패턴-액션 규칙을 이용한 한국어 구문 분석†

나 동 열
연세대학교 전산학과

Syntax Analysis of Korean Using Pattern-Action Rules

Dongyul Ra
Yonsei University, Computer Science Department

요 약

전문가 시스템은 인공지능 분야에서 가장 성공적인 사례로 꼽히고 있다. 본 논문에서는 전문가 시스템에서 채용하고 있는 핵심기술인 패턴-액션 규칙이 자연어 구문 분석 작업에도 성공적으로 적용될 수 있음을 보이고자 한다. 자연어의 문법 규칙을 전문가 시스템의 생성 규칙 형태로 표현하고, 전문가 시스템의 추론 엔진에서 이용하는 알고리즘(특히 전향 추론 방식)을 이용하여 구문 분석을 진행하도록 한다. 이 방법이 부분 자유 어순과 중심어 후행과 같은 특징을 가진 한국어의 분석에도 적용될 수 있음을 보였다.

I. 서 론

본 논문에서는 한국어를 분석하는 기술을 개발하는 데 있어서 언어학 이론에만 전적으로 의존하지 않고 인공지능분야에서 가장 성공한 분야인 전문가 시스템(expert system)의 패턴-액션 규칙을 이용하고 자 한다. 이러한 규칙을 이용하여 자연어를 파싱하고자 하는 방법이 Marcus에 의하여 제안되었으나[4,5,6] 한국어의 분석에 적용된 예는 아직 없었다.

원래의 Marcus 파싱은 자연어를 결정론적으로(deterministic) 분석할 있다는 전제에 기반을 두었으나 우리는 이를 받아 들이지 않았다. 따라서 Marcus 파싱에서 제안된 자료 구조와 제어 구조를 수정하여야 했다. 첫째로 스택과 버퍼를 합쳐서 하나의 스택에 의하여 파싱 과정을 표현하며, 둘째로 파싱 상태(state)을 표시하는데 이용되는 패킷을 갖지 않도록 한다. 이에 따라서 알고리즘도 수정되었다.

이 방법에 따라 기본적인 한국어 문장 구조를 분석할 수 있는 규칙들을 소개하며 이를 이용하여 한국어 문장을 분석하는 과정을 예를 들어 설명하였다.

† 본 연구는 한국통신 장기기초 연구비의 지원에 의한 것임.

Ⅱ. 한국어 문장의 기본 구조

지면 관계상 본 논문에서는 (그림 1) 의 문법에 해당하는 정도의 한국어를 분석할 수 있는 파싱 규칙들만을 소개한다. 파싱되는 한국어의 범위를 넓히기 위해서는 단지 파싱 규칙들만을 추가하면 되므로 용이하게 파서를 확장시킬 수 있다. 한국어는 용언이 맨 뒤에 나오며 그 앞의 체언구와 부사구들의 순서는 자유로운 언어 즉 부분 자유어순이며 중심어 후행인 언어임을 (그림 1)은 가정하고 있다[1,2,3].

```
(g1)  S   ->  [ NPP | ADV ]*   VP
(g2)  NPP ->  NP  P
(g3)  NP  ->  [ NPC P[coord] ]*  NPC
(g4)  NPC ->  NA*  N*  N
(g5)  NA  ->  ADJ | NPP[mod] | S[mod]
(g6)  VP  ->  VA[m] | VA[m] VA[aux]
(g7)  VA  ->  V  E
```

(* 는 0 번 이상의 반복, [ ] 는 grouping, | 는 여럿 중 하나를 택하는 것을 나타내는 메타 기호이다)

(그림 1) 한국어 기본 구조를 나타내는 문법

위 문법에서 쓰인 각 nonterminal 이 지니는 의미는 다음과 같다:

```
S   : 문장이나 절을 이끄는 nonterminal
NPP : 체언구(조사 포함)
P   : 조사
NP  : 맨 끝의 조사를 제외한 체언구
NPC : ~과, ~와 등으로 접속 만들어진 체언구(NP)를 구성하는 성분
NA  : 체언구내의 수식 부분
ADJ : 관형사 또는 수사
ADV : 부사
VP  : 용언구
VA  : 용언( feature [m]은 본용언, [aux] 는 보조용언을 나타냄)
V   : 용언의 어간
E   : 용언의 어미
```

위 그림에서 Nonterminal 밑의 대괄호 내에 표시된 것은 그 nonterminal 이 가져야 하는 feature 를 나타낸다. [coord] 는 '와', '과' 등과 같은 등위 접속 조사임을 나타내는 문법 feature이며, [mod] 는 NPP 의 경우 관형격 조사로 끝나는 체언구를 나타내며 S 의 경우 관형형 어미로 끝나는 절을 의미한다.

