The Asian Journal of TEX, Volume 4, No. 1, June 2010
Article revision 2010/5/13

XeTeX-k*o*: XeTeX 엔진을 위한 한국어 조판 매크로

XeTeX-k*o*: A XeTeX Macro Package for Processing Korean Documents

김도현 Dohyun Kim
동국대학교 법과대학 nomos@ktug.or.kr

Keywords XeTeX, XeTeX-k*o*, Hangul, Korean, OpenType, TrueType, character spacing

Abstract XeTeX-k*o* is a macro package for typesetting Korean documents, including old Hangul texts as well, upon XeTeX engine. XeTeX is a sophisticated TEX-engine which supports quite well full Unicode encoding and OpenType layout features. Using XeTeX itself, however, is not fully satisfactory in the eyes of the Korean, especially because of relatively poor quality of latin/greek/cyrillic glyphs in many Korean fonts. For this reason among others, XeTeX-k*o* has been recently developed, mainly focusing on how to typeset with different fonts between Western and Korean characters. This paper presents current state of XeTeX-k*o* package along with its main features and usages, including among others how to configure Korean fonts, how to change character spacing, and what to be prepared for typesetting old Hangul texts. At the end of the paper, some limitations of current XeTeX-k*o* package will also be discussed as a warning to the users.

1 왜 XeTeX-k*o*인가?

XeTeX은 특별히 따로 매크로 패키지를 제공하지 않아도 자체적으로 한글을 포함하는 한국어 문서를 잘 처리할 줄 안다. 게다가 오픈타입이나 트루타입 폰트도 잘 지원하므로 레거시 엔진을 이용할 때와 달리 따로 폰트를 조성하여 설치해주는 수고를 부담하지 않아도 된다. 이를테면 시스템에 설치되어 있는 은글꼴 트루타입을 이용해 한국어 문서를 조판하기 위해서는 다음과 유사하게 한국어 폰트를 지정해주기만 하면 된다.

```
\usepackage{fontspec}
\setmainfont[Mapping=tex-text]{UnBatang}
\setsansfont[Mapping=tex-text]{UnDotum}
\setmonofont{UnTaza}
\XeTeXlinebreaklocale="ko"
```

전처리부(preamble)에 이렇게 써넣고 문서를 컴파일하는 것으로 비교적 쓸만한 결과물을 얻을 수 있다. 더욱이 마지막 행의 명령으로 유니코드 표준에 따라 줄바꿈도 저절로 지원되니 사용자로서는 이보다 더 편리할 수 없다.

그럼에도 불구하고 XeTeX-k*o*라는 매크로 패키지를 만드는 수고를 감내하는 이유는 무엇인가? 한글이 잘 찍히고 줄바꿈도 거의 문제 없이 이루어지는 마당에 무슨 할 일이 더 남아있는가? 하지만 XeTeX의 원시명령(primitive) 및 서양에서 만들어진 매크로 패키지만으로는 미려한 한국어 문서 조판에 요구되는 몇 가지 사항을 충족시킬 수 없다. 우선 한국어 문장 가운데 등장하는 라틴, 그리스, 키릴 문자 따위마저 모두 한국어 폰트로 식자되는 문제를 지적할 수 있다. 주지하다시피 한국어 폰트에 들어있는 서양 문자 글리프들은 눈높이가 이미 높게 형성된 텍 사용자에게는 대체로 만족스럽지 못한 것이다. 게다가 글리프와 글리프 사이에 들어가야 할 커닝(kerning)이 한국어 폰트에서는 제대로 지원되지 않는 경우가 많다. 또한 리거처(ligature)를 지원하는 한국어 폰트도 거의 없는 형편이다. 그리스 문자나 키릴 문자 글리프가 한국어 폰트에 대체로 들어있지만 전각폭을 가지고 있어서 도저히 주어진대로 쓸 수 없는 경우가 일반적이다.

다시 말해서 문제는 한국어 폰트에 있는 것이다. 한국어 폰트의 라틴, 그리스, 키릴 영역의 글리프들이 서양에서 만들어진 서양 폰트만큼 훌륭한 품질을 가진다면 굳이 XeTeX-k*o*가 필요하지 않을지도 모른다. 그러나 현실은 그렇지 못하거니와 이것은 어쩌면 당연하다고 할 수도 있다. 우리가 아무리 노력한다 해도 서양에서 만든 서양 폰트의 품질만큼 뛰어난 품질을 한국어 폰트의 서양 글리프들에 요구하기는 힘든 노릇이 아니겠는가. 결국 그저 그런 워드프로세서 수준의 결과물에 만족하지 못하는 텍 사용자들을 위해서는 한국어 문서 가운데 등장하는 라틴, 그리스, 키릴 문자를 한국어 폰트가 아니라 서양 폰트로 찍지 않을 수 없다는 결론이 나온다. 즉 한중일 문자와 비 한중일 문자를 서로 다른 폰트로 식자할 필요가 있다.

XeTeX-k*o* 없이 한국어 문서를 조판할 때 발생할 수 있는 또 하나의 문제는 한국어 자간(character spacing)에 관한 것이다. 음절을 네모난 박스 속에 집어넣어 식자하지만 한자에 비해 획수가 상대적으로 적은 한글의 특성 때문에 한글 글리프들을 있는 그대로 식자하면 판면이 듬성듬성해 보이는 경우가 종종 있다. 따라서 예전부터 우리는 한글 글자 사이에 미세한 마이너스 자간을 삽입하여 보기 좋은 한글 문서를 만드는 데 익숙해져 있는 것이다. 한편 이러한 자간 문제는 한글 글자들 사이에서만 발생하는 것은 아니고, 한중일 문장부호, 이를테면 낫표(「」) 따위의 앞뒤에서도 문제가 될 수 있다. 전통적으로 한국어 폰트의 이들 문장부호는 전각(full-width)의 폭을 가지는 것이 보통인데, 가로쓰기 방식이 일반화된 오늘날에는 이들 전각 문장부호들이 그 앞이나 뒤에 지나치게 넓은 공백을 가져오는 문제가 있다. 따라서 전각 문장부호의 앞뒤에 적절하게 마이너스 자간을 넣어주어야 할 필요가 생긴다. 하지만 XeTeX 자체만으로는 한글 사이에, 또는 전각 부호 앞뒤에 마이너스 자간을 설정할 수가 없고, 폰트에서 지정한 글자폭이 있는 그대로 식자되어 만족스럽지 못한 결과를 가져온다.

요컨대 한중일 문자와 비 한중일 문자를 서로 다른 폰트로 식자할 필요성과 더불어 한중일 문자들 사이에서도 적절한 간격 조정이 필요할 경우가 있으므로 XeTeX 자체가 제공하는 원시명령 및 이와 관련하여 서양에서 만들어진 매크로 패키지만으로는 품위 있는 한국어 문서의 조판을 기대하기는 어렵다는 결론이다. 따라서 비록 고되고 지루한 작업이기는 하지만 XeTeX을 기반으로 하는 한국어 문서 조판 매크로를 새로 개발하지 않으면 안 되는 상황에 놓였던 것이다.

2 새로운 매크로의 기본 원리

그렇다면 한중일 문자와 비 한중일 문자를 어떻게 다른 폰트로 찍을 것이며 한중일 문자들 사이에 어떻게 적절한 간격을 삽입할 것인가? 레거시 텍 엔진 시절의 k*o*TEX 매크로는 한중일 문자에 해당하는 바이트에 모두 캣코드(catcode) 13을 부여하여 문자명령(active character)을 만드는, 즉 바이트 하나만으로 자체적으로 명령어의 역할을 하게 만드는 트릭을 사용하였다.[1] 레거시 텍 엔진은 256개의 문자만으로 입력을 받아들이므로 영문자(ASCII 문자)를 제외한 나머지 바이트들의 캣코드를 따로 부여하는 것이 그리 어려운 일이 아니었다. 하지만 유니코드를 엔진 차원에서 지원하는 XETEX에서는 한중일 문자 전부에 대해서 캣코드를 수정한다는 것은 상상하기 곤란한 일이다. 완성된 한글 음절만 하더라도 11,172개의 문자가 있고 여기에 한자, 일본어 가나, 문장부호 따위를 다 합치면 어마어마한 개수의 문자에 새로운 캣코드를 부여해야 하는 문제가 되는 것이다. 게다가 그렇게 캣코드가 수정된 문자 하나하나마다 일일이 매크로 명령을 따로 정의해야 하니 사실상 불가능한 작업이라 하지 않을 수 없다.

