國語 읽기敎育을 위한 專門家 시스템의 設計 및 具現

文 授 悅*1) 韓 判 岩**2)

The Design & Implementation of Expert System for Korean-Pronunciation Education

Moon Soo-Yeal and Han Pan-Am
Department of Computer Science, Kyungnam University

要 約

本 硏究에서는 우리말의 읽기와 쓰기가 다른점에 착안하여 쓰기를 入力하면 소리나는대로 出力하는, 그리고 그에 해당되는 法則과 推論過程을 提示하는 專門家 시스템을 Lisp言語를 사용하여 具現하였다.

하지만 여기에서는 한글자체가 입력되지 않아 英文으로 入力시켜 具現된 것은 追後에 개선해야 할 과제로 지적되며, 아울러 出力도 한글 풀어쓰기로 출력된 점도 지적하는 바이다.

그리고 本 硏究에서 具現된 내용은 읽기敎育과 관계된 각종 法則中의 일부임을 밝혀두고, 여기서는 그 可能性에 대한 例만 提示하였다. 또한 각 法則들의 例外가 많아 시스템을 具現하는데 너무 방대해질 수 있어 例外인 단어는 具現되지 않았음도 밝혀둔다.

第 I 章 序 論

일반적으로 어떤 사람이 Computer 보조교육(CAI:Computer Aided Instruction)에 있어서 인공지능 기법을 응용한 컴퓨터 프로그램을 지능적 컴퓨터 보조교육(ICAI:Intelligence Computer Aided Instruction)이라고 부른다.

컴퓨터 보조교육은 1980년대 부터 인공지능 기법을 적용시켜, 초기에는 주로 특별히 제한된 분야에서 적용 가능한 것만 연구되었으나, 최근에는 보다 보편적이고 일반적으로 컴퓨터교육에 지능(Intelligence)을 부여한 지능적 컴퓨터 보조교육의 구현에 관심이 집중되고 있다.

따라서 본 연구에서는 지식의 표현이 비교적 쉽고, 구현이 간단한 생성시스템(Production System)에 의한 추론방법을 사용하여 국어 읽기교육 전문가 시스템을 구현하였다.

본 연구의 목적은 국어 읽기교육에서 읽기와 쓰기가 다른 단어들을 쉽게 이해하도록 하였으며, 발음시에 표준발음을 할 수 있도록 하였고, 또 그 추

*慶南 大學校 大學院 電算科
**慶南 大學校 電子計算學科 敎授

론과정을 제시하여 이유를 설명함으로써 학생들로 하여금 결과에 대한 신뢰성을 갖고 국어 학습을 할 수 있도록 하였다.

第II章 國語 文法의 特性

본 연구에서 사용된 읽기교육을 위한 국어의 문법적 특성을 각 법칙들에 의해서 살펴보면 다음과 같다.

① 末音法則

우리말에서 音節의 끝소리가 되는 자음은 'ㄱ, ㄴ, ㄷ, ㄹ, ㅁ, ㅂ, ㅇ'의 일곱 소리뿐이다. 따라서 이 일곱소리 이외의 子音이 음절 끝에 오면 그것은 이 일곱 子音 중의 하나로 바뀌는 현상을 末音法則이라 한다.

예) 잎-->[입], 부엌-->[부억],

② 連音法則

音節의 끝에 子音을 가진 形態素가 母音으로 시작되는 形式 形態素, 또는 實質 形態素와 만나면 그 끝 子音이 다음 音節의 첫소리로 發音되는 現象을 連音法則이라고 한다.

예) 實質 形態素일 경우: 옷안-->[온안]-->[오단]

形式 形態素일 경우: 옷이-->[오시]

③ 子音同化

音節 끝자음이 그 뒤에 오는 子音과 만날 때, 어느 한쪽이 다른쪽 음을 닮아서 그와 비슷하거나 같은 소리로 바뀌기도 하고, 양쪽이 서로 닮아서 두 소리가 다 바뀌기도 하는 現象을 子音同化라고한다.

예) 앞날-->[압날]-->[암날], 국물-->[궁물], 섭리-->[섭니]-->[섬니]

④ 口蓋音化

끝소리가 'ㄷ, ㅌ'인 形態素가 모음 'ㅣ'나 반모음 'ㅣ'로 시작되는 形式 形態素와 만나면 口蓋音 'ㅈ, ㅊ'이 되는 現象을 口蓋音化라고한다.

예) 해돋+이-->해돋이-->[해도디]-->[해도지]

굳+이-->굳이-->[구디]-->[구지]

닫+히+다-->닫히다-->[다티다]-->[다치다]

⑤ 頭音法則

'ㄹ'과 'ㅇ'이 단어의 첫소리로 쓰이지 않고, 'ㄴ'이 모음 'ㅣ'나 반모음 'ㅣ' 앞에서 쓰이지 않는 現象을 頭音法則이라고 한다.

예) 락원(낙원)>낙원, 력사(역사)>역사, 녀자(여자)>여자

⑥ 硬音化現象

두 개의 안울림소리가 서로 만나면 뒤의 안울림소리가 된소리로 발음되는 현상을 된소리현상, 또는 硬音化現象이라 한다.

예)먹+고-->[먹꼬], 밥+도-->[밥또]

⑦ 사잇소리現象

두 개의 형태소 또는 단어가 합성명사를 이룰때, 첫음절의 終聲이 울림소

리(ㄴ ㅂ ㅁ ㅇ)이고 뒤의 初聲이 안울림소리이면 뒤의 初聲이 된소리로 나는 現象을 사잇소리 現象이라 한다.

예)김밥-->[김빱], 봄비-->[봄삐], 산길-->[산낄]

第Ⅲ章 專門家 시스템의 槪要

1. 전문가 시스템의 구조

전문가 시스템에서 가장 핵심은 지식에 있다. 전문가 시스템의 지식은 문제를 어떻게 푸는가에 대한 지식과 사용자와 상호작용하는 방법에 대한 지식과는 별도로 문제 영역에 대한 지식을 분리하여 구성된다. 전자의 문제 해당 분야에 관한 지식을 모아 둔 것을 지식베이스라 하고 일반적인 문제 해결에 관한 지식을 추론엔진이라고 한다.

2. 전문가 시스템의 지식표현

전문가 시스템의 지식표현은 여러가지가 있는데 그 중에서 몇가지를 나열하면 다음과 같다.

1) 생성시스템(Production System)에 의한 표현

이 방법은 현재의 문제가 가지고 있는 상황이 조건문의 IF 조건 부분을 만족시키면 THEN 부분의 행위(Action) 부분이 실행된다. 생성 시스템의 지식은 "If Condition, Then Action"의 독립적 형태로 저장, 독립적인 Action의 Set로 표현될 수 있는 처리(Process)나 표현(Representation)과 제어(Control)가 쉽게 분리될 수 있는 문제에 적합한 지식의 표현방법이다.

2) 의미망(Semantic-Net)에 의한 표현

이 방법은 객체(Object)나 개념(Concept), 상황(Situation)등을 표시하는 노드(Node)들과의 관계를 묘사하는 아크(Arc;Link)들로 이루어진다. 노드는 객체, 개념, 상황을 나타내고 아크는 노드사이의 관계를 나타낸다.

Semantic-Net에 의한 표현은 술어논리(Predicate Logic)와 비교해 보면 표현이 자연스럽고 이해하기 쉽다는 장점이 있지만 화살표 반대 방향으로 추론하기 위해 Backpointer를 만들어 주어야하는 번거로움이 있다.

