英韓 번역시스템

金 榮 澤
서울大學校 工科大學
電算機工學科 敎授

Ⅰ. 序 論

1. 기계 번역 시스템의 발전 과정

기계 번역에 대한 연구는 1930년대 부터 시작되었다. 이때의 연구는 사람들의 관심을 끌지 못했으며 컴퓨터가 등장하는 1950년대까지 소수의 연구자들에 의해서만 연구 되었다. 이 당시의 주목할만한 연구로는 1946년 W. Weaver와 A. Booth에 의하여 이루워졌다. 이들은 컴퓨터를 이용하여 문자나, 단어의 발생빈도에 따라 암호를 해독하는 일에 종사한 사람들로서 기계 번역은 번역에 관계되는 언어의 사전을 컴퓨터에 넣고 언어간의 단어만을 교환하면 이루워 질 것으로 생각했었다. 물론 이들도 단어들의 context-sensitivity, 문법변환의 필요성 그리고 관용구 번역의 어려움 같은 문제점들을 인식하고는 있었지만 기계 번역은 궁극적으로 사전의 문제라고 생각했었다.[10]

그러나 연구가 진행됨에 따라 이 방법에는 한계가 노출되었으며 1952년도에는 사전 검색 프로그램과 machinese라고 불리우는 가상적인 중간단계의 언어가 제시되기도 하였다. 이후 몇년간 연구가 계속 진행되었다. 특히 Georgetown대학이 개발한 시스템은 주목할만하며 아직도 핵 물리학에 관한 텍스트를 소련어에서 영어로 번역하는데 사용되고 있다. 이 시스템 최근 version인 SYSTRAN은 현재 미국과 유럽의 EEC에서 사용되고 있다. SYSTRAN외에 또 다른 PAHO 시스템은 1976년도에 개발이 시작되어 1979년에 끝이났으며 최초의 version은 곧 스페인어와 영어간의 번역에 쓰이게 되었다. 1980년도에 들어와 미비한 점을 보완하기 위하여 개발이 진행되었고 이 시스템은 현재 비용과 시간면에서 약 50% 정도의 절약을 가능하게 해준다. 이러한 시스템을 제 1 세대 기계 번역 시스템이라고 부르며 이 시스템들은 텍스트를 독립된 문장들의 집합으로 간주하여 처리하며 약간의 구문분석만을 수행했었다. 그러나 이러한 문제점에도 불구하고 SYSTRAN이나 PAHO 시스템등은 현실적인 사용가치가 입증 되었었다.[9,10]

이후 미국에서는 1966년 ALPAC에 Y. Bar-Hillel이 완전한 기계 번역은 당시의 수준으로서는 불가능하며 가까운 장래에 실현될 가능성이 없다고 보고함으로서 일단 그때 까지의 연구 마저도 거의 중단되고 말았다.[10]

1970년대에 들어서면서 언어학에서 개발된 변형생성문법 이론이나 W. Woods에 의하여 개발된 ATN(Augmented Transition Networks)문법 그리고 C.Fillmore의 격 문법(case grammar) 등이 자연 언어 처리에 효과적으로 적용될 수 있다는 것이 제시되었다. 특히 ATN으로 파서를 만들고 격 문법으로서 입력 문장의 의미를 분석하는 방법은 여러 연구자들에 시도가 되었으며 미국에서는 Lisp 언어를 이용하여 실제로 많이 구현되었다. 이 방법은 목표 언어의 생성 과정과 원시언어의 분석과정을 분리할 수 있게 만들었으며 현재 transfer 방식이라고 불리우는 기계 번역 시스템의 대표적인 구조 형태를 만들었다. 이 당시의 주목되는 연구로는 R. Simmons[12]에 의한 semantic network와 격 문법을 혼합한 방법과 Y. Wilks와 R. Schank[11]에 의하여 연구된 방식이있다. 그들은 입력 문장을 semantic primitive의 집합으로 변형시킨다. Semamtic primitive는 격 문법의 일종이며 입력 문장의 내부표현(internal representation)으로서 모든 언어에 공통되는 의미의 요소이다. 입력 문장이 일단 semantic primitive의 집합으로 바꾸어지면 이들로 부터 목표 언어를 생성해 내는 것은 비교적 용이하게 된다. 이러한 방식에 의한 시스템을 제 2 세대 시스템이라고 부르며 이 시스템들은 그 제 1 세대 시스템들과는 달리 실험용으로만 제작되었다.[9]

