論 文

論文 75-12-2-2

한글 文字의 認識을 위한 代數的 構造

(Algebraic Structure for the Recognition of Korean Characters)

李 柱 根*·朱 薰**

(Joo K. Lee *and* Hoon. choo)

요 약

이 논문은 한글문자의 자동인식을 위한 기초적인 연구로서 기본문자의 구조에 대해서 검토하였다. 기본문자를 凹구조, 선분구조 및 문자 graph의 node와의 연결관계 등 구조를 세가지 측면에서 집합 및 군론에 의한 대수적인 분석을 하고 또 그들의 각 구조의 복잡성에 대한 계층을 고찰하였다. 나아가서 10개의 모음은 한 요소의 Affine 변환에 의한 연속회전으로 이루어지는 회전변환군 속에서 다수의 동치관계가 존재한다는 것을 기술하므로써, 한글문자의 인식에 있어서는 topological 특징외에 기하적 성질이 특히 중요하다는 것을 아울러 지적하였다.

Abstract

The paper examined the character structure as a basic study for the recognition of Korean characters.

In view of concave structure, line structure and node relationship of character graph, the algebraic structure of the basic Korean characters is are analized.

Also, the degree of complexities in their character structure is discussed and classififed.

Futhermore, by describing the fact that some equivalence relations are existed between the 10 vowels of rotational transformation group by Affine transformation of one element into another, it could be pointed out that the geometrical properties in addition to the topological properties are very important for the recognition of Korean characters.

1. 서 론

근래의 정보처리 과정에서 인간에 의한 방대한 작업과정을 기계로서 대행하려는 연구의 일환으로서 문자의 자동인식에 주목하게 되어, Roma문자, Kana문자와 같이 간단하고 종류가 적은 문자(인쇄체)는 이미 실용화하고 있다. 그러나 모든 문자를 인식할 수 있는 통일된 방법은 아직 발표가 없으며, 특히 한글문자와 같이 그 수가 방대하고 또 특수 구조의 문자는 일반적인 방법으로서는 거의 인식이 불가능하다. 한글문자(인쇄체)의 자동인식과 조합문자의 발생 및 한국어의 정보량등에 대하여는 저자에 의하여 몇가지의 발표가 있다.[6]-[7]

본 논문에서는 주로 한글기본문자의 구조적 접근에 대한 본질적인 문제에 주목한다.

인간의 인식 구조속에서 유독 우리들은 문자라고 하는 대상에 한해서 그의 인식모형을 고려할 때, 문자구조에 대한 연구는 필연적으로 인식장치의 구조에 반영되는 것으로서, 이러한 관점에서 문자의 구조에 대한 해석은 그의 본질에 대한 규명과 계층문제에 귀착하게 되는 것이다. 따라서 문자의 구조의 해석이든 그의 역인 생성이든 간에 그들 조직을 관측하는 문제는 보다 높은 인식장치의 연구에 디딤돌이 될 것이다.

문자의 구조에 대한 연구로서 piage는 일찌기 3~6세 아동들의 인식구조의 심리적 발달과정을 관측하였

* ** 正會員, 仁荷大學校, 電子工學科
Dept, of Electronic Engineering, Inha University

接受日字 : 1975年 2月 20日

고, H. sherman[1]은 문자의 선분과 node 관계로서 Roma문자의 graph를 구성하였으며, W. stallinge[2]는 한자의 graph를 8개의 반한 선분으로 양자화 하였다. S. Morie[3] 등은 선분 및 오목구조의 개념을 Roma 문자의 대수적 구조에 도입하였고, Shaw는 graph요소로서 head와 tail을 가진 선분으로서 Roma문자를 구성하였다. 또 Masuda[5] 등은 Kana문자의 graph에서 node 상태의 특징을 문자의 인식에 도입하였다.

본 논문은 S. Morie,[3] K.L. Su[4]등의 연산개념을 한글 기본문자의 구조에 도입하고, 凹 및 선분구조에 대한 집합 및 군론(group theory)의 관점에서 대수적 구조에 대한 분석과 문자 graph의 연결관계를 표현한 node의 특징을 관측하였으며, 또 그들의 각 구조에 대한 계층을 분류, 고찰하였다. 특히 10개의 모음은 한 요소의 Affine변환에 의한 연속된 연산으로 이루어지는 최저변환군 속에서 다수의 동치류(equivalence relation)가 그 집합속에 존재한다는 것을 제시하므로서, 한글 문자의 인식에 있어서는 topological 관점이외에 기하적 성질이 특히 중요시 된다는 것을 다음에 제시한다.

