세종머쉰과 투링머쉰의 등가성에 관하여

한국전자통신연구소
기초기술연구부
정 희 성

1. 개 요

이 논문의 목적은 훈민정음의 구성원리를 현대의 컴퓨터 과학의 입장에서 다시 고찰하여 그 계산적 원리를 구성하고자 하는데 있다. 1930년대를 출발점으로 연구된 인간의 계산 과정에 대한 흥미는 Cybernetics라는 인간기계론의 사조에서 시작되었다. 현대의 컴퓨터의 원리적 동작은 1936년에 영국의 수학자 Turing에 의해서 제안된 Turing머쉰에서 구해지고 있다. Turing에 의한 계산가능성의 제창은 그 간단한 구성에도 불구하고 원리적으로는 현대의 어떤 컴퓨터와도 동등한 문제해결 능력을 가지고 있다. 한편 훈민정음은 음성언어를 표기언어로 바꾸는 기능을 갖는 현대풍으로 말하자면 인간에 의해 운용되는 소프트웨어 시스템이다. 이 논문은 이 시스템의 기능적 구조가 Turing 머쉰과 동등한 동작원리를 갖고 있음을 증명하는 준비적 자료와 그 방법론에 관해 설명한다.

1. 머리말

어느 시스템을 논할때 그 시스템이 갖는 지적기능의 해명은 지극히 중요하다. 지적기능의 규명의 범위는 크게 둘로 나눌 수 있다. 하나는 컴퓨터 과학적 측면이요, 또 하나는 인지과학적 측면이다. 전자는 시스템이 처리하고자 하는 목적과 기계적, 기능적 구성론의 규명이며, 후자는 인간-기계론에서 본 정합성의 규명이다. [1]

어느 시스템의 특질은 일반적으로 추상적인 과정에서 기술된다. 다시 말해서 정보를 처리한다 혹은 계산한다라는 추상적 과정의 분석에서 시스템의 목적, 기능이 논해진다.

훈민정음은 음성언어를 문자언어화 하는 하나의 기호처리계(symbol system)이다. 그 시스템이 갖는 목적이라든가 기능은 마땅히 추상화된 계산과정에서 논해져야 함은 물론이다.

이 논문에서는 훈민정음을 정보 처리시스템으로 간주하고 그 기능적 분석을 위하여 컴퓨터 과학의 계산 개념에서 그 원리를 구한다. 1936년에 A. M. Turing[2]는 인간의 계산과정을 분석하여 유한상태 오토마톤에 한줄의 입출력 테입을 붙여 기억을 무제한으로 사용토록 한 원시적인 테입기계를 고안하여 모든 계산은 이 테입기계로 실행할 수 있다고 주장하는 계산기의 이론을 전개했다. 같은 해 A. Church는 직관적인 의미로 계산할 수 있는 자연수상의 함수는 귀납적 함수(recursive function)라고 제창했다[3]. 두개의 개념은 각각 독립적으로 연구되었으나 Turing의 이론으로 계산할 수 있는 함수와 귀납적 함수의 개념은 동등하다는 증명을 얻었다. 또 K. Gödel[4]의 부호화에 의하면 모든 이산적 정보처리는 원시적으로는 이 원시적 테입기계에 의해 실행할 수 있다는 결과를 얻었다.

이 논문의 결과는 훈민정음의 계산원리에 따르면 Turing 기계와 동등한 원리적, 동작적 계산성을 갖음을 증명한다.

2. 알고리즘의 일반적 특징

알고리즘(algorithm)이란 어느 문제가 주어졌을 때 그것을 푸는 기계적 산법을 말한다. 여기서 알고리즘이라는 것이 갖는 일반적인 특징에 관해서 서술한다.

(1) 알고리즘은 어느 계산(혹은 기호의 유한적인 배치에서 표현된 추상적 대상의 처리)의 절차이다.

(2) 알고리즘과 그 처리대상 및 처리의 결과로 얻어진 대상은 어떤 언어 혹은 기호를 이용하여 쓰여져, 각각을 식별할 수 있는 유한적 표현을 갖는다. 그 표현에 따라 처리대상의 범위는 명확히 정해진다.