이후 이러한 시스템의 문제점을 극복해나감과 동시에 인공 지능 분야에서 개발된 이론들이 기계 번역 시스템을 구현하는데 쓰이게 되었다. 이러한 시스템들은

제 3 세대 시스템이라고 불리우며 이 시스템들은 지식 베이스(knowledge-base)를 이용한다. 현재 미국에서는 보다 개선된 기계번역 시스템을 위한 연구가 활발히 진행되고 있으며 유럽에서는 EUROTRAN 프로젝트가 진행중에 있다.[9]

2. Prolog 언어와 DCG(Definte Clause Grammar)

1980년도에 들어서면서 세계적인 주목을 받게된 Prolog 언어는 R. Kowalski와 A. Colmerauer에 의하여 발표 되었다.

Prolog를 이용한 자연 언어 처리 시스템은 유럽 특히 영국에서 연구가 되었다. 영국에서는 미국에서 개발된 ATN formalism과 동등한 power를 지닌 DCG formaltsi이 개발되었다. DCG formalism에서는 Prolog의 한 predicate이 문법의 한 카테고리(category) 로 일 대 일 대응이 되기 때문에 자연어 처리를 위한 파서를 작성하기가 매우 용이하며 더우기 predicate의 argument를 이용해서 context-sensitvity를 쉽게 나타낼 수 있다.[8]

DCG에 대하여 자세히 살펴보자.[8] 다음의 프로그램은 간단한 영어 문장을 파스할 수 있는 CFG (context-free grammar)이다.

```
sentence→noun-phrase, verb-phrase.
noun-phrase→determiner, noun, rel-clause.
noun-phrase→name.
verb-phrase→trans-verb, noun-phrase.
rel-clause→[that], verb-phrase.
rel-clause→[ ].
determiner→[a].
determiner→[every].
noun→[man].
name→[john].
name→[mary].
tran-verb→[loves].
```

(문법 1)

위의 CFG는 Prolog에 의하여 직접 실행될 수는 없다. 이것은 DGC를 간략시켜 표현한 것 (syntactic sugaring)[8]이며 grammar rule notation[7,8]이라고 불리운다. Prolog에 의하여 실행가능한 CFG파서가 되기 위해서는 문법 2 와 같이 고쳐 작성해야 한다.

```
sentence(S0, S) : -noun-phrase(S0, S1),
                   verb-phrase(S1, S).
noun-phrase(S0, S) : -determiner(S0, S1),
                      noun(S1, S2),
                      rel-clause(S2, S).
noun-phrase(S0, S) : -name(S0, S).
verb-phrase(S0, S) : -trans-verb(S0, S1),
                      noun-phrase(S1, S).
rel-clause(S0, S) : -relative(S0, S1).
                     verb-phrase(S1, S).
rel-clause(S, S)
determiner([a : S], S).
determiner([every : S], S).
noun([man : S], S).
name([john : S], S).
name([mary : S], S).
relative([that : S], S).
trans-verb([loves : S], S).
```

(문법 2)

대표적으로 첫번째 predicate의 의미는 "만약 S0에서 S 까지가 문장을 형성하려면 S0에서 S1까지가 명사구를 형성하여야하고 S1에서 S까지가 동사구를 형성하여야한다"이다. 가령 "John Loves Mary" 문장은 "John"이 명사구를 "Loves Mary"가 동사구를 형성하기 때문에 위의 문법에 의하여 파싱이 된다.

