한글 Common Lisp에서 한글 함수 기능

이창열, 오승준, 임영환
인공지능연구실, 한국전자통신연구소
대전시 대덕단지 사서함 8호 (우편번호 305-606)

요약

본 논문에서는 한글(한국어) ascii 코드와 4가지 한글 표현 원리에 만들어지는 한글 음절을 정의한다. Common Lisp(CL)의 확장된 버전으로 한글이 사용 가능한 한글 CL(HCL)의 소개하고 CL에 추가되는 새로운 한글 함수에 대하여 설명한다.

HCL의 모든 함수는 한글을 다루는 방법에 따라 4가지 타입으로 나뉘어진다. 1) 타입 0 - 한글을 입출력 값으로 취하지 않는 전형적인 CL 함수, 2) 타입 1 - 원래 CL 함수 정의의 변경없이 입력으로 한글을 받아들이는 함수, 3) 타입 2 - 한글을 사용하기 위하여 함수의 정의를 확장해야하는 CL 함수, 4) 타입 3 - 한글 처리를 하기 위하여 새로 설계한 새로운 함수.

위의 타입에 의해 분류되는 각 함수에 대한 정의를 제안하고 한글 편집기에 대하여 소개한다.

1. 서론

Common LISP(CL)[3]은 인공지능 응용프로그램을 작성하기 위하여 가장 보편적으로 사용되는 표준화된 LISP 언어 중의 하나이다.

본 고에서는 CL에 한글 기능을 추가한 HCL에 대하여 논술한다. 한글을 CL에 추가하는 기능은 다음의 이유에 의해 필요하다. 한글은 영어와 같은 자신의 알파벳 집합(한글 ascii 코드 집합)을 가지나 그 표기 원칙이 영어의 그것과 다르다. 영어에서 단어는 단순히 알파벳의 연속이나 한글에서 단어는 한글 ascii 코드에 4가지 한글 표현 원칙[1]을 적용하여 결합한 집합의 시퀀스이다. 이러한 차이로 인하여 한글 자료는 CL에서 효과적으로 다루어지지 않는다.

HCL은 근본적으로 한글이 처리될 수 있도록 새로운 함수를 가진 CL의 확장된 버전이다. HCL은 Emacs[4]와 유사한 입력 편집기와 4가지 타입의 시스템 함수로 나뉘어진다;

타입 0 : 한글을 입력이나 출력으로 가지지 않는 전형적인 CL 함수.
타입 1 : 원래 함수 정의의 변화 없이 입력으로 한글을 받아들이는 함수.
타입 2 : 한글을 받아들이기 위하여 함수의 정의를 확장해야하는 CL 함수.
타입 3 : 한글 처리를 위하여 설계한 새로운 함수.

2장에서는 한글 표기 시스템에 대하여, 3장에서는 4가지 타입에 의하여 재 정의된 함수와 새로운 한글 함수에 대하여 그리고 4장에서는 한글 입력 편집기에 대하여 설명한다.

2. 한글 표기 시스템

2.1. 한글 ascii 코드

터미널이나 디스크에서 한글 코드는 일반적으로 7 비트 ascii 코드에 의해 표현된다. 비록 영어 코드 뿐만 아니라 한글 코드가 이 ascii 코드에서 7비트로 표현되지만 한글 스트링의 시작에는 SI(0X0F), 한글 스트링 끝에는 SO(0X0E)가 존재한다. 이 7비트 코드는 한글 오토마톤 아래서 SI/SO 코드를 제거하고 ascii 코드의 MSB 앞에 1 비트를 추가함으로서 8비트 코드로 변환된다. 8비트로 변환된 코드는 주 기억장치에 저장되어 처리된다. 표 1은 한글 ascii 코드를 나타낸다.

표 1. 한글 ascii 코드

하/상	0	1	2	3	4	5	6	7
0	nul	del	sp	0	@	P	'	p
1	soh	dc1	!	1	ㄱ A	ㅁ Q	a	q
2	stx	dc2	"	2	ㄲ B	ㅂ R	ㅏ b	ㅛ r
3	etx	dc3	#	3	C	ㅃ S	ㅐ c	ㅜ s
4	eot	dc4	$	4	ㄴ D	T	ㅑ d	t
5	enq	nak	%	5	E	ㅅ U	ㅒ e	u
6	ack	syn	&	6	F	ㅆ V	ㅓ f	v
7	bel	etb	'	7	ㄷ G	ㅇ W	ㅔ g	ㅝ w
8	bs	can	(	8	ㄸ H	ㅈ X	h	x
9	ht	em	)	9	ㄹ I	ㅉ Y	i	y
a	lf	sub	*	:	J	ㅊ Z	ㅕ j	— z
b	vt	esc	+	;	K	ㅋ [	ㅖ k	{
c	ff	fs	,	<	L	ㅌ \	ㅗ l	] |
d	cr	gs	-	=	M	ㅍ]	m	}
e	so	rs	.	>	N	ㅎ ^	n	~
f	si	us	/	?	O		o	del