(3) 알고리즘의 표현은 그 표현을 이해하고 그 표현이 가리키는 지시에 따라 알고리즘을 실행할 수 있는 실행자의 존재를 가정하고 있다.

(4) 알고리즘의 표현은 유한개수의 기본적인 처리의 지령과 그 지령간의 실행순서의 규정으로 이룬다. 기본적인 처리의 지령(기본지령)은 그 처리에 붙여진 명칭과 같은 것으로 그 지령에 대해서 실행자는 무조건, 반사적으로 그 지령이 의미하는 처리를 실행하는 것으로 가정되고 있다.

(5) 알고리즘의 실행자에 의한 기본지령의 실행결과는 각 대상에 대해 명확히 정해져 있고 또 기본지령간의 실행순서의 규

정도 명확한 것으로 실행하는 알고리즘의 표현으로 규정된대로 기계적인 반응을 하는 것으로 그 알고리즘을 수행할 수가 있다. 그 수행효과는 동일의 처리 대상에 대해 항상 일정한 것이다.

(6) 알고리즘의 표현은 각 기본지령의 실행과 기본 지령간의 실행 순서에 대한 규정의 명확한 이해가 인간의 일상의 지적 능력에 있어 적당한 훈련을 거치면 가능한 것이 되지 않으면 안된다.

(7) 제2항에서 언급한대로 미리 정해진 범위내의 임의의 처리대상에 대해서 알고리즘의 실행은 유한회수의 기본지령의 실행인에 종료하여 어떤 결과를 얻는다.

이상을 알고리즘의 직관적 규정이라 할 수 있다.

3. 함수의 계산가능성

앞에서의 알고리즘이라는 개념을 정리해 보아도 이것에 수학적으로 일반적 정의를 내리기는 어렵다. 알고리즘을 명확히 하기 위해서는 어느 '표현'을 빌려야 가능하나 '표현'에 쓰여지는 언어, 기호, 쓰는 법등은 천차만별로 일율적인 형식적 규정이 어렵고 제6항의 규정도 수학적으로 표현하기는 거의 불가능하다. 그러나 알고리즘의 명확한 정의 없이 특정의 문제군에 관해 그것을 푸는 알고리즘이 존재한다 혹은 존재하지 않는다는 등의 증명이 가능한 것은 '함수의 계산가능성'이라는 개념으로 환원하여 증명할 수 있다[3].

여기서 임의의 알고리즘 A에 대해 그 처리대상을 x로 하여 A의 실행자가 A를 실제로 실행하여 결과 y를 얻는다면, x를 A에 대하여 정지성의 입력이라 하고 그렇지 못할 경우는 비정지성의 입력이라 하며, y를 입력 x에 대하여 A의 출력이라 부른다.

임의의 집합 D의 각 원소에 대하여 어느 집합 D의 원소를 1대 1로 대응시키는 대응 f를 D상의 함수라 하고 f=D→D'로 표시한다. 그렇게 하면 알고리즘 A는 A에 대한 입력의 모든 집합을 DA로 하면 DA의 각원소 x에 대해 x가 정지성의 입력인 경우에 한해서 출력 y를 대응시키는 DA상의 함수 f를 정할 수 있다. 이 f를 A로 계산할 수 있는 함수라 한다. A가 정지성의 조건을 만족하는 알고리즘인 경우, 'A로 계산할 수 있는 함수'라 한다.

또 어떤 알고리즘으로 계산할 수 있는 함수를 '계산가능'이라 부르기도 한다. 여기서 어느 문제군를 푸는 알고리즘 A가 존재한다고 하자. A는 각 문제 P에 대해 P의 해답에 대응시키는 함수를 계산하는 셈이 된다. 다시말해서 '주어진 문제군을 푸는 알고리즘은 존재한다'라는 물음은 '주어진 함수가 계산가능 한가'라는 물음과 같은 의미를 갖는다. 다만 이경우, 대상이 되는 함수에 대해서는 그 정의역과 치역에 대해 일종의 제한이 있다. 그것은 임의의 알고리즘 A에 대해서 처리대상의 집합 D의 각 원소는 정해진 유한개수의 기호이어야 하고 각원소를 식별할 수 있는 유한적 배치로 표현되어야 한다.