위 그림의 문법 규칙들은 한국어 문장의 기본적인 구조를 나타내기 위한 목적으로도 이용된다.

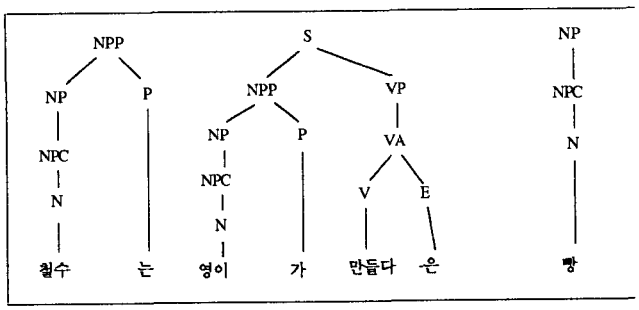
Ⅲ. 자료 구조

본 파서에서 이용하는 주요 자료 구조는 스택으로서 파싱 과정에서 문장의 각 구(phrase)에 해당하는 트리들이 생기면 이를 스택에 넣는다(push). 스택의 top은 가장

최근에 생성된 트리를 가지고 있으며 항상 다음 프로세싱의 촛점이 된다. 파서는 스택의 top 부분에 있는 트리들의 종류 및 구조 그리고 트리 내부의 노드가 가진 feature를 검사하여 다음 동작을 결정한다. 스택은 한 분석 과정을 나타내며 하나의 가능한 파싱 경로(또는 파싱 상태)를 나타낸다고 볼 수 있다[7].

문장 (1) 을 파싱하는 과정에서 형태소 '빵' 이 입력되어 처리되는 과정에서 생기는 자료 구조의 한 장면은 (그림 2) 과 같다.

(1) 철수는 영이가 만든 빵을 길수에게 주었다.



(그림 2) 스택과 그 내용

Ⅳ. 규칙 구조

규칙의 구조는 (그림 3) 에 표시된 것처럼 패턴 부분과 액션 부분으로 구성되어 있다. 패턴 부분은 그 규칙이 선택되어 수행되기 위해서 만족되어야 하는 조건들을 표시한다. 액션 부분은 그 규칙이 선택되었을 때 수행하여야 할 작업들을 표시한다.

[priority]

[Pattern Element 3] [Pattern Element 2] [Pattern Element 1] ==> [action1] ... [action-n]

(그림 3) 규칙의 구조

우선순위(priority)는 두개 이상의 규칙이 동시에 매치될 때 그 중 우선순위가 가장 높은 규칙을 선택하는 데 이용된다. 다른 규칙과의 우선 순위가 필요하지 않을 경우에는 표시하지 않아도 된다.

Pattern Element(PE) 들은 스택의 top 부분의 트리를 조사하는데 이용된다. PE 는 PE1 에서 부터 출발하여 최대 3개까지(즉 PE3 까지) 둘 수 있다. PE1 은 스택 top의 트리에 대해서, PE2 는 그 아래의 트리, PE3 는 그 아래의 트리에 대한 조사를 담당한다. 각 PE 에서 조사되는 사항은 트리의 한 부분(보통 root 또는 오른편 위 부분)의 노드 종류, 노드가 가지고 있는 feature 등에 관한 것이다.

액션 부에서는 스택의 맨 위 두개의 트리를 결합하여 하나의 트리를 만들거나, 새로운 트리를 넣거나(push), 트리의 노드에 새로운 feature를 넣는 등의 작업을 수행한다. 기타 자세한 설명은 아래의 한국어 파싱을 위한 규칙을 소개하는 부분에서 계

속하겠다.

