產業科學技術硏究所 論文集 第三卷 第二號
(J. of Ind. Sci. and Tech. Institute)
Vol.3, No.2, 1989. 10

기계번역에서 자질구조 변환기 개발에 관한 연구

이하규 · 서영훈*
서울대학교 컴퓨터공학과
*충북대학교 전자계산기공학과

A Study on the Development of Feature Structure Transfer System in Machine Translation

Lee Ha Gyu, Seo Young Hoon

Dept. of Computer Engineering, Seoul National University, Seoul, 151-742, Korea
* Dept. of Computer Engineering, Chungbuk National University, Chungju, Chungbuk, 360-763, Korea

Abstract

In this paper, we describe the transfer method and structure of the transfer core system FEASTS developed for the practical use. Improving the problems in the previous transfer systems, it provides effective rule processing method and mechanism for describing rules that are simple but have powerful expressive power in order to increase the practicality of the system FEASTS, the core part of tool-type unidirectional feature structure transfer system, can process the various feature structures for different unification-based grammar theories independently of the languages to process. Because it contains several tools, including debugging tool, that support the construction of transfer part, users can makeup transfer part for specific language pair with ease by constructing transfer dictionaries and rules using FEASTS. In addition, FEASTS can be used as generation part by replacing transfer dictionaries and rules with generation dictionaries and rules because it is independent of the feature structures to process.

Ⅰ. 서 론

기계번역 방식은 트랜스퍼(transfer) 방식과 피봇(pivot) 방식으로 대별되지만 하나의 중간구조를 상정하는 피봇 방식은 범언어적인 중간구조의 설정이 어렵기 때문에 현재는 트랜스퍼 방식을 많이 취하고 있다.[2,7] 그 처리 대상의 구조에 따라 살펴볼 때 트랜스퍼 방식 중에서는 구절구조트리(phrase structure tree)를 대상으로 삼는 트리구조 변환기(tree structure transfer)와 통합기반문법(unification-based grammar) 이론에서 취급하는 자질구조(feature structure)를 대상으로 하는 자질구조 변환기(feature stucture transfer)가 주류를 이루고 있다. 지금까지는 패턴 매치(pattern match) 및 구성에 의해 일련의 트리 변형 과정을 수행함으로써 트랜스퍼 단계를 수행하는 트리구조 변환기가 많이 사용되어 왔으나, 최근 통합기반문법 이론들이 등장함에 따라 자질구조 변환기의 사용이 점차 확산되고 있다.[4,5,10,14]

자질구조 변환기에 대해서는 양방향(bidirectional) 변환기를 비롯해 일부 연구 결과가 발표된 바 있으나 대부분이 시제품 수준이며 실용적인 변환기가 되기 위해서는 많은 다양한 언어 현상을 효율적으로 처리할 수 있도록 변환규칙(transfer rule) 기술기법과 변환방법이 많이 개선될 필요가 있다.[10] 그리고 해석기(analyzer)의 출력구조인 원시언어 해석구조(source language analysis structure)와 생성기(generator)의 입력구조인 목표언어 생성구조(target language generation structure) 사이에는 일대일 대응성이 존재하지 않는 경우가 많으므로 같은 변환규칙으로 양방향 변환을 시도하는 양방향 변환방식은 다양한 언어 표현을 취급해야 하는 실용적인 변환기로는 적합하지 못하다.[5,10]

변환기가 실용성을 지니기 위해서는 먼저 단순하면서도 표현력이 뛰어난 변환규칙 기술기법을 제공하여 문맥제약 조건(context constraint condition)을 충분히 반영하면서도 효과적으로 변환규칙을 작성할 수 있도록 해야만 한다. 그리고 변환규칙과 사전정보를 효율적으로 저장 및 처리할 수 있는 기법도 제공되어야 한다. 여기에 덧붙여 사전과 규칙 자체를 제외한 변환기의 골격부(core part)가 처리대상 언어에 독립적이어서 사전과 규칙의 재구성만으로 다른 언어간의 변환기로 활용될 수 있을 때 그 효용성은 더욱 증대될 수 있다.[5]

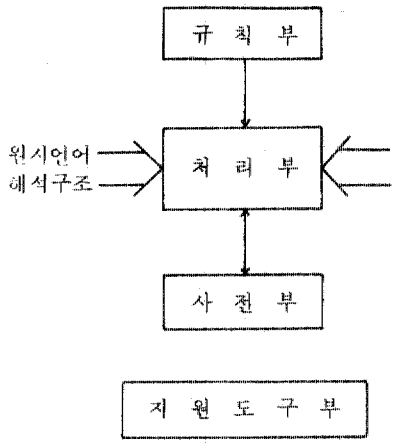
본 논문은 이상의 요건을 고려하여 실용화를 목표로 개발된 변환기 골격 시스템인 FEASTS(FEAture Structure Transfer System)의 구성과 변환방법에 대해 설명한다. FEASTS는 특정 통합기반문법 이론의 자질구조와는 독립적으로 일반 자질구조를 처리할 수 있도록 설계된 도구형태(tool-type)의 단방향(unidirectional) 자질구조 변환기로서 현재 Common Lisp로 구현되어 있다. 먼저 2장에서 FEASTS의 전체적인 구성을 살펴보고, 3, 4, 5장에서는 FEASTS의 세 주요 구성요소인 사전부, 규칙부 및 처리부의 구성과 기능을 각각 설명하고 6장에서 결론을 맺는다.

