한국어정보처리를 위한 최적한 한글 코드에 관한 연구

*변정용
*동국대학교 자연과학대학 전자계산학과

An Optimal Hangul Code System For The Korean Language Processing

*Byun, Jeongyong
*Dept. of Computer Science, Dongguk University at Kyungjoo

요 약

컴퓨터에 의한 한글정보처리의 주체는 한글이며, 객체는 그 처리도구인 컴퓨터라는 전제하에서 한글문자의 개별성에 의한 주체적 파악을 통하여, 한국어정보처리에 최적한 코드로의 개선안을 제안한다. 개선안의 구체적 대상 범주로서 최근의 한국어 정보처리의 응용분야인 자연언어처리, 문자인식, 음성인식 및 합성, 전자출판등이 점차 확대되어 가고 있다는 관점에서 보아서 기존의 코드가 가지고 있는 문제점을 분석하고 이들에 최적한 코드는 무엇이며 어떠한 성격을 가져야 하며, 그들이 기존의 코드가 중요시 하던 처리효율이나 저장 효율의 문제에 어떠한 영향을 미치는지에 대하여 해당 알고리즘을 개발하고 이들에 대한 평가를 해보인다.

I. 서론

컴퓨터에 의한 한글정보처리의 주체는 한글이며, 객체는 도구인 컴퓨터이다. 따라서 객체는 처리의 편의성을 위하여 주체에 물리적 또는 논리적으로 어떠한 제약을 제공해서는 안될 것이다[2]. 이러한 전제 아래서 컴퓨터 처리를 위하여 한글문자는 적어도 정보처리 대상의 본성을 유지하고, 또한 컴퓨터에 의한 처리의 적합성과 효율성을 갖는 코드가 요구된다.

한글코드는 사용 목적에 따라서 그 구조와 성격을 달리하였는 데, 이제까지 한글정보처리에 있어서 응용분야의 대부분은 자료처리였으며 코드의 표준화는 이에 준하여 이루어졌다고 본다. 다른 하나의 관점은 자소와 글자(음절)중 어느 것을 코드의 대상으로 정할 것인지를 선택하는 것이었다. 그리고 정보교환의 국제적인 추세에 상응할 방법이 모색되어 왔다.

본 논문은 한글정보처리의 응용분야가 자연언어처리, 문자인식, 음성인식 및 합성, 전자출판등으로 점차 확대되어 가고 있다는 관점에서 보아 앞으로 이들에 적합한 코드는 무엇이며 어떠한 성격을 가져야 하며, 그들이 기존의 코드가 중요시 하던 처리효율이나 저장 효율의 문제에 어떠한 영향을 미치는지에 대하여 해당 알고리즘의 개발을 통하여 보이고자 한다.

II. 한글문자와 코드 표준화

1. 한글코드의 고찰

한글문자에 대한 코드 표준화의 과정은 1974년에 정보교환용부호 KS C 5601-1974의 제정을 필두로 하여, KS C 5601-1982 (조합형), KS C 5601-1987 (완성형)등의 개정과정[1][10]이 있었다.

구 분	코드 종류
한글 코드	KS C 5601-1974(낱자형) KS C 5601-1982(조합형)
한자 코드	KS C 5714-1979(JIS C 6226)
한글.한자 코드	KS C 5619-1982(완성형) KS C 5601-1987(완성형)
코드 확장법	KS C 5620-1977(ISO R646)

표 1. 표준 코드의 분류표

구 분	코드번호	표현방법
낱자형	KS C 5601-1974	단수바이트
조합형	KS C 5601-1982	복수바이트
완성형	KS C 5601-1987	복수바이트

표 2. 글자구성 및 표현방법

그외에 한자코드 제정등이 있었다. 이들을 코드 표준화 대상에 관하여 분류하면 표 1.과 같고, 글자의 구성 방법에 따라서 또는 표현 방법에 따른 분류는 표 2.와 같다.

2. 한글코드의 분석

한글문자의 코드 표준화는 한글에 대한 관점의 변화에 따라서 여러가지 형태로 나타났다. 하나는 한글정보처리에 있어서 관점을 부분적인 처리의 효율성 또는 전체적인 보편성을 추구함이었고, 다른 하나는 한글문자의 개별성에 의하여 주체적 파악 또는 객체인 도구의 편의성에 의거한 의존적 파악이었다.