V. 파싱 알고리즘

본 논문에서 사용하는 규칙 기반 파싱의 개략적인 알고리즘은 (그림 4) 와 같다.

```
(1)   find all rules whose pattern part is satisfied.
      if no rules are found,
         then if no more words in the input,
(2a)                then finish parsing
                    else
(2b)                     { input next word to stack; go to 1; }
      else
(3a)       {  find one rule with the next highest priority;
(3b)          if no rule left, read next word and go to 1;
(3c)          execute all actions in the action part;
(3d)          if the execution of actions is fine semantically
(3e)            then { update the stack according to this execution;
(3f)                   goto 1; }
(3g)            else  goto 3a to try a rule with next priority.
           }
```

(그림 4) 파싱 알고리즘

파싱을 진행하기 위하여 다음 수행할 규칙을 선택하려면, 스택에 대하여 패턴부가 매치되는 규칙들을 찾아야 한다(위 그림의 1).

만약 스택의 상태에 매치되는 규칙이 없으면 이는 현재의 스택의 상태에서는 가능한 모든 처리가 완료되어 더 이상 수행할 작업이 없음을 의미한다. 따라서 스택에 다음 형태소를 입력하여 분석을 계속 할 수 있도록 한다(2b). 만약 다음 형태소를 입력하고자 할 때 입력할 형태소가 하나도 남아 있지 않으면 파싱을 끝내야 한다(2a).

매치된 규칙이 있는 경우에는(3), 이 중 한개의 규칙을 선택하여야 한다. 이 선택은 우선 순위를 비교하여 결정한다(3a). 이렇게 하여 선택된 규칙의 액션부를 일단 수행하여 본다. 이 결과가 의미적으로 문제되지 않으면, 이 결과에 따라 스택의 내용을 변경한다(3e,f). 그러나 의미적으로 좋지 않으면 매치된 규칙중에서 다음으로 우선순위가 높은 규칙을 가지고 재 시도한다(3g). 만약 매치된 모든 규칙이 실패하면 결국 한개의 규칙도 매치되지 못한 것과 같으며 따라서 다음 형태소를 입력하여야 한다(3b).

파싱은 이와 같이 패턴 매치, 수행, 단어 입력 의 사이클을 반복함으로써 진행된다.

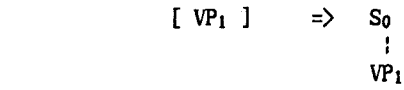
VI. 한국어 파싱을 위한 규칙

1. 절의 분석

한국어에서는 용언이 나타난 후에야 앞의 체언구 및 부사구의 역할을 결정할 수

있다. 용언구에 대한 인식은 다음 규칙에 의한다(용언구 VP의 생성은 뒤에 설명됨).

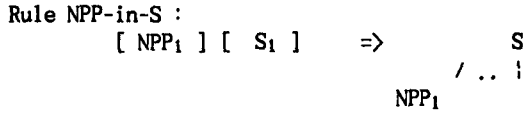
Rule Make-S :



규칙 Make-S 는 용언구가 스택의 top에 나타나면(PE1 이 VP 임에 의하여 표시됨), 이 VP 를 새로운 S 노드에 붙인 후 이 새로운 트리로 하여금 스택 top 의 VP 트리를 대체하도록 한다 (subscript 가 0 이면 새로이 생성되는 노드임을 나타내며, 0 보다 크면 이미 존재하는 노드로서 패턴부와 액션부에서 같은 타입과 같은 subscript 를 가지면 동일한 노드임을 말한다).

규칙 NPP-in-S은 절(clause)에서 용언앞에 나오는 체언구를 분석하는 규칙이다. 이 체언구(NPP)는 스택에서 절(즉 S 트리) 바로 밑에 있는 트리이다. 이 NPP 를 S 노드의 맨 왼쪽에 붙인 후 이 새로운 S 트리가 스택 top 의 두 트리를 대체하도록 한다 (이는 NPP1 트리를 꺼내서 S1 에 붙이는 것과 같은 효과를 가진다).

Rule NPP-in-S :

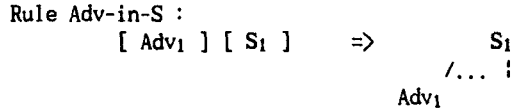


< 추가조건: NPP1 의 조사가 S1 에 대한 격조사로서 적합할 것>

위 규칙의 추가 조건에서는 조사가 암시하는 격이 NPP1 내의 체언이 S1 내의 용언과 관계될 수 있는 격중의 하나인 경우에만 위 규칙의 패턴 매칭이 성공된 것으로 볼 수 있음을 나타내고 있다.

규칙 Adv-in-S 는 절에 나오는 부사가 용언을 수식하도록 한다.