Ⅱ. FEASTS의 구성

그림 1은 FEASTS의 전체적인 구성을 보여준다.

사전부에는 변환에 사용되는 어휘정보와 그 어휘를 탐색키(search key)로 적용 가능한 규칙정보를 저장하는 변환사전과 이들 정보를 조작하는 기능이 포함된다.

규칙부는 변환규칙과 이에 부속된 함수 및



(그림 1) FEASTS의 구성

각 규칙과 연관된 정보를 저장하고 처리하는 기능을 수행한다.

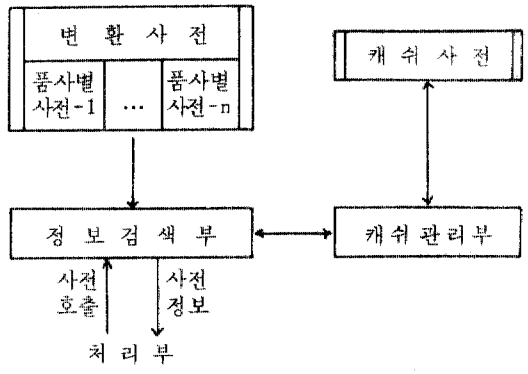
처리부는 사전부의 보조를 받아 자질구조로 주어진 원시언어 해석구조에 관련된 변환규칙을 단계적으로 적용하여 대응되는 목표언어 생성구조를 구성하며 디버깅(debugging)기능도 포함하고 있다.

지원도구부는 자질구조 조작도구를 포함하여 FEASTS의 전체적인 기능을 보조하는 기본적인 함수와, 사용자가 쉽게 내부의 특수한 자료구조를 작성 및 판독하도록 입출력을 지원하는 입출력도구들이 포함되어 있다.

Ⅲ. 사 전 부

그림 2는 사전부의 내부 구성을 보여준다.

변환사전은 변환에 필요한 각종 사전정보를 저장하고 있는 곳으로 이는 기계번역 전체의 사전을 총괄하여 관리하는 사전관리 시스템으로부터 번역사전의 한 뷰(view)로 제공되는 가상(virtual)의 논리적 사전이거나 아니면 변환부를 위해 독립적으로 구성된 물리적인 사



(그림 2) 사전부의 구성

전일 수가 있지만 여기에서는 그 물리적인 구성에 대해서는 고려하지 않고 다만 정보검색부가 참조하는 논리적 구성형식에 대해서만 살펴보기로 한다.

변환사전은 원시언어의 품사범주에 대응하는 품사별 사전으로 구성된다. 각 품사별 사전은 "사전 엔트리(entry)"로 구성되며 탐색의 키는 이 엔트리내의 "키워드(key word)"가 된다. 키워드는 하나의 품사별 사전내에서는 유일식별성(unique identifiability)을 가지므로 같은 품사의 동음이의어인 경우 하나의 키워드만 갖게되고 그들간의 구분은 다음에 설명될 규칙정보와 사전정보 기술로써 이루어져야 한다.

각 사전 엔트리는 그에 부속된 "사전정보 요소"로 구성되며 사전정보 요소는 "규칙정보"이거나 "어휘정보"가 될 수 있다. 규칙정보는 그 어휘에 해당하는 변환규칙의 호출형(calling form)을 기술하는 부분으로 규칙정보 식별자인 'R'과 호출한 "규칙이름"과 규칙호출시 전달될 "실인자(real argument)"로 구성된다. 어휘정보는 해당 키워드의 특성 정보를 기술하는 부분이며 이는 어휘정보 식별자 'I'와 어떤 어휘정보인가를 나타내는 "어휘정보범주" 그리고 이 어휘정보범

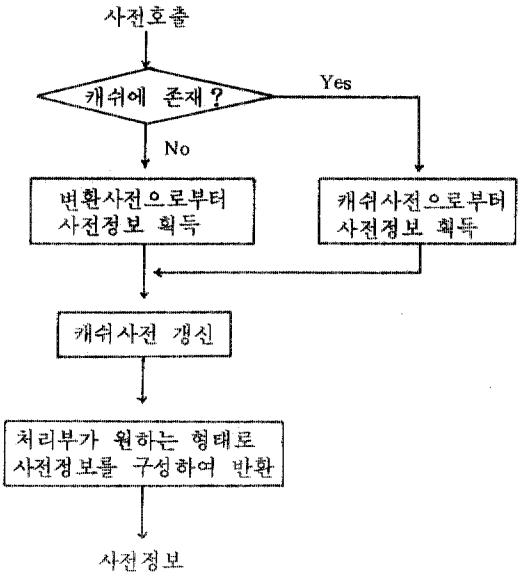
주에 해당하는 "어휘정보 내용"으로 구성한다.