2. 凹구조에 의한 접근

일반적인 집합함수 및 군론의 관점에서 凹구조를 관측하고, 그의 이론적 배경을 한글 기본문자의 구조에 비추어 보기로 한다. 凹구조를 양자화한 기본적 패상태의 하나인 (ㄴ), (ㄷ), (ㅁ) 등은 문자의 기호로서 표현할 수 있다. 그런데 凹구조 속에서 문자기호의 기본 생성원을 어떻게 선택할 것인가 하는 것이 첫째 문제이다. 엄격히 말하면 (ㄴ)은 凹 및 선분구조에 어느 것에도 포함될 수 있는 성질을 갖고 있다. 그것은 인속적인 variation을 속명적으로 가지는 문자기호의 본질적인 특징의 하나로서 구태어 이것을 선분구조인가 아니면 凹구조인가를 분류하는 것은 의미가 없으며, 다만 문자기호에 있어서, 가장 기본적인 패상태의 하나로서 (ㄴ)가 존재하며, 이의 방향을 고려할 때(ㄴ, ㄱ, ㄴ, ㄱ)의 4개의 요소를 가장 간단한 기본 생성원으로 가지는 하나의 집합으로서 凹구조를 생각할 수 있다. 또 다른 패상태 (ㅂ)도 凹구조속에 존재하며, 이의 방향을 고려할 때(ㅂ, ㄲ, ㄷ, ㄸ)도 역시 凹구조의 기본 생성원속에 포함시킬 수 있다. 따라서 凹구조속에 존재하는 이들 생성원의 상호 관계를 살펴볼 때, 우리가 집합 및 군론속에서 볼 수 있는 세가지의 관계 중에서 그것을 찾아 볼 수 있다. 즉 한 mapping의 graph, 동치관계 및 order relation[8][9] 등이 그것이다. 그런데 凹구조의 각 생성원의 관계를 inclusion relation의 관점에서 볼 때는 reflexive, 비대칭 및 transitive의 세가지의 조건을 만족하는 것을 예로서 볼 수 있다. 즉 임의의 집합을 $M=(m_1, m_2, m_3\cdots\cdots, m_n)$, inclusion relation을 $\subset$으로 할 때

i) reflexive;

$m_i \subset m_i$, $m_i \in M$

예) ㄱ⊂ㄱ, ㄷ⊂ㄷ, ……

ii) anti-symmetric;

$m_i \subset m_j$, $m_j \subset m_i \Rightarrow m_i = m_j$

$m_i, m_j \in M$

예) ㄱ⊂ㄲ, ㄲ⊄ㄱ

ㄴ⊂ㄷ, ㄷ⊄ㄴ

iii) transitive;

$m_i \subset m_j$, $m_j \subset m_k \Rightarrow m_i \subset m_k$

$m_i, m_j, m_k \in M$

예) ㄱ⊂ㄲ, ㄲ⊂凹 ⇒ ㄱ⊂凹

따라서 凹구조의 생성원으로서 구성되는 집합(M,⊂)은 위의 세가지 조건을 만족하므로 반순서집합(partially ordered set)을 구성하게 된다. 이러한 凹구조의 반순서 집합에 있어서 생성원(ㄴ, ㄱ, ㄴ, ㄱ) 및 (ㅂ, ㄲ, ㄷ, ㄸ) 상호간의 관계를 poset diagram으로 표시하면 그림 1과 같이 된다.

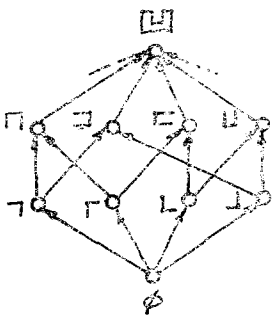


그림 1 반순서 집합의 diagam.
Fig.1 poset diagram

그림 1을 관찰하면 론리 설계에서 흔히 볼 수 있는 covering의 개념이 적용됨을 알 수 있다. 즉, 그림 1에서 link로서 연결된 한쌍의 요소중 위측의 요소(ㅂ, ㄲ, ㄷ, ㄸ)는 아래측의 요소(ㄴ, ㄱ, ㄴ, ㄱ)를 covering하고 있으며, 그 중간은 존재하지 않음을 알 수 있다. 여기서 ϕ와 그의 preceding 요소 사이에는 凹구조의 관점에서는 아무것도 존재하지 않지만 선분구조의 관점에서는 선분구조 자체가 이 사이에 존재함을 알 수 있다(Ⅲ절).

위에서 기술한 凹구조의 반순서집합에 있어서 그 기본생성원을 임의로 취하고, 거기에 적당한 binary operation을 정의하면 우리는 새로운 system(or group)

을 구성하게 되며, 이 system속에서 필요한 문자의 생성을 할 수 있는 동시에, 구성된 문자의 대수적 분류와 계층의 관측이 가능해 진다. 이상의 이론적 배경을 근거로해서 四구조속에서 생성 가능한 문자와 또 생성된 문자들이 어떠한 대수 system에 속하는 가를 규명해 본다.

