인지과학, 제18권 제4호
Korean Journal of Cognitive Science
2007. Vol 18, No. 4, 457~483.

데이터베이스 의미론을 이용한 한국어 구현 시론: 수사-분류사 구조를 중심으로*

최 재 웅†
고려대학교

데이터베이스 의미론(Database Semantics, DBS)은 인간의 의사소통 방식에 대한 종합적인 이론 틀과 분석을 제공하고, 또한 이를 구체적인 컴퓨터 프로그램으로 구현하는 것을 목적으로 하고 있다. DBS의 두 가지 주요 특징으로는 문장 처리 알고리즘으로 좌연접 방식을 취한다는 점과 문장의 의미 내용을 표상하는 데이터베이스로 '어휘은행 (Word bank)를 취한다는 점을 들 수 있다. 본 연구에서는 DBS에 입각하여 한국어의 기본 현상에 대한 분석 및 구현을 시도한다. 우선 간단한 한국어 예를 통해 듣고, 추론하고, 말하는 단계가 어떻게 진행될 수 있는지를 보이고, 이어서 한국어의 특징적 현상중의 하나인 수사-분류사(classifier) 구조가 어떻게 분석되는지를 보임으로써, 영어와 독일어를 소재로 개발중인 DBS가 언어적 특성이 많이 다른 한국어 분석에도 활용될 가능성이 있음을 보인다. 또한 기존 연구에서 제시된 바 있는 좌연접 알고리즘에 대한 한국어 적용상의 문제점을 검토하면서 그에 대한 대안의 방향을 살펴보기로 한다.

주제어 : 데이터베이스 의미론, 좌연접 알고리즘, 어휘은행, 한국어, 수사, 분류사, 전산구현

---

* 본 연구에 도움을 주신 Roland Hausser, 이기용, 이민행, 홍정하 선생님께 감사드린다. 또한 심사위원 두 분의 지적사항도 논문을 다듬는데 도움이 되었다. 특히 제목의 부적합성에 대한 심사위원 한분의 지적에 맞추어 제목을 약간 수정하였다. 4절에 나오는 수사-분류사 구문 분석은 The 16th European-Japanese Conference on Information Modelling and Knowledge Bases, Trojanovice, Czech Republic, May 29, 2006 에 발표된 바 있다(Choe & Hausser 2006). 이 논문은 2005년도 고려대학교 특별연구비 지원에 의하여 연구되었음.

† 교신저자: 최재웅, 고려대학교 언어학과, 연구세부분야: 의미-화용론, 전산언어학
E-mail: jchoe@korea.ac.kr

인지적 행위주에 대한 종합적인 이해 및 구현을 구체적으로 실현하는 방법 중에 하나는 '말하는 로봇'을 만드는 일이다. 이러한 목표를 달성하기 위해서는 궁극적으로 인간의 마음, 지능, 언어 등 인지 작용에 대한 전반적인 이해가 선행되어야 할 것임은 두말할 필요가 없다. 그러나 그러한 궁극적인 목표가 물론 저절로 달성되는 것은 아니고, 이는 다시 장 단기적으로 성취 가능한 소규모 목표들로 나뉘어 접근되어야 것이라 할 수 있다.

데이터베이스 의미론(Database Semantics, DBS: Hausser 1999/2001, 2001, 2004, 2006)은 인간의 의사소통 방식에 대한 종합적인 이론 틀을 제공하고, 또한 이를 구체적인 컴퓨터 프로그램으로 구현하는 것을 목적으로 하고 있다. 인간의 언어활동을 '들어 이해하기'와 '생각하고 추론하기', 그리고 '생각을 말로 표현하기'로 크게 나눌 수 있는 것처럼, DBS에서의 구현 단위도 세 가지로 나눈다. 즉, 구체적인 문법 및 구현 단위를 '듣기'(Hear), '추론'(Think), '말하기'(Speak)의 세 가지로 나누고 있다. 이러한 세 부문은 공통적으로 데이터베이스로서의 어휘은행(Word bank)과 정보처리 알고리즘으로서의 좌연접 문법(Left Associative Grammar: LAG)을 바탕으로 하고 있다.[1)]

본 연구에서는 DBS에 입각하여 한국어의 기본 현상에 대한 분석 및 구현을 시도한다. 우선 간단한 한국어 예를 통해 듣고, 추론하고, 말하는 단계가 어떻게 진행될 수 있는지를 보이고, 이어서 한국어의 특징적 현상중의 하나인 분류사(classifier)를 포함한 명사구 표현이 어떻게 분석되는지를 보임으로써, 영어와 독일어를 소재로 개발중인 DBS가 언어적 특성이 많이 다른 한국어 분석에도 활용될 가능성이 있음을 보인다.

본 논문의 구성은 다음과 같다. 우선 2절에서 기존 연구를 간략히 검토한다. 3절에서는 인지적 행동주로서의 주요 언어활동이라 할 수 있는 듣기, 추론하기, 말

1) DBS의 주요 이론적 특징은 SLIM이란 말로 정리되는 바, 이는 아래와 같이 세 가지 이론적 특성을 뜻한다.
표층 합성적(Surface compositional): 언어 분석시 추상적인 범주의 도입을 피하고 표면구조에 최대한 충실하게 분석함
선형적(Linear): 발화의 시간적 순서에 충실하게 분석함
내적 일치(Internal Matching): 의미 해석은 표현과 맥락의 내적 일치로 설명함

하기를 각 모듈별로 간단한 한국어 문장을 예로 하여 설명한다. 4절에서는 수사-분류사 구문에 대한 세밀한 분석 및 관련 규칙, 결과 표상 등을 제시한다. 이어지는 5절에서는 좌연접 알고리즘에 대한 한국어 적용상의 문제점에 대한 이민행(1993)에서의 논의를 검토하면서 그에 대한 대안의 방향을 살펴보기로 한다. 6절은 결론이다.

기존 연구 검토

DBS의 두 가지 주요 특징으로는 문장 처리 알고리즘으로 좌연접 방식을 취한다는 점과 문장의 의미 내용을 표상하는 데이터베이스로 어휘은행를 취한다는 점을 들 수 있다. 즉, 어휘은행은 DBS에서의 의미적 표상으로 어휘 데이터베이스의 배열로만 이루어져 있으며, 문장이해 단계의 종착점이자, 추론의 바탕이고, 발화의 출발점이 된다.

본 연구가 택한 이론적 틀 속에서 이루어진 한국어에 대한 연구로는, 좌연접 알고리즘을 활용한 여러 연구가 있다. 특히 이기용(1999a)은 좌연접 알고리즘을 이용하여 한국어 형태소 분석에 대한 비교적 체계적이고 종합적인 연구를 시도하였고 이에 대한 구현까지도 성공적으로 보인 바 있다. 좌연접 문법을 이용한 한국어 통사적 분석도 일부 시도되었는 바, 신경구(1992), 이민행(1993), 이기용(1999b), 홍정하·최승철·이기용(2000) 등의 연구를 들 수 있다. 그러나 이러한 연구들은 좌연접 알고리즘을 이용하고 있는 반면에 분석의 결과를 주로 핵어 구 구조 문법(Head-driven Phrase Structure Grammar) 방식으로 제시하고 있다는 점에서, DBS와 차별화 된다. 또한 이러한 방식은 '연속의 원리 (Principle of Continuation)'에 기반한 좌연접 문법의 기본 정신에 충실하다고 보기 어렵다(5절 참조). 이에 반해 DBS에서는 의미적 표상이 어휘 데이터베이스의 배열로만 이루어져 있다. 즉 기존 대다수 문법이론이 취하는 '대체의 원리(Principle of Substitution)'보다는 인간의 언어처리에 직관적으로 더 부합하는 '연속의 원리'를 철저하게 추구하여 언어 처리의 전 과정에 최소한의 구조를 상정한다. 기존의 구조적인 정보는 데이터베이스 내 요소들 사이의 연결관계로만 파악한다. 기존 이론에서 전제로 하고 있는 언어의 계층화된

구조는 본질적이라기보다는 어휘 데이터베이스로부터 도출되는 개념으로 파악될 수 있다. 다시 말해서, 문장의 계층적 구조에 절대적으로 의존하지 않고도 문장의 분석 및 이해가 가능하다고 본다.

