한국어 문장분석의 생성 어휘론적 접근

최병진 (목포대)

요약

본 논문에서는 컴퓨터를 이용하여 문장을 분석하기 위해 통합기반문법이 구현된 PATR라는 문법형식을 살펴보고, 국어문장분석을 지원해 주는 사전을 계승메카니즘이 가능한 형태로 구축하여 사전의 효율적인 구성을 제시하고, 사전과 구문분석기사이의 인터페이스가 어떻게 가능할 수 있는 지를 보여주고자 한다.

키워드: 구문분석, 의미표상, 전자사전, 계승 메카니즘

I. 서론

컴퓨터로 하여금 인간의 언어를 분석하여 이해하도록 하기 위하여 여러 문법이론들이 등장하였다. 그 중에서도 80년대 초부터 등장한 통합 기반 문법은 언어의 전산처리를 위해서 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 있다.

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. 이러한 맥락에서 자연언어시스템에서의 사전의 역할은 아무리 강조해도 지나치지 않다.

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 PATR-Ⅱ문법형식을 바탕으로 하는 QPATR를 이용하여 한국어 문장 분석의 가능성을 제시한 후 문장분석을 지원하는 사전의 구축을 최근 주목을 받고 있는 생성어휘론적인 입장에서 제시하고자 한다.

Ⅱ.본론

II.1. Left-Corner구분분석 알고리즘

QPATR는 Shieber(1996)의 PATR-Ⅱ문법형식을 바탕으로 독일의 Düsseldorf대학에서 개발된 통합기반문법(Unification-Based Grammars) 개발을 위해 일반적으로 사용할 수 있는 도구로, 언어학자가 단순히 문법을 기술함으로써 문장을 분석하는데 사용할 수 있는 도구이다. QPATR를 이용하여 문장을 분석하는 방법을 소개하기 전에 먼저 QPATR에서 채택된 구문분석 알고리즘을 살펴보도록 한다.

QPATR는 bottom-up구문분석 방법 및 top-down구문 분석 방법을 결합한 Left-corner구분분석 방법을 채택하고 있다. Bottom-up 구문분석방식은 모든 범주 n_i $(1 \le i \le k)$[1]에 대해

부분 구조가 모두 발견이 된 경우에만 $n_0 \rightarrow n_1...n_k$ 이라는 규칙이 적용될 수 있는 특징이 있다. 그에 반해 Left-corner 구문분석방법은 이미 n_1 에 의해 지배를 받는 하나의 구조만 인식되기만 하면 n_1을 지배하는 범주와 연속되는 구성성분(자매마디)에 관한 가정을 형성하기 위하여 규칙이 사용된다. 규칙의 왼쪽 구석이 bottom-up으로, 규칙의 나머지 부분은 top-down으로 이루어진다고 말할 수 있다. 구체적으로 left-corner 구문 분석 알고리즘 (Naumann/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 인식하기 위해서 우리는 3개의 STACK과 3가지 절차(Procedure)가 필요하다.

STACK1 : 분석할 문장에서 아직 처리되지 않은 부분
STACK2 : top-down으로 인식될 범주($n_2...n_k$)
STACK3 : 인식된 구성성분(n_0)

1. *procedure* $REDUE_{LC}$
수행조건:
1. 임의의 어휘범주 n_0에 대해서 $n_0 \rightarrow n_1....n_k \in R$ 이거나 또는 $n_1 \in n_0$ 인 규칙이 하나 존재한다.
2. first(STACK2) $\in (V_N \cup V_T)$ 인 경우

입력: first(STACK1)=n_1 인 STACK1
STACK2, STACK3
출력: pop(STACK1)[1]
push($n_2...n_k$ n_1 t, STACK2)
push(n_0, STACK3)

STACK1에서의 첫 번째 기호가 $n_0 \rightarrow n_1...n_k$ 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.

2. *procedure* $MOVE_{LC}$
수행조건:
1. first(STACK2) = t.
2. first(STACK3) = A, [A $\in$($V_N \cup V_T$)]

입력: STACK1, STACK2, STACK3
출력: push(first(STACK3, STACK1)
pop(STACK2)
pop(STACK3)

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

3. *procedure* $REMOVE_{LC}$
수행조건:
first(STACK1) = frist(STACK2)

입력: STACK1, STACK2, STACK3
출력: pop(STACK1)
pop(STACK2)
STACK3

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 top-down으로 인식되었을 경우에만 적용된다.