이와 같이 알고리즘이라는 것을 '계산가능한 수론적 함수'로 바꾸어 정의할 수 있음은 후술하는 Gödel의 부호화 (Gödel Numbering)에 의해서이다.

4. Gödel의 부호화[4]

자연수 이외의 일반적인 기호열의 처리문제는 어떻게 다루워지는가에 대해서 논한다. 논리적 문제등, 수 이외의 기호처리의 문제도 컴퓨터로 풀 수 있다. 실제로 기호를 처리하는 경우에는 예를들어 ASCII 코드에 의한 정보의 외부표현을 컴퓨터 내부에서는 모두 이진기호열(binary symbol string)로 변환하여 데이터로서 취급하고 있다. 이와 같이 부호화의 수단에 의해 수 이외의 정보라도 기호열로 표현할 수 있는 정보라면 자연수로 번역하여 다룰 수 있다. 대상이 되는 기호열 전체는 기껏해야 가산유한개수이므로 기호열을 적당히 자연수에 대응시키는 부호화에 따라 이산적 정보의 기계적 처리문제를 구체적으로 자연수 상의 함수의 계산문제로 바꿀 수 있다.

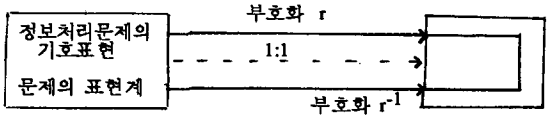

그림1 : 부호화

이 자연수 가운데로의 방법은 K. Godel에 의해 1931년의 불완전성 정리 [4]의 증명 가운데서 도입된 이래, 산술화 혹은 Gödel numbering이라 불리우고 있다. 또 부호화에 의해서 기호열에 대응하는 자연수를 Gödel수라 부른다.

따라서 일반적인 기호열의 처리문제를 생각할 경우에는 그 문제의 데이터라든가 알고리즘등을 기호로 표현하는 형식적인 표현계(소위 언어)로 문제를 생각할 수도 있고, 그 처리계의 표현계를 자연수의 체계 가운데로 부호화하여 생각할 수도 있는 것이다. 그결과 그처리계는 자연수 위에 정의된 자연수치를 추구하는 함수의 계산문제로 환원되어 정의될 수 있다.

이 Gödel의 부호화에 따라 문제의 대상영역은 크게 넓어지고 귀납적 함수(recursive function)은 강력한 개념이 되어 우리가 알고리즘이 존재한다고 '생각하는' 문제는 귀납적인 문제, 다시말해서 부호화에 의해 귀납적인 함수가 되는 문제가 된다. 귀납적 함수에 의해 계산이라든가 결정문제의 결정법 등 알고리즘의 존재를 정확히 고려할 수 있다. 또 그것을 푸는 알고리즘이 존재하지 않은 문제는 현행 컴퓨터의 능력의 원리적인 한계를 가르킨다.

5. Church의 제창

수론적 함수 f(x1, · · · · , xn)가 계산가능하다 함은 (x1, x2, · · · , xn)에 대해 f(x1, · · · · , xn)의 값이 구체적으로 주어지는 방법으로 f의 정의가 내려질 수 있음을 말한다. 다시 말해서 함수에 있어서 '계산 가능성' 이란 일종의 '정의 가능성'이라 말할 수 있다. 자연수의 공리에서 볼 수 있듯이 우리가 모든 자연수 상에 어떤 구체적인 대응을 지운다 할 때, 귀납법을 끄는 것과 같이 '귀납적 정의 가능성'의 개념에 명확한 정의를 부

여하여 그와 같은 수론적 함수를 '귀납적 함수'라 이름지웠다. 한편 A. Church는 '계산'이라는 수리논리 체계에 있어서 수론적 함수의 '정의 가능성'이라는 개념을 도입하여 계산가능이라 생각되는 구체적 수론적 함수가 모두 '정의 가능'임을 확인 시킴과 동시에 '정의 가능성'과 Godel의 '귀납적 가능성'이 동치임을 증명했다[3]. 여기서 Church가 [계산가능한 수론적 함수란 귀납적 함수를 뜻한다]라는 취지를 제창함에 따라 이것을 [Church의 제창]이라 불리우게 되었다.