현행의 표준 코드는 부분적 처리의 효율성과 객체의 편의성에 관점을 두었으며, 그것은 컴퓨터가 한글정보처리에 있어서 행하는 역할이 주로 자료처리였다는 근시적인 판단에 근거하였음이라고 본다. 거시적 관점에서 본다면 전술한 내용에 따라서 훈민정음의 정신에 충실하고 한국어에 대한 원만한 표기체계로써의 기능을 가지며, 한국어 정보처리에서 요구되는 언어현상의 설명을 위한 표현방법등이 가능한 코드가 요구된다.

결론적으로 현행 코드는 처리의 효율성에 주안점을 둠으로써 언어처리나 문자구성에 있어서 제약을 제공하였고, 결과로서 객체의 편의성을 위하여 주체가 제약을 받는 주객전도의 현상이 나타났다. 그 개선 방향은 일차적으로 주체의 원만한 처리가 보장되고, 다음 객체의 처리 효율이 고려되어야 할 것이다.

III. 응용분야와 코드 개선

1. 새 응용분야의 요구

한글정보처리에 관한 새로운 응용분야인 자연언어처리와 음성인식 및 합성등과 같은 분야는 언어현상의 설명을 위한 언어정보를 요구하며, 이들과 함께 전자출판이나 여러가지 정보검색등의 응용분야에서는 글자의 구성등에 관한 자소정보를 요구한다.

2. 완성형 코드의 적합성 평가

전술한 바와 같이 처리 효율에 주안점을 둔 완성형 KS C 5601-1987은 빈도수 0.001%이상의 한글 글자(음절) 2350자를 선정하고, 나머지 부분에 한자 4888자와 기타문자를 선정하여 복수 바이트로써 총 8836자를 수용할 수 있게 하였다[1][11][12]. 확대일로에 있는 새로운 응용분야에 대하여 완성형 코드의 문제의 검토하면 다음과 같다.

(1) 불완전한 정보
(2) 불완전한 문자집합
(3) 표준화의 역행
(4) 훈민정음 정신의 축소화
(5) 알고리즘의 혼란
(6) 한글폰트의 비대화
(7) 문자집합 확장성의 혼란
(8) 소프트웨어 호환성의 약화

3. 한글 코드의 요건

전술한 새로운 응용분야의 요구와 기존의 요구를 망라하며, 정보교환의 국제성등에 따르는 새로운 코드의 개선 방향에서 한글 코드의 요건을 설정하면 다음과 같다[2].

(1) 객체에 의한 주체의 제약불가
(2) 한글의 언어정보 및 자소정보의 유지
(3) 알고리즘의 단순화
(4) 정보교환 및 정보처리에서의 효율성
(5) 완전한 문자집합의 지원
(6) ISO 2022의 코드 확장법에 적합성
(7) 소프트웨어의 호환성

4. 코드의 개선방안

새로운 응용분야에 대응하며, 한글정보처리 전반에 적합한 한글코드의 개선안은 두가지로 나아갈 수 있다. 하나는 표준의 범위에서 가능한 방법의 모색이며, 하나는 표준의 범위를 벗어나서 보다 근본적인 방법의 모색이다. 전자는 KS C 5601-1974를 기본으로 하여 KS C 5601-1982와 KS C 5601-1987의 부속서에 규정되어 있는 51자 낱자형 코드이며, 후자는 51자 코드에서도 자소정보의 일부가 상실되어 있다는 점에서 한글 자판용으로 정의된 33자를 내부 표현 코드로 하는 안이다. 33자 코드와 51자 코드의 정의는 다음과 같다.

(정의 1) 한글 오토마톤이 입력계에서 인식한 글자 중에서 복모음과 복자음을 단일 코드로 변환한 모음 11자와 자음 7자를 포함한 전체 집합을 한글 낱자형 51자 코드 또는 51자 코드라 한다.

(정의 2) 훈민정음의 28자 가운데 현행의 24자에 초성 쌍자음 5자(ㄲ, ㄸ, ㅃ, ㅆ, ㅉ)와 복모음 4자(ㅐ, ㅒ, ㅙ, ㅞ)를 합한 자음과 모음의 집합을 한글 낱자형 33자 코드 또는 33자 코드라 한다.