Rule Adv-in-S :



NPP-in-S 와 Adv-in-S 를 반복하여 적용하여 절의 용언구 앞의 모든 체언구와 부사를 소모시키면 결국 스택에는 S 트리 하나만 남게 되며 이 절이 종결형어미인 경우 문장의 끝이 되므로 그 문장의 파싱이 완료된다.

관형형 어미를 가진 절 즉 관형절의 경우 종결형 어미를 가진 절의 경우보다 까다롭다. 그 이유는 이 절이 수식하는 체언도 이 절의 용언의 한 격(case)를 채워야 하는 경우가 많기 때문이다. 예로서 다음 문장들을 보자.

(2) 창호가 영희를 사랑하는 철수를 때렸다.
(3) 창호가 영희가 사랑하는 철수를 때렸다.
(4) 창호를 영희를 사랑하는 철수가 때렸다.
(5) 창호가 영희를 사랑하는 사실을 철수가 안다.

(2)와 (4)에서 '철수'는 '사랑하는'의 주격을 채우며, (3)에서는 '사랑하는'의 목적격

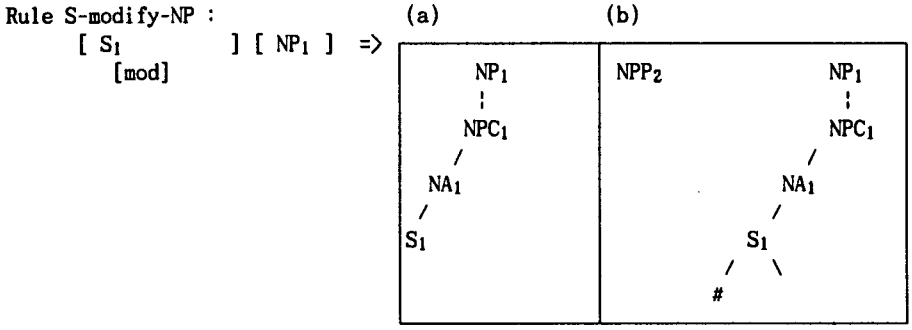
을 채운다. 반면에 (5)에서 '사실'은 '사랑하는'의 아무 격도 채우지 않는다. 위의 사실에 입각하여 다음과 같이 관형절을 분석한다. 먼저 위의 NPP-in-S 와 Adv-in-S 규칙에 의하여 용언구 앞의 체언구를 가능한 한 모두 S 에 붙인다. 더 이상 위 규칙들이 적용될 수 없으면 다음에 따라 나오는 체언구가 분석되어 스택의 top에 마련된다(체언구의 분석은 다음 절에 있음).

조사를 제외한 체언구가 완성되면 바로 앞의 관형절과 이 체언구의 결합을 모색하는데 다음 두가지 경우가 가능하다.

(a): 이 체언구가 위 (5)와 같이 관형절의 용언의 격을 채우는 것이 아니면 바로 관형절을 체언구에 붙이면 된다.

(b): 그러나 위 (2-4)에서 처럼 격을 채울 수 있는 종류의 체언이면 관형절의 맨 왼쪽에 달려 있는 체언구부터 차례로 이 수식되는 체언구가 채우는 격을 이미 채운 것이 있는 지 검사 한다. 이때 있다면 이 체언구는 틀리게 붙여진 경우이므로 떼어 내고 이것 대신에 수식되는 체언구를 붙인다. 없다면 단지 관형절이 수식되는 체언구를 수식하도록 붙인다(attach).

규칙 S-modify-NP 는 위의 (a), (b) 두 경우에 해당되는 작업을 하는 규칙이다. 이 규칙의 액션은 위 (a) 경우에는 action (a) 를, (b) 의 경우에는 action (b) 를 수행한다. PE2 의 [mod] 는 이 절의 용언의 어미가 관형형인지를 조사한다.



위 규칙의 액션 부에서 S_1 내의 # 위치에 원래 NPP_2 가 붙어 있었으나 이것이 S_1 내의 용언에 대하여 NP_1 이 채울 수 있는 격을 채웠으므로 NPP_2 는 절단되어 스택의 한 트리로 격상되며 # 자리에 NP_1 을 가리키는 포인터를 매달아 놓는다.