'데이터베이스 의미론'은 국내에서도 동일 이름으로 연구가 이루어진 바 있으나 (이기용 1999b, 2000, 이기용·홍정하·이종민 2002, 홍정하·최승철·이기용 2000), 주로 좌연접 알고리즘을 이용했을 뿐 어휘은행을 활용하지 못했고, 또 대부분은 본격적인 구현에까지 이르지는 못한 것으로 판단된다.

듣기, 추론하기, 말하기

본 절에서는 아래와 같은 간단한 한국어 문장을 자료로 하여 그 문장의 듣기, 추론하기, 말하기 과정이 DBS내에서 어떻게 이루어지는지를 보인다.

(1) 새 옷이 예쁘다.

즉, (1)번 문장을 예로하여, 주어진 문장이 어떤 절차로 분석되어 어떤 형태로 저장이 되며, 그렇게 저장된 형태가 어떤 절차로 활성화되고, 이어 적합한 표면형으로 발현될 수 있는지를 보이기로 한다.

듣기

(1)이 분석되는 절차는 좌연접 알고리즘에 따라 아래와 같다.

(2) a. 새 + 옷이
b. (새 + 옷이) + 예쁘다

위에 제시된 각 단계별로 주어진 표현간의 결합은 그것을 뒷받침하는 규칙이 있어야 가능하다. 우선, 좌연접 규칙은 모두 아래와 같은 형식을 취하고 있다.

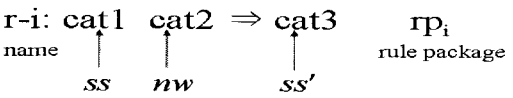

표 1. 좌연접 규칙 도식

<표 1>[2]의 규칙 도식을 보면 모든 규칙은 우선 이름(r-i)을 부여받고, 각 규칙은 그 규칙에 이어서 적용될 수 있는 규칙 패키지(rpi)가 지정된다. 규칙의 입력 조건은 두 개의 범주(cat1, cat2)로 규정된다. 그리고 그 두개의 범주는 주어진 입력 범주인 '문장 시작 (sentence start: ss)' 및 '다음 어휘 (next word: nw)'와 각각 일치하여야 한다. 입력 범주가 규칙의 입력 조건(cat1, cat2)에 부합되면 규칙에 명시된 바에 따라 연산을 거친 뒤에 다음 규칙 적용을 위한 새로운 '문장 시작 ss″'을 도출하게 된다.

위의 규칙 도식에 따라 주어진 표현 (2a)에 적용되는 규칙을 만들어 보면 아래와 같다.

ADN+NN {NP+FV}

adj: α	noun: β	nw-noun -|e|-> ss-mdd
cat: adn	cat: n	ss-adj -|a|-> nw-mdr
mdd:	mdr:	$copy_{ss}$ $copy_{nw}$

표 2. ADN+NN 듣기 규칙

위 규칙은, 규칙 이름(ADN+NN)과 이어지는 규칙 패키지 {NP+FV}를 명시하고 있고, 이어서 입력 범주에 대한 규정 및 연산 내용을 담고 있다. 입력 표현이 위 규칙을 충족할 경우, 명시된 연산이 적용되어 새로운 범주, 즉 <표 1>의 규칙 도식에서 cat3에 해당하는 것이 만들어 지게 되고 이는 곧 다음 규칙을 위한 '문장 시작 ss″'이 된다. ADN+NN 규칙에서는, 규칙 도식에서의 성공적인 규칙 적용의 결과를 나타내는 범주 c3대신, 입력 범주에 적용되는 연산을 명시하고 있다. 그러한 연산이 성공적으로 적용된 결과가 cat3에 해당하는 ss′이 된다. <표 3>의 도출

2) 엄격히 말하면 <표>가 아니나 문서편집 및 상호참조의 편의상 본 논문에서는 규칙, 도출 등을 모두 <표>라 칭하여 일련번호를 붙이기로 한다.

과정은 바로 (2a)에 ADN+NN 규칙이 적용되는 단계를 보여준다.

<표 3>의 도출에서, 7번 줄에 나오는 것이 입력 표현이다.[3] 8-14번의 정보는 사전부로부터 제공받는다. 즉, 사전에 명시된 "새"와 "옷이"의 정보라 할 수 있다. 2-6번 줄에 나오는 내용은 바로 ADN+NN 규칙에 명시된 바와 같다. "새"와 "옷이"의 입력 범주가 각기 ADN+NN 규칙의 cat1과 cat2에 부합되므로,[4] 그 규칙에 명시된 연산을 거쳐 다음 규칙 적용을 위한 '문장 시작 ss'이 도출되고, 거기에 다음 범주(nw)인 "예쁘다"가 더해져서 다음과 같은 새로운 규칙 입력 범주가 생성된

```
1.   2.1
2.   ADN+NN: [adj: alpha]   [noun: beta ]    ss_adj-|a|->nw_mdr
     rp:(NP+FV)
3.           [cat: adn  ]   [cat: n      ]    nw_noun-|e|->ss_mdd
4.           [mdd:      ]   [mdr:       ]    increase ID
5.                                           copy_ss
6.                                           copy_nw

7.           [sur: 새   ]   [sur: 옷이    ]
8.           [adj: new  ]   [noun: dress]
9.           [cat: ADN  ]   [cat: NOM   ]
10.          [sem:      ]   [sem:       ]
11.          [mdd:      ]   [mdr:       ]
12.          [idy: +1   ]   [fnc:       ]
13.          [prn:      ]   [idy: +1    ]
14.                         [prn:       ]
```

표 3. 도출 1-a

```
1.       [sur:        ]   [sur: 예쁘다   ]
2.       [noun: dress]    [verb: pretty ]
3.       [cat: NOM    ]   [cat: NP' V   ]
4.       [sem:        ]   [sem: pres    ]
5.       [mdr: new    ]   [mdr:         ]
6.       [fnc:        ]   [arg:         ]
7.       [idy: +1     ]   [ctn:         ]
8.       [prn: 2      ]   [ctp:         ]
9.       [wrdn: 2     ]   [prn:         ]
```

표 4. 도출 1-b

3) 규칙 적용 내용은 A. Kycia가 Java를 이용하여 DBS구현 도구로 개발한 jDBS (2003)를 근간으로 하여 한국어 문법 패키지를 적용하여 얻은 것이다. 도출 과정으로 제시된 <표 2>는 본래의 프로그램 결과 출력에다가 참고의 편의상 왼쪽 번호를 더해준 것이며, 1번 줄에 나오는 번호 2.1은 도출 단계를 나타낸다. 본 논문에서의 구현은 형태소 분석과정을 거치지 않는 어절 형태(full form)를 기반으로 하여 통사적 결합을 보이고 있다.

4) 3번 줄의 [cat: n]과 9번 줄의 [cat: NOM]은 별도로 제약으로 일치가 가능하다.

```
1   ### Parsing finished successfully.
2   Proplets Added to Wordbank.
3
4   [sur: 새    ] [sur: 옷이   ] [sur: 예쁘다 ]
5   [adj: new   ] [noun: dress] [verb: pretty]
6   [cat: ADN   ] [cat: NOM    ] [cat: DECL    ]
7   [sem:       ] [sem:        ] [sem: pres    ]
8   [mdd: dress] [mdr: new    ] [mdr:         ]
9   [idy: 2     ] [fnc: pretty] [arg: dress   ]
10  [prn: 2     ] [idy: 3      ] [ctn:         ]
11  [wrdn: 1    ] [prn: 2      ] [ctp:         ]
12                  [wrdn: 2    ] [prn: 2       ]
13                                  [wrdn: 3     ]
```

표 5. 도출 1-c

다.[5] 이는 곧 (2b)의 단계에 해당된다 (<표 4>).

이러한 입력 자료에 적합한 규칙이 적용되고 이어서 다른 부수적인 규칙들이 적용되면서 주어진 문장 "새 옷이 예쁘다"가 분석되면 <표 5>에서와 같은 정보가 어휘은행에 추가된다.