이러한 까닭에 무언가 다른 돌파구가 주어지지 않으면 한중일 문자와 비 한중일 문자를 분리하여 취급하려는 XETEX-k*o*의 시도가 벽두부터 난관에 부딪치지 않을 수 없게 된다. XETEX이 서양 문서 조판에 널리 쓰이기 시작한 뒤에도 한동안 XETEX-k*o*가 세상의 빛을 보지 못한 이유가 여기에 있었다. 그러던 차 2008년 즈음에, 한중일 사용자들의 열화와 같은 요청에 못 이겨 결국 새로운 원시명령들이 XETEX에 포함되게 되었으니, 문자 토큰들을 임의의 클래스로 구분하고 특정 문자와 문자 사이에 임의의 텍 명령을 삽입할 수 있게 하는 일련의 원시명령들이 등장한 것이다. 이들 원시명령은 본래 일본어 문장부호의 전후 간격을 조정하기 위하여 만들어졌다고 하므로 [8], 우리의 의도에 정확히 부합하는 것이었다. 이를 이용하여 이제 다음과 같은 조작을 할 수 있게 되었다.

```
1 \newXeTeXintercharclass \mylatin
2 \newXeTeXintercharclass \myhangul
3 \XeTeXcharclass `\A  \mylatin
4 \XeTeXcharclass `\가 \myhangul
5 \XeTeXinterchartoks \mylatin \myhangul = {\myhangulfont}
6 \XeTeXinterchartoks \myhangul \mylatin = {\mylatinfont}
7 \XeTeXinterchartokenstate = 1
```

즉, 새로운 문자 클래스들을 선언하고(1–2행) 각 문자 코드를 이들 클래스에 할당한 다음(3–4행), 식자할 문자열 가운데 할당된 문자 토큰의 조합을 만나게 되면 임의의 텍 명령, 사례에서는 폰트를 바꾸는 명령을 그 사이에 삽입하는 것이다(5–6행). 마지막 제7행은 이러한 토큰 검사 및 임의의 텍 명령 삽입 기능을 켜라는 뜻의 명령이며 이 값이 영(0)이면 반대로 끄라는 의미이다. 물론 사례처럼 단순한 코드만으로 모든 경우를 다 처리할 수는 없다. 특히 문자가

1. 캣코드 수정은 사실 inputenc라는 LATEX 패키지가 수행하는 일이었다. 따라서 k*o*TEX이 자체적으로 캣코드를 수정했다기보다는 다만 inputenc 패키지를 이용했거나 inputenc를 이용하는 패키지를 간접적으로 이용했던 것에 불과하다고 해야 정확한 표현이 될 것이다. '범주코드(category code)'라고도 부르는 캣코드의 기능과 유형에 대해서는 [4, 37–41]을 볼 것.

아닌 토큰, 이를테면 글루(glue), 컨(kern), 페널티(penalty), 박스(hbox) 따위와 문자 토큰이 만났을 때 어떻게 할 것인가도 정의해 두어야 한다. XeTeX은 문자가 아닌 토큰을 모두 255번 클래스로 할당해두고 있으므로 위 코드에는 사실 255번 클래스와 한글이 만났을 때, 한글과 255번 클래스가 만났을 때, 255번 클래스와 라틴 문자가 만났을 때, 라틴 문자와 255번 클래스가 만났을 때 따위도 선언되어 있어야 하지만, 여기서는 지면 관계상 생략하였다. 어쨌든 이렇게 새롭게 제공되는 원시명령을 이용하여 한중일 문자와 비 한중일 문자를 분리하여 그 사이에서 폰트 변경을 한다든가, 한글 문자 사이나 문장부호 앞뒤에 자간을 조정하는 글루를 집어넣는다든가 하는 일을 비교적 수월하게 할 수 있게 되었다.[2]

나아가 XeTeX-k*o*는 이러한 XeTeX의 새로운 원시명령들을 폰트나 자간 조절을 위해서만 이용하는 것이 아니다. 이른바 "자동조사" 기능을 구현하기 위해서도 사용하고 있으니, 이를 통하여 레거시 엔진 상에서 k*o*.TeX이 구현하던 자동조사 기능의 부족한 점을 극복할 수 있었다. 과거 k*o*.TeX에서는 상호참조의 대상이 아닌 라틴 문자열 직후에 자동조사 명령이 왔을 때 자동조사가 제대로 작동하지 않는 한계가 있었다. 예를 들어 "`KTS\는`"이라는 입력이 있을 때 자동조사 "`\는`"이 정상적으로 동작하리라고 보장할 수 없었으니, 이는 라틴 문자들을 문자명령으로 만들 수 없기 때문이었다. 하지만 XeTeX-k*o*에서는 입력 문자열 가운데 한중일 문자를 만날 때 뿐만 아니라 라틴 문자나 숫자를 만날 때에도 매번 해당 문자의 코드를 임시 변수에 저장하여 필요한 때 이용할 수 있게 되었으므로 이 경우에도 자동조사가 정확하게 동작한다. 이것이 가능하게 된 것도 문자 사이 명령 삽입 기능을 제공하는 XeTeX의 새로운 원시명령 덕분이다.

3 글꼴의 분리

이렇게 하여 XeTeX-k*o*는 무려 20여 가지에 달하는 문자 클래스를 각 문자 코드에 할당하였고 이들 문자 클래스가 변경될 때마다 식자할 폰트를 바꿀 수 있도록 매크로를 제공하고 있으니, 이제부터는 XeTeX-k*o* 상에서 폰트를 어떻게 설정하는지 살펴보기로 한다. XeTeX-k*o*는 플레인 텍(plain TeX)도 지원하지만 여기서는 LaTeX을 사용하는 경우에 국한하여 이야기하기로 하겠다.[3]

기본적으로 XeTeX-k*o*에서는 문자의 종류에 따라 세 가지 폰트를 사용할 수 있는데 이를테면 다음과 같이—물론 `\usepackage{xetexko}` 또는 `\usepackage{kotex}`[4] 명령을 내린 다음에—설정한다.

```
\setmainfont[Mapping=tex-text]{Linux Libertine}
\setmainhangulfont[Mapping=tex-text]{나눔명조}
\setmainhanjafont[Mapping=tex-text]{UnBatang}
```

2. XeTeX의 새로운 원시명령을 이용하여 문자 클래스를 나누고 토큰들 사이에 각종 텍 명령을 삽입하는 XeTeX-k*o*의 매크로 파일은 xetexko-space.sty이며 여기에 XeTeX-k*o*의 가장 핵심적인 코드가 담겨있다.
3. 플레인 텍에서의 XeTeX-k*o* 사용법에 대해서는 [6]을 볼 것. LaTeX 패키지인 fontspec의 각종 명령에 대응하는 XeTeX-k*o*의 매크로들은 xetexko-font.sty 파일에서 제공하고 있다.
4. k*o*.TeX의 통합 패키지 모음에서는 `\usepackage{kotex}`만 선언하더라도 XeTeX 엔진이 동작하는지 여부를 검사하여 XeTeX 엔진이 돌고 있다면 자동적으로 xetexko 패키지를 불러들인다.

