論 文 論文 81-18-3-1

한글 Pattern에서 Subpattern분리와 인식에 관한 연구

(A study on the Partial Separation for Subpatterns and Recognition of the Hangeul Patterns)

李 柱 根*, 南 宮 在 贊*, 金 榮 建*
(Lee, J. K. Namkung J. C. and Kim, Y. K.)

要 約

이 논문은 한글 pattern을 subpattern으로 분리하여 인식하는 방법을 제안하였다.

한글 pattern을 6종류의 form으로 형식화하여 그의 표면구조를 식별하고 index mark와 window의 결합에 의한 새로운 algorithm에 의하여 한개의 form pattern을 2～4개의 subpattern으로 분리한다. 다음 subpattern을 regular U-tree grammar에서 production rule을 개선하여 tree의 frontier만으로서 식별한다. 종래의 tree grammar를 사용할 때에 비하여 production의 수가 $\frac{1}{3}$로 감소되고, automaton 처리가 $\frac{1}{3}$로 감소되며, 유연성이 높다.

한글의 실용문자 1,600종의 표준 인쇄체에 대해서 실험한 결과 subpattern의 분리율은 99.1 %이고 subpattern의 인식율은 100%이다.

Abstract

In this paper, the recognition method of Hangeul patterns with the partial separation for the subpatterns is proposed.

First, Hangeul patterns are formalized into six formal patterns and their surface structures are discriminated.

Second, two to four subpatterns from one form pattern are separated by the new algorithm combined with Index mark and Window.

Hangeul patterns are recognized with only frontiers of the tree by defining the regular U-tree grammar for the separated subpatterns.

Compared with the previous tree grammar, this grammar reduces the production rules to 1/3 and simplifies automaton processing and has more flexiblity.

By the simulation result for 1,600 characters of Hangeul patterns, separation rate of subpatterns (24 or 44) is obtained 99.1 % and recognition rate is obtained 100 %.

1. 서 론

문자 및 도형 pattern의 인식에 관한 연구는 근래 급속한 발전을 하여 office automation 화로부터 원격지의 물체판별에 이르기까지 확장되고 있으며, 인식방법으로서는 크게 나누어 결정론적 방법과[1~7, 17~19] syntactic 방법으로[8~15] 분류된다.

결정론적 방법은 pattern의 특징검출에 의한 pattern의 구조와 특징 관계를 취급하는 것으로서 통일된 형식이 확립되어 있지 않고, syntactic 방법은

* 正會員, 仁荷大學校 工科大學 電子工學科
(Dept. of Electronics Engineering, Inha Univ.)
接受日字 : 1980年 12月 29日

pattern의 구성요소를 primitive로 택하고, syntax rule에 의하여 pattern의 구조를 parsing하여 인식한다.

Syntactic 방법은 formal language 이론의 기술과 automaton에 기초를 두고 있으나, 대부분의 문자, 도형 pattern은 다차원으로 기술되므로, 일차원의 string으로 취급되는 형식 언어이론을 직접 도입하기에는 적합하지 않다.

따라서 다차원 문법인 tree grammar가 일반적으로 이용되고 있다.[8,9,10,11]

그러나 pattern의 구조에 따라 primitive가 달라지므로 primitive을 선택하는 문제는 아직 일반화된 형식이 없고, boundary나 skelton에 의하여 묘사된 pattern에서 line segment를 primitive로 택하는 것이 일반적이다.

한글에 대한 인식 문제는 1969년부터 시작되었으며, 최초의 조합문자 인식 연구를 시도한 것은 논문(1)에서 비롯된다. 이것은 한글은 30종류의 form set로 형식화하여 구조적 본질을 분석하고 그것을 다시 6종류로 형식화하여 구조적 특징을 검출하여 pattern 공간을 가변분리하는 방법에 관한 것이다.[1]

논문(7)은 좌표 table의 analysis 및 synthesis에 의한 방법으로서 좌표치의 위치정보와 stroke의 결합관계, 길이의 상대비로서 자음과 모음을 분리하는 방법의 것이다. 유연성이 높아서 필기체에도 응용이 가능 하다는 것이 지적되었다.

논문(9)은 syntactic 방법에 의하여 graph matrix를 top down적으로 점차적으로 식별하는 방법이다. 때문에 자음과 모음이 떨어져 있는 문자는 가능하지만 서로 연결된 문자는 분리되지 않는다. 즉 "국"의 경우는 "ㄱ"로 "는"의 경우는 "ㄴ"과 "ㅜ"로 오인식 하였다.