2.2. 한글 표현 원리

한글의 음절은 다음 4가지 원리[1]에 의하여 정의된다; 1) 결합 구조의 원리는 한글 자소를 결합(합성)하여야만 의미가 있게한다. 즉 자음(2.3.장에서 정의) + 모음(2.3.장에서 정의), 그리고 자음 + 모음 + 자음 처럼 자소가 결합한다. 한글 자소는 순서관계(<) 하에서 선형순서의 관계로 정의된다. 2) 위상 구조의 원리는 한글 음소의 연속을 2차원 위상공간에 사상시키는 6개의 기본 구조로 표현한다. 예를들어 "가", "고", "과", "국", "각", 그리고 "곽"이 각 구조에 대한 사용예이다. 3) 대수 구조의 원리는 한글 음절을 자음 × 모음 또는 자음 × 모음 × 자음의 결합 구조로 나타낸다. 즉 H : 자음 × 모음 → 음절, H : 자음 × 모음 × 자음 → 음절로 나타낸다. 여기서 H는 한글 ascii 코드를 음절로 사상시키는 함수이다. 4) 분출의 원리는 위의 3가지 원칙을 적용하여 만들 수 있는 모든 음절의 부분 집합을 현재 사용중인 음절로(예를 들어 "뒀"과 같은 문자는 위의 3가지 원리를 적용하여 만들어 질수 있으나 현재 사용하는 문자는 아니다.) 정의한다. 초기 3가지 원칙은 Bourbak의 수학 이론[2]에 기반을 가진다.

예를 들어 "한"은 3개의 음소 : " ㅎ ", " ㅏ ", 그리고 " ㄴ "을 위의 3가지 원칙에 적용하여 나타난 음절로서 "한"은 현재 사용중인 문자 집합에 속한다. 그러므로 우리나라 사람은 "한"을 의미를 가질수 있는 음절로 정의한다. 한글의 원리중 4번째 원리를 설명하면 1, 2, 3 원리를 사용하여 나온 모든 문자 중에서 퐷"과 같은 문자가 나올수 있다. 그러나 이 문자는 현재 사용되고 있지 않는 문자이다. 그러므로 이러한 문자 집합은 4번째 공리에 의해서 제거시킨다. 사실 비록 "한"이 1음절 이나 CL 함수는 ascii 코드에 기반을 두고 수행함으로 CL 함수 **length**는 사용하는 한글 코드에 따라 다르게 출력할 수 있으나 위의 원칙에 적용하면 "3"을 결과로 출력한다. 그렇지만 한글 함수 **han-length**는 음절 단위의 연산을 수행함으로 1을 결과로 출력한다. 또한 한글 음절의 순서관계는 한글 사서식 순서(Lexicographic Order)에 따른다.

2.3. 한글 자료

CL에 한글을 포함하는 자료 타입이 이 장에서 설명된다.

한글 문자 : 새로운 33개의 한글 코드가 HCL에서 정의된다: 19개는 자음 14개는 모음. 여기에 쓰여지는 순서가 한글 코드의 크기 증가 순서이다.

자음 :
#\ㄱ #\ㄴ #\ㄷ #\ㄹ #\ㅁ #\ㅂ #\ㅅ #\ㅇ #\ㅈ #\ㅊ
#\ㅋ #\ㅌ #\ㅍ #\ㅎ #\ㄲ #\ㄸ #\ㅃ #\ㅆ #\ㅉ

모음 :
#\ㅏ #\ㅑ #\ㅓ #\ㅕ #\ㅗ #\ㅛ #\ㅜ
#\ㅠ #\ㅡ #\ㅣ #\ㅐ #\ㅒ #\ㅔ #\ㅖ

한글 스트링 : 한글 문자의 연속이다. 예를 들어 "대한민국"은 한글 코드의 연속이다. 한글 스트링은 음절 형태를 가질수도 있고 아닐수도 있다.