그런데 A. M Turing도 이것과 독립적으로 알고리즘이 있는 표준적인 문제를 정하여 알고리즘 A와 B에 대해 'A가 B로 귀착된다' 함은 A와 B가 동일의 함수를 계산하는 것(A와 B가 동치라고도 한다)임을 밝히고 '투링머쉰'(Turing Machine)'이라는 모델을 고안, 투링머쉰으로 계산 가능한 수론적 함수라 함은 '귀납적 함수'임을 증명했다. 그 후 E. Post[5], A. MarKov[6]등 다수의 수학자가 각각 독립적으로 알고리즘의 정식화를 연구하여 그들의 정식화가 모두 투링머쉰과 동치임을 증명했다. 이와 같은 배경에서 수학적으로 증명할 수는 없으나 '실증적으로 증명된 사실'의 것으로 직관적인 의미에서 알고리즘을 갖는 함수를 귀납적 함수로서 등일시 하자는 Church에 의한 제창이 행해졌던 것이다.

6. 오토마톤

오토마톤(automaton)이란 자동기계를 의미하나 옛날에는 주로 자동인형을 가리키는 말이었다. 그러나 정보과학(information science)에서는 입출력을 가진 시스템이라는 술어로 쓰이고 있고 인간이라든가 컴퓨터의 모델로서 기본적인 개념의 하나가 되어 있다.

6.1 S. C Kleene의 분석

유한상태 오토마톤(finite state automaton)을 처음으로 정식화한 것은 S. C. Kleene[7]이다. 이 유한상태 오토마토의 정의를 위해 우선 미리 외부로 부터 기호라는 것이 주어진다. 기호라 함은 미정의의 개념, 혹은 근본개념(primitive concept)이며 임의의 대상이라도 좋다.

이와 같은 기호의 유한집합을 알파베트(alphabet)라 한다. 알파베트을 하나 정했을때, 이 알파베트에서 중복을 허락하는 기호를 꺼내 그것을 한줄로 늘어세운 기호의 유한집합을 알파베트상의 문(sentence), 열(string), 혹은 어휘(word)라고 한다. 0개의 기호열을 공문 혹은 공열(empty string)이라 하고 λ라고 쓰기도 한다. 알파베트의 A상의 모든 어휘의 집합을 A*로 나타내고, A*에서 공문을 집합을 A^+라 쓴다.

$$A^+=A^*-\{\lambda\}$$

또는 알파베트 A상의 어휘에 대한 임의의 집합 L을 형식언어 (formal language) 혹은 언어(language)라고 한다.

$$L \subset A^*$$

다시 유한상태 오토마톤을 수학적으로 정의하면

M=(K, Σ, P, δ, ω, F)로서

(1) K : 상태의 유한집합

(2) Σ : 입력 알파베트 (입력 기호의 유한집합)

(3) P : 출력 알파베트 (출력 기호의 유한집합)

(4) $q_0 \in K$: 초기상태

(5) F⊆K : 종지상태의 집합

(6) $\delta : Kx \to 2^K$; 천이함수 (혹은 $\delta : Kx\Sigma \to K$)

(7) $\omega : K \to P$: 출력함수

로 정의된다. 만일 출력을 생각하지 않을 때에는 오토마톤 M과 P와 ω를 뺀 5 tuple M=(k, Σ, δ, q, F)로 정의 된다. 천이함수와 출력함수는 귀납적으로 각각 Σ에서의 유한길이의 기호열의 집합, 다시말해서 Σ상의 기호의 유한열의 집합 Σ*과 출력기호열의 집합 P*로 확정할 수 있다. 이와 같은 오토마톤에서 천이함수 δ에 따라 상태천이를 행하여 종지상태 F에 도달할 수 있다면 M은 이 기호열을 인식한다고 말한다. M이 인식하는 Σ*의 기호열 전체의 집합 E를 M로 표현가능한 사상(event), 혹은 형식언어론에서는 M이 인식하는 언어 L(M)이라 부른다.