"사전 엔트리" :: = ("키워드" { "사전정보 요소" })
"사전정보 요소" :: = "규칙정보" | "어휘정보"
"규칙정보" :: = (R "규칙이름" { " 실인자" })
"어휘정보" :: = (I "어휘정보범주" { "어휘정보 내용" })

FEASTS는 사전정보 검색의 효율을 높이기 위해 내부적으로 캐쉬사전(cache dictionary)을 유지하고 있다. 캐쉬사전은 일정 크기의내부 기억장소를 차지하고 있으며 구성요소들간에 양방향으로 연결된 해쉬테이블(hash table)을 사용하여 사전정보를 저장한다. 캐쉬사전의 탐색키는 품사별 사전이름과 키워드로 이루어지며 캐쉬사전의 각 엔트리는 자신의 탐색키와 전위 및 후위 엔트리를 가리키는 연결부와 사전정보로 구성된다. 캐쉬관리부는 검색이 수행된 엔트리를 캐쉬의 머리부(head)에 위치시킴으로써 가장 최근에 참조되지않은 엔트리를 항상 바닥부(bottom)에 유지하여, 캐쉬가 만원일 때 검색이 실패하면 이 바닥부의 엔트리를 방출하는 NUR(Not Used Recently)기법으로 캐쉬를 유지관리한다.

정보검색부는 처리부로부터 사전호출을 요청받아 처리부가 원하는 형태로 사전정보를구성하여 반환하는 기능을 수행한다. 처리부가 요청하는 사전호출의 내용은 품사별 사전이름 및 키워드와 규칙정보 식별자 혹은 어휘정보 범주인 정보범주로 구성된다. 정보범주가 R로 주어지면 이를 규칙정보 식별자로 인식하고, 사전정보 요소중에서 규칙정보를 찾아 규칙이름과 실인자로 규칙 호출형을 구성하여 반환한다. 정보범주가 R이 아니면 이를 어휘정보범주로 인식하고, 사전정보 요소중에서 어휘정보 식별자 I를 가지고 그 어휘정보범주가 주어진 정보범주와 일치하는 어휘정보 내용을 반환한다.

정보검색의 과정은 그림 3과 같이 먼저 캐쉬관리부를 통해 캐쉬사전으로부터 이루어지고 만약 캐쉬자전에서 실패하면 변환사전에서 검색을 시도한다. 일단 검색이 수행되고 나면캐쉬관리부는 캐쉬사전을 갱신하고 정보검색부는 그 사전정보를 처리부가 원하는 형태로 구성하여 반환한다.

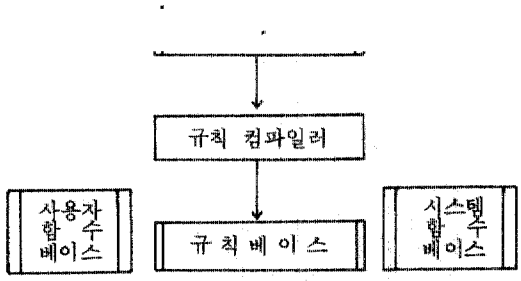


(그림 3) 사전정보 처리과정

이상에서 살펴본 바와 같이 FEASTS는 간단하면서도 일관된 사전정보 구성형식을 유지하여 비교적 사전 구축과 처리부와의 접속성이 용이하도록 지원하고 있으며, 캐쉬사전을 도입하여 검색의 효율을 증대시키고 있다.

Ⅳ. 규 칙 부

규칙부는 그림 4와 같이 구성되어 있다.



(그림 4) 규칙부의 구성

원시 규칙베이스(source rule-base)는 원시규칙 정의형식에 따라 사용자가 기술한 원시 변환규칙을 포함하고 있으며, 규칙 컴파일러(rule compiler)는 이들 원시규칙 정의를 리스프 함수 정의로 변환하고 관련된 문서화 내용을 모아 규칙베이스(rule-base)를 생성하는 기능을 수행한다. 시스템함수베이스(system function-base)는 다음에 설명될 일반 규칙기술자를 처리하는 데 사용되는 리스프함수 정의를 포함하며 이는 FEASTS에 의해 제공된다. 그리고 사용자함수베이스(user function-base)는 원시규칙 정의시에 그 규칙내에서 사용하기 위해 사용자가 정의한 리스프 함수 정의를 포함한다.

