The Asian Journal of TEX, Volume 1, No. 2, October 2007
Article revision 2007/10/31

한글 라텍에서 조사 이형태 교체의 자동화에 관하여

On the Automation of 'JOSA' Allomorph Alterations in Hangul LATEX

김강수 Kangsoo Kim
한글텍사용자그룹 karnes@ktug.or.kr

KEYWORDS Josa, allomorph alteration, Hangul LATEX, koTEX, HLATEX, hLATEXp

ABSTRACT This article presents issues regarding the automation of Josa allomorph alterations in Hangul LATEX systems. Josa is one of the peculiar grammatical elements in Korean, which attaches to nouns to mark their cases or functions (subject, object, etc.) in sentences. In many cases, a single Josa has variants (or allomorphs) that are chosen according to the phonological condition of the preceding noun. In a LATEX document, the phonological feature of a noun cannot be often pre-determined. Therefore, automation of Josa allomorph alteration is required for a Hangul LATEX system. In this paper, we will review automatic Josa allomorph alteration methods in different Hangul LATEX systems, and identifies some related problems.

1 들어가는 말

우리말에는 '조사(助詞)'라는 독특한 문법 범주가 있다. 주로 체언 또는 부사나 연결어미에 결합하여 다른 말과의 문법적 관계를 나타내는 관계언으로서, 이견이 있기는 하나 학교문법에서는 품사(즉 단어)의 하나로 취급되는 의존형태소이다. [3] 조사의 존재는 우리말을 교착어로 분류하게 하는 중요한 특성 가운데 하나이다.

그런데 조사들 중에는 동일한 기능을 수행하는 이형태(異形態; allomorph) 혹은 변이형(variant)을 갖는 것들이 있다. 하나의 기본형(형태소)에 대한 이형태들은 상보적(배타적)으로 분포하며, 대부분의 경우 조사 앞에 선행하는 단어의 음성적 특성에 따라 알맞는 이형태가 선택된다.[1] 예를 들어 주격조사 '–이/–가'는 의미가 완전히 동일하며 '–이'가 자음 아래에서, 그리고 '–가'가 모음 아래에서 분포하여 상보적 분포를 보이는 이형태인 것이다.

조사 자체를 라텍(LATEX)에서 특별히 취급해야 할 필요는 없다. 그러나 매크로로 치환된 문자열의 마지막 음소의 조건, 즉 자음인가 모음인가 또는 특정 음성이 오는가에 따라 라텍이 조사를 달리 선택해야 하는 경우가 있다. 또는 상호참조(cross-reference)와 같이

1. 이러한 이형태의 결정은 예외가 없는 필수적이고 자동적인 교체이다. 어떤 이형태들은 말하는 이의 의도에 따라 수의적이고 비자동적으로 교체되기도 한다.

나중에 결정되는 문자열을 조사가 이어받을 때 그 조사를 사전에 미리 결정하지 못하는 경우도 있다. 이들을 적절하게 처리하는 것은 한글 라텍이 반드시 가져야 하는 기능 중 하나이다.

이 글은 몇 가지 중요한 한글 라텍 매크로들이 이러한 문제를 어떻게 처리하여 왔는지 검토하고, 현재의 해결 방법에 대하여 알아본 후, 이에 연관된 문제들에 관하여 시론적인 제안을 하는 것을 목적으로 한다. 대상으로 하는 한글 매크로 패키지는 hLaTeXp, HLaTeX 과 koTeX 이다.

2 문제의 소재

2.1 조사 결정의 자동화가 필요한 상황

조사를 사전에, 즉 원본을 입력할 때 결정할 수 없는 대표적인 경우는 상호참조이다. 다음 인용문은 이러한 상황을 잘 설명하고 있다.

> 라텍이 제공하는 기능 중의 하나로, `ref`, `pageref` 명령 등을 사용하면 자동으로 생성된 장이나 절 번호를 참조할 수 있습니다. 그런데, 이들 명령을 사용하여 문서를 작성하다 보면 자동으로 표시될 번호를 미리 알 수 없기 때문에 바로 다음에 이어질 조사를 쓰는 데에 어려움이 따릅니다.
>
> **문제 `\ref{sampref}`를 보면** 과 같은 예를 생각해봅시다. 사용자가 문서를 입력할 때 `\ref{sampref}`라는 참조 명령이 생성할 숫자를 미리 알 수 없습니다. 따라서 조사 "을"을 사용해야 할지 또는 "를"을 사용해야 할지 알 수가 없습니다. [6, 6–7쪽]

이러한 상황을 편의상 '참조형 조사 자동처리'라고 부르기로 하자.[2] 참조형 조사 자동처리는 상호참조의 경우뿐 아니라 문헌인용(citation)의 경우에도 역시 필요하다.

이와 다른 경우로서, 매크로로 치환된 문자열을 생각할 수 있다. 예컨대, `\mycar`라는 매크로로 **자동차**라는 문자열을 치환하기 위하여

```
\newcommand\mycar{자동차}
```

라고 정의한 경우를 생각하자. 우선 "`\mycar`는"과 같은 코딩은 별다른 문제가 없는 것처럼 보인다. 그러나, 나중에 원본을 수정하면서, "자동차"라는 문자열 대신 "내 차 모닝"이라고 이 정의를 바꾸면 위와 같은 입력이 "내 차 모닝는"으로 치환되기 때문에 잘못된 결과를 얻는다. 이 위치에서의 보조사 "–은/–는"은 치환되는 문자열의 음성적 상황에 맞게 라텍이 선택할 필요가 있다. 이러한 상황을 편의상 '비참조형 조사 자동처리'라고 부르기로 한다. [1, 23쪽]

2. 이것을 "자동조사(처리)"라고 부르는 것은 hLaTeXp 이래의 관행이다. HLaTeX에서는 0.99 버전까지 "자동조사"라는 용어를 사용하다가, 그 후 조사의 이형태뿐 아니라 매개모음 및 서술격 조사 어간 탈락까지 매크로로 구현하면서 "자동 음운 변화 처리"라고 불렀다. [4, 29쪽] 이 글에서는 이 두 용어를 구별없이 사용하기로 한다.