3) 프레임(Frame)에 의한 표현

Frame은 Semantic-Net에 의한 표현과 아주 유사하지만 Frame의 각 노드의 개념은 속성(Attribute)들과 속성값의 모임으로서 정의되는데, 이때 속성을 슬롯(Slot)이라 한다. 각 슬롯은 슬롯의 정보가 변할 때 마다 실행할 수 있도록 마련되어진 Procedure를 가지고 있다.

Frame에 의한 표현은 데이터의 내용과 형식에 대한 예상이 문제풀이에 중요한 역할을 하는 영역, 예를 들면 시각장면 해석 또는 말의 이해에 잘 이용된다.

第Ⅳ章 읽기 敎育을 위한 專門家 시스템의 設計 및 具現

第1節 專門家 시스템의 設計

1. 읽기敎育의 規則 生成

생성시스템은 기본적으로 조건(Condition)과 행동(Action)의 양쪽에 근거를 둔 지식 표현방법으로 새로운 사실들을 추론해 내는 것을 말한다.

본 연구에서는 7가지의 읽기 교육과 관계된 규칙을 사용하고 있는데, 그 중에서 한 가지만 예를 들어 표현해 보면 다음과 같다.

Ex) 말음규칙-Rule(RULE-2)

1) RULE-2-1
   IF : 입력단어에서 첫음절의 종성이 ㅍ, ㅄ 이면
   THEN: 첫음절의 종성이 ㅂ으로 바뀐다.
2) RULE-2-2
   IF : 입력단어에서 첫음절의 종성이 ㅅ, ㅆ, ㅈ, ㅊ, ㅌ 중의 하나이면
   THEN: 첫음절의 종성이 ㄷ 으로 바뀐다.
3) RULE-2-3
   IF : 입력단어에서 첫음절의 종성이 ㄲ, ㅋ, ㄺ, ㄳ 중의 하나이면
   THEN: 첫음절의 종성이 ㄱ 으로 바뀐다.
4) RULE-2-4
   IF : 입력단어에서 첫음절의 종성이 ㄻ, ㄽ, ㄾ 중의 하나이면
   THEN: 첫음절의 종성이 ㄹ 로 바뀐다.
5) RULE-2-5
   IF : 입력단어에서 첫음절의 종성이 ㄵ 이면
   THEN: 첫음절의 종성이 ㄴ 으로 바뀐다.
6) RULE-2-6
   IF : 입력단어가 위의 규칙중에 적용되는 규칙이 있으면
   THEN: 말음법칙이 성립한다.

위의 규칙에 적용되어 새로 생성된 사실은 저장되고, 그 다음의 단어가 또 적용되어 입력단어의 끝까지 적용이 반복되는 것이다.

이것은 Lisp 언어의 되부름(Recursion)형식을 사용하면 간단히 처리할 수 있으며, 아래의 예는 위의 규칙들을 실제 Lisp 언어로 작성한 것이다.

```
(defun RULE-2 (DATA)
  (cond ((equal DATA nil) nil)
        ((member (caddr DATA) '(ㅍ ㅄ))
              (setq *R-2*
                 (cons (car DATA)
                   (cons (cadr DATA)
                     (cons 'ㅂ nil)))))
        ((member (caddr DATA) '(ㅅ ㅆ ㅈ ㅊ ㅌ))
```

```
                (setq *R-2*
                   (cons (car DATA)
                     (cons (cadr DATA)
                       (cons 'ㄷ nil)))))
          ((member (caddr DATA) '(ㄲ ㅋ ㄺ ㄳ))
                (setq *R-2*
                   (cons (car DATA)
                     (cons (cadr DATA)
                       (cons 'ㄱ nil)))))
          ((member (caddr DATA) '(ㄵ ㄼ ㄾ))
                (setq *R-2*
                   (cons (car DATA)
                     (cons (cadr DATA)
                       (cons 'ㄴ nil)))))
          ((member (caddr DATA) '(ㅄ))
                (setq *R-2*
                   (cons (car DATA)
                     (cons (cadr DATA)
                       (cons 'ㅂ nil)))))
          (T (setq *R-2*
                   (cons (car DATA)
                     (cons (cadr DATA)
                       (cons (caddr DATA) nil))))))
    (cond ((equal DATA nil) nil)
          (T (setq *R-2*
                   (append *R-2*
                     (RULE-2 (cdddr DATA)))))))
```

위의 마지막에 있는 COND 함수는 단어의 한 음절을 적용한 뒤에 그 다음을 적용하여 입력단어의 마지막까지 적용하기 위한 것이다.

2. 生成 시스템에 의한 設計

전문가 시스템의 여러가지 지식표현 방법중에서 쉽게 구현이 가능하고 추론과정이 비교적 효율적인 방법중의 하나인 생성 시스템에 의한 설계는 다음과 같이 크게 3부분으로 나누어져 있다.

1) 생성 규칙들의 집합인 규칙베이스
2) 새로운 사실들을 추론하는 추론엔진
3) 사실들의 집합인 Working Memory

이것을 그림으로 나타내면 아래 그림 4-1과 같고, 본 연구에서 설계한 것을 설명하면 다음과 같다.

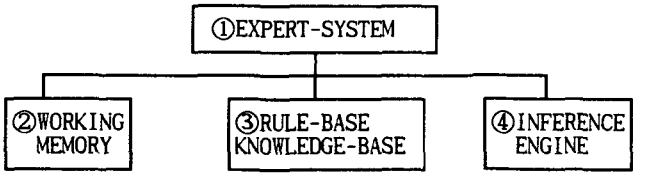

그림 4-1 생성 시스템의 전체 구성도

① 시스템의 표제부분(Expert-System)

이 부분은 읽기교육을 위하여 화일을 인터프리터내에 실행하기 위한 표제(Title) 부분이다.

② 작업영역(Working Memory)

이 부분은 입력단어를 규칙에 적용시켜서 새로 생성된 것을 임시 저장하는 부분이다. 이 시스템에서는 *R-1*, *R-2*, *R-3*, *R-4*, *R-5*, *R-6*, *R-7* 등이 해당된다.

③ 규칙베이스(Rule-Base=Knowledge-Base)

이 부분은 국어 읽기교육에서 적용될 각 부분의 법칙들이 내장되어 있다. 이 시스템에서는 두음법칙-RULE, 말음법칙-RULE, 연음법칙-RULE, 자음동화-RULE, 구개음화-RULE, 경음화현상-RULE, 사잇소리현상-RULE 등이 설계되어 있다.

④ 추론엔진(Inference Engine)

이 부분은 인터프리터로 입력된 단어를 실제 규칙에 적용이 될 수 있도록 시스템을 구동시키는 역할을 하며, 원하는 결과를 출력할 수 있도록 시스템 전체를 총괄하는 부분이다.

이상에서 설계한 내용을 시스템 순서도(Flowchart)로서 나타내면 다음 그림 4-2와 같다.