지금까지 '범주'라는 말로 지칭해온 대상들은 단순 자질구조 행렬로 되어 있다. 규칙에서의 입력 범주, 사전에서의 어휘정보, 그리고 규칙 적용의 결과까지도 모두 단순 자질구조로 되어 있다. DBS에서는 이처럼 범주 정보를 나타내는 행렬을 '명제소 (proplet)'라 칭한다.[6] 명제소는 복합구조를 허용하지 않는 대신, 명제소끼리 서로 연결관계를 형성한다. 즉, 명제소는 크게 두 가지로 나눌 수 있는데, 하나는 다른 명제와 연결되지 않은 '고립 명제소 (isolated proplet)'로서 예를 들어 도출 1-a의

5) 어휘 "새"의 범주는 다음 규칙에 관여하지 않으므로 별도로 저장되어 있다가 최종 결과물에 합해지게 된다.

6) DBS가 본격 제안된 Hausser(1999/2001)의 한국어 번역본(장석진 외, 2002)에서는 proplet의 번역어로 주로 '명제'란 용어를 사용하고 있다. 그러나 '명제'는 proposition의 번역용어로 이미 확립이 되어 있으므로 혼선을 야기한다고 판단되어 '명제소'라는 새로운 용어를 도입하기로 한다. 예를 들어 "동일한 명제에 속하는 명제는 공통의 명제 번호..."(장석진 외, 2002: 407)라는 번역 설명에서 첫 번째와 세 번째 '명제'는 proposition을, 두 번째 '명제'는 proplet을 지칭한다. 본래 '명제소' proplet은 droplet '물방울'을 유추해서 만든 것으로 단순 명제의 기본 부분을 일컬으며, 어휘중에서 내용어(noun, verb, adjective)에 해당하는 정보를 담고 있는 요소로, 데이터베이스의 레코드 (record)에 해당하는 개념이다. 기능어(determiner 등)는 일반적으로 독자적인 명제소를 구성하지 못하고 다른 명제소의 일부로 흡수된다. 자세한 내용은 Hausser(2006) 참고..

```
1. kdbs1.DBS1-HEAR> -N
2. kdbs1.DBS1-THINK> pretty
3. # Think mode #
```

표 6. 도출 2-a

7-14번째 줄에 나오는 두 개의 명제소는 서로 연결되어 있지 않다. 다른 하나는 '연결된 명제소 (connected proplet)'로서 명제소 내부의 일부 값이 다른 명제소와 연결되어 있는 경우를 의미한다. 예를 들어, 도출 1-c를 보면 이제 명제소 [new][7]와 명제소 [dress]는 서로 연결된 값을 지니고 있다. 명제소 [new]의 속성 mdd는 명제소 [dress]의 속성 noun과 서로 동일한 값 dress를 공유하고 있다. 반면 명제소 [dress]의 속성 mdr과 명제소 [new]의 속성 adj는 new라는 값을 공유하고 있다. 마찬가지로 명제소 [dress]와 명제소 [pretty]도 서로 간에 동일한 값으로 연결된 부분들이 있다. 또한 명제소 [new], [dress], [pretty]는 모두 속성 prn값을 공유하고 있다. 이 점이 바로 세 명제소가 하나의 명제를 이루고 있음을 나타낸다.

지금까지 간단한 한국어 문장 "새 옷이 예쁘다"를 예로 하여 DBS의 주요 개념, 규칙 도식, 규칙 적용, 주요 도출과정, 최종 도출 결과 등을 설명하였다. 일단 이러한 방식으로 일련의 문장이나 발화가 어휘은행에 '연결된 명제소'로 저장이 되었다고 가정을 한다면, 그 다음 절차는, 그렇게 저장된 정보가 주어진 자극에 따라 어떻게 활성화 되며 최종적으로 문장으로 발화되는가가 될 것이다. 다음 소절에서는 '추론하기'의 한 과정으로 주어진 자극에 따라 기억된 정보를 항해하면서 필요한 명제소를 '활성화'하는 규칙부를 설명하고, 이어지는 소절에서는 '활성화'된 정보를 문장으로 발화하는 부분을 담당하는 '말하기' 규칙부를 설명하기로 한다.

추론하기

어휘은행에 저장된 정보를 바탕으로 하는 추론 또는 생각의 절차는[8] 우선 추론

7) 명제소 [new]는 [adj: new]라는 속성-값을 취하는 자질구조로 "새"에 해당하는 명제소를 뜻한다. 앞으로도 마찬가지 방식으로 명제소를 구분하여 부르기로 한다.

8) 여기서 말하는 '추론'은 어휘은행 내를 '항해'하는 행위를 포괄하는 것으로, 현재로는 단

V_N_N {N_A_V}

[verb: β arg: # α y prn: m]	[noun: α mdr: y z fnc: β prn: m]	mark α in ss-arg output position nw activate LA-speak

표 7. 추론 규칙 V_N_N

을 위한 적절한 자극을 전제로 한다. DBS에서는 술어를 출발점으로 삼는다.[9]

어휘은행에 저장된 명제소 중에서 선택한 술어 명제소를 시작점으로 하여 그와 연결된 명제소들을 찾아 '항해'하기 위해서는 적절한 항해 알고리즘이 필요하다. 이 역시 앞에서와 마찬가지로 좌연접 규칙 도식에 따라 한국어를 위한 규칙을 <표 7>처럼 만들어 볼 수 있다.[10]

위 규칙의 이름 V_N_N은 술어 명제소를 시작점으로 하여 이와 연결된 명사 명제소를 찾아 이동한 다음 그 명제소에 머무른다는 것을 나타낸다.[11] 위 규칙을 살펴보면 어휘은행내에서 규칙에 주어진 조건인 동사 명제소 [α]와 명사 명제소 [β]의 관계를 만족하는 명제소들을 찾아 나가는 것으로 되어 있다. 두 범주가 동일한 속성-값인 [prn: m]을 지녀야 한다는 점은 어휘은행 명제소들 중 이를 만족하는 연결된 명제소들이 있어야 한다는 것을 의미한다. 이와 같은 규칙이 적용되는 예는

순히 명제소들을 연결해서 문장을 형성해 가는 '검색' 과정을 담당하고 있으나, 추후 여러 논리적 추론도 이 모듈에서 구현이 가능할 것으로 기대된다.

9) 문장의 핵은 술어라는 일반적인 관점을 수용한 것이나, 추론의 출발점이 술어가 반드시 되어야할 이유가 무엇인지는 더 검토가 필요하다고 본다. 예를 들어, 주제에 해당되는 어떤 명사가 '생각하기'의 출발점이란 가정도 충분히 설득력이 있기 때문이다. 그러나 여기에서는 술어 명제소를 출발점으로 삼는 입장을 따른다.

10) 표면형을 다루는 듣기 규칙이나 말하기 규칙과는 달리 추론/생각하기 규칙은 언어적 개별성에 덜 좌우된다. 따라서 이 규칙은 다른 규칙과는 달리 여러 언어에 공통으로 적용될 수 있다.

11) 규칙 내 마지막 연산 activate LA-speak는, 진행된 부분만큼을 발화하기 위한 것으로 특히 '말하기' 모듈에서 활용되는 부분이다. 이는 추론/생각하기와 말하기는 한쪽이 완결된 다음 다른 쪽이 진행되기 보다는, 수시로 교대로 일어난다는 가정을 담고 있다. 다시 말해 어휘은행내의 명제소를 찾아다니는 과정에 발화가 가능한 부분이 일부라도 완결되면 그만큼이라도 바로 발화로 구현되도록 한다는 뜻이다. 다음 소절 논의 참고.

```
1.      2.
2.      V_N_N: [verb: alpha ]   [noun: beta ]
        output pos: nw          rp:{N_A_V}
3.             [arg: beta   ]   [fnc: alpha ]
4.             [prn: i      ]   [prn: i     ]

5.             [sur:        ]   [sur:       ]
6.             [verb: pretty]   [noun: dress]
7.             [cat: DECL   ]   [cat: NOM   ]
8.             [sem: pres   ]   [sem:       ]
9.             [mdr:        ]   [mdr: new   ]
10.            [arg: dress  ]   [fnc: pretty]
11.            [ctn:        ]   [idy: 3     ]
12.            [ctp:        ]   [prn: 2     ]
13.            [prn: 2      ]   [wrdn: 2    ]
14.            [wrdn: 3     ]
```

표 8. 도출 2-b

<표 8>과 같다.

