# LR 구문분석 기법을 이용한 음성 계산기

# Voice Calculator using LR Parsing Technique

유 형 근*, 이 형 준*, 이 강 성*, 김 순 협*
(Hyung-Keun Ryu, Hyung-Jun Lee, Gang-Sung Lee, Soon-Hyob Kim)

## 요 약

본 논문은 음성만을 이용하여 계산을 할 수 있도록 하는 음성 계산기 구현을 위한 기술에 관한 연구이다. 일정한 형식을 갖는 언어에 의하여 발생할 수 있는 다양한 형태의 구문을 언어 문법적 규칙을 기초로 해석하는 기능은 단독어 인식에서 문장 인식으로 넘어가는 과정에서 필수적인 요소이다. 음성 계산기에 HMM과 LR 구문분석 기법을 적용하여, 입력된 문장을 분석하고 잘못 인식된 단어를 문장에 맞도록 하였다. 구문분석하는 중에 수식의 의미도 해석하여 계산 결과를 출력해 내도록 하였다. 구문 분석을 사용하지 않은 방법에 비해서 잘못 인식할 가능성을 크게 줄였다.

## ABSTRACT

This paper presents the technique for voice calculator which can be operated only by voice. It is necessary to continous speech (or sentence) recognition that speech recognizer can interpret and analyze sentences of various forms under the syntactical rules. We adopted HMM and LR parsing method to voice calculator, parsed input sentence for correcting wrong recognized words and print out the result through semantic analysis. And we could get better performance than the system has no parsing method.

## I. 서 론

구문 정보를 이용하여 음성인식의 성능을 개선하기 위한 연구는 국외에는 종종 발표되어 왔으나[1][2][3] 국내에서는 문법적 정보를 음성인식에 적용하려는 노력은 거의 없었다[9]. 문법적 규칙을 갖는 어떤 문장을 해석하기 위해서는 그 문장이 바르게 쓰여졌는가 하는 사실을 판단할 수 있는 방법이 필요하다. 문장의 텍스트 입력인 경우에는 입력을 제대로 했을 경우 여러가지 문장 해석 기법을 이용하여 쉽게 올바른 문장인지를 검사할 수 있게 된다. 하지만 음성인식과 같이 올바로 발음 했음에도 불구하고 그 결과를 확신할 수 없는 경우에는 이와같은 문장해석 및 오류 복구 기능은 더욱 더 필요한 기능이 아닐 수 없다.

구문분석은 문법으로 음성인식 시스템의 언어영역을 표현하고 구문 분석을 통하여 단어단위의 열을 제거 또는 선택한다. 음성인식 시스템에서 가장 자주 사용되는 문법은 유한 상태 문법이다. 이것은 정규문법이라는 것과 같다. 유한상태 문법은 제한된 영역에서 인식성능을 개선하는 데 성공적인 역할을 했다[7]. 그러나 유한상태 문법은 대어휘 음성인식 시스템에서와 같이 인식에 변화있고 효율적으로 대응하기에는 충분하지 못한 문법이다.

촘스키(Chomsky)의 문법의 계층구조에서, context-free 문법은 다음으로 강력한 문법이다[5]. 즉 context-free 문법은 유한 상태 문법(finite-state grammer)보다도 더 유연하게 언어의 특성을 표현할 수 있다. context-free문법에 대하여 CYK(Cocke-Younger-Kasami)알고리즘이나 Earley 알고리즘과 같은 구문 분석 알고리즘도 있으나 일반화된 LR 구

* 광운대학교 전자계산기 공학과
접수일자 : 1993년 9월 10일

문 분석 알고리즘은 자연언어 문법에 가장 효율적인 표현이 가능한 알고리즘이다[7].

LR 구문 분석은 원래 프로그래밍 언어를 위해 개발 되었으며 입력심볼을 읽는 과정에서 에러가 발생하면 바로 에러를 찾을 수 있는 강력한 구문 분석 능력을 갖는다. LR 구문 분석은 구문 분석 테이블을 이용하여 좌측부터 입력 심볼을 읽으며 되부름(backtracking) 없이 구문분석을 수행하므로 실시간 응용에도 적합하다.