2. 체언구의 분석

체언구의 중심은 조사 바로 앞에 나오는 체언이다. Head-noun 규칙은 이 점을 이용하여 체언구(NP)를 처음으로 생성한다.

```
Rule Head-noun :
        [ N1 ]  [ P1 ]     =>   pop out P1 :      NP0
                                                   ¦
                                                  NPC0
                                                    \
                                                     N1
```

"pop out P" 는 P 를 스택에서 꺼내어 다시 입력 string으로 되돌려 보낸다. 그 이유는 이 규칙에서는 조사가 체언구의 끝을 알아내는 목적으로만 이용되기 때문이다. 이 조사는 다음 번 입력시 다시 스택안으로 들어와 NPP 를 만드는 데 이용된다.

다음 규칙은 명사가 여러개 연속적으로 나오는 것을 분석하기 위한 것이다(예: '책상 다리').

```
Rule Aux-noun :
   [ N1] [   NP1  ]      =>          NP1
                ¦                    ¦
              NPC1                  NPC1
             / ¦ \                 /  / ¦ \
           N2 .. Nk             N1  N2 .. Nk
```

다음 규칙은 체언에 관형격 조사가 붙어서 다음 체언을 수식하는 경우에 대한 것이다(예: '나무의 줄기'). PE2 의 [mod] 는 NPP_1 이 관형격 조사가 있는지 조사한다.

```
Rule Modifying-NP :
       [ NPP1   ] [   NP2    ] =>            NP2
          [mod]       ¦                      ¦
                     NPC2                   NPC2
                                          /   ¦ \
                                        NA2
                                       /   \
                                    NPP1
```

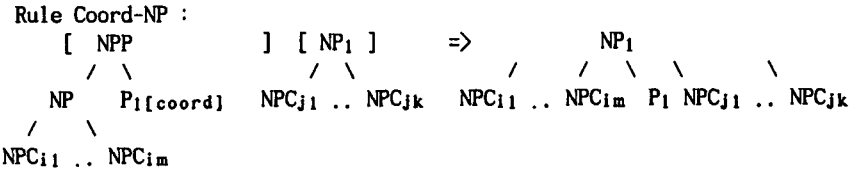
체언구를 수식하는 관형사는(수사, 지시대명사등 포함) 다음 규칙에 의하여 처리된다(예: '새 집').

```
Rule ADJ-NP :
   [ ADJ1 ] [   NP1  ]          =>        NP1
                                           ¦
                                          NPC1
                                         /
                                       NA1
                                      /
                                    ADJ1
```

다음 규칙은 조사를 체언구(NP)와 결합하여 NPP 를 만든다.

```
Rule Make-NPP :
    [  NP1] [ P1]   =>       NPP0
                            /     \
                          NP1      P1
```

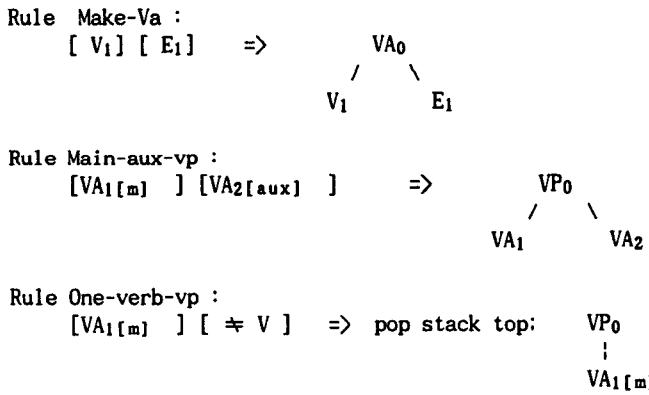
접속조사에 의하여 여러개의 체언구가 모여서 하나의 체언구를 이룰 수 있는 데 이의 처리를 위하여 다음 규칙을 이용한다. 여기에서 $P_{[coord]}$ 는 접속 조사인 '-와',



'-이와', '-과'등을 나타낸다(예: '밤과 대추와 배').

3. 용언구의 분석

용언 구를 분석하는 규칙들을 소개하면 다음과 같다. 규칙 Make-Va 는 어간(V) 과 어미(E) 를 합쳐서 용언(VA) 를 생성한다. 규칙 Main-aux-vp 는 본 용언과 보조 용언을 합쳐서 용언구(VP) 를 생성한다. 본 용언 한개 만으로 구성된 용언구는 규칙 One-verb-vp 에 의해서 생성되는데 이 규칙에서 PE1 는 본 용언 다음에 용언의 어간이 아닌 것(≒ V) 이 오는 것을 보고 용언구를 생성한다. 이 때 스택의 top에 있는 것은 단지 어간이 아닌 것이 오는 것을 확인하는 데만 쓰였으므로 다시 입력 string 으로 되돌려진다.