주어진 자극 'pretty'를 받아들이면 명제소 [pretty]를 시작점으로 하여 어휘은행 내에서 이와 연결된 명제소 [dress]를 찾아 나가는 과정을 위에 주어진 규칙 V_N_N이 담당하게 된다. 그리고 거기까지 찾은 결과를 '말하기' 모듈로 보내서 혹시 발화가 가능하다면 그만큼 발화를 하고 아직 부족하다면 다시 '추론'모듈로 돌아와서는 어휘은행내의 나머지 '항해'를 계속하게 된다. 일단 모든 추론이 완료된다면 그 결과는 <표 9>처럼 나오게 된다.

```
1.    #####  Navigation finished
2.    # Navigation endstate OK.

3.    Activated sequence of proplets:

4.    [sur:         ] [sur:        ] [sur:       ]
5.    [verb: pretty] [noun: dress] [adj: new   ]
6.    [cat: DECL    ] [cat: NOM    ] [cat: ADN   ]
7.    [sem: pres    ] [sem:        ] [sem:       ]
8.    [mdr:         ] [mdr: new    ] [mdd: dress]
9.    [arg: dress   ] [fnc: pretty] [idy: 2     ]
10.   [ctn:         ] [idy: 3      ] [prn: 2     ]
11.   [ctp:         ] [prn: 2      ] [wrdn: 1    ]
12.   [prn: 2       ] [wrdn: 2     ] [trc: 10    ]
13.   [wrdn: 3      ] [trc: 9      ]
14.   [trc: 7       ]
```

표 9. 도출 2-c

위의 결과를 보면, 명제소 [pretty]를 출발점으로 하여 그와 연결된 명제소 [dress]를 찾아 내었고, 이어서 [dress]와 연결된 명제소 [new]를 찾아 내었다. 그리고 이러한 '항해'의 결과, 즉 추론 결과는 활성화된 정보로 임시 저장이 된다. 물론 이 정

보는 '말하기' 모듈에서 이어받는다.

말하기

말하기 모듈의 역할은 추론 모듈에서 활성화된 정보를 실제 표면형으로 바꾸어 주는 일이다. 즉 <도출>이나 <도출>에 나온 정보를 적절한 한국어로 바꾸어 주는 것을 의미한다. 이러한 변환 역시 좌연접 알고리즘을 적용하므로 앞에서 본 도식과 동일한 방식으로 규칙이 구성된다. 이번에도 그러한 규칙중 하나를 <표 10>에 제시한다.

_NN {_FVERB}

verb: β arg: # α y prn: m	noun: α mdr: z y fnc: β prn: m	where z y =NIL or y = δ## lex-nn [noun: α] mark α in [noun: α] if #x and y ≠NIL activate LA-think

표 10. 말하기 규칙 NN

말하기 규칙 _NN은 명사구를 표면에 드러내는 역할을 한다. 앞에서 말했듯이 추론 규칙과 말하기 규칙은 필요에 따라 상호 주고받기가 가능하다. 위 규칙에서 보듯 주어진 조건이 충족되지 않으면 바로 LA-think를 활성화하도록 되어 있다. <표 11>은 이 규칙이 적용된 도출 단계다.

위의 도출은 주어인 '옷이'가 도출되는 단계를 보인다. 즉 규칙에 주어진 조건이 충족되므로 규칙의 연산(lex-nn[noun: α])에 따라 16번 줄의 "옷이"가 표면에 나오게 된다. 명제소로 결과가 나오는 듣기나 추론 모듈과는 달리 말하기 모듈의 최종 결과는 추론하기의 결과로 활성화된 정보가 구체적인 문장으로 모습을 드러낸다.

그러한 절차를 모두 거친 후 <표 12>에서 보듯 말하기의 결과가 표면에 도출된다. 문장 시작을 관형사나 명사로 시작할 수 있도록 하였으므로, 결과는 두 가지로 구현된다.

지금까지 DBS에서 추구하는 '말하는 로봇'의 구체적인 구현 방법을 간단한 한국어 문장을 예로 하여 설명하였다. 듣기, 추론하기, 말하기 각 모듈별로 각각 2

```
1.  6.
2.  -NN: [verb: beta  ]  [noun: alpha]
    lex-nn        rp:{-FVERB}
3.        [prn: i       ]  [cat: n      ]
4.                           [idy: j      ]
5.                           [prn: i      ]

6.        [sur:         ]  [sur:        ]
7.        [verb: pretty]   [noun: dress]
8.        [cat: DECL    ]  [cat: NOM    ]
9.        [sem: pres    ]  [sem:        ]
10.       [mdr:         ]  [mdr: new    ]
11.       [arg: dress   ]  [fnc: pretty]
12.       [ctn:         ]  [idy: 3      ]
13.       [ctp:         ]  [prn: 2      ]
14.       [prn: 2       ]  [wrdn: 2     ]
15.       [wrdn: 3      ]  [trc: 19     ]

16. [Speak:] 옷이
```

표 11. 도출 3-a

개,[12] 4개, 3개의 규칙으로 필요한 도출을 모두 이루어 낼 수 있었다. 즉 DBS는 '말하기 로봇'이 단지 은유가 아니라 실제로 구현될 수 있다는 점을 구체적으로 보여주고 있다는 점에서 흥미롭다. DBS에서 채택한 좌연접 알고리즘은 인간의 문장 이해 및 발화 방식과 잘 부합되고, 어휘은행은 인간의 기억을 모사하고 있다. 이러한 좌연접 분석 방식과 더불어 명제소를 바탕으로 한 의미의 표상방식은 매우 엄격하면서도 단순한 규칙 및 표상 체계에 기반을 두고 있다는 점이 돋보인다.

이 절에서는 간단한 한국어 표현 및 의미표상을 중심으로 듣기-추론하기-말하기 모듈이 어떻게 구성되고 작동되는지를 보였다. 한 가지 강조할 점은, 듣기-추론하기-말하기 규칙들이 모두 철저한 좌연접 방식으로 되어있다는 점이다. 서로 간 구분하기 위해서 규칙 이름(label)은 다른 방식으로 붙이고 있으나 규칙의 형식은

```
1.  ## LA-SPEAK End state OK.
2.  LA-Speak Produced Wordforms:

3.   옷이 예쁘다 .
4.   새 옷이 예쁘다 .

5.  kdbs1.DBS1-SPEAK>
```

표 12. 도출 3-b

12) 듣기 규칙에는 소위 시스템관리차원(house-keeping)의 규칙이 두개 더 필요하였다.

<표 1>에 제시된 좌연접 규칙 도식을 따르고 있다.

이어지는 논의에서는 이러한 분석방식을 좀 더 확대하여 한국어의 특징을 나타내는 현상을 하나 선택하여 다루도록 한다. 아울러 추론하기와 말하기는 듣기의 연장선상에서 처리가 가능하다는 점을 보였으므로, '듣기 규칙'으로 논의를 한정하기로 한다.

수사 및 분류사

분류사는 한국어의 주요 현상중의 하나로, 수사와 결합하여 관련 명사의 수량명사를 한정하거나 단위를 명시해주는 역할을 한다. 또한 한국어의 주요한 특성인 자유 어순과 생략 가능성도 제한된 범위내에서나마 잘 보여준다. 예를 들어, 영어의 "A girl came"이란 표현이 영어에서는 "One (girl) came"정도의 변이형만 있는데 반해서 한국어에서는 모두 20가지가 가능하다.[13] 따라서 한국어 주요 특성을 반영한 이러한 제한된 범위의 현상을 각 문법에서 어떻게 다룰 수 있는가를 보이는 것은 어떤 문법에서이든 시도할만하다고 본다. 특히 좌연접 알고리즘을 취하는 DBS에서는 이러한 다양성이 과연 제대로 통제될 수 있는지, 현상의 다양성만큼이나 규칙도 여러 개가 되어야 하는 것은 아닌지라는 의문이 들 수 있다.

