영-한 기계 번역에서 조건 단일화 기반 변환 문법 해석기

김 남수*, 전 현경*, 박 영진**, 이 용석*

* 전북대학교 컴퓨터과학과 ** 정인대학 전산정보처리과

Transfer Grammar Compiler Based on Conditional Unification for English-Korean Machine Translation

Nam-Su Kim*, Hyun-Gyung Chon*, Young-Jin Park**, Yong-Seok Lee*

*Dept of Computer Science, Chonbuk National University

**Dept of Computer & Information Processing, ChongIn College

요 약

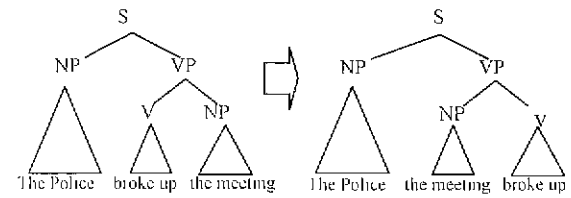
영-한 기계번역에 있어서 변환(Transfer)은 상이한 두 언어의 어순을 생성될 언어의 어순으로 결정하기 위한 변환 규칙에 의해서 영어의 구문 분석된 트리를 생성될 한국어의 구조에 맞게 재구성한다. 이러한 변환은 개발 단계 중 또는 그 후에 언어의 다양한 차이를 수용하기 위해 프로그램 수준에서 잦은 수정을 하게 된다. 이를 극복하기 위해서 본 논문에서는 변환 문법을 이용하여 좀더 체계적이고 확장이 쉬운 문법을 이용하여 변환을 수행하고자 한다. 이를 위해 영어의 구구조 자질들을 쉽게 검사 및 탐색하면서 조건에 맞는 자질들을 나누어 주는 기능을 하는 조건 단일화 연산자를 기반으로 일반적인 문맥자유문법(Context Free Grammar)을 이용한 변환 문법을 제안한다. 또한 변환 문법은 변환 문법 해석기에 의해 변환 프로그램으로 컴파일 됨을 보인다. 이러한 변환 방법은 영-한 변환에 있어서 문법 수준에서 변환 처리를 할 수 있어 변환 프로그램의 개발 및 유지보수에 많은 도움을 준다.

1. 서 론

영-한 기계 번역에 있어서 변환(transfer) 방식이란 분석(analysis), 변환(transfer), 생성(generation)의 세 단계를 통해 번역이 진행 되는데 변환 단계에서는 영어와 한국어의 어휘적 차이 및 구조적 차이를 해소하여 영어에 대한 중간 표현을 한국어의 생성에 적합한 중간표현으로 대응시킨다.

변환방식의 기계번역은 변환 과정에서 두 언어간의 용법이나 의미적 차이를 처리해 줄 수 있기 때문에 언어체계가 상이한 두 언어(예를 들어 영어와 한국어, 영어와 일본어)간의 기계번역에서 용이하다[1,2].

예를 들어 "The police broke up the meeting."의 분석된 영어 구조와 한국어 구조간의 어순을 트리 형태로 살펴보면 다음과 같다.



[그림 1] 예문에 대한 영어-한국어 구구조

변환 단계에서 이러한 상이한 언어군의 어휘와 구조적 차이점을 조절하여 자연스러운 한국어를 얻기 위하여 규칙 기반과 예문 기반, 숙어 기반 등의 영한 기계번역이 시도되었다[2, 3, 4]. 규칙 기반 기계 번역은 대량의 복잡한 변환 규칙의 개발과 확장성의 어려움을 가지며, 예문에 기반한 기계번역은 예문 데이터 베이스로 부터 원시 언어의 구구조와 가장 근사한 예문을 선택하고, 선택된 예문에 표현된 변환 정보를 이용하여 변환시키는 방법이다. 이러한 방법은 특정 영역(domain)의 번역, 많은 예문에 따른 속도 저하 및 자동 예문 추출 과정의 어려움을 가진다[5]. 또한 언어의 다양한 차이점들을 수용하여 구조 변환을 하기 위해 변환의 개발 단계 또는 개발 후 프로그램 수준에서 많은 수정을 하게 된다.

