국어 '와/과' 문장의 통사론과 의미론*

-어휘주의적 분석과 프롤로그로의 구현-

양 정 석
(원 광 대)

- 목 차 -

1. 서 론

이 연구의 목적은 어휘주의적 관점에서 국어의 '와/과' 문장들을 분석하고, 이를 프롤로그 언어로 형식화하여 구문분석과 의미분석이 기계적으로 이루어지도록 구현하는 것이다.[1)]

이 연구에서 다룰 '와/과' 문장이란 조사 '와/과'를 포함하는 문장들을 널리 지칭하는 것이다. 이들 구문에 대한 초기의 연구에서는 이들 모두를 단일한 원리로 묶어 설명하고자 하였으나, 더 깊이 있는 분석이 진행됨에 따라 매우 상이한 통사·의미적 특징을 갖는 여러 개의 구문으로 나뉘어진다는 점이 인식되기에 이르렀다. 홍재성(1985가,나 및 1986)을 따라서, (1)-(4)를 각각 대칭구문, 상호구문, 동반구문, 경쟁구문이라 부르겠다.

(1) 가. 철수와 영호는 싸웠다.
　　나. 철수는 영호와 싸웠다.
(2) 가. 철수와 영호는 서로 공격했다/쳐다보았다.
　　나. 철수는 영호와 서로 공격했다/쳐다보았다.

---

*이 논문은 1996년도 한국학술진흥재단 공모과제 연구비에 의하여 연구되었음.

1. 구문분석과 의미분석은 각각 parsing, semantic analysis를 옮긴 것으로, 전산기를 통한 통사구조의 분석, 전산기를 통한 의미구조의 분석이란 뜻으로 좁혀 사용된다. '어휘주의'란 용어를 취하는 것은, 종래 국어학계에서 어휘와 관련한 독립된 연구 영역을 지칭하는 데에 '어휘론'이란 용어를 쓰는 일이 종종 있기 때문에 이와 구별하기 위하여 시도해 보는 것이다.

(3) 가. 철수와 영호는 기차를 탔다.
나. 철수는 영호와 기차를 탔다.
(4) 가. 철수와 영호는 서로 먼저 선물을 타려고 했다.
나. 철수는 영호와 서로 먼저 선물을 타려고 했다.

(1)-(2)의 예에서 (가)와 (나)의 두 문장은 의미상으로 '교호성(reciprocity)'이라는 특질을 갖는다는 점에서 공통되며, 구조적으로도 접속의 '와'(가)와 여동(與同)의 '와'(나)를 취하는 통사구조를 가진다는 점에서 공통된다. (1)의 대칭구문에서는 대칭동사 '싸우다'가, (2)의 상호구문에서는 부사 '서로'가 갖는 내재적 특질이 이러한 구문의 형성에 결정적인 역할을 한다. 이에 비해서 (3)의 경우는 동사의 내재적 특질은 '와'의 출현에 영향을 주는 바가 뚜렷하지 않다. 이밖에 '와/과' 문장을 이루는 독특한 예로서 (4)와 같은 경쟁구문이 더 있지만, 이 연구에서 구체적으로 다루지는 않는다.

조사 '와/과'를 갖는 문장들은 국어에 대한 생성문법적 연구의 이른 단계에서부터 많은 관심을 받아 왔는데, 아직까지 변형론적 혹은 생성의미론적 분석 태도가 지배적이라고 할 수 있다. Lakoff & Peters(1966)의 방법을 원용한 김완진(1970)을 시작으로, 김영희(1974), 최재희(1985) 등에서 '와/과'를 갖는 구문을 변형론적으로 분석하였다. 김완진(1970)에서의 분석이 주로 접속항 이동변형에 촛점이 맞추어져 있었다고 한다면, 김영희(1974), 최재희(1985)는 대칭구문을 중심으로 한 교호적 구문들을 접속문 축약의 방법을 이용하여 도출하는 점이 특징이다.

홍재성(1985가, 1985나, 1986)에서는 '와/과' 문장들을 '대칭구문, 상호구문, 동반구문, 경쟁구문' 등으로 나누어 각 구문의 통사적 특징을 자세히 분석하였다. 비록 Harris식의 변형 개념을 사용하고 있지만 어휘주의적 분석에 가깝다고 하겠다. 남기심(1990)에서도 어휘주의적 입장에서 '와' 구문에 대한 과거 변형론적 분석의 문제점을 논하고 있다. 그러나 이들은 대칭/상호구문이 갖는 의미구조의 특이성을 중심으로 통사구조와의 연관의 과정을 설명해 주지 못하고 있어 통합적인 이론에의 요구는 아직 충족되지 못하였다고 할 수 있다.

'와/과' 문장에 대해서 변형론적 접근 방법이 옳지 않음은 필자의 앞선 연구 양정석(1996가, 1996다)에서 논한 바 있다. 그 곳에서는 '대칭동사'와 '와/과'가 상호작용함으로써 이루는 '대칭구문', 부사 '서로'와 '와/과'가 상호작용함으로써 이루는 '상호구문'을 중심으로 어휘주의적인 대안을 개괄적으로 제시한 바 있다. 그러나 통사구조와 의미구조의 연결 과정에 대해서 세밀한 기술을 요하는 면이 있고, 대칭/상호구문 이외의 '와/과'를 포함하는 문장들의 통사·의미적 특징을 일관성 있게 기술해 줄 필요를 느끼게 되

었다. 특히, 프롤로그로의 논리적인 구현을 통하여 그 실행가능성의 한계를 구체적으로 드러내는 것이 이들 구문은 물론 국어 문장 구조 전반의 이해에도 도움이 될 것으로 생각한다.

대칭·상호구문을 중심으로 한 '와/과' 문장의 현실을 있는 그대로 기술해 주기 위해서는 그 의미구조에 대한 명시적인 형식화가 필요하다는 것이 필자의 기본적 관점이다. 과거 변형론자/생성의미론자들에 의한 통사구조의 분석에서 의미구조의 성격을 갖는 기저구조가 논의되었던 것은 이러한 필요성에 따른 것이다. 이 연구에서는 과거 변형론자들처럼 통사구조의 논의에 의미구조를 직접 관련시키되, 어휘부의 정보나 통사구조를 기초로 의미구조를 해석하는 방향을 취한다. 특히 도출의 과정에서 어휘부를 다시 참조할 수 있다고 보는 관점은 근래의 원리 매개변인 이론에서와 다른 특징이라 하겠다.

초기의 해석의미론적, 어휘주의적 분석을 대변한다고 할 수 있는 Jackendoff(1972)와, 그 이래로 Jackendoff(1983, 1990) 등에서는 동사의 어휘의미적 구조를 술어해체 분석에 입각해서 기술하고, 그를 바탕으로 한 통사구조로의 연결 또는 대응(correspondence), 그리고 통사구조로부터 의미구조로의 연결, 대응의 과정을 연결이론(Theory of Linking)이라는 이름으로 체계화하고 있다.[2] 이 이론은 GB이론을 비롯한 형식적인 통사론의 중요한 성과들을 수용하면서도 이들에게 결핍된 의미구조에 대한 구체적인 전망을 아울러 제공해 주는 것으로 보여 희망적이다. Montague 의미론이 의미구조의 체계화를 위한 한 대안이 될 수도 있겠지만, 아직 그 표현력이 풍부하지 못하고, 의미 표상의 형식이 통사적인 구조와 매우 거리가 있다는 약점이 지적된다. 특히 국어의 '와/과' 문장의 통사구조를 정확하게 기술해 주기 위해서는 대칭동사와 부사 '서로'가 공통적으로 가지는 '교호성'의 국면에 주목해야 하는데, 이들이 갖는 형식상의 공통성을 명확하게 나타낼 수 있는 문법적 층위는 의미구조의 층위이다. '와/과' 문장을 통사론적으로 분석해 주기 위해서는 이러한 의미구조의 형식 자체를 고려하는 것이 중요하다. 이런 점에서 Jackendoff(1990)의 연결이론과 의미구조 분석 방법은 우리의 목적과 가장 부합한다.

현대 생성문법의 연구 방법에서의 특징은 한 문장에 대해서 두 개 이상의 구조 표상을 설정하고, 이들 사이에 존재하는 규칙을 제시하고자 한다는 점이다. 표준이론에서는

2. 보통 연결이론은 통사적인 논항구조가 결정되는 데에 관여하는 의미론적인 제약을 기술하는 것으로 알려져 있다. 이에 반해, Jackendoff(1990)에서는 통사구조로부터 의미구조(그의 용어로는 '개념구조(conceptual structure)'이다.)로의 연결 과정을 기술하려고 한다. 이 연구에서도 후자의 관점에 따른 연결이론을 받아들이고 있다.

통사구조로서 심층구조와 표면구조, 그리고 음성 표상과 의미 표상을 합하여 네 가지의 표상 층위를 상정하였다. 최근의 이른바 최소주의 이론에서는 공식적인 표상 층위를 논리형태(LF)와 음성형태(PF)의 두 가지로만 한정하고 있다. 필자는 통사구조로서 S구조에 해당하는 것 한 가지만을 인정하고, 문법적인 차이로 반영되는 모든 의미적 차이를 표상하는 층위로서 '의미구조'를 상정하기로 한다. 이는 Jackendoff(1990, 1994)에서 그리고 있는 문법의 조직에 대한 관점과 대체로 같은 것이다. 그러나 구체적인 구구조(phrase structure)의 기술에서는 핵계층 이론을 기능 범주에까지 확대한 Chomsky(1986)의 체계를 부분적으로 수용한다.

이 글의 논의 순서는 다음과 같다. 첫째, 형식적 체계로서의 프롤로그에 대해 소개하며, 이것의 특징과 기본적인 사용례를 보이고, 구문분석의 원리를 설명한다. 다음으로, 국어에서 (1), (2), (3)과 같은 대칭구문, 상호구문, 동반구문이 갖는 통사·의미적 특징을 어휘주의적/해석의미론적 관점에서 기술하여 이들의 통사구조, 의미구조를 확정한다. 셋째로, 이러한 분석 결과를 구문분석기(parser)와 의미분석기(semantic analyzer)로 구현한다.

2. 프롤로그 구문분석의 기초

2.1. 프롤로그 언어의 특징

생성문법 이론은 전산적인 처리 모형을 바탕으로 시작되고 근래에 이르기까지 전산처리와의 연계를 가지고 발전해 왔다. 프롤로그(PROLOG)는 그 이름이 말해 주듯이 논리 프로그래밍(Programming in Logic) 언어이다. 논리적으로 또는 수학적으로 어떤 식을 증명해 가는 일을 기계적으로 수행하도록 구현된 것이다. 그러므로 한 문장에 대해서 구구조 규칙이나 기타 규칙들이 적용 가능하냐 여부를 검증함으로써 그것이 문법적/비문법적 문장임을 판정하는 방식으로 전개되는 생성 통사론의 이론을 그 프롤로그로의 구현 가능성을 통해서 고찰해 보는 일은 매력적인 연구 과제가 아닐 수 없다. 프롤로그 언어는 어떤 문법적 설명이 논리적으로 일관성과 체계성을 갖느냐를 엄밀한 차원에서 검증하는 데에 이용될 수 있는 것이다.

프롤로그 언어가 가지는 형식적 특징 및 기본적인 표현에 대해서 정리해 보기로 한다. 프롤로그 언어는 술어논리 언어의 한 방언이다. 프롤로그는 이렇듯 하나의 언어체계이므로 어휘 체계와 문법을 가지고 있다. 그 어휘는 a, b, c 등과 같은 로마자 소문

자로 시작하는 술어(predicate)와,[3] 영향권(scope)을 표시하는 '(' 및 ')', 그리고 X, Y, A, B 등 로마자 대문자로 시작하는 변수(variable)들로 구성된다.

Xab나 A_a12 따위도 변수의 형식으로 적격(well-formed)하다.[4] 특히 밑줄 '_'로 시작하는 '익명의 변수(anonymous variable)'의 쓰임은 주목되는 바 있다. 이는 변수의 하나이나 예거(instantiation)를 요구하지 않는 형식이므로 논항의 자리가 존재함을 보이기만 할 필요가 있을 경우에 편리하게 사용될 수 있다.

기본적인 표현, 즉 항(term)은 사실(fact)이나 규칙(rule)이나 연접(conjuntion)으로 이루어진다. 항은 X, Y, A 등의 변수로 나타날 수도 있다. 프롤로그의 표기법은 술어논리의 표기법이 그대로 이용된다. '사실'은 술어와 논항으로 구성되고 논항들은 소괄호로 둘러싸도록 되어 있다. 논항은 0 자리, 한 자리, 나아가 무한한 수로 가질 수 있다.(모든 항은 반드시 ' . '으로 끝나야 한다.) 다음은 '사실'이 나타날 수 있는 형식을 네 가지만 예시해 본 것이다.

```
a.        ac(b, c).        bd(X, Y, z).        [a,b,c].
```

세번째의 예로 든 사실은 세 자리의 함수자 bd가 논항으로서 두 개의 변수 X, Y와 상수 z를 취하는 것으로 되어 있다.[5] 네번째의 대괄호로 둘러싸인 형식은 '목록(list)'이다. 이는 '.'을 술어로 하는 사실 '.(a,.(b,c)).'와 꼭 같은 내용을 갖는다.

규칙은 머리(head)와 목(neck)과 몸(body)으로 이루어진다. 다음이 규칙의 한 가지 보기이다. 이 규칙에서 a(X, Y)가 머리 부분이고 :-는 목이며 b(X, c, Y, d)는 몸 부분이 된다. ':-'는 형식논리학의 조건(conditional) 연산자 '<-'와 대응하는 프롤로그의 기호이다.

```
a(X, Y) :- b(X, c, Y, d).
```

앞에서 말한 '사실'들은 규칙에서 목과 몸이 생략된 형식이라 할 수 있다.

또 하나의 중요한 표현은 '연접(conjunction)'이다. 연접 기호 ','를 사이에 두고서 항

---

3. predicate란 용어를, 술어논리나 프롤로그 언어의 관계 명칭을 가리킬 때는 '술어'라 지칭하고, 보통의 문법 용어로는 '서술어'라고 부르겠다.
4. 항의 첫머리에 있는 '_'만이 익명의 변수를 표시한다. 'A_a12'와 같은 경우의 '_' 표시는 익명의 변수와는 관계 없고, 보통의 문자일 뿐이다.
5. 방금 이 예와 같은 것을 입력하여 프롤로그 시스템에 불러 오게 되면 'instantiation(예거(例擧)로 번역하기로 한다.)가 안되었다는 경고 신호가 나타난다. 이와 같은 표현이 독립적인 형식으로 설정될 것이 요구된다면 'X', 'Y'를 '_X', '_Y' 따위의 익명의 변수로 바꾸면 된다.

들이 이어질 수 있다. 다음 (가)와 같은 것이 그 형식을 보여주는데, 이 형식으로는 질의(query)의 목표(goal)로만 쓰이거나, 또는 (나)처럼 규칙의 '몸'의 일부로만 쓰인다.

가. a(X,Y), b(Y).
나. c(X) :- a(X,Y), b(Y).

프롤로그의 기계적인 작동 원리는 한 마디로 단일화(unification)라고 할 수 있다.[6] 단일화는 가장 일반적인 단일화자(unifier)[7]를 발견하여 적용하는 과정이다. 프롤로그가 어떤 식을 증명하기 위해서는, 목표(goal)가 되는 식을, 단위식(unit clause 또는 literal) 별로, 이미 주어진 공식들과 비교하여 패턴의 부합 여부를 검사하게 된다. 이를 위해서, 먼저 변수에 새로운 이름을 붙여 변수들 간의 혼동을 막아야 하고, 또 가장 일반적으로 '대치'되는 방식을 추구하여야 한다는 기본적인 추론 전략이 전제된다. 이와 같은 기본 전략을 resolution이라고 한다.[8]

단일화의 알고리즘을 다음과 같이 정의할 수 있다.

(1) k = 0, 그리고 $\sigma_0 = \epsilon$라고 설정한다.
(2) 만약 $E\sigma_k = F\sigma_k$이면 정지하라. 이 경우 σ_k는 S의 mgu이다. 그렇지 않으면 $E\sigma_k$와 $F\sigma_k$가 거기에서 달라지는 첫 번째 기호를 발견하기 위하여 $E\sigma_k$와 $F\sigma_k$를 왼쪽으로부터 오른쪽으로 비교하라. 그 기호로 시작하는 E의 부분표현(subexpression) E'를 선택하고, 그 기호로 시작하는 F의 부분표현 F'를 선택하라.
(3) 만약 E'와 F'의 하나가 변수 V이고 하나가 항 t이며, V가 t의 (엄밀)부분구성성분 (subconstituent)으로 나타나지 않으면, $\sigma_{k-1} = \sigma_k(V/t)$로 설정한 다음, k를 k+1로 증가시키고 2단계로 돌아가라. 그렇지 않으면 정지하라. 이 경우 S는 단일화될 수 없다.

이 알고리즘은 변수들을 새로 이름지음에 따라 유일하게 된 가장 일반적인 단일화자(mgu)를 산출한다. 그러지 못하면 이 알고리즘은 종결되고, 표현들이 단일화될 수 없다는 판단을 내놓게 된다.

---

6. Shieber(1986)에서 간명히 요약·소개되고 있는, GPSG나 LFG나 HPSG 등의 기본 연산으로서의 단일화는 프롤로그 체계의 기본 작동 원리로서의 단일화와 구별될 필요가 있다. 본질적으로는 둘이 동일한 원리이지만, 후자는 두 특질 구조가 합쳐져서 하나의 특질 구조를 이루어 가는 과정을 일컫는데 비해서 전자는 다음에 제시하는 것처럼, 특별한 방식으로 표현들의 패턴을 합치시키는 과정이다.
7. 'most general unifier'. 이를 아래에서는 mgu로 표시하였다.
8. 단일화 및 resolution의 개념에 대해서 Pereira & Shieber(1987: 62-65) 참조.

