인식될 범주의 기호를 다른 규칙들로부터 분리하는 경계선의 역할을 한다.

1. 여기서 n은 다시 쓰기 규칙에 사용되는 종단 기호(terminal symbol)와 비종단 기호(non-terminal symbol)를 의미한다.
2. V_N: 비종단 기호의 집합, V_T: 종단 기호의 집합, S: 출발기호, R: 규칙(전이함수)
3. 여기서 사용되는 pop(STACK)은 STACK의 첫 번째 요소를 제거하는 것이며, push(Object, STACK)은 Object를 STACK의 첫 번째 요소로 삽입하는 것을 의미한다.

2. *procedure* $MOVE_{LC}$

입력: STACK1, STACK2, STACK3
출력: push(first(STACK3), STACK1)
pop(STACK2)
pop(STACK3)
수행조건:
1. first(STACK2) = t.
2. first(STACK3)= A, [A ∈ ($V_N \cup V_T$)] 인 경우

3. *procedure* $REMOVE_{LC}$

입력: STACK1, STACK2, STACK3
출력: pop(STACK1)
pop(STACK2)
STACK3
수행조건:
first(STACK1) = first(STACK2)

't'가 STACK2의 첫 번째 기호이기 때문에 규칙의 좌측부분에 A라는 기호를 지닌 규칙의 오른쪽 부분이 모두 처리되었다. 즉 유형 A의 구성성분이 인식된 것이다.

절차 $REMOVE_{LC}$에서는 STACK1과 STACK2의 첫 번째 기호가 동일하면, 이 기호를 각각 STACK1과 STACK2에서 삭제한다. 이 때 이 과정은 first(STACK2)가 규칙의 왼쪽 구석을 이루는 범주 n_i 이고 n_i 가 하향식으로 인식되었을 경우에만 적용된다.

이러한 절차를 바탕으로 [표 1]과 같이 문장인식 알고리즘을 정의할 수 있다.

*** 알고리즘**

데이터: 문맥자유문법 G= < V_N, V_T, S, R >, 문장 w= $w_1 ... w_n$ (n ≥ 1)
입 력: STACK1=[$w_1 ... w_n$]
STACK2=[S]
STACK3=[ ]

출 력: true / false
방 법: repeat
if <STACK1 = STACK2 = STACK3 = [ ] >
then <RETURN(true)>
else
if < 수행조건을 만족하는 P가 존재 (P ∈ {$REDUCE_{LC}$, $MOVE_{LC}$, $REMOVE_{LC}$ }) >
then 절차P(STACK1, STACK2, STACK3) 수행
else <RETURN(false)>

[표 1] left-corner 문장인식 알고리즘

이 알고리즘에 따라 다음의 문법과 규칙을 바탕으로 *john likes apples*라는 입력문장이 어떻게 분석되는지 그 과정을 추적하여 보면 [표 2]와 같이 타나낼 수 있다.

G= < V_N, V_T, S, R >는 문맥자유문법 입력문장: [john likes apples]

규칙(R): r1. S → NP VP r2. VP → V NP
r3. NP → john r4. NP → apples
r5. V → likes

STACK1	STACK2	STACK3	절차	적용규칙
시작				
[john likes apples]	[S]	[ ]	$REDUCE_{LC}$	r3
[likes apples]	[t, S]	[NP]	$MOVE_{LC}$	
[NP likes apples]	[S]	[ ]	$REDUCE_{LC}$	r1
[likes apples]	[VP, t, S]	[S]	$REDUCE_{LC}$	r5
[apples]	[t, VP, t, S]	[V, S]	$MOVE_{LC}$	
[V apples]	[VP, t, S]	[S]	$REDUCE_{LC}$	r2
[apples]	[NP, t, VP, t, S]	[VP, S]	$REDUCE_{LC}$	r4
[ ]	[t, NP, t, VP, t, S]	[NP, VP, S]	$MOVE_{LC}$	
[NP]	[NP, t, VP, t, S]	[VP, S]	$REMOVE_{LC}$	
[ ]	[t, VP, t, S]	[VP, S]	$MOVE_{LC}$	
[VP]	[VP, t, S]	[S]	$REMOVE_{LC}$	
[ ]	[t, S]	[S]	$MOVE_{LC}$	
[S]	[S]	[ ]	$REMOVE_{LC}$	
[ ]	[ ]	[ ]	→ true	성공

[표 2] *john likes apples*의 left-corner 구문분석과정

2.2 QPATR를 이용한 문법기술

QPATR를 이용하여 문장을 분석하기 위해서는 구구조규칙과 자질구조의 통합(unification)을 바탕으로 문법을 기술해야 한다.

다음은 QPATR에서 사용되는 구구조규칙의 일반 형태를 자질구조의 미명세 정보와 함께 나타낸 것이다.