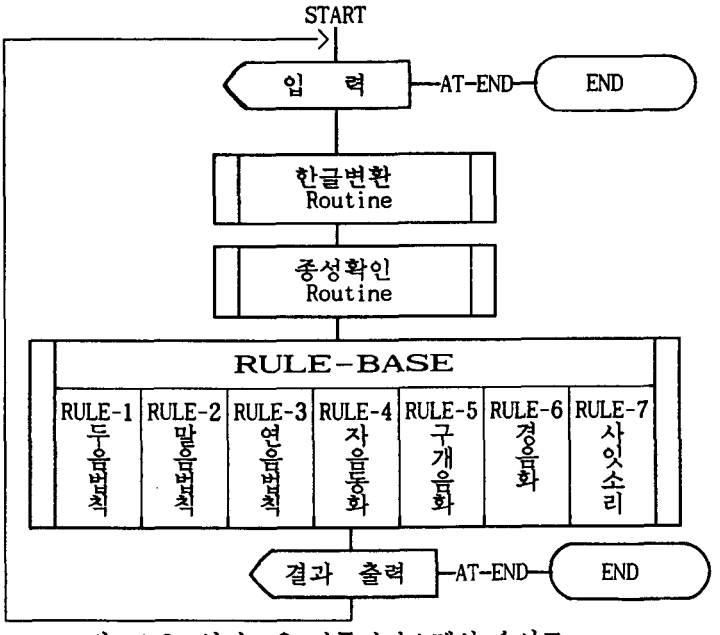

그림 4-2 읽기교육 전문가시스템의 순서도

그리고 시스템 설계시에 유의해야 될 것은 본 연구에서 사용된 Golden사의 Lisp 인터프리터 또는 Lisp Editor인 GMACS내에는 한글이 입력되지 않는다. 그 이유로는 Lisp 인터프리터내의 아스키(ASCII)코드와 2바이트 완성형 한글코드 사이의 충돌이 일어나기 때문이다. 그러나 다른 Editor로 작성된 것을 로드(Load)하면 실행이 된다. 즉 미리 내장되어 있는 글자들은 출력이 되는 것을 의미하는데, 본 연구에서는 PE2를 사용하여 프로그래밍하였다. 입력된 영

문은 그것에 대응하는 한글변환-Routine에 의해 단어 입력시마다 영문을 한글 풀어 쓰기로 변환하여 입력되도록 설계하였다.

예를 들면 "우리나라"라는 단어를 영문 키보드(Key-Board)에 해당하는 "D N F L S K F K"와 같이 입력하는 것이다. 아래의 루틴은 입력을 영문으로 받아 영문에 해당하는 한글을 출력시켜 주는 한글변환-Routine을 Lisp 언어로 작성한 것이다.

```
(setq JA-MO
      '((q . ㅂ) (w . ㅈ) (e . ㄷ) (r . ㄱ) (t . ㅅ) (y . ㅛ)
        (u . ㅕ) (i . ㅑ) (o . ㅐ) (p . ㅔ) (a . ㅁ) (s . ㄴ)
        (d . ㅇ) (f . ㄹ) (g . ㅎ) (h . ㅗ) (j . ㅓ) (k . ㅏ)
        (l . ㅣ) (z . ㅋ) (x . ㅌ) (c . ㅊ) (v . ㅍ) (b . ㅠ)
        (n . ㅜ) (m . ㅡ) (qq . ㅃ) (ww . ㅉ) (qt . ㅄ)
        (fr . ㄺ) (rt . ㄳ) (fq . ㄻ) (ft . ㄽ) (fx . ㄾ)
        (ee . ㄸ) (rr . ㄲ) (tt . ㅆ) (sw . ㄵ) (sg . ㄶ)))
(defun convert (DATA)
  (cond ((equal (cdr DATA) nil)
         (setq WORD
           (let DATA (cons (cdr (assoc (car DATA) JA-MO)) nil))))
        (T (setq WORD
             (let DATA (cons (cdr (assoc (car DATA) JA-MO))
               (convert (cdr DATA))))))))
```

제2절 읽기敎育 專門家 시스템의 具現

1. 시스템의 具現

본 연구에서 사용한 시스템의 환경은 다음과 같다.

Hard Ware : IBM 호환 PC-AT 이상

Main Memory : 1 Mega Byte 이상

AUX-Memory : FDD 1.2 Mega 2대 또는 HDD 10 Mega 이상

사용 한글 : KS 2-byte 완성형 한글

System S/W : GCLISP Version 1.01

Program Size: 20 K Byte

위에서 설계한 시스템을 실행하기 위해서는 먼저 Lisp 인터프리터를 구동시켜야 하며, MS-DOS 상태에서 그림 4-3과 같은 초기 화면을 실행시키기 위해서는 Lisp 언어의 초기 화면을 제어하는 루틴 INIT.LSP에 있는 LOAD 함수를 아래와 같이 수정하면 된다.

```
(LOAD "KE-EXPT.LSP")
```

예) C:\GCLISP\gclisp <enter>

```
GOLDEN COMMON LISP, Version 1.01
Copyright (C) 1985 by Gold Hill Computers

; Reading file INIT.LSP
Type Alt-H for help
; Reading file C:GCLISP\KE-EXPT.LSP

Top-Level
* _
```

그림 4-3 Lisp 실행 초기화면

위의 화면은 GCLISP 인터프리터가 실행된 후 본 연구에 설계한 시스템이 로드된 상태인데, 여기서 시스템을 실행하기 위해서는 다음과 같이 입력하면 된다.

* (EXPERT-SYSTEM)

그리고 아래 화면은 몇가지 단어를 입력시켜서 출력결과를 덤프(Dump)한 것을 나타내고 있다. 입력시에 단어의 입력이 없어도 중지하지 않고 다시 입력이 되도록 설계하였으며, 종료시에는 화면에 표시하는 바와 같이 숫자 999를 타이프(Type)하면 종료를 하고 MS-DOS 상태로 돌아 간다.

```
* (EXPERT-SYSTEM)

단어를 입력하십시오?(종료:999):(Q K F K A)
  입력단어 (ㅂ ㅏ * ㄹ ㅏ ㅁ )은 적용되는 법칙이 없음

단어를 입력하십시오?(종료:999):(D L V)
  입력단어 : (ㅇ ㅣ ㅍ)

  적용법칙 : (말음법칙)

  출력단어 :(ㅇ ㅣ ㅂ)

  *적용이유*
  (음절의 끝소리가 되는 자음은 ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅇ 뿐이다)
  (따라서 이 일곱소리 이외의 자음이 음절의 끝에 오면)
  (일곱자음 중의 하나로 바뀌는 현상)
```

그림 4-4 실행 예의 화면-1

```
단어를 입력하십시오?(종료:999):(R K X D L)
  입력단어 : (ㄱ ㅏ ㅌ ㅇ ㅣ *)

  적용법칙 : (구개음화)

  출력단어 :(ㄱ ㅏ * ㅊ ㅣ *)

  *적용이유*
  (음절의 끝소리가 ㄷ ㅌ 인 형태소가 모음 ㅣ 나)
  (반모음 ㅣ 로 시작되는 형식형태소와 만나면)
  (구개음 ㅈ ㅊ 으로 바뀌는 현상)
```

그림 4-5 실행 예의 화면-2

여기서 입력단어 (ㄱ ㅏ ㅌ ㅇ ㅣ *)가 지식베이스내에 어떻게 적용되어 출력단어 (ㄱ ㅏ * ㅊ ㅣ *)를 생성하는지를 살펴보면 다음과 같다.

먼저 화면으로 입력단어에 해당하는 영문단어 (R K X D L)가 List 형태의 풀어쓰기로 입력되면, 한글변환-Routine에 의해 List (ㄱ ㅏ ㅌ ㅇ ㅣ)로 바뀌어지고 그 다음 종성확인-Routine에 의해 List (ㄱ ㅏ ㅌ ㅇ ㅣ *)로 변환된다.