LR 구문 분석 테이블을 구하는것이 어렵긴 하지만 이미 제작 되어있는 구문 분석기 제작 시스템(YACC)을 이용하여 구문 분석 테이블을 구성하여 구문 분석기를 구현하도록 하면 간단히 해결된다.

HMM(Hidden Markov Model)은 확률적인 음성 모델링으로 음성의 다양한 변화에 쉽게 대처할 수 있는 가장 성공적인 모델로 잘 알려져 있다. 본 연구에서는 단어단위로 구분 발성한 음성 데이터를 실시간으로 인식하도록 TMS320C30보드를 이용하여 실시간으로 녹음과 동시에 계수를 구하여 빠른 인식이 될 수 있도록 하였다.

그림1은 일반적인 언어번역 절차와 음성계산기 처리절차를 설명하고 있다. 실선으로 표시된 그림이 언어번역 절차이고, 점선으로 표시 된것이 음성계산기 처리절차 부분을 나타낸다[6].

마이크를 통하여 수식을 단어 단위로 입력하면, TMS320C30 보드가 저장되어 있는 HMM모델들과 비교하여 최적의 확률을 내어 단어를 PC로 넘겨주게 된다. 즉 이것은 텍스트에서 키보드 입력하는 것과 같은 의미이다. 입력된 단어토큰은 구문 분석기로 입력된다. 구문 분석기는 구문 분석 발생기에 의해 생성된 구문 분석 표로 움직인다. 만일 받아들일 수 없는 토큰이 입력 되었을 때는 단어인식기에서 다음 후보 토큰을 가져온다. 그리고 그것을 구문 분석기에 입력한다. 문법에 맞는 단어이면 구문 분석기가 지시하는대로 행동하면서 의미해석을 통해 수식을 계산하게 된다.

## Ⅱ. 구문분석

### 2.1 일반적인 구문분석 방법[4][5]

구문분석은 크게 하향식(top-down) 방법과 상향식(bottom up) 방법으로 나누어진다.

Top-down 방법은 시작 심볼 (S)로부터 정의된 문법의 생성 규칙을 적용하여 유도과정에 의해 주어진 스트링(w)과 같은 문장을 찾는 과정으로 다음과 같이 표현한다.