한글 pattern의 인식 문제는 몇 백자를 대상으로 인식율이 얼마라고 한다든가, 자모가 서로 떨어져 있는 문자를 대상으로 분리했다고 하는 것은 큰 뜻이 없고, 자모가 수직, 수평으로 복잡하게 연결된 약 25%에 해당되는 문자형을 어떻게 처리하여 분리하였는가가 더욱 중요하다.

본 논문에서는 index와 window의 결합에 의한 새로운 algorithm에 의하여 문자 pattern에서 부분 pattern(subpattern)을 분리하는 방법을 제안하고, 또 분리된 부분 pattern을 종래의 regular U-tree grammar의 production rule을 개선하여 tree의 frontier의 식별만으로서 판정하는 방법을 제안하였다.

그 결과 종래의 tree grammar에 의한 방법에서 입력 pattern이 변형되면 생성규칙이 달라지고, automaton처리에서 expansive tree grammar를 사용함으로써 생성규칙의 수가 증가하는 결점을 제거하였다. 이때 production 수가 $\frac{1}{3}$로 감소되고, automaton 처리가 $\frac{1}{3}$로 간단해진다.

2. Pattern의 부분분리 Algorithm

1) 전처리

입력 image pattern을 CCD sensor에 의하여 표본화하였다. 이때 표본 pattern은

$$P(i,j)=V(1\leq i\leq m,\ 1\leq j\leq n,\ V\in\{0,1\})$$

의 mxn matrix m=n=32bit로 구성하고, 문자는 V=1인 pixel의 집합을 형성한다. 이때 형성된 2치 pattern에서 각 p(i, j)를 pixel로한 pattern내의 각 pixel (V=1)의 topological성질에 따라 전체 pattern의 local operation을 주어 세선화된다.[16] 입력 과정에서 noise로 인하여 1~2 digit로 형성되는 고립 pixel과 필체의 관습으로 인하여 형성되는 "뿔"은 세선화 과정에서 제거한다. 이들 pixel은 문자의 본질적인 요소가 아니므로 제거해도 pattern의 구조에는 영향을 주지 않는다. 골격화된 pattern graph에서 open node, 굴곡점, 분기점을 각각 T, D, B로 하고 loop Ⓛ 구조를 특징으로 추가 한다. 또 pattern 의 처리를 쉽게 하기 위하여 입력 pattern의 크기를 X•Y로 normalize 한다.

한글 pattern에서 6개의 form pattern으로 형식화하여 분리하는 것이 가장 효과적이라는 것은 논문(1)에서 제시되었지만, 본 논문에서는 그 6형식의 form pattern에 그림 1과 같이 windowing하는 방법을 모색한다.

본 논문에서는 일차적으로 문자 하나 하나를 직접 대상으로 하지 않고 유사형의 집합으로 형식화하여 포

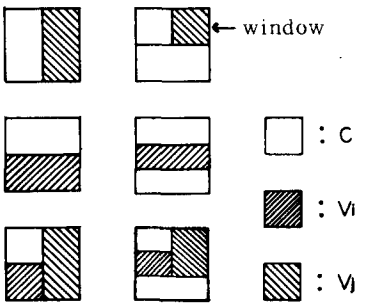


그림 1. Subpattern의 가상영역
Fig. 1. The virtual regions of subpatterns.

괄적인 기하학적 form을 표면구조로 정의한다.

또 표면구조의 부분영역을 부분 patten의 집합으로 정의하고, 그 부분 pattern 집합내에 포함된 개개의 pattern 구조를 내면구조로 정의한다.

이들 정의에 의한 집합 구조를 차례로 처리함으로서 개개 문자를 대상으로 할때에 비하여 system이 체계적이고 간결하며 명쾌한 답을 얻을 수 있다.

2) 표면구조의 식별 Algorithm

i) 횡모음 판정

최좌측에서 T를 시작점으로 하는 횡선분으로서 수색방향의 끝점이 T이거나 B이며 그 길이가 X/2 이상일때 횡모음을 판정하며 이 때의 횡선분을 Xm 이라 정의한다.

ii) 종모음 판정

우측 최상단에서 T를 시작점으로 하는 종선분으로서 수색방향의 끝점이 T이거나 B 이며 그 길이가 Y/2 이상일 때 종모음을 판정하며 이 때의 종선분을 Ym 이라 정의한다.

iii) 종성판정

① Ym= 1 일때는 Ym의 하단부에 B, D, L 가 존재하면 종성이 있음을 판정한다.