표 1. 우리말 조사 규칙

앞단어 끝소리	와/과	을/를	이/가	은/는	(이)라	(으)로
리을(ㄹ) ㄹ 아닌 종성	과	을	이	은	이라	로 으로
중성	와	를	가	는	라	로

2.2 조사 교체의 규칙

우리가 문제삼는 조사 이형태 교체는 (다행히) 자동적 교체이며 음운론적으로 조건된 교체이다. 따라서 조사가 출현하는 위치의 음운적 조건이 명확하면 형태는 자동으로 결정된다. 표 1은 몇 가지 중요한 조사들의 형태 교체를 요약한 것이다. 이 표에서 예시한 조사들은 모두 앞 단어 끝소리라는 음운론적 조건에 의해 형태가 결정된다.

이 가운데, '–라/–이라'의 경우는 특별히 언급할 것이 있다. 이것은 그 자체 하나의 조사라기보다, 유일하게 활용하는 조사인 서술격 조사 '–이(다)"의 활용형 중 하나일 뿐이다.[3] 서술격 조사의 어간은 직전 음운이 모음일 경우 탈락 또는 생략이 허용되는 것으로 보이나, 이것은 강제되는 조건이 아니므로 반드시 '–이'를 생략해야 하는 것은 아니다. 그러나 활용 어미가 '–라'이거나 '–라고', '–라서' 등의 경우에는 어간의 생략형이 더 널리 사용되는 것이 일반적이다. 즉,

이것은 녹차이다. / 이것은 녹차다.

위의 문장은 둘 다 허용되며 앞의 것이 좀더 문어적 느낌을 주는 정도이지만,

* 이것은 녹차이라고 할 수 있다. / 이것은 녹차라고 할 수 있다.

이 예에서는 전자가 거의 쓰이지 않는다.

3 해결 방안의 모색

3.1 hLaTeXp의 자동조사 구현

조사 이형태 선택의 자동화 문제를 처음으로 구현한 것은 hLaTeXp [6]의 `hangul.sty` 패키지였다. 이 패키지는 참조형 조사, 그 중에서도 숫자 형태의 참조 문자열에 대해 조사를 붙일 때 그 조사의 형태를 자동으로 선택하는 코드를 포함하고 있다. 문서가 남아 있지 않아 자세한 사정을 알 수는 없으나 이 때 직면했던 문제는 다음과 같았을 것이다.

① '완성형' 한글 코드의 자소 분리가 매우 어려웠다.[4]

3. '–이(다)'가 과연 조사인지에 대해서 견해가 일치되어 있지 않은 것 같다. 은광희의 [4]에서는 '지정사'라는 용어를 사용하고 있는데 이 글에서는 학교문법을 따라 이를 조사로 간주한다. 국어학적 문제는 이 글의 주제와 무관하다.

4. hLaTeXp는 완성형 한글을 처리하는 패키지였다.

② 라텍으로 이미 식자된 또는 처리된 이전 문자열의 코드값을 역추적하는 것이 어려웠다.

만약 한글 코드의 자소 분리가 가능하였다면 조사 처리는 한결 쉬웠을지도 모른다. 종성 코드에 자음 또는 모음에 해당하는 표지를 넣는 것이 그다지 수고롭지 않은 일이 되었을 수도 있고 이를 역추적하는 데도 큰 노력이 필요하지 않았을 터이다. 그러나 완성형(KSC5601; 후의 KS X 1001)을 표준으로 채택한 결과 직전 문자에서 종성 코드만을 분리해내는 것이 거의 불가능해졌으며, 만약 말음 표지를 붙이려 한다면 수천 자 또는 수만 자에 해당하는 모든 문자별로 각각 처리해야 했을 것이다. 한글 코드에 붙여넣은 말음 표지 매크로가 다른 라텍 매크로와 충돌할 가능성까지 염두에 둔다면 아주 어려운 일이었음을 짐작할 수 있다.

그래서 hLaTeXp에서 선택한 방법은, 대략 다음과 같은 절차를 거치는 것이었다.[5]

① 장절명령 또는 라벨 명령이 불리는 시점에 현재의 `\@currentlabel` 값(숫자)을 `\k@fsa`라는 명령의 인자로 넣어두고 이것을 `.aux` 파일에 기록한다.

② `\ref` 명령이 불리는 시점에서 `\k@fsa`의 인자로 넘어오는 숫자를 계산하여, 1의 자리 숫자에 대해 그 값이 0, 1, 3, 6, 7, 8이면 `\k@fson` 플래그를 활성화하고, 그렇지 않은 경우에는 이 값을 `\k@fsoff`로 한다. 즉, 마지막 음성이 자음인가 모음인가를 표시해두는 것이다.

③ 자동조사 명령 '`\가`'는 직전에 넘어온 `\k@fs*`를 받아서 그 값에 따라 '–이'나 '–가'를 선택한다. 즉, `\k@fson`이면 '이'를, 그렇지 않으면 '가'를 식자하도록 한다.

