트리 합성 문법을 이용한 한국어 파싱

양성일, 나동렬
연세대 전산학과

Parsing Korean Using Tree Combining Grammar

Seong-il Yang, DongYul Ra
Yonsei University
Dept. of Computer Science

본 논문에서는 트리의 점증적인 합성에 의하여 파싱을 진행시켜 나가는 파싱 방법을 소개하며 이의 한국어 파싱에의 응용을 살펴 본다. 이와 같은 트리 합성 파싱(tree combining parsing)을 지원하기 위한 문법인 트리 합성 문법(Tree Combining Grammar)을 소개한다. 우리는 먼저 문맥 자유 문법을 작성한 후 이로 부터 자동적인 변환에 의하여 트리 합성 문법을 얻는 과정을 취한다. 따라서 트리 합성 파싱은 일종의 문맥 자유 파싱(context-free parsing)으로 볼 수 있으나 점증성(incremental), 하향성(top-down), 상향성(bottom-up), 유연성(flexible) 등과 같은 장점을 갖고 있다. 트리 합성 파싱의 유연성을 기반으로 하여 부분 자유 어순, 중심어 후행성과 같은 특성을 가진 한국어를 효과적으로 파싱할 수 있도록 트리 합성 파싱을 확장하는 방법을 살펴 본다.

1. 서 론

자연어를 파싱하는데 있어서 구 구조 문법(Phrase Structured Grammar; PSG), 특히 문맥 자유 문법(Context Free Grammar; CFG)도 이용될수 있다[3, 5]. 예를 들면, Tomita는 CFG에 대한 파싱 방법인 LR 파싱을 확장하여 자연어 파싱에 이용하고 있다[8]. 우리는 CFG에 대한 기존의 파싱 방법과 다른 또 하나의 효율적인 파싱 방법을 소개 한다. 이 방법은 트리의 점증적인 합성에 의하여 파싱을 진행 시켜 나가는 방법으로서 트리 합성 문법(Tree Combining Grammar; TCG)를 이용한다.

트리 합성 문법의 각 규칙(rule)은 전문가 시스템의 생성 규칙(production rule)과 유사한 성격을 가지고 있으며 따라서 패턴부와 액션부로 구성되어 있다[1, 2, 6, 7]. 파싱의 상태는 LR 파싱과 마찬가지로 스택과 그 안의 트리들에 의해 나타낸다. 모호성에 의하여 여러개의 파스가 존재할때 각 파스는 별개의 스택에 의해 표현된다. TCG의 각 규칙의 패턴부는 하나 또는 두개의 PE(Pattern Element)로 구성되며 각 PE는 스택내의 관련되는 트리의 상태를 조사한다. 모든 PE의 매칭이 만족되면 액션부를 수행하는데, 액션부는 매칭된 트리를 확장 또는 변경한다. 트리 합성 문법을 사용한 파싱, 즉 TCG 파싱은 전문가 시스템의 인터프리터와 마찬가지로 패턴 매칭/액션 수행의 사이클을 반복하므로서 이루어진다.

CFG의 각 규칙은 몇개의 TCG 규칙으로 변환된다. 따라서 주어진 CFG에 대해서 TCG를 구할수 있으며, TCG는 CFG와 같은 언어 기술 능력을 갖는다. 하나의 파스는 하나의 스택에 의해 표현되는데, TCG 규칙의 PE1은 스택 정상(top)의 트리를 조사하고, PE2는 바로 그 아래의 트리를 조사한다. TCG 규칙 중에는 입력 스트링에서 다음 단어의 검사, 즉 lookahead 테스트를 하는 규칙도 있다.

TCG 파싱에서는 한 스택에 대해서 매칭되는 규칙이 두개 이상인 경우에는 각 규칙마다 수행 결과를 반영하는 스택이 별도로 생기게 된다. 반대로 매칭되는 규칙이 하나도 없는 경우에는 수행할 작업이 더 이상 없는 경우가 되며, 다음 단어가 입력되어야 한다. 그러나 이때 하향식(Top-down) 정보를 사용하여 다음 단어가 전체적인 파싱의 관점에서 볼때 예측되는(expected) 단어인 경우만 입력된다. 그렇지 않은 경우에는 파싱이 잘못된 것이므로 스택을 제거한다.

TCG의 기본적인 동작 방식은 상향식(Bottom-up) 이다. 그러나 순수한 상향식(Bottom-up) 방식은 비효율적이다. 따라서 TCG 파싱에서는 하향식(Top-down) 정보도 이용하도록 한다. 이는 스택의 한 트리는 바로 다음 트리의 루트 타입에 대한 기대치를 갖도록 하므로써 구현한다. 따라서 트리의 새로운 루트가 만들어 지거나 새로운 단어가 읽혀 들여지기 전에는 항상 이 기대치에 만족되는지를 검사한다. 여기에서의 기대치는 하향식(Top-down) 정보에 의해 구해진다. 따라서 TCG 파싱은 상향식(Bottom-up) 및 하향식(Top-down)을 모두 사용하여 효율성을 얻는다.

TCG 파싱의 또다른 특성은 점증성(incremental)이다. 트리를 생성할 때 LR 파싱에서처럼 모든 자노드(child node)를 만든 후에 부모노드(parent node)를 만드는[4] 대신 맨 왼쪽 자노드가 생기면 즉시 부모노드를 만든다. 다음 자노드가 만들어질 때마다 이를 부모노드에 붙인다. 결과적으로 트리는 점

증적으로 확장된다. 이는 의미 분석과 관련되어 유용한 특성이다.

TCG 파싱 방법은 매우 유연한 파싱 방법이다. 따라서 ECFG(Extended Context Free Grammar)의 파싱을 위해서도 쉽게 TCG 파싱을 확장할수 있으며 이를 기반으로 부분 자유 어순, 중심어 후행과 같은 특성을 가진 한국어를 효과적으로 파싱할수 있도록 기본 TCG 파싱을 확장할수 있다.