여기서 종성확인-Routine을 사용하는 이유는, 본 연구에서 사용한 언어의 제어구조가 되부름에 의존하기 때문에 글자수를 일정하게 정렬함으로써 되부름을 쉽게 구현할 수 있고, 출력이 풀어 쓰기형태로 출력됨으로써 읽기가 난해한 부분을 고려한 것이라 볼 수 있다.

그 다음 지식베이스내에 들어있는 규칙들을 적용시켜 적용이 되는 새로운

사실들을 생성시킨다. 위의 그림 4-5과 같은 경우 입력단어는 지식베이스내의 Rule-2, Rule-3, 그리고 Rule-5등에 적용되어서 새로운 사실들이 생성되어 Working Memory의 *R-2*, *R-3*, 그리고 *R-5*등에 저장되어 있다.

위의 예에서 적용되어진 규칙들을 살펴보면 아래와 같다.

```
* RULE-2 말음법칙이 적용됨(*R-2*: ㄱㅏ ㄷ ㅇㅣ *)
(cond ((member (cadddr DATA) '(ㅅ ㅆ ㅈ ㅊ ㅌ))
            (setq *R-2*
                (cons (car DATA)
                 (cons (cadr DATA)
                  (cons 'ㄷ nil)))))
* RULE-3 연음법칙이 적용됨(*R-3*: ㄱㅏ * ㄷㅣ *)
   ((and (equal (nth 3 DATA) 'ㅇ)
         (not (member (nth 2 DATA) SA-EM))
         (or (equal (nth 2 DATA) '*)
             (member (nth 4 DATA) '(ㅡ ㅗ ㅣ ㅟ))))
          (setq *R-3*
            (cons (nth 0 DATA)
             (cons (nth 1 DATA)
              (cons '*
               (cons (nth 2 DATA)
                (nthcdr 4 *R-2*)))))))
* RULE-5 구개음화가 적용됨(*R-5*: ㄱㅏ * ㅊㅣ *)
   ((or (and (equal (nth 2 DATA) 'ㅌ)
             (member (nth 4 DATA) '(ㅣ ㅑ ㅕ ㅛ ㅠ ㅒ ㅖ)))
        (and (equal (nth 2 DATA) 'ㄷ)
             (equal (nth 3 DATA) 'ㅎ)
             (member (nth 4 DATA) '(ㅣ ㅑ ㅕ ㅛ ㅠ ㅒ ㅖ))))
            (setq *R-5*
             (cons (nth 0 DATA)
              (cons (nth 1 DATA)
               (cons '*
                (cons 'ㅊ (nthcdr 4 DATA)))))))
```

이와 같이 입력 List (ㄱㅏ ㅌ ㅇㅣ *)는 추론엔진을 구동시켜 해당규칙을 적용시켜보면 3가지의 규칙이 적용됨을 알 수 있다. 이런 경우의 최종 결과는 추론엔진내의 스케듈러에 의해 우선 순위를 결정한 뒤에 처리되어 우리가 원하는 결과 List (ㄱㅏ * ㅊ ㅣ *)를 출력하게 된다.

그리고 위의 입력의 경우, 구개음화가 최종 결과로 출력되는데 그 과정을 설명해 보면 다음과 같다.

먼저 추론엔진내의 스케듈러는 입력단어가 말음법칙(*R-2*)과 연음법칙(*R-3*), 그리고 경음화현상(*R-6*)이 적용되는지 비교한다. 그러면 위의 단어는 경음화현상에는 해당되지 않으므로 그 다음의 조건을 비교하게 된다.

그 다음의 조건은 입력단어가 연음법칙(*R-3*)에 적용되지 않고, 그리고 구개음화(*R-5*)에 적용되는지 비교하여 보면 이 조건에도 만족되지 않으므로 그 다음 조건을 비교한다. 그 다음 조건은 입력단어가 자음동화(*R-5*)에 적용되지 않고, 연음법칙(*R-3*)과 경음화현상(*R-6*)이 성립하며 그리고 입력단어가 구개음화(*R-5*)에 적용되는지를 검사하지만 이 조건도 또한 성립하지

않는다.

그 다음은 위의 모든 조건이 성립하지 않고, 구개음화(*R-5*)가 성립하는가를 비교해 보면 이 조건은 만족하므로 추론엔진은 이 법칙과 관련된 적용법칙 그리고 적용이유 등을 출력하게 되는 것이다.

2. 具現結果 分析

위에서 언급한 읽기교육을 위한 전문가 시스템은 인공지능 기법을 응용함으로써 기존의 언어들이 처리하기 어려운 문자처리 부분에 탁월한 효율을 나타낼 수 있다. 특히 학생들의 국어 읽기교육에 상당한 관심과 흥미를 가질 수 있도록 설계, 구현되어 있어 컴퓨터를 이용한 국어 학습에 도움이 될 것이다.

본 연구의 결과로 다음의 몇가지를 알 수 있었다.

첫째로, 推論過程의 提示를 들 수 있다. 이것은 인공지능 기법의 특징으로 시스템이 올바르고 정확하게 구현될 수 있게 해 주는 동시에, 사용자에게는 추론과정을 제시하여 줌으로써 결과에 대한 신뢰성을 갖게 한다.

둘째로, 읽기敎育에서 效率的인 活用을 들 수 있다. Lisp 언어는 인터프리터 방식에 의해 단어를 입력함과 동시에 바로 실행 가능하기 때문에, 학생들이 대화식으로 사용할 수 있어 CAI 시스템을 이용한 ICAI 시스템을 구현하는데 효율적이라고 볼 수 있다.

세째로, 시스템의 擴張 可能性이다. 이것은 전문가 시스템이 인간의 전문가와 같이 하기 위해서 해당 분야의 지식과 새로운 사실들을 기존의 설계 변경없이 축적 및 확장 가능함을 말한다. 즉 지식베이스내의 규칙들을 추가, 갱신 그리고 삭제가 용이함을 말한다.

네째로, 特定 分野의 學生들을 訓練시키는데 사용될 수 있다. 즉 학생들이 전문가 시스템을 사용함으로서 전문가가 가지고 있는 지식은 물론 문제 해결의 과정을 습득할 수 있다. 또한 전문가 시스템이 내리는 판단 및 결과에 대하여 그 근거를 조회함으로써 학생들의 학습에 많은 도움이 될 수 있으며, 전문가 시스템을 사용하는 그 자체가 교육의 일부로 생각할 수 있다고 본다.

마지막으로, 知識 表現의 重要性을 들 수가 있다. 인공지능에서 아무리 작은 규모의 문제라도 실용적 응용을 위해서 요구되는 지식의 양은 방대하고, 지식 상호간에 밀접한 관계를 갖고 있기 때문에 분류하기도 어려우며, 또 계속 변화하기 때문에 어떻게 지식을 체계적으로 저장하고 효율적으로 사용할 수 있는가에 더욱 연구가 되어야함을 알 수 있었다.

하지만 本 硏究에 있어 다음과 같은 問題點도 提起된다.

첫째로, 本 硏究에서 사용한 한글은 KS 완성형 한글을 사용하였기 때문에, 한글자체의 非互換性으로 인한 다른 한글, 예를 들면 조합형 한글, S/W 한글 등에는 사용할 수 없는 短點이 있다. 이러한 문제점의 해결은 추후에 한글 상호변환 루틴을 만들게 되면 가능할 것이다.