먼저 `hangul.sty`[6]는 필요한 조건들을 다음과 같이 선언하고 있다.

```
77 \newif\ifk@fs
78 \newif\ifk@rfs
79 \DeclareRobustCommand*{\k@fsoff}{\global\k@fsfalse}
80 \DeclareRobustCommand*{\k@fson}{\global\k@fstrue}
81 \DeclareRobustCommand*{\k@rfsoff}{\global\k@rfsfalse}
82 \DeclareRobustCommand*{\k@rfson}{\global\k@rfstrue}
```

`\k@fson`과 `\k@fsoff`는 단지 `\k@fs`라는 조건이 참인지 거짓인지만을 나타낸다. `\k@rfs` 조건은 ㄹ 종성을 처리하기 위한 것이다.

`\label` 명령을 재정의하여 라벨을 `.aux`에 쓸 때 `\k@fsa` 코드를 붙이도록 한다.

```
174 \ifauto@josa
175 \def\label#1{\@bsphack
176   \protected@write\@auxout{}%
177         {\string\newlabel{#1}{{\@currentlabel}
178           {\thepage\noexpand\k@fsa@org{\noexpand\number\noexpand\c@page}}}}%
179   \@esphack}
180 \fi
```

5. 이 스케치에서는 주격조사 '–이/–가'의 선택만을 예로 들었다.
6. 이 글에서 다루는 `hangul.sty`의 버전은 1998/11/20이다.

이러한 조작의 결과, 첫번째 LaTeX 실행시 .aux 파일에 예컨대 다음과 같은 행을 쓴다.

```
\newlabel{samplelabel}{{4\k@fsa {4}.2\k@fsa {2}} {10\k@fsa {10}}}
```

다음 실행시 이 행이 처리되는데, 이 때 \k@fsa 명령은 다음과 같이 정의되어 활성화 된다.

```
83 \DeclareRobustCommand*{\k@fsa}[1]{{\count0=#1\relax
84   \ifnum\count0<0 \count0=-\count0 \fi
85   \count1=\count0 \divide\count1 10 \multiply\count1 10 \advance\count0 -\count1
86   \ifcase\count0
87    \k@rfson \k@fson \or%0
88    \k@rfsoff\k@fson \or%1
89    \k@rfsoff\k@fsoff\or%2
90    \k@rfson \k@fson \or%3
91    \k@rfsoff\k@fsoff\or%4
92    \k@rfsoff\k@fsoff\or%5
93    \k@rfson \k@fson \or%6
94    \k@rfsoff\k@fson \or%7
95    \k@rfsoff\k@fson \or%8
96    \k@rfsoff\k@fsoff   %9
97    \else\k@fson\k@rfson \fi}}
```

84–85 행은 넘어온 인자(number)의 일의 자리 수를 구하는 것이다. 그런 다음 0–9의 각 경우에 대하여 종성 표지 조건들을 설정한다.[7]

이제 입력된 원본에서 '\ref{samplelabel}\가'라는 매크로를 만나면,[8]

```
287 \DeclareRobustCommand*{\^^b0}[1]{%
288   \ifx#1^^a1\@yi@ \else
289   \ifx#1^^fa\@gwa@\fi
290   \fi}
```

위의 정의에 의하여 \@yi@라는 매크로를 부르게 되는데, 이것은

```
275 \def\@yi@{\ifk@fs 이\else 가\fi}
```

로 정의되어 있다. 즉, \k@fson이면 '이'가 선택되는 것이므로 참조 문자열(숫자)의 일의 자리가 0, 1, 3, 6, 7, 8이면 주격조사로 '이'가 식자된다.

이런 방식으로 hLaTeXp에 다음과 같은 조사 자동처리 매크로가 정의되어 있다.

\이, \가, \은, \는, \을, \를, \과, \와, \로, \으로

hLaTeXp의 조사 이형태 선택의 자동 처리는 그 선구적인 의미가 크지만 아직은 본격적인 조사 자동 선택이라고 말하기 어려운 면이 있다.

① 자동조사가 \label, \ref에 한정되고 \cite에 대해서는 구현되지 못했다.

7. 0은 '영'이라고 읽고 10은 '십'이라고 읽으므로 서로 다른 끝소리이지만 ㄹ이 아닌 자음이라는 조건이 같으므로 별도로 처리하지 않고 일의 자리만으로 처리해도 상관없다.
8. '가'는 완성형 한글 코드 B0A1이다.

② 참조 문자열이 숫자, 영문자, 일부 한글 부호문자인 경우만 정의되어, 일반적인 문자열인 경우에 대한 대책이 없었다.

③ 참조가 아닌 상황의 자동 조사 문제가 해결되지 않았다.

④ `\label` 명령을 자체적으로 재정의하는 다른 라텍 패키지들과 함께 사용될 경우 충돌을 피할 수 없다.

⑤ pdf 제작을 위한 hyperref 패키지에서 문제를 일으켰다.

위의 ③, ⑤항은 후술하는 HLaTeX에서도 여전히 문제가 되었던 것이므로 이후 상술할 것이다.

3.2 HLaTeX의 자동조사 구현

HLaTeX의 조사 선택 알고리즘은 기본적으로 hLaTeXp를 계승하고 확장한 것이다.[9] 즉, 말음 표지 매크로를 미리 붙여두었다가 식자 시에 말음 표지 매크로의 인자를 검토하여 조사 이형태를 선택하는 아이디어를 여전히 사용하고 있다. 그러나 숫자에만 적용되던 말음 표지를 완성형 한글 문자까지 확장한 것과 `\cite`에 대하여 부분적이나마 자동조사가 동작하도록 한 것이 HLaTeX의 기여에 해당한다.