33자 코드에 대한 타당성은 51자 코드의 분석을 통한 문제점을 살펴보고, 이들이 갖는 문제의 해결관점에서 찾아본다.

(1) 복모음과 복자음의 단일 코드화
(2) 입력계의 복잡화 (단일코드화 알고리즘)
(3) 컴퓨터 내.외부의 자소정보 불일치
(4) 코드배열에서 모음의 자소정보 결핍
(5) 고어에서 쓰는 자소들의 미포함

(1)(2)(3)은 복자음과 복모음의 단일 코드화에 기인하는 것으로 단일 코드 표현의 지양에 의하여 해결가능하다. (4)(5)는 표 3.의 모음 특성과 고어의 자소를 코드에 포함되도록 배열하면 표 4.와 같이 된다. 고어문제는 자음 11자, 모음 16자의 추가 여지가 있다는 점으로 추후 연구로 미루고 본

논문에서는 제외하였다.

	세로꼴	가로꼴	세로꼴 가로.세로꼴
양성	ㅏ ㅑ	ㅗ ㅛ	ㅐ ㅒ 와 왜 외
음성	ㅓ ㅕ	ㅜ ㅠ	ㅔ ㅖ 워 웨 위
중성	ㅣ	ㅡ	의

표 3. 모음의 형태

51자 코드에서 상실한 정보는 복모음의 자소 정보와 모음꼴에 관한 정보이다. 전자는 언어현상 설명을 위한 언어정보로서, 예를 들면 '밟은'은 '발근'으로, '않다'는 '안타'로 소리가 나는 현상의 설명에 관련된다. 모음꼴 정보는 한글폰트의 합성에 필요한 정보를 제공해 준다.

	4열	5열	6열	7열
0000 0		ㅋ		
0001 1	ㄱ	ㅌ	ㅏ ㅐ ㅑ ㅒ	ㅗ ㅛ
0010 2	ㄲ	ㅍ		
0011 3	ㄴ	ㅎ		
0100 4	ㄷ			
0101 5	ㄸ			ㅜ ㅠ ㅡ
0110 6	ㄹ			
0111 7	ㅁ			
1000 8	ㅂ			
1001 9	ㅃ		ㅓ ㅔ ㅕ ㅖ	
1010 a	ㅅ			
1011 b	ㅆ			
1100 c	ㅇ			ㅣ
1101 d	ㅈ			
1110 e	ㅉ			
1111 f	ㅊ			

(가)

6열	7열
세로꼴 (양)	가로꼴 (양)
	(음)
(음)	(중) 가로
	세로

(나)

표 4. 33자 코드표

5. 내부 표현 방법

51자와 33자 코드의 내부 표현은 처리의 편의를 위하여 8비트계 코드 확장법에 따른다. 두 코드의 대상 문자의 집합은 ISO 2022의 코드 확장법[16]에 의거할 때 96보다 작으므로 단수 바이트 코드계에 속한다.

IV. 한글정보처리계의 평가

1. 입력계

51자 코드의 경우에는 한글 오토마톤이 인식한 글자에 대하여 복자음과 복모음을 단일 코드로 변환작업이 이루어지며, 33자의 경우는 단지 내부 코드로의 변환만 한다.

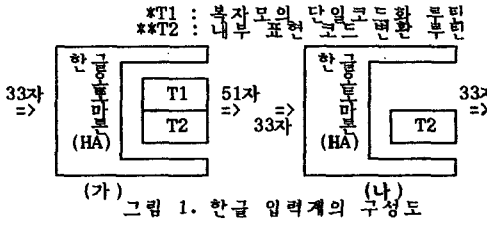

그림 1. 한글 입력계의 구성도

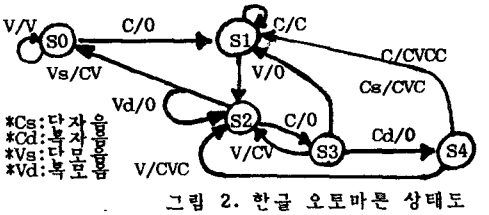

그림 2. 한글 오토마톤 상태도

33자 코드의 한글 글자 인식을 위한 한글 오토마톤은 밀리형[15]으로 나타내면 다섯 상태로 된다. 그림 2.는 기존의 오토마톤[6][7]을 구현과정에서 변형시킨 것이며, 계산량은 O(n)이다.