본 절에서는 수사 및 분류사가 관련 명사 뒤에 나오는 경우와 명사 앞에 나오는 경우로 나누어 여러 경우들을 살펴보기로 한다.

수사-분류사 후치 구조

아마도 가장 전형적인 분류사 결합 방식은 다음과 같은 구조가 될 것이다.

(4) 소녀가 한 명 (잔다.)

13) 여기에 주어가 완전히 생략되거나, 또는 명사구 일부가 문장의 다른 요소를 건너가는 것으로 보이는 양화사 유동까지 포함한다면 변이형의 가짓수는 더욱 늘어난다. 이 논문에서는 그러한 경우는 논외로 한다.

그런데 (4)는 조사의 분포에 따라 여러 변이형이 가능하다.

(5) a. 소녀 한 명이
b. 소녀가 한 명이
c. 소녀 한 명

또한 분류사도 선택적이다. (5)와 달리 아래는 분류사가 들어있지 않는 표현들이다.

(6) a. 소녀가 하나
b. 소녀가 하나가
c. 소녀 하나가
d. 소녀 하나

(4)-(6)의 예들은 조사와 분류사의 선택 가능성으로 인한 표면형의 다양성을 보여준다. 이러한 다양한 표현들을 분석하기 위해서 DBS의 관점에서는 크게 두 가지가 문제가 된다. 하나는 각 표현별 최종 의미 표상이 어떻게 되느냐 하는 문제이고, 다른 하나는 그러한 의미표상을 도출하기 위한 좌연접 규칙이 무엇인가라는 점이다. 내용어를 기준으로 하여 어휘은행을 형성하는 DBS의 방식을 따른다면 명사인 '소녀'가 독자적인 명제소를 구성한다는 데는 의문의 여지가 없다. 반면 수사와 분류사는 기본적으로 선행하는 명사의 의미를 더 구체화하는 기능을 가지고 있다는 점에서 독자적인 명제소를 구성한다기 보다는 명사 명제소의 자질 값으로 수렴될 수 있다. 분류사는 선행명사의 단위화를 촉발하고, 수사는 그 단위의 수량을 나타낸다. 의미표상과 관련한 또 한 가지 문제는 조사의 분포에 따른 의미 차이를 어떻게 반영할 것인가 하는 문제이다. 이 정보 역시 명사의 자질 값으로 표현될 수 있다. 다만, 표면형을 충실히 반영하는 DBS의 정신에 따라 표면에 조사가 있는 경우만 그러한 격 정보가 명사에 표시되도록 하면 될 것이다.

한편 표면형으로부터 의미표상으로의 변환에는 3절에서와 마찬가지로 철저한 좌연접 알고리즘에 의존하게 된다. (4)와 (5)에서의 명사-수사-분류사 구성을 위해서는 [명사-수사], [수사-분류사]에 작용하는 규칙이 각각 필요할 것이다. 이를 좀 더

세밀히 살펴보기로 한다. 우선 '소녀가', '한', '명'에 대한 각각의 어휘정보는 <표 13>과 같다.

```
1 [  sur: 소녀가,          [  sur: 한,          [  sur: 명,
2    noun: girl,           noun: @1,            noun: @1,
3    cat: (n),             cat: (cl' n' n)      cat: (cl)
4    case: (nom)           case: ()             case: ()
5    sem: (),              sem: (one),          sem: (individual),
6    mdr: (),              mdr: ()              fnc: nil
7    fnc: @2               fnc: nil             ]
8 ]                        ]
```

표 13. 어휘부 정보: .'소녀가'. '한'. '명'

위와 같은 어휘부에서의 정보를 바탕으로 이제 (4)-(5)에 제시된 네 가지 표현들을 도출하기 위해서는 두 개의 규칙이 필요하다.[14] 첫 번째는 명사와 이어지는 수사를 결합하는 규칙이다.

```
1 RULE_NOUN_NUM (RULE_NUM_CL RULE_NOUN_FV)
2 [  noun: SS1
3    cat: (NUM' CL' NP)
4    case: (CS1)
5    sem: SSL1
6 ]
7 [  noun: NW1
8    cat: (NUM)
9    sem: NWL1
10 ]
11 acopy(NWL1 SSL1)
12 cancel(NUM')
```

표 14. NOUN NUM 규칙

위의 규칙에 따르면, 입력범주가 충족될 경우 '다음 어휘'의 sem 값을 '문장 시작'의 sem 값에 더해주고, 이어서 '문장 시작'의 cat 자질의 첫 번째 값인 NUM'을 삭제하게 된다. NUM'을 삭제하는 이유는 '다음 어휘'가 NUM이라서 '문장 시작'의 cat의 값이 충족되기 때문이다.

14) 4절에 나오는 어휘부, 규칙, 도출 등은 모두 실제 구현된 경우엔 구현된 형태로 제시하도록 한다. 다만 3절에서와 마찬가지로 참고의 편의상 왼쪽에 일련 번호를 붙였다. 4절에서의 구현은 J. Handle이 새롭게 개발한 구현도구 JSLIM을 활용하였다.

도출에 필요한 또 다른 규칙은 NOUN_NUM에 이어 적용될 규칙으로 NUM_CL이 있다 (<표 15>).

```
1 RULE_NUM_CL {RULE_NOUN_FV RULE_CL_NOUN}
2 [  noun: SS1
3    cat: (CL' NP)
4    sem: SSL1
5 ]
6 [  noun: NW1
7    cat: (X)
8    case: (Y)
9    sem: NWL1
10 ]
11 acopy(NWL1 SSL1)
12 cancel(CL')
```

표 15. 규칙 NUM CL

규칙 NUM_CL도 앞서 나온 규칙 NOUN_NUM과 마찬가지로 '다음 어휘'에서 sem자질 값을 '문장 시작'으로 더해주는 것과 '문장 시작'의 cat자질 값에 남아있는 CL'을 삭제하게 된다. 실제로 규칙 NUM_CL이 적용되는 범주는 "한 명"에 상응하는 범주라기보다는 "소녀가 명"에 상응하는 범주가 된다. 그런데 현재의 구현 시스템에서는 '문장 시작'은 자동으로 저장이 되나 '다음 어휘'는 구체적인 명령이 없이는 저장되지 않도록 되어있다.[15] 따라서 규칙 NOUN_NUM과 NUM_CL을 적용하게 되면 "한"과 "명"에 상응하는 명제소의 어휘적 정보는 모두 앞에 나오는 명사 "소녀가"로 통합되지만 저장이 되지는 않으므로 결국 "소녀가"에 상응하는 명제소만 남게 된다. 이러한 규칙과 어휘 정보를 바탕으로 (4)를 도출하면 <표 16>과 같은 결과가 나오게 된다.[16]

위의 도출 결과를 보면 "한"과 "명"의 어휘 정보가 "소녀가"에 흡수되어 각각 (one)과 (individual)이라는 sem자질에 추가된 정보로 남게 되었다. (5a)-(5c)를 도출하기 위해서는 어휘부에 "소녀"와 "명이"를 추가하되 이미 주어진 "소녀가"와 "명"의

15) '문장 시작'과 '다음 어휘'를 선택적으로 저장할 수 있게 시스템을 바꾼다면 규칙을 좀 더 간결하게 할 수도 있을 것이다.