HLaTeX의 자동조사 매크로 중 '\가'는 `hangul.sty`[10]에서 다음과 같이 정의된다.

```
\def\^^b0{\@ifnextchar^^ed\only@gt\ga@or@gwa}
\def\ga@or@gwa#1{%
  \ifx#1^^a1\@josa{가}{0|}\else
  ...
  \fi
}
```

그렇다면 여기서 `\@josa`란 무엇인가. 이것은 아직 확정되지 않은 매크로로서, 상황에 따라 `\@firstoftwo`가 되거나 `\@secondoftwo`가 될 것이다. 이 상황을 제어하기 위한 매크로로서, `\@jung`, `\@jong`, `\ri@ul`이 마련되어 있다. 즉,

```
\DeclareRobustCommand*{\@jung}{\gdef\is@rieul{n}%
  \global\let\@josa\@firstoftwo\ignorespaces}
\DeclareRobustCommand*{\@jong}{\gdef\is@rieul{n}%
  \global\let\@josa\@secondoftwo\ignorespaces}
\DeclareRobustCommand*{\ri@ul}{\gdef\is@rieul{y}%
  \global\let\@josa\@secondoftwo\ignorespaces}
```

말하자면, `\@jung`인 경우 `\@josa`는 `\@firstoftwo`로 재정의되므로 중성(모음) 아래서는 `\@josa`의 두 인자 가운데 첫번째 것 '가'가 선택되는 것이다. 이들이 모두 robust 명령으로 정의된 이유는 인자들이 옮겨다니는 과정에서 풀리지 않게 하려는 것이다. [8, 847쪽]

9. HLaTeX 1.0.1 버전은 유니코드 UTF-8 입력을 받아들이는 Omega(Lambda) 패키지를 중심으로 구성되어 있다. 그러나 이 글에서 문제삼는 것은 `hangul.sty`을 중심으로 하는 라텍 패키지로서의 HLaTeX이다. Lambda에서 한글 자동조사 문제는 이와는 다른 방식으로 구현되었다고 할 수 있으나, 이에 대한 고찰은 이 글의 범위를 넘는다. 후일 다른 글에서 이 문제를 다룰 수 있기를 희망한다. 은광희의 같은 글 [4]를 참조.
10. 이 글에서 다루는 `hangul.sty`은 HLaTeX 1.0.1 버전에서 `euc-kr.tex` 버전 1.0을 부른다.

이 매크로들의 명칭이 의미하는 바는 자명하여, 앞 단어의 끝소리가 각각 중성(모음)인지 종성(자음)인지 아니면 ㄹ 자음인지를 나타내고 있다. 한편, 이러한 \@jung, \@jong 등은 \make@josa라는 매크로에 의하여 결정된다. 이 매크로는 후에 참조될 모든 참조자에 미리 붙여서 .aux 파일에 쓴다.

```
\newlabel{test}{{1\make@josa{1}}{1\make@josa{1}}}
\newlabel{test2}{{테스트\make@josa{테스트}}{1\make@josa{1}}}
```

그 후 이 참조자를 호출하면, 예컨대 숫자의 경우

```
\ifnum‘0=\k@second\@jong\else
\ifnum‘1=\k@second\ri@ul\else
\ifnum‘3=\k@second\@jong\else
\ifnum‘6=\k@second\@jong\else
\ifnum‘7=\k@second\ri@ul\else
\ifnum‘8=\k@second\ri@ul\else
....
```

와 같은 방식으로 \make@josa가 \@jung, \@jong, \ri@ul 가운데 하나로 바뀐다.

숫자나 영문자가 아닌 한글 또는 한글 기호문자의 경우에는 조금 특별한 방법을 이용한다. HLaTeX 배포 파일 중의 하나인 josa.tab에는 기호문자, 한자, 한글 중에서 끝소리가 모음이거나 ㄹ인 것을 미리 정의해두고 있다. hangul.sty이 이 파일을 읽어서 각 문자에 대하여 \@jung, \@jong 등의 값을 미리 설정해두는 것이다. 이를 통해 일반 문자의 경우까지도 자동 조사를 처리할 수 있게 되었다.

이제 남은 일은 참조자에 따라(숫자든 문자든) \make@josa를 붙여서 .aux에 쓰게 하는 것이다. 한글 이름(names), 예컨대 \indexname의 경우,

```
191 \def\ksnamedef#1#2{%
192 \AtBeginDocument{\@namedef{ks#1}{#2\protect\make@josa{#2}}
193                  \@namedef{#1}{\expandafter\protect\csname ks#1\endcsname}}}
194 \ksnamedef{indexname}{찾아보기}
```

와 같은 방식으로 정의하여 “찾아보기”라는 문자열 다음에 자동조사 명령을 쓸 수 있게 한다. \bibitem과 \label 명령도 \make@josa를 붙이도록 다음과 같이 정의한다.

```
856 \def\label#1{\@bsphack
857   \protected@write\@auxout{}%
858         {\string\newlabel{#1}%
859          {{\@currentlabel\string\make@josa{\@currentlabel}}%
860           {\thepage\string\make@josa{\thepage}}}}%
861   \@esphack}
```

HLaTeX은 hLaTeXp의 열 개의 자동조사 명령 이외에 ‘\|’와 ‘\—’라는 특별한 매크로 두 개를 더 두고 있다. 이것은 매개모음이나 서술격조사 어간의 탈락을 일반화하여 처리할 수 있게 하는 것이다. 즉, \|는 모음 아래서는 아무 글자도 찍지 않고 자음 아래서는 ‘이’를 선택하도록 하는 매크로이다.