X0 → X1 X2
<X_0 cat> = s
<X_1 cat> = np
<X_2 cat> = vp
<X_0 head> = <X_2 head>
<X_0 head subject> = <X_1 head>

우리가 2.1에서 *john likes apples.*라는 문장을 분석하기 위해서 사용했던 구구조규칙은 위의 형식을 빌어 개개의 구성성분에 대하여 제한을 줄 수 있도록 <문법 1>과 같이 자질구조의 통합을 포함하는 구구조규칙으로 나타낼 수 있다.

1. S → NP VP
   S/head *= VP/head,
   VP/subcat/first *= NP,
   VP/subcat/rest *= nil,
   NP/head/cas *= nom.
2. VP → V NP
   VP/head *= V/head,
   V/subcat/first *= VP/subcat/first,
   V/subcat/rest *= NP,
   V/subcat/rest/rest *= nil.
3. NP → 'john'
   NP/head/agr/num *= sg,
   NP/head/agr/pers *= 3,
   NP/head/agr/gen *= msc,
   NP/head/cas *= nom.
4. NP → 'apples'
   NP/head/agr/num *= pl,
   NP/head/agr/pers *= 3,
   NP/head/cas *= acc.
5. V → 'likes'
   V/head/agr/num *= sg,
   V/head/agr/pers *= 3,
   V/subcat/rest/first *= NP,
   NP/head/cas *= acc.
6. V → 'like'
   V/head/agr/num *= pl,
   V/head/agr/pers *= 3,
   V/subcat/rest/first *= NP,
   NP/head/cas *= acc.

<문법 1> 자질구조의 통합을 포함하는 구구조규칙

<문법 1>에서 규칙 1과 규칙 2는 추상적인 구성성분구조를 형성하고, 이 규칙에 의하여 어휘삽입을 위한 자리들이 마련된다. 규칙 3부터 규칙 6을 통하여 어휘삽입이 이루어지는데, 이러한 과정은 통합을 통하여 이루어진다. 규칙 2의 VP는 규칙 1의 VP와 통합하고, 여기에서 V ↔ VP ↔ S사이의 머리자질의 계승(inheritance)이 이루어진다. 여기서 규칙을 통한 어휘정보의 전달은 구문분석과정(parsing)을 통하여 자동적으로 이루어진다. QPATR를 이용하여 문장 분석을 하기 위해서 우리가 하여야 할 일은 바로 규칙 내에서 통합이 어떻게 이루어져야 할 지를 명시적으로 규정하는 일이며, 이러한 정의에 따라 어휘정보는 구문분석기를 통해서 자동적으로 전달된다. <문법 1>에서 자질통합의 미명세 정보를 포함하는 구구조문법을 바탕으로, QPATR의 구문분석기는 *john likes apples*라는 문장을 옳은 문장으로 인식하는데, 그 이유는 우리가 규칙 1에서 명사구 NP의 자질구조에 대하여 NP는 머리자질 '*cas:nom*(주격)'이라는 제한(constraints)을 설정하였기 때문이다. 이 제한에 의해서 우리는 사전목록에서 이 제한을 충족하는 NP로 *apples*가 아닌 *john* 이라는 단어만 선택할 수 있다.[4] 계속해서 동사의 하위범주화의 틀에서 *likes*는 하위범주화목록의 두 번째 요소가 '*cas:acc*(목적격)'이라는 머리자질을 가져야 한다는 제한을 보여준다.

이와 같이 개별 어휘에 미명세 정보로 정의된 제한은 규칙 1과 규칙 2에 의한 통합을 시도하면서 문법적인 문장을 판별하는 데 기여한다. 이 과정에서 정보전달과정은 구문분석기에 의해서 자동으로 이루어지지만, 정보전달방법은 선언적으로 정의되어야 하며, 이것이 바로 문법개발자가 담당해야 할 일이다. 이러한 점에서 이론 중심의 문법학자들이 컴퓨터를 통하여 자신의 이론을 검증하는 데 QPATR를 유용하게 사용될 수 있다.

또한 <문법 1>에서 문장주어에 대한 주격의 제한은 통사적인 차원에서 도입된 반면, 동사의 목적어에 대한 4격 제한은 어휘적인 차원에서 이루어졌다. 이러한 결정은 문법개발자에 의해 이루어진 것이며, 이러한 정의는 QPATR가 이론적으로 어떠한 제한도 받지 않으며, 따라서 이론적으로 가능한 문법의 모델설정을 위해 매우 다양한 가능성을 제시하고 있다고 볼 수 있다. 한 편, 언어학적으로 동기화된 제한 설정(constraints building)의 또 다른 가능성으로 우리는 주어와 동사간의 일치현상을 생각해 볼 수 있다.

7. NP → 'john'
   NP/head/agr/num *= sg,
   NP/head/agr/pers *= 3,
   NP/head/agr/gen *= msc,
   NP/head/cas *= acc.
8. NP → 'apples'
   NP/head/agr/num *= pl,
   NP/head/agr/pers *= 3,
   NP/head/cas *= nom.