지금, 四구조와 inclusion relation으로서 구성되는 반순서집합(M, ⊂)의 기본 요소들을 그의 생성원으로 하는 임의의 집합에 대한 binary연산은 여러가지로 정의한다.[3][4] 그런데 여기서 가장 중요한 문제의 하나는 복잡성에 대한 계층문제로서 주어진 연산(operation)에 대하여 가능한 계층이 간단하고 또 많은 문자를 구성할 수 있는 연산을 정의하는 것이 효과적인 것이다. 그런데 계층문제가 중요시되는 이유는, 궁극적으로 인식장치의 설계에 큰 비중을 갖고 반영되기 때문이다. 일반적으로 문자기호의 연산에 대한 identity의 존재는 문자의 본질상 무의미한 것이므로 monoid 이상의 group는 고려할 필요가 없는 것이다. 따라서 본 논문에서는 四구조속에서 가장 간단한 (ㄴ, ㄱ, ㅗ, ㄱ)을 생성원으로 하는 Abelian semi-group에 속하는 문자와 또 그것을 단계적으로 각 group의 조건을 만족하는 연산을 정의하므로써 필요한 한글 기본문자를 생성하고, 이들은 대수적으로 분류하는데 주목한다.

a) 대수 연산에서 closure, associative, Commutative의 세가지 조건을 만족하는 연산을 다음과 같이 정의한다.

정의 1; 연산 ＊는 왼쪽요소의 오른쪽변이, 오른쪽 요소의 오른쪽변과 중첩되도록, 왼쪽요소를 오른쪽으로 평행이동 하는 것으로 정의한다.〔이때 해변당이 없을 때는 가상변이 존재하는 것으로하여 중첩한다.[2][6]〕

이때 연산 ＊은 집합 $M=(m_1, m_2, \cdots, m_b)$로 표시하면 system (M, ＊)은

i) closure;

$m_i * m_j \in M,\ m_i, m_j \in M$

예) ㄱ＊ㄱ=ㄲ∈M

⋮

ii) associative;

$(m_i * m_j) * m_k = m_i * (m_j * m_k)$

$m_i, m_j, m_k \in M$

예) ㄴ＊(ㄱ＊ㄱ)=ㄴ＊ㄲ=ㅁ

(ㄴ＊ㄱ)＊ㄱ=ㅁ＊ㄱ=ㅁ

⋮

iii) commutative;

$m_i * m_j = m_j * m_i,\ m_i, m_j \in M$

예) ㅗ＊ㄱ=ㅁ　ㄱ＊ㅗ=ㅋ

ㄱ＊ㅗ=ㅁ　ㅗ＊ㄱ=ㅋ

⋮

위의 연산에 의하여 생성되는 한글 기본문자는

ㄱ=ㄱ＊ㄱ

ㄴ=ㄴ＊ㄴ

ㄷ=ㄴ＊ㄱ=ㄱ＊ㄴ

ㅁ=ㄱ＊ㄴ=ㄴ＊ㄱ

이들 4개의 문자는 closure, associative, Commutative의 세가지 조건을 만족하므로 Abelian semi-group에 속한다. 또 이들중 "ㄱ, ㄴ"의 문자는 四구조의 기본생성원(ㄴ, ㄱ, ㅗ, ㄱ) 중에서 선택되고, 문자 "ㄷ, ㅁ"은 이들 생성원에 정의 1의 연산에 의하여 생성된다. 따라서 四구조의 관점에서 볼 때 문자 "ㄱ, ㄴ"은 기본자음중에서 가장기본이 되며, 또 기본생성원속에 이미 포함되고 있으므로 계층이 1이고, 문자 "ㄷ, ㅁ"의 계층은 2가 된다.

다음, 이들 문자이의의 기본문자들은 어떤 group속에 포함되는 가를 살펴 보기로 한다.

b) 연산의 조건 closure, associative을 만족하고, commutative을 만족하지 않을 때는

정의 2; 연산 ＊은 왼쪽항의 오른쪽변이 오른쪽항의 왼쪽변에 중첩되도록 왼쪽항을 오른쪽으로 평행 이동하는 것으로 정의 한다(이때 해당 변이 없으면 가상변이 존재하는 것으로 한다).