3.3 새로운 문제점

HLaTeX의 조사 이형태 자동 선택 방식은 숫자가 아닌 문자까지 자동조사를 붙일 수 있다는 점 외에도 `\cite`에 자동조사를 처리할 수 있다는 장점을 가지고 있었다. 그러나, 일반적인 의미에서 다음과 같은 한계를 가졌다.

① Lambda와 달리 라텍 매크로로는 참조나 인용이 아닌 일반적 텍스트에 자동조사를 붙일 수 없었다. 즉, `\make@josa`가 미리 설정되어 있지 않은 문자만이 드러나는 상황에서는 자동조사가 작동하지 않는다. 이것은 위의 두 매크로 패키지에서 채택한 알고리즘이 가진 한계였다.

② 문헌 인용에 있어서 흔히 사용되는 cite나 natbib와 같은 패키지는 그 자체에서 `\bibitem`을 재정의한다. 이 재정의가 `hangul.sty`보다 나중에 온다면 자동조사 처리를 위한 정의가 무력화될 수 있다. 반대로 `hangul.sty`를 이들보다 나중에 로드하면 이번에는 natbib 등이 제대로 동작하지 않는다. 즉, 여러 패키지를 활용해야 하는 상황에서 `\make@josa`를 일일이 붙여주는 방식은 패키지 충돌의 가능성을 항상 안고 있다.

한편, 2000년대에 접어들면서 pdfLaTeX의 사용이 일반화되고 라텍으로 고품위 pdf를 제작[11]하는 문제가 제기되면서 주목받은 hyperref라는 패키지가 HLaTeX의 자동조사 처리 기능과 충돌한다는 것이 알려졌다. hyperref과 hangul 패키지를 동시에 로드하면, 모든 라벨과 참조자들에 붙은 `\make@josa`가 모두 풀려서 다른 패키지에서 이를 처리할 방법이 없게 되어 에러를 보이게 된 것이다. 이러한 현상은 ulem, endfloat와 같은 중요한 패키지에서도 나타났다. KTUG에서는 궁여지책으로 `hangul-nojosa`라는 제3의 패키지를 만들게 되는데 이것이 하는 일은 `hangul.sty` 내의 조사 관련 코드를 완전히 제거한 것이었다. 그 결과, 다른 패키지와의 충돌은 없어졌으나 이번에는 자동조사라는 한국어 문헌 조판에 매우 중요한 기능을 사용할 수 없게 되는 결과를 가져왔다. [2, 4쪽] HLaTeX 1.0.1에서 제공하는 khyper라는 패키지는 부분적으로 hyperref와의 충돌을 완화해 주지만 hyperref의 모든 기능을 사용할 수 있게 한 것은 아니었던 까닭에 여전히 문제를 안고 있다.

4 새로운 시도

4.1 hangul-k 패키지

2004년 5월 경, 김도현은 종래와 다른 새로운 자동 조사 처리 방식을 발표하였다. 이것은 어차피 `hangul.sty`에서는 사용할 수 없었던 hyperref이 상호 참조시에 만들어내는 참조 문자열을 역으로 추적하여 이 문자열 자체로부터 자동조사 결정에 필요한 값(jong/jung/rieul)을 알아내는 방식이었다. [1]

11. 주로 문제가 되었던 것은 pdf 한글 bookmark, pdf 내의 hyperlink, pdf 파일의 한글 텍스트 검색과 추출 등에 관련된 것이었다.

이 새로운 자동 조사 처리 루틴을 hangul-nojosa와 결합하여 hangul.sty를 대체하고자 한 것이 hangul-k.sty이다.

우선, hyperref 패키지의 \hyper@link라는 매크로를 다음과 같이 재정의한다.[12]

```
\let\hyper@link@ORIG\hyper@link
        \def\hyper@link#1#2#3{\hyper@link@ORIG{#1}{#2}{#3}%
          \begingroup\let\null\@empty\xdef\make@josa{#3}\endgroup
        }
```

여기서도 \make@josa라는 매크로가 사용되고 있지만, 이 매크로를 .aux 파일에 쓰지는 않는다. 즉, 하이러링크로 만들어지는 참조부에 대해서만 이 매크로가 실행되도록 한 것이다. \make@josa로 들어온 문자열을 분석하여 마지막 한 글자를 취한다.

```
\providecommand*{\josa@LastThreeChar}[1]{%
  \@ifemptyarg{#1}{0000}{\expandafter\@lastthreechar#1‘\^^M}}
\def\@lastthreechar#1#2‘\^^M{%
  \@ifemptyarg{#2}{000#1}{\@@lastthreechar#1#2‘\^^M}}
\def\@@lastthreechar#1#2#3‘\^^M{%
  \@ifemptyarg{#3}{00#1#2}{\@@@lastthreechar#1#2#3‘\^^M}}
\def\@@@lastthreechar#1#2#3#4‘\^^M{%
  \@ifemptyarg{#4}{0#1#2#3}{\@@@@lastthreechar#1#2#3#4‘\^^M}}
\def\@@@@lastthreechar#1#2#3#4#5‘\^^M{%
  \@ifemptyarg{#5}{#1#2#3#4}{\@@@@lastthreechar#2#3#4#5‘\^^M}}
```