16) 물론 제대로 된 도출을 위해서는 NOUN_NUM과 NUM_CL 외에도 규칙이 더 필요하다. 규칙 패키지를 보면 이어지는 동사와 결합하기 위한 규칙 NOUN_FV와 문장 종결 기호를 처리할 규칙 FV_PM이 등이 있으나, 여기에는 제시하지 않기로 한다.

```
+++ Final State +++

[sur : 소녀가                ]  [sur : 잔다      ]
[noun: girl                  ]  [verb: sleep     ]
[cat : (n)                   ]  [cat : (v)       ]
[case: (nom)                 ]  [case: ()        ]
[sem : ((one) (individual))]  [sem : (present)]
[mdr : ()                    ]  [arg : (girl)    ]
[fnc : sleep                 ]  [ctn : (0 nil)   ]
[wrd : 1                     ]  [wrd : 4         ]
[prn : (1)                   ]  [prn : (1)       ]
```

표 16. 도출 4

정보에다가 각기 case값만 적절히 표시해 주면 된다. 그러나 규칙은 위에 주어진 두 규칙으로 충분하다.[17]

이번에는 명사 뒤에 수사만 나오고 분류사가 없는 구문 (6)을 살펴보자. (6)을 도출하기 위해서는 어휘부에는 '하나'에 대한 정보도 필요하다. 이는 위에 주어진 '한'과 마찬가지 형태가 된다. 그러면 새로운 규칙이 필요한가? 그럴 필요가 없다. 왜냐하면 규칙 NOUN_NUM에 이미 반영되어있기 때문이다. 그 규칙에 이어지는 규칙으로 NOUN_FV가 바로 명사-수사-동사로 이어지는 표현을 다룰 수 있게 해 준다. 예를 들어 (6c)인 "소녀가 하나가 잔다"는 <표 17>에서처럼 분석이 된다. (4)의 분석결과와 비교해 보면 "소녀가" 명제소의 sem자질 값에 (individual)의 유무의

```
+++ Final State +++

[sur : 소녀가 ]  [sur : 잔다      ]
[noun: girl   ]  [verb: sleep     ]
[cat : (n)    ]  [cat : (v)       ]
[case: (nom)  ]  [case: ()        ]
[sem : ((one))]  [sem : (present)]
[mdr : ()     ]  [arg : (girl)    ]
[fnc : sleep  ]  [ctn : (0 nil)   ]
[wrd : 1      ]  [wrd : 3         ]
[prn : (1)    ]  [prn : (1)       ]
```

표 17. 도출 5

17) 시스템상 어휘부에 형태소 분석이 추가될 경우 간소화될 수 있다. 영어와 독일어의 경우엔 동일 시스템에 형태소 분석도 구현이 되어 있으나 한국어의 경우엔 아직 통합 시스템이 만들어지지 않았다. 그러나 좌연접 규칙을 이용한 한국어 형태소 분석은 이미 별도로 제시된 바 있다 (이기용 1999a).

```
RULE_CL_NOUN (RULE_NOUN_FV)
[  noun: SS1
   cat: (CL)
   sem: SSL1
]
[  noun: NW1
   cat: (X NP)
   case: (CS2)
]
ecopy(NW.sur SS.sur)
ecopy(NW.noun SS.noun)
cancel(X)
ecopy(NW.cat SS.cat)
ecopy(NW.case SS.case)
ecopy(NW.sem SS.sem)
ecopy(NW.fnc SS.fnc)
```

표 18. 규칙 CL_NOUN

차이만 있다. 앞에서 보였듯이 (4)에는 분류사 "명"의 존재로 인해 (individual)값이 추가되었다.

이번에는 수사와 분류사가 명사 앞에 나오는 경우들을 살펴보기로 한다.

수사-분류사 전치 구조

수사와 분류사가 명사 앞에 나오는 표현은 다음과 같다.

(7) 한 명의 소녀가

(7)에서 분류사 "명"과 조사 "의", "가"가 선택적이라는 점을 고려하면 아래 표시된 것과 같이 총 여섯 가지의 가능성이 나온다.

(8) 한 (명(의)) 소녀(가)

(8)의 어떤 표현도 기본적인 의미에서는 명사 후치 구조의 표현들과 다르지 않다고 본다면, 의미표상도 같게 나와야 할 것이다. (8)의 여러 표현을 다루기 위한 좌연접 규칙으로는 [수사]-[분류사] 연쇄와 [분류사]-[명사] 연쇄를 다룰 규칙이 각

각 필요할 것이다. 이 중에서 [수사]-[분류사]를 다루기 위한 규칙은 수사-분류사 후치 구조 표현들을 다루기 위해 도입된 규칙 NUM_CL과 크게 보아 중복되므로 같은 규칙으로 다룰 수 있다면 바람직 할 것이다. 그런데 이 점은 이미 앞에 나온 규칙 NUM_CL에 반영이 되어 있다. 그 규칙을 보면 이어지는 규칙 패키지에 규칙 CL_NOUN이 명시되어 있다. 따라서 수사-분류사 전치 구조의 표현들을 다루기 위해서 추가로 필요한 규칙은 CL_NOUN 하나다.[18]

이 규칙에 따라 (7)을 분석-도출해 보면 그 결과는 <표 19>가 된다.

```
1 +++ Final State +++
2
3 [sur : 소녀가                    ]  [sur : 잔다      ]
4 [noun: girl                      ]  [verb: sleep     ]
5 [cat : (n)                       ]  [cat : (v)       ]
6 [case: (nom)                     ]  [case: ()        ]
7 [sem : (one (individual) ())]       [sem : (present)]
8 [mdr : ()                        ]  [arg : (girl)    ]
9 [fnc : sleep                     ]  [ctn : (0 nil)   ]
10 [wrd : 1                        ]  [wrd : 4         ]
11 [prn : (1)                      ]  [prn : (1)       ]
```

표 19. 도출 6

그런데 위의 규칙 CL_NOUN은 수사-명사로 된 "한 소녀가"같은 구조도 함께 처리할 수 있다. <표 20>은 "한 소녀"를 처리하는 중간 과정이다.

이처럼 규칙 CL_NOUN을 하나 추가해 줌으로써 (8)에 제시된 여섯 가지 가능한 표현들을 모두 분석할 수 있고 이는 다른 예과 마찬가지로 실제 검증을 거쳤다.

---

18) 규칙 CL_NOUN은 10번째 줄-16번째 줄에 명시된 연산이 좀 복잡한 편이나, 이는 앞 절에서 지적한 바와 같이 현재의 구현 시스템상의 제약으로 인한 것이다. 그 규칙의 기본적인 취지는 수사-분류사로 이어지는 표현의 정보를 뒤따르는 명사에 수렴시키자는 것이나 현재 구현된 시스템상 '문장 시작'이 무조건 저장되도록 되어있는 제약을 피하기 위해서 뒤따르는 명사의 정보를 수사-분류사의 합인 '문장 시작'으로 복사하는 방편을 취했다. 즉 '문장 시작'을 저장하지 않도록 하는 장치만 있다면 위의 기본 취지도 살리면서 다른 규칙과 마찬가지로 간략하게 만들 수 있다.

```
1 1.2
2 RULE_CL_NOUN      ecopy(NW.sur SS.sur)      {RULE_NOUN_FV}
3                   ecopy(NW.noun SS.noun)
4                   cancel(X)
5                   ecopy(NW.cat SS.cat)
6                   ecopy(NW.case SS.case)
7                   acopy(NW.sem SS.sem)
8                   ecopy(NW.fnc SS.fnc)
9 [noun: $SS1 ]  [noun: $NW1         ]
10[cat : ($CL)]  [cat : ($X $NP)     ]
11[sem : $SSL1]  [case: ($CS2)       ]
12
13[sur : 한     ]  [sur : 소녀가          ]
14[noun: @1    ]  [noun: girl         ]
15[cat : (num)]  [cat : (num' cl' n)]
16[case: ()    ]  [case: (nom)        ]
17[sem : (one)]  [sem : ()           ]
18[mdr : ()    ]  [mdr : ()           ]
19[fnc : nil   ]  [fnc : @2           ]
20[wrd : 1     ]  [wrd : 2            ]
21[prn : (1)   ]  [prn : (1)          ]
```

표 20. 도출 7

수사 명사 단독 구조

수사 관련 구문으로 마지막으로 검토해 볼 내용은 수사-분류사나 명사가 독자적으로 쓰인 경우들이다. 즉 아래와 같은 표현들이다.

(9) a. 소녀(가) 잔다.
b. 하나(가) 잔다.
c. 한 명(이) 잔다.

