한국어 ATIS 질의문의 QLF 표현과 분석방법에 관한 고찰

Representing & Analyzing QLFs for Korean ATIS Queries

박 현 석*
(Hyun-Seok Park)

요 약 문맥을 고려한 한국어 데이터베이스 인터페이스 시스템을 개발하는 데는 영어권 언어 시스템을 개발하는 데 있어서 별로 거론되지 않았던 여러 가지 문제점이 존재한다. 예를 들어, 한국어의 처리는 구문구조적 제약이 비교적 엄격하지 않기 때문에 적은 수의 규칙으로도 문장이 분석될 수 있지만, 이러한 점은 구문구조분석단계에서 모호성을 남기게 되어 의미구조분석 단계나 뒤에 연속되는 문맥관련 모듈을 설계할 때는 큰 부담으로 작용한다. 이러한 점을 고려하면서, 본 논문에서는 SRI에서 개발한 CLE라는 시스템을 기초로 개발된 한국어 데이터베이스 인터페이스 시스템을 구축하면서 직면한 여러 가지 어려움을 기술하고자 하며 특히 QLF라는 의미구조를 한글에 적용시키면서 관찰된 여러 가지 문제들에 중점을 두고 논의를 전개해 나가고자 한다.

주제어 한국어 정보처리, 의미분석론

Abstract In developing a Korean database interfacing system exist many difficult problems which are not usually pointed out in developing an English version. Basically, the difficulty in implementing deep level Korean database interfacing system lies in the fact that Korean has fewer syntactic restrictions. This means that while less burden will be placed on semantic analysis, more burden will be placed on contextual processing for Korean when compared to English. This paper discusses various difficulties which we came across while developing a prototype Korean DB interfacing system (the Korean Core Language Engine), especially focusing on issues of representing and resolving Quasi Logical Forms.

1. 서론

인간과 컴퓨터 간의 효과적인 인터페이스를 구현하기 위해서 현재 많은 연구가 진행되고 있으며 그 중의 하나로 현재 ATIS(Air Travel Information System)라는 항공여행에 관련된 도메인을 기초로 미국과 유럽 등에서 자연어 DB 인터페이스 시스템과 음성인식 시스템을 경쟁적으로 개발하고 있다. 하지만 ATIS와 같은 도메인에서 문맥을 고려한 한글 데이터베이스 인터페이스 시스템을 개발하는 데는 영어권 언어의 시스템을 개발하면서는 지적되지 않았던 많은 어려움이 존재한다.

한국어는 영어와 비교할 때 구문구조와 관련한 규칙들이 좀 느슨하지만 오히려 이로 인해 시스템을 설계하는 것이 좀 더 어렵다. 예를 들어 한글에서는 동사와 주어 사이에 일치가 없다는 점이나, 자유 어순이라는 점, 대용형의 잦은 생략, 술어 인자의 부재, 정관사나 부정관사와 같은 문법적 장치가 미비하다는 점 등이 모두 문맥을 고려한 한글 인터페이스 시스템을 만드는 데 큰 부담이 된다. 다시 말해, 영어 시스템과 비교할 때, 한국어 시스템의 설계는 구문 구조 분석 모듈이나 의미 구조 분석 모듈에서 약간 더 간단해 보일 수도 있으나, 이것은 곧 뒤에 연속되는 문맥관련 모듈이나 데이터 베이스 인터페이스 모듈을 설계할 때는 큰 부담이 된다.

* 성신여자대학교 컴퓨터정보학부 객원교수(정보통신부 지원) & 동경대학교 정보과학과 TSUJII 연구소 연구원
주소 : 136-742
서울특별시 성북구 동선동 3가 249-1
성신여자대학교 컴퓨터정보학부
전화 : 02 920-7555
FAX : 02 920-7157
e-mail : hspark@cs.sungshin.ac.kr &
hsp20@is.s.u-tokyo.ac.jp

본 논문에서는 CLE(Core Language Engine) 시스템에 기반한 한글 DB 시스템을 구축하면서 직면한 여러 가지 문제들에 대해 논의하고 이러한 문제들을 해결하기 위해 한국어판 CLE에서 채택한 접근방법중 몇가지를 예를 들어 가며 설명하고자 한다 [4,6]. 특히 시스템에서 채택한 QLF(Quasi Logical Form)라고 하는 의미 구조를 다루는데 있어 한글만의 특성으로 발생한 여러 가지 문제에 중점을 두고자 한다. 2장에서는 한글 질의문을 처리하는데 있어서의 난점을 기술하고, 3장에서는 한국어 CLE 시스템을 간략히 소개한다. 4장에서는 한글 ATIS 질의문을 QLF로 표시하는데 대한 문제를 예를 들어 설명하고 있고, 5장에서는 4장에서 예로 든 QLF를 문맥상 해석하여 RQLF로 만드는 과정을 설명하고 있다. 마지막으로 6장에서는 ATIS상에서 관찰된 몇가지 문제점들을 지적하며 향후 연구 방향을 토론하고 결론을 맺는다.