둘째로, 國語文法 具現時의 어려움이다. 이것은 쉽게 구현되는 법칙도 있는

반면에, 연음법칙의 경우 한 음절의 뒷소리가 실질 형태소인 경우와 형식 형태소인 경우가 다르게 연음됨으로 양쪽의 정확한 구분이 필요하다. 그러나 우리말에서 실질 형태소인 경우만 하더라도 체언(명사, 대명사, 수사), 용언의 어간, 감탄사, 관형사, 그리고 부사등이 있어 이들을 전부 구현하기에는 너무 방대해지는 어려움이 있다.

따라서 본 연구에서는 그 중에서 빈번하게 사용되는 일부분만 구현하였음을 밝혀둔다. 그러나 앞으로의 컴퓨터 교육 추세로 볼 때, 여기에서 사용된 시스템은 지능적 컴퓨터 보조교육의 구현에 도움이 될 것이며, 우리의 한글에 대한 관심과 컴퓨터를 이용한 인공지능 기법의 효율적인 구현에도 도움이 될 것이다.

第Ⅴ章 結 論

本 硏究에서는 우리말의 읽기교육을 위한 시스템을 知識의 表現이 比較的 쉽고, 具現이 간단한 生成 시스템에 의한 推論方法을 使用하여 具現하였다.

專門家 시스템은 人間 專門家의 機能 및 役割을 프로그램화 한 것이라고 본다면 人間 專門家가 가지고 있는 知識과 專門的 判斷에 이용되는 規則 및 精神 能力을 모두 프로그램화 하는 것은 現在로선 不可能한 일이기도 하지만, 知識工學 및 소프트웨어工學이 발전함에 따라 不可能한 것도 아니다. 人工知能의 여러 應用分野 중에서 專門家 시스템이 가장 먼저 實用化가 되었다는 점을 보아도 계속 발전할 可能性이 다른 分野보다도 밝다고 하겠다.

하지만 本 硏究에서 사용된 Golden社의 GCLISP 인터프리터는 한글 入力時에 英文과의 衝突로 인하여 한글자체가 入力되지 않아 英文으로 入力시켜 具現할 수밖에 없는 것은 追後에 改善해야 할 課題로 지적되며, 아울러 出力도 List 형태의 한글 풀어쓰기로 出力된 점도 지적하는 바이다.

그리고 本 硏究에서 具現된 내용은 읽기敎育과 관련된 각종 法則중의 일부임을 밝혀두고, 여기서는 그 可能性에 대한 例만 제시하였다. 또한 각 法則들의 例外가 많아 시스템을 具現하는데 너무 방대해질 수 있어 例外인 單語는 具現되지 않았음도 밝혀둔다.

* 參考 文獻

1) 유석인, 「전문가 시스템의 소개」, 『情報科學會誌』, 第 6 卷, 第 2 號, 1988, pp.6-7.

2) 박충식, 김재희, 「인공지능의 지식표현 방식에 대한 비교 고찰」,

3) 김진형, 이종한, 백귀업, 「PC 고장 접수 및 진단을 위한 전문가 시스템의 구현」, 『情報科學會誌』, 第 4 卷, 第 4 號, 1988.

4) 게리 헨드릭스, 「인공지능의 실체와 전망」, 『PC 저널』, 서울:매일경제, 1990, 3월號.

5) 박영수, 「기계번역에서 본 한국어의 특징」, 『情報科學會誌』, 第 7 卷, 第 6 號, 1989.

6) 박경환, 김영택, 「논리적 구문에 대한 규칙 기반 프로그램 구성」, 『情報科學會論文誌』, Vol.16, No.4 1989.

7) 백영균, 「교육에 전문가 시스템을 도입하기 위한 기초 연구」, 『情報科學會誌』, 第 7 卷, 第 3 號, 1989.

8) 임영환, 신동하, 김창석, 「전문가 시스템의 추론방법의 경향과 응용 분야 결정요소」, 『情報科學會誌』, 第 3 卷, 第 3 號, 1988.

9) 고영근, 『국어문법의 연구』, 탑출판사, 1983.

10) 남기심, 고영근, 이익섭, 『현대 국어문법』, 계명대학교 출판부, 1979.

11) 이종락, 『Common LisP 입문』, 서울:세화출판사, 1989.

12) 이을한외 6인, 『국어학 신강』, 서울:박문각, 1977.

13) 이은정, 『한글 맞춤법 표준어해설』, 서울:대제각, 1988.

14) 유석인, 전주식, 한상영 『人工知能』, 서울:상조사, 1990.

15) 최인렬, 최인현, 『Common LISP』, 서울:대림출판사, 1989.

16) David S. Touretzky, COMMON LISP, The Benjamin/Cummings Publishing ComPany, Inc, 1990.

17) D.A. Waterman and F.Hayes-Roth, Pattern-Directed Inference System, Academic Press, Inc., Orland, Florida, 1978.

18) Gerge F.Luger, William A.Stubblefield, ARTIFICIAL INTELLIGENCE, The Benjamin/Cummings Publishing ComPany, Inc, 1989.

19) Greg P. Kearsley, Artificial Intelligence & Instruction, Addison-Wesley Publishing ComPany, 1987.

20) Lee Brownston외 3인, Programming ExPert Systems in OPS5, Addison-Wesley Publishing ComPany, 1986.

21) Paul H. David King, EXPERT SYSTEMS, John Wiley & Sons, Inc, 1985.

22) Patrick H. Winston, LISP , Addison-Wesley Publishing ComPany, 1989.

23) Rodney A. BROOKS, PROGRAMMING IN COMMOM LISP, John Wiley & Sons, Inc, 1985.

24) Tayun Khanna, FOUNDATIONS OF NEURAL NETWORKS, Addison-Wesley Publishing ComPany, 1990.