(9)에 나온 여섯 가지 가능한 표현들을 처리하기 위해서 별도의 규칙이 필요하지 않다. (9a)나 (9b)의 경우엔 앞에 나온 문장들을 위해 필요한 주어-동사 (NOUN_FV, 각주 17 참고) 규칙으로 충분히 다룰 수 있고, (9c)의 경우엔 규칙 NUM_CL이 먼저 적용된 후 규칙 NOUN_FV이 적용된다. 예를 들어 (9c)의 결과는 <표 21>이다.

```
+++ Final State +++

[sur : 한                    ]  [sur : 잔다      ]
[noun: @1                    ]  [verb: sleep     ]
[cat : (n)                   ]  [cat : (v)       ]
[case: ()                    ]  [case: ()        ]
[sem : (one (individual))]  [sem : (present)]
[mdr : ()                    ]  [arg : (@1)      ]
[fnc : sleep                 ]  [ctn : (0 nil)   ]
[wrd : 1                     ]  [wrd : 3         ]
[prn : (1)                   ]  [prn : (1)       ]
```

표 21. 도출 8

기능불확실성 문제

이 절에서는 좌연접 알고리즘의 한국어 적용에 대한 기존의 문제제기 및 해결방안을 살펴보고, 그러한 해결방안의 문제점을 지적한 후 대안의 방향을 제시하도록 한다.

일찍이 Hausser의 좌연접 문법을 한국어에 적용한 이민행(1993)에 따르면, 한국어에서의 명사구의 "기능불확실성(Functional Uncertainty)"이 좌연접 문법의 적용을 어렵게 한다. 기능불확실성이란 복문에서 문장을 시작하는 명사구가 주절에 속하는지 아니면 종속절에 속하는 지를 파악하기 어렵다는 점을 일컫는다. 종속절이 시작되는 지점에 여러 가지 형태로 표시되는 영어와는 달리 한국어의 경우 그러한 경계표시가 표면에 명시적으로 드러나지 않는다. 다음 예문을 보자.

(10) 철수가 동생에게 준 책을 빼앗았다.

좌연접 알고리즘은 철저하게 왼쪽부터 오른쪽으로 문장을 분석하는 방식이므로, 우선 문장의 시작인 "철수가"를 분석하게 된다. 그런데 (10)에서 "철수가"는 술어 "준"의 주어가 될 수도 있고, 술어 "빼앗았다"의 주어가 될 수도 있다. 그러한 불확실성은 좌연접 알고리즘의 적용에 문제를 야기한다는 주장이다.

그에 대한 해결책으로 이민행(1993)에서는 "분석대상이 되는 표현체안에 들어있는 동사들의 수를 미리 세어, 이를 근거로 가능한 통사적인 중의성을 분석의 시

초에 미리 제한하는 방법"을 제안하고 있다. 그러한 기능을 하는 "count-v"라는 함수가 어느 시점에서 적용이 되는지 명확하지 않은 듯하지만, 해당 문장 분석의 시초 이전에 알아야할 정보라 판단되므로 좌연접 알고리즘 적용의 전제가 되어야 할 것으로 보인다. 즉, 주어진 문장을 분석하기 위해서는 우선 count-v함수를 적용하여 동사가 두개 이상인지를 확인한 뒤에 두개 이상이면 그에 적합한 "문장 시작" 범주로 좌연접 알고리즘 적용을 시작해야 할 것으로 보인다.

그렇다면 이민행(1993)에 제시된 해결 방안은 문제가 있다. 미리 문장을 검색하여 동사의 수를 파악하고 그것을 좌연접 알고리즘 시작에 반영한다는 것은 time-linear라는 좌연접 알고리즘의 근본 정신에 반하는 것이라고 판단된다. 좌연접 알고리즘은 인간의 언어 이해나 발화 행위에 대한 성찰에 바탕을 둔 것으로 사람이 문장을 들을 때 문장을 구성하는 단어가 연쇄적으로 나오는 대로 바로바로 순서대로 이해절차를 거친다고 보는 것이다. 만일, 문장을 미리 다 듣고, 동사의 개수를 파악한 다음 다시 처음부터 그 문장에 대한 (본격적인) 분석을 한다면, 그 것은 인간의 언어 이해활동 양식과 배치되는 것으로 보인다.

물론 이민행(1993)에서 제기된 문제는 타당하다. 그리고 그러한 문제의 해결을 위해 철저한 좌연접 알고리즘의 적용을 일부 유연하게 바꾸어야한다는 주장에도 동의할 수 있다고 본다. 일반적으로 동사 후치 구조를 취하는 한국어의 특성상 좌연접 알고리즘의 적용에 어려움이 있다. 그리고 그러한 특성의 연장으로 절 경계가 영어와는 달리 해당 절 말미에 표시되기 때문에, 철저하게 좌연접 알고리즘을 따를 경우 문장의 일정 지점까지는 아직 문장 구조의 윤곽을 잡기가 어려워진다. 이와 관련한 아주 간단한 예로 타동사 구문에서 동사가 나오기 전의 명사구들, 예를 들어 주어-목적어 연쇄를 어떻게 처리해야할 것인가를 생각해 보자.

(11) 철수가 책을 빼앗았다.

논문 서두에 나온 <표 1>의 좌연접 규칙 도식에 따르면, 처음 나오는 주어 "철수"와 두 번째 나오는 목적어 "책을"을 분석하고, 이어서 "책을"과 동사 "빼앗았다" 순서로 분석을 해야 할 것이다. 그런데 DBS의 구성상, 분석된 주어 명세소와 동사 명세소가 서로 연결이 되어야 할 텐데, 만일 주어-목적어, 목적어-동사 순으

로 분석이 된다면 주어와 동사의 연결을 위한 연산 적용이 어려워질 수밖에 없다.

그러한 문제의 해결 방향은 크게 두 가지로 나누어 볼 수 있을 것이다. 하나는 일반적으로 파서 개발에 활용하는 방식으로, 미리 문장의 틀을 상정해 놓고, 문두부터 분석되는 어휘들을 차례대로 그 틀의 적절한 장소에 넣는 방법이다. 앞에서의 간단한 예에다가 이를 적용해 보면, 우선 가상의 문형, 예를 들어 동사를 하나 설정한 후, 그 동사와 주어를 연결시키고, 이어서 목적어와 그 가상 동사를 연결시키고, 마지막으로 그 가상 동사를 실제 동사인 "빼앗았다"와 결합하는 방법이다. 그렇게 된다면 주어는 가상동사를 매개로 하여 결국 실제 동사와 연결고리를 찾게 된다. 특히 동사가 취할 수 있는 논항의 개수가 제한되어 있으므로 가상동사의 유형을 한정된 수로 설정하여 활용할 수 있을 것이다.

이 같은 방식을 복문에 적용한다면 어떻게 될까? 우선 복문의 얼개를 설정해 놓고 좌연접 방식으로 문장을 분석해 나가면서 매 지점 분석된 명제소를 그 큰 얼개 속에 적절히 배치하는 방식이 있을 수 있다. 그렇게 되면 문장이 끝나는 지점에서 처음 설정한 얼개의 주요 내용들이 해당 명제소로 채워지게 될 것이다.[19] 그러나 한 가지 문제는 동사-논항 관계와는 달리 복문의 유형을 미리 설정하기가 어렵다는 점이다. 문장은 원칙적으로 재귀적으로 종속절을 취할 수 있기 때문이다. 이민행(1993)에서도 지적되었듯이 문두에 나오는 주어 명사구가, 주절의 주어가 될지, 종속절의 주어가 될지, 그 종속절의 하위절의 주어가 될지, 또는 그 이상 깊이 포함된 주어인지 알 수가 없기 때문이고, 그 가능성은 원칙적으로 무한하기 때문이다.

동사 후치 구조 및 절 경계 미명시로 비롯되는 한국어에서의 좌연접 문법적 분석의 어려움에 대한 두 번째 해결 방안으로, 좌연접 규칙의 철저한 표면 인접성 조건을 완화하는 방법도 있다. (11)의 예를 다시 살펴보자면, 우선 "철수가"와 "책을"을 분석한 뒤에, 동사 "빼앗았다"가 나오는 순간 "철수가" 명제소와 "책을" 명제소를 동시에 연결시키도록 하는 것이다. 즉 두 번째 규칙 적용시에 "책을"과 "빼앗았다"의 연결에만 국한하지 않고 아직 제대로 연결되지 못한 "철수"와 "빼앗았다"도 동시에 연결시킬 수 있도록 하는 방안이다. 이는 물론 <표 1>의 좌연접

19) 아마도 이민행(1993)에서의 해결안도 이러한 관점으로도 이해가 가능할 것이다.