2. 출력계

출력계는 음절구성 문자열에서 받침 자음에 의하여 모음꼴이, 모음꼴에 의하여 자음꼴이 결정되면 합성에 의하여 글자의 모양을 구성한다. 표 3.를 표4.의 (가)와 같이 배열하고, (나)는 표 3.의 구역표시이며, (나)를 이용한 음절 합성과정은 다음과 같다.

(1) 받침의 유무에 따라 모음꼴 결정
(2) 5번 비트가 0이면 세로꼴(6열),
1이면 가로꼴(7열)
(3) 6열의 3번 비트가 0이면 양성, 1이면 음성
(4) 7열의 3,4번 비트가 0,0이면 가로꼴, 양성
(5) 7열의 3,4번 비트가 0,1이면 가로꼴, 음성
(6) 7열의 3,4번 비트가 1,0이면 가로꼴, 중성
(7) 7열의 3,4번 비트가 1,1이면 세로꼴, 중성

3. 편집계

가. 비교 알고리즘

51자 및 33자 코드에 대한 한글 문자열의 비교는 불일치 지점에서 해당 자소의 특성을 검사한다. 51자 코드에서 한글의 자소정보를 이용하면 다음과 같은 규칙이 존재함을 알 수 있다[3].

(1) 날자이면 서로 빤다.
(2) 모음-모음이면 빤다.
(3) 자음-모음이면 모음쪽이 작다.
(4) 자음-자음이면 다음 자소의 검사에서
자음-자음이면 배고, 자음-모음이면
모음쪽이 작다.

33자 코드는 복모음과 복자음이 존재하므로 다음의 규칙을 더하면 51자와 33자를 포괄하는 알고리즘이 된다.

(1) 모음과 자음이면 모음쪽이 작다.
예) S : ㅁ ㅏ ㄹ ㄱ ㅗ ㄱ ㅣ (말고기)
T : ㅁ ㅏ ㄹ ㄱ ㅇ ㅡ ㅁ (맑음)

(2) 자음과 자음이면 한 문자 내다 보아서 모음인 쪽이 작으며, 만약 자음이면 서로 빤다.
예) S : ㅂ ㅏ ㄹ ㅂ ㄷ ㅏ (밟다)
T : ㅂ ㅏ ㄹ ㅇ ㅡ ㅁ (발음)

(3) 복자음인 경우와 합께 고려한다. 즉 자음과 모음이면 한 문자의 뒤를 되돌아 보아서 모음과 모음이면 모음쪽이 크고, 자음과 자음이면 모음인 쪽이 작다.
예) S : ㄱ ㅗ ㅈ ㅏ ㅇ (고장)
T : ㄱ ㅗ ㅏ ㅎ ㅏ ㄱ (과학)

그림 3.는 51자 코드의 비교 알고리즘에 33자 코드에 대한 (1)(2)(3)의 규칙을 부가하여 알고리즘의 (3)에 추가하여 확장한 것이다.

```
HanStrcmp(S[i],T[i])
  (1) S[i] = T[i]이고, i = 0이면
            return(S[i] - T[i]);
  /* i>0 */
  (2) S[i] = T[i] = 모음이면
            return(S[i] - T[i]);
  (3) S[i] # T[i]이고,  /* 33자 코드 */
       S[i-1]=T[i-1]=자음이고,
            S[i] = 모음이면 return(음수);
            그렇치 않으면   return(양수);
       S[i-1]=T[i-1]=모음이고,
            S[i] = 모음이면 return(양수);
            그렇치 않으면   return(음수);
  /* S[i] = T[i] = 자음이면 */
  (4) S[i+1] = T[i+1] = 자음이거나
      S[i+1] = T[i+1] = 모음이면
            return(S[i] - T[i]);
  /* S[i+1] # T[i+1] */
  (5) S[i+1] = 모음이면 return(음수);
      그렇치 않으면,    return(양수);
```

그림 3. 33자 코드의 비교 알고리즘

나. 검색 알고리즘

검색의 경우는 51자와 마찬 가지로 영문의 알고리즘[13]에서 검색이 성공하면, 그 때 한글 특성에 관하여 검사한다. 그림 4.은 51자의 알고리즘에 33자의 특성을 반영시킨 부분이다.