2.2. 구문분석의 세 가지 전략

국어를 문맥자유 문법(context-free grammar)에 의해서 기술할 수 있는 언어(문맥자유 언어)라고 가정하기로 하면, 국어의 문법 G는 다음과 같은 형식으로 기술된다.

$$G = \langle V, \Sigma, P \rangle$$

여기서 V는 이 문법을 이루는 유한수의 어휘들(어휘체: vocabulary)이고, Σ는 $\Sigma \subset V$ 즉 단말 기호열[9]로 나타날 수 있는 기호들(단말 기호: terminal symbols)이며, P는 '산출'을 의미하는 'productions'의 약자인데, $P \subseteq (V-\Sigma) \times V^*$ 즉 유한수의 구구조 규칙들이다.[10] P는 A ->> γ 와 같은 형식으로 되어 있다.

일례로, "사람 온다"를 구성하는 어휘들만을 단말 기호열로 갖는 언어체계를 가정하면, 이 언어의 어휘들(어휘체) V와 단말 기호 Σ는 다음과 같은 집합으로 명시할 수 있다.

V = { IP, I', I, 사람, NP, VP, 오 , ㄴ다 }
Σ = { 사람, 오 , ㄴ다 }

$=>_G$를 다음과 같이 정의해 보자.

$$\Gamma A \Gamma' \ =>_G \ \Gamma \gamma \Gamma' \ \text{ iff } \ A \text{ ->> } \gamma$$

$=>_G$의 재귀적, 이행적인 닫힘(reflexive, transitive closure)을 $=>_G^*$와 같이 나타내면, $=>_G^*$는 0번을 포함하여 1번 이상의 조처(규칙 적용)를 통해서 Γ로부터 Γ'이 도출되는 되는 경우에 $\Gamma \ =>_G^* \ \Gamma'$이 성립함을 뜻한다.

언어 L은 어떤 문법 G를 통하여 얻어지는 단말 기호열인 문장들의 집합이라고 할 수 있다. yield를 다음과 같이 정의하면,

$$\text{yield}(G, A) = \{\omega \in TS^* \mid A =>_G^* \omega\}$$

---

9. 이를 논저에 따라 '최종 연결체' 또는 '종단 기호열' 등으로 옮기기도 한다.
10. 어떤 기호에 위첨자로 '*'를 덧붙이면 이 기호가 0개 또는 1개 이상 실현됨을 뜻한다. '+'가 덧붙은 것은 1개 이상 실현됨을 나타낸다. AB는 A와 B의 선형결합(線型結合: concatenation)을 뜻한다.

문법 G에 의해 정의되는 언어 L(G)는 다음과 같다.

L(G) = yield(G, S)

여기서 S는 '시초(start)' 기호 또는 '문장(sentence)' 기호이다.

구문분석의 문제를 일반화해서 말하면, 어떤 단말 기호열 Δ가 A라는 범주를 갖는지를 확인하고, 확인이 될 경우 그 통사구조를 제시하는 일이라고 할 수 있다. 어떤 단말 기호열이 A 범주(보통 문장 S)를 갖는다는 것을 확인하려면 $\Delta' = \Delta$의 관계를 만족하는 Δ'를 A로부터 도출해 내면 된다. 이런 과정을 연역적인 규칙들로써 형식화할 수 있다.

용어와 표기상의 약속을 몇 가지 정해 놓기로 한다. $\Gamma \Rightarrow \Delta$를 연속단계(sequent)라고 한다. Γ로부터 몇 번(0번도 포함)의 조처(단계)를 거쳐서 Δ가 얻어짐을 보이는 것이 구문분석의 문제이다. 또, 추정되는 범주를 표시하기 위해서는 범주 위에 윗줄을 긋는다. 그러면, $\overline{wA}B \Rightarrow \Delta$는 "만약 남아 있는 입력이 Δ이고 w와 A가 발견된다면 B는 완성된다."는 뜻이 된다.

문법 G를 바탕으로 한 연역 체계 N_G를 다음과 같이 정의할 수 있다.[11]

$\Rightarrow \Delta$ [입력 공식] : $\Delta \in \Sigma^*$인 모든 Δ에 대해서

$\Rightarrow$ A ->> γ [문법 공식] :
(A ->> γ) $\in$ P인 모든 A ->> γ에 대해서

$$\frac{\Rightarrow \Delta \qquad \Rightarrow A \multimap \gamma}{\overline{\gamma A} \Rightarrow \Delta} \text{ [시초 규칙]}$$

$$\frac{\Gamma \overline{A} \Gamma' \Rightarrow \Delta \qquad \Rightarrow A \multimap \gamma}{\Gamma \overline{\gamma} \Gamma' \Rightarrow \Delta} \text{ [확대 규칙]}$$

이 연역 체계를 바탕으로 Δ가 범주 A를 갖는다는 것을 보이기 위해서는 $\Rightarrow \Delta$로부터 $\overline{\gamma A} \Rightarrow \Delta$를 연역하면 된다.

이상은 도출을 통한 단말 기호열의 인식 과정을 보인 것이다. 이를 프롤로그로 구현

11. 다음의 '시초 규칙'과 '확대 규칙'처럼 삼단논법의 형식으로 표현되는 연속단계 계산(sequent calculus)의 정의 방법은 UCLA 언어학과 교수 E. Stabler의 1994년 제3, 4학기 강의 및 이로부터 발전된 초고인 E. Stabler(1996)을 따른 것이다.

하여 다음과 같이 간단한 프로그램을 얻을 수 있다.[12]

```
< 프로그램 n.pl >
:- op(100, xfy, ->>).
:- op(120, xfy, =>).[13]
recognize(L) :- proof([] => L).
proof(Sequent) :- goal_sequent(Sequent).
proof(Sequent) :- infer(Sequent, Sequent1), proof(Sequent1).
goal_sequent(DeltaA => Delta) :- predict(Delta, [_], DeltaA).[14]
% 문법
ip ->> [np, I1].     i1 ->> [vp, i0].
i0 ->> [nta].        np ->> [salam].
vp ->> [o].
% [시초 규칙]
infer([] => Delta, IAlphaA => Delta) :-
        A ->> Alpha, predict(Alpha, [A], IAlphaA).
% [확대 규칙]
infer(GammaIAEta => Delta, GammaIAlphaEta => Delta)  :-
        append(Gamma, [i(A)|Eta], GammaIAEta), A ->> Alpha,
        predict(Alpha, Eta, IAlphaEta),
        append(Gamma, IAlphaEta, GammaIAlphaEta).
predict([], L, L).  predict([E|L1], L2, [i(E)|L3]) :- predict(L1, L2, L3).
append([], L, L).  append([E|L1], L2, [E|L3]) :- append(L1, L2, L3).
```

이 프로그램을 이용하여 'recognize([salam, o, nta]).'와 같은 질의(query)를 입력하면 프롤로그로부터 'yes'라는 대답이 나오는데,

```
?- recognize([salam, o, nta]).
?- yes.
```

12. 다음에 제시하는 프로그램 n.pl, tdp.pl, bup.pl, lcp.pl, 그리고 <부록>의 lcpl.pl은 E. Stabler의 것을 필자가 우리말 예를 위하여 약간 수정한 것이다. 프로그램에서 우리말을 로마자로 표기할 경우 예일 로마자화(Yale Romanization) 방식을 따르기로 한다. '%' 표시는 뒤에 이어지는 설명(comment)이 프로그램의 본 내용에 포함되지 않도록 구별하는 프롤로그의 기호이다.

13. 이상 두 개의 규칙은 '연산자 선언(operator declaration)'으로서, 프롤로그에 내장되지(built-in) 않은 연산자(operator)들이 갖는 결합 계층상의 지위와, 이들이 놓이는 위치(앞, 가운데, 뒤), 그리고 그 형식을 선언하는 의미를 갖는다. 뒤에 제시하는 프로그램들에서는 이 부분을 생략한다.

14. [_]은 익명의 변수로 되어 있는 목록(list)이다.

이는 [salam, o, nta], 즉 "사람오ㄴ다."라는 기호열이 연역 체계 N_G를 가진 언어의 문법적인 문장으로 인식되었다는 뜻이다.

프로그램 n.pl은 인식기일 뿐이며, 아직 구문분석기는 아니다. 구문분석기는 기호열을 인식할 뿐만 아니라 인식한 결과를 통사구조로 제시할 줄 알아야 한다. 통사구조를 얻기 위해서는 proof 술어 및 goal_sequent 술어에 논항을 한 개 더 두어 인식된 통사구조를 넘겨 주도록 손질하면 된다.

이 전에 더 고려할 것이 있다. n.pl은 입력된 기호열의 구조에 대해서 추정을 하되, 기호열을 이루는 문장의 기호 모두를 생성해 낸 다음 입력의 문장 기호열과 합치하는지를 판정하는 비효율적인 방식으로 되어 있다. 그래서 표준적인 하향 구문분석(top-down parsing)의 전략에서는 맨 왼쪽의 범주만이 다시쓰여지고, 맨 왼쪽에 쓰여진 단말 기호가 입력의 첫번째 기호와 합치할 경우에 한해서 둘 다가 제거되게 된다.

표준적인 하향 인식(top-down recognition)의 연속단계 계산은 다음과 같이 표시할 수 있다. 이는 문법 G를 포함한 하향 인식 체계 TD_G를 정의한 것이다.

$\Rightarrow \Delta$ [입력 공식] : $\Delta \in \Sigma^*$인 모든 Δ에 대해서

$\Rightarrow A \rightarrow\!\!> \gamma$ [문법 공식]

: $(A \rightarrow\!\!\!\!\!\!\rightarrow \gamma) \in P$인 모든 $A \rightarrow\!\!\!\!\!\!\rightarrow \gamma$에 대해서

$$\frac{\Rightarrow \Delta \qquad \Rightarrow A \rightarrow\!\!\!\!\!\!\rightarrow \gamma}{\gamma A \Rightarrow \Delta}$$ [시초 규칙]

$$\frac{A\Gamma \Rightarrow \Delta \qquad \Rightarrow A \rightarrow\!\!\!\!\!\!\rightarrow \gamma}{\gamma \Gamma \Rightarrow \Delta}$$ [추정 규칙]

$$\frac{w\Gamma \Rightarrow w\Delta}{\Gamma \Rightarrow \Delta}$$ [제거 규칙]

이러한 하향 인식의 체계를 하향 구문분석기로 구현하면 다음 tdp.pl과 같다. 앞서의 n.pl보다는 다소 많은 수의 문법 규칙들이 이용된다.

```
<프로그램 tdp.pl >
parse(L, SS) :- proof([] => L, SS).
proof(Sequent, SS) :- goal_sequent(Sequent, SS).
proof(Sequent, SS) :- infer(Sequent, Sequent1), proof(Sequent1, SS).
goal_sequent([A] => [], SS) :- arg(1, A, SS).15)
% 문법 : n.pl에 비해서 9개의 규칙이 추가됨.
```

```
ip(ip/[DP, I1]) ->> [dp(DP), i1(I1)].  i1(i1/[VP, I0]) ->> [vp(VP), i0(I0)].
i0(i0/[-nta]) ->> [nta].               dp(dp/[D1]) ->> [d1(D1)].
d1(d1/[NP, D0]) ->> [np(NP), d0(D0)].  d0(d0/[-to]) ->> [to].
np(np/[N1]) ->> [n1(N1)].              n1(n1/[N0]) ->> [n0(N0)].
n0(n0/[-salam]) ->> [salam].           n0(n0/[-chelswu]) ->> [chelswu].
vp(vp/[V1]) ->> [v1(V1)].              v1(v1/[V0]) ->> [v0(V0)].
v0(v0/[-o]) ->> [o].
% [시초 규칙]
infer([] => Delta, IAlphaA => Delta) :-
      A ->> Alpha, predict(Alpha, [A], IAlphaA).
% [제거 규칙]
infer([i(W)|Gamma] => [W|Delta], Gamma => Delta).
% [추정 규칙]
infer([i(A)|Gamma] => Delta, IAlphaGamma => Delta) :-
      A ->> Alpha, predict(Alpha, Gamma, IAlphaGamma).
predict([],L,L).  predict([E|L1], L2, [i(E)|L3]) :- predict(L1, L2, L3).
```

이 프로그램을 이용하여 "철수도 온다."와 같은 문장의 문법성 여부와 그 통사구조를 얻고자 하면 'parse([chelwu, to, o, nta], SS).'와 같은 질의를 입력하면 된다. 그러면 다음과 같은 응답이 프롤로그로부터 나온다.

```
?- SS = ip/[dp/[d1/[np/[-chelswu], d0/[-to]]], i1/[vp/[v1/[v0/[-o]]], i0/[-nta]]].
   yes.
```

보통의 국어 문장은 왼쪽으로 가지를 쳐 가는 구조가 기본적이라 할 수 있다. 이상의 구문분석기를 통하여 설명되는 국어 문장의 예는 매우 간단하여 이 점이 잘 드러나지 않는다. 하지만, 위 예에서만 보더라도, i0인 'nta(-ㄴ다)'의 왼쪽에 'vp(오-)'가 놓이는데, 이 vp는 다시 왼쪽으로 명사구나 보문을 취할 수 있다. 하향 구문분석의 방식은 이렇게 왼쪽으로 가지쳐 가는 구조를 분석하기에 효율적이지 못한 것으로 알려져 있다. 하향 구문분석과는 정반대의 방식으로 상향(bottom-up) 구문분석의 방법이 있는데, 여기서는 그러한 문제가 해소된다.

상향 인식의 체계 BU_G는 다음과 같이 정의된다. γ^r은 어떤 기호열 γ를 이루는 기호들의 순서를 거꾸로 한 기호열이다. 가령 $[dp, i1]^r$은 [i1, dp]와 같다. $[dp]^r$은 그대로

15. 이는 A의 첫번째 논항을 취하여 SS로 건네준다는 뜻이다. arg는 프롤로그에 내장된 술어(built-in predicate)이다.

[dp]이다.

=> Δ [입력 공식] : $\Delta \in \Sigma^*$인 모든 Δ에 대해서
=> A ->> γ [문법 공식]
: (A -» γ) ∈ P인 모든 A -» γ에 대해서

$$\frac{\gamma\Gamma \Rightarrow \Delta \qquad \Rightarrow A \multimap \gamma^r}{A\Gamma \Rightarrow \Delta}$$ [환원 규칙]

$$\frac{\Gamma \Rightarrow w\Delta}{w\Gamma \Rightarrow \Delta}$$ [옮기기 규칙]

우리말처럼 왼쪽으로 가지를 쳐 가는 구조가 문제를 일으키는 이유는 하향 구문분석기가 입력된 자료(기호열) 없이도 맨 왼쪽의 구조에 대해서 계속 추정을 해 갈 수 있기 때문이다. 계속적으로 점차 더 복잡한 구조를 추정하여 만들어내게 되므로 무한 루프(loop)로 빠져들 수 있는 것이다. 상향 구문분석기는 추정되는 구조를 만들어 가기 전에 입력의 내용을 점검하도록 짜여져 있으므로 하향 구문분석기의 이와 같은 문제를 해소할 수 있다.

위 상향 인식 체계를 구문분석기로 구현하면 다음과 같다. 이를 bu.pl이라고 부르기로 한다. 앞의 하향 구문분석기에서와 마찬가지로 통사구조를 얻기 위한 장치가 첨가되어야 한다.

```
< 프로그램 bup.pl >
parse(L, SS) :- proof([] => L, SS).
proof(Sequent, SS) :- goal_sequent(Sequent, SS).
proof(Sequent, SS) :- infer(Sequent, Sequent1), proof(Sequent1, SS).
goal_sequent([A] => [], SS) :- arg(1, A, SS).
% [옮기기 규칙]
infer(Gamma => [W|Delta], [W|Gamma] => Delta).
% [환원 규칙]
infer(Gamma => Delta, [A|GammaRemainder] => Delta)  :-
        A ->> Alpha, reverse(Alpha, [], AlphaR),
        append(AlphaR, GammaRemainder, Gamma).
reverse([H|L0], L1, L2) :- reverse(L0, [H|L1], L2).   reverse([], L, L).
append([], L, L).   append([H|L0], L1, [H|L2]) :- append(L0, L1, L2).
% 문법
ip(ip/[DP, I1]) ->> [dp(DP), i1(I1)].        i1(i1/[VP, I0]) ->> [vp(VP), i0(I0)].
```

```
i0(i0/[-nta]) ->> [nta].                dp(dp/[D1]) ->> [d1(D1)].
d1(d1/[NP, D0]) ->> [np(NP), d0(D0)].   d0(d0/[-to]) ->> [to].
np(np/[N1]) ->> [n1(N1)].               n1(n1/[N0]) ->> [n0(N0)].
n0(n0/[-salam]) ->> [salam].            n0(n0/[-chelswu]) ->> [chelswu].
vp(vp/[V1]) ->> [v1(V1)].  v1(v1/[V0]) ->> [v0(V0)].  v0(v0/[-o]) ->> [o].
```

이 상향 구문분석기는 왼쪽으로 가지쳐 가는 구조(좌분지 구조: left-branching structure)에 적합하다. 그러나 반대로 오른쪽으로 가지쳐 가는 구조에서는 비효율적이다. 우리말이 기본적으로 좌분지의 특징을 가진다고 할 수는 있지만 조금만 복잡한 구문에서도 비전형적인 경우들이 혼하게 발견된다.