※ 附 錄

```
;*************************************************************************
;*                EXPERT SYSTEM  for                                     *
;*                     KOREAN-PROUNCIATION EDUCATION                     *
;*                     Author : Moon Soo-Yeal                            *
;*                     Date   : April, 20, 1991                          *
;*                     System S/W : GC-LISP Version 1.01                 *
;*************************************************************************

;*************************************************************************
(defun EXPERT-SYSTEM ()
  (setq LIST nil)
  (format t "~%~%단어를 입력하십시오?(종료:999):")
  (setq LIST (read))
  (cond ((equal lIST 999) (EXIT))
        ((equal LIST nil) (EXPERT-SYSTEM)))
  (convert LIST)   (check WORD)
  (RULE-1 WORD)
  (cond ((not (equal *R-1* nil))
              (format t "~%  ~S   ~S~%" IN-P WORD)
              (format t "~%  ~S   ~S~%" O-RULE R-1-P)
              (format t "~%  ~S   ~S~%" OUT-P *R-1*)
              (format t "~%  ~S~% ~S" O-OOP P-1)
              (format t "~% ~s~%" P-1-1)
              (format t " ~s" P-1-2)
              (EXPERT-SYSTEM)))
  (RULE-2 WORD)
  (RULE-3 WORD)
  (RULE-4 *R-2*)
  (RULE-5 WORD)
  (RULE-6 *R-2*)
  (RULE-7 *R-2*)
  (cond ((and (equal WORD *R-2*) (equal WORD *R-3*)
              (equal WORD *R-4*) (equal WORD *R-5*)
              (equal WORD *R-6*) (equal WORD *R-7*))
              (format t "~%입력단어 ~S 은(는) 문법에
                         적용되는 규칙이 없음" WORD)
              (EXPERT-SYSTEM))
        ((and (equal WORD *R-3*)
              (equal *R-2* *R-6*)
              (not (equal WORD *R-2*)))
              (format t "~%  ~S   ~S~%" IN-P WORD)
              (format t "~%  ~S   ~S~%" O-RULE R-2-P)
              (format t "~%  ~S   ~S~%" OUT-P *R-2*)
              (format t "~%  ~S~% ~S" O-OOP P-2)
              (format t "~% ~s~%" P-2-1)
              (format t " ~s" P-2-2)
              (EXPERT-SYSTEM))
        ((and (not (equal WORD *R-3*))
              (equal WORD *R-5*))
              (format t "~%  ~S   ~S~%" IN-P WORD)
              (format t "~%  ~S   ~S~%" O-RULE R-3-P)
              (format t "~%  ~S   ~S~%" OUT-P *R-3*)
```

```
            (format t "~%  ~S~% ~S" O-OOP P-3)
            (format t "~% ~s~%" P-3-1)
            (format t " ~s" P-3-2)
            (EXPERT-SYSTEM))
      ((and (not (equal WORD *R-4*))
            (equal *R-2* *R-6*)
            (equal WORD *R-5*))
            (format t "~%  ~S   ~S~%" IN-P WORD)
            (format t "~%  ~S   ~S~%" O-RULE R-4-P)
            (format t "~%  ~S   ~S~%" OUT-P *R-4*)
            (format t "~%  ~S~% ~S" O-OOP P-4)
            (format t "~% ~s~%" P-4-1)
            (format t " ~s" P-4-2)
            (EXPERT-SYSTEM))
      ((not (equal WORD *R-5*))
            (format t "~%  ~S   ~S~%" IN-P WORD)
            (format t "~%  ~S   ~S~%" O-RULE R-5-P)
            (format t "~%  ~S   ~S~%" OUT-P *R-5*)
            (format t "~%  ~S~% ~S" O-OOP P-5)
            (format t "~% ~s~%" P-5-1)
            (format t " ~s" P-5-2)
            (EXPERT-SYSTEM))
      ((and (not (equal WORD *R-6*))
            (not (equal *R-2* *R-7*)))
            (format t "~%  ~S   ~S~%" IN-P WORD)
            (format t "~%  ~S   ~S~%" O-RULE R-6-P)
            (format t "~%  ~S   ~S~%" OUT-P *R-6*)
            (format t "~%  ~S~% ~S" O-OOP P-6)
            (format t "~% ~s~%" P-6-1)
            (EXPERT-SYSTEM))
      ((not (equal WORD *R-7*))
            (format t "~%  ~S   ~S~%" IN-P WORD)
            (format t "~%  ~S   ~S~%" O-RULE R-7-P)
            (format t "~%  ~S   ~S~%" OUT-P *R-7*)
            (format t "~%  ~S~% ~S" O-OOP P-7)
            (format t "~% ~s" P-7-1)
            (format t "~% ~s" P-7-2)
            (format t "~% ~s" P-7-3)
            (EXPERT-SYSTEM))
      (T (EXPERT-SYSTEM))))

;*************************************************************************
 (setq JA-MO
      '((q . ㅂ) (w . ㅈ) (e . ㄷ) (r . ㄱ) (t . ㅅ) (y . ㅛ)
        (u . ㅕ) (i . ㅑ) (o . ㅐ) (p . ㅔ) (a . ㅁ) (s . ㄴ)
        (d . ㅇ) (f . ㄹ) (g . ㅎ) (h . ㅗ) (j . ㅓ) (k . ㅏ)
        (l . ㅣ) (z . ㅋ) (x . ㅌ) (c . ㅊ) (v . ㅍ) (b . ㅠ)
        (n . ㅜ) (m . ㅡ) (qq . ㅃ) (ww . ㅉ) (qt . ㅄ)
        (fr . ㄺ) (rt . ㄳ) (fq . ㄻ) (ft . ㄽ) (fx . ㄾ)
        (ee . ㄸ) (rr . ㄲ) (tt . ㅆ) (sw . ㄵ) (sg . ㄶ)))
 (setq JA-EM   '(ㅂ ㅃ ㅈ ㅉ ㄷ ㄸ ㄱ ㄲ ㅅ ㅆ ㅁ ㄴ ㅇ ㄹ ㅎ ㅋ
                 ㅌ ㅊ ㅍ ㄳ ㄵ ㄶ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅄ))
 (setq SA-JA   '((ㄱ . ㄲ) (ㄷ . ㄸ) (ㅂ . ㅃ) (ㅅ . ㅆ) (ㅈ. ㅉ)))
 (setq SA-EM   '(ㄳ ㄵ ㄶ ㄺ ㄻ ㄼ ㄽ ㄾ ㄿ ㅀ ㅄ))
 (setq MO-EM   '(ㅛ ㅕ ㅑ ㅐ ㅔ ㅗ ㅓ ㅏ ㅣ ㅠ ㅜ ㅡ ㅒ ㅖ))
```

```
(setq END-EM  '(ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅇ))
(setq IN-P    '입력단어:)   (setq O-RULE  '적용법칙:)
(setq OUT-P   '출력단어:)   (setq O-OOP   '*적용이유*)
(setq R-1-P   '두음법칙)   (setq R-2-P   '말음법칙)
(setq R-3-P   '연음법칙)   (setq R-4-P   '자음동화)
(setq R-5-P   '구개음화)   (setq R-6-P   '경음화현상)
(setq R-7-P   '사잇소리현상)
(setq P-1 '(ㄹ 과 ㅇ 이 단어의 첫소리로 쓰이지 않고))
(setq P-1-1 '(ㄴ 이 ㅣ 반모음 ㅣ 앞에서 쓰이지 않은 현상.))
(setq P-1-2 '(예외. 의존명사는 적용되지 아니함.
                ex.그럴 리가. 한 닢))
(setq P-2 '(음절의 끝소리가 되는 자음은 ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅇ 뿐임))
(setq P-2-1 '(따라서 이 일곱소리 이외의 자음이 음절의 끝에 오면))
(setq P-2-2 '(그것은 이 일곱자음 중의 하나로 바뀌는 현상.))
(setq P-3 '(자음으로 끝나는 음절에 종속적으로 따르는 말이))
(setq P-3-1 '(모음으로 시작되면. 앞음절의 끝소리가))
(setq P-3-2' (모음의 첫소리로 바뀌는 현상.))
(setq P-4 '(끝자음이 그 뒤에 오는 자음과 만날 때 어느 한 쪽이))
(setq P-4-1 '(다른 쪽을 닮아서 비슷하거나 같은 소리로
                                        바뀌기도 하고.))
(setq P-4-2 '(양쪽이 서로 닮아서 두 소리가 다 바뀌기도 하는 현상.))
(setq P-5 '(끝소리가 ㄷ.ㅌ 인 형태소가 모음 ㅣ 나 ))
(setq P-5-1 '(반모음 ㅣ 로 시작되는 형식형태소와 만나면))
(setq P-5-2 '(구개음 ㅈ.ㅊ 이 되는 현상.))
(setq P-6 '(두 개의 안울림 소리가 서로 만나면 뒤의 소리가))
(setq P-6-1 '(된소리로(ㄲ ㄸ ㅃ ㅆ ㅉ) 발음되는 현상.))