(1) 일단의 검색이 성공이면 패턴의 음절이 모음절이면 텍스트 문자의 상대가 종성이면 성공, 초성이면 실패이다.

예) P : ㅅ ㅏ ㄹ ㅁ (삶)
S : ㅅ ㅏ ㄹ ㅁ ㅓ ㅅ ㅣ (살며시)

(2) 패턴의 문자가 텍스트 음절의 복자음의 일부와 같으면 실패로 본다. 즉 텍스트 문자열의 둘 앞을 내다 보아서 자음이면 복자음로써 실패이다.

예) P : ㅁ ㅏ ㄹ (말)
S : ㅁ ㅏ ㄹ ㅋ ㅇ ㅡ ㅁ (맑음)

```
검색 알고리즘 :
  (1) S[i]에서 일치일 때,
      S[i]=P[i]="한글"이면 [2]로 간다.
      그렇치 않으면 검색 성공!!!.
  (2) S[i]=자음일 때,
      S[i+1]=자음이고, S[i+2]=모음이면
            검색 성공!!!
      S[i+1]=자음이고, S[i+2]=자음이면
            검색 실패!!!
  (3) S[i]=모음일 때,
      S[i+1]=자음이면 검색 성공!!!
      S[i+1]=모음이면 검색 실패...
```

그림 4. 33자 코드의 검색 알고리즘

(3) 패턴의 문자가 텍스트 음절의 복모음의 일부와 같으면 실패로 본다. 패턴이 모음으로 끝나면 텍스트 한 문자 앞을 내다 보아서 모음이면 복모음로써 실패이다.

예) P : ㅅ ㅏ ㄱ ㅗ (사고)
S : ㅅ ㅏ ㄱ ㅗ ㅏ (사과)

4. 텍스트와 화면 커서의 사상

날자형 코드에서 일반적으로 문제삼는 것은 텍스트 커서 위치와 화면 커서간에 대응성을 갖지 못한다고 하는 점이다. 편집기의 명령중에서 커서의 좌우 및 상하이동에서 그예를 발견할 수 있다. 이경우에는 자소정보를 이용한 사상함수를 만들 수 있다.

가. 상하이동

일반적으로 편집기의 예를 들면 텍스트내에 여러가지 제어문자가 삽입되어 있어서 상하이동 명령의 수행은 이동 대상 라인의 텍스트 내용을 검사하는 사상함수에 한글 오토마톤을 중심으로 구성된 음절수 계산 루틴을 추가한다. 51자 코드는 이를 위한 오토마톤이 4개의 상태로 구성되는 이 사상함수는 라인의 문자열 포인터와 이동거리를 인수로 갖는다.

나. 좌우이동

매번 상하이동 사상함수에서 이동거리의 값이 +1 또는 -1인 경우이다. 그런데 -1인 경우는 알고리즘에 추가 규칙을 요구하므로 현재 화면커서 위치가 N이면 N-1을 값으로 준다.

5. 평가

가. 응용 분야의 적용 범위

KS C 5601-1987은 그 적용범위가 자료처리에 국한하여 처리의 효율에 주안점을 두었던 코드이므로 언어정보와 자소정보등을 요구하는 새로운 응용분야인 한글정보처리 분야에서 그 실용성은 상실한다. 그 개선안으로서 제안된 51자 코드는 대부분의 문제를 해결하지만 그 역시 복자음과 복모음에 관한 정보를 일부 상실하고 있으므로 33자 코드는 한글의 개별성에 대한 모든 표현이 가능하다는 점에 의하여 그 비중이 증가일로에 있는 새로운 응용분야의 코드로서 적합성이 높다고 본다.

나. 실행 효율

처리계를 입력계, 편집계, 출력계로 나누어서 볼 때, 입력계에서 한글 오토마톤은 입력수준의 철자오류의 검사와 내부표현 코드로 입력 문자를 변환하며, 51자는 그 과정에서 복모음과 복자음을 단일 코드화 한다. 출력계에서는 문자변환기에 의하여 하나의 글자를 합성한다.