이때 연산 ＊은 집합 $M'=(m_1', m_2', \cdots\cdots, m_b')$로 표시하면 system(M′, ＊)은

i) closure;

$m_i' * m_j' \in M',\ m_i,\ m_j \in M'$

예) ㄱ＊ㄴ=ㄱㄴ∈M′

⋮

ii) associative;

$(m_i' * m_j') * m_k' = m_i' * (m_j' * m_k')$

$m_i', m_j', m_k' \in M'$

예) ㄱ＊(ㄱ＊ㄷ)=ㄱ＊ㅍ=ㄲ

(ㄱ＊ㄱ)＊ㄷ=ㅍ＊ㄷ=ㄲ

⋮

iii) non-commutative;

$m_i' * m_j' \neq m_j' * m_i',\ m_j', m_j' \in M'$

예) ㅂ＊ㅁ=ㅂㅁ

ㅁ＊ㅂ=ㅁㅂ,　ㅂㅁ≠ㅁㅂ

⋮

이러한 연산에 의하여 생성되는 한글의 기본문자는 다음과 같다.

ㅗ=ㅗ＊ㄴ

ㅜ=ㄱ＊ㄱ

ㅛ=ㅗ＊ㄴ＊ㄴ=ㄴ＊ㄴ

ㅠ=ㄱ＊ㄱ＊ㄱ=ㅜ＊ㄱ

ㅛ=ㄱ＊ㄱ＊ㄷ

이들 "ㅗ, ㅜ, ㅛ, ㅠ, ㅛ"등 5개의 기본문자는 non-

commutative을 만족하므로 Non-abelian semi-gronp에 속한다. 다음 이들 문자의 계층은 생성원(ㄴ, ㄱ, ㅗ, ㅏ)를 기준으로 할 때 "ㅚ"는 계층이 6이 되어 가장 복잡해지고, "ㅛ", "ㅟ"는 계층이 3이 되며, "ㅗ, ㅜ"는 2가 된다. 이러한 계층문제는 현재 가장 발전했다고 하는 영상 인식방식에서 효과적이 된다. 다음 나머지 기본문자는 보다 더 광범위한 Groupoid에 대한 연산을 주고, 그들의 생성에 대해서 고찰해 본다.

c) 연산이 closure, non-associative, non-commutative의 조건에서는

정의 3; 연산 *은 왼쪽항의 아랫변과 오른쪽항의 윗변이 중첩되도록 왼쪽항을 오른쪽으로 수평, 수직으로 이동하는 것으로 한다(해당변이 없을 때는 가상변이 존재하는 것으로 한다[3][4]). 동시에 중변이 존재할 때는 중변을 중첩한다.

이 경우의 연산은 집합 $G=(g_1, g_2, \cdots\cdots, g_b)$로 표시하면 system $(G, *)$은

i) close;

$g_i * g_j \in G,\ g_i, g_j \subset G$

예) ㄱ * ㄷ = ㄹ $\subset G$

⋮

ii) non-associative;

$g_i * (g_j * g_k) \neq (g_i * g_j) * g_k$

$g_i, g_j, g_k \subset G$

예) ㄴ * (ㄴ * ㄱ) = ㄴ * ㅏ = ㅏ

(ㄴ * ㄴ) * ㄱ = ㅌ * ㄱ = ㅏ

ㅏ ≠ ㅏ

iii) non-commutative;

$g_i * g_j \neq g_j * g_i$

$g_i, g_j \in G$

예) ㄱ * ㄷ = ㄹ

ㄷ * ㄱ = ㄼ, ㄹ ≠ ㄼ

⋮

이때 연산은 Abelian semi-group 및 Non-abelian semi-group를 제외한 non-associative, non-commutative의 조건을 만족하는 Groupoid를 이룬다. 이때 생성되는 문자들은 항상 중변을 가지게 된다.

또한 정의 1, 2에서의 연산은 수평 이동만을 고려한데 반하여 정의 3에서는 한거름 나아가서 주어진 연산은 수직 이동을 고려한 연산이다(이상에서 연산 기호는 구별하였지만 구별하지 않아도 무방하다).

이와 같은 연산에 의하여 생성될 수 있는 한글 기본문자는 다음과 같다.

ㄹ = ㄱ * ㄷ

ㅛ = ㅗ * ㅁ

ㅇ = ㅁ * ㅂ

ㅌ = ㄷ * ㄴ

ㅋ = ㄱ * ㄱ

ㅑ = ㄴ * ㄱ

ㅕ = ㅗ * ㄱ

ㅒ = (ㄴ * ㄴ) * ㄱ

ㅖ = (ㅗ * ㅗ) * ㄱ

그리데 "ㅇ(ㅇ)"는 (ㅇ = ∩ * U), (⊂⊂ㅁ, U⊂ㅂ)로서 구성되지만 그의 topological 성질은 "ㅁ"와 동일하므로 "ㅇ"는 Abelian semi-group에 포함 가능하다. 이상 四구조에 의하여 생성되는 기본문자는 "ㄱ, ㄴ, ㄷ, ㄹ, ㅁ, ㅂ, ㅇ, ㅋ, ㅌ, ㅍ, ㅑ, ㅒ, ㅕ, ㅖ, ㅛ, ㅚ, ㅜ, ㅟ," 등의 18개의 문자이며, 나머지 문자들은 다음 선분구조에서 관측한다. 여기서 "ㅇ"와 "ㅇ"는 구별이 없이 통용되는 규칙에 따른다.