4. 여기서 제시한 구구조규칙과 사전목록은 구문분석과정의 설명을 위해 의도적으로 단순화시켜서 제시하였다. 사전의 어휘목록을 확장함으로써 생겨나는 ***apples likes john*** 이나 ***john like apples*** 와 같은 비문법적인 문장의 해결을 위해 7쪽에서 계속해서 수정, 보완된다.

만일 앞에서 제시한 <문법 1>의 구구조규칙에 어휘항목(7, 8)을 추가한 후, *apples likes john*이나 *john like apples*라는 문장을 구문분석하여 보면 어떻게 될까? QPATR는 앞에서 정의한 구구조문법을 바탕으로 *john like apples*와 *apples likes john*이라는 문장을 모두 문법적인 문장으로 인식한다.[5]

개개의 어휘항목 *john*, *apples*, *like*, *likes*에 대해 수의 일치자질을 위한 속성(*head:agr:num*)이 정의되어 있으며, 또한 그 속성에 대해 각각 다른 속성 값을 가지고 있으므로 주어진 문장을 비문법적인 문장으로 인식해야 하는데, 그렇지 못하다. 그 이유는 '주어와 동사의 수의 일치에 대한 자질이 통합 가능해야 한다'는 제한이 그 어느 곳에도 명시화되어 있지 않기 때문이다. 이 문제의 해결을 위해 우리는 다음과 같은 정의를 <문법 1>의 1번 규칙에 추가함으로써, 주어-동사간의 수의 일치에 대한 제한을 설정할 수 있다.

NP/head/agr *= VP/head/agr

이 정의는 아주 단순하여 <문법 1>에서는 별로 큰 의미는 없어 보이지만, 나중에 사전 어휘항목을 확장하였을 경우 *I likes apples* 나 *They likes apples* 같이 불필요하게 생성되는 비문의 구조를 배제할 수 있다.

이밖에 문법기술에 있어서 주목할만한 점은 하위범주화목록의 구성이다. 규칙 1에서 VP의 하위범주화목록이 첫 번째 요소 뒤에서 바로 닫히었다. 우리는 한 눈에 이 정의가 단 하나의 주어를 가진 자동사의 문장만 분석 가능하게 한다는 것을 알 수 있다. 그러나 우리는 사전에 2개의 보족어를 필요로 하는 *likes*라는 타동사를 가지고 있다. 여기서 우리는 통합을 할 때에 모순에 부딪히는 것은 아닌가?

이에 대한 해답은 규칙 2에서의 미명세 정보 속에서 찾을 수 있다. 규칙 2의 정의를 자세히 살펴보면, V의 범주에 고유의 하위범주목록을 열어놓고 있으며, 그 첫 번째 요소가 VP의 하위범주목록의 첫 번째 요소와 공지표를 가지고 있다. 여기서는 완전한 목록이 통합되는 것이 아니라 단지 부분적인 정보만 통합이 이루어진다. 이러한 기술로 우리는 규칙 1을 각각 다른 하위범주화목록을 가지는 동사의 문장에도 적용할 수 있다.

2.3 한국어 문장분석

본 연구에서 중점을 두고 있는 것은 한국어 문장분석보다는 한국어 문장분석을 지원해 줄 수 있는 사전을 구성하고, 그 구성된 사전이 실제로 한국어 문장분석에 사용되어 문장의 의미를 정확히 표상해 줄 수 있는지의 가능성을 제시하는 것이다.

본 논문에서는 한국어 문장분석을 위해서 여러 가지 모델이 있겠으나, 이민행 (1994)의 구구조규칙을 수용, 이를 본 논문의 목적에 맞도록 약간 수정하여 QPATR로 다음과 같이 기술하였다.