$S \Rightarrow \ldots \Rightarrow w$

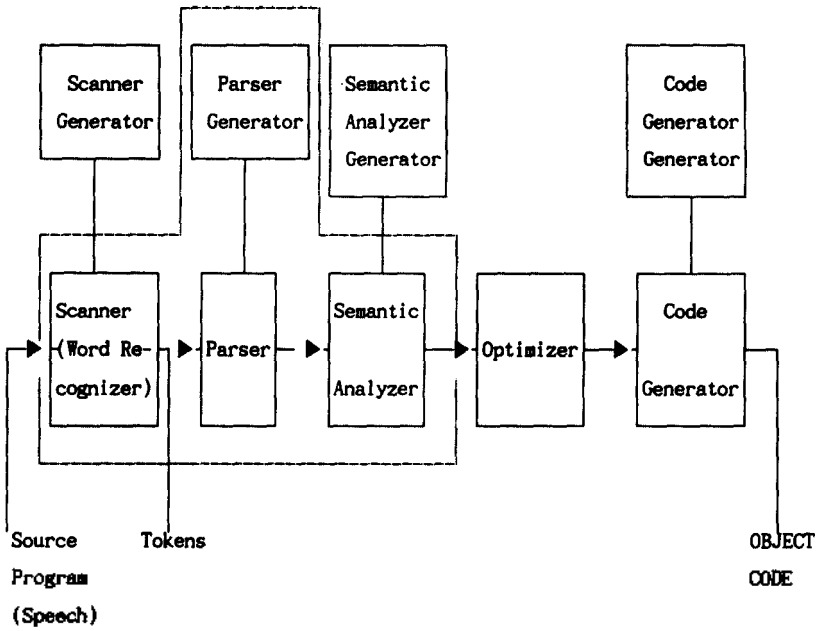


그림 1. 일반적인 언어번역 절차와 음성 계산기처리 절차

Fig 1. General language translation procedure and voice calculator procedure

시작 심볼로부터 유도를 해 나가면서 주어진 스트링을 생성할 수 있으면 올바른 문장이고, 그렇지 않으면 틀린 문장이다.

대표적인 구문분석 방법으로는 LL(Left to right scanning, Left parse) 구문 분석법이 있으며 backtracking하지 않고 결정적으로 구문 분석할 수 있는 방법으로 top-down 방법으로는 가장 많이 사용한다.

Bottom-up 방법은 주어진 스트링(w)으로 부터 문법의 시작 심볼(S)로 대치되는 구문 분석방법이다. 어떤 문장 형태에서 생성규칙의 우변을 좌변으로 바꾸는 것을 reduce한다고 한다. 따라서, bottom-up 방법은 주어진 스트링으로부터 reduce를 계속하여 시작 심볼로 갈 수 있으면 올바른 문장이고 그렇지 않으면 틀린 문장으로 간주한다. Reduce과정을 기호 $\Rightarrow$로 나타내면 bottom-up방법은 다음과 같다.

$w = \Rightarrow \cdots = \Rightarrow S$

이런 방법중의 대표적인 구문분석 방법으로는 LR 구문 분석법이 있다. 왼쪽에서 오른쪽으로 읽으면서(Left to right scanning) 출력으로 우파스(Right parse)를 생성하므로 붙여진 이름이다. LR 구문 분석기는 스택의 top과 입력 심볼에 따라 구문 분석을 결정할때 구문 분석 테이블을 참조한다.

### 2.2 LR 구문 분석법[4][5][6]

LR 구문 분석은 원래 프로그래밍 언어를 번역하기 위해 발전되어온 기법으로 context-free 문법에 적용할 수 있다. LR 구문 분석기는 두개의 구문테이블(action 테이블과 goto테이블)로 구성되어 있는 LR 구문분석 테이블을 이용하여 구문분석을 한다. action 테이블은 스택의 맨 위에 있는 현재상태와 현재 입력 심볼 a로 부터 다음 분석을 결정한다. 4가지의 action이 있는데 shift, reduce, accept, error이다.

· Shift : shift는 LR 구문 분석 표에서 S#으로 나타난다. #은 새로운 상태번호이다. 표에서 이 항목을 참조하면 현재의 입력 심볼을 스택에 push하고 새 상태 번호 #을 push한다.

· Reduce : reduce는 R#으로 나타나며, #은 적용될 생성규칙 번호이다. 스택의 맨위는 생성규칙의 우변을 포함하고 있으므로 주어진 생성규칙에 따라 스택에서 생성규칙 우변요소 만큼 제거하고 난 후 표의 GOTO부분을 다음 상태를 얻기위하여 참조한다. 그리고 생성규칙의 좌변과 새로운 상태를 스택에 push한다.

· Accept : accept는 표의 accept항에 의하여 나타나며, 이 항목을 얻었을 경우에는 입력 스트링을 받아들인다. 이것으로 구문분석은 끝난 것이다.

· Error : 표에서 어떤 내용도 지시되지 않으면 구문 에러이다.