2. 한글 질의문 처리의 난점

현재 구축된 한국어판 CLE 시스템은 600개 정도의 ATIS라는 항공여행관련 질의문을 기본 데이터로 하여 개발되었다. 한국어판 ATIS 말뭉치는 참가한 사람들에게 한 사람당 약 10개 정도의 연속된 질의문을 만들도록 하여 수집되었다. 물론 현재의 한국어 말뭉치는 그 크기와 모아진 방법 등 여러 면에서 영어의 ATIS 말뭉치와는 그 성격이 좀 다르나, 본 논문에서는 편의상 한국어판 ATIS 말뭉치라고 부르기로 한다. 한편 이렇게 모아진 질의문들을 관찰할 때 가장 먼저 눈에 뜨인 점은 대부분의 사람들은 문장단위질의와 문맥연관질의를 구별하지 않고 혼용한다는 사실이었다. 다음의 연속된 질의문을 관찰해 보자.

문1)

(a) 서울에서 동경까지 가는 비행편을 나열하라!

(b) 런던에서 파리까지 가는 비행편도 나열하라!

위 두 개의 연속된 질의문에서는 모두 '비행편'이라는 단어가 나오게 되는데 문1(a)에 나오는 '비행편'과 문1(b)에 나오는 '비행편'은 문맥상 서로 아무 관련이 없다. 다시 말해 문1(b)의 '비행편'은 문1(a)의 "서울에서 동경까지 가는 비행편"이라는 선행어를 참조(refer)하지 않는다.

문2)

(a) 서울에서 동경까지 가는 비행편을 나열하라!

(b) 가장 싼 비행편은?

그와는 대조적으로 문2에 나오는 연속된 질의문에서는 문2(a)의 '비행편'과 문2(b)의 '비행편'이 서로 참조관계에 있다는 것이다. 만약 두 문장에 나오는 '비행편'이라는 단어가 서로 연관이 없다고 해석이 된다면 문2(b)를 수행한 결과로, 서울에서 동경까지 가는 비행편 중 가장 싼 비행편의 정보가 출력되기보다는 데이터베이스 전체 중에서 가장 싼 비행편에 대한 어떤 정보가 출력될 것이다. 이러한 문제들은 한국어에서 정관사나 부정관사가 존재하지 않는 관계로 그 심각성이 더해진다. 영어에서는 이 문제가 다소 덜 심각할 수 있는데, 이는 문맥과 표현 의도에 따라 정관사나 부정관사와 같은 문법적 장치를 써서 'flights'나 'a flight', 'the flight' 등등의 서로 다른 표현으로 '비행편'이라는 단어를 표현할 수 있기 때문이다.

한국어에서는 수(Number)나 성에 대한 정보가 잘 표현되지 않는다. 특히 한국어에서는 복수표현이 영어처럼 엄격하지 않다. 물론 명사 뒤에 '들'과 같은 접미사를 붙여 복수임을 좀더 명확히 표현할 수는 있지만 대부분 생략된다.

문3) 서울까지 가는 비행편을 나열하라.

문4) 서울까지 가는 비행편을 예약하라.

문3과 문4는 문장 맨 끝에 나오는 동사를 제외하고는 문의 구조가 동일하다. 하지만 대개의 사람들은 문3의 '비행편'은 '모든 비행편들'이라고 해석하고 문4에 나오는 '비행편'은 문맥에 따라 한두대 정도의 비행편이라고 해석하는 것이 일반적이다. 이와 같은 해석은 물론 "사용자가 가능한 한 많은 비행편에 관한 정보를 알고 싶기는 해도 모든 비행편을 예약하기를 원하지 않는다"는 지극히 상식적인 사실에 근거하고 있을지도 모른다. 한편 위의 두 문장은 아래와 같이 좀더 명확한 질의로 표현될 수도 있다.

문5) 서울까지 가는 비행편들을 모두 나열하라.

문6) 서울까지 가는 비행편을 하나 예약하라.

하지만 사용자가 늘 문5나 문6처럼 질의를 분명하게 표현하지도 않을 뿐더러 문5나 문6이 문3이나 문4에 비해 더 자연스러운 질의문이라고 보기도 어렵다. 이러한 일련의 문제를 계속 더 기술하기보다는 현재 시스템에서 실제 한글 질의를 어떤 의미구조로 표현

했으며 어떻게 다루고 있는지를 예를 들어 가며 4장과 5장에서 설명하고자 한다.