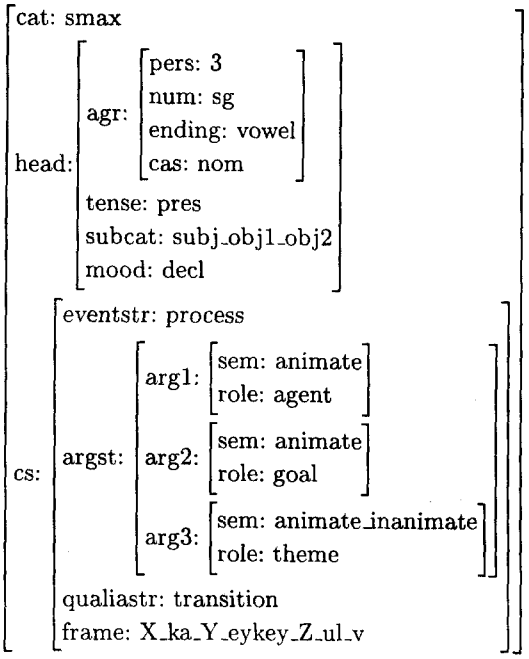
5. 여기서 정의한 문법에 의하면, *apples like john*이라는 문장도 문법적인 문장으로 인식한다. 그러나 이러한 문장도 의미선택제약에 대한 제한을 설정하면, QPATR에 의해 비문법적인 문장으로 판단된다.

```
1 # smax(SM) - - - > s(S), mood(M)::
    SM/head *= S/head,
    SM/mood *= M/mood.

2 # s(S) - - - > kp(KP), tp(TP)::
    S/head *= TP/head,
    TP/head/agr *= KP/head/agr,
    TP/subcat/first *= KP.

3 # kp(KP) - - - > n(N), k(K) ::
    KP/head *= N/head,
    N/head/agr *= K/head/agr.

4 # vp(VP) - - - > v(V)::
    VP/head *= V/head,
    VP/subcat *= V/subcat.

5 # tp(TP) - - - > vp(VP), t(T) ::
    TP/head *= VP/head.

6 # vp(VP) - - - > xp(XP), vp(VP1)::
    VP/head *= VP1/head,
    VP/subcat/first *= VP1/subcat/first,
    VP/subcat/rest *= XP,
    VP/subcat/rest/rest *= nil.

% 어휘부
minho   lex n(F)::
        F/head/agr/pers *= 3,
        F/head/agr/num *= sg,
        F/head/agr/ending *= vowel,
        F/cs/sem *= human,
        F/cs/trans *= minho_.

chayk   lex n(F)::
        F/head/agr/ending *= cons,
        F/cs/sem *= artifact,
        F/cs/trans *= book_.

ka      lex k(F)::
        F/head/agr/cas *= nom,
        F/head/agr/ending *= vowel.

ul      lex k(F)::
        F/head/agr/cas *= acc,
        F/head/agr/ending *= cons.

eykey   lex k(F) ::
        F/head/agr/cas *= dat,
        F/head/agr/ending *= vowel.

n       lex t(F)::
        F/head/tense *= pres.

sonye   lex n(F)::
        F/head/agr/pers *= 3,
        F/head/agr/num *= sg,
        F/head/agr/ending *= vowel,
        F/cs/sem *= human,
        F/cs/trans *= girl_.

ca      lex v(F)::
        F/head/subcat *= subj,
        F/cs/eventstr *= state,
        F/cs/argst/arg1 *= X,
        X/sem *= animate.

senmulha lex v(F)::
        F/head/subcat *= subj_obj1_obj2,
        F/cs/eventstr *= process,
        F/cs/argst/arg1 *= X,
        X/sem *= animate,
        X/role *= agent,
        F/cs/argst/arg2 *= Y,
        Y/sem *= animate,
        Y/role *= goal,
        F/cs/argst/arg3 *= Z,
        Z/sem *= animate_inanimate,
        Z/role *= theme,
        F/cs/qualiastr *= give_act,
        F/cs/frame *= 'X_ka_Y_eykey_Z_ul_v'.

ta      lex mood(F)::
        F/head/mood *= decl.
```

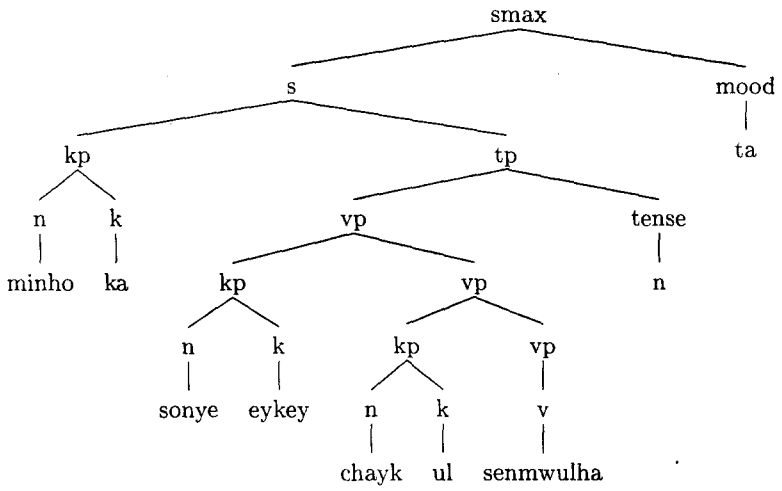
<문법 2> QPATR를 이용한 한국어 문법 기술

이와 같이 QPATR에 한국어 문장분석을 위한 사전항목과 구구조규칙을 정의하고, '민호가 소녀에게 책을 선물한다.'라는 문장을 입력하여 실제로 컴퓨터를 이용하여 분석을 해보면 [표 3]과 [표 4]의 결과를 얻게 된다.[6]

6. [표 3]의 자질 구조에서 cs 는 'conceptual structure'를 의미하며, 'event structure'나 'qualia'의 세부정보는 여기서는 편의상 간략하게 나타내기로 한다.



[표 3] 컴퓨터를 이용한 분석 결과: 자질구조



[표4] 컴퓨터를 이용한 분석 결과: 구성성분구조

여기에서 필자가 논하고자 하는 것은 바로 사전의 구성이다. 자연언어처리를 지원해 주기 위한 사전의 많은 정보가 잉여적(redundant)이라는 것을 우리는 알고 있다. 이러한 잉여적인 정보는 형태론, 통사론, 의미론적인 층위에서 모두 발견할 수 있다. Chomsky의 변형생성문법 이래로 사전의 잉여적인 정보를 최소화하고, 변별적인 정보를 중심으로 어휘정보를 표상하기 위한 연구가 지속적으로 이루어졌으며, 현재에는 계승메카니즘을

중심으로 계층구조적인 사전(hierarchical lexicon)을 구축하려는 연구가 주를 이루고 있다.