(setq P-7 '(앞의 말의 끝소리가 울림소리(ㄴ ㄹ ㅁ ㅇ)이고))
(setq P-7-1 '(뒤말의 첫소리가 안울림 예사소리이면 뒤의 예사소리가))
(setq P-7-2 '(된소리(ㄲ ㄸ ㅃ ㅆ ㅉ)로 바뀌는 현상.))
(setq P-7-3 '(예외. 고래기름 기와집 밤송이))

;*************************************************************************
(defun convert (DATA)
  (cond ((equal (cdr DATA) nil)
           (setq WORD
             (let DATA (cons (cdr (assoc (car DATA) JA-MO)) nil))))
        (T (setq WORD
             (let DATA (cons (cdr (assoc (car DATA) JA-MO))
               (convert (cdr DATA))))))))

;*************************************************************************
(defun check (DATA)
  (cond ((equal DATA nil) nil)
        (T (if (or (equal (caddr DATA) nil)
                   (member (car(cdddr DATA)) MO-EM))
               (setq WORD
                 (let DATA
                  (cons (car DATA)
                    (cons (cadr DATA)
                       (cons '* (check (cddr DATA))))))))
               (setq WORD
                 (let DATA
                  (cons (car DATA)
                    (cons (cadr DATA)
                      (cons (caddr DATA)
```

```
                        (check (cdddr DATA)))))))))))

;*********************************************************************
(defun RULE-1 (DATA)
   (cond ((equal DATA nil)  nil)
         ((and (or  (equal (car DATA) 'ㄹ)
                    (equal (car DATA) 'ㄴ))
               (member (cadr DATA) '(ㅏ ㅕ ㅛ ㅠ ㅖ ㅣ )))
                     (setq *R-1* (cons 'ㅇ (cdr DATA))))
         ((equal (car DATA) 'ㄹ)
                   (setq *R-1* (cons 'ㄴ (cdr DATA))))))

 ;*********************************************************************
(defun RULE-2 (DATA)
  (cond ((equal DATA nil) nil)
        ((member (caddr DATA) '(ㅍ ㅂㅅ))
               (setq *R-2*
                  (cons (car DATA)
                    (cons (cadr DATA)
                      (cons 'ㅂ nil)))))
        ((member (caddr DATA) '(ㅅ ㅆ ㅈ ㅊ ㅌ))
               (setq *R-2*
                  (cons (car DATA)
                    (cons (cadr DATA)
                      (cons 'ㄷ nil)))))
        ((member (caddr DATA) '(ㄲ ㅋ ㄺ ㄳ ))
               (setq *R-2*
                  (cons (car DATA)
                    (cons (cadr DATA)
                      (cons 'ㄱ nil)))))
        ((member (caddr DATA) '(ㄺ ㄻ ㄾ))
               (setq *R-2*
                  (cons (car DATA)
                    (cons (cadr DATA)
                      (cons 'ㄹ nil)))))
        ((member (caddr DATA) '(ㄵ))
               (setq *R-2*
                  (cons (car DATA)
                    (cons (cadr DATA)
                      (cons 'ㄴ nil)))))
        (T (setq *R-2*
                 (cons (car DATA)
                   (cons (cadr DATA)
                     (cons (caddr DATA) nil))))))
  (cond ((equal DATA nil) nil)
        (T (setq *R-2*
             (append *R-2*
                (RULE-2 (cdddr DATA)))))))

;*************************************************************
(defun RULE-3 (DATA)
  (cond ((equal DATA nil) nil)
        ((or (equal (nthcdr 3 DATA) nil)
             (equal (nth 2 DATA) '*))
               (setq *R-3* DATA))
```

```
((and (equal (nth 2 DATA) (nth 2 *R-2*))
      (equal (nth 3 DATA) 'ㅇ)
      (not (member (nth 2 DATA) SA-EM))
      (not (equal (nth 5 DATA) '*))
      (member (nth 4 DATA) '(ㅡ ㅗ ㅣ ㅓ)))
       (setq *R-3*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons (nth 2 DATA)
            (cons (nth 2 *R-2*)
             (nthcdr 4 *R-2*)))))))
((and (member (nth 2 DATA) END-EM)
      (not (member (nth 2 DATA) SA-EM))
      (equal (nth 3 DATA) 'ㅇ)
      (member (nth 4 DATA) '(ㅡ ㅗ ㅣ ㅓ)))
       (setq *R-3*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons (nth 2 DATA)
            (cons (nth 2 DATA)
             (nthcdr 4 *R-2*)))))))
((and (equal (nth 3 DATA) 'ㅇ)
      (not (member (nth 2 DATA) SA-EM))
      (or (equal (nth 2 DATA) '*)
          (member (nth 4 DATA) '(ㅡ ㅗ ㅣ ㅓ))))
       (setq *R-3*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons '*
            (cons (nth 2 DATA)
             (nthcdr 4 *R-2*)))))))
((and (equal (nth 3 DATA) 'ㅇ)
      (not (member (nth 2 DATA) SA-EM))
      (not (or (equal (nth 2 DATA) '*)
               (member (nth 4 DATA) '(ㅡ ㅗ ㅣ ㅓ)))))
       (setq *R-3*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons '*
            (cons (nth 2 *R-2*)
             (nthcdr 4 *R-2*)))))))
((and (member (nth 2 DATA) SA-EM)
      (equal (nth 2 DATA) 'ㄱㅅ))
       (setq *R-3*
          (cons (nth 0 DATA)
           (cons (nth 1 DATA)
             (cons (nth 2 *R-2*)
              (cons 'ㅅ (nthcdr 4 DATA)))))))
((and (member (nth 2 DATA) SA-EM)
      (equal (nth 2 DATA) 'ㅂㅅ))
       (setq *R-3*
          (cons (nth 0 DATA)
           (cons (nth 1 DATA)
             (cons (nth 2 *R-2*)
              (cons 'ㅅ (nthcdr 4 DATA)))))))
```

```
      ((and (member (nth 2 DATA) SA-EM)
            (equal (nth 2 DATA) 'ㄺ))
              (setq *R-3*
                 (cons (nth 0 DATA)
                  (cons (nth 1 DATA)
                    (cons (nth 2 *R-2*)
                     (cons 'ㅅ (nthcdr 4 DATA)))))))
      ((and (member (nth 2 DATA) SA-EM)
            (equal (nth 2 DATA) 'ㅈ))
              (setq *R-3*
                 (cons (nth 0 DATA)
                  (cons (nth 1 DATA)
                    (cons (nth 2 *R-2*)
                    (cons 'ㅈ (nthcdr 4 DATA)))))))
      ((and (member (nth 2 DATA) SA-EM)
            (equal (nth 2 DATA) 'ㄻ))
              (setq *R-3*
                 (cons (nth 0 DATA)
                  (cons (nth 1 DATA)
                    (cons 'ㄹ
                         (nthcdr 4 *R-2*))))))
      ((and (member (nth 2 DATA) SA-EM)
            (equal (nth 2 DATA) 'ㄼ))
              (setq *R-3*
                 (cons (nth 0 DATA)