3. 한글 CLE 시스템

CLE (Core Langauge Engine) 시스템은 영국 캠브리지 소재 SRI 연구소에서 개발되었다 [4]. 이 시스템은 이미 영어, 불어, 스웨덴어에 대해 개발되어 있으며, 소규모이지만 스페인어와 독일어에 대한 응용도 현재 시도되고 있다. 이 시스템을 바탕으로 한국어판을 개발하였는데 ATIS 관련 데이터베이스 인터페이스 모듈을 접합하여 기본적인 프로토타입 시스템을 완성했다. 현재의 시스템은 키보드에서 입력한 한글 문장을 받아 처리하고 사용자가 원하는 항공 스케줄이나 기타 데이터 등을 보여 준다. 데이터베이스에 대한 액세스는 입력된 문장을 SQL 타입의 질의문으로 바꾸어 줌으로써 진행되는데 그 방법은 AET [1]라는 방법에 그 이론적 기초를 두고 있다 [3,5]. 예를 들어,

kcle〉〉 *필라델피아에서 달라스까지 가고 식사를 제공하는 비행편을 나열하라!*

라는 질의문이 입력되었을 때, 시스템은 아래와 같은 6개의 필드 정보를 보여준다.

만약 어떤 시스템이 대화체나 생략문을 고려하지 않고 단지 문장단위 질의문만을 염두에 두고 개발한다면 실용적인 측면에서 많은 난점이 있을 것이다. 이러한 이유로 제한된 범위에서의 담화 질의문과 생략문을 허용하고 있다. 실용적인 시스템을 설계하기 위해서는 점차적으로 그 전의 질의에 관련된 내용에 대해 더 구체적으로 질의하는 것을 가능하게 해 주어야 한다[2]. 예를 들어 위의 질의에 연이어 다음과 같은 생략문을 시스템은 올바르게 처리할 수 있어야 한다.

비행편	출발요일	목적지	도착지	출발시각	도착시각
AA1067	월화수목금토일	PHL	DFW	6:32 am	8:39 am
AA891	월화수목금토일	PHL	DFW	7:44 am	9:56 am
US1111	월화수목금 일	PHL	DFW	8:30 am	10:47 am
DL931	월화수목금토일	PHL	DFW	9:00 am	11:04 am
YX304	월화수목금	PHL	DFW	11:00 am	3:10 pm
...	...	...	...	...	...

kcle〉〉 *(그중에서) 가장 싼 비행편은?*

비행편	출발지	목적지	코드	제약조건	비용	편도
AA	PHL	DFW	V	-	594.0	편도

만약 문맥을 고려하지 않는다면 이 질의문에서의 '비행편'의 구체적인 의미가 파악될 수 없기 때문에 사용자가 첫 번째 질의에 부연한 정보를 얻기 위해서는 일부 구문을 다시 반복해야 했을 것이다. (예를 들어, 생략된 표현이 허용되지 않는다면 위 문장은 "필라델피아에서 달라스까지 가고 식사를 제공하는 비행편중 가장 싼 것은 무엇인가?"라고 다시 질의해야 한다.)

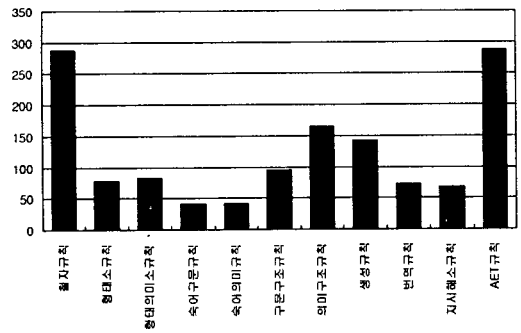
시스템 구성: 현재의 시스템은 (그림 1)에서와 같이 크게 12개의 모듈화된 구조로 되어 있다.

1. 형태소 분석 (Segmentation and Morphological Analysis)
2. 숙어 단위의 구문구조 분석 (Phrasal Parsing)
3. 구문 구조 분석 (Syntactic Analysis)
4. 의미 구조 분석 (Semantic Analysis)
5. 선택 제약 필터링 (Sortal Filtering)
6. 선호도 적용 (Preference Metric Application)
7. 번역 (Translation)
8. 참조 해소 (Reference Resolution)
9. 양화사 스코핑 (Quantifier Scoping)
10. QLF에서 TRL로의 변환 (Translation from QLF to TRL)
11. TRL에서 SQL로의 변환 (Translation from TRL to SQL)
12. 질의문의 실행 (Evaluation of the Query)

(그림 1) 시스템의 모듈화

(그림 1)에 보이는 각 단계에서는 여러 종류의 규칙들이 선언되어 적용되며, (그림 2)에는 이러한 규칙들과 그 수가 나타나 있다. 각 단계에서 다음 단계로 넘어갈 때 어떤 특정한 데이터 구조들을 넘겨 주게 되는데 예를 들어 시스템에서 채택된 QLF라고 하는 의미구조는 4번째 단계인 의미구조 분석단계에서 생성된다. 7번째 단계는 기본적인 한국어 문법에 의해 생성된 QLF를 ATIS 관련 한국어 문법의 부분집합에 의해 생성된 QLF중 어느 하나로 변환해 주는 과정에 필요하며, 이 모듈의 역할은 뒤에 나올 지시해소규칙[3]의 수를 줄여서 (그림 1)의 8번째 단계에서 해야 할 일의 부담을 줄이는 역할을 한다.