② Xm= 1, Ym = 0 일때는 Xm의 하단부에 B, D, L 가 있으면 종성을 판정한다.

i),ii),iii)의 판정이 처리되면 pattern의 표면구조가 식별된다.

a) $Xm=1, Ym=0, C_k=0$ 이면 CV_i

b) $Xm=1, Ym=0, C_k=1$ 이면 $CV_i C_k$

c) $Xm=0, Ym=1, C_k=0$ 이면 CV_j

d) $Xm=0, Ym=1, C_k=1$ 이면 $CV_j C_k$

e) $Xm=1, Ym=1, C_k=0$ 이면 $CV_i V_j$

f) $Xm=1, Ym=1, C_k=1$ 이면 $CV_i V_j C_k$

이상의 경우와 같이 pattern 의 표면구조는 6 개의 formal pattern 으로 분류된다.

3) Subpattern의 분리 algorithm

본인들은 앞서 window-cuting algorithm[4] 에 의해서 pattern의 부분분리를 실현한 바 있다. 이에 추가하여 본 논문에서는 index 개념을 도입하였다.

i) 정의

Subpattern이 상호 연결된 B점 및 모음성분중 다른 subpattern과의 연결 가능한 T점을 index mark (#)라 정의한다.

단 index mark(#)점은 다음 algorithm에 의하여 부가된다.

ii) Index algorithm

Xm, Ym의 시작점 T로부터 수색하여 B가 있으면 Bx, By라 하고 Bx, By에 연결된 단선분을 수색하여 그 끝점이 T, D, B이면 다음과 같이 index mark(#)를 부가한다.

(a) Bx → T, T = #

(b) Bx → D, Bx±1 = #(+ : 하단, − : 상단)

(c) Bx → B, Bx±1 = #(+ : 하단, − : 상단)

(d) By → T , T = #

(e) By → D , By − 1 = #

(f) By → B , By + 1 = #

Xm과 Ym이 연결되었을 때는 By − 1에 #를 부가한다.

(g) Xm → By, By − 1 = #

Xm의 우단 및 Ym의 하단에도 #를 부가한다.

(h) Xm → T , T = #

(i) Ym → T , T = #

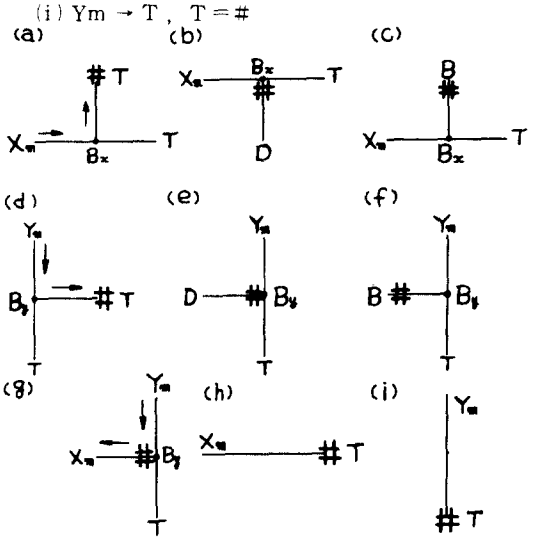


그림 2. Index의 예
Fig. 2. Examples of index.

iii) Window algorithm

a) 초 성

① Xm 및 Ym 상에 #가 존재하면 #(Xm)을 경계로한 상측영역 및 #(Ym)을 경계로한 Ym의 좌측영역을 window로 한다.

② Xm, Ym상에 # = 0 이면 Xm의 상측영역 및 Ym의 좌측영역을 window로 한다.

b) 중 성

① Ym = 0, Xm = 1 이고 Xm에 연결된 단선분의 종단에 #가 존재할 때는 #점과 Xm을 포함하는 영역을 window로 한다.

② Xm = 0, Ym = 1 이고 Ym에 연결된 단선분의 종단에 #가 존재할 때는 #점과 Ym 을 포함한 영역을

window로 한다.