세 가지 모두 fontspec 패키지 [7] 방식으로 폰트를 설정할 수 있으며 그 가운데 `\setmainfont`는 fontspec 패키지가 제공하는 명령 그대로여서 XeTeX-k*o*가 따로 매크로 조작을 가하지 않는다. 이 첫 번째 명령은 원칙적으로 라틴, 그리스, 키릴 문자 뿐만 아니라 아랍, 헤브루 등등 한중일 문자가 아닌 모든 문자에 대해 적용된다. 한편 두 번째와 세 번째 명령은 fontspec의 매크로를 XeTeX-k*o*가 수정한 것으로 이름 그대로 각각 한글과 한자를 식자하는 데 쓰인다. 여기서 한글이라 함은 완성된 한글 음절 영역(U+AC00부터 U+D7A3까지)과 더불어 옛한글에 쓰이는 한글 조합 자모 영역(U+1100부터 U+11FF, 그리고 U+D7B0부터 U+D7FB까지)[5] 및 방점(U+302E와 U+302F)을 지칭하는 말이다. 한글 호환 자모 영역(U+3131부터 U+308E까지)은 한중일 기호로 취급하여 한글 폰트가 아니라 한자 폰트로 식자하게 하였다. 즉, 고유한 의미의 한자 영역뿐만 아니라 한중일 문장부호와 그밖의 한중일 기호들이 모두 한자 폰트로 식자된다.

fontspec 패키지가 세리프체에 해당하는 `\setmainfont`뿐만 아니라 `\setsansfont` 및 `\setmonofont` 명령을 제공하듯이 XeTeX-k*o*도 이에 대응하여

```
\setsanshangulfont
\setsanshanjafont
\setmonohangulfont
\setmonohanjafont
```

명령을 마련해 두고 있다. 앞의 두 명령은 산세리프 글꼴에 해당하고 뒤의 두 명령은 고정폭 글꼴에 해당하는 것이다. XeTeX-k*o*가 제공하는 이들 폰트 명령에도 fontspec 패키지의 폰트 옵션들을 그대로 다 사용할 수 있다.

한편 fontspec 패키지는 세리프, 산세리프, 고정폭에 해당하는 세 가지 글꼴 설정 명령 이외에도 임의의 폰트를 지시하는 명령으로 `\newfontfamily`와 `\newfontface`를 제공하고 있다. 전자는 이름대로 굵은 글꼴, 이탤릭 글꼴, 굵은 이탤릭 글꼴을 한꺼번에 간편하게 지시할 수 있는 명령인 반면, 후자는 이 가운데 하나의 글꼴만 지시하는 명령이다 [7]. 역시 이들 명령에 대응하여 XeTeX-k*o*는 다음과 같은 매크로를 제공하고 있으며 한글과 한자에 적용된다는 점을 제외하고는 fontspec의 그것들과 기능이 동일하다.

```
\newhangulfontfamily
\newhanjafontfamily
\newhangulfontface
\newhanjafontface
```

나아가 fontspec 패키지는 본문 중에서 임시로 폰트를 변경할 수 있는 명령도 제공하고 있는데 `\fontspec` 명령이 그것이다. 이는 사실 fontspec 패키지의 핵심 매크로에 해당하는 것으로서 패키지 이름이 바로 이 명령에서 유래하였던 것이다. 짐작하였다시피 XeTeX-k*o*는 이 명령에 대응하여 한글 및 한자 영역에 대하여 적용되는 매크로도 물론 제공하고 있다.

```
\hangulfontspec
\hanjafontspec
```

5. 이 중에 0x115A–0x115E, 0x11A3–0x11A7, 0x11FA–0x11FF, 0xA960–0xA97C, 0xD7B0–0xD7C6, 0xD7CB–0xD7FB는 2009년 10월 발표된 유니코드 5.2 버전에서 새로 들어간 자소들이다.

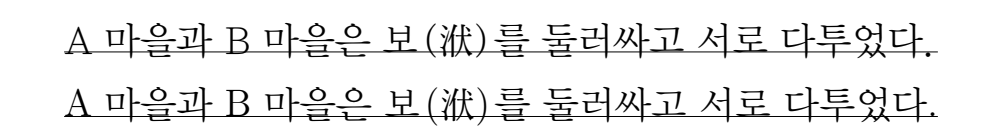

그림 1. 영문자와 괄호 · 마침표 등 문장부호를 한글 폰트(위)와 영문 폰트(아래)로 각각 식자한 예

또한 예전 k*o*TEX에 이와 유사한 기능을 하는 다른 이름의 명령이 있었던 점을 고려하여 이들 명령을 각각 `\adhochangulfont` 및 `\adhochanjafont`로도 쓸 수 있게 하였다.

끝으로 fontspec 패키지는 `\addfontfeature(s)` 명령도 제공하고 있는데, 이미 활성화되어 있는 글꼴에 임시로 폰트 옵션을 추가하여 글꼴에 색깔을 잠시 입힌다거나 특정 오픈타입 속성(OpenType layout features)을 잠시 켜거나 끄는 작업을 할 수 있는 것이다. 이에 대응하는 XHTEX-k*o*의 명령은 다음과 같으며 fontspec의 경우와 마찬가지로 명령어 이름 끝의 "s"는 있어도 좋고 없어도 좋다.

```
\addhangulfontfeature(s)
\addhanjafontfeature(s)
```

지금까지 한글 및 한자 영역에 해당하는 여러 폰트 지시 명령을 소개하였거니와, fontspec 패키지의 각종 명령에 대응하는 이들 XHTEX-k*o* 매크로의 구체적인 사용법은 fontspec의 그것과 전혀 동일하고 여기서 그것을 상술할 여유가 없으므로 자세한 것은 fontspec 패키지 매뉴얼을 참조하기 바란다 [7].

4 유연한 글꼴 명령

이렇게 하여 비 한중일 문자 영역과 한글 영역 그리고 나머지 한중일 영역 사이의 글꼴 분리를 구현할 수 있었다. 그런데 경우에 따라서는 문자 영역 별로 분리되어 고정된 글꼴 지정을 좀 더 유연하게 처리하고 싶은 때가 있을 수 있다. 대개 문장부호(punctuations)와 관련하여 이러한 필요성이 제기되지만 반드시 문장부호에 국한된 것만은 아니다.

이를테면 그림 1에서 제시된 두 줄은 언뜻 큰 차이가 없어보지만 자세히 관찰해보면 한글과 영문자 · 괄호 · 마침표 사이의 어울림 정도가 아랫줄보다 윗줄이 더 뛰어나다는 데 누구도 이의를 달지 않을 것이다. 아랫줄의 영문자 "A"와 "B"는 베이스 라인이 한글에 비해 조금 위로 올라 붙은 느낌이다. 또한 한자를 둘러싸고 있는 괄호는 아랫줄의 경우 오히려 밑으로 너무 내려와 있다. 게다가 결정적으로 아랫줄의 마침표는 한글에 비해 다시 너무 위로 올라가 식자되어 있다. 그림의 윗줄은 영문자 · 괄호 · 마침표를 모두 한글 폰트, 즉 윤명조로 식자한 결과이고 아랫줄은 영문 폰트, 즉 라틴 모던 로만체로 식자한 결과이다.

이렇게 한글이 주가 되는 문서에서 잠깐 영문자나 숫자가 등장할 때는 비록 라틴 문자 영역에 속하기는 하지만 이들을 한글 폰트로 식자하는 쪽이 더 바람직하다는 결론이다. 또한 괄호나 마침표 따위의 문장부호 역시 한글 문장에서는 한글 폰트로 식자하는 쪽이 더 낫다. 특히 괄호의 경우에는 괄호 속의 문자들이 한글인지 영문자인지 판별해서 한글인 때에는 한글

표 1. 유연한 글꼴 명령의 문자 할당 및 그 예시

이름	의미	예시
alphs	아래에 해당 않는 비 한중일 문자	a b c A B C
nums	라틴 숫자(수식 제외)	1 2 3 4 5 0
parens	라틴 괄호	( ) [ ] { } < >
quotes	악상그라브 및 어포스트로피와 그 리거처	‘ ’ “ ”
hyphens	라틴 하이픈과 그 리거처, 슬래시	- – — /
colons	라틴 쌍점, 쌍반점, 엠대시, 엔대시	: ; – —
puncts	라틴 마침표, 물음표, 느낌표, 쉼표	. ? ! ,
cjksymbols	한중일 구두점과 상징기호	。、「」▲ ☞