파싱이외에도 DCG는 세가지 중요한 기능을 제공하는데 이 기능은 언어 분석에 효과적으로 사용될 수 있다.

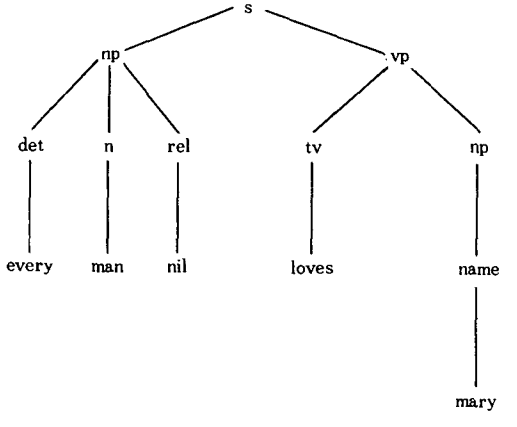
첫째로 파스 트리를 생성할 수 있는 기능이다. 이것은 문법 3 과 같이 문법 1 을 변형하여 얻을 수 있으며 파스 트리는 프로그램이 실행될때 패턴 매칭에 의하여 자동적으로 만들어 진다.

```
sentence(s(NP, VP)) : -noun-phrase(NP),
                       verb-phrase(VP).
noun-phrase(np(DET, NOUN, REL)) :
                       -determiner(DET),
                        noun(NOUN),
                        rel-clause(REL).
noun-phrase(np(NAME)) : -name(NAME).
verb-phrase(vp(TV, NP)) : -trans-verb(TV),
                           noun-phrase(NP).
rel-clause(rel(that, VP)) : -[that], verb-phrase(VP).
rel-clause(rel(nil)).
```

```
determiner(det(W)) : -[W], {is-determiner(W)}.
noun(n(W)) : -[W], {is-noun(W)}.
name(name(W)) : -[W], {is-name(W)}.
trans-verb(tv(W)) : -[W], {is-tran(W)}.
is-determiner(a).
is-determiner(every).
is-noun(man).
is-name(john).
is-name(mary).
is-trans(loves).
```

(문법 3)

위의 문법에 의하여 "Every man Loves Mary" 라는 문장은 파싱의 결과로 구조 s(np(det(every), noun(man), rel(nil)), vp(tv(loves), np(name(mary)))를 생성하며 이 구조를 그림으로 그리면 다음과 같다.



둘째로 predicate에 여분의(extra) 조건들을 표시할 수 있다.

```
date(D, M) : -mouth(M), [D], {int(D), 0<D, D<32}.
```

이 predicate의 뜻은 "달 M에 속하는 날 D는 달 M을 나타내는 predicate month 뒤에 D를 씀으로서 표시된다. 이때 D는 정수이며 0보다는 크고 32보다는 작아야 한다"이다.

세째로 문맥 의존성(context dependency)을 predicate의 argument를 이용하여 나타낼 수 있다. 문법 4는 여분의 argument를 이용하여 determiner, noun 그리고 verb사이의 "수"(단수, 복수)의 일치성을 처리할 수 있다. 예를 들어 문법 4는 "Every man loves some girl"은 처리하지만 "All men that lives love a woman"은 처리할 수 없다.

```
sentence(s(NP, VP)) : -noun-phrase(N, NP),
                       verb-phrase(N, VP).
noun-phrase(N, np(DT, NO, RL)) :
                       -determiner(N,DT),
                        noun(N, NO),
                        rel-clause(N,RL).
noun-phrase(singular, np(NAME)) : -name(NAME).
verb-phrase(N, vp(TV, NP)) :
                       -trans-verb(N, TV),
                        noun-phrase(N1,NP).
rel-clause(N, rel(that, VP)) : -[that],
                                {is-determiner(W, N)}.
rel-clause(N, rel(nil)) : -[ ].
determiner(N, det(W)) : -[W], {is-determiner(W, N)}.
determiner(plural, det(nil)) : -[ ].
noun(N, n(ROOT)) : -[W], {is-noun(W, N, ROOT)}.
name(name(W)) : -[W], {is-name(W)}.
trans-verb(N, tv(ROOT)) : -[W], {is-trans(W,
                                 ROOT)}.
                                {is-interger(W, N, ROOT)}.
is-determiner(every, singular).
is-determiner(all, plural).
is-noun(man, singular, man).
is-noun(men, plural, man).
is-name(mary).
is-noun(woman, singular, woman).
is-trans(likes, singular, like).
is-trans(like, plural, like).
is-trans(loves, singular, love).
is-trans(love, plural, love).
```