6.2 Kleene의 정리

어느 사상이 표현가능하기 위해서는 다시 말해서 언어가 인식되기 위한 필요충분조건으로서 S. C. Kleene는 다음과 같은 의미로 사상이 정규(regular)인 것을 정리로서 증명했다. 정규성(regularity)는 정규표현이 나타내는 기호열의 집합으로 표현된다.

[Kleene의 정리] [8] 유한상태의 오토마톤에 의해 인식되는 집합은 정규집합(regular set)이며, 그반대도 성립한다.

6.3 발전

이미 지적한대로 오토마톤이란 원래 생체의 기능을 표현하는 모델로서 제안된 것이나 전자회로의 설계이론으로 발전되어 다음과 같은 문제가 연구되었다.

(가) 주어진 오토마톤의 해석 :

(ㄱ) 주어진 오토마톤이 어느 특정의 초기상태에서 출발하여 특정의 출력을 얻기 위한 입력열의 집합을 정규표현으로 기술할 것.

(ㄴ) 주어진 두개의 오토마톤이 '같은 입력열에 대해서는 같은 출력열을 생성한다'라는 의미에서 등가(equivalent)인가 어떤가의 판정

(ㄷ) 주어진 오토마톤이 보다 간단한 오토마톤을 이은 것으로 분해 가능한가 어떤가의 판정.

(나) 오토마톤의 구성 :

(ㄱ) 지정된 기능을 갖추면서 가능한한 간단한 오토마톤을 구성할 것.

(ㄴ) 주어진 오토마톤을 지정된 구성요소에 의해 구성할 것.

(ㄷ) 지정된 구성요소에서 그것의 중첩구성도 임의의 오토마톤으로 구성할 수 있는가 없는가의 판정 - 이것은 완전성 (Complenteness)의 문제라 불리운다.

6.4. 변모

투링머쉰과 오토마톤은 기본적으로 그 동작원리를 같이하는 것으로 1936년 A. M Turing에 의해서 제안되었다. 투링머쉰은

계산기계의 모델로서가 아닌 인간의 지적사고의 한계를 찾기 위한 것이었다. 그는 인간의 계산 과정을 분석하여 유한상태 오토마톤에 한개의 입출력 테입기계(tape machine)를 고안하여 모든 계산은 이 테입기계로 실행할 수 있다는 계산기계의 이론을 전개했다.

7. 투링머쉰의 정의

7.1. 투링머쉰의 구성

투링머쉰은 양방향으로 무한이 늘어난 한개의 테입을 가지고 있다. 그 테입은 그림 1과 같이 구획으로 나누어져 있고, 각 구획에는 미리 정해진 테입 기호의 알파벳 S0, S1, · · · ·, Sn (S0는 공백이라하자) 의 하나를 기억할 수가 있다. 기계 본체는 유한 갯수의 내부상태 q0, q1, · · ·, qm을 가진 유한상태 오토마톤이고, 이 테입은 읽고(read), 쓰는(write), 헤드(head)를 가지고 있어 한번에 한개의 구획의 내용을 읽거나, 쓰거나 할 수 있다.

그림 1 : 투링머쉰

7.2. 투링머쉰의 동작

투링머쉰이 동작을 개시할 때, 내부상태는 미리 주어진 초기 상태에 있다. 초기상태를 q0라 하고, 동작의 개시시점을 헤드가 위치하는 테입 위의 구획에서 기호를 읽어 들인다.

주어진 시점에서의 기계의 동작은 그 때의 내부상태 qo와 헤드가 읽어 들인 구획내의 기호 Sj, 에 의해 일율적으로 정해져 다음의 (1) - (3) 의 어느 동작을 행한다 :

(1) 지금 머리가 보고있는 구획의 기호 Sj를 지우고 새로운 기호 Sk로 다시 쓴다.

(2) 한 구획만 오른쪽으로 머리를 움직인다.

(3) 한 구획만 왼쪽으로 머리를 움직인다.