3. 선분구조 및 혼합구조에 의한 접근

四구조를 더욱 세분하여 양자화하면 선분구조에 속하게 된다. 그러나 어떤 구조에 치우치는가 하는 문제와 또 가장 간단한 구성이 가능한 특징이 어떤 구조에 속하는가 하는 문제에 주목하게 되는데, 이러한 고찰은 기계적인 인식장치의 구성문제에 반영되기 때문이다.

앞에서 한글 기본문자의 대부분은 四구조로서 구성 가능하지만, 사선을 가진 "ㅅ, ㅈ, ㅊ"와 직선 "ㅡ, ㅣ"의 문자는 선분구조속에 포함된다.

1. 선분구조

한글의 기본문자에 대한 선분구조는 기본적으로(ㅡ, ㅣ, ㅅ, ㅈ)의 4개의 생성원으로서 구성이 가능하다. 또 이들 선분을 각 2등분해서 보다 기본적인 8개의 선분으로 양자화하면 그림 2과 같은 8방향 선분을 나타낸다. 이들 선분을 생성원으로 했을 때는 앞에서 四구조로서 기술한 대부분의 문자도 구성이 가능하다. 그러나, 여기서는 四구조에 의하여 구성한 나머지의 문자들에 대해서 선분구조로서 고찰하겠으며, 다음에 그의 연산을 정의 한다.

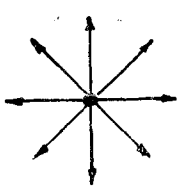
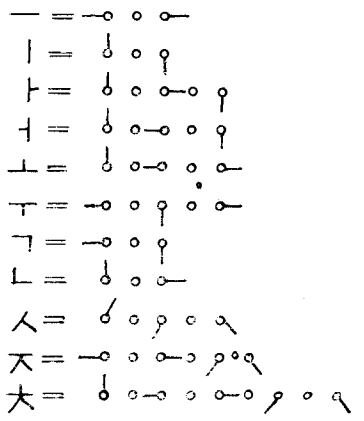


그림 2 8방향선분
Fig. 2 8 segments

정의 4; 연산(○)은 각선분의 끝단의 node (○)를 합치는 것으로 정의 한다.



표—A 문자이 선분연산

이들 문자들은 closure, associative 및 commutative 를 만족하므로 Abelian semi-group이다. 또 모음 "—, ㅣ"는 선분구조의 기본 생성원 중에서 이미 문자로서 선택되고 있으며, "ㄱ, ㄴ, ㅏ ㅓ, ㅗㅜ" 등은 선분구조와 凹구조의 어느쪽에도 포함되는 것을 볼 수 있다.

이것은 이들 문자가 양 구조의 친이점에 존재하는 문자들이기 때문이다. 이들 문자의 계층을 살펴보면 선분구조가 훨신 복잡해지는 경우가 있다. 즉 "ㅁ, ㄹ, ㅂ"의 문자의 경우 凹구조상에서는 계층이 4인데 반해서 선분구조에서는 계층이 4~6이 되는 것을 볼 수 있다. 물론 이경우 양구조의 차일뿐 그것이 실제적 인식기구의 설계에 전적적으로 간단해 진다고만 볼 수 있는 것은 아니다. 그것은 전자적 system 이외의 system 을 생각할 수 있기 때문이다.

2. 혼합구조

이것은 선분구조와 凹구조의 합성을 의미하는 것으로서, 한글의 기본문자에서는 "ㅎ"가 이 경우에 해당한다. 즉

$$\left.\begin{matrix} \cdot — = ㅗ \\ \cap * \cup = ㅇ \end{matrix}\right\} ㅗ + ㅇ = ㅎ \qquad (5)$$

이상에서 24개의 기본문자 중에서 23개는 선분 및 凹구조에 의하여 생성가능하였다. 그러나 유독 "ㅎ"자만은 선분구조의 "ㅇ"와 凹구조인 "ㅇ"의 혼합형태로 생성된다는 사실은 구조적인 측면에서 볼때 "ㅎ"는 이미 다른 기본문자와는 달리 완전한 2개의 기본구조의 조합임을 보여주고 있으며, 지극히 흥미로운 사실로서 조합문자의 개념이 엿보인다. 우연한 일치로 기본문자의 마지막 문자란 점에서 조합문자와의 경계면을 이루고 있다고 볼 수도 있다.