원시규칙 정의는 다음과 같이 "규칙이름", "형식인자(formal argument)", "국부변수(local variable) 선언부", "문서화 내용(documentation)" 및 "규칙기술자(rule descriptor)"로 구성한다.

"원시규칙 정의"::=("규칙 이름"
({"형식인자"} "국부변수 선언부")
"문서화 내용"
{"규칙기술자"})
"국부변수 선언부"::=∈ | &AUX
{"국부변수"}
"국부변수"::="변수" | ("변수" "초기값")
"규칙기술자"::="일반 규칙기술자" |
"함수 규칙기술자"
"일반 규칙기술자"::="제1피연산자"
"연산자" "제2피연산자"
"함수 규칙기술자"::="리스프 함수" |
"변수"

"규칙이름"은 변환사전의 규칙정보에서 그 규칙의 호출형을 기술하는 데 사용되며 유일한 이름을 지녀야 한다.

형식인자는 규칙실행시에 실인자를 전달받으며, 모든 규칙은 원시규칙 정의에 명시적으로 나타나지는 않지만 처리부에 의해 규칙처리시에 처리대상이 되는 원시 자질구조를 전달받는 특별한 형식인자 '%S'를 항상 암시적으로 지니고 있다.

국부변수 선언부는 그 규칙의 실행시에 규칙내에서 국부적으로 사용하기 위한 변수들을 선언하는 부분으로, 상수나 변수 혹은 리스프 함수로 주어지는 "초기값"을 지닐 수도 있다. 그리고 모든 규칙은 국부변수 선언부에는 나타나지 않지만 그 규칙의 실행결과인 목표 자질구조를 저장하는 특별한 국부변수 '%T'를 항상 암시적으로 지니고 있다.

문서화 내용은 그 규칙의 내용이나 특기사항을 기술하여 규칙실행시 참고하기 위해 사용되는 단일문자열(single string)로서 규칙의 문서화 기능을 도와준다.

규칙기술자는 "일반 규칙기술자" 이거나 "함수 규칙기술자"이며 처리부는 규칙실행시 이들 규칙기술자를 차례로 평가(evaluation)하여 모두 성공하게 되면 목표 자질구조

인 %T의 값을 반환하고 그렇지 못하면 평가를 중단하고 "NIL"을 반환한다.

함수 규칙기술자는 변수이거나 리스프 함수의 호출형이며 사용자는 리스프가 제공하는 일반적인 함수와 사용자함수베이스에 정의된 사용자 정의 함수 그리고 FEASTS의 지원도구부에서 제공하는 함수와 실제로 변환을 수행하는 'TRF' 함수를 사용할 수 있다. TRF는 자질구조를 인자로 취하여 현재 자질구조를 처리하는 것과 같은 방법으로 인자로 주어진 하위 자질구조를 변환하는 함수로서 FEASTS는 실제로 최외곽 자질구조에 이 TRF를 적용함으로써 변환을 시작한다.

일반 규칙기술자는 두 개의 피연산자를 취하는 중위 표현의 규칙기술자로서 같은 연산자라도 피연산자의 종류에 따라 다른 해석기능을 지니고 있으므로 먼저 FEASTS에서 사용되는 특수 구조에 대하여 설명하기로 한다.

FEASTS는 입출력과 규칙기술에 있어서 리스프의 일반적인 s-expression외에도 "자질구조", "경로구조(path structure)", "수의구조(optional structure)"를 지원하고 있다.

"자질구조" :: = 〔{"자질이름" "자질값"}〕
"경로구조" :: = 〈"자질구조"{"자질이름"}〉
"수의구조" :: = '{'{"수의적 요소"}'}'

자질구조는 통합기반문법에서의 자질구조를 나타내는 것으로, 같은 자질구조내에서는 유일한 "자질이름(feature name)"을 갖고 각 자질이름에는 하나의 해당 "자질값(feature value)"이 할당되며 이들 자질값은 다시 자질구조가 될 수도 있다. 그리고 자질값 중 공유되는 부분은 Common Lisp의 공유구조(shared structure)와 같은 방식으로 표현된다.[8,15] 경로구조는 자질구조나 이를 지시하는 변수를 머리(head)로 하고 1개 이상의 자질이름으로 이루어진 경로(path)를 몸체(body)로 하여, 주어진 자질구조내에서 해당 경로의 자질값을 지시한다. 수의구조는 수의구조에 나타난 "수의적 요소"들중의 하나임을 의미하는 구조이다. 자질구조와 경로구조는 규칙내에서 다른 곳에서도 사용될 수 있지만 수의구조는 일반 규칙기술자중에서 다음에 설명될 비교 연산자(compare operator)의 피연산자로만 사용가능하다. 그리고 이들 세 특수 구조는 일반 s-expression과 같이 입출력에 그대로 사용될 수 있다.