상향과 하향의 구문분석기 모두가 일면적으로 장점 및 결함을 가지고 있어, 결함을 피하고 장점만을 살리고자 하는 노력이 일어나게 된 것은 자연스러운 일이다. 그 결과로 고안된 것이 좌변 구문분석(left-corner parsing)의 기법이다.

이 방법의 특징은 우선 구성성분의 '좌변(left-corner)'이 중요성을 갖는다는 데에서 찾아볼 수 있다. 한 구성성분의 좌변이란 그 구성성분의 맨 왼쪽 딸성분(daughter)이다. 따라서 어떤 산출(production) A ->> γ의 좌변은 γ의 왼편에서 첫번째에 있는 범주이다. 하향 구문분석 방법은 산출 A ->> γ에서 γ를 전혀 고려하지 않은 상태로 이 산출을 이용한다. 상향 구문분석은 모든 γ가 다 제시된 후에 이 산출을 사용한다. 이에 비해서, 좌변 구문분석 방법은 γ의 첫번째 요소—이것이 A의 좌변이다—가 제시된 다음에 이 산출을 사용한다. 따라서 이 방법은 상향 구문분석이 좌변에서 이루어지는 것으로 생각할 수 있다. 그러나 좌변에서 상향의 구문분석이 이루어진 다음 하향 구문분석의 방식으로 자매성분(sister)들을 예측하는 작업이 수행된다.

이러한 방법의 인식 체계, 즉 좌변 인식 체계를 LC_G라고 부르자. LC_G는 다음과 같다.

=> Δ [입력 공식] : $\Delta \in \Sigma^*$인 모든 Δ에 대해서

=> A ->> γ [문법 공식]

: (A -» γ) ∈ P인 모든 A -» γ에 대해서

$$\frac{\gamma\,\Gamma \Rightarrow \Delta \qquad \Rightarrow A \rightarrow\!\!\!\!\rightarrow \gamma\,\eta}{\eta\,A\Gamma \Rightarrow \Delta}$$ [좌변 규칙] 단, $\gamma \in V$이거나 $\gamma\,\eta = \lambda$

$$\frac{\gamma\,\overline{A}\Gamma \Rightarrow \Delta \qquad \Rightarrow A \rightarrow\!\!\!\!\rightarrow \gamma\,\eta}{\eta\,\Gamma \Rightarrow \Delta}$$ [좌변/제거 규칙]

단, $\gamma \in V$이거나 $\gamma\,\eta = \lambda$

$$\frac{\Gamma \Rightarrow w\Delta}{w\Gamma \Rightarrow \Delta}$$ [옮기기 규칙]

$$\frac{\overline{w}\Gamma \Rightarrow w\Delta}{\Gamma \Rightarrow \Delta}$$ [옮기기/제거 규칙]

이를 구현한 것이 다음과 같은 좌변 구문분석기이다. 이를 lcp.pl이라 이름한다.

```
< 프로그램 lcp.pl >
parse(L, SS) :- proof([] => L, SS).
proof(Sequent, SS) :- goal_sequent(Sequent, SS).
proof(Sequent, SS) :- infer(Sequent, Sequent1), proof(Sequent1, SS).
goal_sequent([A] => [], SS) :- arg(1, A, SS).
% [좌변/제거 규칙]
infer([B, i(A)|Gamma] => Delta, IBetaGamma => Delta) :-
        A ->> [B|Beta], predict(Beta, Gamma, IBetaGamma).
predict([], L, L).  predict([E|L1], L2, [i(E)|L3]) :- predict(L1, L2, L3).
% [공좌변/제거 규칙]
infer([i(A)|Gamma] => Delta, Gamma => Delta) :-
        A ->> [].
% [좌변 규칙]
infer([B|Gamma] => Delta, IBetaAGamma => Delta) :-
        A ->> [B|Beta], predict(Beta, [A|Gamma], IBetaAGamma).
% [공좌변 규칙]
infer(Gamma => Delta, [A|Gamma] => Delta) :-
        A ->> [], emptycheck(Gamma).
emptycheck([i(i1(_))|_]).          % 공범주 여부 확인
% [옮기기/제거 규칙]
infer([i(W)|Gamma] => [W|Delta], Gamma => Delta).
% [옮기기 규칙]
infer(Gamma => [W|Delta], [W|Gamma] => Delta).
% 문법[16]
cp(cp/[C1]) ->> [c1(C1)].          c1(c1/[IP, C0]) ->> [ip(IP), c0(C0)].
c0(c0/[-n]) ->> [n].[17]           c0(c0/[-nta]) ->> [nta].
ip(ip/[DP, I1]) ->> [dp(DP), i1(I1)]. i1(i1/[VP, I0]) ->> [vp(VP), i0(I0)].
```

16. 이 구문분석기는 "성미도 급한 사람은 온다."와 같은 복합문도 처리할 수 있게 되어 있다.
17. "일찍 온 사람"과 같은 경우의 어미 '-ㄴ'을 C 범주로 본다. 이 연구에서는 다음에서 보는 것처럼 종결어미 '-ㄴ다'도 C 범주로 처리하였다.

```
i0(i0/[]) ->> [].[18]                    dp(dp/[D1]) ->> [d1(D1)].
d1(d1/[NP, D0]) ->> [np(NP), d0(D0)]. d0(d0/[-to]) ->> [to].
np(np/[N1]) ->> [n1(N1)].                n1(n1/[N0]) ->> [n0(N0)].
n0(n0/[-salam]) ->> [salam].             n0(n0/[-sengmi]) ->> [sengmi].
vp(vp/[V1]) ->> [v1(V1)].                v1(v1/[V0]) ->> [v0(V0)].
v0(v0/[-o]) ->> [o].                     v0(v0/[-kupha]) ->> [kupha].
```

좌변 구문분석에서는 프로그램에 의하여 검사될 가능한 구성성분의 좌변을 미리 검사하여 예거(instantiation)해 놓음으로써 실행 효율을 크게 향상시킬 수 있다.[19]

3. '와/과' 문장의 어휘주의적 분석

이 글에서 '어휘주의'는 널리 '변형론' 또는 '변형주의'와 대립되는 개념으로 쓴다. '와/과' 문장에 대한 종래의 통사론적 처리는 변형론적 입장이 지배적인 것이었다. 이러한 변형론적 입장도 그 처리 방법을 자세히 살펴 보면 두 가지로 더 나눌 수 있다. 서두에서 들었던 다음 문장들을 다시금 검토해 보자.

(1) 가. 철수와 영호는 싸웠다.
　　나. 철수는 영호와 싸웠다.

(1나) 문장은 (1가)와 대응되는 기저구조로부터 'NP와' 성분이 뒤로 이동하여 얻어지는 것이라고 하는 설명 방법이 있다. 이를 접속항 이동변형설이라고 부르기로 한다.[20] 이는 (1)의 대칭구문뿐만 아니라, 다음 (2)와 같은 상호구문, (3)과 같은 동반구문, 나아가 (4)와 같은 경쟁구문들이 일관적으로 'NP와 NP' 구조와 'NP이 NP와' 구조를 갖는 점을 규칙으로 포착하는 의의가 있다.

---

18. 이는 공범주(empty category)로서의 i0 범주를 설정한 것이다.
19. instantiation을 '예거(例擧)'로 옮기기로 한다. 변수가 사례(instance)나 사례들의 집합으로 확인되어 구체화되는 것을 의미한다. <부록>으로 제시하는 구문분석기, 즉 lcpl.pl에서는 생성된 link와 연결하는 두 자리 술어 link를 두었는데, 이것이 예거된 link(v1(_),np(_)) 등과 연결되게 되어 있다. 가능한 구성성분의 좌변을 추정하여 생성해 내는 프로그램(xgenlink.pl)이 따로 필요하다.
20. 이러한 설명법은 김완진(1970), 김영희(1974)를 비롯하여 대부분의 관련 논의에서 가정하여 왔다.

(2) 가. 철수와 영호는 서로 공격했다.
나. 철수는 영호와 서로 공격했다.
(3) 가. 철수와 영호는 기차를 탔다.
나. 철수는 영호와 기차를 탔다.
(4) 가. 철수와 영호는 서로 먼저 선물을 타려고 했다.
나. 철수는 영호와 서로 먼저 선물을 타려고 했다.

한편, (1)-(4)의 (가) 문장에 대해서, 기저에서 두 문장의 접속으로 설정한 다음, 그리로부터 선후행 문장 간에 동일 성분이 생략되는 과정을 거쳐 표면의 'NP와 NP' 구조가 생겨난 것이라고 설명하는 방법이 제시되기도 하였다. 이를 접속문 축약변형설이라고 부를 수 있다.[21]

이러한 두 가지 변형적 처리법이 가지는 문제점에 대해서는 양정석(1996가, 다)에서 상세하게 보인 바 있다. 여기서는 이러한 논의의 결과로서 'NP와 NP'가 갖는 내부 통사구조, 이러한 형식을 갖는 문장들 및 'NP이 NP와' 형식을 갖는 문장들이 보이는 통사·의미적 행태와 그 통사구조를 차례로 제시하면서 부분적으로 변형론적 처리법의 문제를 지적하고자 한다.

3.1. 'NP$_1$와 NP$_2$'의 통사구조

명사구 접속, 즉 'NP$_1$와 NP$_2$'의 구조를 기술하는 방법으로 크게 두 가지를 생각할 수 있다. 전통적인 가정에서는 접속항이 몇 개이든 모두 동등한 통사적 지위를 가진다고 본다.[22] 명사구 접속 구성을 이와 같이 볼 경우, 구구조의 이론에서 이를 일반적인 구성과는 별도로 처리해 줄 수밖에 없다.

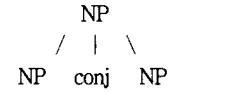
다른 가능성은 접속항 사이에 계층성의 차이가 있다고 보는 것인데, 이러한 가정은 접속조사 '와'를 머리성분으로 보아 핵계층 이론을 확장적으로 적용할 수 있다는 장점을

21. 김완진(1970)에서는 주로 (3)과 같은 구문에 한해서 접속문 축약변형설을 받아들이고 있다. 김영희(1974)에서는 대칭동사 구문의 일부('무대칭서술어'와 '비대칭서술어'의 구문)에 대해서 접속문 축약변형설을 통하여 설명하고 있다. 최재희(1985)에서는 대칭동사 구문 모두에 대해서 접속문 축약설을 확대하고 있으나, 반면에 접속항 이동변형설은 받아들이지 않음으로써 김완진(1970), 김영희(1974)에서 두 설명법을 함께 이용하는 것과 대조된다. 성광수(1979)에서도 비슷한 견해를 보이고 있다.

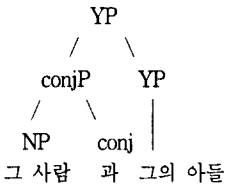
22. '병렬구조(parallel structure)'로 보는 방법은 최근 새로이 시도되고 있는 또 한 가지 관점이라고 할 수 있으나, 표준적인 구구조의 이론과는 다른 구구조의 개념과 함께 별도의 새로운 원리를 도입해야 하는 부담을 지니고 있다고 판단된다. Goodall(1987), Moltmann(1992) 등에서 이러한 방법이 시도되고 있다.

지닌다. 이러한 가능성도 두 가지로 더 갈라 볼 수 있다. 접속항 중의 하나는 다른 것에 부가되는 것으로 볼 가능성이 한 가지이고,[23] 명사구 접속 구조 전체가 XP구조를 이루면서 한 접속항은 X'구조, 나머지 접속항은 이에 자매항이 되는 명시어(specifier)가 되는 것으로 볼 가능성이 다른 한 가지이다.[24] 이 세 가지 가능성을 다음에 표시한다.

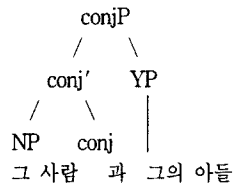
(5)



(6) 가.



나.



이 글에서 주장하는 바는 맨 마지막의 가능성, 즉 (6나)이다. 단, '와/과'의 통사 범주를 'conj' 대신 'P'로 보는 것이다. 이는 일단 명사구 접속의 '와/과'와 그밖의 경우에 나타나는 '와/과'를 하나로 처리할 수 있는 이점이 있다.

'와/과'의 통사범주를 P 교점에 설치하는 것은 '와/과'가 국어의 다른 격조사들 '에', '에게', '로', 그리고 복합조사인 '에서', '에게서', '로부터', 일부의 '부터', '까지' 등과 함께 후치사라는 독립된 범주를 이룬다고 보기 때문이다. 가령, (7가)에서 '현재의 상태로'는 '로'의 함수적 성격에 따라 '그 여자를'에 대한 서술어로 기능할 수 있다. '로'가 실현되지 않은 (7나)가 불가능한 것은 '로'의 이러한 기능에 대한 증거이다.[25]

---

23. Munn(1993)에서 이러한 처리법을 실행하고 있다. Munn(1993: 12)에 의하면 이에 앞서 1986년과 1987년의 그의 논문에서 영어의 선행 접속항을 명시어로 하고 후행 접속항을 보어로 하는 핵계층 이론적 접근을 처음으로 시도했다고 한다.그는 영어의 'NP and NP' 구성에서 'and'가 머리성분으로서 뒤의 NP를 보어로 취하여 BP(Boolean phrase)를 이루고, 이것이 다시 앞의 명사구 NP에 부가되는 것으로 설정한다.

24. 이에도 또 다른 하위 처리법들이 더 고안될 수 있다.

25. 여기서는 Rothstein(1983)에서의 '통사적 함수(syntactic function)', 즉 '논항(argument)'과 상대되는 개념으로 함수란 용어를 쓴다. 함수는 논항으로 채워져야(포화되어야) 하는데, (7나)에서 '현재의 상태'는 함수일 수 없으면서 어느 논항에 의해서도 취해질 수 없으므로 비문이 된 것이다.

(7) 가. 나는 그 여자를 현재의 상태로 사랑한다.
나. *나는 그 여자를 현재의 상태 사랑한다.

동반구문에서는 다른 부사 요소 없이 'NP와'가 주어나 목적어에 대한 서술어로 기능한다. 이 때 '와'는 생략할 수 없다. 이 사실은 '와'를 후치사로 볼 때 무리 없이 설명할 수 있다.

(8) 가. 철수가 영희와 떠났다.
나. 아버지는 철수를 영희와 떠나보냈다.
(8)' 가. *철수가 영희 떠났다.
나. *아버지는 철수를 영희 떠나보냈다.

'철수와 영호'나 '철수와 영호와 순기'를 후치사구로 보게 되면 국어에서 핵계층 도식을 일반화할 수 있게 된다. 핵계층 이론의 기본 동기는 머리성분과 최대 투사 사이에 동심성이 유지되는 점을 포착할 수 있다는 것이다. 모든 구는 하나의 머리성분을 가지고 있고, 이 머리성분의 문법적 특질이 상위의 관할 교점으로 삼투되어 최대투사의 문법적 성격을 결정한다.[26] 이 점은 뒤의 구문분석기에서 어휘부의 '와/과'의 특질 conj가 PP로 전해지는 것으로 구현되게 된다.

명사구 접속의 구성은 교호적 구문에서 복수성 명사구들과 동일한 특징을 가지면서 동사와 선택 관계에 놓인다. 'NP_1와 NP_2'가 문장의 주어나 목적어 위치에 나타날 때, 동사의 선택적 특질은 'NP_2'에 부과되지 않는다. 이는 표면적으로 유사한 국어의 다른 구성인 'NP_1의 NP_2' 구성과는 같지 않은 점이다.

(7) 가. 철수와 영호가 싸웠다.
나. *영호가 싸웠다.(교호적 의미로)
(8) 가. 철수의 친구가 싸운다.
나. 친구가 싸운다.
(9) 철수와 영희가 걷는다.

(8)에서 'NP_1의 NP_2' 구성의 통사적, 의미적 중심이 되는 머리성분은 '친구'일 수밖에 없으나, (7)의 'NP_1와 NP_2' 구성에서는 그렇게 보기 어렵다. (9)에서 '걷다'는 (7가)의 '싸우다'와 같은 선택제약이 없다. 선택제약은 원래 어휘개별적인 성격을 갖는 개념이

26. 이 점이 4.1.절의 문법에서는 Phi 또는 F, G, H에 주어지는 특질(ec, moved, rstr, com, reci 등)이 중간 투사범주 X'와 최대투사 범주 X''로 전해지는 것으로 구현되게 된다.

다. '싸우1'과 '싸우2, 걷'의 선택제약의 차이는 어휘부에서 이들 어휘 하나하나에 대하여 기술할 수밖에 없다.

'아이들'이나 '여러명의 불량배, 군중' 같은 집합적 명사(구)들도 '싸우다'와 같은 동사의 주어로 나타난다.

(10) 가. 아이들이 싸운다. 나. 여러 명의 불량배가 싸운다.
다. 군중이 싸운다.
(11) ?철수, 영희 싸웠다.
(12) 가. ?*철수, 그리고 영희가 싸웠다.
나. ?*철수 및 영희가 싸웠다.

(7가)를 이들과 비교해 보면 '와'가 복수성을 만들어 낸다고 볼 수밖에 없다. (11)에서도 '와' 없이 복수성을 표현하고 이에 따라 '싸우다'와의 선택 관계가 성립하는 것을 어느 정도 인정할 수는 있다. 이 경우 '와'의 생략으로 해석할 소지가 있기 때문이다. (12가, 나)에서는 그것이 불가능하다는 것은 '그리고, 및'과는 달리 '와/과'가 복수성을 결정하는 어휘적 요인임을 말해 주는 증거이다.[27]

이러한 점을 기술해 주기 위해서는 '와/과'를 머리성분으로 보고, 이 머리성분의 특질이 삼투되어 구의 성격을 결정한다고 보는 것이 필요하다. 따라서 '명사구 접속' 구성이 'P'인 머리성분 '와/과'에 의하여 투사되어 'PP'를 이룬다고 기술하는 것은 이상의 선택제약의 사실을 포착하는 타당한 방법이 된다.