2. TCG 문법

2.1 TCG 규칙의 종류

가능한 TCG 규칙의 종류에는 3가지가 있다. 이들 각각에 대하여 검토해 보면 다음과 같다.

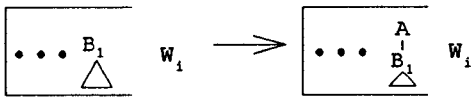
2.1.1 루트 생성(root-creating) 규칙

루트 생성 규칙은 현재 스택 정상에 위치한 파스 트리에 대해 새로운 루트 노드를 만드는 TCG 규칙이다. 이러한 규칙의 일반적인 모습은 다음과 같다.

```
                                  A
T1:  [ B1 ] | first(B2) =>        |
                                  B1
```

'=>'의 왼쪽은 패턴부이고 오른쪽은 액션부이다. PE1인 [B_1]은 스택 정상에 있는 트리의 루트가 B_1이어야 함을 나타낸다. '|first(B_2)'는 다음에 읽을 단어가 first(B_2)에 속하는지를 검사하는 것이다[1]. 이 조건은 B_2가 만들어질 가능성이 있는지를 미리 검사하는 것으로서 lookahead 테스트이다.

루트 생성 규칙의 액션부는 패턴부에서 확인된 B_1 노드 위에 A 노드를 만들어 새로운 루트로 붙이라는 것을 의미한다. 이 규칙의 처리 내용은 다음의 <그림1>과 같다.



<그림 1> 루트 생성 규칙의 동작

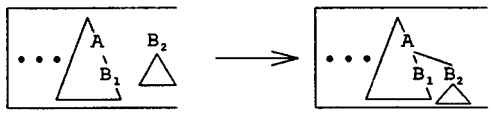
2.1.2 트리 합성(tree-combining) 규칙

트리 합성 규칙은 스택 정상에 있는 두개의 파스 트리를 하나의 트리로 합성하는 규칙이다. 이러한 규칙의 일반적인 모습은 다음과 같다.

```
T2:  [ A ] [ B2 ] =>   A
      |               / \
      B1             B1  B2
```

이 규칙은 패턴부가 PE1, PE2의 두개로 이루어져 스택의 꼭대기에 있는 두개의 트리를 검사한다. PE1인 [B_2]는 스택 정상에 있는 트리의 루트가 B_2 이어야 함을 나타낸다. 단, 이때 B_2 노드는 완성(complete) 상태이어야 한다[2]. TCG 규칙에서 PE1은 항상 완성(complete) 상태이어야 매칭이 된다. PE2는 스택의 정상에서 2번째 트리의 루트가 A이며 그 자노드 중 가장 오른편에 달려 있는 자노드가 B_1이어야 매칭이 됨을 지정하는 것이다.

액션에서는 B_2 트리를 A 노드와 자식으로 B_1의 오른쪽에 붙이게 된다. 이 규칙을 적용할때는 먼저 스택의 정상에 B_2 트리가 완성되어 있는지를 확인한다. B_2 트리가 완성되어 있는 것이 확인되면 스택에서 B_2 트리 아래에 위치하는 트리의 오른쪽 변을 검사하여 맨 오른쪽 자노드로 B_1 노드를 갖는 A 노드가 있는지를 알아보고, 발견되면 여기에 B_2 트리를 오른편에 추가하여 붙인다. 처리 내용은 다음의 <그림2>와 같다.



<그림 2> 트리 합성 규칙의 동작

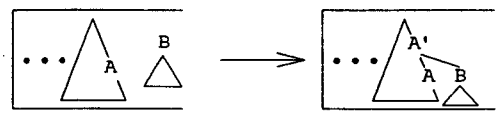
2.1.3 재귀 처리(recursion-handling) 규칙

재귀 처리 규칙은 기존에 작성되어 있는 파스 트리에 노드를 끼워 넣는 규칙이다. 이것은 CFG에서 좌순환(left recursion) 이 있는 규칙을 처리하기 위해 마련된다. 재귀 처리 규칙의 일반적인 모습은 다음과 같다.

```
                                    |
                                    A'
T3:  [ A<CP> ] [ B ] =>            / \
      |                           A   B
      |                           |
```

PE1인 [B]는 앞서 설명된 다른 규칙들과 같이 스택 정상의 트리의 루트가 B임을 나타낸다. PE2의 '-A-'은 해당 트리(즉 정상에서 2번째 트리)의 오른쪽 변에 A 노드가 있어야 하며 <cp>는 이 A 노드가 완성(complete) 상태이어야 함을 나타낸다.

이 규칙의 액션부에서는 먼저 새로운 노드인 A'를 만든다('은 단지 새로운 노드임을 나타내기 위한 것임). A' 노드는 A 노드와 같은 타입을 가지는 노드이다. 이 새 노드를 A 노드의 부모 노드에 붙이고, A'의 자노드로 A 노드를 붙인다. 이로서 새로 만들어진 A' 노드는 A와 A의 부모 노드 사이에 끼워 넣어 지게 된다. 다음 A' 노드의 오른쪽 자노드로 B 노드를 붙여 액션을 완성한다. 이 규칙의 실행 모습은 다음 <그림3>으로 잘 알수 있다.



<그림 3> 재귀 처리 규칙의 동작

---

[1] first(B_2)는 B2를 루트로 하는 파스 트리를 구성하는 terminal중 가장 왼편에 나올수 있는 것의 집합으로 first(B_2) = { a | $B_2 \Rightarrow^{*}$ aα, a is a terminal }이다.

[2] B_2가 완성(complete) 상태이려면 B_2에 모든 자노드가 붙어 있어야 한다. 즉 B_2 -> B F G 에서 B_2에 모든 자노드 E, F, G가 붙은 후에 B_2는 완성 상태가 된다. 그 이전의 B_2는 미완성 상태이다.

재귀 처리 규칙은 PE2에 대응이 되는 노드와 같은 노드를 새로 생성하여 트리에 추가하고 PE1의 대응 노드를 붙인다.