4. Graph에 의한 분석

앞에서는 주로 선분 및 凹구조의 관점에서 한글기본문자의 대수적 구조를 검토하였다. 이 절에서는 좀더 선분구조의 성질을 구명하고, graph의 연결관계에 대한 계층문제를 검토한다. H. sherman[5]은 문자의 선분을 graph의 branch로 보고, 연결점은 node로 본 문자 graph를 구성하였다. 그러나 graph는 node의 인접관계만을 표현하기 때문에 실제의 문자 인식에서는 미흡한 점이 있다.

여기서 우리는 한글의 모음집합의 한 특징으로 나타나는 역관계(ㅏ, ㅓ), (ㅗ, ㅜ), (ㅑ, ㅕ), (ㅛ, ㅠ)의 성질을 관측하고, graph의 node의 계층을 주목한다.

1. Affine변환에 의한 모음의 동치류

Affine변환은 잘알려진 수학의 한분야이지만, 본 논문에서는 그의 개념을 한글의 모음집합에 도입하여 그의 성질을 관측한다.

일반적으로 V^n공간의 한 basis를 $e_1, e_2 \cdots, e_n$, V^m공간의 basis를 $f_1, f_2, \cdots f_n$라 할 때

$$\left.\begin{matrix} V^n = (e_1, e_2, \cdots, e_n) \\ V^m = (f_1, f_2, \cdots, f_n) \end{matrix}\right\} \qquad (8)$$

V^n의 변환을 생각하면

$$T : V^n \rightarrow V^m$$

$$\begin{pmatrix} T(e_1) \\ T(e_2) \\ \vdots \\ T(e_n) \end{pmatrix} = \begin{pmatrix} a_{11}\ a_{21} \cdots a_{m1} \\ a_{12}\ a_{22} \cdots a_{m2} \\ \cdots\cdots\cdots \\ a_{1n}\ a_{2n} \cdots a_{mn} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{pmatrix} \qquad (9)$$

V^n의 임의의 vector element v는

$$\left.\begin{matrix} {}^{\forall} v \equiv V^n \\ v = \sum_{i=1}^{n} X_i e_i \end{matrix}\right\} \qquad (10)$$

(9), (10) 식으로 부터

$$T(v) = \sum_{i=1}^{n} X_i T(e_i) \qquad (11)$$

$$\therefore\ T(v) = \begin{pmatrix} a_{11}\ a_{12} \cdots a_{1n} \\ a_{21}\ a_{22} \cdots a_{2n} \\ \cdots\cdots\cdots \\ a_{m1}\ a_{m2} \cdots a_{mn} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} = Av \qquad (12)$$

이것은 Vector공간 V^n에서 Vector공간 V^n에의 Affine 변환을 의미한다. 지금 모음 "ㅏ"에 대하여 2차원 Euclid 평면상에서 각 node의 좌표축에 의한 Vector 집합 P로 표시하면 그림 3에서 $P_i\begin{pmatrix} x_i \\ y_i \end{pmatrix}$

$$P = [P_1, P_2, P_3, P_4] = \begin{bmatrix} 0, 0, 1, & 0 \\ 1, 0, 0, & -1 \end{bmatrix} \qquad (13)$$

이때 P에 대하여 Affine변환 (12)식에 대한 회전변환 (θ)는

$$R_n = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \qquad (14)$$

이다 (13)식에 대한 회전변환은 (14)식의 $\theta = 90°$ 일때

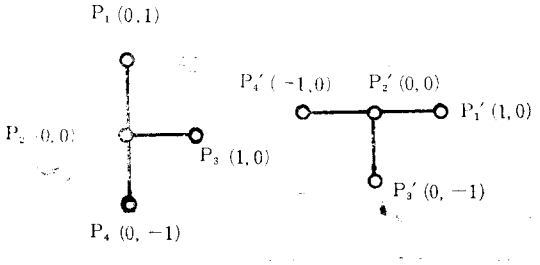


그림 3 그림 4

그림 4 그라프의 Affine 변환.
Fig.3 Affine transform(ㅏ↔ㅜ)

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & -1 \end{bmatrix}=\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (14)$$

따라서 새로운 좌표 P′는

$$P'\begin{pmatrix} x \\ y \end{pmatrix}=[P'_1, P'_2, P'_3, P'_4]=\begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (15)$$

$$\text{즉, } P'_1=\begin{bmatrix} 1 \\ 0 \end{bmatrix},\ P'_2=\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$P'_3=\begin{bmatrix} 0 \\ -1 \end{bmatrix},\ P_4'=\begin{bmatrix} -1 \\ 0 \end{bmatrix} \quad (16)$$