그리하여, 본체의 내부상태도 새로운 상태 qr로 천이하여다음의 동작의 준비를 한다. 기계의 동작은 다음과 같은 유한갯수의 4항 계열에 따라 완전히 규정된다 :

	상 태	기 호	다음의 형태
(ㄱ)	q1	Sj	qr
(ㄴ)	q1	Sj	qr
(ㄷ)	q1	Sj	qr

이 4항의 유한계열을 투링머쉰이라고 한다. 각 시점에 있어서 기계의 동작은 그때의 내부상태 q1와 읽어들인 기호 Sj에 의해 결정되므로 이와 같은 투링 머쉰를 결정성 투링 머쉰 (deterministic turing machine)이라 한다.

기계의 동작 가운데에서 다음의 상태 qr 과 읽은 기호 Sj의 한쌍(qr, Sj)가 정하는 4항이 투링머쉰에 존재하지 않을 때, 투링머쉰은 동작을 정지한다. 이와 같은 상태 qr을 종지상태 (혹은 수지상태)라 한다. 그림 2는 한 인간이 흑판에 쓰여진 문제를 계산하는 과정을 모방하는 형태를 나타내고 그림 3은 투링머쉰의 본체에 대한 개념도이다.

8. 세종기계의 개념

「기계」라는 말의 의미에는 여러가지가 있다. 기계공학에서 쓰여지고 있는 「기계」의 좁은 의미에서 컴퓨터 과학, 공학에서 기계의 의미까지 여러가지가 있다. 이미 지적한대로 투링기계도 인간의 계산절차를 모방시키는 것을 목적으로 고안된 추상기계이다. 여기서 중요한 개념은 어느 기계의 동작이 투링기계를 써서 설명될 때, 우리는 "이 기계는 투링 기계이다"라고 말할 수 있다는 점이다.

우리가 일상 쓰고 있는 컴퓨터도 정보처리기계, 곧 계산기계(computing machine)이다. 그것은 하드웨어와 소프트웨어로 구성되나 하드웨어는 넓은 의미로 계산과정의 물리적 실현이며, 소프트웨어는 알고리즘 기술에 지나지 않는다. 따라서 정보처리기계의 목적이라든가 기능은 실현된 계산기계에서 구해지는 것이 아니라 "정보를 처리한다" 혹은 "계산한다"라는 추상적인 과정에서 구해지는 것이다. 과학적으로 이 분석을 행하여 명확한 수학적 모델을 얻고나서 그 물리적 실현이 논해진다. 정보처리란 소위 기호열에 의해 표현된 이산적 정보를 유한적 조작에 따라 어느 이산적 정보로 교환하는 것이다. 따라서 정보처리의 대상이 되는 것은 어떤 형태로든 형식화된 것이 아니면 안되고 또 형식화된 대상과 그것에 첨가된 조작에 대해 '의미'를 부여해야 하는 것이다.

전통적으로 문자란 말의 단위를 일정의 약속 밑에서 치환하여 기록하는 기호로 정의된다. 그러나 컴퓨터 시대, 정보화 시대의 발상에서 다시 정의하면 그것은 고정된 말과 그것을 사용하는 이용자를 잇는 인터페이스로서 인간·기계계의 인테페이스(man-machine interface) 혹은 휴먼 인테페이스(human interface)로서 정보처리 시스템의 의미를 갖는다.

원래 훈민정음의 창제 목적이 우리말의 음성언어를 표기언어화 하는 기호처리계, 곧 표기 시스템(writiug system)의 건설에 있었다. 구체적으로 훈민정음은 음성 언어의 연속적 정보를 문자라는 이산적 정보로 교환하는 기능을 갖고 있다. 따라서 훈민정음을 정보처리기계로서 시스템적인 논의의 가능성은 충분하다.

여기서 우리는 훈민정음이라는 시스템을 보다 형식적으로 구체화하는 의미에서 세종머쉰(Sejong machine)이라 부르기로 한다.

8.1 세종머쉰의 정의

세종머쉰의 수학적 구조에 대한 논의는 이미 행한 바 있다 [9][10]. 여기서는 세종머쉰을 인식기계로서의 오토마톤, 생성기계로서의 정규문법(regular gnammer) [11], 대수적 표현으로서의 정규표현 (requlor expression)[11]으로 그 구성적 관계를 설명한다.