<문법 2>에서 하나의 문장에만 사용되는 단어의 어휘정보를 표상할 때에도 상당히 많은 정보가 중복되어 있음을 알 수 있다. 만일 이러한 어휘정보를 지닌 어휘항목이 늘어나게 되면, 우리는 개별 어휘항목에 대해서 잉여적인 어휘정보를 반복하여 정의하는 수고를 해야 한다. 그러나 계승메카니즘을 도입하여 사전을 구축한다면, 우리는 각 사전의 개별 층위(음운론, 형태론, 통사론, 의미론, 화용론 등)에 유형계층(type hierarchy)을 설정함으로써, 변별적이고 잉여적인 정보를 분류하여 각 유형에 적합하게 어휘정보를 정의할 수 있다.

사전구축에 대한 이러한 생각은 현재 대부분의 자연언어처리 시스템에 도입되어 구현되고 있다. QPATR시스템에서도 사전의 어휘정보를 계승메카니즘이 가능한 형태로 구현할 수 있다. 그러나 QPATR시스템 안에서 유형계층을 설정하여 어휘정보를 실제로 구축했을 때, 우리는 몇 가지 문제를 제고해볼 필요가 있다.

우선 QPATR의 계승메카니즘은 단조적(monotonic)이므로 상위유형과 하위유형사이에 모순되는 내용이 있을 경우, 정보의 계승이 이루어질 수 없다. 이를 보완하기 위해서 우리는 부수적인 절차를 추가로 정의할 필요가 있다. 따라서 유형계층과 어휘항목을 위한 매크로를 정의해야 하며, 사전의 어휘정보 구축은 자연히 복잡해진다. 또한 사전을 개발할 때에 어휘정보정의를 위한 속성의 설정에 있어서, 대부분의 자연언어처리 시스템에서와 마찬가지로 QPATR시스템에 전적으로 의존해야 한다. 즉, 사전 개발자가 QPATR시스템의 구조와 문법개발자가 설계한 문법을 충분히 파악하고 있어야 하므로 어휘정보의 구축을 위한 시간과 노력이 필요 이상으로 소요된다. 이러한 이유로 QPATR시스템 안에서 계층구조적인 사전을 구축하는 것은 조금은 복잡한 작업이라고 할 수 있다.

한편 인공지능 분야에서 다양한 지식표상언어가 개발되었는데, 그 중에서도 비단조적인 해석(non-monotonic interpretation)을 가능하게 하는 지식표상언어로 DATR라는 지식표상언어가 있다. DATR는 1989년에 R. Evans와 G. Gazdar에 의해 개발되었는데, 비단조 계승메카니즘을 가능하게 하는 지식표상언어로 인공지능에서 필요로 하는 여러 가지 지식을 표상할 수 있을 뿐만 아니라, 사전지식을 구축하는 데 아주 유용하게 쓰일 수 있다.

DATR를 이용하여 사전의 어휘정보를 구축할 경우, 사전을 독립적으로 개발하는 것이 가능하다. 즉, 문법 개발자와 사전 개발자의 역할이 명확히 구분될 수 있다. 또한 사전의 어휘정보를 구축할 때에도 언어학의 여러 층위의 지식들이 각각 개별적으로 구축될 수 있으며, 이렇게 독립적으로 구축된 정보는 쉽게 통합할 수 있어서, 어휘정보를 효율적으로 구축할 수 있다.

이러한 이유로 본 논문에서는 한국어 문장분석을 지원해 주는 사전을 DATR라는 지식표상언어로 표상하고 사전의 어휘정보가 어떻게 QPATR라는 독립된 구문분석기와 연결되어 성공적으로 문장을 분석하는지를 보여 주고자 한다.

2.4 DATR를 이용한 어휘정보의 표상

앞에서 제시한 "민호가 소녀에게 책을 선물한다."라는 문장을 분석하기 위하여, 필요한 어휘항목의 정보를 DATR를 이용하여 <표 5>와 같이 나타낼 수 있다. 우리는 <표 5>에서 개별 어휘항목의 어휘정보에 포함되어 있는 내용이 다분히 잉여적인 것을 알 수 있다. 이러한 잉여 정보는 어휘항목을 지배하고 있는 상위 범주에 정의될 수 있다. 따라서 개별 어휘항목에는 변별적이고 고유한(idiosyncratic) 정보만을 정의하면 된다.