③ Xm=1, Ym=1일때 Xm, Ym에 연결된 단선분의 종단에 #가 존재할 때 #(Xm) 및 Xm을 포함하는 window와 #(Ym) 및 Ym을 포함하는 widow로 한다.

c) 종 성

① Xm=1이고 Xm의 바로 하단에 #=1이면 #점을 window의 상단으로 한다.

② Xm=1이고 #=0이면 V 의 하측에 B, D, L이 존재할 때 Xm의 하단을 window의 상단으로 한다.

③ Ym=1이고 Ym의 하단에 #=1이면 #점을 window의 상단으로 한다.

④ Ym=1이고 Ym의 하단에 #=0이고 Ym의 하단부에 B, D, L이 존재할 때 Ym의 하단을 window의 상단으로 한다.

이상의 algorithm에 의하여 처리된 예는 그림 3 및 4와 같다.

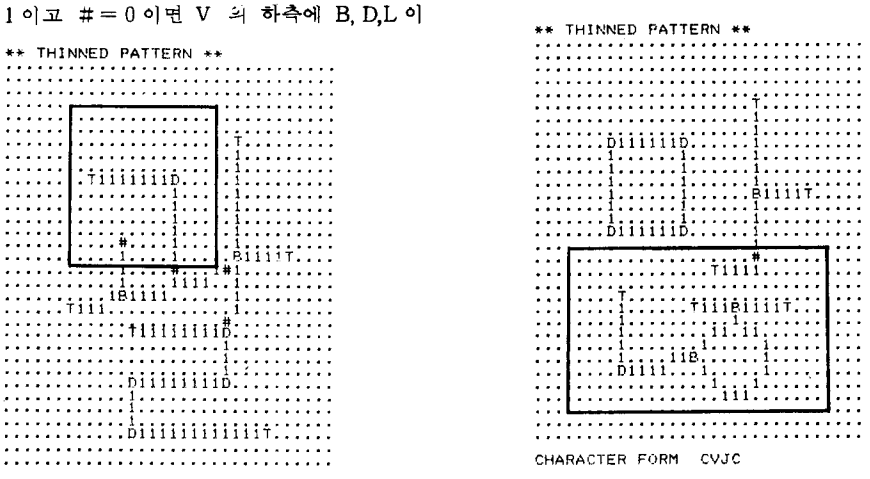


그림 3. 처리 예
Fig. 3. Processing examples

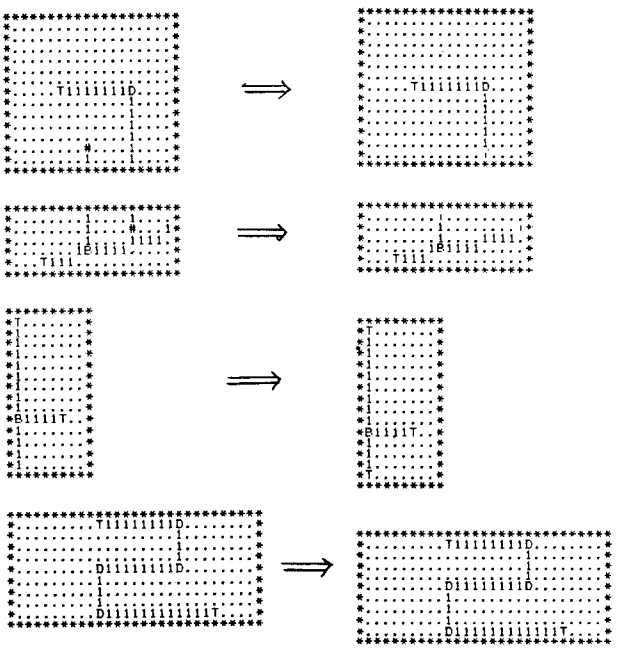


그림 4. "괄"의 처리 예
Fig. 4. Processing examples of the "KWAL"

3. 부분 Pattern의 인식 Algorithm

앞에서 분리된 부분 pattern의 집합에서 개개의 내면구조를 인식하기 위한 primitive는 그림 4와 같다.

종래의 tree grammar를 이용하는 graph pattern의 인식에서는 line segment의 시작점을 start node로 하였기 때문에 tree의 production과 automaton 처리가 극히 복잡하다.