(문법 4)

3. 외국의 동향과 한국의 상황

Prolog를 이용한 자연 언어 처리 시스템은 근래에 들어와서 유럽, 일본, 미국등지에서 시작되었다. 유럽에서는 프랑스, 영국, 헝가리등에서 활발히 연구가 되고 있으며 일본에서는 제 5 세대 컴퓨터 연구의 일환으로 일본어〈-〉영어 간의 기계 번역 시스템에 대한 연구가 진행되고 있다. 일본에서는 이미 상품화된 시스템도 있다. 미국에서는 80년대에 들어와 SRI International에서 D. Warren과 F. Pereira에 의하여 시작되었다.[9]

우리 나라에서는 자연 언어처리 일반 특히 기계 번

역에 대한 연구는 몇년전 부터 진행되고 있었다. 그러나 대부분 언어 현상 혹은 시스템 구성의 부분적 문제를 해결하려는 방향으로 진행되어 왔었다. 지금까지 구현된 시스템들로는 사전 구성[3], 국어의 구문 구조 분석[2] 그리고 기계 번역을 위한 한국어 품사의 자동 분류[6]에 대한 것 등이있다. 최근에는 특정한 도메인 하에서 사용자에게 편의를 제공하기 위하여 자연언어 인터페이스에 대한 연구[4,5]가 활발히 진행되고 있다. 그러나 본 논문은 이와는 달리 Prolog로 구현된 기기 번역 시스템의 전체적 구조를 제시하고 간단한 예로서 그 작동을 보이고저 한다.

Ⅱ. 本 論

1. 본 시스템 구조와 작동

그림1은 전체 시스템의 실제적인 구성도이다. 시스템을 이와 같이 두 부분으로 분리하여 구현한 이유는 개념적으로 서로 상이한 기능을 하는 두 부분(스캐너, 파서와 발생기)을 각기 따로 구현함으로서 관리하고 확장하기가 쉽기 때문이다. 이 시스템은 두 개의 병렬 Prolog 프로세스로 구성되어 있으며 파이프 p1과 p2에 의하여 동기화(synchronisation)된 상태에서 작동을 한다. 첫번째 프로세스는 드라이빙 루틴, 스캐너, 파서 그리고 사전으로 되어 있고 두번째 프로세스는 드라이빙 루틴, 발생기 그리고 사전으로 되어있다.

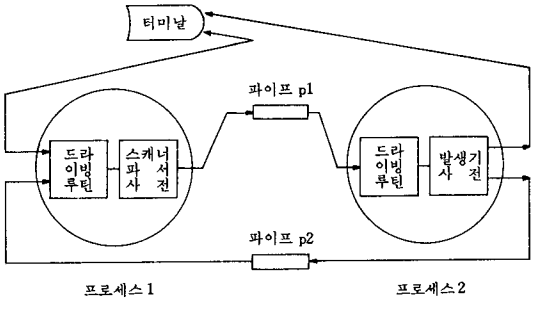


그림 1.