'와/과' 구성을 핵계층 구조로 기술하는 근원적인 동기 중 하나는 의미구조와의 대응을 간편하게 한다는 점이다. '와'를 머리성분으로 설정하고, 두 개의 접속항을 '와'가 취하여 구 전체가 새로운 의미적 성격을 갖도록 한다는 구상이 핵계층 구조로서 간결하게 구현될 수 있기 때문이다.

다음과 같이 의미구조로의 대응을 형식화해 볼 수 있다. '사랑하다'와 같은 동사가 주어와 목적어를 두 논항으로 취하여 (13가)와 같은 의미구조를 이루어간다고 할 때, 명사구 접속의 구성에서 논항들의 어순은 이와 반대가 된다. 이는 유형(type)에 대한 표시를 생략했을 뿐, 일반 형식의미론에서의 표기와 다름이 없다. 뒤의 의미분석기에서도 이러한 형식으로 기호화(encode)한다.

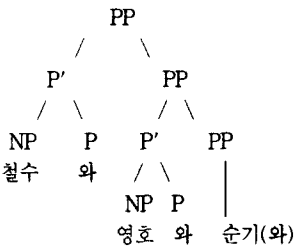
27. (11)의 문법성을 인정할 경우에는 '와/과'의 자리에 공범주가 실현된 것으로 볼 수 있다.

(13) 가. [철수가 [영희를 사랑하]v']vp 니다
=> λXλY[love(Y, X)] => λY[love(Y, 영희)]
=> [love(철수 , 영희)]
나. [[[철수]와]p' [[영희]∅]pp]pp
=> λXλY[and(X, Y)]=> λY[and(철수 , Y)]
=> [and(철수 , 영희)]

첫번째의 통사구조로부터, 두번째의 동사나 '와/과'의 어휘의미 표상을 이용하여 논항의 대치가 행해진 다음 세번째의 의미구조가 이루어지는 과정을 간단히 볼 수 있다. 어순은 다르지만 두 경우 모두에서 먼저 보어 성분의 의미가 대치되고 다음으로 명시어 성분이 대치되는 것으로 설명할 수 있다.[28] 뒤에서 제시할 의미분석기는 이 과정을 기계적으로 구현하고 있다.

(6나)와 같은 핵계층 구조는 접속항이 세 개 이상일 경우에도 확대·적용될 수 있다. '철수와 영호와 순기' 같은 형식은 다음과 같은 구조로 표시된다. (6나)의 conj는 P로 대체되었다.

(14)



앞서 (6나)의 경우에나, (14)의 경우에나 마지막 접속항은 후치사구(PP)로 기술되어, 국어의 명사구 접속의 형태를 잘 반영해 준 것이라고 생각된다.

3.2. 동반구문의 통사·의미적 특징

동일한 '와/과' 문장이라도 '서로', '함께', 또는 '각각'과 같은 부사들에 의하여 의미 해석이 달라진다.

---

28. 이를 형식의미론에서는 베타 환원(β-reduction)이라고 부른다. 이에 상응하는 절차를 이 연구에서는 '논항 융합'이라 부른다.

(1) 가. 철수와 영호가 서로 우리를 도왔다.
나. 철수와 영호가 함께 우리를 도왔다.
다. 철수와 영호가 각각 우리를 도왔다.

각 문장에서 'NP$_1$와 NP$_2$이'는 독립 성분으로 기능함이 분명하지만, 이들의 의미 해석에서 주어를 이루는 '철수', '영호'의 행위나 사건이 따로따로 혹은 합동으로 일어나는 것은 '각각'이나 '함께'에 의한 것이다.

'와/과' 문장에서 부사들은 스스로 어휘적 의미를 가지고 의미 해석에 있어 중요한 역할을 수행한다. 뒤에서 보일 것이지만, 상호구문에서는 '서로'가 교호성의 의미특질(연산자)를 가지고서 통사구조 자체를 변화시키는 역할을 한다. 그 경우의 'NP와'는 필수 성분으로 해석된다. 그러나 '함께'는 부사로서 'NP와'를 보어로 취하면서 그 자신의 구는 부가어로 기능하게 한다.

'함께' 또는 'NP와 함께'는 부사구로서 주어나 목적어에 대하여 서술어의 역할을 한다. 다음 예에서 '함께'는 각각 목적어 '철수와 영희'와 주어 '그들'의 서술어 노릇을 한다. 이는 일반적인 서술화 규칙이 적용되는 환경적 조건을 다 만족하는 것으로 보인다.[29]

(2) 가. 그들은 철수와 영희를 함께 떠나보냈다.
나. 그들은 함께 철수와 영희를 떠나보냈다.

또한, 다음과 같은 문장들에서 'NP와 함께'는 독립된 부사어 성분으로 각각 목적어 '철수'와 주어 '그들'의 서술어 역할을 한다.

(3) 가. 그들은 철수를 영희와 함께 떠나보냈다.
나. 그들은 영희와 함께 철수를 떠나보냈다.

'NP와 함께' 구성에서 'NP와'는 부사 '함께'의 보어가 되는 것으로 본다. 그런데 '동반구문'에서 '함께' 없이 'NP와'의 형식만으로도 부가어를 이루는 경우가 있다.

(4) 가. 김씨가 아들과 함께 달린다.
나. 김씨가 아들과 달린다.

이런 경우, '함께'가 부사로서 보어를 취할 수 있는 것처럼 '와/과'도 후치사(P)로서

29. Rothstein(1983)에서는 일차 서술어 및 부가어인 이차 서술어가 그 주어와 상호 성분통어한다고 하였다.

보어를 취할 수 있다고 하는 것이 자연스럽다.

흥미로운 예는 다음과 같은 것이다.

(5) 가. 김씨는 아들과 며느리에게 편지를 썼다.
나. *김씨는 아들에게 며느리와 편지를 썼다.

여기에는 두 가지 변형론적 설명 방식과 관련되는 난점이 다 존재한다. 특히 접속항 이동 변형의 방법을 택할 경우, (5나)와 같은 비문이 생기는 이유는 변형에 대한 제약으로 기술할 수 있을 것이다. 접속항 이동 변형 규칙은 'NP와 NP'와 같은 구조적 조건이 충족될 때 항상 적용되는 것으로 설정해야 할 것인데, 위 예문은 이 규칙의 적용이 불가능하다는 것을 보여주기 때문이다. 그러나 이러한 제약을 접속항 이동 변형의 제약으로 설정하기는 어렵다. 다음은 가능하지만,

(6) 가. 그는 철수와 영수를 떠나보냈다.
나. 그는 철수를 영수와 떠나보냈다.

(7나)처럼 비문이 되는 조건을 명확하게 한정하기 어렵다는 것은 접속항 이동변형설에 대해서 문제를 제기한다.[30]

(7) 가. 그들이 콩과 팥을 팔았다.
나. *그들이 콩을 팥과 팔았다.

부가어가 '서술화'에 의한 해석가능성을 가질 때 기본적으로 주어와 목적어에 한정하여 그 서술어 역할을 하는 것은 여러 언어에서 보편적인 현상이다. 위 예는 이와 같은 맥락에서 잘 설명된다. 즉 (5나)가 불가능한 이유는 '아들에게'에서 후치사(P)로 간주되는 '에게'에 의하여 서술화의 조건인 성분통어 관계가 성립될 수 없게 되기 때문이라고 설명할 수 있다. 이와 같은 경우, 먼저 'NP와'가 부가어로서 생성되고, 다음으로 서술화 규칙이 목적어와의 주술관계를 허가해 주게 된다고 본다.[31]

30. 남기심(1990)에서는 'NP와'의 이탈형이 가능한 것은 이 NP가 '행위자'일 경우라고 지적한 바 있다. 그러나 그 엄밀한 조건은 제시하기 어렵다고 말하고 있다. '행위자 조건' 같은 것을 이탈 변형에 대한 제약 조건으로 설정할 수 없다는 데에 변형설의 큰 문제가 있다.

31. 구문분석기에서 (3가)는 부가어 '영희와 함께'가 VP의 오른쪽에 부가되는 것으로 본다. (3나)의 '영희와 함께'가 부가되는 위치에 대해서는 잠정적으로 IP의 왼쪽에 부가되는 것으로 처리한다.

3.3. 자동사적 대칭구문의 통사·의미적 특징

필수 성분에 나타나는 격조사 '에', '로'의 의미는 동사의 어휘의미와 밀접하게 결합하여 상호작용하므로 서로 분리하기 어렵다. '와'도 같은 맥락에서 고려해 줄 수 있다. 대칭동사 '싸우다'와 특정 국면에서 공통되는 의미를 갖는 동사로 '닿다, 대다, 치다, 부딪치다, 만나다' 등을 들 수 있다.

(1) 가. 손이 천정에 닿았다.
　　나. 그는 손을 천정에 대었다.
(2) *철수가 책상에 쳤다./*철수가 책상과 쳤다./철수가 책상을 쳤다.

(1)에서의 '닿다'와 '대다'는 신체의 한 부분인 '대상(Theme)'이 '처소(Location)'로서의 어떤 사물에 접촉하게 되는 과정을 표현한다. '닿다'에 비해서 '대다'는 '행위자(Agent)'의 작용이 더 가해진다. 그 격조사 선택의 측면으로 보면, 두 동사 모두 '에/에게'를 취한다. (2)의 '치다'는, '와'는 물론이고 '에/에게' 도 용인하지 않는다. '치는' 행위의 대상이 되는 '책상'은 처소로 해석될 법한데도 '에'가 불가능한 것은 특이한 현상이라고 하겠다. '대다'의 경우는 '에/에게'를 필수적으로 요구하는 동사로 비교할 만하나, 반면 '와'는 취하지 않는다. '부딪치다'는 '에/에게'뿐만 아니라 '와'를 필수적으로 취하기도 하나, 목적어는 취하지 않는다. 이 동사는 '싸우다'와 의미적으로도 상당히 근접한다고 볼 수 있다.

(3) 가. 철수가 책상에 부딪쳤다./?책상이 철수에게 부딪쳤다.
　　나. 철수가 영호와 부딪쳤다./?철수가 책상과 부딪쳤다.
　　다. *철수가 책상을 부딪쳤다./*책상이 철수를 부딪쳤다.

'싸우다'는 필수 성분으로서 오로지 'NP와'만을 취한다. 이 점에서 '와'와 '에/에게'를 다 취하되 목적어는 취하지 않는 '부딪치다'와 차이를 보이며, 'NP와'와 목적어를 다 취하되 'NP에/에게'만은 취하지 않는 '만나다', '결혼하다'와도 구별된다. '헤어지다'는 특이하게 '에게서/로부터'와 '와'를 다 갖는 동사이다.

(4) *철수가 영호에게 싸웠다./철수가 영호와 싸웠다./철수가 영호를 싸웠다.
(5) *철수가 아내에게 만났다./철수가 아내와 만났다./철수가 아내를 만났다.
(6) 철수가 영희{에게서,로부터} 헤어졌다./철수가 영희와 헤어졌다.
　　/*철수가 영희를 헤어졌다.

'와'가 '에/에게'나 '에서/에게서/로부터', '를' 등의 다른 격조사와 교체하는 양상을 중심으로 하여, 동사들의 유형을 다음과 같이 나누어 볼 수 있다.[32]

(7) 치다(를-*) x< y > / *
'싸우다/결혼하다'류(*-와): * / x< y^COM >
'부딪치다/어울리다'류(에-와): x< y^LOC > / x< y^COM >
'헤어지다'류(에서/로부터-와): x< y^SOC > / x< y^COM >
'만나다'류(를-와): x< y > / x< y^COM >

필수 성분으로서 'NP와'를 취하는 구문은 항상 '서로'의 실현을 용인한다. 이 점은 이들 구문의 동사가 '교호성(reciprocity)'을 갖는다는 증거로 간주될 수 있다. 이에 따라, 대칭동사인 '싸우다'의 어휘의미구조는 '와'를 취하는 동사 구문의 교호적 의미구조를 반영하여 기술해 줄 수 있을 것이다.

'와/과' 문장의 두 가지 경우, 즉 'NP와 NP이' 구조와 'NP이 NP와' 구조가 의미에 있어서 동질적인 점을 포착하는 일이 문제로 떠오른다. 변형론적 연구에서, 접속문 축약 변형설은 의미구조와 형식적으로 일치하는 기저구조를 설정함으로써 이 점을 포착하려고 노력하였다. 접속항 이동변형설은, 변형에 의한 의미의 변화는 불가능한 것이므로, 접속항 'NP와'가 이동하기 전이나 후나 두 구조 사이에 의미의 동일성이 유지된다고 봄으로써 만족하였다. 그러나 이 연구에서는, 서로 다른 통사적 과정을 거치는 두 구문이 동일한 의미구조를 갖게 됨으로써 두 구조가 연관되는 것으로 설명한다.

'NP와 NP' 구조나 'NP이 NP와' 구조는 모두 기저에서 생성된다. 이들은 서로 다른 통사구조로 되어 있으나 그 의미구조의 형식에 있어서는 동질적인 구조로 표상된다. (8)과 같은 예문에서의 대칭동사 '싸우다'는 두 자리 서술어로 간주되는데, 그 어휘의미구조는 구체적으로 다음과 같다.

(8) 가. 철수가 영호와 싸웠다.
나. 싸우1:[[AFF/+caus(x,)],
[[CS(x , [GO/+cntc,+cols(x , [TO([AT(y)])])])] AND
[CS(y , [GO/+cntc,+cols(y , [TO([AT(x)])])])]]]

32. 오른편에 표시한 것은 논항들의 수와 통사구조상의 지위, 격조사의 형식를 표상하는 '어휘통사구조'이다. LOC는 동사가 격조사로 '에/에게/한테'를 취한다는 점을 어휘개별적 특징으로 명시해 놓은 것이며, SOC와 COM은 각각 '에서/에게서/으로부터'와 '와/하고'에 대한 선택적 정보를 표시한 것이다. 이에 대해서는 양정석(1997다)를 참고할 수 있다.

이로부터 논항 융합(argument fusion)이 이루어진 결과, 즉 전체 문장의 의미구조는 다음과 같다. 두 논항 변수에 대입(융합)되는 명사구의 의미를 각각 [A], [B]로 표시한다.

(9) "철수가 영호와 싸웠다."의 의미구조 :
[[AFF/+caus([A],)],
[[CS([A], [GO/+cntc,+cols([A]), [TO([AT([B])])])])] AND
[CS([B], [GO/+cntc,+cols([B]), [TO([AT([A])])])])]]]

다음 (10) 문장들에서의 '싸우다'는 한 자리 서술어로 간주하여 (8)의 '싸우다'와는 별도의 어휘항목으로 갈라 본다.[33]

(10) 가. 철수와 영호가 싸웠다. 나. 철수와 영호와 순기가 싸웠다.
다. 아이들이 싸웠다. 라. 두 사람은 싸웠다.
(11) 싸우2 : [[+reciprocal],
[AFF/+caus(x ,)],
[[CS(x , [GO/+cntc,+cols((x), [TO([AT([ ])])])])] AND
[CS([ ], [GO/+cntc,+cols(([ ]) , [TO([AT(x)])])])]]]

'싸우2'가 갖는 어휘의미구조를 기초로 해서 얻어질 수 있는 의미구조로는 크게 두 가지가 가능하다. 먼저, 주어 명사구가 복수성을 갖지 않는 경우 [+reciprocal] 연산자는 작동되지 않으므로 다음과 같은 의미구조가 해석된다.

(12) "철수가 싸웠다."의 의미구조 :
[[+reciprocal], [AFF/+caus([A],)] ,
[[CS([A], [GO/+cntc,+cols([A]), [TO([AT([ ])])])])] AND
[CS([ ], [GO/+cntc,+cols([ ]) , [TO([AT([A])])])])]]]

비교호적 해독은 '싸우2'의 x 논항에 단순히 주어 명사구의 의미가 대치(융합)되어 얻어진다. 그러나, 교호적 해독을 위해서는 논항 융합의 과정에서 특질 '[+reciprocal]'로 표시된 교호성 연산자가 적용된다. 교호성 연산자가 작용하면 다음과 같은 의미구조가 얻어진다.[34]

---

33. '[ ]'는 암시 논항(implicit argument)이다. 여기서는 한 어휘의미구조에 암시 논항이 둘 이상 나타날 때 이들은 모두 동일한 것으로 가정한다.
34. 주어 명사구가 복수성을 갖는 점이 [+i]로 표시된다. 양정석(1996가) 참조.

(13) "철수와 영호가 싸웠다."의 의미구조(교호적 해독):

[[AFF/+caus([+i,-b],)] ,
AB

[[CS([ELT([AB])]i , [GO/+cntc,+cols([ELT([AB])]i),
[TO([AT([ELT([AB])]j)])])])] AND
[CS([ELT([AB])]j, [GO/+cntc,+cols([ELT([AB])]j),
[TO([AT([ELT([AB])]i)])])])]]]

단, i =/= j

논항 자리에 채워진 내용은 다르지만, [ELT([AB])]i와 [ELT([AB])]j가 각각 [A]와 [B]로 바뀌면, 결과적으로 (9)의 의미구조와 공통된 구조를 얻게 된다. 'ELT'의 적용은 의미구조를 대상으로 일괄적으로 이루어진다고 본다. 이상을 근거로 하여 'NP이 NP와' 구조(9)와 'NP와 NP이'구조(13)의 의미적 공통성을 설명할 수 있게 된다.