2.2 CFG에서 TCG로의 변환

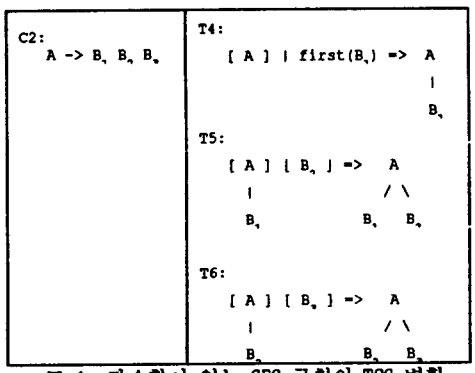
2.2.1 좌순환(left recursion)이 없는 CFG 규칙

GF.는 FC.다 든여하 능력을 갖는다. 모든 CFG에 대하여 대 하몇의 GFC나 칙규은 있다. 하나의 CFG규칙은 몇개의 TCG 은적자 로동도지, 이어루 록정은 자동적으로 이루어지도록 할수 있다. TCG는 Chomsky Normal Form이나 Greibach Normal 홈rm만 으블 수형(normal form)의 , 만으로 붙수 있지만, 매우 다른 형식을 갖는다.

일반적인 CFG 규칙으로 C1과 같은 규칙을 살펴보자.

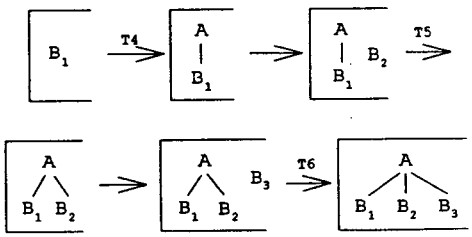
C1: A -> B_1 B_2 ... B_k , (A≠B_1)

C1이 TCG 규칙으로 전환되면 k개의 TCG 규칙이 생성된다. A ≠ B_1은 C1이 직접 좌순환을 갖지 않음을 나타낸다.



<표-1> 좌순환이 없는 CFG 규칙의 TCG 변환

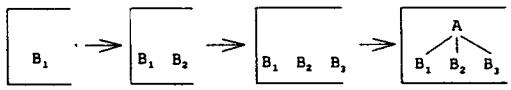
CFG 규칙 C1에서 생성되는 TCG 규칙은 1개의 루트 생성 규칙과 k-1개의 트리 합성 규칙들로 이루어진다. 이렇게 만들어지는 TCG 규칙들은 지정된 순서를 갖는다. 루트 생성 규칙이 먼저 적용되고, 트리 합성 규칙들이 차례대로 순서를 잇는다. 이것은 CFG의 구성을 TCG에서는 트리의 구성에 대해 각 단계별로 상세히 나누어 기술하기 때문이다. <표-1>은 k=3 인 경우의 예를 보인 것이다.



<그림 4> TCG 파싱의 모습

<표-1>에서 T4, T5, T6를 수행하고 나면 A 노드에 B_1과 B2, B3를 붙인 트리를 얻게 되는데 이 과정은 <그림 4>와 같다.

CFG 규칙 C2가 TCG로 변환되어 적용되는 모습은 LR 파싱과 비교될 수 있다. LR 파싱의 경우, B_1, B_2, 마지막으로 B_3가 스택에 나타나고 나면 다음에 'reduce A -> B_1 B_2 B_3' 액션 에 의해 A 트리가 생성되는 반면(그림5), TCG 파싱에서는 그림에서 보는 것과 같이 점증적으로 트리가 구성된다(incremental parsing).



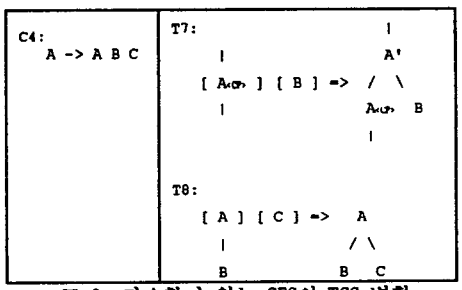
<그림 5> LR 파싱의 모습

2.2.2 좌순환이 있는 CFG에 대한 TCG

좌순환(left recursion)을 갖는 CFG 규칙에 대한 TCG 규칙들은 좌순환이 없는 경우와 다르게 된다. 예를 들어 CFG C3와 같은 규칙을 살펴보자.

C3: A -> A B_1 B_2 ... B_k

여기에서 C3는 좌순환을 갖고 있다. 이때는 1개의 재귀 처리 규칙과 k-1개의 트리 합성 규칙으로 구성된다. <표-2>는 좌순환이 있는 CFG에 대해 구성되는 TCG를 예를 들어 보인 것이다.

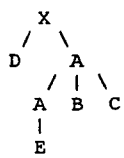


<표-2> 좌순환이 있는 CFG의 TCG 변환

<표-2>에 나와 있는 TCG를 사용하여 다음과 같이 좌순환 규칙이 있는 CFG에 대한 TCG 파싱 과정을 검토하여 보자.

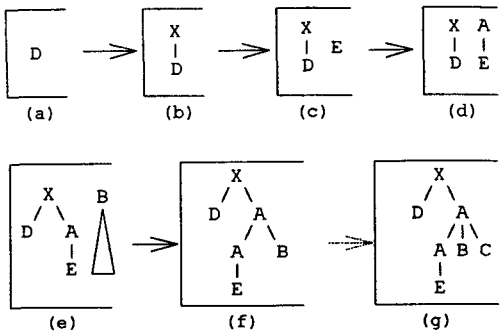
X -> D A
A -> A B C
A -> E

위의 CFG에 대해서는 <그림6>과 같은 파스 트리의 구성이 가능해 진다.



<그림 6> 파스 트리의 예

파스 트리를 TCG 규칙의 적용에 의하여 구축하는 과정을 순서대로 살펴보면 다음과 같다.