일반 규칙기술자의 연산자는 크게 치환 연산자(assignment operator)와 비교 연산자로 구분된다.

치환 연산자는 ' : = ' 하나가 존재하며 피연산자의 종류에 따라 다음과 같이 조금씩 다른 기능을 수행한다.

○ 제1피연산자가 일반 변수이고 제2피연산자가 일반 s-expression인 경우 : 제1피연산자의 변수값을 제2피연산자를 평가한 값으로 치환

○ 제1피연산자가 일반 변수이고 제2피연산자가 경로구조인 경우 : 제1피연산자의 변수값을 제2피연산자의 경로가 지시하는 자질구조내의 자질값으로 치환

○ 제1피연산자가 리스프의 치환 위치 표시자(car, cdr, nth 등)인 경우 : 제2피연산자의 종류에 따라 그 값을 제1피연산자가 지시하는 위치의 값으로 할당(리스프의 setf 기능)[8,15]

○ 제1피연산자가 경로구조인 경우 : 제2피연산자의 종류에 따라 그 값을 제1피연산자가 지시하는 자질구조내의 경로에 해당하는 자질값으로 할당

비교 연산자에는 '=', '==', '/=', '

/==' 의 네 종류가 있다. '=' 는 리스프의 'eq', '==' 는 'equal', '/=' 는 'not eq', '/==' 는 'not equal' 의 의미를 지니고 있지만 피연산자의 종류에 따라 같은 연산자라도 다음과 같이 조금씩 다른 기능을 수행한다.

o 두 피연산자가 모두 일반 s-expression 이거나 모두 경로구조인 경우 : 두 피연산자의 s-expression 을 평가한 값이나 경로구조가 지시하는 자질값을 연산자의 성질에 따라 비교 연산을 수행

o 한 연산자가 수의구조이고 다른 한 연산자는 일반 s-expression 이거나 경로구조인 경우 : 일반 s-expression 이나 경로구조의 값에 대해 수의구조의 원소 여부를 연산자의 성질에 따라 판단

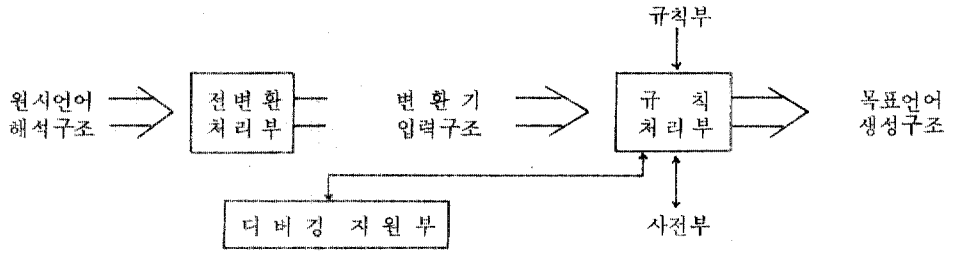
o 두 연산자가 모두 수의구조인 경우 : 두 수의구조의 수의적 요소들 사이에 공통부분이 존재하는가 여부를 연산자의 성질에 따라 판단

이상에서 살펴본 바와 같이 FEASTS는 비교적 간단한 규칙구문(rule syntax)과 함께 세가지 특수구조를 지원하고, 피연산자에 따라 조금씩 다른 기능을 수행하는 같은 의미의 연산에 대해서는 operator overloading 기법을 도입하여 동일한 연산자를 사용가능하도록 하는 일반 규칙기술자를 제공하고, 규칙 컴파일러로 하여금 이를 피연산자에 따라 적절히 해석하도록 함으로써 비교적 간단히 규칙을 기술할 수 있도록 해준다. 그리고 규칙내에서 리스프나 지원도구부의 함수뿐만 아니라, 사용자 함수베이스를 도입하여 사용자 정의 함수까지 사용가능하도록 함으로써 사용자로 하여금 보다 쉽게 표현력이 뛰어난 규칙을 기술할 수있도록 지원하고 있다.

V. 처 리 부

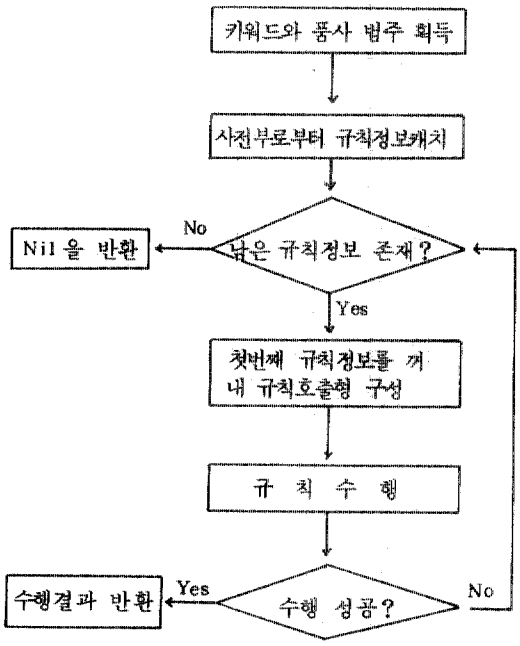
그림 5는 FEASTS 처리부의 내부 구성을 보여준다.