다음, 교호성을 갖는 (10)의 문장들이 동질적임은 주어 명사구들이 복수성을 띠며, 한자리 동사 '싸우다'가 어휘의미구조에 교호성의 연산자를 가지는 것으로 설정함으로써 포착한다. 이들은 모두 중의성을 갖는 문장이다. 비교호적 해독일 때는 '싸우2'의 어휘의미구조에서 [+reciprocal]이 작동하지 않고 논항 융합만 행해지며, 교호적 해독일 경우에는 역시 '싸우2'의 어휘의미구조에서 그것이 작동하는 것이다.

Jackendoff(1990)에서는 통사구조뿐만 아니라 의미구조('개념구조')도 독립된 적격성 조건(well-formedness condition)을 가지며, 나아가 한 의미구조를 다른 의미구조와 연관짓는 규칙이 존재한다고 말하고 있다. 의미구조-의미구조의 대응을 맺어 주는 규칙으로 '추론규칙'과 '보충해석 규칙'이 있다.[35] [+reciprocal]이 주어의 복수성에 의해 촉발되어 행하는 작용이나, ELT 연산자가 그 논항인 복수성의 의미 성분에서 부분 의미요소를 추출하는 작용은 의미구조를 바탕으로 새로운 형식의 의미구조를 이끌어 내는 것이므로 추론규칙의 일부이다. 일단 의미구조가 얻어지면, 추론규칙은 완성된 의미구조의 형식에 근거하여 일괄적으로 적용된다.

3.4. 타동사적 대칭구문의 통사·의미적 특징

다음 두 가지 종류의 타동사적 대칭구문의 존재는 역시 홍재성(1986)에서 주목되었다.

35. Jackendoff(1990) 참조. '이다' 구문의 해석과 관련하여 추론규칙과 보충해석 규칙의 역할을 논의한 양정석(1996나)를 아울러 참고할 수 있다.

(1) 철수는 영희와 선물을 교환했다.
(2) 철수는 영희를 순이와 혼동했다.

두 가지 구문은 모두 '서로'를 수의적으로 용인하고, '함께'를 거부하는 특징을 보인다. 그런데 (1)과 (2)는 각각 (1)', (2)'의 명사구 접속의 구조와 대응한다는 점에서 차이를 보이기도 한다.

(1)' 철수와 영희는 선물을 교환했다.
(2)' 철수는 영희와 순이를 혼동했다.

전자는 주어의 명사구 접속에, 후자는 목적어의 명사구 접속에 대응된다. 전자의 특징을 보이는 동사들의 다른 예로 '바꾸다, 맞바꾸다' 등이 있고 후자의 하위부류를 이루는 동사에 '바꾸다, 섞다, 뒤섞다, 비교하다' 등이 있다.

흥미로운 것은 '바꾸다'의 경우 두 가지 쓰임을 다 가질 수 있다는 점이다. 따라서 다음 네 가지 경우의 '바꾸다'를 각각 독립된 어휘항목으로 설정한다.

(3) 가. 철수는 영희와 자리를 바꾸었다.(바꾸다1)
　　나. 철수와 영희는 자리를 바꾸었다.(바꾸다2)
(4) 가. 철수는 돈을 양식과 바꾸었다.(바꾸다3)
　　나. 철수는 돈과 양식을 바꾸었다.(바꾸다4)

(1) 또는 (3가) 유형의 동사들이 보이는 어휘의미적 특징에는 매우 독특한 점이 있다. 그 어휘의미구조는 대체로 다음과 같다.

(5) '바꾸다1'의 어휘의미구조: [[CS(x, [GO/+poss(y, [TO(z)])])] AND
[CS(z, [GO/+poss(y, [TO(x)])])]]

먼저 x, y, z 등의 논항 변수들의 위치를 살펴 보면, 앞서의 대칭구문이나 상호구문과 관련하여 제시했던 '교호성'의 의미구조와 별반 다른 점이 없다. 다만 목적어 '자리를'은 위 의미성분이나 아래 의미성분이나 모두 y로 표시되어 변화가 없다.

논항들의 위치에 주의하면서 (2) 또는 (4가)의 어휘의미구조 형식을 검토해 보면 다음과 같다. 다음은 (4가)의 '바꾸다'가 갖는 어휘의미구조의 핵심적인 부분만을 보인 것이다. 사동성의 영향을 받는 부분만이 교호적인 구조를 보이고 있다. 이도 역시 '교호성'의 의미구조의 또 한 가지 형태로 인정할 수 있을 것이다. 여기서 주어는 고정되고,

나머지 두 논항들의 순서가 서로 바뀐 의미 성분이 'AND'로 접속되어 있다.[36]

(6) '바꾸다3'의 어휘의미구조: [[CS(x, [GO/+poss(y, [TO(z)])] AND
[GO/+poss(z, [TO(y)])])]]

(1), (3가) 유형의 동사들이 어휘의미구조 형식과 관련하여 보이는 특이성은 논항들의 동일지시성과 관련한 것이다. (1)에서 철수가 영희에게 준 '선물'과 영희가 철수에게 준 '선물'은 동일한 것이 아니다. (3가)에서도 철수가 영희에게 넘겨 준 '자리'와 영희가 철수에게 넘겨 준 '자리'는 동일한 것이 아니다. 하지만 '바꾸다, 맞바꾸다, 교환하다' 등의 동사들이 이루는 타동사적 대칭구문에서는 목적어 자리에 나타나는 명사구 성분이 항상 이와 같은 의미론적 특징을 가질 것을 요구한다. 심지어 수식어를 동반한 경우라도 이 관계는 유지된다.

(7) 철수는 영희와 앉았던 자리를 바꾸었다.

여기서 '앉았던 자리'는 철수가 앉았던 자리와 영희가 앉았던 자리 두 가지로 다 해석되어야 한다. 이 사실은 어휘의미구조에서 설치되는 논항 변수들이 실재 세계에서의 논항들의 동일성과 결부되는 것은 아니며, 단지 한 어휘 단위 내에 나타나는 논항들이 통사구조의 성분들과 연결되는 관계에 있어서의 동일성과 차별성을 표시해 주는 것임을 말해주는 것이다. (1), (3가)류의 동사들이 보이는 어휘의미적 특이성은 이 점을 드러내 준다는 점에서 중요성을 갖는다. 이를 명시적으로 나타내 주는 한 가지 방법은 (5)와 같은 경우 논항 변수 'y'에 서로 다른 지시 지표 'i'와 'j'를 덧붙이는 것이다.

(5)' [[CS(x, [GO/+poss(y_i, [TO(z)])])] AND
[CS(z, [GO/+poss(y_j, [TO(x)])])]]

이상에서는 '바꾸다1'과 '바꾸다3'만을 고려하였다. 명사구 접속 또는 복수성의 명사구가 주어 위치에 나타나는 '바꾸다2', 그리고 목적어 위치에 나타나는 '바꾸다4'의 어휘의미구조를 기술하는 것은 쉽지 않은 문제이다. 앞서 논의한 자동사적 대칭구문의 경우에 준하여, '바꾸다2'와 '바꾸다4'의 어휘의미구조를 다음과 같이 설정하기로 한다. 이

36. 이와 같은 교호성의 의미구조는 자동사적 대칭구문과 상호구문에서 연산자 'reciprocal'에 의하여 도출된 의미구조와는 다르다. 타동사적 대칭구문은 이렇게 더욱 협소한 어휘개별적 특성을 띠게 된다고 할 수 있다. '서로'에 의해서 형성되는 타동사적 상호구문은 존재하지 않는다.

경우는 앞서와는 달리, 소유(+poss)의 의미 영역이 아닌 동일시(+ident)의 의미 영역에 속하는 것으로 표시하고자 한다.

(8) 가. 바꾸다2: 철수와 영희는 자리를 바꾸었다.
[[CS(x, [GO/+poss(y, [TO([ ])])])] AND
[CS([ ], [GO/+poss(y, [TO(x)])])]]
나. 바꾸다4: 철수는 돈과 양식을 바꾸었다.
[CS(x, [[GO/+ident(y, [TO([ ])])] AND
[GO/+ident([ ], [TO(y)])]])]

사동성 CS와 관련한 부분이 반복되지 않는 것으로 설정한 것은 '서로'가 개입하는 다음과 같은 예를 고려할 때 타당성을 얻는다.

(9) 가. 철수는 돈을 양식과 서로 바꾸었다.
나. 철수는 돈과 양식을 서로 바꾸었다.

'서로'가 갖는 교호성의 의미 작용은 주어에 가해지지 않음을 알 수 있다. 이러한 점을 포착하기 위하여 (8가, 나)의 형식이 의의 있는 것으로 생각된다.[37]

이러한 타동사적 대칭구문에 대해서 변형론적 설명, 특히 접속문 축약변형설이 어떤 해결책을 내놓을 것인지는 상상하기 어렵다. 접속항 이동변형설은 'NP와 NP' 구조와 'NP이 NP와'가 서로 연관됨을 통사적인 차원에서 보일 수는 있겠지만, 그것은 통사적인 차원에서의 임시적인 일반화일 뿐, 각 구문에 대응하는 의미구조를 제시하기 위해서는 이제까지 보아 왔던 통사구조-의미구조의 연결에 관한 온갖 사항들이 다시금 고려되어야 한다.

통사구조와 의미구조의 연결에 관해서 더 복잡한 고려가 필요한 예가 있다. 다음 절에서 논의할 상호구문과 확장 대칭구문이 그것이다.

---

37. (8나)에는 교호성 연산자 '+reciprocal'이 설치된다. 목적어 '돈과 양식을'이 갖는 [+i] 특질을 촉발자로 하여 교호성 연산이 일어나는 것으로 본다. 앞서 자동사적 대칭구문과는 달리, 이 경우는 사동성 및 행위자를 제외한 부분, 즉 다음과 같이 AND에 의하여 결합된 의미 성분이 교호성 연산의 범위가 되는 것으로 본다.
[[GO/+ident(y, [TO([ ])])] AND [GO/+ident([ ], [TO(y)])]]
자동사적 대칭구문의 경우 교호성 연산의 범위는 전체 의미 성분이었으나, 그 경우에도 역시 AND에 의하여 결합된 의미 성분 내부가 문제되었다. 이러한 연산은 의미구조에 대하여 행해지는 '추론규칙'의 성격을 갖는 것이므로, 적용 범위가 되는 의미 성분이 이와 같은 형식적 특징을 갖는다는 사실은 주목할 만한 것이다.

3.5. 상호구문, 확장 대칭구문과 재구조화

'NP와 NP' 구조가 'NP이 NP와' 구조와 서로 관련되는 현상은 다음과 같은 상호구문에서도 나타난다. 다음의 문법성의 대비를 살필 때, 'NP와 NP' 또는 'NP이 NP와'의 형식은 '서로'로 말미암아 형성되는 것임을 알 수 있다.

(1) 가. 철수는 영호와 서로 공격하였다/쳐다보았다.
　　나. 철수와 영호는 서로 공격하였다/쳐다보았다.
(2) 가. *철수는 영호와 공격하였다/쳐다보았다.(교호적 의미로)
　　나. *철수와 영호는 공격하였다/쳐다보았다.(교호적 의미로)

이들 경우 '서로'는 대명사가 아니라 부사로 쓰인 것이다. '서로'가 대명사로서의 목적어가 아니라는 것은 다음을 통해서 안다.

(3) 가. *철수는 영호와 서로를 공격하였다/쳐다보았다.
　　나. 철수와 영호는 서로를 공격하였다/쳐다보았다.

(1나)를 기저적인 구조로 하여 (1가)의 구조를 변형에 의하여 이끌어 내는 방법이 옳지 않다는 점은 양정석(1996가)에서 밝힌 바 있다. 이렇게 볼 때 (1가)와 같은 구문은 생성문법의 일반적인 관점에서 볼 때 매우 특징적인 면을 드러내고 있다. 동사가 부사 '서로'와 결합함으로써 새로운 하위범주화 및 어휘의미적 특징을 구현한다는 점이 그것이다. 2자리의 타동사이던 '공격하다/쳐다보다'가 '서로'와 상호작용함으로써 일반 대칭동사의 교호적 성격을 이루어 가는 것이다. 이 점을 포착하기 위하여 양정석(1996가, 1997가)에서는 (4)와 같은 어휘규칙을 설정하였다.

```
(4) ┌ V               ┐     ┌  [[서로][V]]                        ┐
    │ x< y >          │ --> │  x< y^COM >                         │
    │ ┌[AFF(x, y)]┐   │     │ ┌ [AFF( x ,   )]                  ┐ │
    └ └[F(x, y)]  ┘   ┘     └ └[[F( x, y )] AND [F( y, x )]]    ┘ ┘
```

타동사의 어휘통사구조 형식 'x< y >'이 2자리 대칭동사의 형식 'x< y^COM >'로 바뀌며, 역시 보통의 타동사의 어휘의미구조가 교호적인 어휘의미구조 형식으로 바뀌는 점이 어휘규칙으로 포착되어 있다.

양정석(1996가)에서는 '서로'에 의한 상호구문이 (5가)와 같은 '확장 대칭구문'[38]과 유

사한 통사적 양상을 보임을 관찰하였다. 그러나 (6)의 문법성 대비가 보여주는 바는 '교류를'과 '하-'가 인접함에 비해 '서로'와 '공격하-'는 인접하지 않는다는 것이다.

(5)가. 남한이 북한과 교류를 하였다/시작하였다.
나. 철수는 영호와 서로 공격하였다.
(6)가. *남한이 북한과 오래 전부터 교류를 남 모르게 하였다/시작하였다.
나. 철수는 영호와 몇 분 전에 서로 이유 없이 공격하였다.

양정석(1997가)에서는 (5가)의 '교류를 하-'가 통사적으로 한 단위가 되어 '북한과'를 보어로 취하게 된다고 설명하고, 이를 재구조화의 한 예라고 보았다. 재구조화의 조건으로 '교류하-'와 같은 어휘가 어휘부에 존재함으로써 어휘의미적 단위를 이룰 수 있는 근거가 있어야 하고, '교류를'과 '하-' 또는 '시작하-'가 인접해야 한다는 점을 제시하였다. (6)의 차이는, '교류를 하/시작하-'가 재구조화하기 위해서 표면 통사구조에서의 인접 조건을 지키지만, '서로'와 '공격하-'는 그렇지 못하다는 점을 보여준다고 하겠다.

(5나)의 경우에도 '서로'와 '공격하-'가 한 어휘의미적 단위를 이루고 있음을 알 수 있다. 이는 '교류하다'가 대칭동사로서 한 어휘의미적 단위를 이루는 것과 비슷한 것이다. (4)는 '서로'와 '공격하-'가 한 의미 단위를 이루는 사실을 어휘부의 근거로서 제시해 주는 의미가 있다. 다만, 표면 통사구조에서의 인접조건은 지켜지지 않는다. 그러나 통사구조의 다른 층위로서 논리형태(LF)를 가정한다면, (5나) 뿐만 아니라 (6나)에서도 2자리 타동사 '공격하-'가 '서로'와 관계하여 대칭동사와 같은 교호적 의미구조를 형성함을 설명해 줄 수 있다. 이에 따라, 양정석(1997가)에서는 (5나)가 (7)과 같은 구조를 갖는데, 논리형태 부문에서 '서로'가 머리성분 이동(head movement)을 수행하여 '공격하-'와 인접하게 되는 것이라고 제안하였다. 이에 따르면 머리성분 '서로'가 이동한 결과로 생겨난 논리형태 표상은 (8)과 같은 것이 될 것이다.

(7) 철수는 $[[\text{영호와 서로}]_{AdvP}\ [\text{공격하}]_V]_{VP}$ 였다.
(8) 철수는 $[[\text{영호와 } t_i]_{AdvP}\ [\text{서로}_i[\text{공격하}]_V]_V]_{VP}$ 였다.

(8)의 구조에 대해서 통사 부문의 재구조화가 적용된다고 하자. 이는 (5가)의 경우와 (5나)의 경우가 하나의 원리에 의하여 지배되는 현상임을 밝히는 결과가 되었다.

이상에서와 같이, (4)와 같은 어휘규칙은 통사 부문으로부터 독립된 순수한 어휘부의

38. 이는 홍재성(1986)의 용어이다.

규칙으로 남아 있게 된다. 이러한 규칙은 직접 통사 부문에서 작동하는 것이 아니고, 다만 통사 부문의 특정 규칙(논리형태 재구조화)을 유발하는 근거가 된다.

(1나)의 'NP와 NP이' 구조도 마찬가지로 설명할 수 있다. 이 경우에 '서로'는 보어를 취하지 않고 홀로 부사구를 이룬다. '공격하-'와 관련되어 하나의 어휘의미적 단위를 이루는 점에서도 같지만, 이 역시 대칭동사들이 'NP와 NP이' 구조를 이룰 때의 어휘의 미구조를 갖게 된다.

상호구문의 이상과 같은 통사·의미적 양상은 'NP와 NP이' 구조와 'NP이 NP와' 구조를 변형 관계로 맺어진다고 함으로써 설명해 줄 수가 없다. 이러한 양상은 '서로'가 갖는 어휘적인 특질로부터 말미암은 것이다. 두 구조 사이의 의미적 동질성은 결과적으로 얻어진 문장의 의미구조가 동질적임으로 해서 설명되는 것뿐이다.