<그림 7> 좌순환이 있는 파스 트리의 구성

<그림7-a>에서와 같이 D가 스택에 나타나면 이것은 루트 생성 규칙에 의하여 X가 루트인 트리로 확장된다(그림7-b). 얼마후 B가 스택에 나타나게 되고(그림7-c) 이것은 다시 A가 루트인 트리로 확장된다(그림7-d). X와 A의 두 트리는 트리 합성 규칙에 의해 그림7-e와 같은 X 트리가 되고 얼마후 B 트리가 나타난다(그림7-e). 여기에서 <그림7-f>로 진행하기 위해서는 재귀 처리 규칙이 필요해 진다. 재귀 처리 규칙에 의해 B가 새롭게 만들어진 A 노드의 오른편 밑에 붙여지고, 원래의 A는 새롭게 만들어 지는 A의 왼편 아래에 붙게 된다(그림7-f). 새롭게 만들어진 A는 나중에 나타나게 되는 C를 트리 합성 규칙에 의해 붙임으로서 완성되어, 전체 트리인 X를 얻는다(그림7-g).

3. TCG 파싱

3.1 기대(expectation) 정보의 이용

TCG 파싱의 기본적인 동작 방식은 bottom-up이다. 즉 맨 밑의 단어에서 시작하여 단어를 모아 트리를 만들고 이 트리들을 모아 더 큰 트리를 만드는 방식으로 진행한다. 그러나 이러한 순수 상향식(bottom-up) 방식의 단점은 전체적인 관점에서 볼때 쓸모가 없는 트리를 많이 만들수 있다는 점이다.

이를 교정하기 위해서 TCG 파싱에서 하향식(top-down) 정보를 이용할수 있도록 스택에서 다음으로 나타나는 트리에 대한 예측 기대(expectation)정보를 활용하고자 한다. 즉 스택의 정상에 새로운 트리가 생성 또는 확장될때 지금까지 스택내의 처리 결과에 비추어서 이 트리가 합법적인(legal) 것인지 조사하여 합당한 경우만 파싱이 진행되게 한다.

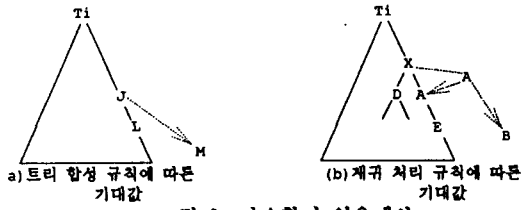
T_1, ..., T_k라는 트리가 스택의 밑에서부터 꼭대기까지 있다고 가정하면, 각각의 트리 T_i는 다음의 트리(즉 T_{i+1})가 무엇이어야 한다는 기대(expectation)를 가진다. T_i에 의해 기대되는 다음 트리의 가능한 루트 타입을 모은 집합을 expt(T_i)라 하고 T_{i+1}의 루트가 expt(T_i) 내에 있을때 T_{i+1}이 합법적인 것으로 간주된다.

T_i의 다음 트리에 대한 기대값이란 트리 T_i에 결합될수 있는 트리의 루트 타입들로서 함수 expt(T_i)의 값이 된다.

3.1.1 expt(T_i)의 계산

J -> L M N 이 있을때, J노드가 T_i의 오른편 가장자리에 있는 노드 중 가장 낮은 곳에 있는 미완성(incomplete) 노드이고 L이 이러한 J에 가장 최근 붙여진 자노드라면, T_i는 다음에 나오는 트리로 M트리(즉 M이 루트 타입인 트리)가 생성되기를 기대하게 된다(그림8-a). 따라서 M은 expt(Ti)의 원소가 되어야 한다(expt(Ti) ∋ M).

여기에서 가장 밑에 있는 미완성(incomplete) 노드를 선택하는 이유는 다른 상위의 노드가 완성되기 전에 먼저 밑의 것이 완성되어야 하기 때문이다.

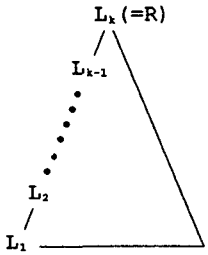


<그림 8> 좌순환이 없을때와 있을때의 기대값

A -> A B C의 좌순환이 있는(left recursion) 규칙에 따르면 TCG 상에서 완성(complete)된 A 노드와 B가 재귀 규칙에 의해 합성되어 상위의 A를 생성할수 있으므로 B가 expt(T_i)의 원소가 될수 있다(expt(T_i) ∋ B). 즉, 좌순환이 있는 규칙 A -> A B C이 있고, A가 트리 T_i의 오른편 가장자리에 있다면, 이 트리의 기대(expectation) 값에는 CFG 규칙의 오른편의 두번째 심볼에 해당되는 B도 포함된다(그림8-b).

3.1.2 기대(expectation) 정보의 이용

R가 비단말 기호(nonterminal)라고 할때, leftedge(R)라는 함수를 정의하자. 이 함수는 루트가 R인 트리의 왼편 가장자리에 달리게 될수 있는 노드들을 알아낸다. (그림10)에서 {L_1, L_2, ..., L_k}는 leftedge(R)의 값이된다.



<그림 9> leftedge(R)에 의해 구해지는 트리의 노드들

leftedge(R)에는 R 자신도 포함된다. 트리 T_i의 기대값에 의거하여 다음 트리인 T_{i+1}가 합법한 가의 여부는 expt 함수와 leftedge 함수를 이용하여 구현할수 있다.

TCG 파싱 과정에서 이와 같은 기대 테스트를 하여야 하는 상황에는 다음 두가지가 있다. 첫번째 경우는 스택의 정상에 있는 트리에 루트 생성 규칙에 의해 새로운 루트를 붙이는 경

우이다(그림10-a). 이때는 새로 생성되는 노드 Q가 T_i의 기대(expectation)를 만족시키는 경우에만 규칙을 수행한다. 이를 위한 기대 테스트는 다음과 같이 나타낼수 있다.

기대 테스트-1: (T_i에 의해 어떤 M이 기대되고 M은 Q로 부터 만들어 질수 있다.)