전변환 처리부는 원시언어 해석구조에 대해 FEASTS 내에서 특수하게 사용되는 자질값을 찾아 규칙 처리부의 직접적인 입력이 되는 보다 보강(augmented)된 자질구조인 변환기입력구조를 생성하는 기능을 담당한다. 특수자질에는 원자값(atomic value)을 취하는 'fnames' 와 'app-sub' 그리고 자질구조를 취하는 'mother' 와 'next' 가 있고 이들은 모두 규칙내에서 참조가 가능하다. fnames 는 원시언어 해석구조의 현재 자질구조에 나타난 모든 자질이름을 지니고 있으며, app-sub 는 이중에서 하위범주(subcategory)인 자질만 포함한다. mother 는 바로 외곽의 자질구조를 가리키며 next 는 병렬접속에서 이웃하는 다음 자질구조를 가리킨다.



(그림 5) 처리부의 구성

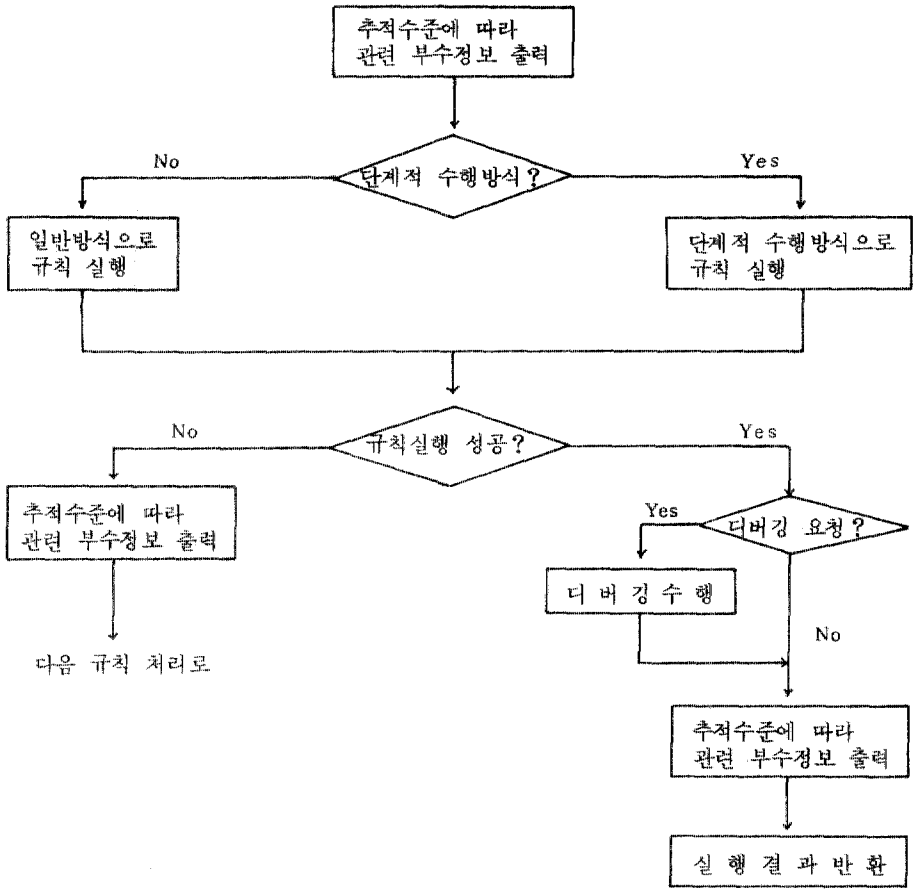
규칙 처리부는 그림 6과 같이 사전부의 보조를 받아 원시 자질구조에 규칙부의 해당 변환규칙을 적용하여 목표 자질구조를 생성한다. 먼저 현재 처리대상이 되는 원시 자질구조에서 키워드와 그 키워드의 품사범주를 획득하여 사전부로부터 그 키워드에 해당하는 규칙정보를 캐치(catch)한다. 다음 차례로 이들 각규칙에 대해 원시 자질구조를 형식인자 '%S'의 값으로 전달하여 규칙 호출형을 구성하고 이를 수행한다. 규칙이 성공적으로 수행되고 나면 그 수행결과인 목표 자질구조 '%T'의 값을 반환하며, 실패하면 남은 규칙이 없을때까지 수행을 계속하고 모두 실패할 경우는 NIL을 돌려주고 상위 자질구조로 백트래킹(backtracking)이 일어난다. 그리고 각 규칙 기술자를 평가할 때 TRF 함수를 만나면 부프로그램(subprogram)처리와 같이 그 단계의 자질구조 처리를 일단 중지하고 TRF의 인자로 주어진 하위 자질구조에 대해 동일한 방법으로 변환을 수행한 후 다음 규칙 기술자를 계속 평가한다.