(5가, 나)를 통하여 통사적 기제로서의 재구조화의 존재를 확인하였다. 이밖에도 다음과 같은 것들을 재구조화의 사례로 더 들 수 있다. (9)는 (5가)와 같은 유형의 재구조화 구문이다.

(9) 가. 기업주들이 공장을 폐쇄{를,는} 한다.
　　나. 그 아이가 성원아파트로 배달{을,은} 갔다.
　　다. 경애가 여류비행사를 꿈{을,은} 꾸었다.
(10) 가. 임원진이 단상에 자리{를,는} 잡았다.
　　나. 그 여자가 참 상냥{을,은} 하다.

(9)의 예들은 복합 명사구를 포함하는 (9)'와 대응된다는 점에서 (10)과 같지 않다.

(9)' 가. 기업주들이 공장의 폐쇄{를,는} 한다.
　　나. 그 아이가 성원아파트로의 배달{을,은} 갔다.
　　다. 경애가 여류비행사의 꿈{을,은} 꾸었다.
　　라. 남한이 북한과 교류{를,는} 하였다. (=(5가))
(10)' 가. *임원진이 단상에의 자리{를,는} 잡았다.

(10)과 같은 부류에 속하는 예로는 다음을 더 들 수 있다.

(11) 가. 중국이 전쟁을 시작을 했다.
　　나. ?*중국이 전쟁의 시작을 했다.
(12) 가. 미국이 북한과의 교류를 시작을 했다.
　　나. ?*미국이 북한과의 교류의 시작을 했다.

양정석(1997가)에서는 (9), (10)과 같은 예들을 모두 '재구조화'의 개념 하에 묶어 보았다. 선행하는 명사 또는 명사 상당의 요소가 보조사와 함께 후속하는 동사와 결합하는 것이 그 형식적인 양상이다. 이러한 절차가 일어나는 것은 이들 인접하는 요소들이 하나의 어휘의미적 단위를 이루려 하기 때문이라 생각된다. 이것을 '의미 단위 조건'과 '인접 조건'으로 나누었다.

조사 '를' 대신 '는' 등의 보조사가 개입할 수도 있다. 특히 (10나)의 경우 '를'이 격표지로 쓰였다고 보기는 어렵다. 또, (9), (10)의 예들이 보이는 중요한 특징은, 이러한 구성을 이루는 동사로는 '하다' 대신 '시작하다, 끝내다, 계속하다'와 같은 시상적 변이 의미를 표현하는 동사들이 나타날 수 있는 것이다. 이러한 구문의 통사구조에서 이들을 '하다'와 동일한 지위를 갖는 것으로 간주하는 것이 바람직하다.

이러한 사실들을 모두 고려해 주기 위해서는 어휘부에 '폐쇄를 하', '상냥을 하', '배달을 가', '꿈을 꾸', '자리를 잡'이 각각 '폐쇄하', '상냥하', '배달가', '꿈꾸', '자리잡'의 존재로 말미암아, 이들의 어휘의미구조를 건네받아 자기것으로 하는 과정, 즉 재구조화가 상정되어야 한다.[39]

4. 프롤로그로의 구현

4.1. 통사론을 프롤로그로 구현하기

3절에서는 'NP와 NP' 구성의 통사구조를 핵계층 구조로 표상하였고, 동반구문의 통사구조에 대한 전체적인 관점을 보였으며, 자동사적 및 타동사적 대칭구문의 통사구조가 어휘적 특질에 따라 형성되는 것으로 간주하였다. 또 상호구문의 경우, '서로'가 부사이면서 보어를 취하는 구문의 존재에 주목하였고, 이와 관련되는 재구조화 구문이 그 통사구조에 있어서 아주 독특한 면을 가진다는 것을 관찰하였다.

3절에서의 관찰을 구문분석기의 문법 부분에 반영할 수 있다. 구문분석기는 2절에서 고려한 바대로 좌변 구문분석(left-corner parsing)의 기법에 입각하여 구현한다. 문법의 형식이 달라짐에 따라 구문분석기의 형식화에 있어서도 부분적으로 달라진 것이 있으나 본질적인 작동의 방식은 2절에서 보인 것과 같다. 실행 기법 상의 차이로는 link check을 수행하여 예상되는 좌변의 범주들을 미리 예거(instantiation)함으로써 실행 효율의 극대화를 꾀하였다는 점이다. 문법 부분을 제외한 구문분석기 프로그램은 <부록>으로 제시하였다.[40]

---

39. 재구조화에 대해서 더 자세한 것은 양정석(1997가)를 참고할 것.

이 절에서는 구문분석기 중 그 문법 부분만을 들어 가면서, 앞선 통사·의미적인 특징들에 대한 반영의 측면을 중심으로 설명하고자 한다.[41]

```
(1) X² 규칙 : 명시어(specifier)를 위한 규칙
( x(2,Cat,x(2,Cat)/[X1],Phi,F,_) ->> [x(1,Cat,X1,Phi,F,_)] ) :- nospecifier(Cat).
nospecifier(c). nospecifier(v). nospecifier(d). nospecifier(p).
nospecifier(n). nospecifier(adv). nospecifier(adn).
( x(2,Cat,x(2,Cat)/[Spec,X1],Phi,F,_) ->>
      [x(2,SpecCat,Spec,_,_,_), x(1,Cat,X1,Phi,F,_)] ) :- left_spec(Cat, SpecCat).
left_spec(i,d).

(2) 'NP와 NP(와)' 구성을 위한 규칙 :
( x(2,Cat,x(2,Cat)/[X1,PSpec],Phi,F,_) ->>
      [x(1,Cat,X1,Phi,F,_), x(2,PSpecCat,PSpec,Phi,_,_)] ) :-
      right_spec(Cat, PSpecCat, Phi, G, H).
right_spec(p,n,conj,0,0). % right_spec(p,p,conj,0,0).[42]

(3) 부가어 규칙 : X²에 부가되는 것으로 설정
( x(2,Cat,x(2,Cat)/[Adjunct,X2],Phi,F,_) ->>
      [x(2,AdjunctCat,Adjunct,_,_,_), x(2,Cat,X2,Phi,F,_)]) :-
      left_adjunct(AdjunctCat, Cat).
left_adjunct(adv,v).
( x(2,Cat,x(2,Cat)/[X2,Adjunct],Phi,F,_) ->>
      [x(1,Cat,X1,Phi,F,_), x(2,AdjunctCat,Adjunct,_,_,_)]) :-
      right_adjunct(Cat,AdjunctCat,F).
right_adjunct(v,adv,ec). right_adjunct(v,p,ec).

(4) 가. X¹ 규칙 : 보어를 취하지 않는 경우
( x(1,Cat,x(1,Cat)/[X0],Phi,F,_) ->>  [x(0,Cat,X0,Phi,F,_)] ) :- nocomplement(Cat).
nocomplement(v). nocomplement(n).
nocomplement(adv). nocomplement(adn).
   나. X¹ 규칙 : 보어를 취하는 경우
( x(1,Cat,x(1,Cat)/[Complement,X0],Phi,F,_) ->>
      [x(2,CompCat,Complement,H,G,_), x(0,Cat,X0,Phi,F,_)] ) :-
```

40. 실지로 실행하기 위해서는 link 생성기(부록의 프로그램에서 xgenlink.pl로 칭한 것)가 더 필요하며, 분석된 통사구조를 화면상에서 읽기 편하게 해 주는 프로그램 pptree.pl을 쓸 수 있다.
41. 아래의 문법 프로그램은 yjsgram.pl이라 칭하기로 한다.
42. 이는 'NP와 NP와' 형식의 명사구 접속을 위해서 필요하다.

```
        complement(Cat,CompCat,Phi,H,F,G).
  complement(c,i,0,0,fe,ec).    complement(c,i,0,0,rstr,moved).    complement(c,i,0,0,fe,moved).
complement(c,i,0,0,rstr,ec). complement(i,v,0,0,ec,rstr). complement(v,d,_,_,rstr,_).
  complement(d,n,0,0,0,0). complement(p,n,conj,0,0,0). complement(p,n,com,0,0,0).
  complement(d,p,0,conj,0,0).complement(adv,p,reci,com,0,0).
  complement(adv,p,com,com,0,0).
```

[43]

다. 머리성분 이동으로 I^0에 위치하는 동사들의 투사를 위하여

```
( x(1,Cat,x(1,Cat)/[Complement,X0],_,F,_) ->>
        [x(2,CompCat,Complement,_,G,_), x(0,Cat1,X0,_,F,_)] ) :-
        complement2(Cat,Cat1,CompCat,F,G).
  complement2(i,v,v,moved,ec).
```

(5) 가. X^0 규칙 : N-D-V 재구조화를 위한 규칙

```
( x(0,v,x(0,v)/[PredNoun,D,V0],_,rstr,_) ->>
        [x(0,n,PredNoun,_,X,_), x(0,d,D,_,_,_), x(0,v,V0,_,Y,_)] )
        :- restructuring_condition(X,Y).
  restructuring_condition(pred,lv).
```

나. X^0 규칙 : I-C 재구조화를 위한 규칙

```
  x(0,c,x(0,c)/[I0,C0],_,_,_) ->> [x(0,i,I0,_,_,_),x(0,c,C0,_,_,_)].
```

(6) 가. X^0 규칙 : 어휘부 대응규칙

```
  x(0,v,x(0,v)/[x(0,n)/[-kyolyu],x(0,d)/[-D],x(0,v)/[-ha]],_,rstr,
      Y^X^[Dsem,kyolyuha(X,Y)]) :- readj(D,Dsem).
  x(0,v,x(0,v)/[x(0,n)/[-sicak],x(0,d)/[-D],x(0,v)/[-ha]],_,rstr,
      Y^X^[Dsem,begin(X,Y)]) :- readj(D,Dsem).
  x(0,v,x(0,v)/[x(0,n)/[-sangnyang],x(0,d)/[-D],x(0,v)/[-ha]],_,rstr,
      X^[Dsem,sangnyangha(X)]) :- readj(D,Dsem).
  x(0,v,x(0,v)/[x(0,n)/[-cali],x(0,d)/[-D],x(0,v)/[-cap]],_,rstr,
      Y^X^[Dsem,inch(be(X,Y))]) :- readj(D,Dsem).
  readj(nun,[contrast]). readj(un,[contrast]). readj(lul,[lulsem]). readj(ul,[lulsem]).
  readj(man,[only]). readj(to,[also]).
  x(0,c,x(0,c)/[x(0,i)/[-et],x(0,c)/[-ta]],_,_,X^[[perf_dec],X]).
  나. X^0 규칙 : 어휘부 대응규칙
  x(0,v,x(0,v)/[x(0,n)/[-N0],x(0,d)/[-D],x(0,v)/[-sicakha]],_,rstr,
      Y^X^[[Dsem,inceptive],Sem]) :-
      x(0,v,x(0,v)/[x(0,n)/[-N0],x(0,d)/[-D],x(0,v)/[-ha]],_,_,Sem).
  다. X^0 규칙 : 어휘부 대응규칙
```

43. 이와 같은 부분은 '영호와 서로'를 한 부사구로 처리하기 위하여 필요하다.

```
x(0,v,x(0,v)/[x(0,adv)/[-selo],x(0,v)/[-kongkyekha]],reci,rstr,
    Y^X^[aff(X,noarg), and(attack(X,Y), attack(Y,X))]).
x(0,v,x(0,v)/[x(0,adv)/[-selo],x(0,v)/[-chyetapo]],reci,rstr,
    Y^X^[aff(X,noarg), and(look-at(X,Y), look-at(Y,X))]).
```

(7) X^0 규칙 : 일반 어휘항목들

```
x(0,c,x(0,c)/[-nunta],_,_,X^[[dec],X]) ->> [nunta].
x(0,i,x(0,i)/[],_,ec,X^X) ->> [].
x(0,d,x(0,d)/[-nun],_,_,X^[[contrast],X]) ->> [nun].
x(0,d,x(0,d)/[-lul],_,_,X^[[lulsem],X]) ->> [lul].
x(0,d,x(0,d)/[],_,ec,X^X) ->> [].
x(0,p,x(0,p)/[-wa],_,_,X^Y^[[i,~b],and(X,Y)]) ->> [wa].
x(0,p,x(0,p)/[-wa],com,_,X^X) ->> [wa].
x(0,p,x(0,p)/[],_,ec,X^X) ->> [].
x(0,n,x(0,n)/[-chelswu],_,_,[chelswu]) ->> [chelswu].
x(0,n,x(0,n)/[-yengho],_,_,[yengho]) ->> [yengho].
x(0,n,x(0,n)/[-kakok],_,_,[kakok]) ->> [kakok].
x(0,n,x(0,n)/[-kyolyu],_,pred,_) ->> [kyolyu].
x(0,n,x(0,n)/[-sicak],_,pred,_) ->> [sicak].
x(0,v,x(0,i)/[-ssawu],reci,moved,X^[[reciprocal], aff(X,noarg),
    and(cs(X,go([cntc,cols],X,to(at(i_arg)))),
    cs(i_arg,go([cntc,cols],i_arg,to(at(X)))))]) ->> [ssawu].
x(0,v,x(0,i)/[-ssawu],reci,moved,Y^X^[aff(X,noarg),
    and(cs(X,go([cntc,cols],X,to(at(Y)))),cs(Y,go([cntc,cols],Y,
    to(at(X)))))]) ->> [ssawu].
x(0,v,x(0,i)/[-nolayha],_,moved,X^[sing(X)]) ->> [nolayha].
x(0,v,x(0,i)/[-nolayha],vt,moved,Y^X^[sing(X,Y)]) ->> [nolayha].
x(0,v,x(0,v)/[-kongkyekha],vt,moved,Y^X^[attack(X,Y)]) ->> [kongkyekha].
x(0,v,x(0,v)/[],_,ec,X^X) ->> [].
x(0,v,x(0,i)/[-chyetapo],_,moved,Y^X^[look_at(X,Y)]) ->> [chyetapo].
x(0,v,x(0,v)/[-ha],_,lv,Y^X^[aff(X,Y)]) ->> [ha].
x(0,v,x(0,v)/[-sicakha],_,lv,Y^X^[[inceptive],aff(X,Y)]) ->> [sicakha].
x(0,v,x(0,i)/[-kyolyuha],reci,moved,Y^X^[kyolyuha(X,Y)]) ->> [kyolyuha].
x(0,adv,x(0,adv)/[-ppalli],_,_,_) ->> [ppalli].
x(0,adv,x(0,adv)/[-ppanhi],_,_,_) ->> [ppanhi].
x(0,adv,x(0,adv)/[-selo],reci,_,_) ->> [selo].
x(0,n,x(0,n)/[-selo],reci,_,[reciprocal,xi]) ->> [selo].
x(0,adv,x(0,adv)/[-hamkke],com,_,X^[hamkke(X)]) ->> [hamkke].
```

앞서 2절에서 제시했던 구문분석 프로그램들의 문법 부분과 비교해서 크게 달라진 점은, 구구조 규칙을 핵계층 이론을 반영한 형식으로 바꾸었다는 점이다. 가령 np, n1, n0은 각각 x(2,n), x(1,n), x(0,n)로 바뀌었는데, 이는 핵계층 이론에서 각각 X''(= X^2), X'(= X^1), X(= X^0)에 해당하는 것이다. 이 문법에서 국어의 통사범주로서 설정된 것은 v, n, adv, adn, p, d, i, c로, 모두 8개이다.

핵계층 규칙은 무한한 통사구조의 생성을 가능토록 허용하고, 그에 따르는 제약들로 하여금 한정된 수의 통사구조만을 적격한 것으로 얻어지게 하는 것이 이 구문분석기의 실행 방식이다. 근래의 원리 매개변인 이론에서, 또는 일반화 구구조 문법(GPSG)에서 문법을 설명하는 기본적인 방식이 이와 같은 것이다.

먼저, 명시어를 취하는 규칙인 (1)과 보어를 취하는 규칙인 (3)은 원칙적으로 그 범주가 무엇이든 적용될 수 있도록 짜여져 있다. 그러나 실제적인 구현에 있어서는 명시어로서 DP(또는 NP)를 요구한다는 점을 제약으로서 부여하는 것이 필요하다. 'left_spec(i,d).'와 같은 사실이 이 제약을 표시해 준다.[44]

명시어를 취하는 규칙은 어순상 명시어가 왼쪽에 위치하는 경우, 즉 (1)과, 오른쪽에 위치하는 경우, 즉 (2)로 나누었다. 앞에서 논의한 'NP와 NP(와)' 구성의 내부 통사구조는 규칙 (2)에 반영되었다. 이 구성을 후치사구(PP)로 설정하고, '와'를 접속항에서나 보어에서나 후치사(P)로 처리하는 것이다. 이러한 방식은 접속조사 '와'를 명사구 접속의 머리성분으로 보아 핵계층 이론에 따라 체계적으로 기술하기 위한 것이지만, 특히 위와 같은 구문분석기의 처리에서 매우 효과적이다. 명사구 접속 구조를 위한 별도의 장치를 따로 도입하지 않아도 되는 것이다. 또, 3.1.절에서도 말한 바와 같이, 의미분석기와 연관지어 의미구조를 얻어가는 과정에서도 유용성이 크다.

각 규칙의 좌변, 이를테면 x(2,Cat,x(2,Cat)/[X1],Phi,_,_) 또는 x(0,c,x(0,c)/[-ta],_,_,_)에서 네번째 논항과 다섯번째 논항(Phi나 '_'로 표시된 부분)은 어휘개별적인 문법적 정보를 특질로 표시하기 위한 것이다. 특히 네번째 논항은 통사·의미적인 하위범주 정보를 기재하였다. 부사 '함께'와 '와/과'는 com으로, 대칭동사나 '서로'는 reci로, 보어를 갖는 동사는 cv로 표시한 것이 그것이다.[45] 다섯번째 논항에는 재구조화와 머리성분 이동을 구현하기 위한 정보들이 표시된다. 공범주는 ec로, 이동된 범주는 moved로, 그리고 재구조화하는 두 요소를 표시하기 위해서 pred와 lv로 나타내었다. 재구조화된 결

44. 이 제약이 없을 경우 link를 생성하고 예거(instantiation)를 하는 과정에서 실패하여 'existence error'라는 신호가 나타나게 된다.