                  (cons (nth 1 DATA)
                    (cons (nth 2 *R-2*)
                    (cons 'ㅂ (nthcdr 4 DATA)))))))
      ((and (member (nth 2 DATA) SA-EM)
            (equal (nth 2 DATA) 'ㄾ))
              (setq *R-3*
                 (cons (nth 0 DATA)
                  (cons (nth 1 DATA)
                   (cons (nth 2 DATA)
                    (cons (nth 2 *R-2*)
                    (cons 'ㅌ (nthcdr 4 DATA)))))))
      (T (setq *R-3* DATA)))
  (cond ((equal DATA nil) nil)
      (T (setq *R-3*
           (cons (nth 0 *R-3*)
            (cons (nth 1 *R-3*)
             (cons (nth 2 *R-3*)
              (RULE-3 (nthcdr 3 *R-3*)))))))))

;*************************************************************************
(defun RULE-4 (DATA)
  (cond ((equal DATA nil) nil)
      ((or (equal (nthcdr 3 DATA) nil)
           (equal (nth 2 DATA) '*))
             (setq *R-4* DATA))
      ((and (equal (caddr *R-2*) 'ㅂ)
            (member (car (cdddr *R-2*)) '(ㅁ ㄴ)))
            (setq *R-4*
              (cons (nth 0 DATA)
               (cons (nth 1 DATA)
```

```
          (cons 'ㅁ (nthcdr 3 DATA))))))
((and (equal (caddr *R-2*) 'ㄷ)
      (member (car (cdddr *R-2*)) '(ㅁ ㄴ)))
      (setq *R-4*
        (cons (nth 0 DATA)
         (cons (nth 1 DATA)
          (cons 'ㄴ (nthcdr 3 DATA))))))
((and (equal (caddr *R-2*) 'ㄱ)
      (member (car (cdddr *R-2*)) '(ㅁ ㄴ)))
      (setq *R-4*
        (cons (nth 0 DATA)
         (cons (nth 1 DATA)
          (cons 'ㅇ (nthcdr 3 DATA))))))
((and (equal (nth 3 *R-2*) 'ㄹ)
      (member (caddr *R-2*) '(ㅁ ㅇ)))
       (setq *R-4*
        (cons (nth 0 DATA)
         (cons (nth 1 DATA)
          (cons (nth 2 DATA)
           (cons 'ㄴ (nthcdr 4 DATA)))))))
((and (equal (nth 2 *R-2*) 'ㄴ)
      (equal (nth 3 *R-2*) 'ㄹ))
        (setq *R-4*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons 'ㄹ (nthcdr 3 DATA))))))
((and (equal (nth 2 *R-2*) 'ㄹ)
      (equal (nth 3 *R-2*) 'ㄴ))
        (setq *R-4*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons (nth 2 DATA)
            (cons 'ㄹ (nthcdr 4 DATA)))))))
((and (equal (nth 3 *R-2*) 'ㄹ)
      (equal (nth 2 *R-2*) 'ㅂ))
        (setq *R-4*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons 'ㅁ
            (cons 'ㄴ (nthcdr 4 DATA)))))))
((and (equal (nth 3 *R-2*) 'ㄹ)
      (equal (nth 2 *R-2*) 'ㄷ))
        (setq *R-4*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons 'ㄴ
            (cons 'ㄴ (nthcdr 4 DATA)))))))
((and (equal (nth 3 *R-2*) 'ㄹ)
      (equal (nth 2 *R-2*) 'ㄱ))
        (setq *R-4*
         (cons (nth 0 DATA)
          (cons (nth 1 DATA)
           (cons 'ㅇ
            (cons 'ㄴ (nthcdr 4 DATA)))))))
(T (setq *R-4* DATA)))
```

```
  (cond ((equal DATA nil) nil)
        (T (setq *R-4*
             (cons (nth 0 *R-4*)
              (cons (nth 1 *R-4*)
               (cons (nth 2 *R-4*)
                (RULE-4 (nthcdr 3 *R-4*)))))))))

;*********************************************************************
(defun RULE-5 (DATA)
  (cond ((equal DATA nil) nil)
        ((or (equal (nthcdr 3 DATA) nil)
             (equal (nth 2 DATA) '*))
             (setq *R-5* DATA))
        ((or (and (equal (nth 2 DATA) 'ㅌ)
                  (member (nth 4 DATA) '(ㅣ ㅑ ㅋ ㅛ ㅠ ㅒ ㅖ)))
             (and (equal (nth 2 DATA) 'ㄷ)
                  (equal (nth 3 DATA) 'ㅎ)
                  (member (nth 4 DATA) '(ㅣ ㅑ ㅋ ㅛ ㅠ ㅒ ㅖ))))
              (setq *R-5*
               (cons (nth 0 DATA)
                (cons (nth 1 DATA)
                 (cons '*
                  (cons 'ㅊ (nthcdr 4 DATA)))))))
        ((and (equal (nth 2 DATA) 'ㄷ)
              (member (nth 4 DATA) '(ㅣ ㅑ ㅋ ㅛ ㅠ ㅒ ㅖ)))
               (setq *R-5*
                (cons (nth 0 DATA)
                 (cons (nth 1 DATA)
                  (cons '*
                   (cons 'ㅈ (nthcdr 4 DATA)))))))
        (T (setq *R-5* DATA)))
  (cond ((equal DATA nil) nil)
        (T (setq *R-5*
             (cons (nth 0 *R-5*)
              (cons (nth 1 *R-5*)
               (cons (nth 2 *R-5*)
                (RULE-5 (nthcdr 3 *R-5*)))))))))

;*********************************************************************
(defun RULE-6 (DATA)
  (cond ((equal DATA nil) nil)
        ((or (equal (nthcdr 3 DATA) nil)
             (equal (nth 2 DATA) '*))
               (setq *R-6* DATA))
        ((and (not (member (nth 2 DATA) '(ㄴ ㄹ ㅁ ㅇ)))
              (not (member (nth 3 DATA) '(ㄴ ㄹ ㅁ ㅇ))))
                (setq *R-6*
                  (cons (nth 0 DATA)
                   (cons (nth 1 DATA)
                    (cons (nth 2 DATA)
                     (cons (cdr (assoc (nth 3 DATA) SA-JA))
                        (nthcdr 4 DATA)))))))
        (T (setq *R-6* DATA)))
  (cond ((equal DATA nil) nil)
        (T (setq *R-6*
```

```
            (cons (nth 0 *R-6*)
             (cons (nth 1 *R-6*)
              (cons (nth 2 *R-6*)
               (RULE-6 (nthcdr 3 *R-6*)))))))))

;*********************************************************************
(defun RULE-7 (DATA)
  (cond ((equal DATA nil) nil)
        ((or (equal (nthcdr 3 DATA) nil)
             (equal (nth 2 DATA) '*))
               (setq *R-7* DATA))
        ((and (member (nth 2 DATA) '(ㄴ ㄹ ㅁ ㅇ))
              (equal (car (assoc (nth 3 DATA) SA-JA)) (nth 3 DATA))
              (not (member (nth 3 DATA) '(ㄴ ㄹ ㅁ ㅇ))))
                (setq *R-7*
                  (cons (nth 0 DATA)
                   (cons (nth 1 DATA)
                    (cons (nth 2 DATA)
                     (cons (cdr (assoc (nth 3 DATA) SA-JA))
                        (nthcdr 4 DATA)))))))
        (T (setq *R-7* DATA)))
  (cond ((equal DATA nil) nil)
        (T (setq *R-7*
             (cons (nth 0 *R-7*)
              (cons (nth 1 *R-7*)
               (cons (nth 2 *R-7*)
                (RULE-7 (nthcdr 3 *R-7*)))))))))
```