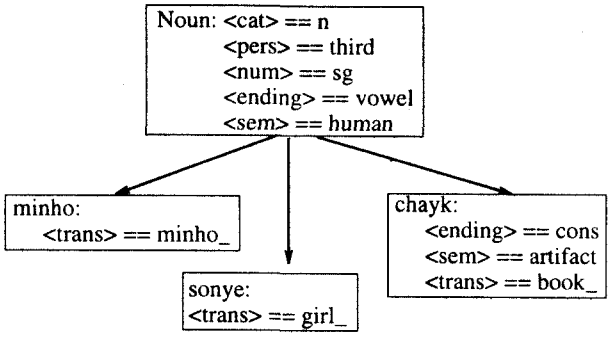
MINHO	<cat> == n	SONYE	<cat> == n
	<pers> == third		<pers> == third
	<num> == sg		<num> == sg
	<ending> == vowel		<ending> == vowel
	<sem> == human		<sem> == human
	<trans> == minho_.		<trans> == girl_.
CHAYK	<cat> == n	CA	<cat> == vi
	<ending> == cons		<type> == verb
	<sem> == artifact		<subcat> == one
	<trans> == book_.		<event> == process
			<arg1> == animate.
KA	<cat> == postpos	SENMULHA	<cat> == vt
	<type> == case		<event> == process
	<cas> == nom		<arg1 sem> == animate
	<ending> == vowel		<arg1 role> == agent
I	<cat> == postpos		<arg2 sem> == animate
	<type> == case		<arg2 role> == goal
	<cas> == nom		<arg3 sem> == animate_inanimate
	<ending> == cons.		<arg3 role> == theme
LUL	<cat> == postpos		<qualiastr> == send
	<type> == case		<frame> == 'X_ka_Y_eykey_Z_ul_v'
	<cas> == acc		<comp> == three.
	<ending> == vowel.	N	< > == TENSE
EYKEY	<cat> == postpos		<tense> == pres.
	<type> == case	DA	< > == MOOD
	<cas> == dat		<mood> == decl.
	<ending> == vowel.		

[표 5] DATR를 이용한 사전 어휘 정보

[표 5]의 개별 어휘항목에 반복되어 정의된 어휘정보에 대하여, 상부계층에 어휘범주를 설정하고 정보를 공유하게 함으로써 우리는 어휘정보를 단순화 할 수 있다. 즉 어휘항목 '민호', '소녀', '책'에 대하여 우리는 '명사'라는 어휘범주를 상위유형으로 설정하고, 이 명사범주에 *<pers>*, *<num>*, *<ending>*, *<sem>*의 속성과 해당 값을 정의할 수 있다.[7]

[그림 1]에서 '민호'와 '소녀'에 대해서는 단지 의미표상(trans)에 대한 정보만을 정의하여 주고, 인칭, 수, 어미, 의미자질에 대한 정보는 모두 상위범주인 '명사'로부터 물려받는다는 것을 알 수 있다. 반면에 어휘항목 '책'은 종결어미(ending)와 의미자질(sem)에 대하여 상위범주와 모순된 정보를 지니기 때문에 하위범주에 개별 어휘항목 특유의 정보를 정의한다.

7. 여기서 '인칭(pers)'이나 '수(num)'의 속성-값을 잉여적인 것으로 명사범주에 정의하는 것은 큰 문제가 없으나, '어미(ending)'나 '의미자질(human)'에 대한 속성-값을 상위범주에 설정하는 데는 문제제기가 있을 수 있다. 이러한 문제는 어휘정보의 경제성을 고려할 때, 무엇을 중요시하느냐는 관점의 차이에서 생겨날 수 있다.



[그림 1] 어휘항목의 유형계층

이러한 정의가 가능한 이유는, 비단조 논리(non-monotonic logic)[8]의 해석이 가능한 비단조 결장치 계승(non-monotonic default inheritance)[9]에서는 예외적인 정보가 일반적인 정보보다 더 특수하게 간주되어 우선권을 가지기 때문이다. 즉, 계승의 관계에서 특수하게 정의된 정보와 일반적인 정보사이에 논리적 불일치가 생기는 경우, 특수하게 정의된 정보가 우선적으로 선택되며, 그 밖의 경우에 대해서는 자동적으로 상위계층으로부터 정보를 그대로 물려받게 된다. 계층구조에서의 위치가 아래쪽 깊은 곳에서 나타날수록, 그 정보는 보다 특수하게 정의된 것이다.

우리는 [표 5]의 어휘항목에 대하여 '격조사'와 '동사', '자동사', '타동사' 등의 문법범주를 설정할 수 있으며, 그 밖에 어휘정보의 정의에 필요한 '하위범주화(SUBCAT)', '머리자질(HEAD)', '논항구조(ARGSTR)', '사건구조(EVENTSTR)' 등의 메타범주를 유형계층에 설정할 수 있다.