폰트로, 영문자일 때에는 영문 폰트로 식자하면 좋을 것이다. 현재 한창 개발 중인 LuaTEX 엔진을 이용한다면 루아 프로그래밍을 통해 사용자 명령을 최소화하면서 거의 자동적으로 이런 문제를 처리할 수 있겠지만, 기껏해야 토큰 수준의 단순한 텍 매크로만 가능할 뿐인 XETEX 엔진 위에서는 다소 번잡하더라도 다양한 사용자 명령을 제공하고 사용자가 적절히 이들 명령을 선택하게 함으로써 의도한 효과를 얻을 수밖에 없을 것이다. 그리하여 XETEX-k*o* 는 다음과 같은 일련의 매크로들을 마련하였다.

```
\latinalphs      \hangulalphs      \hanjaalphs      \prevfontalphs
\latinnums       \hangulnums       \hanjanums       \prevfontnums
\latinparens     \hangulparens     \hanjaparens     \prevfontparens
\latinquotes     \hangulquotes     \hanjaquotes     \prevfontquotes
\latinhyphens    \hangulhyphens    \hanjahyphens    \prevfonthyphens
\latincolons     \hangulcolons     \hanjacolons     \prevfontcolons
\latinpuncts     \hangulpuncts     \hanjapuncts     \prevfontpuncts
\latincjksymbols \hangulcjksymbols \hanjacjksymbols \prevfontcjksymbols
```

언뜻 복잡해 보이지만 이들 매크로에는 뚜렷한 일관성이 있음을 쉽게 알 수 있다. 우선 latin, hangul, hanja, prevfont 가운데 하나의 접두어가 붙어 있으니, 각각 라틴 폰트, 한글 폰트, 한자 폰트, 직전 폰트로 식자하라는 의미이다. 다른 것들은 그 뜻이 자명하므로 따로 설명할 것은 없을 터이나, 직전 폰트에 대해서는 약간의 부연 설명이 필요하겠다. 여기서 prevfont는 예컨대 마침표를 식자함에 있어 “하였다.” 처럼 마침표 직전에 한글이 왔다면 마침표도 한글 폰트로 찍고 그렇지 않고 영문자가 직전에 왔다면 마침표도 영문 폰트로 찍으라는 뜻이다. 물론 한자 다음의 마침표는 한자 폰트로 식자하게 된다. 마침표, 쉼표, 물음표, 느낌표, 쌍점 따위의 구두점은 대개 그 앞에 문자가 오고 그 뒤에는 공백이 오게 되므로 직전 문자에 종속적인 성질을 가지고 있고, 따라서 한글 폰트로만 또는 영문 폰트로만 식자하기보다는 앞서 그림 1에서 본 것처럼 직전 문자의 폰트를 따라가는 쪽이 바람직한 것이다.

한편 각 명령마다 alphs, nums, parens, quotes, hyphens, colons, puncts, cjksymbols 가운데 하나가 이름에 포함되어 있으니 그 의미는 표 1에 예시와 함께 제시되어 있다. 따라서 예컨대 `\latinalphs` 명령은 알파벳이나 그리스 문자 따위를 라틴 폰트로 식자하게 하고,

\hangulparens는 라틴 괄호를 한글 폰트로 식자하게 한다. 그런데 표에서 의문스러운 점이 하나 있을 수 있는데, 동일한 모양의 줄표(—)가 hyphens 항목에서도 colons 항목에서도 발견된다는 것이다. 여기서 hyphens 항목의 줄표는 하이픈(-)을 세 번 잇따라 써서 리거처로 표현된 것이고 colons 항목의 줄표는 직접 유니코드 U+2014를 입력하여 엠대시(emdash)로 식자한 것이다. 이렇게 동일한 모양을 다른 항목에 굳이 배치한 이유는 XeTeX-k*o*에서 colons 구두점 앞뒤에 미세한 간격이 삽입되도록 해 두었기 때문이다. 즉 사용자가 하이픈 세 개를 연달아 입력하여 줄표를 표현하면 "한글—한글" 처럼 식자되고 U+2014를 직접 입력하여 표현하면 "한글—한글" 처럼 식자된다. 보이는 바처럼 후자의 경우 줄표 앞뒤에 아주 미세하지만 간격이 들어가게 된다(후술하겠지만 이 간격의 크기는 사용자가 조절할 수 있다). LuaTEX이라면 노드(node) 수준에서 프로그래밍을 구사할 수 있으므로 이 두 가지 경우를 동일하게 취급할 수 있지만, 기껏해야 토큰(token) 수준의 매크로만 다룰 수 있는 XeTeX에서는 하이픈과 하이픈의 리거처를 구별하기가 무척 어렵고 그렇다면 U+2014를 colons 항목에 배치하여 줄표 전후의 미세간격 여부를 사용자가 선택할 수 있게 한 것이다.

이상과 같은 유연한 글꼴 명령들은 그 수가 너무 많아 실제 사용에 있어 번거롭다고 하지 않을 수 없다. 따라서 XeTeX-k*o*는 표 1의 모든 항목의 문자들의 글꼴을 한꺼번에 설정할 수 있는 매크로도 제공함으로써 좀 더 간편한 방법도 가능하게 하였다. 다음의 것들이 이에 해당한다.

```
\latinmarks \hangulmarks \hanjamarks \prevfontmarks
```

여기서 이를테면 \hangulmarks는 알파벳, 숫자, 기타 모든 문장부호들을 전부 한글 폰트로 식자하라는 명령이니 나머지 명령들도 그 뜻이 명백할 것이다. 한편 이러한 유연한 글꼴 설정은 다음과 같이 보다 세련된 방식으로도 가능하다.

```
\xetexkofontregime
  [puncts=prevfont,hyphens=prevfont,colons=prevfont,cjksymbols=hanja]
  {latin}
```

이 명령은 표 1의 항목들 가운데 puncts, hyphens, colons에 해당하는 문자들은 직전 문자의 폰트를 따라가고 cjksymbols에 해당하는 문자들은 한자 폰트로 식자하며 나머지 문자들, 즉 alphs, nums, parens, quotes에 해당하는 문자들은 라틴 폰트로 찍으라는 의미이다. 다시 말해서 옵션의 키(key)는 표 1의 첫 번째 칼럼에 제시된 이름들 가운데 어느 하나이고 각 키의 값(value)은 latin, hangul, hanja, prevfont 가운데 어느 하나인 것이다. 인자에도 역시 이 네 가지 가운데 하나가 와야 한다. 만약 사례에서 옵션을 전부 생략하였다면 \latinmarks 명령을 내리는 것과 동일한 효과를 가져온다. 사용자는 취향에 따라 이렇게 xkeyval 패키지에 의존하는 방식으로도 유연한 글꼴 설정을 할 수 있는 것이다.

사용자가 아무런 설정도 하지 않았을 때는 XeTeX-k*o*의 기본값이 적용되는데, 기본값은 방금 \xetexkofontregime 명령의 예시에서 제시되었던 것과 동일하다. 즉 모두 라틴 폰트로 식자하되 puncts, hyphens, colons는 직전 폰트로, cjksymbols는 한자 폰트로 찍는 것이다.

지금까지 fontspec의 글꼴 설정 방식에 대응하여 한글 및 한자 폰트를 지시하는 방법과 이렇게 설정된 폰트를 문장부호 따위에 대해 유연하게 변경할 수 있는 방법에 대해 알아보았다. 이제 글꼴 설정에 대한 기본적인 이야기는 이 정도로 마치고 XeTeX-k*o*의 제작을 불러온 또 하나의 요인, 즉 글자 사이의 미세 간격 설정에 대하여 알아보기로 하자.