LR 구문 분석 알고리즘은 다음과 같다.

```
스택은 상태 0으로 초기화한다.
입력 스트링 끝에 $를 추가한다.
While Action ! = Accept And Action ! = Error Do
    Let Stack = s0X1s1...XmSm and
      remaining Input = ai ai+1... $
       {s는 상태번호 ; X'는 종단, 비종단기호의 열}
    Case Table[sm, ai] is
      S# : Action : = Shift
      R# : Action : = Reduce
      Accept : Action : = Accept
      Blank : Action : = Error
EndWhile
```

LR 구문분석을 적용하기 위해서는 LR 구문 분석 테이블이 필요한데 이것을 수작업으로 구하기는 어려운점이 많다. 따라서 이미 제작 되어있는 구문 분석기 제작 시스템을 많이 이용하는데 가장 많이 이용하는 것이 UNIX의 유틸리티로 제공되는 YACC이다. Context-free 문법을 입력하면 구문 분석 테이블(y.output)과 구문 분석하는 소스프로그램을 작성하여 y.tab.c라는 화일로 출력해준다. 일반적인 언어번

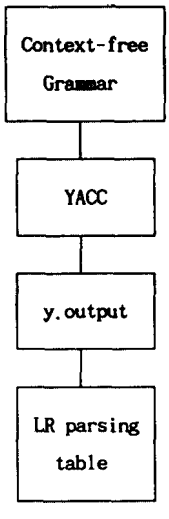


그림 2. LR 구문 분석 테이블을 구하는 절차
Fig 2. Procedure for LR parsing table

역에서는 y.tab.c라는 화일을 컴파일하여 사용하지만 음성에 적용하기 위해서는 다른 행동을 취해야 할 때가 많으므로 y.output에 의하여 출력된 내용을 PC에서 작성된 LR 구문 분석 프로그램에 이용하도록 변환하여 사용하였다.

## Ⅲ. 인식 시스템

### 3.1 인식 시스템의 구성

실시간 처리를 위하여 단어인식은 TMS320C30 보드에서 수행된다. 입력음성을 70-3500Hz 대역 필터를 통과시킨후, 16비트를 한 샘플로 8KHz 샘플링 한다. 그리고 $1-0.95z^{-1}$로 pre-emphasis를 한후, 128샘플을 한 프레임으로 Hamming 창을 씌우고, 10차의 LPC cepstrum을 구한다. 이 벡터들을 VQ에서 찾아 코드워드 인덱스 열을 구하고, HMM모델과 Viterbi 알고리즘을 이용해 비교하여 가장 확률이 높은 단어 인덱스와 모든 단어의 확률 계산 결과를 PC에 넘겨준다. PC에서는 인식된 단어를 구문 분석 테이블에서 검사하여 만일 error action이 나온다면, 다음으로 높은 확률을 갖는 단어를 검사하는 것을 분법에 맞을 때 까지 반복한다. 그림3에 전체적인 개략도를 나타낸다.

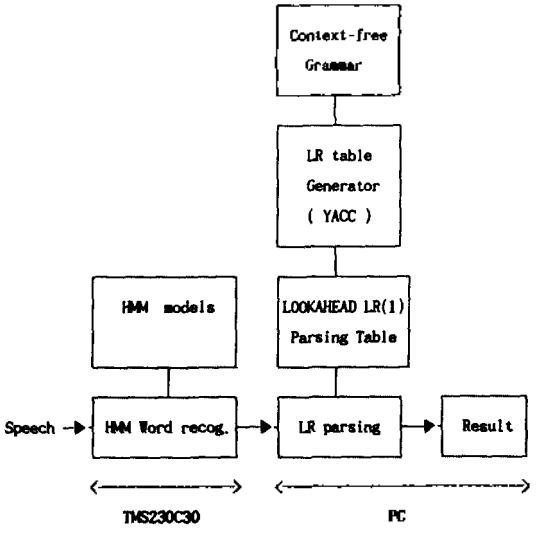


그림 3. HMM-LR 인식 시스템
Fig 3. HMM-LR recognition system

PC와 TMS320C30 간의 인터페이스와 인식및 구문분석 처리 절차를 블럭도로 나타내면 그림4와 같다.