이러한 계승메카니즘에 기반하여 [표 5]의 어휘부를 DATR로 표상하면 [표 6]과 같이 나타낼 수 있다.[10]

8. 비단조논리의 자세한 설명은 Reiter (1980) 참조.
9. 비단조 계승메카니즘은 일반적인 현상과 예외 현상, 그리고 예외 현상에 대한 예외 현상을 표현하는 데 아주 적합하다. 우리는 이러한 메카니즘을 통해 언어 현상의 일반규칙화에 있어서 'blocking'의 문제들을 명확하게 설명할 수 있고 컴퓨터로 구현할 수 있다.
10. DATR이론의 구체적인 내용은 Evans and Gazdar (1990)와 Barg (1996) 참조

NOUN:	< > == LEXEM	VERB:	< > == LEXEM
	<pers> == third		<type> == verb
	<num> == sg		<pers> == third
	<ending> == vowel		<num> == sg
	<sem> == human		<comp> == one.
	<type> == noun		
	<cat> == n.	DI_TR_VERB:	< > == INTR_VERB
INTR_VERB	< > == VERB		<cat> == vt
	<cat> == vi.		<comp> == two.
MINHO	< > == NOUN	SONYE	< > == NOUN
	<trans> == minho_		<trans> == girl_
CHAYK	< > == NOUN	CA	<cat> == INTR_VERB
	<ending> == cons		<event> == process
	<sem> == artifact		<arg1> == animate.
	<trans> == book_		
KAS	< > == LEXEM	SENMULHA	< > == DI_TR_VERB
	<cat> == postpos		<event> == process
	<type> == case		<arg1 sem> == animate
KA	< > == KAS		<arg1 role> == agent
	<cas> == nom		<arg2 sem> == animate
	<ending> == vowel		<arg2 role> == goal
I	< > == KAS		<arg3 sem>
			== animate_inanimate
	<cas> == nom		<arg3 role> == theme
	<ending> == cons		<qualiastr> == send
			<frame>
			== 'X_ka_Y_eykey_Z_ul_v'
EUL	< > == KAS		<comp> == three.
	<cas> == acc		
	<ending> == cons.	N:	< > == TENSE
EYKEY	< > == KAS		<tense> == pres.
	<cas> == dat	DA:	< > == MOOD
	<ending> == vowel.		<head> == decl.

[표 6] 유형 계층에 의한 DATR의 어휘정보 표상

한편 계승메카니즘이 허용되는 경우, 하나의 어휘장(lexical field)에서 유사한 의미를 지니는 단어들에 대해, 의미상으로 개별 단어들을 포함하는 상위어에 어휘정보를 정의함으로써 사전을 효율적으로 구성할 수 있다. 즉 우리가 '선물하다'와 비슷한 의미를 지닌 단어로서 '주다'라는 어휘항목의 정보를 정의해야 할 경우, 우리는 먼저 '주는 행위(GIVE_ACT)'를 나타내는 동사에 대한 상위범주를 설정하여, 주는 행위에 대한 일반적인 정의를 할 수 있다. 그리고 "주다:< > == 주는 행위"와 같은 형태로 정의를 하여 줌으로써 "주다"라는 동사의 모든 일반적인 어휘정보는 "주는 행위"라는 상위계층으로부터 물려받게 하고, 그 외에 "주다"라는 동사가 지니는 독특한 의미만을 추가로 정의하면 된다.[11]

2.5 QPATR와 DATR의 연결

우리는 앞에서 DATR를 이용하여 사전을 구축할 때에 사전을 효율적으로 구축할 수 있으며, 문법개발과 구분하여 독립적으로 사전을 개발할 수 있다는 것을 보았다. 그런데 이와 같이 독립적으로 구축된 사전이 QPATR를 이용하여 문장분석을 하는 데 이용되기 위해서는 [표 5]의 어휘항목의 형태에서 <문법 2>의 어휘부에서 제시하고 있는 어휘항목의 형태로 바뀌어야 한다. 그러기 위해서는 <문법 2>의 어휘항목에 속성-값을 위해

11. 어휘장을 중심으로 하는 단어들의 개별정의에 대한 내용은 앞으로의 연구과제로 제시하고자 한다.

정의되어 있는 경로(예: F/head/agr/ending *= vowel)를 DATR사전에서도 생성해 낼 수 있도록 정의하여야 한다.

QPATR에서 정의된 미명세 정보(예: F/head/agr/ending *= vowel)는 자질구조를 나타낸다고 2.2에서 언급한 바 있다. 즉 QPATR에서 어휘정보 "F/head/agr/ending *= vowel"은 자질구조 "[head:[agr:[ending: vowel]]]"를 의미하는 것이다.

이러한 점을 고려하여, DATR에서 어휘정보를 생성할 때에, 어휘정보 뿐 아니라, 어휘정보를 위한 속성-값의 경로도 자질구조로서 함께 생성하게 하면, DATR의 어휘항목은 QPATR의 어휘항목과 동일한 기능을 하면서 문장분석을 지원하여 준다.