\@josa라는 매크로는 두 개의 인자를 취하는데, 첫번째 인자를 \@jung, 두번째 인자를 \@jong 등으로 정의하여 둔다. 그런 다음 \make@josa를 처리하여 마지막 문자를 취하고 그 문자에 따라 \@jung, \@jong, \ri@ul 중 하나의 값을 되돌려 준다.

```
\def\@josa#1#2{%
  \def\@jung{#1}\def\@jong{#2}\def\ri@ul{\@ifemptyarg{#1}{}{#2}}%
  \edef\lastthree@make@josa{\josa@LastThreeChar{\make@josa}}%
  ...
    \if 0\@last\@jong\else   % 0이면 \@jong 되돌림. 즉 #2.
      \if 1\@last\ri@ul\else
      \if 2\@last\@jung\else
      \if 3\@last\@jong\else
      \if 4\@last\@jung\else
      \if 5\@last\@jung\else
      \if 6\@last\@jong\else
      \if 7\@last\ri@ul\else
      \if 8\@last\ri@ul\else
      \if 9\@last\@jung\else
  ...
}
```

만약 생성된 문자열의 마지막 문자 \@last가 기호문자나 한글 또는 한자라면 josa.tab에서 적절한 값을 가져오도록 하는 것은 hangul.sty의 경우와 동일하다.[13] 그런 다음에, \가와 같은 자동조사 매크로를 이를테면

12. hyperref의 로딩 위치 때문에 이 재정의는 AtBeginDocument에서 이루어진다.
13. 다만, hangul-k의 josa.tab은 HLaTeX의 것에 몇 문자를 더 추가하고 있다.

```
\def\가{\@josa{가}{이}}
```

와 같이 정의해둔다.[14] hangul-k에서는 '\라'와 '\이라'를 새로이 추가하였고 이것은 koTEX에 이어졌다.

hangul-k의 최대 장점은, .aux 파일에 아무런 조작을 가하지 않는다는 것이다. 그 결과 hyperref은 물론이고 상당히 많은 다른 스타일과의 충돌을 해결하였다. 한편, 여전히 해결해야 할 문제로 남은 것은 다음과 같다.

① hyperref에 대한 의존성은 반드시 바람직하지 않다. 모든 문서에 hyperref을 로드하도록 요구하는 것은 약간 불필요한 면이 있다.

② hyperref에 의해 생성되는 hyperlink 텍스트가 아닌, 일반적인 평범한 텍스트에 자동조사를 적용하지 못하였다. hangul-k에서는 궁여지책으로 \jong과 같은 말음 표지를 문서 작성자가 강제로 부여하는 방법을 제안하였으나, 말끔하게 이런 표지 없이 자동조사가 작동하는 것이 바람직할 것이다.

이 두 가지 문제는 hangul-ucs, 즉 koTEX/utf에서 해결된다.[15]

4.2 koTEX의 방법[16]

한글 문헌을 UTF-8 인코딩으로 조판하는 hangul-ucs 패키지는 초창기에 앞에서 설명한 hangul-k와 마찬가지로 hyperref 패키지에 의존하는 방법이 차용되었다. 이 당시 hangul-ucs는 한글 문자의 식자를 ucs 패키지에 의존하고 있었는데, 이 패키지에 알맞도록 hangul-k의 2바이트 문자를 위한 자동조사 정의들을 유니코드 UTF-8 인코딩에 맞게 포팅하는 정도였다.

그러나 곧, 다음과 같은 문제가 드러났다. 첫째, hyperref 패키지는 온라인 pdf 문서를 만드는 데 특화되어 있는 것인 반면, hangul-ucs는 다중목적의 패키지로서 굳이 온라인 문서에만 사용될 것은 아니라는 점. 온라인 문서가 아닌 도서나 저널을 조판하는 데 있어서 hyperref이 굳이 필요할 것은 없다는 것이다. 두번째, 더 심각한 문제로서 hyperref 패키지가 제공하는 \hyper@link 매크로에 약간의 후킹을 추가하는 hangul-k의 방식이 프레젠테이션을 제작하는 데 쓰이던 beamer와 계속 충돌하였다는 것이다. hyperref 패키지에 의존하는 방법을 고수하는 한, 이 충돌을 제거하는 것이 쉽지 않아 보였다.

결국 우리는 hangul-ucs에서의 자동조사 구현에 있어서 hyperref에 대한 의존성을 제거하기로 결정했다. 그러나 이미 언급한 바 HLATEX에서 겪은 바 있는 hyperref과의 충돌을 다시 불러들이고 싶지는 않았다. 이런 양자의 요구를 모두 충족하는 지난한 과업의 유일한 해결책은 \ref, \pageref, \cite와 같은 자동조사에 연관된 주요 명령을 재정의하는 것으로 생각되었다. 이 명령이 불러오는 참조자를 특정 토큰 리스트에 집어넣고 필요하다면 이 토큰 리스트를 분석하여 조사를 결정하게 하자는 것이다.

14. 실제로는 ^^b0^^a1을 정의해야 한다.
15. 현재의 koTEX/euc 버전은 이 부분에서 설명한 hangul-k의 자동조사 방식을 따르고 있다.
16. 이 절은 김도현이 집필한 내용을 거의 그대로 재현하였다.