(가)정규문법에 의한 세종머쉰의 정의

훈민정음의 구성원리에 의한 문자의 생성개념을 다음의 4가

지 요소를 추상화여 표현할 수 있다.

$$G=(V, \Sigma, P. S)$$

일반적으로 이 G를 생성문법(generative gammer)혹은 문법이라고 한다. 또 P의 규칙에 따라 S를 바꾸어 쓰는 체계(rewritiug System)라는 의미에서 바꾸어 쓰기 시스템이라고도 한다. 여기서 V와 Σ는

(1) V = Σ ∪ {C, V, CH}

(2) Σ = {ㄱ, ㄷ, ㄷ, ㄹ, ㅁ, ㅂ, ㅅ, ㅇ, ㅈ, ㅊ, ㅋ, ㅌ, ㅍ, ㅎ, ㅏ, ㅑ, ㅓ, ㅕ, ㅗ, ㅛ, ㅜ, ㅠ, ㅡ, ㅣ}

(3) P는 다음의 규칙으로 이루는 집합이다.

(ㄱ) CH → C. V

(ㄴ) CH → C. V. C

(ㄷ) C → ㄱ | ㄴ | ㄷ | ㄹ | ㅁ | ㅂ | ㅅ | ㅇ | ㅈ | ㅊ | ㅋ | ㅌ | ㅍ | ㅎ

(ㄹ) V → ㅏ | ㅑ | ㅓ | ㅕ | ㅗ | ㅛ | ㅜ | ㅠ | ㅡ | ㅣ

(4) S

Σ*의 기호열 α로 S → α가 되는 것의 집합은 다시쓰기시스템 G로 생성된 언어 L(G)이며 세종머쉰이 생성하는 출력이다.

(나) 오토마튼에 의한 세종머쉰의 정의

정규문법 G=(V, Σ, P, S)가 주어졌을 때, 그 G가생성하는 언어 L(G)를 인식하는 오토마튼 M를 세종머쉰으로 정의할 수 있고 그 상태천이도는 그림 3과 같다. 이것은 세종머쉰의 조작적 의미론에 해당하며 현재 알려진 인식기계중에서 가장 경제적인 모델이다[9.10].

(다) 정규표면에 의한 세종머쉰의 정의

G= (V, Σ, P, S)에 있어서 생성규칙 P는:

$$CH \rightarrow CV$$

$$CH \rightarrow CVC$$

이다. 한글 문자의 전체집합은 "집합 CV와 CVC의 합집합이다" 와 같은 의미이므로 집합의 식으로서는 정규표면의 등식 :

$$CH \rightarrow CV \cup CVC$$

를 얻는다. 따라서 정규표현의 귀납적 정의에 따라 CH가 정규표현이라면 CH^+도 정규표현이므로

$$CH^+=(CV \cup CVC)^+$$

의 정규표현을 얻어 정의할 수 있다.

이상의 세종머쉰의 정의간에는 서로 구성적 등가성 관계가 성립한다. 다시 말해서 오토마튼에서 정규문법을, 정규문법에서 오토마튼을, 정규문법에서 정규표현을 각각 얻을 수 있다. 그 반대도 가능하다.

8.3 세종머쉰과 튜링 머쉰의 등가성에 관한 증명

어느 기계간의 등가성(equivalent)의 관계에 관한 문제에는 크게 구조등가관계(structural equiualent)와 동작등가관계(behavior equivalent)가 있다. 이미 지적한대로 어느 기계의 동작을 튜링 머쉰의 동작원리에 따라 모방할 수 있을 때 그 기계는 튜링 머쉰이라고 말할 수 있음을 지적했다.

Kleene의 표준형 정리[7]에서 보는 바와 같이 'Turing 계산가능함수와 귀납적 함수의 등가성' 혹은 'Turing 계산가능함수와 귀납적 부분함수의 등가성'에 관한 증명 방법론을 답습하면 다음과 같은 명제는 증명할 수 있다.

[정리] 세종 머쉰과 Turing 머쉰은 등가이다.