$\exists M$ s.t. (expt(T_i)$\ni$M
& leftedge(M)$\ni$Q), for some nonterminal M

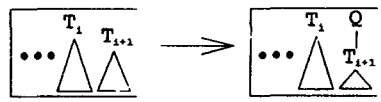
이 기대 테스트가 실패하는 경우는 패턴 매칭 검사가 실패하는 것으로 간주하여 현재의 규칙 적용이 실패하게 되며 실행부는 작동되지 않는다.

기대 테스트를 하여야 하는 두번째 상황은 새로운 단어를 스택으로 읽어들여(shift) 올 때이다(그림10-b). T_i가 스택의 가장 윗부분에 있는 트리이고, W_j를 새롭게 읽혀 들여지는 단어라고 할때 W_j는 새로운 트리 T_{i+1}가 되며 따라서 이 트리는 T_i의 기대를 만족하여야 한다. 이 기대 테스트는 스택에 적용되는 규칙이 없을 때마다 새로운 단어를 읽어 들이기 직전에 이루어진다. 이 기대 테스트에 합격되어야 만 비로소 새로운 단어는 스택으로 읽혀들여진다. 이 검사는 다음과 같이 나타낼수 있다.

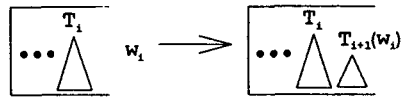
기대 테스트-2: (T_i에 의해 어떤 M이 기대되고 M은 W_j로 부터 만들어 질수 있다.)

$\exists M$ s.t. (expt(T_i)$\ni$M
& leftedge(M)$\ni$preterminal(W_j))
, for some nonterminal M

이와 같은 기대 테스트를 하므로써 파싱이 전체적인 문법에 맞게 진행되도록 한다.



(a) 기대 만족에 의한
루트 생성 규칙의 수행



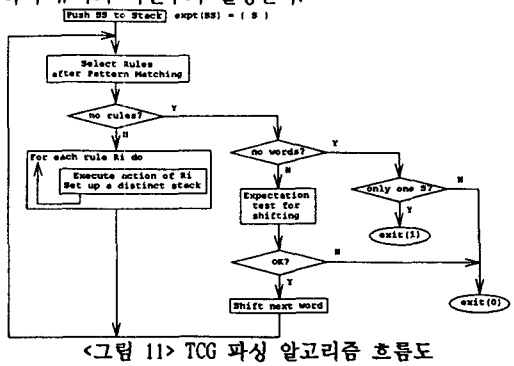
(b) 기대 만족에 의한
새로운 단어의 shift

<그림 10> 기대 테스트를 위한 두가지 경우

3.2 TCG 파싱 알고리즘

TCG 규칙은 인공지능에서 전문가 시스템(production system)의 패턴/액션 규칙과 같다. 따라서 우리 파싱 알고리즘도 패턴을 검사하고 이에 만족된 규칙의 액션을 행하는 루프 구조로 이루어 진다. 알고리즘의 모습은 <그림11>과 같다.

알고리즘의 시작은 빈 스택에 SS라는 시작을 알리는 문법 심볼을 넣으면서 시작한다. SS는 파싱을 시작한다는 의미로만 이용된다. SS의 기대치는 { S }로 지정한다. 그 다음은 규칙들을 현재의 스택과 패턴 매칭하는 것으로 루프를 시작한다. 각 규칙의 매치는 규칙의 패턴부를 스택과 비교하여 이루어진다. 여기에서 패턴부가 만족되면 그 규칙이 매치된 것으로 판단하여 규칙의 액션부가 실행된다.



<그림 11> TCG 파싱 알고리즘 흐름도

하나의 스택에 대해 복수개의 규칙이 매치될수 있다. 이것은 모호성이 발생하기 때문으로, 복수개의 규칙이 매치될 때에는 각 규칙마다 독립된 스택이 생성되며 각 스택은 병렬적으로 수행한다. 각 스택은 하나의 파스을 나타내며 다중 스택은 다중 파스을 나타낸다. 각 스택들은 각각이 독립적인 파싱의 진행을 나타내며, 따라서 각 스택은 또 하나의 새로운 스택처럼 위의 알고리즘에 똑같이 적용되어 파싱을 진행해 나간다.

하나의 스택에 대해 패턴 검사에서 매치되는 규칙이 없을 때에는 새로운 단어를 스택으로 읽어 들이게 된다(shift). 새로운 단어를 읽어들일 때는 반드시 스택의 가장 꼭대기 트리에 의한 기대 테스트를 한다(기대 테스트-2). 기대 테스트가 실패하면 이 파스는 적법하지 않으므로 제거된다(exit(0)). 기대를 만족시키면 새로운 단어를 스택으로 읽어 파싱을 계속한다.

전체 문장에 대한 파싱이 끝나 더이상 읽어 들여 올 단어가 없을 때에는 한 스택 내에 전체 문장을 포괄하는 오직 하나의 S만 생성되었는지를 검사한다. 하나의 S만 존재한다는 점은 하나의 문장에 대해 파싱이 끝나 S를 루트로 하는 하나의 파싱 트리가 구성되었다는 뜻이므로 파싱은 성공한 것이다. S의 검사에 따라 성공적인 종료(exit(1))와 실패된 종료(exit(0))로 나뉘어 해당 파스를 종료한다.