5 미세 간격 조정

전술한 바와 같이 XeTeX-k*o*의 한글 및 한자 폰트 지정은 fontspec 패키지가 제공하는 방식에 따르며 fontspec 매뉴얼에 나와있는 글꼴 옵션들을, 폰트가 이를 지원하는 한, 모두 사용할 수 있다. 그런데 이러한 fontspec 상의 글꼴 옵션 이외에 XeTeX-k*o* 자체에서 제공하는 글꼴 옵션들도 사용할 수 있으니 그것은 바로 미세 간격 조정에 관한 것들이다. 이를테면 한글 글자 사이에 마이너스 자간을 주기 위해서는

```
\setmainhangulfont[interhchar=-.04em]{UnBatang}
```

처럼 글꼴 옵션을 줄 수 있는데 이는 fontspec이 제공하는 것이 아니라 XeTeX-k*o*가 마련해 둔 옵션이다. 이렇게 하면 한글과 한글 사이에 -0.04em 만큼의 마이너스 자간이 글루로 삽입되어 줄바꿈이 허용되는 동시에 줄바꿈이 이루어지지 않는 한글 문자 사이의 간격은 미세하게 좁혀지게 되는 것이다. `interhchar` 옵션을 주지 않으면 0pt의 글루가 삽입되어 간격 조정 없이 단지 줄바꿈 허용만 이루어진다.

애초에 XeTeX-k*o*는 글꼴 옵션으로 자간을 조정하지 아니하고 그 대신 사용자로 하여금 `\xetexkointerhchar`라는 명령을 재정의하도록 함으로써 문서 전체의 한글 자간 조정이 가능하게 하였었다(플레인 텍에서 XeTeX-k*o*를 사용한다면 지금도 이러한 초기 방법으로 자간 조정을 할 수 있을 따름이다). 하지만 곰곰이 생각해보면 자간이란 폰트에 종속되는 성질을 가진다고 하지 않을 수 없다. 한글이 자면에 꽉 차게 디자인된 폰트를 사용하는 경우에는 마이너스 자간을 주지 않은 채 문서를 만들어야 할 터이고, 여백이 있게 글리프가 디자인된 폰트를 사용하는 경우에는 듬성듬성해 보이지 않도록 마이너스 자간을 주어 문서를 만드는 것이 마땅하다. 이렇게 자간이란 문자(character)의 속성이 아니라 폰트의 글리프(glyph)에 따르는 속성임을 깨닫게 되자 모든 한글에 대해 적용되는 `\xetexkointerhchar` 재정의 방법 대신에 특정 한글 폰트에만 적용되는 자간 조정을 구현할 필요가 대두되었고, 따라서 글꼴 옵션 기능을 제공하는 LaTeX 패키지인 fontspec의 매크로를 해킹하여 자간 조정을 위한 옵션을 추가하게 된 것이다. 이제 위 예시와 같이 은바탕 폰트의 옵션으로 한글 마이너스 자간을 지시하였다면 본문 중에 오직 은바탕 폰트가 사용되는 영역에 국한해서 한글 자간 조정이 작동하게 되었다.

이러한 글꼴 옵션을 받아들이는 명령들을 나열하면 다음과 같으며 모두 한글 및 한자 폰트를 설정하는 매크로들이다.

```
\setmainhangulfont        \setmainhanjafont
\setsanshangulfont        \setsanshanjafont
\newhangulfontfamily      \newhanjafontfamily
```

표 2. 자간 관련 주요 글꼴 옵션 및 그 기본값

이름	의미	기본값
interhchar	한글과 한글 사이	0pt
hu	한글 · 한자와 라틴 문자 사이	0.06em (라틴 괄호는 0.12em)
postmathskip	한글 · 한자와 인라인 수식 사이	hu 값의 두 배
quotewidth	라틴 따옴표의 폭	글리프의 폭(natural width)

```
\newhangulfontface          \newhanjafontface
\hangulfontspec             \hanjafontspec
\adhochangulfont            \adhochanjafont
\addhangulfontfeature(s)    \addhanjafontfeature(s)
```

당연하게도 `\setmainfont`처럼 fontspec 자체가 제공하는 명령에는 이러한 자간 조정 옵션을 사용할 수 없다. 또한 비록 한글 및 한자 폰트의 설정 명령이라 해도 고정폭 글꼴을 설정하는 명령에도 사용할 수 없음을 유의해야 한다. `\setmonohangulfont` 따위는 문자 그대로 고정폭 글꼴을 위한 것이고 고정폭 글꼴에 대해서는 한글만의 자간 조정이 비록 불가능은 아닐지라도 결코 바람직하지 않으므로 처음부터 구현 대상에서 제외하였다.

이렇게 한글 및 한자 글꼴 명령에 지시할 수 있는 자간 조정 옵션 가운데 가장 중요하다고 생각되는 것 몇 가지를 우선 표 2에 제시하였으니 표의 이름들이 옵션 키가 되고 여기에 사용자가 적절한 값을 부여하게 되는 것이다. 값은 반드시 길이(dimen) 값으로 써야 하므로 숫자 뒤에 pt, em, ex 따위의 단위를 붙여야 한다.

interhchar에 대해서는 굳이 자세한 설명을 요하지 않을 것이나 다만 이 자간이 한글과 한글 사이에서만 동작하고 한글과 한자, 한자와 한자 사이에서는 아무런 효력이 없음은 지적해 둘 필요가 있겠다. 한자는 일반적으로 획수가 많아 자면에 꽉 차게 디자인되므로 마이너스 자간이 오히려 가독성을 해칠 우려가 있기 때문이다. 즉, 한자 앞뒤에는—줄바꿈이 허용된다면—언제나 0pt의 글루가 들어갈 뿐이고 이것을 사용자가 쉽게 조절할 수 있는 옵션은 없다. 또한 여기서 한글과 한글 사이라 함은 완성된 현대 한글 음절 영역에 속하는 글자들 사이뿐만 아니라 옛한글 자모 조합으로 이루어지는 음절 사이도 지칭하는 것임을 밝혀두고자 한다.

두 번째의 hu라는 이름은 김강수 [1]에 의하여 처음 사용된 것으로서 hangul unit을 줄여 부른 말이다. 한국어 문서 조판에서 문장부호와 띄어쓰기 공백의 폭을 제어하기 위한 기본 단위로 제안되었으며 레거시 엔진 하의 예전 ko.TEX에서 채택되어 사용되고 있었다. 그 유산을 물려받은 XETEX-ko에서도 hu라는 이름을 채택하였으나 한글 조판의 기본 자간 단위라는 본래의 의미는 다소 퇴색되고 단지 한글 또는 한자와 라틴 문자 사이의 자간을 지칭하는 용어로 쓰이고 있다. 즉 한글 · 한자와 영문자 · 숫자 등이 만나면 그 사이에 hu 옵션으로 지시된 값만큼 간격이 삽입되는 것이다. 그러나 한글 · 한자와 라틴 괄호 · 따옴표 · 쌍점류(표 1의 parens, quotes, colons에 해당하는 것)가 만나면 hu 값의 두 배만큼 공백이 들어간다. 앞서 엠대시 전후의 간격을 사용자가 설정할 수 있다고 하였는데, 바로 hu 값을 재조정함으로써 가능하다. 한글 · 한자와 영문자 · 숫자 사이의 간격은 그대로 두고 굳이 괄호류 앞뒤의 간격만

hu = 0pt 한글English한글(漢字) $a^2 + b^2$은
hu = 0.06em 한글English한글(漢字) $a^2 + b^2$은
hu = 0.12em 한글English한글 (漢字) $a^2 + b^2$ 은
hu = 0.24em 한글 English 한글 (漢字) $a^2 + b^2$ 은

그림 2. hu 값 조절에 따른 자간 변화 예시. 한글과 괄호, 한글과 수식 사이에는 한글과 영문자 사이 간격의 두 배에 해당하는 간격이 들어간다.

조정하고자 한다면 글꼴 옵션에 의한 간편한 방법은 사용할 수 없고

```
\def\XKsmallskip{\hskip 0.125em plus 0.1ex minus 0.1ex}
```

처럼 매크로를 재정의해야 한다. 이런 것을 원한다면 xetexko-space.sty의 소스를 참고하라.