QPATR에서 사용하고 있는 속성-값의 경로를 자질 구조로 나타내기 위해서는 DATR에서 다음과 같은 정의가 필요하다.[12]

```
LEXEM:      <> == ([ CAT , HEAD ]).
CAT:        <> == ( cat ':' "<cat>").
HEAD:       <> == (head ':' <value "<type>" > )
            <value noun> == ([ AGR ] , CS )
            <value case> == ([ AGR ] )
            <value verb> == ( [ SUBCAT:<> ] , CS )
            <value> == ().
AGR:        <> == ( agr ':' [ PER , NUM , ENDING ] )
            <value case> == (agr ':' [ CAS:<> , ENDING:<> ] ).
PER:        <> == (per ':' "<pers>").
NUM:        <> == (num ':' "<num>").
ENDING:     <> == (ending ':' "<ending>").
CAS:        <> == (cas ':' "<cas>" ).
TENSE:      <> == ([ head ':' [ tense ':' "<tense>" ] ] ).
MOOD:       <>== ([ head ':' [ mood ':'"<mood>" ] ] ).
CS:         <value noun> == ( cs ':' [ SEM , TRANS ])
            <value verb> == ( cs ':' [ EVENTSTR:<>, ARGSTR:<> ])
            <value> == ().
SEM:        <value noun> == ( sem ':' "<sem>" ).
TRANS:      <value noun> == ( trans ':' "<trans>" ).
EVENTSTR:   <> == (eventstr ':' "<event>").
ARGSTR:     <> == (argstr ':' [ ARG1 <case "<comp>" > ])
            <case two> == (, ARG2:<>)
            <case three> == (, ARG2:<>, ARG3:<>)
            <case> ==().
ARG1:       <> ==( arg1 ':' [ SEM1 , ROLE1 ]).
ARG2:       <> ==( arg2 ':' [ SEM2 , ROLE2 ]).
ARG3:       <> ==( arg3 ':' [ SEM3 , ROLE3 ]).
SEM1:       <> == (sem ':' "<arg1 sem>" ).
ROLE1:      <> == (role ':' "<arg1 role>" ).
SEM2:       <> == (sem ':' "<arg2 sem>" ).
ROLE2:      <> == (role ':' "<arg2 role>" ).
SEM3:       <> == (sem ':' "<arg3 sem>" ).
ROLE3:      <> == (role ':' "<arg3 role>" ).
QUALIASTR:  <> == (qualiastr ':' "<qualia>").
SUBCAT:     <> == ( subcat ':' < "<comp>" > )
            <none> == none
            <one> == subj
            <two> == subj_obj
            <three> == subj_obj1_obj1
            <unspecified> == ( [ ] ).
```

12. DATR사전을 이용하여 앞의 예문을 분석한 결과는 [표 3], [표 4]의 결과와 동일하다.

3. 결론

지금까지 우리는 한국어 문장을 분석하기 위한 문법을 QPATR로 기술하고 이를 지원해 주는 사전을 DATR로 구축함으로써 구문분석과 어휘정보가 어떻게 연결되어 문장을 분석하는지를 개략적으로 살펴보았다. 언어학자의 문법적인 지식이 QPATR에서 어떻게 이용될 수 있으며, 계승메카니즘을 이용한 사전 구축의 방법이 매우 효율적임을 확인하였고, QPATR의 구문분석기와 DATR의 사전 어휘정보가 연결되어 문장을 분석하고, 그 결과를 언어 보편적인 구조로 표상할 수 있음도 확인하였다. 본 논문에서 언급되지 않은 내용, 즉, 문장입력 전의 형태소 해석 작업이나, 문장분석 후의 의미표상결과는 각각 모듈작업을 통하여 각각 다른 작업 파일로 저장되어 계속해서 다른 모듈에서의 언어처리(텍스트 이해, 기계번역 등)에 이용될 수 있다.

본 논문에서는 일반 언어학적인 이론을 근거로 하여 컴퓨터로 국어 문장을 분석할 수 있는 가능성을 보여줌으로써 전산언어학에 근접할 수 있는 통로를 보여주는 것이 하나의 작은 바램이다. 앞으로 이러한 시스템을 가지고 실제로 문법의 이론적인 부분을 기계적으로 검증하면서 시험해 볼 수 있다면, 국어 정보처리의 발전에 기여할 수 있으리라 확신한다.

끝으로 DATR를 이용하여 생성어휘론적인 관점에서 동사를 중심으로 대량의 어휘정보를 구축하는 일은 앞으로의 과제로 남겨둔다.

참고 문헌

이민행. 1994. *국어와 독일어의 대조통사론과 기계번역*, 제 51집. 독일문학.

Barg, Petra. 1996. *Automatischer Erwerb von linguistischem Wissen: ein Ansatz zur Inferenz von DATR-Theorien.* Niemeyer, Tübingen.

Evans, R. and G. Gazdar. 1990. The DATR papers. In *Cognitive Science Research Paper(CSRP)*, volume 139, Brighton. University of Sussex.

Naumann, S. and H. Langer. 1994. *Parsing.* B.G.Teubner, Stuttgart.

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence*, 13:81–132.

Shieber, S. M. 1986. An introduction to unification-based apporaches to grammar. *CSLI Lecture Notes*, 4.

접수일자: 2000년 10월 8일
게재결정: 2000년 12월 2일