이 시스템의 전체적인 작동은 다음과 같다. 사용자가 터미날에 영어문장을 입력하면 프로세스 1이 이것을 받는다. 이 문장은 프로세스 1의 스캐너와 파서에 의하여 한글 내부 구조로 된다. 프로세스 1은 이 생성된 한글 내부 구조를 파이프 p1을 통하여 프로세스 2에 넘겨준다. 프로세스 2는 파이프 p1을 통하여 받은 한글 내부 구조를 이용하여 한글 문장을 생성한다. 만약 한글 발생과정이 성공적으로 끝이나면 번역된 문장은 터미날에 출력이 되고 프로세스 2는 파이프 p2를 통하여 success 신호를 프로세스 1에 보낸다. 이 신호를 받으면 프로세스 1은 다시 처음의 상태로 돌아가 터미날로 부터 영어 문장을 받을 준비에 들어간다. 만약 failure 신호가 파이프 p2를 통하여 프로세스 1에 전달되면 프로세스 1은 error 신호를 사용자에게 내 보내고 사용자로 부터의 명령을 기다리게 된다.

2. 스캐너

스캐너는 입력된 문장을 리스트화 시키고 단어들의 어형을 분석한다. 즉, He had kicked balls.는 [[he, s], [have, 9, g], [kick, 5, gp], [ball, p],.]. 로 변환한다. 명사에서 s는 단수 p는 복수를 의미하고 동시의 경우에 g는 과거 gp는 과거와 과거 분사가 동일 하다는 것을 의미한다.

3. 파서

1) 파서와 파싱

문법(grammar)을 이용하여 자연언어를 처리하는 시스템은 대부분 파서를 가지고 있다.

파서는 일종의 자동 기계(automata)이며 본 시스템이 사용하는 파서와 사전의 일부를 Prolog의 grammar rule notation[7,8]을 써서 나타내면 다음과 같다.

```
sentence→subject-phrase, verb-phrase.
subject-phrase→determiner, adjective-list,
                noun-relative.
assist-phrase→determiner, adjective-list,
                noun-relative.
object-phrase→determiner, adjective-list,
                noun-relative.
agent-phrase→ determizer, adjective-list,
                noun-relative.
adjective-list→[ ].
adjective-list→ adjective-list, adjective.
noun-relative →noun, relative-clause.
relative-clause→relative, subject-phrase,
                verb-phrase.
relative-clause→[ ].
infinitive-clause→u-verb, assist-phrase.
infinitive-clause→v-verb, object-phrase.
infininitive-clause→x-verb, agent-phrase.
infinitive-clause→w-verb.
verb-phrase→u-verb, adjective.
verb-phrase→u-verb, assist-phrase.
verb-phrase→v-verb, object-phrase.
```

verb-phrase→u-verb, infinitive-clause.
verb-phrase→v-verb, infinitive-clause.
verb-phrase→w-verb, infinitive-clause.
verb-phrase→x-verb, agent-phrase.
u-verb→auxiliary-verb, n-p, be-verb.
u-verb→be-verq, n-p.
v-verb→past-participle, n-p, transitive-verb.
w-verb→auxiliary-verb, n-p, intransitive-verb.
x-verb→past-participle, n-p, be-verb,
transitive-verb.
x-verb→auxiliary-verb, n-p, be-verb,
transitive-verb.
determiner→[ ].
determiner→[a].
determiner→[the].
adjective→[happy].
noun-→[i].
noun→[book].
n-p→[not].
n-p→[ ].
be-verb→[be].
transitive-verb→[sing].
intransitive-verb→[walk].
auxiliary-verb→[can].
past-participle→[have].
relative→[that].
relative→[who].

(문법 5)

파서는 영국에서 개발된 DCG[7,8]를 이용하여 작성하였다.

파싱에는 크게 top-down과 bottom-up 두가지 형태가 있으며 DCG에 의한 파싱은 top-down 방식에 의하여 파싱을 한다. 여기서 파싱이라고 하는 것은 입력된 영어 문장 각 단어들의 품사의 결정과 이 단어들을 적절히 결합하여 직접구성성분(immediate constituent)[1]을 형성하여 궁극적으로 입력된 문장의 구절표지(phrase marker)[1]를 형성하는 것을 말한다.