1) Abductive Equivalential Translation
2) Incremental Interpretation
3) Reference Resolution Rules



(그림 2) 각 모듈에 필요한 규칙

8번째와 9번째 모듈은 QLF에서 아직 정해지지 않은 여러 가지 메타 변수들의 값을 초기화하는 역할을 한다. 마지막으로 10, 11, 12번째 모듈은 질의문을 QLF 의미구조를 좀더 컴퓨터가 처리하기 쉬운 SQL 타입의 질의로 바꾸며, 그 이론적인 바탕은 AET[4]에 기초하고 있다 [7].

4. QLF 표현에 관한 문제

2절에서는 한국어 질의문을 다루는 데 있어서 몇 가지 어려움을 살펴 보았다. 이 절에서는 문7에 대해서 생성된 QLF를 (그림 3)에 제시하고 그 의미를 설명하고자 한다.

문7) *필라델피아까지 가고 식사를 제공하는 비행편을 나열하라.*

QLF: QLF는 기존의 의미구조들이 지나치게 복잡하다는 점과 이들을 해석하는데 상호호환성(reversibility)이 보장되지 않는다는 단점때문에 좀더 실용적으로 자연어처리 시스템에 의미구조를 접목하기 위해서 개발되었다. 실제로 CLE 시스템은 의미구조가 실용 시스템에 자연스럽게 접목된 매우 드문 예 중의 하나이다. QLF는 의미해석의 방법으로 MSI[5] 방식을 기초로 하고 있다[5]. QLF의 기본적 구조는 1차원 논항구조이나, 양수사나 최상급 등을 표현하기 위해 몇몇 고차원 논항구조가 쓰여 진다.

QLF의 구조는 특정언어와 무관하게 설계되었지만, 여전히 그 설계된 구체적인 내용이나 예 등은 영어에 필요한 의미구조 표현을 염두에 두고 설계되었다. 특히 한국어는 구문구조 그 자체는 영어에 비해서 매우 단순한 대신 여러 가지 형태소의 존재 유무에 따라 미묘한 의미 차이가 난다. 이러한 모든 것을 QLF에 제대로 반영하기 위해서는 기존의 QLF에 어느 정도의 수정이 불가피하였다.

(그림 3)은 문7의 문장에 대해 의미 분석 단계 후 시스템에 의해 생성된 8개의 QLF들 중 가장 높은 선호도를 받은 QLF의 예이다. 이 때 '식사', '비행편', '나열하라' 등의 단어는 각각 'meal1', 'flight1', 'list1'이라는 의미술어로 표현되었다. QLF에서의 중요한 논항구조에는 term/6 구조와 form/5 구조가 있는데, term은 주로 명사구를 form은 주로 논항구조를 나타내는데 쓰인다 [4]. term이나 form 구조의 두 번째 인자는 속해있는 구조체의 범주를 분류하는데 사용한'다. 여기에는 구문구조정보, 양수사정보와 형태소에 관련된 정보를 포함하게 된다.

예를 들어 (그림 3)의 1은 전체 form구조의 범주를 verb(no,no,no,imp,y,med,na)라고 표현하고 있는데 각각의 인자가 뜻하는 바는, 현재의 동사가 시제의 값은 없고 (no), 완성형 시제가 아니며(no), 진행형이 아니고(no), 명령형이면서 (imp) 능동형이며 (y), 존칭이 중간정도(med)임을 나타내고 특별하게 주의를 요하는 어미나 조사가 붙어 있지 않음을 의미한다. 이와 같이 QLF안에는 접미사에서 나타날 수 있는 여러 가지 시제(tense)나 상(aspect)에 관한 정보도 나타낼 수 있다. 이런 면에서 기존의 일반적인 의미구조들 보다는 상당히 많은 정보를 QLF 그 자체에 직접 나타내고 있다고 보아야겠다.

가산명사 '비행편'의 의미 범주 구조는 (그림 3)의 3에서와 같이 4개의 인자를 갖는 **ref/4** 논항구조로 표시된다. 여기에서 **ref**의 두 번째 인자는 참조성(Referentiality)을, 세 번째 인자는 수(Number)를 나타내고 있으며, 네 번째 인자는 어순에 관한 정보를 표시한다. 전절에서 암시하였다시피 한국어에서는 참조성이나 수에 관한 정보가 단순히 의미 구조 분석 후에 바로 결정될 수가 없는데 이런 이유로 (그림 3)의 3에서 I나 J 같은 변수가 아직 초기화되어 있지 않다. 물론 예외도 있다. 예를 들어 "그 비행편"이라고 아주 구체적으로 지시사를 사용하여 사용자가 언급하면 영어의 'the flights'와 비슷한 대용어로서 역할을 확실하게 하지만, 지시사를 사용하는 표현이 다소 어색하므로 사용자가 이런 표현을 꺼린다.