한편 hu 값은 인라인(inline) 수식과 한글 · 한자 사이 간격에도 영향을 미치게 되어 있어 hu의 두 배에 해당하는 간격이 삽입된다. 따라서 hu 값을 변경하면 인라인 수식과 한글 · 한자 사이 간격도 변경하는 셈이 된다. 라틴 문자와 한글 · 한자 사이 자간은 그대로 두고 인라인 수식과 한글 · 한자 간격만 조정하려면 글꼴 옵션으로 postmathskip을 따로 지시하면 된다. 괄호류에 대한 간격만 따로 조절하는 간편한 방법은 제공되지 않지만 수식만 특별 취급하여 따로 설정할 수 있도록 한 것이다.

quotewidth는 `\hangulquotes` 또는 `\hangulmarks`를 사용하는 경우에 발생할 수 있는 문제에 대비하기 위한 것이다. 한글 폰트에 들어있는 따옴표—악상그라브나 어포스트로피를 이용하는 경우를 말함—는 전통적으로 전각 폭을 가지고 있었으므로 지나치게 넓은 공백이 따옴표 앞뒤에 들어가는 것처럼 보일 수 있다. 이때 quotewidth를 글꼴 옵션으로 지시하면 따옴표가 그 값에 해당하는 만큼의 폭으로 강제 설정된다. 물론 최근에 출시되고 있는 한글 폰트의 따옴표는 전각이 아닌 좁은 폭인 경우가 대부분이므로 굳이 이 옵션을 줄 필요가 없을 것이다. 옵션을 설정하지 않으면 폰트가 가지고 있는 글리프 본래의 폭(natural width)으로 식자된다.

오래 전에 출시된 한글 폰트에는 따옴표뿐만 아니라 각종 한중일 문장부호들이 전각의 폭을 가지는 경우가 많다. 이를테면 고리점(。), 모점(、), 낫표(「 」), 인용표(《 》), 가운뎃점(·) 따위가 그러하니 이들 한중일 문장부호—아스키 문자의 글리프 대체나 리거처로서가 아니라 유니코드 문자로 직접 입력하는 따옴표(U+201C 따위)도 포함한다—를 만나면 XeTeX-k*o*는 강제로 그 폭을 0.5em으로 좁혀 식자하고 필요하다면 그 앞이나 뒤에 미세한 글루를 삽입한다. 이 경우 quotewidth와 같은 글꼴 옵션이 마련되어 있지 않으므로 굳이 따로 설정을 원

한글 “한글” 한글
한글“한글”한글

그림 3. `\hangulmarks` 하에서 어도비 명조에 `quotewidth=0.5em` 옵션을 주지 않았을 때(위)와 주었을 때(아래)

표 3. 온점 · 반점 · 물음표 · 느낌표의 위치 및 자간 옵션과 그 기본값

이름	의미	기본값
lowerperiod	한중일 문자 다음의 온점을 끌어내림	0pt
lowerquestion	한중일 문자 다음의 물음표를 끌어내림	0pt
lowerexclamation	한중일 문자 다음의 느낌표를 끌어내림	0pt
lowercomma	한중일 문자 다음의 반점을 끌어내림	0pt
preperiodkern	한중일 문자와 온점 사이 간격	0pt
prequestionkern	한중일 문자와 물음표 사이 간격	0pt
preexclamationkern	한중일 문자와 느낌표 사이 간격	0pt
precommakern	한중일 문자와 반점 사이 간격	0pt
postperiodkern	한중일 문자 다음의 온점 뒤 간격	0pt
postquestionkern	한중일 문자 다음의 물음표 뒤 간격	0pt
postexclamationkern	한중일 문자 다음의 느낌표 뒤 간격	0pt
postcommakern	한중일 문자 다음의 반점 뒤 간격	0pt
interpunctskern	한중일 문자 다음의 구두점 사이 간격	0pt

하는 사용자는 xetexko-space.sty의 소스를 참조하여 `\precjkopenparen` 따위의 매크로를 재정의하는 수밖에 없다. 다만 현대적인 한글 폰트에서는 이들 문장부호의 폭이 이미 반각 또는 그 이하로 들어가 있는 경우가 많으므로 오히려 0.5em을 강제하는 것이 부자연스러울 수 있다. 그런 때에는 단순히

```
\disablecjksymbolspacing
```

이라는 명령만 선언해두면 폰트가 가지고 있는 글리프의 자연적인 폭으로 식자된다. 이렇게 quotewidth만 따로 특별히 취급한 것은 매크로 수준에서 노드 레벨에 접근할 수 없는 XeTeX의 한계에 기인한다.

지금까지 자간을 제어할 수 있는 주요 옵션을 서술하였거니와 표 3은 XeTeX-k*o*가 제공하는 글꼴 옵션 가운데 나머지 것들을 보여주고 있다. 모두 라틴 마침표 · 물음표 · 느낌표 · 쉼표에 관한 옵션들로서 한글 · 한자 다음에 이들 문장부호가 왔을 때 필요하다면 그 전후 간격이나 세로 위치를 조절할 수 있게 하였다. 전술한 바와 같이 이들 문장부호는 원칙적으로 직전 문자의 폰트를 따라가도록 하였으므로 한글 · 한자와의 베이스 라인 부조화 문제는 이제 거의 발생할 여지가 없지만 그러한 염려가 완전히 불식되었다고 장담할 수는 없기에 lower… 옵션들을 여전히 유지하고 있다. 또한 일부 한글 폰트의 이들 문장부호가 비정상적인 앞뒤 간격을 가지고 있는 경우가 있어서 pre… 및 post… 옵션들을 최근 추가하였으나, 일반적으로는 이들 옵션을 사용할 일이 거의 없을 것이다.

마지막의 interpunctskern 옵션에 대해서는 조금 부연할 필요가 있겠다. 본디 맞춤법 규정에는 말줄임표를 "…" (U+2026)으로 입력하게 되어 있다. 그런데 컴퓨터 자판에서 이 특수문자를 입력하기가 번거로운 까닭에 요즘에는 간단히 온점 세 개를 잇따라 써서 말줄임표를 대신하는 경향이 많은 듯하다. LaTeX 사용자들은 `\ldots` 명령을 이용하기도 하지만

이를테면 웹에서 복사해 온 문서를 일일이 수정하기란 여간 귀찮은 일이 아닐 수 없다. 그런데 대부분의 폰트에서 온점의 폭이 매우 좁은 까닭에 세 개를 잇달아 썼을 때 온점들이 다닥다닥 붙은 결과를 보여주는 때가 많다. 이때 interpunctskern 옵션을 적용하면 구두점들을 그대로 이어 써도 마치 `\ldots`를 쓴 것처럼 지시한 간격을 그 사이에 삽입하게 되는 것이다. 다만 한글 · 한자 뒤의 구두점에만 적용되고 영문자 뒤에서는 동작하지 않음을 유의하면 될 것이다. 이는 표 3의 모든 옵션에 대해 동일하다.

6 옛한글의 구현

국내외 표준에 따르면 옛한글—1933년 한글 맞춤법 통일안이 제정되기 이전의 한글 문서를 의미하지만 현대에도 이를테면 제주도 방언을 표현하는 경우 따위에 여전히 필요하다—은 유니코드의 조합 자모 영역(U+1100부터 U+11FF, 그리고 U+D7B0부터 U+D7FB까지)을 사용하여 이른바 "첫가끝 코드"로 입력되어야 한다. 그런데 우리나라에서 널리 쓰이는 워드프로세서들이 애초 이러한 표준 대신에 유니코드 기본 다국어 평면(BMP, Basic Multilingual Plane)[6]의 개인 사용 영역(PUA, Private Use Area)에 자주 쓰이는 옛한글 음절들을 완성된 형태로 임의 할당한,[7] 이른바 "한양 PUA 코드"라고 알려진 비표준 코드만 지원함으로써 오히려 표준보다 널리 사용되게 되었고 결과적으로 옛한글 처리에 큰 혼란을 야기하였다 [2].