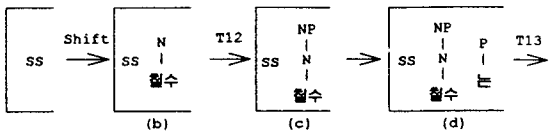
CFG	TCG
C5: S -> NP S	T9: [NP] | first(S) => S | | NP NP T10: [S] [S] => S | / \\ NP NP S
C∈: S -> VP	T11: [VP] => S | VP
C7: NP -> N P	T12: [N] | first(P) => NP | | N N T13: [NP] [P] => NP | / \\ N N P
C8: VP -> V E	T14: [V] | first(E) => VP | | V V T15: [VP] [E] => VP | / \\ V V E

<표-3> 한국어 파싱을 위한 CFG와 TCG 규칙

3.3 한국어 분석 과정

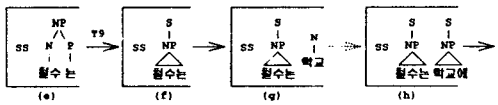
TCG 파싱 알고리즘의 이해를 돕기 위해 간단한 한국어 문장에 대한 파싱 과정을 살펴보자. 이를 위해 <표-3>과 같이 CFG 규칙 및 이에 따르는 TCG 규칙을 준비한다.

S는 전체 문장을 나타내는 문법 심볼이다. NP는 명사구를 나타내고, VP는 동사구를, V와 N은 각각 동사와 명사를 나타낸다. V와 E는 각각 동사와 어미를 나타내는 문법 심볼이다. 이 문법을 사용하여 '철수는 학교에 간다.'라는 간단한 문장을 파싱해 보자.



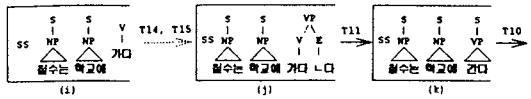
작성된 TCG를 적용하기 전에 스택에는 먼저 (a)와 같이 SS가 들어 있다. 이 스택을 매치하는 TCG 규칙은 없으므로 기대 테스트를 통과하면 다음 단어를 읽어들이게 된다. 이때 expt(SS) = { S }이고 leftedge(S) = { S, NP, VP, N, V }가 된다. 다음 단어 '철수'의 preterminal은 N이므로 기대 테스트-2를 만족하며 따라서 스택에는 다음 단어가 입력된다(b).

스택의 꼭대기에 N 노드가 생성된 상태에서 규칙의 적용을 계속하면 루트 생성 규칙인 T12가 매치된다(이 루트 생성 규칙은 기대 테스트-1을 만족하므로 이 규칙이 매치된 것이다). 따라서 규칙의 액션부에 의해 (c)와 같이 NP가 루트인 트리가 생성된다. 이때 만들어지는 NP 트리는 완성되지 않은 트리이다. 따라서 이 스택에 매치되는 규칙들이 없게 되고 새로운 단어를 읽어들이게 된다. NP 트리에 의한 기대는 NP 노드가 미완성 상태이며 다음 붙여야 할 자노드는 P 이므로 expt 함수값은 { P }이다. 다음 단어 '가'의 preterminal은 P이므로 기대 테스트-2를 통과하며, 따라서 다음 단어가 스택에 읽혀들여진다(d).

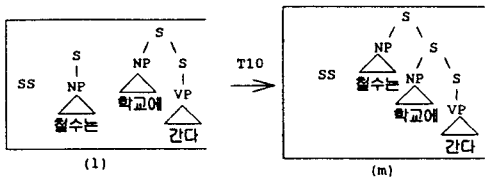


이 스택은 T13 규칙에 의해 P가 NP에 자식 노드로 달리게 되고, 이로서 NP 트리는 완성이 된다(e). 따라서 NP는 완성의 상태가 되고, T9에 의해 S를 루트 노드로 붙이게 된다(f).

이 스택에 매치되는 규칙이 없게 되고, 스택에 새로운 단어를 읽어들이기 위해 S에 의거한 기대 검사를 한다. S 노드는 미완성 상태이며 다음에 붙여야 할 자노드는 S 이므로 expt(S) = { S }이고, leftedge(S)는 위에서 구한 것과 동일하게 { S, NP, VP, N, V } 임을 알수 있다. 다음 단어 '학교'의 preterminal이 N이므로 기대 테스트를 만족시키게 되고 이 단어가 스택으로 읽혀들여 진다(g). 이 N 트리는 앞서의 '철수는'과 마찬가지 과정에 의해서 '학교에'가 (h)에서와 같이 S 트리가 된다.



이 S 트리의 S 노드는 미완성 상태이므로 T10의 패턴은 매치되지 않는다.(주:PE1은 완성된 노드이어야 매치된다.) 결국 아무 규칙도 매치되지 않으나 기대 테스트-2를 통과하므로 다음 단어인 '가다'가 스택에 입력된다(i). 스택에 들어온 '가다'는 곧장 T14에 의해 VP 트리로 만들어 지며 T15에 의해 여기에 어미 'ㄴ다'가 붙어 완성된 VP가 된다(j). (j)에서 보이는 VP는 완성된 것이므로 T11에 의해 S를 루트로 갖는 트리가 된다(k).

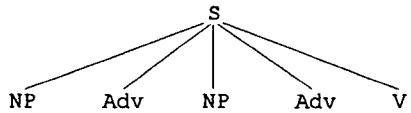


(k)의 스택 정상의 S는 완성 상태이며 따라서 T10이 매치되어 이 규칙에 의해 (l)과 같은 모습이 된다. 이 스택에 대해 다시 T10이 매치되어 수행되고 스택은 (m)이 되며 이로서 파싱이 끝난다.