(그림 3)의 2는 또 다른 종류의 term 표현인데, 생략된 주어를 표현하고 있다. 한국어에서는 흔히 대

4) Abductive Equivalential Translation
5) Monotonic Semantic Interpretation

```
[imp,
 form(l([필라델피아까지,가고,식사를,제공하는,비행편을,나열하라])),
     verb(no,no,no,imp,y,med,na), 1
     A, B^[B,
        [list1, A,
         [term([C,ref(pro,zero,D,l([])),E,F^[entity,F],G,H], 2
         term(l([필라델피아까지,가고,식사를,제공하는,비행편을]),ref(bare,I,J,l([K-L|M])), 3
             N, O^[and,[flight1,O],
                    form(l([필라델피아까지,가고,식사를,제공하는]), conj(s,고1), 4
                    P,Q^[Q, [island,
                             form(l([식사를,제공하는]),verb(pres,no,no,no,y,med,na),R,
                             S^[S, [serve1,R, v(N),
                             term(l([식사를]),ref [bare,T,U,l(M)],K,V^[meal],V1,W,X)]),Y)], 5
                            [island,
                             form(l([필라델피아까지,가고]), verb(pres,no,no,no,y,med,na),
                             Z, A1^[A1, form(l([필라델피아까지]),[prep('까지'), B1,
                                    C1^[C1,v(Z),
                                    term(l([필라델피아까지]),proper_name(D1),E1,
                                        F1^[name_of,F1,philadelphia1],G1,H1)],I1),
                                        [go_Move,Z,v(N)]], J1)]],K1)], L1,
          M1)]], N1)]
```

(그림 3) Unresolved QLF

용형 자체가 생략되는 경우가 많은데 이 항은 주격 NP 대용과 비슷하게 다루어지고 있으며 다만 범주 표현에서 **zero**라는 인자를 사용하여 그 항이 생략된 대용표현(zero anaphora)임을 표시하고 있다. 한편 '**list1**'의 목적격 슬롯은 '**flight1**'이고 이것은 다시 두 개의 수식구에 의해 수식받게 되는데 이때 (그림 3)의 **island**라는 연산자는 어떤 term 구조가 **island**가 표시하고 있는 관계절을 벗어나서 스코프 해석을 하는 것을 막기 위해 쓰여진 일종의 제약이라고 생각할 수 있다. 이 두 구조는 다시 **conj**/2라는 연산자에 의해 대등접속문 형식으로 묶여 있다 ((그림 3)의 4 참조).

5. QLF 해석에 관한 문제

문맥이나 지식체계가 고려되지 않는 시스템에서는 문장의 정확한 뜻을 파악하기가 매우 힘든다. CLE에서는 어떤 입력 문장에 대해 처음에는 문맥이 고려되지 않은 UQLF(Unresolved QLF)를 생성하며, 이때 UQLF 구조 안에는 아직 그 값이 정해지지 않은 많은 메타 변수가 존재한다. 이렇게 생성된 UQLF의 메타 변수 각각에 대해 어떤 값을 주어서 생성한 구조를 RQLF(Resolved QLF)라고 한다. 여기에서 한 가지 명시해야 할 것은 이렇게 문맥을 고려하여 만들어진 RQLF는 단지 구조상 UQLF의 부분집합일 뿐, 구조 자체가 UQLF와 완전히 다른 것은 아니라는 점이다. 즉 QLF는 UQLF와 RQLF의 두 단계로 표현되지만 이 과정에서 그 구조 자체가 완전히 변하지는 않는다. 그림 3에서 보인 QLF도 좀더 엄밀히 말하여서는 UQLF이다.

문맥해석이란 UQLF에 있는 모든 변수들을 초기화시킴으로써 RQLF로 만들어주는 과정을 말한다 [4]. 이러한 과정은 뒤에 곧 설명될 일련의 지시해소법칙을 재귀적으로 적용함으로써 그림 1의 단계 8과 9에서 일어나게 된다. 예를 들어 그림 4의 구조는 그림 3의 UQLF에 지시해소규칙(reference resolution rules)을 연속적으로 적용함으로써 얻어 진 것인데 현재 68개의 지시해소규칙이 정의되어 있다 (그림 2 참조). 먼저 지시해소의 과정은 UQLF와 문맥 지식으로부터 어떤 함수를 정의해 RQLF로 변환하는 과정으로 생각할 수 있는데 다음은 지시해소규칙 중 몇 가지를 나열했다.

```
[imp,
 quant(forall, 1
       A,
       [and,
        [FLIGHT1,A],
        [and,
         quant(exists,
               B,
               [event,B],
               [and,
                [GO01,B,A],
                [To_Direction,B,SF(airport(TYO))]]), 2
         quant(exists, 3
               C,
               [MEAL1,C],
               quant(exists,D,[event,D],
                              [SERVE1,D,A,C])))],
       quant(exists,
             E,
             [and,
              [event,E],
              quant(exists,F,[current_time,F],
                             [precedes_in_time,F,E])],
             [LIST1,E,'the system',A]))] 4
```