한글 심볼 : 한글 심볼은 리스트, 시퀀스, 그리고 어레이 타입을 가지는 곳에서 사용된다.

```
>(setq color '(붉은색 푸른색 노란색))
(붉은색 푸른색 노란색)
```

한글 리드테이블 : 한글 33개의 문자를 CL 리드테이블에 추가하고 한글의 구문을 constituent로 한다.

3. 한글 Common LISP에서 함수

한글은 HCL에서 문자, 스트링, 그리고 스트림 같은 자료 타입에 사용된다. HCL에서 함수는 4개의 타입으로 분류된다. 타입 0에는 수치처리 함수로 +, -, *, / 그리고 **subtype** 등이 존재한다. **Car, equal** 그리고 **quote**는 전형적인 타입 1이다. **Length** 와 **char<**는 타입 2이다. **Han-length, han-replace**, 그리고 **han-atom**은 타입 3함수이다. 타입 2와 타입 3에서 함수는 문자, 시퀀스, 스트링, 그리고 입출력과 관계된 함수를 다룬다.

이 장에서는 타입 2와 타입 3 함수가 자세히 설명된다. 타입 2 함수의 이름은 CL[3]에 설명되어 있으므로 단순히 함수의 의미만 다루고 타입 3 함수는 음절 단위의 처리에 기반을 두었다. 그러나 HCL은 한글을 음절 단위 뿐만 아니라 음소 단위의 연산도

가능하다.

CL의 표준화 스펙은 [3]에 명령어 형식, 기능, 사용예가 잘 기술되어 있다. 그러므로 여기서는 각 명령어의 사용예나 기능에 대하여 설명은 간략히 한다. 특히 새로 추가되는 한글 함수(han- 으로 시작하는 함수)는 기존 함수(한글 함수에 han-을 제거한 함수)와 비교하여 명령어 형식이 거의 차이가 없다. 예를 들어 기존 함수 length에 대하여 han-length는 한글 함수로서 명령어 형식이 같다.

3.1. 진위 검사 함수(Predicate)

CL의 진위검사 함수 31개는 모두 타입 0 혹은 타입 1 이며 한글을 위하여 추가되는 진위검사 함수에는 다음과 같은 것이 있고 이들은 타입 3 이다.

o **han-atom** *object* : object가 순수 한글 문자로 구성된 아톰인가 검사후 사실이면 T, 그렇지 않으면 NIL을 돌려준다.

o **han-string** *object* : object가 순수 한글 문자로 구성된 스트링인가 검사후 사실이면 T, 그렇지 않으면 NIL을 돌려준다.

3.2. 문자

CL의 문자 조작 함수는 36개로서 이중 21개는 타입 1, 15개는 타입 2 함수이다. 그리고 한글만을 위한 타입 3 함수도 9개 정의하였다.

<u>타입 2 함수</u>

o **standard-char-p, graphic-char-p, string-char-p, alpha-char-p, alphanumericp, char>, char>=, char<, char<=, char-code, char-bit, char-font, code-char, char-int, int-char**

한글 문자에 대한 정의는 code, bit, font만 정의하면 된다. bit와 font의 특성은 0이고 code는 한글 ascii 코드표(표 1.)에 등록된 값이다.

<u>타입 3 함수</u>

o **han-char-p** *char*
o **han-consonant-char-p** *char*
o **han-vowel-char-p** *char*

여기에 있는 함수 3개는 한글 문자의 기본 집합에 속하는가에 대한 비교이다. 한글 코드를 완성형으로 구현시 함수 **han-consonant-char-p, han-vowel-char-p**는 한글 함수에서 제거되어야 한다.

o **han-char=** *han-1-string &rest han-1-strings*
o **han-char/=** *han-1-string &rest han-1-strings*
o **han-char>** *han-1-string &rest han-1-strings*
o **han-char>=** *han-1-string &rest han-1-strings*
o **han-char<** *han-1-string &rest han-1-strings*
o **han-char<=** *han-1-string &rest han-1-strings*

위의 함수는 각각 한글 한 음절의 크기를 비교하는 함수로서 이들의 크기는 한글 음절의 사전 배열 순서이다(한글 사서식 순서).