2) 파싱 과정의 문제점

파싱 과정중 문제가 되는 것은 backtracking[7,8] 현상이며 이는 잘못 분류되어 결합된 단어나 직접구성성분에 의하여 발생하며 올바른 구절표지를 형성할때까지 계속된다. Backtracking에 의하여 파서는 애매한(ambiguous) 문장에 대하여 해석 가능한 모든 경우를 검사할 수가 있으며 이것에 의하여 문장의 애매성을 해결할 수 있다.

DCG에 의한 top-down 파싱은 left-recursion[14,15]에 의한 무한 루프에 빠질 염려가 있다. 이 문제를 해결함과 동시에 같은 품사가 연속으로 있는 문장, 즉 The big strong man kicks balls. 라는 문장에서 연속으로 있는 형용사 big strong을 파싱하기 위해서는 다음과 같이 프로그램을 작성하면 된다.

adjective-list→[ ].
adjective-list→adjective.
adjective-list→adjective-list, adjective.

그러나 이 프로그램은 big strong을 거꾸로 파싱을 한다. 이 현상은

adjective-list→[ ].
adjective-list→adjective, adjective-list.

으로 프로그램을 작성하면 해결된다.

3) 한글 내부 구조 생성

영어 문장과 한글 문장은 문장에 있어서 각 문장요소들의 순서가 다르다. 따라서 한글 문장에 맞도록 영어의 문장 요소들의 순서를 바꾸어 주어야 한다. 이것은 predicate head[7]의 패턴 매칭 부분에서 instantiate[7,8]되는 변수의 순서를 바꾸어 줌으로서 해결할 수 있다.

구절표지는 파싱하는 과정에서 패턴 매칭에 의하여 자연히 형성된다. 이렇게 하여 형성된 구절표지는 생성될 한글 문장의 심층 구조(deep structure)[1]를 나타낸다고 할 수 있다.

4. 발생기

1) 한글 동사 발생

파서로 부터 받은 한글 내부 구조는 한글 문장 생성에 필요한 모든 정보를 가지고 있다. 발생기는 이 정보를 이용하여 한글을 발생한다. 특히 한글 동사를 발생하는 부분은 발생기의 약 반을 차지하며 이 부분은 다음의 fact를 이용한다.

kunst(pux, h, nil, d, 1).
kunst(aux, h, nil, d, 2).
kunst(pux, h, nil, j, 3).
kunst(aux, h, nil, j, 4).
kunst(pux, h, not, d, 5).
kunst(aux, h, not, d, 6).
kunst(pux, h, not, j, 7).
kunst(aux, h, not, j, 8).

kunst(pux, g, nil, d, 9).
kunst(aux, g, nil, d, 10).
kunst(pux, g, nil, j, 11).
kunst(aux, g, nil, j, 12).
kunst(pux, g, not, d, 13).
kunst(aux, g, not, d, 14).
kunst(pux, g, not, j, 15).
kunst(aux, g, not, j, 16).

이 16가지의 fact는 미래 완료 시제를 제외한 나머지 시제에 대한 문장의 긍정 평서문, 부정 평서문, 긍정 수식문, 부정 수식문을 표시하는 최소의 것이며 pux는 완료 aux는 조동사, h는 현재 g는 과거, nil은 긍정문 not는 부정문 그리고 d는 평서문 j는 수식문을 나타낸다. pux aux h g nil not d 그리고 j는 파서가 생성하여 발생기에 넘겨주는 표시기로서 이들의 조합은 영어의 동사 부분을 한글 동사로 번역하는데 결정적인 역활을 한다. aux, g, not, j는 입력된 문장은 조동사가 포함된 과거 부정 수식문이라는 것을 나타낸다.