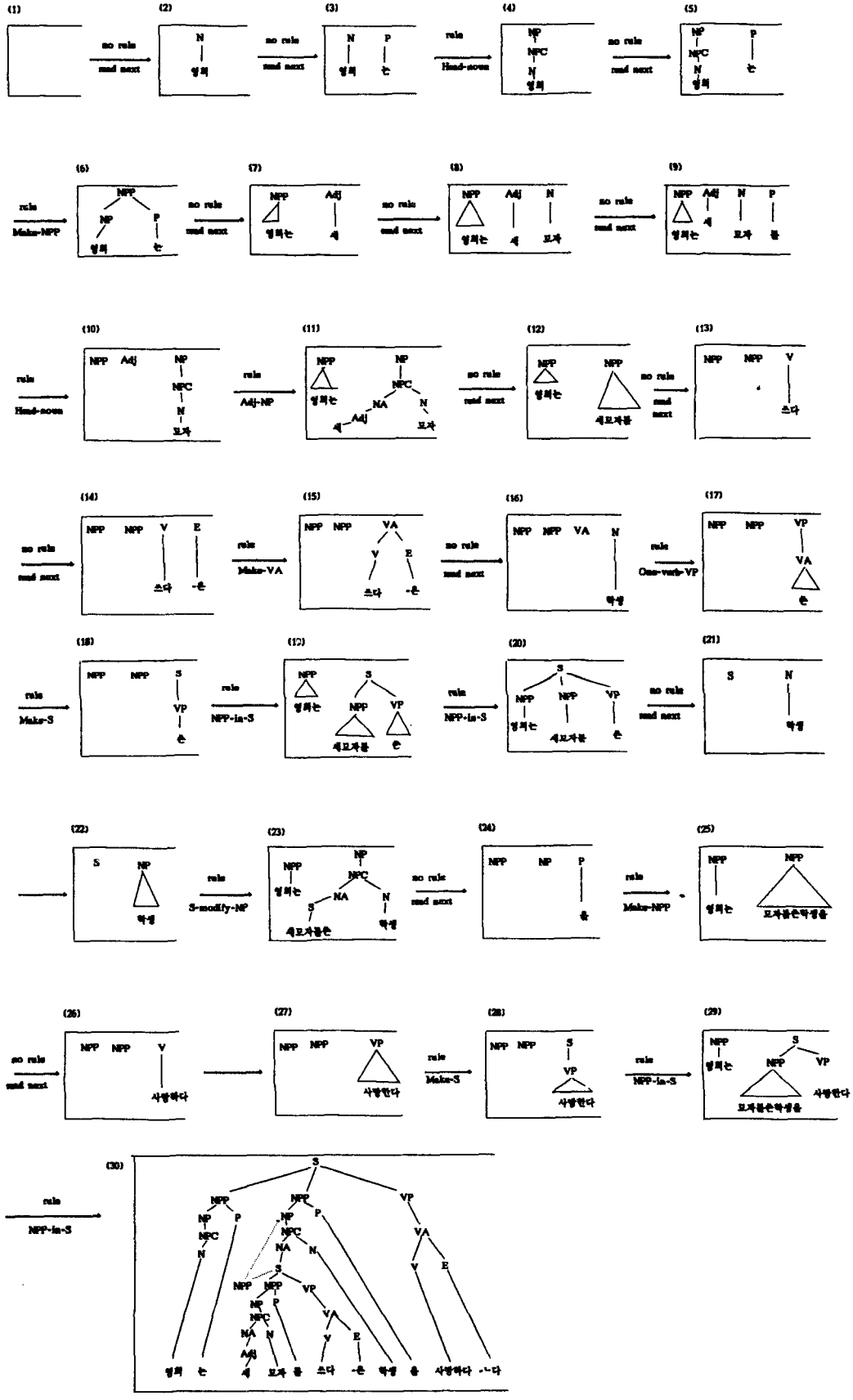


Ⅶ. 파싱 진행 과정의 예

본 장에서는 위에서 지금까지 소개한 규칙 기반 파싱 시스템을 이용하여 한국어 문장을 파싱하는 과정을 (6)의 파싱을 예로 들어 설명하고자 한다. 다음 (그림 5)는 파싱의 과정을 나타낸 것이다.