(그림 6) 규칙 처리과정

FEASTS는 그림 7과 같이 규칙수행시 사용자가 디버깅을 할 수 있도록 단계적 수행방식(step mode)과 함께 디버깅 단계 및 추적(trace) 기능을 제공하고 있다. 단계적 수행방식과 디버깅 단계는 'on-off'형태의 플래그(flag)값을 조정함으로써 서로 독립적으로 작동될 수 있으며, 단계적 수행방식이 적용되면 Common Lisp의 'step mode'로 규칙이 실행되고[8,15], 디버깅 요청이 있으면 규칙 실행이 성공적으로 완료된 후 다음 단계로의 진행을 일단 중지하고 사용자로 하여금 그 규칙수행과 관련된 여러가지 부수정보를 조사할 수 있도록 디버깅 단계로 들어간다. 추적기능은 정수로 주어지는 추적수준(trace level)을 설정함으로써 조정될 수 있으며 FEASTS는 규칙수행마다 이 추적수준에 따라 관련 부수정보를 사용자에게 제공한다. 디버깅 단계와 추적수준에 따라 받아 볼 수 있는 부수정보에는 현재 처리중인 원시언어 입력구조, 변환기 입력구조, 원시 자질구조, 목표 자질구조와 규칙 및 관련된 문서화 내용과 사전정보 등이 포함되어 있다.

이상에서 살펴 본 바와 같이 FEASTS는 전체적으로 보아 하향식(top-down) 백트래킹 방식으로 변환규칙을 처리하며 규칙 디버깅을 위한 여러가지 기능을 제공하고 있다. 그림 8은 이러한 규칙 처리과정이 구체적으로 어떻게 이루어지는가를 가상적인 입력을 예로 들어 보여준다.



(그림 7) 규칙 수행과정

```
(R-v-1(ew1)
   〈%T PRED〉 : = ew1
   〈%T CAT〉  : = 〈%S CAT〉
   〈%T fn-1〉 : = (TRF 〈%S fn-1〉)
   〈%T fn-3〉 : = (TRF 〈%S fn-2〉))
(R-v-2(ew1)
   〈%T PRDD〉 : = ew1
   〈%T CAT〉  : = 〈%S CAT〉
   〈%S fn-2 NUM〉 : = 〈%S fn-1 NUM〉
   〈%T fn-1〉 : = (TRF 〈%S fn-1〉)
   〈%T fn-3〉 : = (TRF 〈%S fn-2〉))
(R-n-1(ew1)
   〈%T PRED〉 : = ew1
   〈%T CAT〉 : = 〈%S CAT〉
   〈%T NUM〉 : = 〈%S NUM〉))
```

a. 변환규칙

명사사전

```
( " kw2 " (R R-n-1 " ew2 "))
( " kw3 " (R R-n-1 " ew3 "))
```

동사사전

```
( " kw1 " (R R-v-1 " ew1 ")
          (R R-v-2 " ew1 "))
```

b. 변환사전

```
sf  --> (PRED "kw1"
         CAT VERB
```

```
        fn-1〔PRED "kw2"
             CAT NOUN
             NUM PL〕
        fn-2〔PRED "kw3"
             CAT NOUN〕〕
```

c. 변환기 입력구조

```
TRF(sf) ; 수행 시작
        ; kw1의 변환규칙 R-v-1적용
          〈%T PRED〉 : = "ew1" ; 성공
          〈%T CAT〉  : = VERB ; 성공
          (TRF 〈%S fn-1〉) ; 수행 시작
                ⋮
        ; 성공
          〈%T fn-1〉 = 〔PRED "ew2"
                       CAT NOUN
                       NUM PL〕
          (TRF 〈%S fn-2〉) ; 수행 시작
               〈%T PRED〉 : = "ew3" ; 성공
               〈%T CAT〉  : = NOUN; 성공
               〈%T NUM〉  : = NIL ; 실패
          (TRF 〈%S fn-2〉); 수행 전체 실패
                           -> 백트래킹
        ; kw1의 변환규칙 R-v-1적용 실패
        ; kw1의 변환규칙 R-v-2적용
          〈%T PRED〉 : = "ew1"; 성공
          〈%T CAT〉  : = VERB; 성공
          〈%S fn-2 NUM〉 : = PL; 성공
          (TRF 〈%S fn-1〉); 수행 시작
                ⋮
        ; 성공
          〈%T fn-1〉 : = 〔PRED "ew2"
                         CAT NOUN
                         NUM PL〕
          (TRF〈%S fn-3〉); 수행 시작
                          〈S fn-2〉 = = 〉
                          〔PRED "kw3"
                           CAT NOUN
                           NUM PL〕
                〈%T PRED〉: = "ew3"; 성공
                          : = NOUN ; 성공
                〈%T NUM〉 : = PL ; 성공
        ; 성공
          〈T fn-3〉 : = 〔PRED "ew3"
                       CAT NOUN
                       NUM PL〕
        ; kw1의 변환규칙 R-v-2적용 성공
        TRF(sf) = = 〉〔PRED "ew1"
                      CAT VERB
                      fn-1〔PRED "ew2"
                           CAT NOUN
                           NUM PL〕
                      fn-3〔PRED "ew3"
                           CAT NOUN
                           NUM PL〕〕
```