4. 한국어 파싱을 위한 TCG의 확장

4.1 확장된 문맥 자유 문법(ECFG)의 필요성

한국어 문장은 가장 주된 단어인 동사가 가장 마지막에 나오고, 이 앞에 나오는 부사와 명사구들간의 위치는 자유로운 부분 자유 어순의 특성을 가진다. 이러한 한국어의 문장 구조는 다음과 같이 평면적인 구조로 나타내면 편리하다[3]. 문장내의 NP 구문들은 같은 등급의 계층 구조로 파싱이 되어야 한다(그림12)



<그림 12> 평면 구조

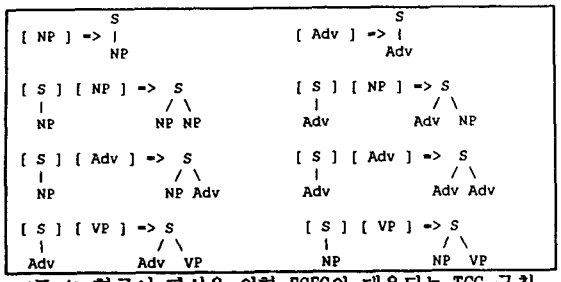
위 3.3에서 보인 CFG는 이와 같은 평면 구조를 생성할수 없으며 따라서 문제가 있다. 이를 해결하기 위해서는 다음 규칙에서처럼 ECFG(Extended Context-Free Grammar)의 사용이 필요해 진다[3].

```
E1:  S -> (NP|Adv)* VP
```

ECFG 문법에 대한 파싱은 일반적인 CFG 알고리즘을 이용할 수 없다. 그러나 TCG 파싱 방법을 확장하여 쉽게 ECFG의 파싱

[3] |은 alternative, *은 반복을 나타낸다.

을 구현할수 있다. 위의 ECFG 규칙에 대한 TCG 규칙은 <표-4>과 같다.



<표-4> 한국어 파싱을 위한 ECFG와 대응되는 TCG 규칙

4.2 중심어 후행, 오른쪽 우선 파싱을 위한 TCG의 확장

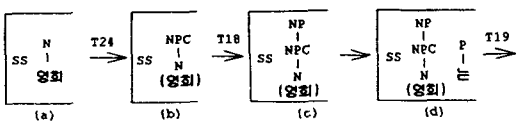
문장의 구조에서 동사는 문장의 중심되는 의미를 전달하며 따라서 이것을 중심어라 부른다. 명사구와 동사에 대한 관계를 밝히는 것이 파싱의 주된 임무중 하나이다. 이때 명사구와 동사의 위치 관계는 배제할수 없는 정보이다. 주로 동사에 가까운 명사구가 그 동사의 격을 채울 가능성이 크므로 문장에서 동사를 먼저 찾은 후, 여기에서 시작하여 오른쪽에서 왼편으로 파싱을 진행시키면서 동사에 가장 가까운 명사구들을 먼저 동사에 연결시키는 방법이 필요하다. 즉 아래에서 NP_k, NP_{k-1}, ..., NP_1 순으로 V에 연결된다.

$$NP_1 \quad NP_2 \quad \cdots \quad NP_k \quad V$$

이러한 방식을 역방향 파싱, 혹은 오른쪽 우선 파싱이라 부르며, 한국어와 같은 중심어 후행 언어에서 유용한 파싱 방법이다. 지금까지 소개한 TCG 파싱을 확장하면 중심어 후행의 오른쪽 우선 파싱도 가능하게 된다. 이를 위해 <표-5>와 같은 ECFG 규칙을 만들고, 이에 따르는 TCG 규칙을 구성하였다. 여기에서 $S_{[mod]}$는 문법 범주는 S이나 [mod]특성(feature) 즉 관형형 특성을 가진 절; 즉 관형절을 뜻한다. NPC는 명사구에서 조사를 제외한 부분을 나타낸다. Adj는 관형사이다.

오른쪽 우선 방식은 T16과 T17에 의해 구현된다. 즉 용언구가 나온 후에야 S 루트를 만들며(T16), 그후 오른쪽 NP 부터 S의 왼쪽 자식으로 붙여 나간다(T17). 그러나 NP의 구축, VP의 구축은 과거와 같이 순방향(left-to-right)로 진행한다(T18 ~ T26). T17의 액션에서 왼편으로 경사지게 한 연결 표현(/)은 S 노드가 루트인 트리에 대해 자노드 NP를 붙일때는 맨 왼편 자식이 되게 붙인다는 뜻이 된다. 이것은 오른쪽 우선 파싱을 가능하게 만드는 규칙이다.

이렇게 구성된 규칙을 바탕으로 '영희는 철수가 쓴 모자를 좋아한다.'라는 문장을 분석하여 보자.

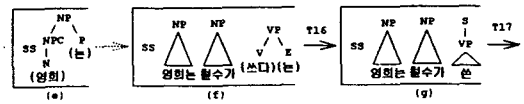


기본적인 파싱 알고리즘은 3.2에서 설명한 바와 동일하다. SS가 먼저 스택에 들어온다. expt(SS) = { S }, leftedge(S) = { S, NP, NPC, $S_{[MOD]}$, Adj, N }이며 다음 단어는 명사인 '영희'이고 기대를 만족 시키므로 입력된다(a). 스택에 들어온 '영희'는 T24에 의해 NPC 루트로 된다(b). 새로 생성된 NPC는 T18 규칙에 의해 NP 노드를 새로운 루트로 생성할수 있다(c).

ECFG	TCG
E2: S -> NP* VP	T16: S / [VP] => VP T17: S / [NP] [S] => NP
E3: NP -> NPC P	T18: NP / [NPC] | first(P) => NPC T19: NP / [NP] [P] => NP, NPC, P
E4: NPC -> $S_{[mod]}$ NPC	T20: NPC / [$S_{[mod]}$] | first(NPC) => S T21: NPC / [NPC] [NPC] => S, NPC
E5: NPC -> Adj N	T22: NPC / [Adj] | first(N) => Adj T23: NPC / [NPC] [N] => Adj, Adj N
E6: NPC -> N	T24: NPC / [N] => N
E7: VP -> V E	T25: VP / [V] | first(E) => V T26: VP / [VP] [E] => V, V E

<표-5> 중심어 후행, 오른쪽 우선 파싱을 위한 ECFG와 TCG 규칙

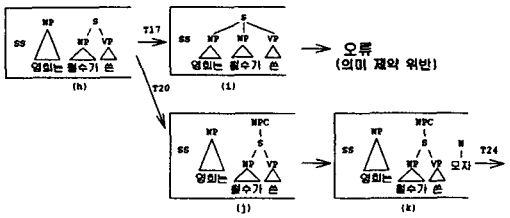
여기에서 주의 해야 할점은, 현재의 NP 트리를 만들기 위해 적용된 루트 생성 규칙은 앞서 SS에 의한 기대를 만족시키므로(기대 테스트-1) 수행 된 것이다. 현재 스택에 대해서는 더이상 매치시킬 규칙이 없으므로 기대 테스트-2를 하게 되고 NP 트리의 가장 하위 미완성 노드인 NP에 대해 요구되는 다음번 노드가 조사 P임을 알수 있다. 따라서 조사인 '는'이 스택에 들어오게 된다(d).