따라서 본 논문에서는 이러한 변환 처리를 좀더 체계적으로 수행하기 위해 정규화된 툴로서 지원할 수 있는 조건 단일화 기반의 변환 문법 및 변환 문법 해석기를 제안한다. 즉 변환 처리를 위해 변환 문법을 기술한 후 해석기에 의해 생성된 변환 프로그램을 이용하여 변환이 수행되도록 하는 방법이다.

이러한 방법은 변환 시스템에서 어순의 차이를 프로그램 수준에서 처리하지 않고 단지 어순을 고려한 변환 문법만 개발하여 변환 문법 해석기에 의해 생성된 프로그램을 실행함으로써 구구조 변환이 가능하므로 기계번역에서 변환 모듈의 개발 및 유지보수에 많은 도움을 준다.

2. 조건 단일화 기반 변환 문법

본 논문에서 제안한 변환 문법은 일반적인 CFG(Context Free Grammar)문법 형식에 조건 단일화 연산자들의 기능들을 확장(augment)시켜 다음과 같은 정형화된 틀을 사용하고 있다.

(<A> -> (<B> <C>)

```
( (조건 단일화식)
  (조건 단일화식)
  ...        ))
```

<변환 문법 해석기를 위한 문법 표현 형식>

문법기술의 편의를 위해 규칙의 왼쪽 구성소부터 차례로 x0, x1, x2, 등으로 표시한다 위의 문법의 처음 세 요소(<A> → <B> <C>)는 문맥자유형식의 문법 규칙이다 이러한 변환 문법은 입력된 트리의 자질 구조를 어떻게 문법적인 구조로 나누어야 하는지를 기술한 것이다

2.1 조건 단일화 연산자

단일화란 두 자질구조 안의 정보를 포함하는 최소한의 정보를 가지는 자질구조를 찾는 연산을 말한다 또한 조건 단일화는 자질구조 안의 특정 속성값을 동적으로 검사한 후 각 문장 성분이 가지는 문법적 관계를 검사하여 조건에 맞는 단일화만을 수행한다는 전략이다[6]

조건 단일화 표현 방식은 아래와 같으며, 조건 단일화를 수행하기 위한 몇 가지 연산자들의 기능은 다음과 같다

조건 단일화의 예	실제 예
(1) PATH=PATH	(x1 = (x0 np))
PATH=ATOMS	((x1 func) = subj)
(2) PATH==PATH	(x1 == (x0 vp np))
(3) PATH=cATOMS	((x0 subcat) =c T1)
(4) *DEFINED*	((x0 np) = *DEFINED*) ..
(5) *UNDEFINED*	((x0 np) = *UNDEFINED*)
(6) *OR*	*OR* (((x0 subcat) =c T1)) (((x0 subcat) =c I0))

[그림 2] 조건 단일화의 표현방식

[그림 2]의 실제 예에 있는 x0 는 [그림 1]과 같이 분석된 영어 구구조 NP 와 VP 의 자질 구조로 구성되어 있다.

(1) **=** : x1 에 값이 존재하면 새로운 값과 단일화를 수행한다 [그림 2]에 나열된 예제 중 '(x1 = (x0 np)) 는 변환에서 x0 의 자질 값 'NP'를 x1 에 복사하는 역할을 하게 된다.

(2) **==** : 값을 복사 후 삭제(이동)의 기능을 한다. '(x1 == (x0 vp np))'는 'VP'내의 NP'를 x1 에 복사 후 x0 의 해당 'NP'를 삭제한다 결과적으로 x0 의 'VP'에는 'V'만 존재하게 된다

(3) **=c** 해당 값이 존재하면 조건은 성공하여 다음에 기술된 연산자들을 수행하게 된다

(4) ***DEFINED*** 해당 값이 존재하면 성공하게 된다. 위의 예제에서 '(x0 np)'가 존재하면 참이 되어 다음에 기술된 단일화 연산자들을 실행하게 된다.