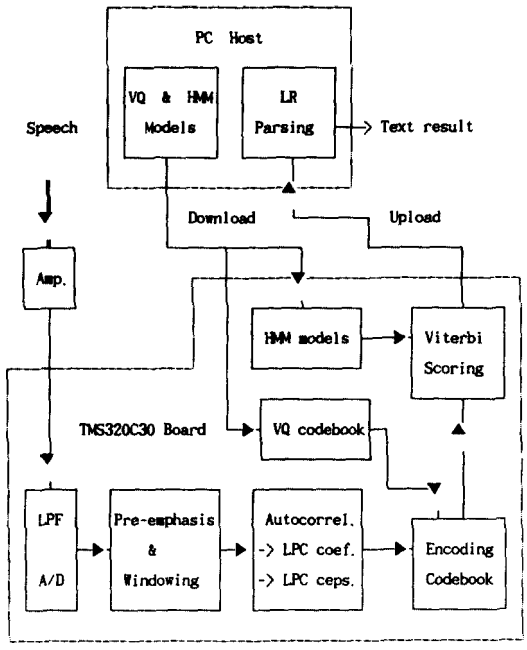


그림 4. PC와 TMS320C30 인식시스템 블럭도
Fig 4. Block diagram of recognition system using PC and TMS320C30

수식의 발성은 단어단위로 이루어지게 되며 각 단어 발성이 끝날 때마다 자동으로 끝점 검출을 해서 인식단어를 PC에 넘겨주게 된다.

PC에서는 각 reduce action에 주어진 수식의 값을 계산해 나아간다. accept action이 나오게 되면 계산 결과를 출력하고 다음 문장을 대기하는 상태로 들어간다.

### 3.2 단어목록

다음 표1에 음성 계산기에 사용된 단어 목록을 나타낸다.

표 1. 음성계산기 단어목록
Table 1. VOICE CALCULATOR WORD LIST

| 번 호 | 단 어 | 기 호 | 분 류 |
|---|---|---|---|
| 1 | 더하기 | + | |
| 2 | 빼 기 | − | |
| 3 | 곱하기 | * | |
| 4 | 나누기 | / | 연산자 |
| 5 | 지수승 | ^ | |
| 6 | 괄호열고 | ( | |
| 7 | 괄호닫고 | ) | |
| 8 | 공 | 0 | |
| 9 | 일 | 1 | |

| | | | |
|---|---|---|---|
| 10 | 이 | 2 | |
| 11 | 삼 | 3 | |
| 12 | 사 | 4 | |
| 13 | 오 | 5 | 피연산자 I |
| 14 | 육 | 6 | (number) |
| 15 | 칠 | 7 | |
| 16 | 팔 | 8 | |
| 17 | 구 | 9 | |
| 18 | A | A | |
| 19 | B | B | |
| 20 | C | C | 피연산자 II |
| 21 | X | X | (variable) |
| 22 | Y | Y | |
| 23 | 는(은) | = | 문장끝기호 |
| 24 | 저장 | store | 명령어 |

### 3.3 생성규칙

YACC에 입력한 수식 생성규칙은 다음과 같다.

```
%token variable number store
%%
state  :  inst
       |  exp
       ;
inst   :  comm variable
       ;
comm   :  store
       ;
exp    :  exp'+' term
       |  exp'-' term
       |  term
       ;
term   :  term'*' p
       |  term'/' p
       |  p
       ;
p      :  factor'^' p
       |  factor
       ;
factor :  '-' factor
       |  varible
       |  number
       |  '('exp')'
       ;
```

종단 기호에는 variable과 number, store가 있고 variable은 A, B, C, X, Y라는 변수를 의미하고, number는 10개의 숫자로 구성된 단어열이며, store는 저장 명령어이다.

수식을 말하는 방법은 다음의 예와 같다.