d. 변환규칙 적용과정과 수행결과

(그림 8) 규칙처리 과정의 예

Ⅵ. 결 론

본 논문에서는 실용화를 목표로 개발된, 도구형태의 단방향 자질구조 변환기 골격 시스템인 FEASTS의 구성과 변환방법에 대해 설명하였다. FEASTS는 비교적 작성이 용이하면서도 표현력이 뛰어난 규칙구문을 제공하고 이를 효과적으로 처리하기 위해 규칙 컴파일러를 제공하고 있다. 그리고 FEASTS는 효과적인 사전정보 저장 형식과 처리기법뿐만 아니라 디버깅 기능을 지닌 규칙처리 방법과 함께 여러가지 사용자 지원도구를 포함하고 있기 때문에, 변환사전과 변환규칙을 구축하여 실제로 양언어간의 변환부를 구성할 때 이를 용이하게 지원할 수 있다. 또한 FEASTS는 처리대상 언어에 독립적일뿐만 아니라 특정 자질구조에 제한을 두고 있지 않기 때문에,원시 자질구조로서 목표언어 생성구조를 취하고 이에 맞게 생성사전(generation dictionary)과 생성규칙(generation rule)을 구축하여 자질

구조로부터 목표문장(target sentense)을 생성하는 생성기(generator)로도 쉽게 전용될 수 있다.

현재 LFG(Lexical Functional Grammar)에 기초하여 변환사전과 변환규칙을 구축하면서 FEASTS에 대한 검증이 진행중에 있으며, 앞으로 대량의 규칙이 구축될 때 이를 용이하게 유지관리할 수 있도록 규칙관리 기법이 좀 더 보완되어야 할 것으로 생각된다.

참 고 문 헌

1. 강승식, 심광섭, 장병탁, 권혁철, 우치수, 김영택, "KSHALT: 영-한 기계 번역 시스템", 춘계 인공지능 학술발표회 논문집, 한국 정보과학회 인공지능 연구회, pp. 113-132, 1987.
2. 김재훈, 이진선, 정희성, "한영 양국어간의 위상구조 변환을 위한 Transfer 모델", '88 가을 학술발표논문집, 한국 정보과학회, Vol.15, No.2, pp.233-236, 1988.
3. 서영훈, "한국어 통사 분석기에 관한 연구", '87 가을 학술발표논문집, 한국정보과학회, pp.581-584, 1987.
4. 심광섭, "변환 방식에 의한 기계번역시스템에서의 변환기에 관한 연구", 석사학위 논문, 서울대학교 전자계산기공학과, 1988.
5. 이하규, "기계번역에서 트랜스퍼 모형과 구현", 석사학위 논문, 서울대학교 컴퓨터공학과, 1989.
6. Bresnan, J. *Lexical-Functional Grammar*, CSLI, 1987.
7. Hutchins, W.J. *Machine Translation: Past, Present, Future*, Ellis Horwood Limited, 1986.
8. Guy, L.S. *Common LISP*, Digital Press, 1984.
9. Karttunen, L. *D-PATR A Development Environment for Unification-Based Grammars*, CSLI, 1986.
10. Kudo, I. and NOMURA, H. "Lexical-Functional Transfer: A Transfer Framework in a Machine Translation System Based on LFG," Proceedings of COLING '86, pp.112-114, 1986.
11. Sag, I.A., Kaplan, R., Karttunen, L., Kay, M., Pollard, C., Shieber, S. and Zaenen, A. "Unification and Grammatical Theory", CSLI, 1987.
12. Shieber, S.M., Pereira, F.C.N., Karttunen, L., and Kay, M. *A Compilation of Papers on Unification-Based approaches to grammar*, CSLI, 1986.
14. Tucker, A.B. "Current strategies in machine translation research and development," in: *Machine Translation: Theoretical and methodological issues*, Nirenburg, S. (Ed.), Cambridge Univ. Press, pp.22-41, 1987.
15. Wilensky, R. *Common LISP craft*, W. W. Norton & Company, Inc., 1986.
13. Stuart, M.S. *An introduction to unification-based approaches to grammar*, CSLI, 1986.