편집계에서는 문자열의 비교 및 검색등이 이루어 지는데 그림 2.과 그림 3.에서 보는 바와 같이 51자 코드와 33자 코드의 알고리즘은 근소한

차이를 보이며 비교의 경우 33자 코드의 알고리즘은 51자 코드의 경우를 포괄한다. 또한 이들은 한글 오토마튼의 개입없이 자소정보의 이용만으로 해결이 가능하였다.

다. 저장 효율

33자 코드에 대하여 51자 코드, 완성형, 조합형등의 코드는 저장효율의 측면에서 자료축약의 의미를 가지므로 자료축약의 한 방법으로 볼 수 있다. 그것은 한글 한 음절을 표기하는데 몇 바이트가 필요한가하는 점에서 그렇다.

즉, 33자 코드 : 2 - 4바이트
51자 코드 : 2 - 3바이트
조합형 : 2바이트
완성형 : 2바이트

음절자소수	갯수	빈도수(%)
2	56,380	48.2
3	58,721	50.2
4	1,764	1.6
5	0	0.0
총음절계	116,865	100.0

표 5. 음절의 빈도수

한글 음절의 빈도수나 글자 종류의 분석[5]을 통하여 볼 때, 33자 코드로 하였을 때 한음절의 구성 자소수와 그들의 빈도수는 다음과 같으므로 디스크 효율은 표 5.에서 보는 바와 같은 정도이므로 51자와 33자 코드에 대한 저장 효율의 저하는 예상되나 이것은 한글 코드의 필연성에 기인한다는 점에서 문제로 볼 수 없다.

V. 결론

KS C 5601-1987의 그 대상과 용도에 관한 관점이 단순한 자료처리 및 그 처리의 효율에 두었다는 결론을 도출해내었으며, 그런 결과로 인하여 새로운 응용분야에서 요구되는 언어현상의 설명이나 글자의 구성에 관련된 문제에 부응할 수 없는 코드임이 입증되었다.

그런 결론에 대한 대안으로서 코드의 개선안을 두가지 측면에서 제안하였고 관련 알고리즘의 개발을 통하여 한글정보처리 일반에 있어서 적합성과 실효성이 있음을 보았다. 또한 새로운 컴퓨터의 응용분야에 대한 요구에 따름을 보았고 미래에 한글 정보처리의 전반에 걸친 코드 표준화의 시금석이 될 수 있을 것으로 본다.

앞으로 33자 각 자소에 대한 적합성 평가와 자판의 적정한 글자수에 관한 연구 및 한글 고어의 표현방법에 관한 연구가 요구된다.

참고문헌

[1] 박동순.이재현.최은만.강석,컴퓨터 한글코드의 사용과 표준화에 관한 연구, 정보과학회지, 제6권, 제1호, 1988

[2] 변정용, 한글문자의 지적처리에 관한 연구,한국전자통신연구소, 1989

[3] 변정용.이계영,7단위 한글낱자 부호의 문자열 검색 및 비교 알고리즘에 관한 연구, 동국대 경주캠퍼스 논문집 제7집,1988

[4] 변정용.이태경.이계영.허정식, 자연어 사서 구성에 관한 연구, 대한전자공학회, '89하계 학술대회 논문지,1989

[5] 유재원,우리말역순사전,정음사,1984

[6] 정희성, 한글정보의 지적처리, 동경대 박사논문, 1987

[7] 정희성, 한글문자의 구조원리에 대한 과학적 고찰, 전자통신, 제10권, 제4호, 1989

[8] 최현배, 한글갈, 1974

[9] 홍진수.최상현,한글부호계에 관한 연구, 한국정보과학회논문지, Vol. 13, No. 4, 1989

[10] 한국공업규격, KS C 5601-1974, KS C 5601-19872, KS C 5601-1987

[11] 요시히코 오노, UNIX의 일본어화의 실현방법, 정보처리학회지, Vol.27, No. 12, 1986

[12] 일본공업규격, JIS 6226, JIS 6228

[13] Boyer and Moore, "A Fast String Searching Algorithm",CACM, Vol. 20, 1977

[14] Chuck Card and Donald Prigge, "The world of standard", BYTE, PP130-142, 1983

[15] Peter J. Denning,Jack B.Dennis and Joseph E. Qualitz, Machine,Languages and Computation,PP92-97,Twoer Press,1980

[16] Jim Flemming and William Frezza, "NAPLPS:A New Standard for Text and Graphics", BYTE, PP203-254, 1983