2+(4-5/9) = 이 더하기 괄호닫고 사 빼기 오 나누기 구 괄호닫고 는
12+23 = 일이 더하기 이삼 은
A^2^3 = 에이 지수승 이 지수승 삼 은
store A = 저장 에이 는(현재 계산결과가 변수 A에 저장된다)
A-B = 에이 빼기 비 는
A = 에이 는(변수 A의 값이 출력된다)

### 3.4 에러 교정 알고리즘

TMS320C30 보드를 통해 입력된 단어는 우선 그 단어에 해당되는 문자로 변환된다. 예를 들어 '더하기'란 단어는 '+' 기호로 변환된다. 이 입력 심볼에 대한 다음 action을 구문 분석 테이블에서 스택의 맨 위 숫자가 나타내는 현재상태와 입력심볼이 만나는 곳에 지정되어 있다. 만일 이 action이 지정되어있지 않다면 현재 상태에서 지금의 입력 심볼은 나타날 수 없다는 것을 나타내는 에러 action이므로 화자가 올바른 문장을 말하고 있다고 가정한다면 지금의 입력 심볼은 오인식 되었다는 결론이 나온다. 따라서 HMM에서 다음으로 높은 확률을 갖는 단어를 인식 단어로 선정한다. 그리고 그 단어를 심볼로 변환한후 위와같은 절차를 거친다. action이 에러가 나오지 않을 때까지 하면 다음 입력 심볼을 문법에 맞게 처리할 수 있다.

## Ⅳ. 인식 실험 및 고찰

### 4.1 끝점검출

수식은 단어단위로 발성하고 인식은 실시간 처리해야 하므로 시작과 끝을 나타내는 종단점 검출은 음성입력과 동시에 실시간으로 이루어져야 한다. 음성샘플들이 타이미 인터럽트에 의해 버퍼에 저장되는 동안 전처리, 자동상관계수, 켑스트럼 계수등이 동시에 구해지게 된다. 이때 자동상관 계수의 첫번째 값 R0을 한 프레임의 에너지로 처리한다. 만일 R0이 무음구간을 나타내는 임계값 이하를 8프레임 이상 지속하게되면 단어의 끝으로 가정하고 끝점 검출을 하고 인식에 바로 들어간다.

### 4.2 실험자료와 HMM

남성화자 1인이 표1의 단어목록을 10회 발음하였다. 10회 발성한 데이터 중에서 3회의 전체 데이터는 64개의 코드워드를 갖는 VQ코드북 작성을 위해 k-means 알고리즘에 사용되었다. 그리고 5회의 데이터 모두 상태 4를 갖는 HMM을 구성하기위해 사용되었다. 10차의 LPC 켑스트럼을 특징 벡터로 이용하였다. 사용한 HMM 모델은 전형적인 left-to right모델이다.

### 4.3 인식실험

HMM에 대한 단어인식 실험과 저장된 데이터 베이스하에서 LR 구문 분석에 의한 단어교정율을 실험을 통하여 확인한다.

먼서 단어인식 실험은 24개의 단어를 10회 발성한 자료 전체를 대상으로 인식실험하였다. 240개 단어중 13개를 잘못 인식하여 단어 인식율은 전체 평균 95%가 나왔다.

인식 결과는 다음과 같다.

| 입력단어 | 인식단어 |
|---|---|
| 공 | 지장, 시수능, 잘호담고(각1개) |
| 이 | B (6개) |
| 사 | 삼 (1개) |
| 오 | 구 (2개), 공 (1개) |

두번째로 그룹단위의 단어 구분률 평가인데 각 그룹단위의 단어 구분율은 평균 97%였다. 이 실험은 위에서의 단어인식과 같은 실험이나 다른 관점에서 해석한다: 여기서의 그룹단위의 단어 구분율은 같은 그룹내에 인식이되면 인식이 올바로 되었다고 보는 것이다. 예를 들어 숫자음 "공"을 "구"로 인식한 것은 단어단위로는 잘못인식 한 것이지만 같은 숫자음 그룹내에서의 인식이므로 옳다고 평가한다는 것이다. 이렇게 평가하는 이유는 이 비율이 어떤 단어가 그룹외의 단어로 인식이 됐을 때 LR 구문 분석을 수행함으로 교정가능한 최대 비율을 나타내기 때문이다. 단어 구분율이 97%이므로 교정율은 이 3%에 달려있다. 즉 3%의 비율로 그룹간의 구분이 안될 경우가 발생하고 이 경우에 잘못된 단어를 LR 구문 분석이 잡아 줄 수 있는 것이다. 위의 오인식 단어에서 입력단어 "공"과 "이"인 경우만 다른 그룹에서 오인식이 됐으며, "사"와 "오"는 같은 그룹내의 단어로 구문적으로는 회복할 수 없는 오류이다.