프로그램을 예를 들어 좀더 자세히 살펴보자.

```
Y=can, bauen(U1, V5), (Nm=2, wohl([0, 1]);
                       ⋮
                       Nm=16, wohl([0, 8])));
```

이 프로그램은 조동사가 can인 문장을 처리하는 부분인데 bauen을 통하여 어간(문법적 의미가 아니고 용언의 활용중 변하지 않는 부분)과 어미(어간에 대응하는 의미)를 결합한후 kunst에서 찾은 Nm의 값에 따라 문장의 나머지 부분을 wohl을 이용하여 출력하게된다.

동사는 다음과 같은 형태로 사전속에 있다.

tran(sing, "[부르]", "[불러]", "[불리]"), "[부르]"는 현재형이고 "[불러]"는 과거형 "[불리]"는 수동형이다.

위 bauen(U1, V5)에서 U1이 "[부르]"로 V5는 "[ㄹ]"로 instantiate[7,8] 되면 bauen(U1, V5)는 "부를"을 만든다. 만약 Nm이 aux, g, not, j에 의하여 16이되면 whol([0, 8])은 "수 없었던"을 만든다. bauen에 의하여 생성된 것과 wohl에 의하여 생성된 것을 합치면 "부를 수 없었던"이 되고 이것은 could not sing의 수식형과 동일한 의미를 나타낸다고 할 수 있다.

동사나 형용사의 활용(어미 변화)을 위한 fact에는

mo-h("[ㄴ다]", "[는]", "[지]", "[ㄹ]", "[야만]", "[서는]").
mo-p("[ㄴ다]", "[는]", "[지", "[ㄹ]", "[었다]", "[었었다]", …).
mo-g("[ㅆ다]", "[ㅆ던]", "[ㅆ었다]", "[ㅆ었던]").
ad-h(1, "[ㅏ다]", "[ㅏㄴ]", "[ㅏ고]", "[ㅐ서]", …).
ad-g(1, "[ㅐㅆ다]", "[ㅐㅆ었다]", "[ㅐㅆ었었고]").

등이 있다. mo-h는 동사의 현재형 어미, mo-p는 수동형어미 그리고 mo-g는 과거형을 나타낸다. ad-h는 형용사의 현재형어미 그리고 ad-g는 과거형 어미를 나타낸다. 형용사는 몇 가지 형태에 따라서 어미의 형태가 다르기 때문에 이 구분에 따라 네 가지 종류로 어미를 분류하였다.

위에서 언급 없이 사용한 wohl은 다음의 fact 들을 이용한다. 이 fact들은 wohl에 의하여 몇개씩 합쳐져서 의미있는 덩어리를 이룬다. 예를 들어 wohl([0, 3])은 "수 있었다"를 생성한다.

yax(0, [" ", 수, " "]).
yax(1, [있다]).
yax(2, [있는, " "]).
yax(3, [있었다]).
yax(30, [" ", 않다]).
yax(50, [" ", 된다]).

5. 예

① He is a person.
그는 사람이다.

② He is not a person.
그는 사람이 아니다.

③ He was a boy.
그는 소년이었다.

④ He was not a child.
그는 어린이가 아니었다.

⑤ He writes a story.
그는 이야기를 쓴다.

⑥ She does not write a story.
그녀는 이야기를 쓰지 않는다.

⑦ He can write a letter.
그는 편지를 쓸 수 있다.

⑧ He could not ride a horse.
그는 말을 탈 수 없었다.

⑨ He may throw balls.
그는 공들을 던질 것이다.

⑩ He might not throw balls.
그는 공들을 던지지 않을 것이었다.

⑪ You must understand the story.
당신은 그 이야기를 이해해야만 한다.

⑫ You must not kick the balls.

당신은 그 공들을 차서는 안된다.

⑬ He must be a gentleman.
그는 신사가 틀림없다.

⑭ She must not be a doctor.
그녀는 의사일리가 없다.