예를 들어, \ref와 \pageref 명령은 hangul-ucs의 중심 엔진인 dhucs.sty에서 다음과 같이 재정의된다.

```
\AtBeginDocument{%
  ...
  \let\@setref@dhucs@orig\@setref
  \def\@setref#1#2#3{%
    \@setref@dhucs@orig{#1}{#2}{#3}%
    \ifx#1\relax\else
      \gdef\josa@hyper@link@toklist{\expandafter#2#1}%
    \fi}%
}%.
```

이제 \ref{fig:...}나 \pageref{tab:...}와 같은 코드는 원래 이 명령이 수행하던 기능을 그대로 수행할 뿐 아니라[17] 피참조문자열이나 페이지 번호와 같은 숫자를 \josa@hyper@link@toklist라는 매크로에 저장하여 나중에 이를 처리할 수 있도록 하고 있음을 볼 수 있다. cite나 natbib와 같이 참고문헌을 처리하는 다른 패키지에서도 이와 유사한 재정의를 활용한다. 그러므로 이러한 패키지와 함께 사용될 경우 상호간 같은 명령을 따로 재정의하는 데 따른 위험이 없는 것은 아니라고 생각한다. 그런 종류의 충돌이 발견되거나 보고되면 자동 조사 처리 메카니즘을 확장하여 나가야 할 것이다.

\josa@hyper@link@toklist에 저장된 토큰을 분석하는 일은 \이나 \가와 같은 자동조사 명령이 불릴 때 활성화된다. 우리말 조사 '가'(U+AC00)는 UTF-8 인코딩으로 0xEA 0xB0 0x80이라는 세 개의 옥텟으로 표현되며, '이'(U+C774)는 0xEC 0x9D 0xB4로 인코딩된다. "other character"(catcode 12) 범주에 속하는 문자는 그 앞에 이스케이프 문자를 추가하여 "control symbol"로 만들 수 있으므로, 자동조사 명령을 다음과 같이 정의한다.

```
\def\^^ea#1#2{%
  \ifx#1^^b0\ifx#2^^80% 가
     \make@josa{가}{이}%
    \else\error@josa\fi
  \else\error@josa\fi}
\def\^^ec#1#2{%
  \ifx#1^^9d\ifx#2^^b4% 이
     \make@josa{가}{이}%
    \else\error@josa\fi
  \else\error@josa\fi}
\def\error@josa{\PackageError{dhucs}%
  {Undefined Control Sequence}{}}
```

입력 문서의 원본에서 \ref{...}\가와 같은 코드를 만나면 \make@josa라는 매크로 명령으로 확장된다. 이 명령은 \josa@hyper@link@toklist에 저장된 토큰을 검토하여 첫번째 인자('가') 또는 두번째 인자('나') 가운데 하나를 되돌려준다.

```
\def\make@josa#1#2{%
  \josa@hyper@link
```

17. latex.ltx의 \@setref 명령과 비교해보면, 위의 정의는 원본 정의를 거의 그대로 가져다 쓰고 있음을 알 수 있다.

```
    % ...
    \ifnum\@josa=\@ne \ifx#1\empty #1\else#2\fi% rieul
    \else \ifnum\@josa=\tw@ #1% jung
    \else #2% jong
    \fi\fi}
```

이 코드에서 보는 대로, `\make@josa`는 `\@josa`라는 카운터를 활용하는데, 이 카운터는 0, 1, 2 가운데 하나가 된다. 0은 이 토큰 리스트의 마지막 음소가 (ㄹ이 아닌) 종성 자음임을 의미하며, 1이면 'ㄹ', 2면 종성이 없는 모음으로 끝난다는 것을 뜻하는 것이다.

```
\newcount\@josa
\def\jong{\global\@josa\z@}
\def\rieul{\global\@josa\@ne}
\def\jung{\global\@josa\tw@}
```

토큰열의 마지막 글자를 얻기 위해서 다음과 같이 하였다.

```
\def\josa@hyper@link{%
  \begingroup
  \edef\dh@josa@temp{\josa@hyper@link@toklist}%
  \toks@{}%
  \expandafter\josa@hyper@link@\dh@josa@temp\@nil
  \expandafter\josa@hyper@link@@\the\toks@
    \empty\empty\empty\empty\@nil
  \endgroup}
```

여기서 `\josa@hyper@link`는 또다른 매크로 `josa@hyper@link@`를 부르는데, 이것은 후미재귀(trail recursion) 기법 [7, 103–104쪽]을 이용하여 토큰을 역순으로 임시 토큰 리스트 `\toks@`에 저장한다.

```
\def\josa@hyper@link@#1{%
  \ifx#1\@nil\let\next\relax
  \else \let\next\josa@hyper@link@
    \if\noexpand#1\relax\else
      \edef\act{\noexpand\toks@{#1\the\toks@}}\act
    \fi
  \fi\next}
```

이제 마지막 문자가 제일 처음 위치에 온 또다른 매크로 `\josa@hyper@link@@`를 얻었는데, 이것은 다시 `\josa@hyper@link`에 의하여 호출되어 첫문자가 무엇인지를 확인하고 이로부터 적절한 조사 이형태를 결정하게 한다.

```
\def\josa@hyper@link@@#1#2#3#4\@nil{%
 ...
  \ifx 1#1\rieul\else % il
  \ifx 3#1\jong\else  % sam
  \ifx 6#1\jong\else  % yuk
  \ifx 7#1\rieul\else % chil
  \ifx 8#1\rieul\else % pal
  \ifx 0#1\jong\else  % yong
```

```
    \ifx l#1\rieul\else % el
    \ifx m#1\jong\else  % em
    \ifx n#1\jong\else  % en
    \ifx L#1\rieul\else % el
    \ifx M#1\jong\else  % em
    \ifx N#1\jong\else  % en
      \jung
    \fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi\fi
    \ifx n#2\ifx g#1\jong\fi\fi %skating
    % ...
    \ifx i#3\ifx m#2\ifx e#1\jong\fi\fi\fi %time
    % ...
    }
```

조사 선택을 위해서는 마지막 문자 하나만 있으면 되지만, 이런 간단한 추적 방법은 어떤 경우 마지막 문자를 적절하게 얻는 데 실패하는 경우도 있다. 그래서 마지막 한 글자만을 체크하지 않고 원래의 토큰열에서 마지막 두세 글자를 얻어서 검사하도록 하고 있다.