(5) ***UNDEFINED*** 해당 값이 존재하지 않으면 성공하게 된다. 위의 예제에서 '(x0 np)'가 존재하지 않으면 참이 되어 다음에 기술된 단일화 연산자들을 실행하게 된다

(6) **'OR*** 단일화 제약의 이점기술은 단일화를 선택적으로 실행하도록 하는 연산자이다 예를 들어, 위의 예제에서 x0 의 subcat(동사의 하위범주)이 자동사(I0) 또는 타동사(T1)인지의 여부에 따라 서로 다른 조건 단일화 연산자들이 실행된다.

이러한 조건 단일화 연산자는 분석된 영어 구구조의 자질들을 쉽게 체크 및 탐색하면서 조건에 맞는 자질들을 나누어 주는 기능을 가지고 있다

2.2 변환 문법

다음은 앞에서 설명한 조건 단일화 연산자와 CFG 문법 형식으로 서론에서 기술된 예제 "The police broke up the meeting "을 처리하기 위한 변환 문법이다.

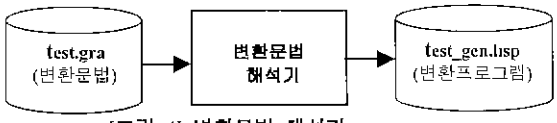
```
(<SENTENCE> → (<SUBJ> <OBJ> <PRED>)
    ( (x0 subcat) =c T1)
        (x1 = (x0 np))
    (x2 == (x0 vp np))
    (x3 = (x0 vp v)) ))

(<SUBJ> → (<DET> <N>)          (<OBJ> → (<DET> <N>)
    ( (x1 = (x0 det))              ( (x1 = (x0 det))
      (x2 = (x0 np)) ))              (x2 = (x0 np)) ))

(<DET> → ( @DET )              (<N> →  ( @N )
    ( (x1 = x0) ))                 ( (x1 = x0) ))

(<PRED> → ( @PRED )
    ( (x1 = x0) ))
```

[그림 3] 예문의 변환을 위한 문법

위의 문법은 목적어로 명사구를 갖는 타동사구를 변환하는 문법이다 변환의 입력인 구문분석된 결과는 x0 인 <SENTENCE>에 입력되며 문장의 주어가 되는 자질인 (x0 np)가 x1 으로 복사되고, 목적어가 되는 자질인 (x0 vp np)는 x2 로 복사된 후 (x0 vp np)는 삭제된다. 또한 동사인 (x0 vp)에 남아있는 v 를 x3 에 복사하게 된다. 이러한 조건 단일화 연산자 및 변환 문법에 의해 영어의 구구조를 자연스럽게 한국어의 구구조로 바꾸게 되는 것이다

3. 변환문법 해석기

[그림 4]와 같이 변환문법 해석기는 변환 문법들을 컴파일하여 직접적으로 호출하여 실행할 수 있는 LISP 함수들로 생성한다



[그림 4] 변환문법 해석기

[그림 3]의 문맥자유형식의 문법규칙을 위한 번역 방법은 기존에 여러 연구들에서 발표되어 왔다[7, 8] 그리고 조건 단일화 연산자식의 해석은 다음과 같은 함수들을 통해 변환 프로그램으로 생성되어 진다.

```
(setq equation (cdddr  statements))
(defun compile-equation (equation)
(dolist (process unify_equations))
  (let ( )
      (case (cadr process)
        (=
          ((pathp (caddr process))
           '(P=P ,(list (car process)) ,(list (caddr process)))
           ((or (atomp (caddr process) (atomp (car process)))
           '(P=A ,(list (car process)) , (list (caddr process)))
        (=C
          '(P=CA ,(list (car process)) ,(list (caddr process)))
))))
(defmacro P=P (path1 path2) '(setq x (path=path ',path1 ',path2)))
(defmacro P=A (path1 path2) '(setq x (path=atoms ',path1 ',path2)))
(defmacro P=CA (path1 path2) '(setq x (path=catoms ',path1 ',path2)))
```