3.3. 한글 시퀀스 조작 함수

CL의 시퀀스 조작 함수는 37개로 이중 5개는 타입 1, 32개는 한글 문자가 포함된 시퀀스를 사용할 때 그 의미를 분명하게 재 정의하여야 하는 타입 2 함수이다. 추가적으로 타입 3을 위하여 13개 함수를 정의 하였다. CL의 시퀀스 조작 함수 중 32개는 한글 스트링으로 정의된 시퀀스를 받을때 문제가 생긴다. 이때 생기는 문제는 한글 스트링 조작의 기본 단위 즉 인덱싱이 어떻게 되느냐이다. 여기서 구현한 HCL에서는 이 기본 단위를 앞에서 정의한 한글 문자 단위로 한다. 이는 한글 스트링에서 한글이 음절 단위가 아니라 음소 단위로 이루어 짐을 뜻한다.

타입 2 함수

o **elt, subseq, length, reverse, nreverse, fill, replace, remove, remove-if, remove-if-not, delete, delete-if, delete-if-not, remove-duplication, delete-duplication, substitute, substitute-if, substitute-if-not, find, find-if, find-if-not, position, position-if, position-if-not, count, count-if, count-if-not, mismatch, search, sort, stable-sort, merge**

```
>(subseq "한글" 3 5)
 "그"

> (length "이순신")
8
```

상기 함수는 분류하고 합치며 찾는 기능을 가진 함수로서 구현한 한글 코드에 따라서 결과가 다르다. 여기서는 한글에 대하여 자소 단위로 처리한다.

타입 3 함수

o **han-subseq** *han-string start &optional end* : 한글의 음절을 한 단위로 인덱스한 Subsequence를 돌려준다.

```
>(han-subseq "인공 지능" 2 4)
 "지능"
```

o **han-length** *han-string* : 한글의 음절을 한 단위로 계산한 길이 값을 돌려준다.

```
>(han-length "한글")
 2
```

o **han-reverse** *han-string* : 한글의 음절을 한 단위로 계산한 역 순서 스트링 값을 돌려준다.

```
>(han-reverse "인공지능")
 "능지공인"
```

o **han-remove** *han-1-string han-string &key :from-end :test :test-not :start :end :count :key* : 한글의 음절을 한 단위로 계산한 소거로서 주어진 한 음절이 제거된 스트링을 돌려준다.

>(han-remove "나" "가나다라" :test #'han-char>)
"가다라"

o **han-delete** *han-1-string han-string &key :from-end :test :test-not start :end :count :key*

han-remove의 Destructive 함수이다.

o **han-delete-duplicates** *han-string &key :from-end :test :test-not :start :end :count :key* : 한글의 한 단위로 중복된 음절을 제거한다.

o **han-substitute** *han-1-new-string han-1-old-string han-string &key :from-end :test :test-not :start :end :count* : 한글 스트링의 주어진 한 음절을 다른 한 음절로 대치한다.

>(han-substitute "지" "저" "인공저능")
"인공지능"

o **han-find** *han-1-new-string han-string &key :from-end :test :test-not :start :end :key* : 한글 스트링의 한 음절을 찾는 함수로 첫번째 찾아진 한 음절을 돌려준다.

>(han-find "나" "가나다라")
"나"

o **han-position** *han-1-string han-string position &key :from-end :test :test-not :start :end :key* : 한글의 음절을 한 단위로 계산한 위치 값을 돌려준다. 제공되는 문자가 없으며 NIL을 돌려준다(각 명령어에서 설명이 안된것은 CLTL 책[3]을 참조하면 된다).

>(han-position "다" "가나다라")
2

o **han-count** *han-1-string han-string count &key :from-end :test :test-not :start :end :key* : 한 음절을 한 단위로 계산한 값을 돌려준다.

>(han-count "다" "가나다라")
1

o **han-search** *han-string1 han-string2 &key :from-end :test :test-not :key :start1 :start2 :end1 :end2* : *han-string2*에 *han-string1*이 포함되어 있는지를 검사하여 첫번째 발견된 위치를 돌려준다.

>(han-search "간다" "나는 간다.")

3

o **han-sort** *han-string predicate &key :key* : han-string을 음절 단위로 분류한다.

>(han-sort "라다나가")
"가나다라"

o **han-merge** *han-string1 han-string2 predicate &key :key* : 두개의 han-string을 음절 단위로 합친다.

>(han-merge "라다나가" "가나다라")
"라가다나나다가라"

3.4. 한글 스트링 조작 함수

CL의 스트링 조작 함수는 21개로 그중 15개는 타입 1, 6개는 타입 2 함수이다. 한글을 위하여 새로이 정의되는 함수는 8개가 있다.