규칙 도식을 외형적으로 위배하는 결과가 된다. 그러나 좌측부터 차례로 우측으로 분석해 나간다는 좌연접 알고리즘의 기본정신을 위배한 것은 아니고 다만 바로 인접한 표현들, 또는 바로 인접한 명제소끼리만 연결한다는 점만 완화시키자는 의미다. 좌연접 도식에서 규칙이 적용된 후에도 "문장 시작"에 추후로 접근할 수 있는 가능성을 열어두자는 뜻이다. 이러한 해결책을 따를 경우 <표 1>의 좌연접 도식에 나오는 2개 범주 일치 방식도 유지시킬 수가 있다. 규칙의 적용을 표면에 나오는 바로 선행하는 범주로만 한정하지 않고, 특정 지점까지 분석되어 임시 저장된 범주 중 해당 규칙의 "문장 시작"에 부합하는 범주에 차례로 적용되도록 하는 방법이 가능하다.

이러한 방식을 기능불확실성 문제에 적용할 경우, 문두에 나오는 주어가 종속절의 동사에 연결될 수 있는 가능성과, 그 후에 나오는 동사, 예를 들어 주절의 동사와 연결될 가능성 둘 다를 열어두자는 의미가 된다. 물론 그 다음에 동사가 더 나온다면 그 동사와 연결된 가능성 역시 허용된다.

이렇게 될 경우엔 여러 가능성을 동시에 검토해야하므로 문장 분석기의 부담이 가중될 위험성이 있을 것이다. 그러나 어떤 방식으로 분석하든 어느 정도의 중의성 부담은 피치 못할 현상이고, 또한 언어 사용자들의 문장 이해 방식 역시 원칙적으로는 그러한 부담이 영향을 미치고 있다고 보아야 할 것이다. 예를 들어 포함관계가 복잡한 문장에 대해선 화자들이 문장 이해에 어려움을 겪는다는 것은 이미 잘 알려진 사실이다.

결 론

이 연구에서는 언어 사용자들의 언어 사용 방식을 가장 근접하게 모사하는 한 가지 이론인 DBS에 근거하여 한국어 분석의 가능성을 모색해 보았다. 인지적 행동주로서의 언어사용 방식인 언어의 이해, 추론하기 및 생각하기, 그리고 생각을 말로 표현하기 등을 구체적인 한국어를 예로 하여 구현해 보았다. 또한 한국어의 주요한 특성 중 하나인 수사-분류사 구조에 대한 다양한 표현방법을 DBS에서 어떻게 처리할 수 있는지를 보였다. 좌연접 알고리즘 및 어휘은행 방식으로 20가지의

다양한 표현이 비교적 간결하고 일관성 있게 처리될 수 있다는 점을 실제 구현을 통해 세밀하게 보였다. 마지막으로 DBS의 핵심 알고리즘인 좌연접 규칙에 대하여 제기된 문제와 관련하여 기존의 해결 방안의 문제점을 지적하였고, 그에 대한 새로운 대안을 모색하였다.

의미 표상에 대한 기존 이론들을 살펴보면 전통적으로는 해당 문장의 구조에 상응하는 매우 정교하게 구조화된 의미 표상을 취하는 방식을 따른다. 그러나 다른 한편으로는, 전산처리의 어려움 등을 감안하여, 문장의 의미 정보가 크게 훼손되지 않는 범위 내에서 가급적 평탄한(flat) 표상방식을 선택하는 경향도 있다. Copstake et al. (2001)이 제창한 최소귀환의미론(Minimal Recursion Semantics)도 그중에 하나라 할 수 있다. DBS의 명제소 및 어휘은행 역시 이처럼 평탄한 의미표상구조를 기본으로 하고 있다. 이러한 표상 방식 및 그 표상을 다룰 수 있는 처리 알고리즘이 언어 사용자의 언어행위--이해, 추론, 발화--를 모사해 낼 수 있다는 것이 입증된다면, 그러한 시스템이 인지적 행위주로서의 기능을 수행할 수 있을 것으로 기대할 수 있을 것이다.

참고문헌

신경구 (1992), "선형구구조문법의 한국어 분석," 어학연구 28:1, 125-147.

이기용 (1999a), 전산 형태론, 고려대학교 출판부.

이기용 (1999b), "데이터베이스 의미론의 기초: 자질 구조에서 테이블로", 1999년도 한글 및 한국어 정보처리 학술대회 발표 논문집, 297-303.

이기용 (2000), "Developing database semantics as a computational model", in *Proceedings of the 14th Pacific Asia Conference on Language, Information and Computation*, Tokyo, 231-242.

이기용 외 (2000), "Development of a multilingual information retrieval and check system based on database semantics," *LDV-Forum*, 16-1/2, *Journal for Computational Linguistics and Language Technology(Zeitschrift für Computerlinguistik und Sprachtechnologie)*, Germany, 81-99.

이기용, 홍정하, 이종민 (2002), "언어정보처리를 위한 데이터베이스 의미론 체계의 구축", 언어 27:3, 417-438.

이민행 (1993), "좌측연접문법을 이용한 한국어 파싱의 가능성과 문제점", HCI '93 학술대회 발표논문집, 포항공대, 21-29.

장석진 외 (옮김) (2002), 전산언어학의 기초, Korterm 시리즈 7, 한국문화사. (Hausser 1999/2001 번역본)

홍정하, 최승철, 이기용 (2000), "데이터베이스 의미론을 위한 한국어 피동형의 전산적 처리", 2000년도 한글 및 한국어 정보처리 학술대회 발표 논문집, 411-418.

Choe, J, & R. Hausser (2006) "Treating Quantifiers in Database Semantics", in *Information Modeling and Knowledge Bases XVIII*, edited by M. Duží, H. Jaakkola, Y. Kiyoki and H. Kangassalo, [Paper presented in *The 16th European-Japanese Conference on Information Modelling and Knowledge Bases*, Trojanovice, Czech Republic], Amsterdam: IOS Press Ohmsha, 122-141.

Copstake, A. et al. (2001), "Minimal Recursion Semantics - An Introduction", *Language & Computation*, 1:3, 1-47.

Hausser, R. (1999/2001), *Foundations of Computational Linguistics, Human-Computer Communication in Natural Language*. 2nd Edition 2001, Berlin, New York: Springer-Verlag.

Hausser, R. (2001), "Database Semantics for Natural Language", Artificial Intelligence 130:1, Dordrecht: Elsevier, 27-74.

Hausser, R. (2004), "The Major Constructions of English in Database Semantics", Manuscript.

Hausser, R. (2006), *Database Semantics for Natural Language Communication: A Computational Model of Interpretation, Inference, and Production*, Berlin, New York: Springer-Verlag.

1 차원고접수 : 2007. 11. 19
최종게재승인 : 2007. 12. 17

(*Abstract*)

A pilot implementation of Korean in Database Semantics: focusing on numeral-classifier construction

Jae-Woong Choe

Korea University

Database Semantics (DBS) attempts to provide a comprehensive and integrated approach to human communication which seeks theory-implementation transparency. Two key components of DBS are Word bank as a data structure and Left-Associative Grammar (LAG) as an algorithm. This study aims to provide a pilot implementation of Korean in DBS. First, it is shown how the three separate modules of grammar in DBS, namely, Hear, Think, and Speak, combine to form an integrated system that simulates a cognitive agent by making use of a simple Korean sentence as an example. Second, we provide a detailed analysis of the structure in Korean that is a characteristic of Korean involving numerals, classifiers, and nouns, thereby illustrating how DBS can be applied to Korean. We also discuss an issue raised in the literature concerning a problem that arises when we try to apply the LAG algorithm to the analysis of head-final language like Korean, and then discuss some possible solution to the problem.

Keywords : Database semantics, left-associative algorithm, word bank, numerals, classifiers, implementation