(그림 4) 단순화된 RQLF

```
refrule(form(_,_,_,_,R),already_resolved_for
m,_) :- nonvar(R).
refrule(term(_,ref(pro,zero,_,_),_,_,_Q,R),ze
ro_hearer_imp_ref,major) :- var(R).
refrule(term(_,ref(bare,_,_,_),_,_,_Q,R),bare
_np_def_resol,major) :- var(R).
refrule(form(_,ell(topic),_,_,_),topic_ellipsis
_candidate,major).
        ...
```

예를 들어 already_resolved_form() 규칙은 R이라는 메타변수가 초기화되었는지를 검사하고 루프를 벗어날 때 쓰이는 규칙이며, zero_hearer_imp_ref() 규칙은 명령어 질의문에서 문맥상 청자가 누구인지를 찾아주는 규칙이다. 또한 bare_np_def_resol() 규칙은 수량명사를 다룰 때 쓰는 규칙이며, topic_ellipsis_candidate() 규칙은 생략문을 다루기 위한 여러 규칙 중의 하나이다.

(그림 4)는 (그림 3)의 UQLF에 여러 가지 지시해소규칙들을 재귀적으로 적용함으로써 얻어진 RQLF의 단순화된 표현이다. 그림 4-2 에서는 '까지'라는 접미사가 'To_Direction'라는 의미로 해석된 것을 알 수 있다. 여기에서 '까지'라는 접미사는 단순히 한 가지 의미만이 있는 것이 아니므로 지시해소단계에서 그 의미가 정해져야 한다[6]. 또한 그림 4-4 에서 list1의 주체가 ent(the system)이라고 초기화되었는데 이것은 현재의 청자가 시스템임을 말하는 것이다. 이것은 zero_hearer_imp_ref()라는 규칙을 적용함으로써 얻어진 것인데 매우 간단한 규칙으로서 프로로그로 선언된 실제 코드는 다음과 같다.

```
zero_anaphora_imp_ref(Pro,_Sal,[imp|_QLF],ref
t(exists,ent(Hearer)),Score) :-
    qlf_expr_category(Pro,ref(pro,zero,_,_)),
    current_hearer(Hearer),
    refrule_score(zero_anaphora_imp_ref,Score).
```

이 규칙은 전체 QLF가 명령형(imp)일 때 적용가능하며 변수 Pro의 범주가 zero인지를 확인한 다음 지식베이스에서 현재의 청자를 찾아서 지정해 준다.

이러한 간단한 규칙 이외에 매우 복잡한 규칙도 있는데 그 대표적인 예가 가산명사를 다루는 bare_count_np_resol()이라는 규칙이다. 이 규칙이 복잡한 이유는 한국어에서 관사와 같은 문법적 장치가 없고 복수 표현마저도 명확하지 않은 관계로 가산

6) 예를 들어 ATIS 도메인상에서 'To_Direction'이라는 의미 이외에 'By_Temporal'이라는 의미도 있다. 이것은 동사와 명사구 그리고 다른 여러 가지 상황을 고려해 implicit_relation_candidiate()이라는 지시해소규칙에 의해 결정된다.

문#7	형태소분석	프류닝1	숙어분석	프류닝2	구문구조분석	의미구조분석	의미선호도적용	지시해소	스코핑	문장생성	추론
한국어	1.43	0.12	0.07	0.37	0.82	2.95	4.78	13.12	1.58	44.10	15.67
영어	0.13	0.57	0.15	1.01	2.19	3.88	4.92	5.20	1.34	3.31	10.07

(그림 5) 각 단계의 수행 시간

명사의 취급이 용이하지 않기 때문이다. 일단 모든 가산명사는 그 앞에 특별히 '그'와 같은 지시사가 붙지 않는 이상 최하 두 가지 해석이 가능하다. 하나는 대용어로 처리하는 것이고 다른 하나는 선행어 참조가 불가능하다고 해석하는 경우이다[7].

물론 어떤 가산명사에 대해 선행어들 중에서 선택제약상 서로 참조 가능한 객체를 전혀 발견할 수 없을 때는 단순히 새로 언급된 객체라고 해석[8]하면 된다. 하지만 ATIS 도메인상에서 '비행편'과 같은 명사는 흔히 선행문에서 몇번이고 언급되었다고 가정하는 것이 합리적이다. 다만 언급되었다 하더라도 그 뒤에 나오는 '비행편'이라는 단어의 참조 가능성을 결정하기가 쉽지는 않다. 한글에서는 복수표현이 뚜렷하지 않다는 점까지 고려한다면 뚜렷한 문맥상의 키워드나 지시자가 있기 전에는 모든 한국어 가산명사는 적어도 네 가지의 해석(참조가능과 참조불가능; 단수와 복수)이 가능하다는 논리가 나오게 된다.