45. 이밖에, 처소의 조사 '에'는 loc로, 경로의 조사 '으로'는 pat, 인용의 조사 '고'는 quo 따위로 나타내는데, 이는 양정석(1997다)에서 어휘통사구조의 일부로 표시된 것들과 대응하는 것이다.

과로 생겨난 머리성분을 표시하기 위해서는 rstr이란 특질을 부여하였다. 각 어휘에 있는 이러한 정보를 근거로 하여 보어의 선택 및 재구조화 규칙이 작동된다. 또, 머리성분 이동 변형이 공범주 및 이에 대한 선택의 정보로써 기호화된다. 여섯번째 논항 자리는 어휘의미구조를 설치하는 용도로 쓰였다. 의미분석기는 기초적으로 이 곳의 정보를 이용한다.

위의 문법 프로그램에서 머리성분 이동 변형을 구현한 원리는, 변형을 이동된 머리성분과 그 흔적 사이의 관계로 파악한 데에 있다. (4)의 보어 선택에 대한 정보와, (7)의 어휘기재항의 정보를 이용하여 흔적이 특정의 위치 외에는 나타날 수 없도록 제약해 놓은 것이다. 이와 같이 함으로써 '와/과' 문장 중의 하나인 동반구문의 통사구조를 무리 없이 기술할 수 있게 된다. 다음과 같은 경우, 동사는 VP 내부로부터 오른쪽 부가어를 건너 I^0로 이동하였는데, 이러한 구조는 위의 문법에 의해서 정확하게 예측된다.

(8) 가. 그는 [[철수를 ti]$_{VP}$영수와 함께]$_{VP}$ 떠나보내i 써다.
나. 그는 [[철수를 ti]$_{VP}$영수와]$_{VP}$ 떠나보내i 써다.

이 프로그램에서 특기할 만한 점은 재구조화에 대한 통사적 처리가 반영되었다는 것이다. 우선, X^0 규칙으로서 'N-D-V 재구조화를 위한 규칙'라고 한 것, 즉 (5가)는 어휘부 재구조화를 처리하기 위한 것이다. 어휘부 재구조화의 통사적 효과는 인접한 요소들을 묶어 하나의 머리성분 범주를 부여해 주는 역할만을 한다. (5나)는 선어말 어미, 즉 I 범주와 어말 어미, 즉 C 범주가 인접하여 재구조화한 것으로 처리한 것이다.

통사론의 구현인 구문분석기에서 재구조화와 관련된 부분은 일단 (5)로 국한되지만, 의미분석기와 관련됨으로써 이 부분이 다시 중요한 역할을 하게 된다. 이 점을 단계적으로 설명하면, 먼저 통사구조에서의 재구조화, 즉 '시작'과 '을'과 '하'가 한 단어 범주 'V'로 허가되는 절차가 (5)로써 구체화된다. 규칙의 우변에서 'pred'와 'lv'의 조건을 충족하는 것들에 한하여 적용되게 되어 있다.[46] 다음으로, 이러한 '숙어적' 머리성분(head)이 의미와 대응되는 것을 보장해 주기 위하여 (6)의 어휘부 대응규칙이 필요하다. 그 중 한 예를 다시 적은 다음 (9)은 '시작을/은/만 하-'와 같은 구성의 의미를 얻기 위해서 필요한 규칙이다. 이는 단어가 음성 형식과 어휘의미의 대응을 명세화해 주는 것처럼 '시작을 하-'도 숙어로서 부분적인 통사구조-의미구조의 대응에 관한 정보를

46. pred와 lv는 각각 서술성 명사와 경동사(light verb)로 대표되지만, 이 외에도 재구조화가 적용되는 '자리를 잡-', '상냥을 하-' 따위가 있음에 주의해야 한다. 이 점에 대해서 양정석(1996가)를 참고할 수 있다.

가져야 한다는 생각을 반영한 것이다. 의미분석기는 이와 같은 형식으로부터 어휘적 의미, 즉 Y^X^[Dsem,begin(X,Y)]를 가져다가 문장 의미의 합성에 이용하게 된다.

(9) x(0,v,x(0,v)/[x(0,n)/[-sicak],x(0,d)/[-D],x(0,v)/[-ha]],_,rstr,
Y^X^[Dsem,begin(X,Y)]) :- readj(D,Dsem).

(9)는 어휘부에 숙어적인 구성 '시작-하'가 존재하지 않으면 '시작을/은/만 하-'를 포함한 문장은 의미 해석을 받지 못한다는 것을 나타내고 있다. 완성된 의미구조를 얻어야만 주어진 문장이 완전히 적격한 것으로 받아들여진다는 원리를 설정한다고 하면, 재구조화와 관련한 일련의 과정은 어휘부 내의 절차이자 의미 해석을 위한 의미론적 절차이며, 동시에 문장의 적격성을 허가받기 위한 통사적인 절차라고 볼 수도 있다. 이는 또 통사적인 절차가 궁극적으로 어휘부의 정보를 근거로 하여 행해짐을 보여준다.

그러나 (10)의 확장 대칭구문과 (11)의 상호구문은 단순치 않은 측면을 포함한다. 또, 이 둘은 서로 다른 특징을 갖고 있기도 하다.

(10) 남한이 북한과 교류를 하였다.
(11) 철수가 영호와 서로 공격하였다.

(10)에서 '북한과'는 '교류'와 함께 명사구를 이루지만, '교류'와 '하-'는 조사의 개재에도 불구하고 서로 인접함으로써 하나의 통사적 단위처럼 행동하며, 하나의 어휘적 의미와 대응된다. 양정석(1997가)에서는 이러한 측면을 'D구조 재구조화'로 설명한 바 있다. 이 결과로 얻어지는 통사구조 층위는 S구조이다. (11)은 '논리형태 재구조화'로 설명되는데, 부사 '서로'가 '공격하-'로 머리성분 이동을 수행함으로써 '서로'의 흔적이 부사구(AdvP) 안에 남게 된다고 하였다. 이 결과로 얻어지는 표상 층위는 논리형태 층위이다.

먼저 (11)을 살펴 보자. '서로'가 논리형태에서 상위의 머리성분으로 이동하는 현상에 대해서 논리형태 재구조화의 개념을 도입하여 설명할 수 있다고 하였다. 이동이란 개념을 다만 상위 구 머리성분과 하위 구 머리성분 사이의 관계로 간주할 수 있다. 이와 같이 논리형태 표상이 하위 구와 상위 구 사이의 관계를 통사구조 성립을 위한 한 제약으로서 설정해 주는 기능만을 갖는다면, 이를 독립된 표상 층위로 설정하지 않고서도 동일한 효과를 얻을 수 있다. 위 구문분석기에 다음 규칙을 더함으로써 통사구조 T를 얻어내게 할 수 있다. 이는 위에 제시된 parse(P, T) 규칙에, 뒤따르는 제약들을 가

하는 형식으로 되어 있다. 제약으로는 어휘부 대응규칙 (6다)가 이용되는바, '서로'와 공범주인 V, '공격하-'의 관계를 확인해 준다.[47]

```
(12) parse2(P,T) :- parse(P,T), subtree(X,T), subtree(Y,T), subtree(Z,T),
                    subtree(XP,T), subtree(YP,T), subtree(ZP,T),
                    head-of-mp(X,XP), head-of-mp(Y,YP),
                    head-of-mp(Z,ZP), head-of-mp(Y,ZP),
                    head-of-mp(X,ZP)
                      Y=x(0,v)/[], x(0,v,x(0,v)/[X,Z],reci,rstr,_).
```

(12)에서 사용된 술어의 정의는 (13)과 같은 일련의 술어들을 통하여 얻어진다. 이들 및 (14)를 이용하여, (1)-(7)에 의해 생겨나는 통사구조들에 다양한 제약을 부과할 수 있다. (12)는 부분구조(subtree)들 사이의 관계에 바탕을 둔 정의이며, (13)은 부분구조가 갖는 범주 명칭('x(0,v)'나 'x(2,v)'등)들 간의 관계에 대한 정의이다.

```
(13) head(X) :- X=x(0,_)/_.
     mp(X) :- X=x(2,_)/_.
     adjunct(X,Y) :- mp(X), mp(Y),
          Y=x(2,Z)/ [x(2,Z)/_,X] .
     daughter(X,Y) :- Y=x(_,_)/ [X,_] ;
          Y=x(_,_)/ [_,X] ; Y=x(_,_)/ [X] .
    head-of-mp(X,Y) :- head(X), mp(Y), mp(Q),
          subtree(Q,Y), daughter(Z,Q), Z=x(1,_)/_.

(14) subtree(T/Subtrees, T/Subtrees).
     subtree(Subtree, _/Subtrees) :-
        member(Tree, Subtrees), subtree(Subtree, Tree).
     member(Element, [Head|Tail]) :- member_(Tail, Head, Element).
     member_(_, Element, Element).
     member_([Head|Tail], _, Element) :- member_(Tail, Head, Element).
```

47. 양정석(1997가)에서는 (11)의 경우, '서로'가 논리형태의 머리성분 이동(head movement)를 행하여 동사 '공격하-'와 인접하게 된다고 설명하였다. 필자의 취지는 어휘부 재구조화, D구조 재구조화, 논리형태 재구조화가 공히 인접 조건 및 어휘의미 단위 조건에 지배된다는 점을 보이는 것이었다. 현재의 구문분석기에는 아직 인접 조건에 관한 것이 직접적으로 구현되지 않았다. 그러나 어휘의미 단위 조건은 (6)의 어휘부 대응규칙을 이용하도록 함으로써 구현된 것이라고 하겠다.

```
root(A,A/_).
parent(A, B, A/L) :- member(B/_, L).
parent(A, B, _/L) :- member(Tree, L), parent(A,B,Tree).
dominates(A,B,Tree) :- parent(A,B,Tree).
dominates(A,B,Tree) :- parent(A,C,Tree), dominates(C,B,Tree).
dominates_or_eq(A,A,_).
dominates_or_eq(A,B,Tree) :- dominates(A,B,Tree).
sisters(A,B,Tree) :- subtree(_/Subtrees, Tree),
                     select(A/_, Subtrees, Remainder),
                     member(B/_, Remainder).
select(A,[A|Remainder],Remainder).
select(A,[B|L],[B|Remainder]) :- select(A,L,Remainder).
exclusively_dominates(A,B,A/[B/_]).
exclusively_dominates(A,B,A/[_/[T]]) :- exclusively_dominates(A,B,A/[T]).
exclusively_dominates_or_eq(A,A,_).
exclusively_dominates_or_eq(A,B,Tree) :-exclusively_dominates(A,B,Tree).
c_commands(A,B,_/Subtrees) :- select(A1/A1trees,Subtrees,Remainder),
                              exclusively_dominates_or_eq(A1,A,A1/A1trees),
                              member(B1/B1trees,Remainder),
                              dominates_or_eq(B1,B,B1/B1trees).
c_commands(A,B,_/Subtrees):-member(T,Subtrees), c_commands(A,B,T).
```

확장 대칭구문 (10)에서 관찰되는 'D구조 재구조화'도 이와 같은 방식으로 구현할 수 있다. 그 규칙은 (15)와 같다.

```
(15) parse3(P,T) :- parse(P,T), subtree(X,T), subtree(Y,T), subtree(Z,T),
                    subtree(XP,T), subtree(YP,T),
                    subtree(ZP,T), head-of-mp(X,XP),
                    head-of-mp(Y,YP), head-of-mp(Z,ZP),
                    head-of-mp(Y,ZP), head-of-mp(X,ZP)
                      x(0,v,x(0,v)/[X,Y,Z],_,rstr,_).
```

4.2. 의미분석기를 구현하기

구문분석기에 의하여 얻어진 통사구조를 바탕으로 하고, 어휘기재항의 어휘의미구조를 이용하여 문장의 의미구조를 얻어 내는 일이 의미분석기가 하는 일이다. 종래 생성

문법 연구에서는 이와 같이 의미구조를 얻어내는 일을 의미 해석이라고 불러 왔다. 여기에는 동사의 어휘의미구조에 명사구 등의 논항의 의미가 채워지는 과정인 '논항 융합'과, 부가어들의 의미가 적용되어 전체 문장의 의미를 얻어 가는 부가어 의미해석 규칙이 대표적인 절차로서 포함된다.

의미 해석의 과정을 기술하기 위해서 기초적으로 고려해야 할 것은 의미구조를 어떠한 형식으로 표상하느냐 하는 문제이다. 이 연구에서는 Jackendoff(1990)의 의미 표상 방법을 따라 각 어휘의 의미(어휘의미구조)를 어휘부에서 기재해 주고, 이것을 이용하여 논항 융합을 중심으로 한 의미해석 규칙을 구현하였다. 4.1.절의 프로그램 (6), (7)에서 어휘기재항의 마지막 논항으로 설정된 것이 어휘의미구조이다.

다음 대칭구문에서 (1가)의 의미구조는 (2)와 같은 꼴이 된다.

(1) 가. 철수는 영호와 싸웠다. 나. 철수와 영호는 싸웠다.
(2) [aff([chelswu], noarg),
    and(cs([chelswu],go([cntc,cols], [chelswu], to(at([yengho])))),
        cs([yengho], go([cntc,cols], [yengho], to(at([chelswu])))))].

동사 어휘의미구조에 있는 논항 변수에 명사구의 의미를 채워넣는 '논항 융합'은 다음과 같은 의미분석기로 구현된다. 동사의 어휘의미구조는 구문분석기의 어휘부에서 가져와 이용하기 때문에, 의미분석기 프로그램 자체는 이처럼 간단하다.

```
(3) reduce(Arg^Expression, Arg, Expression).
    combine(Sem1,Sem2,Sem) :- reduce(Sem1,Sem2,Sem).
    combine(Sem1,Sem2,Sem) :- reduce(Sem2,Sem1,Sem).
    sem(_Root/[T1,T2],Sem) :- sem(T1,Sem1), sem(T2,Sem2),
    combine(Sem1,Sem2,Sem).
    sem(_Root/[R1/T1],Sem) :- sem(R1/T1,Sem).
    sem(X,Y) :- ( x(0,_,X,_,_,Y) ->> _ ), X = x(0,_)/_.
    sem(X,Y) :- x(0,_,X,_,_,Y).
```

구문분석기에 의해 얻어진 통사구조 T를 이용하여 'sem(T,S).'와 같은 질의를 하면 그 의미구조로 S의 값을 얻게 된다.

'NP와 NP' 구조인 (1나)도 그 의미구조의 기본 형식이 (2)와 같다. (1나)의 의미 해석의 제1단계로는, 한 개의 논항 변수를 갖는 어휘의미구조에 논항 융합이 이루어져 (4)를 얻게 된다.[48]

(4) [[reciprocal], aff([and([i,¯b], [chelswu], [yengho])], noarg),
and(cs([and([chelswu], [yengho])], go([cntc,cols],
[and([chelswu], [yengho])], to(at(i_arg)))),
cs(i_arg, go([cntc,cols], i_arg, to(at([and([chelswu],[yengho])])))))].

여기까지는 (3)의 의미분석기에 의해 얻어진다. 다음으로 추론규칙이 적용되어야 더 완전한 의미 해석을 얻게 된다. 주어 논항의 의미 성분에서 복수성([i,¯b]로 표시)을 확인하여 교호성 연산자 [reciprocal]을 작동시킨다. (1나) 문장의 경우는 '와'가 그 자신의 어휘의미구조에서 복수성의 의미특질을 가지는 것으로 설정된다. 이 점에 관한 한, 다음 문장들도 (1나)와 같은 절차를 거쳐 의미구조가 얻어진다. 이들도 완성된 의미구조에서 주어 논항의 의미 성분에 있는 [+i,-b]의 특질을 참조하여 교호성 [+reciprocal] 연산자를 적용한다.

(5) 가. 아이들이 싸웠다. 나. 군중이 싸웠다.

교호성 연산자의 실행은 다음과 같은 절차로 이루어진다. 1) (4)와 같이 명시적인 논항이 복수성 명사구로 채워진 구조를 얻은 다음, 2) 복수성을 갖는 첫번째 논항에 elt연산자를 적용시켜 and로 결합된 두 의미 성분의 논항에 교차적으로 논항의 의미를 대입한다. 즉 구별되는 두 의미 elt_j(α)와 elt_j(α)를 두 의미 성분에서 교차되도록 대입한다. 위 예와 같이 α가 '철수와 영호'인 경우에는 elt(α)i와 elt(α)j는 각각 '철수'와 '영호'가 된다. 결국 다음과 같은 의미구조를 얻는다.

(6) [[reciprocal], aff(and([i,¯b],chelswu,yengho), noarg),
and(cs(chelswu,go([cntc,cols], chelswu, to(at(yengho)))),
cs(yengho, go([cntc,cols], yengho, to(at(chelswu))))))].

이는 작용의미층의 형식만을 제외하고는 (2)와 같은 꼴이 되었다.

(4)로부터 (6)과 같은 구조를 얻어내는 프롤로그의 규칙은 다음과 같이 기술할 수 있다. (4)를 S로 지칭하기로 한다.