---

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)

이를 바탕으로 우리는 인식알고리즘을 다음과 같이 정의 할 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 인식 알고리즘

이 알고리즘에 의해 실제로 "john like apples"라는 문장이 어떻게 분석되는지 그 과정을 살펴보면 표 1과 같다.

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

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

STACK1	STACK2	STACK3	절 차	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]	$REDUCE_{LC}$	
[ ]	[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 구분분석 과정

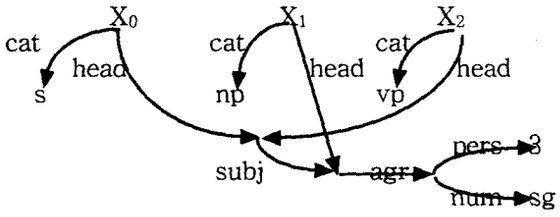
II.2. QPATR를 이용한 문법기술

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 것으로 이러한 형태는 그래프의 형태로 나타낼 수 있다.

$X_0 \rightarrow X_1\ X_2$
$\langle X_0\ cat\rangle = s$
$\langle X_1\ cat\rangle = np$
$\langle X_2\ cat\rangle = vp$
$\langle X_0\ head\rangle = \langle X_2\ head\rangle$
$\langle X_0\ head\ subject\rangle = \langle X_1\ head\rangle$



우리가 Ⅱ.1에서 "John likes apples"라는 문장을 분석하기 위해서 사용했던 구구조규칙은 위의 형식을 빌어 개개의 구성성분에 대하여 제한을 줄 수 있도록 자질구조의 통합을 포함하는 다음과 같은 구절구조규칙으로 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 *= Obj,
   Obj/head/cas *= acc.

5. V → 'like'
   V/head/agr/num *= pl,
   V/head/agr/pers *= 3,
   V/subcat/rest/first *= Obj,
   Obj/head/cas *= acc.

규칙 1과 규칙 2는 추상적인 구성성분구조를 형성하고, 이것을 통해서 어휘삽입을 위한 자리들이 마련되어진다. 규칙 3-5를 통하여 어휘삽입이 이루어지는데, 이러한 과정은 통합을 통하여 이루어진다. 규칙 2의 VP는 규칙 1의 VP와 통합하고, 여기에서 V ↔ VP ↔ S사이의 머리자질의 계승이 이루어진다.

규칙을 통한 정보의 전달은 구문분석과정(parsing)을 통하여 자동적으로 이루어진다. QPATR를 이용하여 문장 분석을 하기 위해서 우리가 해야할 일은 바로 규칙 내에서 통합이 어떻게 이루어져야 할 지 명시적으로 규정하는 일이며, 이러한 정의에 따라 어휘정보는 구문분석기를 통해서 자동적으로 전달되어진다. 자질 통합과 함께 확장된 구구조문법을 통하여 *"Apples likes John."*보다는 *"John likes apples"*를 옳은 문장으로 허용하는데, 그 이유는 우리가 규칙 1에서 명사구 NP의 자질구조에 대하여 NP는 머리자질 *'cas:nom'* 라는 하나의 제한을 설정하였기 때문이다. 이 제한에 의해서 우리는 사전목록에서 이 제한을 충족하는 NP로 *"apples"* 가 아닌 *"john"* 이라는 단어만 선택할 수 있다. 계속해서 동사의 하위범주화틀에서 *"likes"*는 하위범주화목록의 두 번째 요소가 *'cas:acc'* 라는 머리자질을 가져야 한다는 제한을 보여준다.

여기서 정보전달의 방법이 언어학자에게 중요하다. 문장주어에 대한 주격의 제한은 통사적인 차원에서 도입된 반면, 동사의 목적에 대한 4격 제한은 어휘적인 차원에서 이루어졌다. 이러한 결정은 문법개발자에 의해 이루어진 것이며 이 예는 QPATR가 이론적으로 어떠한 제한도 받지 않으며, 따라서 이론적으로 가능한 문법의 모델화에 매우 다양한 가능성을 제

시하고 있다고 볼 수 있다.

언어학적으로 동기화된 제한 설정(Constraints Bilding)의 또다른 가능성으로 우리는 주어와 동사간의 일치현상을 생각해 볼 수 있다. 만일 앞의 확장된 구구조규칙을 근거로 비문법적인 *"john like apples"*라는 문장을 구문분석하여 보면 어떻게 될까? 위에서 정의한 우리의 구구조문법은 *"john like apples"*라는 문장을 문법적인 문장으로 인식한다. 비록 *"john"*과 *"like"*의 어휘항목에 수의 일치자질을 위한 속성(*head:agr:num*)이 존재하고, 또한 이에 대해 각각 다른 속성값을 갖고 있으므로 비문법적인 문장으로 인식해야 하는데, 그렇지 못하다. 그 이유는 이 두 개의 자질에 대한 제한(두개의 자질이 통합가능해야 한다는 제한)이 그 어느 곳에도 명시화되어 있지 않기 때문이다. 이를 보완하기 위해 우리는 주어-동사간의 수의 일치에 대한 제한을 규칙 1에 다음과 같이 정의함으로써 이러한 문제를 해결할 수 있다.