[그림 5] 문법 해석을 위한 함수의 예

위의 [그림 5]에서 'statements'는 주어진 하나의 문법 표현이고, 'equation'은 조건 단일화식이 할당될 때, 그 안의 각 단일화 식 중 두 번째 요소가 어떠한 종류의 단일화 연산자 인지 식별하여 해석함수가 생성하게 된다. 여기서 LISP 의 'defmacro'

를 사용하여 함수가 평가되지 않고, 결과 함수 내에 코드들이 삽입될 수 있도록 하였다. 또한 기본적인 트리 단일화에 대한 알고리즘은 [9]을 사용하였다

[그림 5]의 함수를 통해 [그림 3]의 변환문법을 해석한 결과는 아래와 같은 형태의 LISP 함수가 생성된다.

```
(DEFUN  TRANS-FIRST (X)
 ( TRANS-SENTENCE X))

(DEFUN  TRANS-SENTENCE (X0)
 (분석된 영어 구구조의 자질들을 x0 에 초기화)
 (AND (P=CA (X0 SUBCAT) T1)
      (P=P (X1) (X0 NP))
      (P==P (X2) (X0 VP NP))
      (P=P (X3) (X0 VP V)))

  (LET ((RESULT-STRING ""))
    (AND (LET ((TEMP NIL))
         (SETQ TEMP (APPLY (QUOTE TRANS-SUBJ)
                   (LIST (GETVALUE FS (QUOTE (X1))))))
         (IF TEMP (SETQ RESULT-STRING
          (CONCATENATE (QUOTE LIST)  RESULT-STRING
           (IF (EQUAL (LENGTH TEMP) 1) TEMP (LIST TEMP))))
         (PROGN (SETF RESULT-STRING "") NIL)))

           (LET ((TEMP NIL))
           (SETQ TEMP (APPLY (QUOTE TRANS-OBJ)
              (변수 x2 를 위와  같은 형식으로 처리)

           (LET ((TEMP NIL))
           (SETQ TEMP (APPLY (QUOTE TRANS-PRED)
             (변수 x3 를 위와  같은 형식으로 처리)

(DEFUN  TRANS-SUBJ (X0)
  (P=P (X1) (X0 DET))
  (P=P (X2) (X0 NP N))

(DEFUN  TRANS-OBJ (X0)
  (P=P (X1) (X0 DET))
  (P=P (X2) (X0 NP N))
```

[그림 6] 변환문법 해석기에 의해 생성된 함수들의 예

[그림 6]과 같이 변환문법 해석기는 각각의 LHS 심볼을 호출할 수 있는 형태의 함수로 생성하고, 생성된 LHS 심볼의 함수는 RHS 심볼의 함수를 순서대로 호출하여 실행 후 변수 *(RESULT-STRING)*에 결과를 저장하여 리턴하는 구조로 되어있다

또한 확장된 조건 단일화 연산자는 해당 LHS 심볼의 함수 내에 포함된다. 이러한 함수는 입력문장을 변환하기 위해 변환문법 해석기에 의해 생성된 프로그램을 수행하면 단일화식을 위한 함수가 호출되어 단일화가 실행된다

4. 변환문법에 의한 영-한 구조 변환의 예

본 장에서는 분석된 영어의 구구조 트리를 조건 단일화 기반의 변환 문법을 이용한 영-한 구조 변환의 예를 보인다.

예를 들어, 1 장에서 언급된 예문 "The police broke up the meeting"의 구문분석 결과는 다음과 같다

```
((S ((NP   ( (DET ((LEX THE) (POS DET) (ROOT THE). .))
             (NP   ((N   ((LEX POLICE) (POS N)
                                  (ROOT POLICE)  ))
      ))))
      (VP ((V ((LEX BROKE_UP) (POS V) (ROOT BREAK_UP)  ))
           (NP ((DET ((LEX THE) (POS DET) (ROOT THE)  ..))
                (NP ((N ((LEX MEETING) (POS N)
                                  (ROOT MEETING) ..)))))
      ))
 (HEAD BREAK_UP) (TENSE PAST) (SUBCAT T1)  ))
(HEAD BREAK_UP) (SUBCAT T1) (TENSE PAST)  )))
```