⑮ I like to write a letter.
나는 편지 쓰기를 좋아한다.

⑯ I like to run.
나는 달리기를 좋아한다.

⑰ He does not like to run.
그는 달리기를 좋아하지 않는다.

⑱ The song is written by me.
그 노래는 나에 의하여 쓰인다.

⑲ Her song was not written by me.
그녀의 노래는 나에 의하여 쓰이지 않았었다.

⑳ She throw the balls kicked by him.
그녀는 그에 의하여 차여진 그 공들을 던진다.

㉑ I will throw the balls which have been kicked by him.
나는 그에 의하여 차이었던 그 공들을 던질 것이다.

㉒ Many people want to make success.
많은 사람들이 성공을 하기를 원한다.

㉓ I must be happy.
나는 행복함에 틀림없다.

㉔ I am happy.
나는 행복하다.

Ⅲ. 結 論

본 논문은 Prolog를 이용하여 만들어진 영어-한글 기계 번역 시스템을 설명하였다. 이 시스템은 transfer 방법을 변형하여 구현하였으며 상이한 기능을 하는 두 개의 프로세스로 구성되어 있다.

시스템은 스캐너, 파서, 발생기, 사전으로 구성되어 있고 스캐너와 파서가 입력된 영어 문장을 이용하여 한글 문장을 생성하는데 필요한 모든 정보를 만들어 내며 발생기는 이것과 사전을 이용하여 한글 문장을 발생한다. 파서는 DCG를 이용하여 작성하였으며 발생기는 반 정도가 영어 동사를 한글 동사로 번역하는 부분이며 이 부분은 kunst와 yax fact를 이용한다.

앞으로 대명사, 관사의 처리와 문장의 의미 분석 문제에 대한 많은 연구가 요구되며 지식 베이스(know-lege-base)를 이용한 시스템 구성에 대한 연구도 요구된다.

參 考 文 獻

[1] 남기심, 이정민, 이홍배, 언어학 개론, 탑 출판사, 1983.

[2] 한성국, 한국어의 Machine Translation을 위한 구문 구조 분석, 인하 대학교 대학원 석사학위 논문, 1981.

[3] 최형석, 국어의 처리를 위한 기계 사전에 관한 연구(Ⅰ), 인하 대학교 대학원 석사 학위 논문, 1984.

[4] 문일민, 김종상, "제한된 영역인 한글 전자 우편 시스템에서의 자연언어 인터페이스에 관한 연구" 84 가을 학술 발표집, p3-15, 한국 정보 과학회, 1984년 가을.

[5] 이석호, 임해철, 김성기, "자연 한글 질의어 처리를 위한 인터페이스의 설계 및 구현", 84 가을 학술 발표집, p190-195, 한국 정보 과학회, 1984년 가을.

[6] 박상규, "기계 번역을 위한 한국어 품사의 자동 분류 방법", 석사 학위 논문, 한국과학기술원, 1984.

[7] Clocksin, W.F., Mellish, C.S., *Programming in Prolog* Springer-Verlag, 1981.

[8] Pereira, C.N., Warren, D.H., "Definite Clause Grammars for Language Analysis - A Survery of the Formalism and a Comparision with Augmented Transition Networks," *Artificial Intelligence 13* (1980).

[9] Tucker, A.B., Jr., A perspective on machine translation: theory and practice. *CACM 27, 10* (April 1984), 322-329.

[10] Barr, A., Feigenbaum, E.A., *The Handbook of Artificial Intelligence,* vol. 1, William Kaufmann Inc., 1981.

[11] Schank, R.C., *Conceptual Information Processing,* North - Holland Publishing Company, 1975.

[12] Simmons, R.F., "Semantic Networks: Their Computation and Use for Understanding English Sentences," *Computer Models of Thought and Language,* W.H. Freeman and Company, 1973.

[13] Aho, A.V., Ullman, J.D., *Principles of Compiler Design,* Addison - Wesley Publishing Company, 1978. *