가산명사를 참조가능한 객체로 취급할 때는 별 문제가 없으나 참조 불가능하다거나 문맥상 새로운 객체라고 해석했을 때는 좀 더 미묘한 문제가 발생할 수 있다. 그림 4의 예에서 'flight1' 혹은 '비행편'에 관한 양화사는 'forall'(그림 4-1 참조)이라고 정해진 반면에 'meal1'과 관련된 양화사는 'exists'(그림 4-2 참조)라고 지정되어 있다. 얼뜻 보기에는 별로 중요한 문제가 아닌 듯 보이지만 질의의 출력결과가 크게 달라질 수 있다.

"필라델피아까지 가고 식사를 제공하는 비행편을 나열하라"라는 문장을 해석하는데 있어서 '비행편'에 관련된 양화사가 'exists'로 잘 못 주어졌을 경우에는 비행편 한대에 대한 정보만 나열될 수 있다. 반면에 'meal1'의 양화사가 'forall'로 잘못 주어졌을 경우에는, 어떤 비행편도 모든 식사를 서비스하지는 않기 때문에 출력이 나오지 않을 수 있다. (즉 다른 곳에서 제공할 식사편까지 제공할 수 있는 비행편은 없다.)

7) referential reading vs. existential reading

8) existential reading

한국어에서는 뚜렷이 표현되지 않는 양화사 처리에 관한 문제에는 많은 변수가 있기 때문에 완벽한 해결책을 제시한다는 것이 무리이지만 위 문제에 대해서는 대략 다음과 같은 논리로 접근하였다. 예를 들어, ATIS 도메인에서 '비행편'이라는 단어가 '나열하라'의 목적어일 경우에는 양화사 'forall'과 결합하는 해석이 강하고 '비행편'이 '예약하라'의 목적어이면 'exists'와 결합하는 해석이 강하다. 이와 같이 어떤 동사와 명사의 의미, 그리고 그들의 관계 등의 정보가 주어졌을 때 그에 알맞은 양수사와 관련 선호도 값을 돌려주는 함수를 정의해 이 문제에 접근하고 있다[9] [6].

6.토론 및 향후과제

현재의 시스템은 기존의 12만줄 정도의 CLE 프로로그 코드에 한국어 처리를 위해 약 2만줄 정도의 코드를 추가한 것 이외에, 한글 ATIS에 관련된 한글 데이터베이스 인터페이스 모듈을 개발하기 위해 약 만줄 정도의 프로로그 코드가 기존의 코드위에 다시 쓰여지거나 수정됨으로써 개발되었다. 현재 약 600개 정도의 질의문을 처리할 수 있으나 아직 개발의 초기 단계이므로 임의의 질의문에 대한 시스템의 성능분석은 시도되지 않았다.

수행시간: 그림 5는 문7의 문장과 그와 유사한 영어 질의문[10]간에 각 모듈별로 수행시간을 비교해 본 데이터이다 [6]. 형태소 분석 단계에서는 한국어 동사의 특성으로 인해 전반적으로 영어보다 시간이 많이 걸리지만 구문구조 분석 등에서는 훨씬 시간이 적게 걸리는 것을 알 수 있다. 물론 영어와 한국어의 구문구조 분석이나 형태소 분석을 각각의 단계에서 비교를 한다는 것은 다소 무리가 있지만 중요한 것은

9) 이 때 일반적으로 똑같은 조건일 때의 선호도는 참조 가능한 해석(referential reading)에 주어지게 된다. 그 이유는 만약 어떤 선행어들과도 관련은 없지만 선택제약상 관련을 분리시키기 곤란한 어떤 새로운 객체를 언급할 때는 사용자 개개인이 좀더 확실하게 표현하여 그 관계를 명확히 표현하려고 노력한다는 경험상의 관찰 때문이다.

10) "Show me the flights which go to Philadelphia and serve a meal."

문맥관련 모듈을 수행할 때 한국어의 경우 월등히 시간이 많이 걸린다는 점이다. 이는 앞에서 언급한 한국어 QLF 표현의 모호성으로 인한 것이다. 예를 들어 그림 3의 UQLF에 대해서 약 32개의 RQLF가 생성되었고 이것만 보아도 이러한 문맥관련 모듈에서 얼마나 많은 부하가 걸리는지를 짐작할 수 있다. (이렇게 생성된 32개의 RQLF는 다시 적용된 지시해소 규칙의 비용(Cost)에 따라 선호도가 계산되고 있다 [6].) 물론 이와 같은 비교는 영어와 한글 시스템의 접근 차이와 그 밖의 많은 요소를 고려하지 않은 것이기 때문에 객관성이 결여된 것이기는 하지만 그래도 어느 정도 시사하는 바가 있다 하겠다.