[그림 7] 예문의 구문분석 결과

위의 영어 구구조를 입력으로 앞에서 설명한 [그림 3]의 조건 단일화 기반의 변환문법을 해석기에 의해 생성된 프로그램을 통해 변환을 실행하면 다음과 같은 결과가 생성된다

```
( (((@DET   ((LEX THE) (POS DET) (ROOT THE)  ))
    (@N     ((LEX POLICE) (POS N) (ROOT POLICE)  )))
   ((@DET   ((LEX THE) (POS DET) (ROOT THE)  ))
    (@N     ((LEX MEETING) (POS N) (ROOT MEETING)  )))
   (@PRED   ((LEX BROKE_UP) (POS V) (ROOT BREAK_UP)
             (TENSE PAST) (SUBCAT T1)  )) )
```

[그림 8] 예문에 대한 변환 프로그램의 결과

생성된 구조를 살펴보면, 전체 구조는 3 개의 구성요소로 되어 있는데 첫번째는 주어인 "the police"와 두번째는 목적어인 "the meeting", 세번째는 동사인 "broke up"으로 구성되어 위의 *S-V-O* 의 영어 구조를 한국어의 어순인 *S-O-V* 으로 어순변환이 이루어진다

5. 결 론

본 논문에서는 구구조 변환을 위한 조건 단일화 기반의 변환 문법과 변환 문법 해석기를 보였다.

영어의 구구조를 한국어 구구조로 재구성하기 위한 변환 프로그램을 개발하는 데에는 많은 어려움이 있다 그러나 조건 단일화 기반 변환 문법과 변환 문법 해석기를 이용한 구구조 변환은 단지 변환 문법만을 수정 및 보완하여 이를 해석기 I 에 의해 생성된 프로그램으로 처리를 할 수 있어 문법 수준에서의 변환 처리가 가능하기 때문에 유지보수에 많은 도움을 준다 또한, 일반적인 CFG 문법에 조건 단일화 문법을 확장했으므로 영어 구구조를 탐색하면서 조건 단일화 연산자에 의해 한국어 구구조로 직접적인 변환이 가능하다.

이러한 방법으로 영어 97 문장 (영어 동사의 하위범주 (subcategory)별로 단문 약 41 문장, 중문 약 35 문장, 복문 약 21 문장) 을 실험해 보았다 실험을 통해 단문 및 중문의 어순 변환이 자연스럽게 이루어 졌으며, 복문에서는 접속사의 종류에 따라 위치가 달라지는데, 이는 조건 단일화 연산자에 의해 처리가 가능하였다

참고문헌

[1] 옥철영, 김영백, "구길 변환을 위한 한영 동사 사전 구성," 한국정보과학회 논문지, Vol 19, No 5, 1992

[2] 이현아, 이공주 김길창, "자연스러운 번역을 위한 두 단계 영-한 변환시스템," 한국정보과학회 학술발표논문집, Vol 24 No 2 1997

[3] 김종혁, "한-영 기계번역에서의 예문을 이용한 변환," 한국과학기술원 석사학위 논문 1995

[4] 이호석, 김영백, "영어-한국어 기계번역을 위한 연어와 숙어 트랜스퍼 사전" 정보과학회 논문지, Vol 20, No.7, 1993

[5] 심광섭, 김영택, "기계번역시스템," 정보과학회지 12-8, 1994.

[6] 황이규, "확장 PATRII 문법 번역기의 구현" 전북대학교 석사학위 논문, 1995

[7] Aho, A V and J. D Ullman, "Compilers Principles, Techniques, and Tools," Addison Wesley, 1985

[8] Tomita M "Efficient Parsing for Natural Language A Fast Algorithm for Practical Systems," Kluwer Academic Publishers, 1989

[9] Tomita M, Knight K "Pseudo-Unification and Full-Unification" CMU-CMT-88-MEMO, 1987