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

이것은 우리의 문법에 별달리 큰 의미는 없어 보이는 것 같지만, 나중에 사전어휘항목을 보충하였을 경우 'I likes apples' 나 'They likes apples'와 같은 비문의 구조를 배제할 수 있다.

또 하나의 중요한 점은 하위범주화목록의 구성이다. 규칙 1에서 VP의 하위범주화 목록이 첫 번째 요소 뒤에서 바로 닫히었다. 우리는 한 눈에 이것이 단 하나의 주어만 나타나는 자동사가 있는 문장만을 분석할 수 있다는 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?

여기에 대한 해답은 규칙 2에서의 미명세 정보 속에 들어 있다. 규칙 2의 정보를 자세히 보면, V 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 정보만이 통합이 이루어진다. 이러한 기술로 우리는 규칙 1을 임의의 긴 하위범주화목록을 지는 동사가 나오는 문장에 이용할 수 있다.

II.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 있는 가능성을 제시하는 것이다.

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

```
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.
```

```
% Lexicon

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_.

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

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

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

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

sonyeo 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.

senmwulha  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_nonanimate,
        Y/role *= goal,
        F/cs/argst/arg3 *= Z,
        Z/sem *= animate_nonanimate,
        Z/role *= theme,
        F/cs/qualiastr *= give_act,
        F/cs/frame *= 'X_ga_Y_ege_Z_eul_v'.

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

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

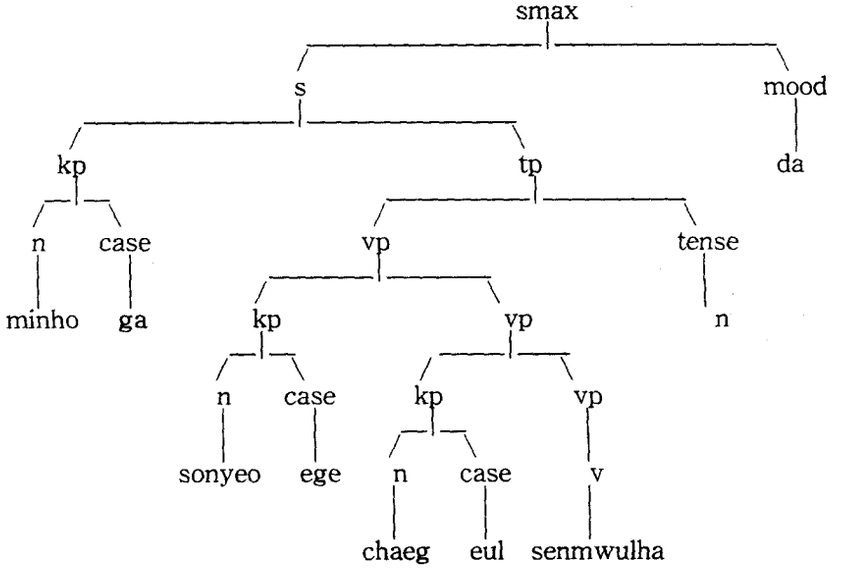


표 3. 컴퓨터를 이용한 분석 결과: 구성성분구조

3) <표 4>의 자질 구조에서 cs 는 'conceptual structure'를 의미하며, event structure, qualia의 세부정보는 여기서는 편의상 간략하게 나타내기로 한다.

■ feature structure:

```
[ cat : smax
  head : [ agr : [ pers : 3
                   num : sg
                   ending : vowel ]
           cas : nom
           tense : pres
           subcat : subj_obj1_obj2
           mood : decl ]
  cs : [ eventstr : process
         argst : [ arg1 : [ sem : animate
                            role : agent ]
                   arg2 : [ sem : animate_nonanimate
                            role : goal ]
                   arg3 : [ sem : animate_nonanimate
                            role : theme ] ]
         qualiastr : transition
         frame : X_ga_Y_ege_Z_eul_v ] ]
```

표 4. 컴퓨터를 이용한 분석 결과: 자질구조

여기에서 본인이 관심이 있는 것은 바로 사전이다. 자연언어처리를 지원해 주기 위한 사전의 많은 정보가 잉여적(redundant)이라는 것을 우리는 알 수 있다. 이러한 잉여적인 정보는 형태론, 통사론, 의미론적인 층위에서 모두 발견할 수 있다. Chomsky의 변형문법이래로 사전의 잉여적인 정보를 최소화하고, 변별적인 정보를 중심으로 어휘정보를 표상하기 위한 연구가 지속적으로 이루어졌으며, 현재에는 계승 메카니즘(inheritance)을 중심으로 계층구조적인 사전(hierarchical lexicon)을 구축하려는 연구가 주를 이루고 있다.

앞에서 하나의 문장에 사용되는 단어의 어휘정보를 표상하는데 있어서 상당히 많은 정보가 중복되어 있음을 알 수 있다. 만일 이러한 어휘정보를 지닌 어휘항목이 늘어나게 되면, 우리는 개별어휘항목에 대해서 잉여적인 어휘정보를 반복하여 정의하는 수고를 해야한다. 그러나 우리가 계승 메카니즘을 도입하여 사전을 구축한다면, 우리는 각 사전의 개별 층위(형태, 통사, 의미적)에 유형 계층(type hierarchy)을 설정하고, 변별적이고 잉여적인 정보를 분류하여 각 유형에 적합하게 어휘정보를 정의할 수 있다.

그렇다면 이러한 생각을 QPATR시스템에서도 구현할 수 있을까? 물론 구현할 수 있다. 단지 QPATR의 계승 메카니즘은 단조적(monotonic)이므로 상위유형과 하위유형간의 모순되는 내용이 있을 경우, 정보의 계승이 이루어질 수 없다. 이를 보완하기 위해서는 비단조적인 해석(non-monotonic interpretation)을 가능하게 하는 계승 메카니즘이 필요하다. 이러한 계승메카니즘을 가능하게 하는 지식표상언어로서 DATR가 있는데, 이는 R. Evans와 G. Gazdar에 의해서 1989년에 개발된 지식표상언어이다. DATR는 인공지능에서 필요로 하는 여러 가지 지식을 표상할 수 있지만, 특히 사전지식을 구축하는데 아주 유용하게 쓰일 수 있다.

이제 앞에서 제시한 '민호가 소녀에게 책을 선물한다.'라는 문장을 분석하기 위하여 우리는 DATR를 이용하여 원하는 어휘항목의 정보를 속성-값을 이용하여 <표 5>와 같이 사전을 구축할 수 있다.

DATR을 이용하여 사전을 구축할 경우, <표 5>에서 보는 바와 같이 개별 어휘항목의 어휘정보에 포함되어 있는 정보가 다분히 잉여적인 것을 알 수 있다. 이러한 잉여 정보는 어휘항목을 지배하고 있는 상위 범주에 정의될 수 있으며, 따라서 개별 어휘항목의 정보는 보

```
MINHO: <> == NOUN
       <pers> == third
       <num> == sg
       <ending> == vowel
       <sem> == human
       <trans> == minho_.

CHAYK: <> == NOUN
       <ending> == cons
       <sem> == artifact
       <trans> == book_.

CA:  <> == INTRANS_VERB
       <type> == verb
       <subcat> == one
       <event> == process
       <arg1> == animate.

GA:    <> == KAS
       <type> == case
       <cas> == nom
       <ending> == vowel.

REUL:  <> == KAS
       <type> == case
       <cas> == acc
       <ending> == vowel.

SONYEO: <> == NOUN
        <pers> == third
        <num> == sg
        <ending> == vowel
        <sem> == human
        <trans> == girl_.

SENMWULHA: <> == DI_TRANS_VERB
           <event> == process
           <arg1 sem> == animate
           <arg1 role> == agent
           <arg2 sem> == animate_nonanimate
           <arg2 role> == goal
           <arg3 sem> == animate_nonanimate
           <arg3 role> == theme
           <qualiastr> == send
           <frame> == 'X_ga_Y_ege_Z_eul_v'
           <comp> == three.

I:     <> == KAS
       <type> == case
       <cas> == nom
       <ending> == cons.