이러한 혼란을 지양하기 위하여 XeTeX-k*o*는 비표준의 한양 PUA 코드 입력은 원칙적으로 지원하지 않고 있다. 물론 사용자가 개인 사용 영역의 문자들을 한글 클래스로 할당함으로써 한양 PUA 코드로 입력된 문서도 그리 어렵지 않게 조판할 수 있는 길이 있지만 어디까지나 비표준이고 권장할 것도 못되기 때문에 처음부터 고려 대상에서 제외하였다. 다시 말해서 XeTeX-k*o*에서는 옛한글 문서를 반드시 첫가끝 코드로 입력해야만 정상적인 처리가 보장된다. 한양 PUA 코드를 첫가끝 코드로 변환하는 유틸리티도 k*o*.TeX 컬렉션에 포함되어 있으므로[8] 이것이 큰 문제는 아니라고 할 것이다.

그러면 첫가끝 코드의 옛한글 문서를 식자하기 위한 올바른 폰트 설정 방법은 무엇인가? 자모 코드 입력으로부터 품위 있는 결과물을 얻기 위해서는 한글 폰트가 옛한글과 관련한 몇 가지 오픈타입 속성들(OpenType layout features)을 지원해야 한다. 이 조건이 충족된다면 다음과 같은 글꼴 설정으로 옛한글을 조판할 수 있다.

```
\setmainhangulfont
    [Script=Hangul, Language=Korean, YetHangul=On]
    {UnBatang}
```

XeTeX 엔진은 오픈타입 속성들을 대부분 잘 지원하므로 단지 당해 속성들을 사용하라는 지시를 엔진에 전달해주기만 하면 된다. 여기서 `YetHangul=On/Off`가 바로 그러한 기능을 수행하는, XeTeX-k*o*의 또다른 글꼴 옵션인 것이다. `On` 값은 오픈타입 속성 가운데 ccmp, ljmo,

6. U+0000부터 U+FFFF까지의 65,536개의 문자집합을 일컫는 말이다
7. 5,299자에 이르는 완성형 옛한글 글자들이 U+E0BC부터 U+F66E까지 차지하고 있다.
8. hypua2jamo라는 이름의 스크립트를 실행하면 된다.

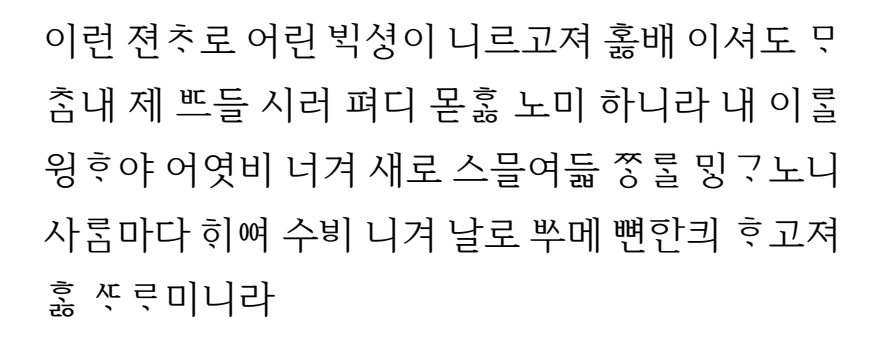

그림 4. 함초롬 바탕체를 이용한 옛한글 조판 예시. 한국텍학회가 GSUB 테이블 따위를 집어넣어 수정한 LVT 계열 폰트를 이용하였다.

vjmo, tjmo를 한꺼번에 활성화하는 일을 하고 `Off` 값은 ljmo, vjmo, tjmo를 비활성화하는 일을 한다. 오픈타입 사양 중에 ljmo, vjmo, tjmo는 오로지 옛한글만 위한 것들이지만 ccmp는 한글 이외에 다른 언어에서도 많이 이용되는 속성이므로 비록 사용자가 `YetHangul=Off`를 지시했더라도 ccmp만은 끄지 않고 그대로 유지하게 하였다.[9] 그러나 통상적으로 `Off`를 선언해야 할 경우를 상상하기란 쉽지 않다. 옛한글 조판 원리를 설명하는 학술문서 정도를 생각할 수 있을 것이다. 어쨌든 폰트만 갖추어져 있다면 XeTeX에서 옛한글을 조판하기란 무척 간단한 일이다.

문제는 이러한 속성을 지원하는 폰트가 아직은 그다지 흔치 않다는 데 있다. 마이크로소프트 오피스 플러스팩에 들어있는 옛한글 글꼴 4종이 그러한 지원을 하고 있으나 방점 글리프도 없을 뿐만 아니라 무엇보다 상용글꼴이라는 장벽이 가로막고 있다. GPL 라이선스를 따르는 공개폰트인 은바탕에도 이러한 속성이 들어있지만, 작동 실험용의 성격이 강해 그 품질이 도저히 출판물에 사용할 수 없는 수준이다. 그런데 최근 한글과컴퓨터사가 공개한 함초롬바탕 · 돋움 글꼴에 한국텍학회가 옛한글 조판을 위한 오픈타입 속성들을 집어넣어 질 좋은 옛한글 GSUB (glyph substitution) 지원 글꼴 두 세트가 탄생하였으니, 자유로이 쓸 수 있는 무료 폰트인데다 품질도 우수해서 옛한글 조판의 큰 획을 긋는 역사적 사건으로 평가되고 있다. 그러나 이렇게 수정된 함초롬 폰트를 제3자에게 자유로이 배포할 수 있느냐의 문제가 라이선스 상 아주 명확한 것은 아니어서 아직 대외적으로 글꼴 자체를 배포하지는 못하고 있는 형편이다.[10] 요컨대 완전히 자유로우면서도 품위 있는 옛한글 글꼴이 현재로서는 존재하지 않는다고 하지 않을 수 없다.

이러한 곤경에 대처하기 위하여 XeTeX-k*o*는 한양 PUA 코드를 입력 차원이 아니라 출력 차원에서 지원하기로 결정하였다. 입력은 첫가끝 코드로만 이루어져야 하지만 출력은 한양

9. 오픈타입 속성의 스펙에 대해서는 `http://www.microsoft.com/Typography/OTSpec/featurelist.htm`을 볼 것. 폰트 제작자는 옛한글 속성을 ljmo, vjmo, tjmo라는 이름으로만 구현해야 하는 것은 아니며 필요하다면 얼마든지 다른 이름을 쓸 수도 있다. 게다가 오픈타입 스펙의 ljmo, vjmo, tjmo의 설명 중에 잘못된 점도 있으니 이들은 사실 문맥에 따른 글리프 대체(contextual glyph substitution)인데도 자소 조합 리거처로 서술해 놓은 것이다. 그럼에도 불구하고 이 이름으로 옛한글 속성을 구현하는 것이 그간의 관례였고 이 관례에서 벗어나는 것은 다양한 프로그램 간의 호환성 문제를 야기할 수 있음을 주의할 필요가 있다.

10. `http://www.ktug.or.kr/xe/?mid=KTUG_open_board&document_srl=7947`

PUA 코드를 지원하는 글꼴을 이용할 수 있도록 하였거니와 옛한글 GSUB 테이블을 가진 글꼴은 구하기 어렵지만 PUA 영역에 옛한글 음절을 가진 글꼴은 비교적 수월하게 찾을 수 있기 때문이다. 즉,

```
\setmainhangulfont[Mapping=jamo-pua]{New Batang}
```

이와 같이 통상적으로 쓰는 `Mapping=tex-text` 대신 `Mapping=jamo-pua`를 글꼴 옵션에 사용하면 조합 자모로 입력된 옛한글 텍스트를 PUA 옛한글 완성형 음절로 변환하여 출력해 준다. XeTeX에는 teckit 라이브러리가 포함되어 있어 입력 문자열을 맵핑 테이블에 따라 실시간으로 변환할 수 있으며 또한 한글텍사용자그룹(KTUG)에 의하여 성공적으로 수행된 "한양 PUA 테이블 프로젝트"의 성과물이 이미 있으므로,[11] 결심만 하면 지원 자체는 그리 어려운 일이 아니었다.