48. 암시 논항(implicit argument)은 'i_arg'로 표시하며, AFF 함수의 비어 있는 논항 위치는 'noarg'로 설정된다. '+caus' 등의 특질을 각 술어의 첫번째 논항에 별도로 설치할 수 있으나, 이 글에서는 생략한다.

```
(7) sem2(T,S,SEM):- sem(T,S), ground(S),
      subterm(Sub,S), Sub = [[reciprocal],aff(A,noarg),_],
      substitute(A,elt_i(A),S,S1), substitute(i_arg,elt_j(A),S1,SEM), ! .
    subterm(Term,Term).
    subterm(Sub,Term) :- functor(Term,F,N), subterm(N,Sub,Term).
    subterm(N,Sub,Term) :-
      N > 1, N1 is N - 1, subterm(N1, Sub, Term).
    subterm(N,Sub,Term) :- arg(N,Term, Arg) , subterm(Sub, Arg).
    substitute(Old,New,Old,New).
    substitute(Old,New,Term,Term) :- atomic(Term), Term \= Old.
    substitute(Old,New,Term,Term1) :- ground(Term), functor(Term,F,N),
      functor(Term1,F,N), substitute(N,Old,New,Term,Term1).
    substitute(N,Old,New,Term,Term1) :- N > 0, arg(N,Term,Arg),
      substitute(Old,New,Arg,Arg1), arg(N,Term1,Arg1), N1 is N - 1,
      substitute(N,Old,New,Term,Term1).
    substitute(0,Old,New,Term,Term1).
```

이는 완성된 의미구조를 이용하여 그 부분적인 형식을 바꾸는 규칙으로서, 추론규칙의 성격을 갖는 것이다.

추론규칙은 아니나, 추론규칙과 같이 의미구조에 대해서 적용되는 규칙이 있다. (3)에서 구현한 의미분석기로는 앞서 (2나)의 문장 "철수와 영호는 싸웠다."에 대한 의미구조로서 잘못된 형식이 다수 생성된다. 하지만 이들 중에서 부적격한 구문들은 많은 경우 변수가 채워지지 않은 것들이라는 공통성을 갖는다. 그러므로 이러한 구조들을 제거하는 규칙이 의미구조에 대해서 적용되는 것으로 설정할 필요가 있다. 이것은 의미구조에 대해서 적용되는 적격성 조건의 하나로서 "논항 변수는 채워져야 한다."와 같은 원리가 포함되어야 한다는 것을 말해 준다.[49] 이와 같은 것을 프롤로그에서 구현하는 일은 간단하다. 구문분석기로부터 T가 주어졌을 때, 이에 대응하는 의미구조로서 적격한 것을 얻기 위해서는 다음과 같이 설정한다.

```
(8) sem3(T,SEM) :- sem(T, SEM), ground(SEM).
```

'ground(X)'는 프롤로그에 내장된 술어로서, 항 'X'가 변수를 포함하지 않는다는 것을 확인한다. [50]

---

49. Jackendoff(1983: 9.5.절, 1990: 2.1절)에서는 이 점에 주목하고, 이를 '어휘적 변수 원리(lexical variable principle)'라고 부른 바 있다.

5. 결 론

이 연구에서는 조사 '와/과'를 갖는 문장이 여러 가지의 상이한 구문 구조를 가진다는 것을 보였으며, 각각의 구문이 갖는 통사·의미적인 특징을 기술함에 있어서 변형론적 관점과 대비되는 어휘주의적 관점이 옳다는 것을 밝혔다. 각 구문에 대한 통사·의미론적 설명을 보이는 한편, 이를 바탕으로 프롤로그 언어를 통하여 구현함으로써, 이 연구의 어휘주의적인 분석이 기계적인 구현에 있어서도 효율성이 높음을 보였다.

'와/과' 구문에 대해서는 이미 여러 연구자들이 그 구조와 문법적 과정을 여러 가지로 제안하여 왔는데, 대칭동사의 어휘의미적인 분석으로부터 시작하여 그 구문의 통사적 행태, 의미적인 특징을 일관적으로, 그리고 통합적으로 설명하는 일은 과거의 연구에서는 찾아보기 어려웠다. 변형론적/생성의미론적인 관점에서의 연구는 그 의미구조를 고려하고는 있으나 통사구조와의 연결 과정에는 많은 문제점들이 발견된다. 또한, 근래의 일부 어휘주의적인 분석에서는 이들 구문의 통사구조상의 특징을 하나하나 밝혀내고는 있으나, 의미구조와의 연결 과정에 대해서는 구체적으로 해답을 제시하지 못하고 있어 통합적인 이론에의 요구는 아직 충족되지 못하고 있던 상태였다.

이 연구에서는 대칭/상호구문과 동반구문을 대상으로, 통사구조와 어휘부, 통사구조, 의미구조 상호간의 연결 관계를 해명하였으며, 복수성 명사구와 명사구 접속이 공통 특질을 갖는 사실에 대한 통사론적, 의미론적 처리 방안을 제시하였다. 또한 상호구문 및 '확장 대칭구문'으로 지칭되던 '와/과' 구문의 예가 재구조화와 관계되는 현상임을 밝혔다.

특히 프롤로그로의 구현과 관련하여 특징적인 점을 들면 다음과 같다.

1) 'NP와 NP와' 구성을 핵계층 이론에 따라 기술하였으며, 이를 프롤로그로 규칙화하였다.

2) 동반구문은 'NP와'나 'NP와 함께'가 주어에 대해서 서술화하느냐 목적어에 대해서 서술화하느냐에 따라, 전자는 IP의 왼쪽에 부가되는 부가어로, 후자는 VP의 오른쪽 부가어로 해석하였다. 오른쪽 부가어가 나타나는 경우를 위하여 동사의 I로의 머리성분 이동은 매우 유용한 점이 있다.

3) 대칭구문은 두 개의 어휘항목으로 분리하여 그 어휘적 특질에 따라 통사적인 논항의 실현 및 의미구조의 도출을 설명하였는데, 특히 복수성 주어 구문의 경우에는 의미구조의 연산자 특질 [+reciprocal]의 작동에 의해서 의미를 얻어 간다.

---

50 'ground(X)'는 이미 (7)의 일부로 이용되었다.

4) '시작을 하-'처럼 어휘부 재구조화로 설명되는 현상들은 문법에 재구조화 규칙과 어휘부 대응규칙을 설정함으로써 구현하였다. 이와 달리, '교류를 하-'와 같이 D구조 재구조화로 설명되는 현상, '서로 쳐다보-'와 같이 논리형태 재구조화로 설명되는 현상은 문법에 어휘부 대응규칙만을 설정하되, 잠정적으로 생성된 통사구조들에 재구조화와 관련된 제약을 가함으로써 적격한 구조만을 얻어낼 수 있도록 하였다. 'D구조화 재구조화' 및 '논리형태 재구조화'와 관련된 통사적 측면이 의미 해석에서도 참조됨으로써 통사론과 의미론이 상호 긴밀히 협조하는 양상을 보였다.

이 연구의 분석 방법을 어휘주의적 방법이라 칭하였던 것은 궁극적으로 어휘부에 기재된 정보가 통사구조 및 의미구조를 해석해 가는 데에 결정적인 역할을 하는 것으로 보기 때문이다. 이와 같은 어휘주의적인 분석 방법은 프롤로그로 구현하는 데에도 유리하다. 좌변 구문분석(left-corner parsing) 전략에 입각한 구문분석기, 그리고 이를 바탕으로 한 의미분석기를 프롤로그를 통하여 구현함으로써 앞서의 어휘주의적인 통사구조, 의미구조의 분석이 기계적으로 실행 가능함을 입증하였다.

구문분석기와 의미분석기는 생성문법에서 통사적 과정 및 의미 해석 과정의 기술에 대응하는 것이다. 이들 프롤로그 프로그램을 자세히 이해해 가는 과정에서 구구조 규칙이나 핵계층 규칙, 변형 규칙, 의미해석 규칙, 재구조화 등의 의의를 좀더 정확하게 인식할 수 있다. 또, 이러한 구문분석기와 의미분석기를 확장하여 생성문법의 여러가지 이론들에서 고안된 규칙이나 원리를 기계적으로 구현하고 이에 따라 그 이론의 엄밀성이나 일관성을 점검하는 목적으로 사용할 수 있을 것이다. 나아가, 언어학 이론의 교육 등에 유용한 여러가지 응용 프로그램들을 개발하는 데에 직접 이용될 수 있다. 국어학의 통사론 강좌 등에서 이를 이용하여 생성문법의 원리와 기계적 구현 방법 등을 교육한다면 이론적으로나 실용적으로나 그 효과가 클 것으로 생각된다.

참 고 문 헌

김영택(1994). 자연언어처리. 교학사.

김영희(1974). '와'의 양상, 국어국문학 65·66 합병호.

_____(1986). 복합명사구, 복합동사구 그리고 겹목적어. 한글 193.

_____(1991). 셈숱말 '각각'의 문법, 동방학지 71-72합집.

김완진(1970). 문접속의 '와'와 구접속의 '와', 어학연구 6-2.

남기심(1990). 토씨 '와/과'의 쓰임에 대하여, 동방학지 66.
_____(1993). 국어 조사의 용법. 서광학술자료사.
박병수·안상철(1988). 영한 기계번역을 위한 일반 구구조 규칙의 수립, 언어 13-1.
서정목(1993). 한국어의 구절구조와 엑스-바 이론, 언어 18-2.
서정수(1975). 동사 '하'의 문법. 형설출판사.
_____(1994). 국어문법. 뿌리깊은나무.
성광수(1978). 체언 접속과 공격, 한글 162호.
송경안(1992). Prolog and Montague Grammar, Proceedings of '92 SICOL.
송경안·조경숙(1989). LPSG의 의미분석, 언어 14권 통합호.
송석중(1982). 조사 '과', '를', '에'의 의미분석, 말 7. 연세대 한국어학당.
신경구(1992). A linear parsing strategy on Korean based on Linear Phrase Structure Grammar, Proceedings of '92 SICOL.
안희돈(1988). Preliminary remarks on Korean NP, Baek, E, ed., Papers from the 6th International Conference on Korean Linguistics.
_____(1991). Light verbs, VP-Movement, Negation and Clausal Architecture in Korean and English. Doctoral dissertation, University of Wisconsin-Madison.
안희돈·윤항진(1989). Functional categories in Korean, Kuno et al., eds., Harvard Studies in Korean Linguistics III.
양정석(1991). 재구조화를 특징으로 하는 문장들, 동방학지 71-72.
_____(1996가). 대칭구문과 상호구문의 의미 해석, 언어 21권 1·2합병호.
_____(1996나). '이다' 구문의 의미 해석, 동방학지 91.
_____(1996다). '-와/과' 문장의 통사구조, 남기심(엮음), 국어문법의 탐구 III. 태학사.
_____(1997가). 재구조화 재고, 국어국문학 118.
_____(1997나). 이심적 의미구조-동사의 논항 연결과 관련하여-, 배달말 22호.
_____(1997다[1995]). 국어 동사의 의미 분석과 연결이론(개정판). 박이정출판사.
유동석(1995). 국어의 매개변인 문법. 신구문화사.
이기용(1992가). Computing information by equation solving, Proceedings of '92 SICOL.
_____(1992나). 계산의미론에 입각한 한국어 분석, 김태옥교수화갑논총.
이상섭·송만석(1992). 전자사전 구현을 위한 어휘 데이타베이스의 설계와 국어 정보 분석 도구의 개발, 연세논총 28집 별책. 연세대학교.
이익환(1984). 현대의미론. 민음사.

임홍빈(1972). NP-병렬의 {와/과}에 대하여, 서울대 교양과정부 논문집 4.
____(1987). 국어의 명사구 확장 규칙에 대하여, 국어학 16.
최재웅(1987). Anti-Quantifiers and A Theory of Distributivity. Doctoral dissertation, University of Massachusetts.
최재희(1985). 국어 명사구 접속의 연구, 한글 188.
최현숙(1988). Restructuring Parameters and Complex Predicates- A Transformational Approach-. Doctoral dissertation, MIT.
홍재성(1984). '-부터/까지' 복합구성에 대한 별견, 한불연구6, 연세대 한불문화연구소.
____(1985가). 한국어 자동사적 대칭동사의 통사론적 정의, 인문과학 53. 연세대.
____(1985나). 한국어 경쟁구문에 관한 몇 가지 지적, 한글 187.
____(1986). 현대 한국어 대칭구문 분석의 한 국면, 동방학지 50. 연세대 국학연구원.
Abney, S.(1987a). English Noun Phrases and its Sentential Aspects, Doctoral dissertation, MIT.
____(1987b). Licensing and Parsing, NELS 17 vol.1.
Alshawi, H. ed.(1992). The Core Language Engine, MIT.
Berwick et al. eds.(1991). Principle-Based Parsing. Kluwer Academic Publishers.
Chomsky, N.(1964[1957]). Syntactic Structures. Mouton & Co.
____(1965a). Three models for the description of language, Luce, Bush & Galanter(eds.), Readings in Mathematical Psychology, Vol. II.
____(1965b). Aspects of the Theory of Syntax. MIT.
____(1970). Remarks on nominalization, Jacobs & Rosenbaum eds., English Transformational Grammar.
____(1981). Lectures on Government and Binding. Foris.
____(1986). Barriers. MIT.
Clocksin, W. & C. Mellish(1984[1981]). Programming in Prolog, Springer-Verlag.
Dougherty, R.(1970,1971). A grammar of coordinate conjoined structures I-II, Language 46-4., 47-2.
Goodall, G.(1987). Parallel Structures in Syntax. Cambridge University Press.
Jackendoff, R.(1972). Semantic Interpretation in Generative Grammar. MIT.
____(1977). X' Syntax. MIT.
____(1983). Semantics and Cognition. MIT.
____(1990). Semantic Structures. MIT.

_____(1991). Parts and boundaries, Cognition 41.
_____(1994). Lexical insertion in a post-minimalist theory of grammar, ms.
Lakoff, G. & S. Peters(1966). Phrasal conjunction and symmetric predicates, reprinted in Reibel & Schane eds. 1969. Modern Studies in English.
Moltmann, F.(1992). Coordination and Comparatives. Doctoral dissertation, MIT.
Munn, A.(1993). Syntax and Semantics of Coordination. doctoral dissertation, University of Carolina.
Pereira, F. & Shieber, S.(1987). Prolog and Natural-Language Analysis. CSLI.
Pullum, G.(1985). Assuming some version of X-bar theory, CLS 21.
Rothstein, S.(1983). The Syntactic Forms of Predication. Doctoral dissertation, MIT.
Shieber, S.(1986). An Introduction to Unification-Based Approaches to Grammar. CSLI.
Stabler, E.(1993). The Logical Approach to Syntax. MIT.
_____(1996). Computational Minimalism: Parsing and Acquiring languages with movement. ms. UCLA.

<부록 : lcpl.pl >

```
:- op(120,xfy,=>).  :- op(110,xfy,->).  :- op(100,xfy,->>).  :- op(100, fx,-).
:- ensure_loaded(library(pptree)).[51]   :- ensure_loaded(yjssa).[52]
:- ensure_loaded(xgenlink).[53]          :- ensure_loaded(yjsgram).[54]
:- genlinks.
parse(L, Tree) :- proof([] => L, Tree),Tree=x(2,c)/_.
proof(Sequent, Tree) :- goal_sequent(Sequent, Tree).
proof(Sequent, Tree) :- infer(Sequent, Sequent1), proof(Sequent1, Tree).
```

51. 구문분석된 통사구조를 읽기 쉽게 해 주는 프로그램 pptree.pl을 불러들임. 이 연구에서 얻어진 구문분석기와 의미분석기를 실지로 실행하기 위해서는 이하의 프로그램들이 모두 갖추어져야 하나 지면 관계로 구문분석기 프로그램만 제시하는 것이다. 필자가 사용하는 SICStus Prolog System은 1993년 Swedish Institute of Computer Science에서 개발한 것인데 퍼스널 컴퓨터에서 사용할 수 있도록 조정된 것이다. 3.5인치 디스켓 세 장이면 모두 복사가 가능하므로, 이 연구에서 얻어진 프로그램들과 함께 필요한 사람들에게 제공될 수 있다.
52. 의미분석기 yjssa.pl을 불러들임. 이는 4.2.절에서 부분적으로 설명되어 있다.
53. link 생성기 xgenlink.pl을 불러들임.
54. 4.1.절의 문법 yjsgram.pl을 불러들임.

```
goal_sequent([A] => [], Tree) :- arg(3, A, Tree). % A의 3번째 논항
% [좌변/제거 규칙]
infer( [B,i(A)|Gamma] => Delta, IBetaGamma => Delta ) :-
        A ->> [B|Beta],  predict(Beta,Gamma,IBetaGamma).
predict([],L,L).  predict([E|L1], L2, [i(E)|L3]) :- predict(L1, L2, L3).
% [공좌변/제거 규칙]
infer( [i(A)|Gamma] => Delta, Gamma => Delta ) :- A ->> [].
% [좌변 규칙]
infer( [B|Gamma] => Delta, IBetaAGamma => Delta ) :-
        A ->> [B|Beta], predict(Beta,[A|Gamma],IBetaAGamma),
        reducible(Gamma,A).
reducible([],_).  reducible([i(A)|_],B) :- link(A,B).
% [공좌변 규칙]
infer( Gamma => Delta, [A|Gamma] => Delta ) :- A ->> [],  reducible(Gamma,A).
% [옮기기/제거 규칙]
infer( [i(W)|Gamma] => [W|Delta], Gamma => Delta ).
% [옮기기 규칙]
infer( Gamma => [W|Delta], [W|Gamma] => Delta ) :- reducible(Gamma,W).
```