(16)식의 좌표에 의하여 다시 Euclid 평면상에 도시하면 그림 4와 같은 문자도형으로 된다. 다시 말하면 모음 "ㅏ"와 "ㅜ"는 상호 Affine변환에 의하여 다른 한쪽이 생성된다. 이와 같이 하여 문자(ㅏ, ㅗ, ㅓ, ㅜ) 및 (ㅑ, ㅛ, ㅕ, ㅠ)의 각 집합은 연속회전변환군 속에서 동치류가 존재한다는 것을 다음에서 볼 수 있다. 즉

θ	0°	90°	180°	270°
	ㅡ → ㅣ			
	ㅏ → ㅗ → ㅓ → ㅜ			
	ㅑ → ㅛ → ㅕ → ㅠ			

이와 같이 연속회전 변환에 의하여 형성되는 것이 그 집합속에 존재하기 때문에 문자 graph는 동치류가 되어 버린다.

문자의 구조상으로는 한 문자를 가지고 그것을 계속 회전시켜 mirror영상에 나타나는 도형을 실제문자로 하였다고 보면 이것은 기묘한 조직인 것이다. 그러나 이러한 문자는 현대 인식이론에서 식별이 지극히 어려운 문제를 초래한다. 따라서 한글의 문자인식에 있어서는 topological골격 만으로서는 인식이 극히 어렵다는 것 뜻하며 2차적으로 기하적 성질이 또한 중요하다는 것을 알시한다. 그것은 조합문자의 경우에도 나타난다. 즉 "어↔오, 몸↔물, 농↔움, 논↔국, 용↔운"등이 그 것이다. 나아가서, "쌀—쎌, 빵—뻥—삥, 깍—꺽—끽, 탐—텀, 똘—뚤—뜰"등과 같은 극심한 유사성(similarity) 문제는 일지기 문자의 자동인식 연구에서 시도된바 없는 것으로서 지극히 난해의 것이다. 이 문제에 대한 해결의 한 방법을 저자의 논문[3]에서 제안된바 있다.

2. graph의 계층문제

앞에서는 凹 및 선분구조에 대한 계층문제를 관측하였지만, 이 절에서는 문자 graph의 node 특징에 주목한다.[5]

즉 branch의 연결점만을 주시하고, 선분의 시작점과 끝나는 점을 그림 6의 graph와 같이 표시하기로 한다. 여기서 (○)은 node이며, 그 속의 숫자는 그 node에 연결된 branch의 수를 의미한다.

①은 선분의 끝, ③은 3개 branch의 연결점이며, 그 node에는 3개 branch가 연결되었다는 것을 의미한다. 이러한 연결점을 특징으로 한 계층상의 분류는 다음과 같다.

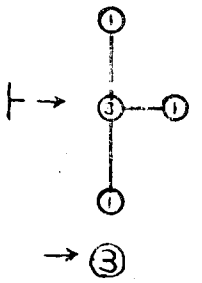


그림 5 문자 그라프의 "노드" 특징
Fig. 5 node feature of the character graph.

0	◎ ; ㅇ(○)
1	① ; ㅡ, ㅣ
2	② ; ㄱ, ㄴ
3	③ ; ㅏ, ㅓ, ㅗ, ㅜ, ㅅ

이상은 연결 node가 한개 뿐인 문자의 경우이며, 한 node에 연결된 branch수는 최대로 3개 이상은 존재하지 않는다.

다음에 node의 조합관계를 관측하면 "ㄷ"의 경우는 ②와 ②의 조합으로 이루어 진다.

4	②—② ; ㄷ
5	②—③ ; ㅋ(ㅊ), ㅈ(ㅉ)
6	③—③ ; ㅑ, ㅕ, ㅛ, ㅠ, ㅈ

위에서 ②—③이란 표현은 백지를 바탕으로한 결합과 선분결합과의 양쪽을 고려한 것을 의미한다. "…"는 선분이 없이 연결된 상태를 뜻하는데, 실제 기본문자에서는 "ㅎ", "ㅊ", "ㅌ"등이 이에 해당된다. 이상의 표에서 좌측 제1열의 숫자는 node ○속의 수치를 합한 것으로서, graph 전체의 node 상태를 가장 간단히 정량화한 것이며, 그것은 계층을 의미한다. 위에서

"ㅏ", "ㅛ"는 계층이 ②—③으로 같은데, 이두 문자의 구별은 branch의 방향이 중요시 되는 것이다.

즉 그림 7에서 보인 바와 같이 topological 골격보다 기하적 방향이 그들 문자의 구별을 가지온다.