타입 2 함수

o **char, schar, string<, string<=, string>, string>=**

위 함수는 스트링의 순서를 알려주는 함수인데 한글일 경우는 한글 문자 코드가 연속이라고 생각하고 그 크기를 비교한다. 즉 이 함수는 한글의 사전식 배열을 알기 위해서는 사용될 수 없다.

타입 3 함수

o **han-char** *han-string index* : 음절 단위로 주어진 위치의 음절을 돌려준다.

o **han-string=** *han-string1 han-string2 &key :start1 :end1 :start2 :end2*
o **han-string/=** *han-string1 han-string2 &key :start1 :end1 :start2 :end2*
o **han-string<** *han-string1 han-string2 &key :start1 :end1 :start2 :end2*
o **han-string<=** *han-string1 han-string2 &key :start1 :end1 :start2 :end2*
o **han-string>** *han-string1 han-string2 &key :start1 :end1 :start2 :end2*
o **han-string>=** *han-string1 han-string2 &key :start1 :end1 :start2 :end2*

음절 단위로 한글 사전 순서로 비교가 정의된다.

o **han-string-trim** *han-1-string-list han-string* : *han-string*의 시작과 끝 지점에서 일치되는 *han-1-string-list*를 제거한 부분 스트링을 돌려준다.

3.5. 한글 입출력 함수

o **han-format** *destination control-string &rest arguments* : 기존 Format은 한글이 들어갈 때 정확히 작동되지 않을 수도 있다. 이를 위하여 새로 **han-format**을

정의하였다.

4. 한글 입력 편집기

CL 인터프리터에서 프로그램을 입력할 경우 시스템에서 기본적으로 제공하는 라인 편집 기능을 이용한다. 그러나 한글 프로그램을 입력할 때는 이 기능이 적절하게 동작되지 않기 때문에 추가적으로 라인 편집기능을 제공해야 한다.

o. 현재 커서 위치에서 그줄 끝까지 지운다.(CTL + k)(ConTroL 키를 누른 상태에서 키보드 k를 친다.)
o. 이전 줄의 내용을 가져온다.(CTL + p)
o. 다음 줄의 내용을 가져온다.(CTL + l)
o. 최근에 입력한 한 음절을 제거한다.(Del)
o. 현 커서 위치에서 좌로 커서를 이동시킨다.(CTL + h,CTL + b)
o. 현 커서 위치에서 우로 커서를 이동시킨다.(CTL + f)
o. 현 라인의 마지막 음절로 커서를 이동시킨다.(CTL + e)
o. 현 커서 위치의 오른쪽 문자를 지운다.(CTL + d)
o. 현 라인을 모두 지운다.(CTL + u)
o. 현 커서의 왼쪽 한 단어를 지운다.(CTL + w)

편집기능을 제공하는 제어키로 제공한 위의 기능을 기존의 편집기 Emacs[4]에서 제공하는 제어 키 조합을 참조하였다. 위와 같은 기능을 제공하기 위하여 입력 편집기는 편집용 버퍼, 편집된 버퍼, 한글 오토마톤, 편집 명령어 해석 함수, 각 편집 명령어 수행 함수가 필요하다.

5. 결 론

HCL은 타입 0, 타입 1, 타입 2, 타입 3, 그리도 입력 편집기 명령어로 구성되었다. HCL은 KCL[5]을 사용하여 구현하였으며 SUN의 BSD UNIX와 80386 프로세서를 사용하는 System V UNIX에서 운용하고 있다. 본 논문에서는 CL에 한글 기능을 제공하는 시스템으로 4가지 타입 분류에 의하여 분류된 기존 함수의 정의 확장과 새로운 한글 함수를 HCL의 기능으로 제안한다. 기존 함수 정의를 확장한 것은 부분적으로 한글 코드를 사용하는 방법에 따라 의미의 형태가 다를 수 있으나 새로운 한글 함수(타입 3)는 한글을 구현한 코드와 무관하게 정의될 수 있는 함수이다.

참고 문헌

1. 정희성, 박사학위논문 : "Korean Language Information Processing", University of Tokyo, 1986.
2. Bourbaki, N., "Lachitecture des mathematiques : la mathematique ou les mathematiques", Paris, A. Blanchard, 1962.
3. Guyl.Steele JR., Common LISP The Language, Digital Press, 1984.
4. Richard Stallman, GNU Emacs Manual, Fifth Edition, Version 18, Free Software Foundation, 1986.
5. Taiichi Yuasa and Masami Hagiya, Kyoto Common LISP Report, Kyoto University, 1986.