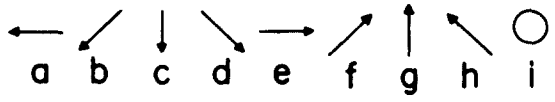


그림 5. Primitives
Fig. 5. Primitives

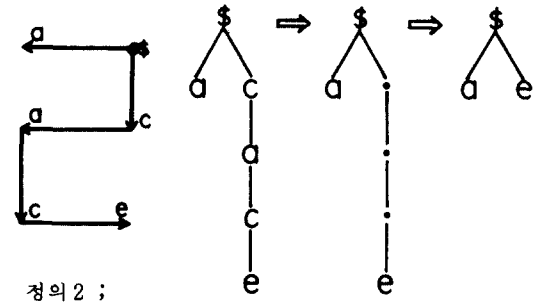
본 논문에서는 이와는 달리 한글기본 pattern의 구조는 최초에 나타나는 연결 node를 start node로 정의할 때 모든 pattern의 frontier가 전부 달라진다는 점에 착목하여 tree grammar에서 nonterminal symbol을 제거하고 frontier만으로서 tree를 생성하였다. 이 방법은 저자들이 앞서 중간발표를 한 바 있다.[8]

정의 1 ;

i) RUTG(Regular U-tree grammar)의 production에서 root와 frontier 사이의 node간의 directed edge는 단지 연결점만으로 사용된다.

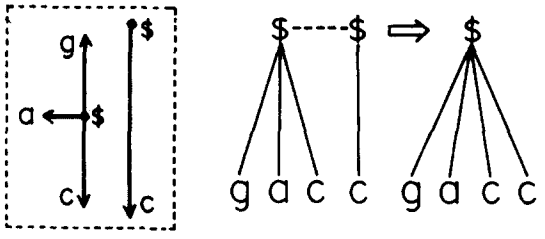
ii) Start node는 최상단으로부터 최초에 나타나는 연결 node를 선택한다.

iii) 연결 node가 존재하지 않은 pattern에서는 좌측 또는 최상단을 start node로 한다.



정의 2 ;

두개 이상의 분리된 graph pattern이 window내에 존재할 때 두 start node는 결합한다.



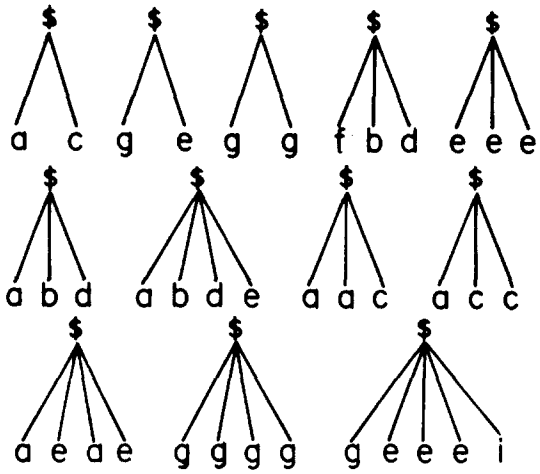
이상의 정의에 따라 종래의 tree grammar에서 보는 parsing이 현저히 개선될 뿐 아니라 정의 2에서 또 하나의 중요한 개념이 제기된다. 즉 종래의 tree system에서는 일반적으로 graph가 분리되었을때 한 개 tree로 생성할 수 없기 때문에 pseudo operator를 primitive로 사용하는 예가 있다.[9] 그러나 본 논문에서는 처리 과정에서 필연적으로 windowing되고, 또 nonterminal symbol은 고려치 않으므로 두 개 graph는 결합되어 tree가 구성된다. 이상으로부터 RUTG는

i) $G = \langle B, \sigma, P, \Gamma \rangle$

$A = (\$, a, b, c, d, e, f, g, h, i)$

$B : \Gamma \cup A$

$P :$



ii) U-TA(U-tree automaton)