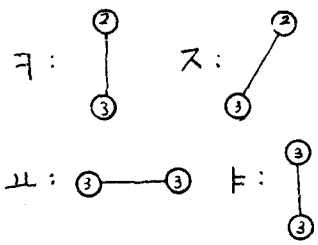


그림 7 node 특징의 유사상
Fig. 7 Similarity of node feature

이상에서 주의할 것은 "ㅈ"는 그림 6와 같이 표시함에 따라 계층의 차이가 생기게 되는데, 이점 필기체의 경우에 교려에 넣어야 한다. 다음은 3개의 node로서 구성되는 경우의 계층을 살펴보면

7 | ②—③—② ; ㅌ
  | ①…③—③ ; ㅛ, ㅊ

또 node가 4개인 경우는

8 | ②—②—②—② ; ㅁ, ㄹ
10 | ③—③—②—② ; ㅂ

이들 구조에서는 계층이 가장 복잡한 기본문자는 "ㅂ"이다. 또 "ㅎ"는 ①…①…◎로서 계층을 2로서 표시 가능하며, 한글의 기본자모에 node 검출에서는 최대의 계층은 10으로 표현 가능하다.

5. 결 론

한글 기본문자의 구조적인 성격을 파악하기 위해서 四구조, 선분구조 및 문자 graph의 연결관계등 3가지의 조직 관점에서 검토한 결과

(1) 24개의 기본문자중에서 18개 문자가 四구조에 의하여 생성되었으며, "ㄱ, ㄴ, ㄷ, ㅁ"등은 주어진 연산에 의하여 Abelian semi-group 속에서 생성되었고, 기타 문자들은 이들로 부터 계단적인 발견에 의하여 "ㅗ, ㅜ, ㅛ, ㅠ, ㅍ"는 Non-abelian semi-group, "ㄹ, ㅂ, ㅇ, ㅌ, ㅋ, ㅏ, ㅓ, ㅑ, ㅕ"등은 Groupoid 속에서 생성되었다.

(2) "ㅡ, ㅣ" 및 "ㅅ, ㅈ, ㅊ"는 선분구조에 의하여 생성되며, Abelian semi-group에 포함된다.

(3) "ㅇ"는 "ㅁ"에 포함되며, "ㅎ"은 四구조와 선분구조에 의하여 형성되는 혼합구조를 이룬다.

(4) "ㅁ, ㄹ, ㅂ"은 四구조에서는 계층이 2인데 반해서 선분구조에서는 가장 복잡한 문자이다. 이와 같은 사실은 선분구조의 표현과 四구조의 표현의 상보적인 특징을 나타내고 있는데 기인된다고 생갈된다. 이점을 착목하였을때, 한글 기본문자의 인식장치의 설계에 있어서 그들 각각의 특징을 상보적으로 고려하는 것이 보다 더 효과적이 될 것으로 생각된다.

(5) 모음은 두 군으로 나누어 그들 은 각각 한 문자의 Affine변환군에 의하여 다수의 문자가 동치류가 생성되었다 이것은 한글의 자동인식에 있어서는 topological 골격만으로서는 식별이 극히 어려워지며, 기하적 성질이 특히 중요시 된다는 것을 명시하였다.

(6) 한글 문자의 기본 생성원으로서는(ㅡ, ㅣ, ㅅ, ㅅ)의 선분과 loop ㅇ 및 연결점이 중요하다고 보인다. 그러나, 세가지 과정의 구조에 대해서 광범위 검토하였으므로 견해에 따라서 다각도로 그 특징을 선택할 수 있을 것으로 생각되며, 한글 문자의 자동인식 문제에 다소의 자료가 될 것으로 기대된다.

이 연구는 산학협동 재단의 연구비에 의하여 이루어진 것을 밝히며, 동재단과 지원하여준 인하대학에 감사 한다.

참 고 문 헌

1. H. Sherman; A quasi-topological method for the recognition of line patterns. Information processing, proceeding of U.C. (1960)
2. W. Stallings; Recognition of a printed chinese characters by automatic pattern analysis computer Graphics Vol.1 No.1 (Apr. 1972)
3. S. Morie; Algebraic structure of characters. J.J. E. Apr. 1974 and sept. 1974.
4. K.L. SU; Symbolic Representation of the chinese Written Language (1) Georgia Jnatitute of Tech. Research report No.E. 21—620—SU—1 (Nov. 1972)
5. I. Masuda; Machine Recognition of Hand printed Japanese Characters. J.ECE vol. 1 55—D No.10.
6. Joo K. Lee; Korean character recognition by decomposition method and its display by combination method. Ph. D. dissertation in keio university. 1972. 12.
7. Joo K. Lee; Korean charactor display by variable combination method. Keio Engineering Reports Vol. 26 No. 10. 1973.
8. B. Baumslag; Group Theory. MacGraw-Hill. 1968.