(6) 영희 는 새 모자 를 쓰다 -은 학생 을 사랑하다 -ㄴ다.
(영희는 새 모자를 쓴 학생을 사랑한다.)

파싱은 맨 처음 빈 스택의 상태에서 출발한다(스텝 1). 이에 대하여 어느 규칙도 매치하지 못하므로 다음 형태소를 스택안으로 입력한다(스텝 2). 스텝 3에서 명사뒤에 조


(1)
(2)
(3)
(4)
(5)
no rule
read next
rule
Head-noun
(6)
(7)
(8)
(9)
Make-NPP
(10)
(11)
(12)
(13)
Adj-NP
(14)
(15)
(16)
(17)
Make-VA
One-verb-VP
(18)
(19)
(20)
(21)
Make-S
NPP-in-S
(22)
(23)
(24)
(25)
S-modify-NP
(26)
(27)
(28)
(29)
(30)
영희는
세
모자를
쓰다
-은
학생
을
사랑하다
-ㄴ다

(그림 5) 파싱 진행 과정

사가 나타나므로 체언구를 만들고 조사는 스택 밖으로 밀어 내어 아직 입력되지 않은 형태소로 된다. 따라서 다음 번 입력에서는 이 조사가 다시 입력된다. 매 스텝마다 그 때의 스택에 매치되는 규칙을 찾고 그에 따른 액션을 수행하여 다음 스텝의 스택으로 변경되는(update) 방식으로 아래 그림을 따라 나가면 파싱 과정을 이해하는 데 큰 어려움이 없으며 따라서 각 스텝에 대한 자세한 설명은 생략한다.

각 스텝 사이에는 적용된 규칙의 이름이 표시되어 있다. 그림을 보기 쉽게 하기 위해서 구축되는 과정이 앞에 나온 트리들은 그 하부 구조를 자세하게 표시하지 않았다. 그림에서 점선으로 된 화살표는 나타나야 할 몇 단계의 스텝을 생략하였음을 나타낸다. 스텝 (30)에서 모든 입력을 소모되었고 스택에는 S 가 root 인 하나의 트리만이 존재하므로 파싱은 성공적으로 끝난다.

구문 트리가 구축될 때마다 의미 분석도 병행하므로 파싱이 끝나면 문장 전체를 나타내는 구문 트리이외에 의미를 나타낼 수 있는 중간 표현도 생성되도록 한다.

Ⅶ. 결 론

본 논문에서는 인공지능 분야에서 가장 성공적인 결실을 거둔 전문가 시스템에서 사용되는 정보 처리 방법인 생성 규칙을 사용하여 한국어를 파싱하는 방법을 연구하였다. 생성 규칙을 이용한 파싱 방법인 Marcus 파싱으로 분류될 수 있는 이 파싱 방법에 의하여 적지 않은 양의 한국어 문장들을 분석할 수 있음이 입증되었다. 이 방법에 의하면 한국어의 특징인 어순의 부분 자유성의 문제도 쉽게 해결된다.

본 논문에서 소개한 파싱 방법은 이해하기 쉽고, 구현하기 쉬우며, 수정 및 확장이 용이함과 같은 장점을 갖추고 있다.

참 고 문 헌

[1]서영훈, 김영택, "활성 차트를 이용한 중심어 후행언어의 파싱", 한국정보과학회 논문지, Vol. 17. No. 1., Jan. 1990.

[2]윤덕호, 김영택, "미지문법관계 속성을 이용한 LFG에서의 한국어 문장분석 연구", 한국정보과학회 논문지, Vol. 16, No. 5, Sep. 1989.

[3]조혁규, 권혁철, "단일화와 차트를 이용한 한국어 구문분석시스템의 구현", 한국정보과학회 논문지, Vol.17, No.4, July, 1990.

[4] Charniak, E., "A Parser with Something for Everyone", in Parsing Natural Language, M. King(editor), Academic Press, London, 1983.

[5] Hirst, G., "Semantic Interpretation against Ambiguity", Ph. D. Dissertation, Department of Computer Science, Brown University, 1984.

[6] Marcus, M., A Theory of Syntactic Recognition for Natural Language, Cambridge, Massachusetts, The MIT Press, 1980.

[7] Tomita, M., "An Efficient Augmented-context-free Parsing Algorithm", Computational Linguistics, 13(1-2), 1987, 31-46.