$M = \langle Q, \sum, F \rangle$

$Q = \{ g_\Gamma \} \cup F$

$\sum = \{ t_s, t_r \}$

$r = \{ a, b, c, d, e, f, g, h, i \}$

$F = \{ q_{il}, q_{jm}, q_{kn}, q_{il}', q_{kn}' \}$

q_{il} ; $l = 1, 2, 3, \cdots\cdots, 19$ (C(자음))

q_{jm} ; $m = 1, 2, 3, \cdots\cdots, 14$ (V(모음))

q_{kn} ; $k = 1, 2, 3, \cdots\cdots, 11$ (C_k(바침))

q_{il}', q_{kn}' ; 변형 pattern

a) t function

$t_r(\lambda) = q_r$

$q_{i\neg} = t_s(\overline{q_a, q_c})$

$q_{i\llcorner} = t_s(\overline{q_g, q_e})$

$q_{i\text{ㅂ}} = t_s(\overline{q_g, q_g})$

$q_{i\text{ㅅ}} = t_s(\overline{q_f, q_b, q_d})$

$q_{iㅌ} = t_s(\overline{q_e, q_e, q_e})$
$q_{iㅈ} = t_s(\overline{q_a, q_b, q_d})$
$q_{iㅊ} = t_s(\overline{q_a, q_b, q_d, q_e})$
$q_{iㅋ} = t_s(\overline{q_a, q_a, q_c})$
$q_{iㅌ} = t_s(\overline{q_a, q_c, q_c})$
$q_{iㅍ} = t_s(\overline{q_a, q_e, q_a, q_e})$
$q_{iㅎㅎ} = t_s(\overline{q_g, q_g, q_g, q_g})$
$q_{iㄴㅎ} = t_s(\overline{q_g, q_e, q_e, q_i})$

b) M function

$\rho(s(a,c)) = t_s(\rho(a,c))$
$= t_s(\rho(a), \rho(c))$
$= t_s(t_a(\lambda), t_c(\lambda))$
$= t_s(q_a, q_c)$
$= q_{iㄱ}$

$\rho(s(g,e)) = t_s(\rho(g, e))$
$= t_s(\rho(g), \rho(e))$
$= t_s(t_g(\lambda), t_2(\lambda))$
$= t_s(q_g, q_e)$
$= q_{iㄴ}$

$\rho(s(g,g)) = t_s(\rho(\rho(g, g))$
$= t_s(\rho(g), \rho(g))$
$= t_s(t_g(\lambda), t_g(\lambda))$
$= t_s(q_g, q_g)$
$= q_{iㅂ}$

$\rho(s(e,e,e)) = t_s(\rho(e,e,e))$
$= t_s(\rho(e), \rho(e), \rho(e))$
$= t_s(t_e(\lambda), t_e(\lambda), t_e(\lambda))$
$= t_s(q_e, q_e, q_e)$
$= q_{iㅌ}$

$\rho(s(f,b,d)) = t_s(\rho(f,b,d))$
$= t_s(\rho(f), \rho(b), \rho(d))$
$= t_s(t_f(\lambda), t_b(\lambda), t_d(\lambda))$
$= t_s(q_f, q_b, q_d)$
$= q_{iㅅ}$

$\rho(s(g,g,g,g)) = t_s(\rho(g,g,g,g))$
$= t_s(\rho(g), \rho(g), \rho(g), \rho(g))$
$= t_s(t_g(\lambda), t_g(\lambda), t_g(\lambda), t_g(\lambda))$
$= t_s(q_g, q_g, q_g, q_g)$
$= q_{iㅎㅎ}$

$\rho(s(a,b,d,e)) = t_s(\rho(a, b, d, e))$
$= t_s(\rho(a), \rho(b), \rho(d), \rho(e))$
$= t_s(t_a(\lambda), t_b(\lambda), t_d(\lambda), t_e(\lambda))$
$= t_s(q_a, q_b, q_d, q_e)$
$= q_{iㅍ}$

$\rho(s(a,e,a,e)) = t_s(\rho(a, e, a, e))$
$= t_s(\rho(a), \rho(e), \rho(a), \rho(e))$
$= t_s(t_a(\lambda), t_e(\lambda), t_a(\lambda), t_e(\lambda))$
$= t_s(q_a, q_e, q_a, q_e)$
$= q_{iㅍ}$

$\rho(s(g,e,e,e,i)) = t_s(\rho(g,e,e,e,i))$
$= t_s(\rho(g), \rho(e), \rho(e), \rho(e), \rho(i))$
$= t_s(t_g(\lambda), t_e(\lambda), t_e(\lambda), t_e(\lambda), t_i(\lambda))$
$= t_s(q_g, q_e, q_e, q_e, q_i)$
$= q_{iㄴㅎ}$

4. Simulation 결과

한글의 실용문자는 1,600[11]여 자로 밝혀져 있으며, 국정교과서의 표준인쇄체와 1,600 종류를 실험의 대상으로 하였고 그 중 자모가 서로 떨어져 있는 문자를 분리하는 것은 별의미가 없으므로 대표적인 문자만 택하고, 주로 자모가 서로 연결된 CV_iC_k, CV_jC_k, CV_iV_j, $CV_iV_jC_k$ 형의 문자를 주로 관심의 대상으로 하였다.