스택에 들어온 조사는 T19 규칙에 의해 NP 트리를 완성하게 된다(e). 같은 방식으로 '철수가'에 대한 NP 트리가 구성되고[4], 뒤 이어 나오는 동사 '쓴'도 원형 동사인 '쓰다'와 어

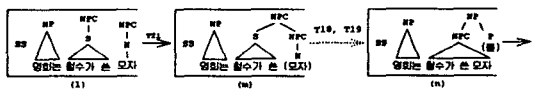
[4] ECFG 규칙에 따른 중심어 후행, 오른쪽 우선을 위한 규칙에서의 기대 테스트는 앞에서 설

미인 '는'으로 구분되어 스택에 들어와 VP 트리를 구성한다 (f). '쓰다'와 '는'으로 구성된 VP 트리로부터 T16 규칙에 의해 S가 만들어 진다(g).

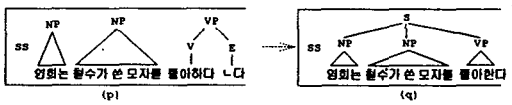


이 스택에 대해서 오른쪽 우선 분석 규칙인 T17 규칙이 매치되며 NP '철수가'가 S의 왼쪽 자노드로 붙여진다(h). T17 규칙은 오른쪽 우선 파싱을 이끄는 규칙이다. 중심어로 등장한 동사 '쓴'에 대해 역방향으로(오른쪽에서 왼쪽으로) 나아가면서 파싱 트리를 확장하도록 한다.

스택(h)에 대해서 2개의 규칙 T17, T20이 매치되며 스택은 (i)와 (j) 두개로 갈라진다. (i)는 '영희는'을 '쓴'에 연결시킨 것이나, (j)는 '영희는'이 없이 S를 닫는(close) 경우이다. 그러나 (i)로 진행된 스택은 '영희는'이 '쓴'의 목적격을 채워야 하나 의미 제약을 위반하므로 파싱이 실패되며 스택은 소멸된다. 스택 (j)에 대해 아무 규칙도 매치되지 않으나, NPC 트리는 N을 기대하게 되고, 이에 의해 '모자'가 스택에 들어온다(k).



스택(k) 에 대해서는 T24 규칙에 의해 NPC가 생성된다(l). 이 스택에 대해서도 두개의 규칙이 적용 가능하다. 하나는 (m)으로 가게하는 T21이고, 또 하나는 T18이다. 루트 생성 규칙인 T18은 기대 테스트-1에 실패하므로 적용되지 않는다. (m)에서 만들어진 NPC 트리는 T18 규칙에 의해 NP 트리로 바뀌고, 뒤를 이어 나오는 '를'은 NPC의 기대를 만족하므로 스택에 읽혀져 T19 규칙에 의해 (n)과 같은 모습이 된다. 이로서 NP 트리는 완성 상태가 된다.



'좋아하다'는 V가 되며 'ㄴ다'는 E 이므로 (p)에서와 같이 VP가 생성되고(T25, T26), 여기에서 T16에 의해 S가 생성된후 T17 규칙이 두번 적용되어 (q)와 같은 완성된 파스를 얻는다.

5. 결 론

본 논문에서는 문맥 자유 문법으로 기술된 문법을 파싱할 수 있는 새로운 파싱 방법인 TCG 파싱을 소개 하였다. TCG 파싱을 위해서는 먼저 CFG 문법을 TCG 문법으로 변환한 후, TCG 문법 규칙에 의거하여 파싱을 수행 한다. 따라서 TCG는 CFG와 같은 문법 기술 능력을 갖는다. TCG 파싱은 점증성, 하향성, 상향성, 유연성등과 같은 장점을 갖는다. 이러한 기본 TCG 파싱을 확장하여 ECFG를 파싱할수 있게 하며, 이로서 부분 자유어순, 중심어 후행 특성을 가진 한국어를 효과적으로 분석할 수 있다. 특히 동사와 명사구 사이의 관계를 결정하는 데 있어서는 오른쪽에서 왼쪽으로 진행하는 파싱 방법을 개발하였다.

명한 CFG의 경우에 대한 기대 테스트와 달라 진다. 여기서의 E2, T16, T17의 경우에는 NP 트리는 NP와 VP를 기대하도록 하여야 한다.

[참고 문헌]

[1] 나동렬, '패턴-액션 규칙을 이용한 한국어 구문 분석,' 제4회 한글및 한국어 정보처리 학술발표 논문집, pp. 131-140, 1992.

[2] 나동렬, '병렬 분석과 의미 여과를 이용한 자연어 파싱,' 한국 정보 과학회 논문지, 20권 2호, pp. 264-277, 1993.

[3] 나동렬, '한국어 파싱에 대한 고찰,' 한국 정보 과학회지, 12권 8호, pp. 33-46, 1994.

[4] Aho, A. V. and J. D. Ullman, The Theory of Parsing, Translation and Compiling, Englewood Cliffs, NJ: Prentice Hall, 1972.

[5] Earley, J., 'An efficient context-free parsing algorithm,' Communications of the ACM, Vol. 13, No. 2, pp. 94-102, 1970.

[6] Lytinen, S. L., 'Dynamically Combining Syntax and Semantics in Natural Language Processing,' Proceedings of the 5th National Conference on Artificial Intelligence, pp. 574-578, 1986.

[7] Marcus, M., A Theory of Syntactic Recognition for Natural Language, Cambridge, MA: The MIT Press, 1980.

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