필드정보에 관한 문제: 현재의 ATIS 말뭉치를 관찰해 보면 질의문을 처리하는데는 기존의 계산 언어학에서 잘 다루어지지 않았던 많은 어려움이 있음을 쉽게 관찰할 수 있게 된다. 다음의 연속된 질의문을 고찰해 보자.

문 8.
(a) 동경에서 서울까지 가는 비행편을 나열하라.
(b) 그중에서 가장 싼 비행편을 나열하라.
(c) 가격은?

문 8(a)를 수행하면 동경에서 서울까지 가는 비행편의 정보를 나열하게 된다. 이 때 '나열하다'라는 동사의 목적어가 '비행편'일 경우 '비행편 이름', '출발요일', '목적지', '도착지', '출발시각', '도착시각'의 각각 6개의 필드가 자동적으로 화면상에 나열되도록 되어 있다. 문 8(b)를 수행할 때는 '동경에서 서울까지 가는 비행편' 중 가장 싼 것을 나열하기는 하나 역시 '나열하다'라는 동사의 목적어가 '비행편'이라는 객체이기 때문에 가장 싼 비행편에 대해서 똑같이 6개의 필드가 나열된다.

문제는 사용자가 그 비행편의 가격을 알고 싶을 경우이다. 6개의 필드 중 가격에 대한 정보가 없기 때문에 사용자가 문8(c)와 같은 질의문을 던지는 것은 당연하며, 이 때 현재 시스템에서는 목적어가 가격일 때는 표의 가격을 포함한 7개의 필드가 화면상에 나타나도록 규정하고 있으므로 사용자의 목적을 만족시킬 수 있다. 문제는 문8(c)의 가격이라는 명사는 문 8(a)나 (b)의 그 어느 선행어와도 연계될 수 없다는 것이다. 물론 사용자가 '그 비행편의 가격은?'이라고 친절하게 물었을 때는 상관이 없지만 실제 상황에서 사용자가 시스템 개발자의 측면을 고려하면서 쉽게 질의를 한다고 기대하는 것은 무리다. 현재의 시스템에서는 먼저 '가격은'이라는 속성이 질의문의 표현상에 있을 때는 7번째 번역단계(그림 1 참조)에서 '그것의 가격은?'이라고 자동적으로 번역한 다음 '그것'이라는 대용어의 객체를 뒤에 나오는 8번째 단계에서 찾아 주고 있는데 앞으로 이러한 문제에 대한 좀 더 객관적이고 체계적인 연구와 접근이 필요하다.

7. 결론

본 논문은 프로토타입 한국어 DB 인터페이스 시스템을 구축하는데 있어서 어려운 점중 그 일부를 예시주로 설명하였다. 현재까지 한국어 담화나 의미분석에 대한 많은 이론적 연구가 진행되었다 [2,3]. 뿐만 아니라 포항공대에서 개발하고 있는 SKOPE 엔진과 같은 시스템은 의미분석 단계까지 개발이 진전되어 있으며 생략, 대용 현상 처리에 대한 구체적인 방법론까지도 제시하고 있다 [1]. 하지만 여전히 문맥을 고려한 실질적인 한국어 데이터베이스 인터페이스 시스템이 본격적으로 개발된 예는 거의 없다. 이것은 시스템을 구축하는 것 자체에 대한 어려움보다는 한국어를 다루는데 있어서의 이론적인 문제가 제대로 정립되지 않은 점과, 이로 인한 설계상의 난점이 그 주된 이유라 하겠다. 이런 점에서 앞으로 한국어 질의에 관련된 의미구조나 시스템 구축상의 설계에 대한 논의가 더욱 활발해져야 하며 더 많은 데이터와 이에 대한 세밀한 분석이 시도되어야 한다.

References

[1] 노현철, 이근배, 이종혁, 박재득, (1998), 한국어 담화 특성에 기반한 영역독립 생략 및 대용 처리, 정보과학회 논문지 (B), 25-12, 1845-1857

[2] 신현숙 (1985), 의미분석의 방법과 실제, 한국문화사

[3] 한재현 (1981), 생략과 대용현상, 박사논문, 전북대학교 영어영문학과, 한시문화사

[4] Hiyan Alshawi, David Carter, Richard Crouch, Stephen Pulman, Manny Rayner, and Arnold Smith, (1992), *Clare: A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine*, Technical Report CRC-028, SRI International, **http://www.cam.sri.com**

[5] Hiyan Alshawi and Richard Crouch, (1992), *Monotonic Semantic Interpretation*, Proceedings of the 30th Annual Meeting of the ACL

[6] Hyun Seok Park, (1997), *The Korean Core Language Engine*, PhD Thesis, University of Cambridge

[7] Manny Rayner, (1993), *Abductive Equivalential Translation and its application to Natural Language Processing*, PhD Thesis, Stockholm University, Sweden