이상의 복잡한 과정을 거쳐 조사 자동 선택에 필요한 메카니즘을 거의 구현했다고 생각한다. 이제 남은 것은 원래의 토큰열의 마지막 글자가 아스키 문자가 아니라 CJK 문자일 경우이다. 이를 위해서 CJK 문자를 식자하는 매크로 `\unihangulchar`를 추가한다.

```
\DeclareRobustCommand*\unihangulchar[1]{%
  %...
  \global\@josa=#1
  %...
}
```

`\unihangulchar`의 인자는 유니코드 코드포인트이다. 즉 '가'를 AC00으로 식자한다. 위의 코드는 이 유니코드 코드 포인트를 `\@josa` 카운터에 할당하는 것뿐이다. 그 다음에 위에서 말한 `\make@josa` 명령에 의해 조사가 결정된다. `\make@josa` 매크로에는 이렇게 할당된 유니코드 코드포인트 번호를 적절한 조사 유형으로 구분하는 다음과 같은 코드가 포함되어 있다.

```
\ifnum\@josa>44031 \ifnum\@josa<55204 % hangul syllables
  \advance\@josa-44032 % "AC00
  \count@=\@josa \divide\count@28 \multiply\count@28
  \advance\@josa-\count@ \advance\@josa4519 % "11A7
  \ifnum\@josa=4519 \jung
  \else\ifnum\@josa=4527 \rieul % "11AF
  \else \jong \fi\fi \fi\fi
```

이 코드는 유니코드 Consortium이 제공하는 "Hangul Decomposition"을 텍 코드로 구현한 것이다.[9] 이것은 한글 음절 문자(U+AC00…U+D7A3)의 유니코드 번호를 각각에 해당하는 한글 자모 조합(U+1100…U11FF) 유니코드 번호로 변환하고 이로부터 종성만을 취하여 해당 문자가 모음으로 종결되는지, ㄹ 종성인지 기타 종성인지 구별하고 있다. 한자와 기호문자일 경우는 어떻게 하는가? 한자는 그 한글음을 얻어서 같은 방식으로 종성

유형을 얻고 기호문자의 경우에도 비슷한 방법을 사용한다.[18]

5 맺는말

조사 이형태 선택의 자동화는 한글 라텍에서 빠뜨릴 수 없는 기능이다. 지금까지 구현된 아이디어는 크게 두 가지로 나눌 수 있는데 그 하나는 말음 표지를 미리 `.aux`에 기록해 두었다가 이를 나중에 말음 코드로 변환하여 조사를 선택하는 방법이고, 다른 하나는 직전 문자열을 직접 분석하여 이로부터 조사를 결정하는 방법이다.

조사 이형태 선택 문제가 해결되어 온 과정을 살펴보면 한글 라텍의 극적인 발전 과정을 한눈에 볼 수 있다. 또한 유니코드를 중심으로 하는 kωTEX으로 이행해야 할 당위성이 극명하게 드러난다. kωTEX이 유니코드를 채택함으로써 얻을 수 있었던 것은 비단 확장 문자의 식자에 한정된 것이 아니라 내부적인 처리과정에 있어서도 대단히 유용한 이점을 획득하였음을 알 수 있다.

현재 자동 조사에 관한 문제는 kωTEX에 와서 거의 해결되었다. 그러나 조사 선택을 위하여 상당한 노력을 기울이지 않으면 안되는 상황을 피할 수는 없다. 향후 한글 라텍 또는 텍 관련 시스템을 구축하는 데 있어서 새로운 텍 엔진을 구현하거나 요구하게 될 때, 텍의 `\sfcode`나 `\efcode`와 유사한 새로운 원시명령(primitive) `\josacode` 같은 것을 추가한다면 위에서 언급한 번거로운 절차를 일소할 수 있을 것이다. 현재 luaTEX이 빠르게 발전하고 있으므로 자동 조사 선택의 구현에 있어서도 본격적으로 프로그래밍 언어를 활용할 수 있는 새롭고 효율적인 방법을 기대한다.

참고 문헌

1. 김강수, *hangul-k* 사용자 안내서, 2004. http://faq.ktug.or.kr/faq/HANGULkStyle
2. ———, *HLATEX* 이후 *LATEX*의 발전, 2006. http://faq.ktug.or.kr/faq/Karnes
3. 남기심·고영근, 『표준 국어문법론』(개정판), 탑출판사, 1992.
4. 은광희, 한글라텍 길잡이, 2005. http://project.ktug.or.kr/hlatex/hlguide.pdf
5. 은광희·김도현·김강수, 한국어 텍 *kωTEX v0.1.0* 사용 설명서, 2007. http://project.ktug.or.kr/ko.TeX/kotexguide.pdf
6. 차재춘, 한글 *LATEX* 사용자 설명서, 1995. http://knot.kaist.ac.kr/htex/
7. V. Eijkhout, *TEX by Topic*, Addison-Wesley, 1992.
8. F. Mittelbach, M. Goossens, and et. al., *The LATEX Companion*, 2nd edition, Addison-Wesley, 2004.
9. Unicode Consortium, *Unicode Normalization Forms*, Unicode Standard Annex #15, 2006. http://unicode.org/reports/tr15/

18. 더 자세한 구현은 `dhucs.sty` 소스 코드를 참고하라.