표 1.

분자수	pattern forms	subpattern 분리 (%)	분리된 subpattern의 인식 (%)
1600	CV_i	100	100
	CV_iC_k	100	
	CV_j	100	
	CV_jC_k	98	
	CV_iV_j	100	
	$CV_iV_jC_k$	96.6	
	평 균	99.1	100

Subpattern의 인식율 실험결과는 표 1과 같다.

본 방식은,

1) 우선 표 1과 같이 형식화된 formal pattern 의 포괄적인 표면구조를 식별한다.

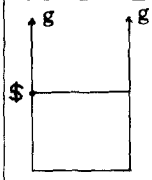
2) 다음에 index-window에 의하여 부분 구조를 분리한다.

3) 분리된 부분 pattern 집합내에 포함된 개개의 pattern을 식별함.

Form의 식별률은 100%이고, subpattern의 분리률은 99.1%이다. 이 때, 약 1%의 error는 극히 유사한 문자 "의"와 "익", "과"와 "파" 등에서 생기기 쉽고 thining 과정에서 "긋"과 "웃" 등에서 main algorithm만으로서는 애매해지는 예가 있는데, 이것은 보조기능의 부가로서 해결이 가능하다. 부분 pattern은 44개로 할 것인가, 33개, 24개로 분리할 것인가는 설계자의 판단에 맡길 것이나 본 방식에서는 어느 쪽으로도 분리할 수 있는 모든 information 이 들어 있다. 그러나, 실제로 pattern 인식 system은 컴퓨터의 입력장치란 점을 감안할 때 단말장치와 연동하게 되고, 또 단말장치에서 문자를 합성하게 되므로 컴퓨터 내부 정보는 44개로 분리하는 것이 유리하다고 판단된다.

부 록

본 방법의 예



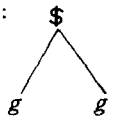
RUTG;

$G = \langle B, \sigma, P, \Gamma \rangle$

$A = (\$, a, b, c, d, e, f, g, h, i)$

$B = \Gamma \cup A$

P :



UTA;

$q_{\#} = t_s(q_g, q_g),\ q_g = t_g$

$e(\$(g, g)) = t_s(e(g, g))$

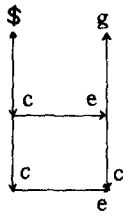
$= t_s(e(g), e(g))$

$= t_s(t_g(\lambda), t_g(\lambda))$

$= t_s(q_g, q_g)$

$= q_{\#}$

종래 방법의 예



RUTG;

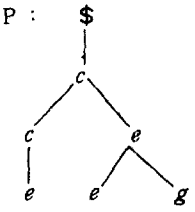
$G = (v, r, p, s)$

$V_T = (\$, a, b, c, d, e, f, g, h, i)$

$r(\$) = 1,\ r(c) = \{2, 1, 0\}$

$r(e) = (2, 0),\ r(g) = c$

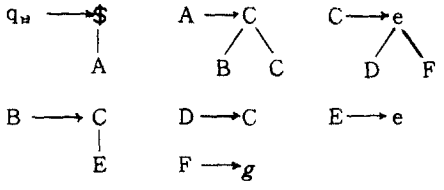
P :



Expensive Grammar;

$G(t) = (v, r, p', s)$

$V_N = (A, B, C, D, E, F)$



UTA;

$q_{\#} = f_s(q_A)\quad q_A = f_c(q_B, q_C)$

$q_B = f_c(q_E)\quad q_C = f_e(q_D, q_F)$

$q_D = f_c\quad q_E = f_e\quad q_F = f_g$

$\rho(\$(c(c(e), e(c, g))))$

$= f_s(\rho(c(c(e), e(c, g))))$

$= f_s(f_C(\rho(c(e), e(c, g))))$

$= f_s(f_C(f_C(e(e), f_E(e(c), \rho(g)))))$

$= f_s(f_C(f_C(f_e(\lambda)), f_e(f_C(\lambda), f_g(\lambda))))$

$= f_s(f_C(f_C(q_E), f_e(q_D, q_F))))$

$= f_s(f_C(q_B, q_C))$

$= f_s(q_A)$

$= q_{\#}$

본 방식의 분리방법과 인식방법은 대단히 효과적이라는 것을 실험결과 확인하였으며 실험 장치는 micro-computer system으로서 구성하고 앞에서 기술한 분리 algorithm, tree grammar 및 automaton에 대한 처리 program은 basic 으로 하였다.