### 4.4 구문분석 과정

다음은 입력된 한 수식에 대한 구문 분석 과정을 나타낸다. shift와 goto action은 생략하고 reduce action만 기술한다.

14+3*(A-3)=에 대한 구문 분석 과정

| 입력단어열 | 현재단어 | reduce action |
|---|---|---|
| | 14 | |
| 14 | + | factor : number |
| | | p : factor |
| | | term : p |
| | | exp : term |
| 14+ | 3 | |
| 14+3 | * | factor : number |
| | | p : factor |
| | | term : p |
| 14+3* | ( | |
| 14+3*( | A | |
| 14+3*(A | - | |
| | | factor : variable |
| | | p : factor |
| | | term : p |
| 14+3*(A- | 3 | |
| 14+3*(A-3 | ) | |
| | | factor : number |
| | | p : factor |
| | | term : term-p |
| | | exp : term |
| 14+3*(A-3) | = | |
| | | factor : (exp) |
| | | p : factor |
| | | term : term*p |
| | | exp : exp+term |
| | | state : exp |
| | | accept : state $ |
| | | |
| | | 5.000000 |

## V. 결 론

LR 구문분석기법과 HMM을 이용하여 음성 계산기 구현에 관해 연구 실험하였다. 가장 성공적으로 알려진 인식기법인 HMM과 컴퓨터 언어 번역에 주로 사용되고 있는 context-free 문법을 표현 할 수 있는 LR 구문분석 기법의 결합은 의미 없는 단어인식

만을 수행하는 HMM에 문법적인 제약을 주어 단어를 오인식할 가능성을 더욱 줄여 주었다. 수식을 입력하는 동안 구분분석을 통해 수식을 계산할 수 있도록 하여 음성계산기로서의 역할을 충분히 수행하도록 하였다.

## 참 고 문 헌


1. H. Sawai, "TDNN-LR Continuous Speech Recognition System Using Adaptive Incremental TDNN Training," ICASSP 91, 1991.
2. S.Nakagawa, "Spoken Sentence Recognition by Time-Synchronous Parsing Algorithm of Context-Free Grammar," ICASSP 87, April 1987.
3. H.Ney, "Dynamic Programming Speech Recognition Using a Context-Free Grammar," ICASSP 87, April 1987.
4. Alfred V. Aho, Jeffrey D. Ullman, Principles of Compiler Design, ADDISON WESLEY PULISHING COMPANY, 1977.
5. 오세만, 컴파일러 입문, 성익사 1988.
6. Karen A. Lemone, Design of Compilers Techniques of Programming Language Translation, CRC Press 1992.
7. K. Kita, T. Kawabata and H. Saito, "HMM Continuous Speech Recognition Using Predictive LR Parsing," ICASSP 89, 1989.
8. L.R. Bahl, F. Jelinek, R.L.Mercer, "A Maximum Likelihood Approach to Continuous Speech Recognition," IEEE Trans. Pattern Anal. March. Intell. PAMI-5, 1983.
9. 은종관 외, "한국어 연속 음성 인식 시스템 개발," 한국음향학회 제10회 음성통신 및 신호처리 워크샵 논문집 제SCAS-10권 1호, 1993.



▲유 형 근 : 제10권 1호 참조
▲이 형 준 : 제10권 1호 참조
▲이 강 성 : 제11권 1E호 참조
▲김 순 협 : 제10권 1호 참조