EGE:   <> == KAS
       <type> == case
       <cas> == dat
       <ending> == vowel.
```

표 5. DATR을 이용한 사전 어휘 정보

다 더 단순화되어, 특유의(idiosyncratic) 정보만을 포함하면 된다.

또한 계승메카니즘이 허용되는 경우 어휘장에서 유사한 의미를 지닌 단어들에 대해, 이를 포함하는 상위어에 어휘정보를 정의함으로써 사전을 효율적으로 구성할 수 있다. 즉 우리가 '선물하다'와 유사한 단어로서 '주다'라는 어휘항목의 정보를 정의해야 할 경우 우리는 간단히 "주다:<> == 선물하다"와 같은 형태로 정의를 하여 줌으로써 "주다"라는 동사의 모든 어휘정보는 "선물하다"라는 동사의 어휘정보로부터 물려받게되고, 이 외에 "주다"라는 동사가 지니는 독특한 의미만을 추가로 정의만 하면 된다.

이와 같은 어휘정보는 문법의 개별층위에서 모듈별로 구축될 수 있으며, 이렇게 구축된 DATR 사전은, PATR 구문분석과정에서 컴파일되면서 시스템에 읽어져, 문장을 분석하는데 이용될 수 있다.[4)]

III. 결론

지금까지 우리는 한국어 문장을 분석하기 위해서 필요로 하는 문법을 기술하고 이를 지원해주는 사전을 구축함으로써 자연언어처리가 어떻게 이루어지는지를 개략적으로 살펴보았다. 언어학자의 문법적인 지식이 자연언어의 전산처리에 어떻게 이용될 수 있으며, 생성어휘론에서 추구하는 사전 구축의 방법이 자연언어처리에 쉽게 이용될 수 있음을 보았고, 이를 바탕으로 문장의 언어보편적인 구조를 무리없이 표상할 수 있음도 확인하였다. 본 논문에서 언급되어지지 않은 내용, 즉, 문장입력전의 형태소 해석 작업이나, 문장분석 후의 의미표상 결과는 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

4) DATR사전을 이용하여 앞의 예문을 분석한 결과는 <표 3>, <표 4>의 결과와 동일하다.

가능성을 보여줌으로써 전산언어학에 근접할 수 있는 작은 통로를 보여주는 것이 작은 바램이다. 앞으로 이러한 시스템을 가지고 실제로 문법의 이론적인 부분을 기계적으로 검증하면서 시험해 볼 수 있는 계기가 되어 지금까지 축적된 언어학적 이론이 다양하게 응용이 된다면, 국어 정보처리의 발전에 이바지 할 수 있으리라 확신한다.

참고문헌

양정석. 1995. 국어동사의 의미 분석과 연결이론, 도서출판 박이정.

이민행. 1994. "국어와 독일어의 대조통사론과 기계번역", 독일문학 51집.

이정민 · 남승호 · 강범모. 1998. "한국어 술어의 어휘의미에 대한 생성적 연구방법", 인지과학 제 9 권 3호.

장석진. 1995. 정보기반 한국어 문법, 한신문화사.

조자경. 1995. "의미선택 제약의 기술", 독일문학 57집.

Bouma, G. 1992. "Feature Structures and Nonmonotonicity", *Computational Linguistics* 18, 183-203.

Briscoe, T. *et al.* (eds.) 1993. *Inheritance, Defaults and the Lexicon.* Cambridge: Cambridge Univ. Press.

Daelemans, W. and G. Gazdar. 1990. *Workshop on Inheritance in Natural Language Processing.* Tilburg: ITK (Institue for Language Technology and AI).

Daelemans, W., G. Gazdar, and K. De Smedt. 1992. "Inheritance in Natural Language Processing", in: *Computational Linguistics* 18, 205-217.

Evans, R. and G. Gazdar. 1990. *The DATR Papers.* Cognitive Science Research Paper CSRP 139, University of Sussex, Brighton.

Flickinger, D. 1987. *Lexical Rules in the Hierarchical Lexicon.* Dissertation. Stanford Univ.

Kilbury, J. *et al.* 1992. "New Lexical Entries for Unknown Words", *Theorie des Lexikons: Arbeiten des Sonderforschungsbereichs 282,* no. 29, 1-26.

Koelzer, A. 1992. "Eine Semantische Repräsentation für die SIMLEX-Grammatik", *Theorie des Lexikons: Arbeiten des Sonderforschungsbereichs 282,* no. 29, 38-82.

Koenig, J. 1999. *Lexical Relations.* CSLI: Stanford

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

Pustejovsky, J. 1995. *The Generative Lexicon.* The MIT Press.

Shieber, S.M. 1986. *An Introduction to Unification-Based Apporaches to Grammar,* CSLI Lecture Notes Nr. 4. Stanford.

Touretzky, D. S. 1986. *The Mathematics of Inheritance Systems.* London/Los Altos: Pitman/Morgan Kaufmann.