<증명방법> Kleene의 표준형 정리에서는 Turing 계산가능 함수 전체를 크래스 T라 하고 귀납적 부분함수 전체 크래스 R라 하여 여기서 확인하고자 하는 명제는

$$T = R$$

이다. 따라서 2명제을 T ⊃ R과 T ⊂ R로 나누어 증명한다.

따라서 여기서 세종 계산가능 함수를 S라 하면 새로운 명제

$$S = R$$

이 증명대상이 되고 S ⊂R과 S⊃R를 밝히면 그 명제는 증명이 된다. 구체적인 증명방법에 대해서는 참고문헌을 참고바란다[12].

9. 전망

훈민정음을 하나의 정보처리 시스템의 기계론에서 보았을 때 그것은 투링의 계산기계와 원리적으로 같은 계산론의 구성과 동작원리를 갖고 있음은 증명가능하다. 그 의미는 서기 1443년, 투링 머쉰보다 500여년전에 그것과 계산원리를 같이 하는 계산기계가 실제로 설계되어 운영되어 왔다는 사실은 과학사적 측면의 새로운 발견을 뜻하며 컴퓨터의 역사는 재조명되어야 한다. 참고로 투링 머쉰이 개념이 발표되자 가능하게 작는 투링의 만능계산기(uniuersal machine)를 구체적으로 만들자는 경쟁이 일어난 적이 있다[13]. 작다라고하는 척도로서는 테입기호의 수와 상태수와의 곱이 쓰여진다. 경쟁의 역사를 표로한 것이 표 1이다.

표 1 소형 만능계산기를 만든 경쟁사

작 자	년	기호수	상태수	척 도
Takahasi	1958	6	12	72
Ikeya	1958	6	10	60
Watanabe	1960	6	8	48
Minsky	1960	6	7	42
Watanabe	1961	5	8	40
Minsky	1961	6	6	36
Minsky	1961	4	7	28
Goto	1967	3	7	27
Sejong	1443	28	5	140

<참고 문헌>

[1] 정희성, 한국형 사용자 인터페이스 설계를 위한 고찰, 전자통신, Vol. 42, No. 4, 1989.

[2] Turing, A. M., *On computable numbers, with an application to The* Entscheidunqs problem, Proc. London Math. Society, ser. 2, Vol. 42, Vol. 43, 1936

[3] Church, A., An Unsolvable Problem of Elem ntary Number Theory, American Jounual of Mathematics, Vol. 58, 1936.

[4] Davis, M. ed., The Undecidable, basic papers on undecidable

propositions, unsolvable problems and computable functions, Raven Pre., 1965.

[5] Post, E. L., Finite combinatory processes-formulation I. The journal of symbolic logic. Vol. 1, 1936.

[6] Markov, A. A., The Theory of algorithms, NSF, 1954.

[7] Kleene, S. C., Representation of Events in Nerve Nets and Finite Automata,

[8] Kleene, S. C., General Recursive Functions of Natural Numbers, Math. Ann., Vol. 112, 1936.

[9] 정희성, 한글문자의 구조와 구성원리에 대한 과학적 고찰, 전자통신, Vol. 39, No. 4, 1989.

[10] 정희성, 훈민정음의 창제원리와 그 과학이론, 한국정보과학회, 논문지, 투고(심사중)

[11] Aho, A, V., Hopcroft, J. E., and Ullman, J. D., The Design and Analysis of Computer Algorithm, Addison-Wesley, 1974.

[12] Chung, H., About The Equivalence Between The Sejong Machine and Turing Machine, Information Science, 투고중(1989)

[13] Yamada, s., 정보처리의 과학, 일본공립출판, PP. 126, 1973.

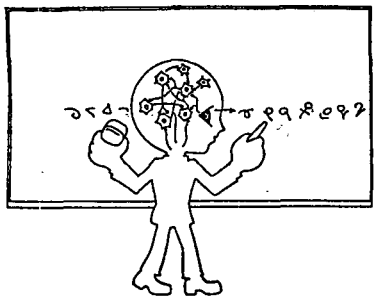

図 2 チューリング機械で模倣される人間の計算過程

図 3 チューリング機械の本体の概念図