본 연구의 효과를 보이기 위하여 부록에 비교 예를 보였다.

參 考 文 獻

1. J. K. Lee, "Korean character Display by Variable Combination and Its Recognition by Decomposition Method," Ph. D. dissertation in Keio University, JAPAN. 1972.
2. J. K. Lee, " A Method for the Recognition of Printed Korean Characters," J. KIEE, Vol. 7, NO 4, pp. 198-209, Dec. 1969.
3. J. K. Lee, " Recognition of Printed Korean Characters (Ⅱ), " J. KIEE, Vol. 7, NO. 1/3 pp. 130-136, Nov. 1970.
4. 이주근, 남궁재찬, 김영건, " Window-Cutting Algorithm에 의한 Character Form 식별 및 subpattern 분리" 전자공학회 학술발표 논문집, vol. 3, NO. 1, pp. 31-35, 1980.
5. J. K. Lee and H. K. Kim, " Automatic Recognition of Handwritten Hangeul by the Phase Rotation," J. KIEE, Vol. 13, NO. 1, pp. 23-30. Mar. 1976.
6. M. H. Kim and C. M. Park, " Algorithms for Machine recognition, Noise Cleaning and Thinning of the Korean Characters, " Proceedings of the International Computer Symposium, Vol. 2, pp. 513-519, 1975.
7. J. H. Lee, " An Algorithm for Automatic Pattern Recognition of Printed Korean Characters, " MS. dissertation in Inha Univ. 1980.
8. 이주근, 김영건, 남궁재찬, " Character Pattern의 부분분리에 의한 인식 Alogorithm" 정보과학 학술발표 논문집, pp. 43-50. Apr. 1980.
9. T. K. Kim and T. Agui, " A Study on the Pattern Recognition of Korean Characters, " J. KIEE, Vol. 14, NO. 5, pp. 15-21, Dec. 1977.
10. K. L. Williams, " A Multidimensional Approach to Syntactic Pattern Recognition," Pattern Recognition, pp. 125-137. Sep. 1974.
11. K. S. Fu and B. K. Bhargava, "Tree System for Syntactic Pattern Recognition," IEEE Trans. on computers, Vol. C-22, No. 12, pp. 1087-1099, Dec. 1973.
12. K. S. Fu and Rosnfeld, " Pattern Recognition and Image Processing," IEEE Trans. on Computers. Vol. c-25, NO. 12, pp. 1336-1346, Dec. 1976.
13. L. Kanel, " Patterns in Pattern Recognition: 1968-1974," IEEE Trans. on Information Theory, Vol. IT-20, NO. 6 pp. 697-722, Nov. 1974.
14. K. Nakata, " Recognition of Handprinted Alphanumeric and Special Characters, " IECE, Vol. 58-D, NO. 8. pp. 442-449, Aug. 1975.
15. K. C. You and K. S. Fu, " A Syntactic Approach to Shape Recognition Using Attributed Grammars, " IEEE Trans. on SMC, Vol. SMC-9, NO. 6, Jun. 1979.
16. C. J. Hilditch, " Linear Skeleton form Square Cupboard, " Machine Intelligence IV, 1966.
17. W. H. HIGHLEYMAN, "Linear Decision Functions with Application to Pattern Recognition," IRE Vol. 50, No. 6, pp 1501-1514, June 1962.
18. J. A. Feldman and Y. Yakimovsky, "Decision theory and artificial intelligence: A. semantics-based region analyzer," Artificial Intelligence, Vol. 5, pp 349-471, 1975.
19. S. Raudys and V. Pikelis, "On Dimensionality, Sample Size, Classification Error, and Complexity of Classification Algorithm in Pattern Recognition," IEEE Trans. on PAMI, Vol. PAMI-2, No. 3, pp 243-251, May 1980.