그런데 jamo-pua 맵핑에 의한 옛한글 조판 지원은 두 가지 단점을 안고 있다. 첫째, 이렇게 만들어진 결과물 PDF에서 텍스트를 추출하거나 복사·붙여넣기를 하면 원래의 입력 코드인 첫가끝은 사라지고 한양 PUA 코드만 남는 문제이다. 이것은 매크로 차원에서 어떻게 손 써볼 수 없는 문제인데, 옛한글 처리의 혼란을 극복하고자 하는 우리의 희망과는 정면으로 배치되는 결과이다.

둘째, 더욱 심각한 것은 jamo-pua 맵핑을 이용했을 때 옛한글에 대한 자동조사 기능이 동작하지 않는다는 문제이다. 일단 XeTeX 엔진의 버그라고 판단되는데, jamo-pua 맵핑과 자동조사 기능을 동시에 켜면 엔진 자체가 에러 메시지도 없이 죽어버리는 것이다. 이 문제는 XeTeX 개발자에게 이미 보고되었으나 빠른 시일 내에 해결될 것 같지는 않다. 결국 어쩔 수 없이 옛한글에 국한해서 자동조사 기능을 꺼 두었지만 현대한글, 한자, 영문자에 대해서는 아무 탈 없이 동작하므로 안심해도 좋다. 그런데 이 문제는 오픈타입 속성을 이용하는 옛한글 조판에서는 발생하지 않는다. 따라서 옛한글 자모에 대해 자동조사를 죽여둔 것은 오직 옛한글 오픈타입 속성을 지원하는 글꼴이 부족하다는 데 따른 현실과의 타협, 즉 jamo-pua 맵핑 지원 때문이며 장차 올바른 옛한글 글꼴이 널리 사용되는 날이 오면 jamo-pua 맵핑 기능은 XeTeX-k*o*에서 삭제되고 자동조사 기능은 온전하게 다시 부활할 것이다. 지금도 완전한 자동조사 기능을 사용 못하는 것은 아니다. 오픈타입 속성을 이용하여 옛한글을 조판함과 더불어 옛한글에 대한 자동조사 기능도 필요한 사용자는 전처리부 따위에

```
\enablejamoautojosa
```

라는 명령을 선언해두기만 하면 된다. 물론 `Mapping=jamo-pua` 옵션은 이 선언을 포함한 문서에서는 결코 등장해서는 안 된다. 구체제(ancien régime)의 유물인 한양 PUA 코드가 완전히 사라지고 표준적인 옛한글 처리만으로 만족스럽게 문서 작성을 할 수 있는 날이 어서 도래하기를 희망한다.

11. `http://faq.ktug.or.kr/faq/HanyangPuaTableProject`

国際連合憲章は、加盟国に「人権の普遍的な尊重及び遵守」を促進してこれを達成するために「共同及び個別の行動」をとる義務を課している。世界人権宣言は、法的拘束力はないものの、「すべての人民とすべての国とが達成すべき共通の基準として」が 1948 年に国際連合総会において採択された。総会は定期的に人権問題を取り上げている。総会の補助機関である人権理事会は、主に調査と技術的な支援を通じて人権の推進を直接担当する。国際連合人権高等弁務官は、国際連合の全ての人権に関する活動を担当する。

그림 5. 일본어 환경을 이용한 문단 예시. 일본어 위키백과의 "국제연합" 항목에서 따왔다.

7 나머지 주요 기능들

지금까지 XeTeX-k*o*의 핵심적 기능을 소개하고 그 사용법을 비교적 상세히 설명하였으니 대체로 글꼴 설정과 글꼴 옵션에 관련한 것들이었다. 이제부터는 아직 소개하지 못한 XeTeX-k*o*의 기능 가운데 주요한 몇 가지를 장황한 설명은 생략한 채 간단히 소개하는 방식을 취하고자 한다. 지금까지 지겹도록 폰트 관련 이야기를 하였지만 아직도 폰트 이야기는 끝나지 않았다. TeX이라는 프로그램 자체가 원래 폰트의존적인 성격이 강하지만 XeTeX은 정도가 더욱 심해서 폰트를 빼놓고는 도무지 그 존재근거(raison d'être)가 사라지게 되는 것이다.

7.1 일본어 · 고문헌

현대 한국어는 라틴 문자 코드로 문장부호를 쓰고 또 띄어쓰기도 한다. 이를테면 마침표는 U+002E, 괄호는 U+0028, 이런 식이다. 하지만 일본어 · 중국어는 띄어쓰기가 없으며 마침표로는 U+3002의 고리점을, 괄호는 U+FF08의 코드를 사용한다. 라틴 문자를 알지 못했던 우리의 옛 문헌에서도 사정은 마찬가지이다.

문제는 폰트에 이들 한중일 문장부호들이 전통적으로 전각(full-width)으로 디자인되어 있다는 것이다. 일본어나 고문헌에서는 이것이 보통은 문제가 되지 않지만 문장부호가 줄 끝에 걸릴 때에는 과다한 공백을 남기고 줄바꿈이 되어 문단의 가지런한 정렬을 방해하는 요소가 된다. 따라서 일본어 조판에서는 이들 전각기호들을 강제로 반각(0.5em)으로 식자하고 나서 마침표나 닫는 괄호 따위라면 그 직후에, 여는 괄호류라면 그 직전에 반각 크기의 글루를 삽입한다. 글루에서 줄바꿈이 이루어지면 해당 글루는 저절로 사라져버리므로,[12] 문장부호의 폭이 줄 가운데에서는 마치 전각인 것처럼 보이고 줄 끝에서는 반각인 것처럼 보이게 되어 가지런하게 문단이 정렬되는 것이다[5].

XeTeX-k*o*의 일본어 환경은 이러한 조판 기법을 그대로 차용하였다. 이를테면,

```
\hanjafontspec[Script=Kana]{Hiragino Mincho Pro}
\begin{japanese} ... \end{japanese}
```

12. 이른바 "버려질 수 있는 아이템"(discardable items)에는 글루 외에도 컨, 페널티가 있다[4, 110쪽].

이렇게 일본어 폰트를 한자 글꼴로 채택하고 japanese 환경을 선언하면 대체로 무난하게 일본어 조판이 행해진다. 우리의 고문헌을 조판할 때에도 폰트만 적절히 바꾸어서 japanese 환경을 이용하면 된다. 이 환경이 제공하는 기능 중에는 한중일 문자 전후에 의도하지 않은 띄어쓰기가 왔을 때—아직 불완전하지만—이를 저절로 없애주는 기능, 라틴 문자와 한중일 문자 사이에 4분각(0.25em)의 자간—그림 5에서는 연도 앞뒤의 공백—을 넣어주는 기능, 고정폭 문자로 이루어지는 일본어 · 고문헌의 특성을 반영하여 판면의 너비를 전각의 정수배로 자동으로 지정하는 기능, 그리고 들어쓰기의 간격을 1전각으로 맞추는 기능 따위가 있다. XeTeX-k*o*는 chinese 환경도 제공하고 있지만, 중국어 조판 규칙에 대해서는 자세히 아는 바가 없는 탓에 다른 점은 모두 japanese 환경과 동일하게 하고 다만 들여쓰기 폭을 2전각으로 만들어 두었을 따름이다.

XeTeX-k*o*의 japanese/chinese 환경은 사실 주로 우리의 고문헌 조판을 염두에 둔 것인지라, 본격적인 일본어 또는 본격적인 중국어 문서를 조판하고자 한다면 pTeX이나 xeCJK 패키지를 고려해야 할 것이다.[13] 하지만 한국어 문서 중간에 등장하는 일본어 · 중국어 문단의 조판 정도는 XeTeX-k*o*만으로도 대부분 그 수요를 충족시킬 수 있다고 본다.

일본어 환경과 관련하여 소개해 두어야 할 명령으로 `\inhibitglue`가 있다. 드물기는 하지만 가끔 XeTeX-k*o*가 자동 삽입하는 자간이 필요 이상으로 넓게 들어가는 경우가 있다. 이때 다음과 같이 `\inhibitglue` 명령을 삽입하면 해당 지점의 자간이 0pt로 강제된다.

```
あっ！\inhibitglue と驚く
```

여기서 `\inhibitglue`가 없다면 XeTeX-k*o*는 전각 느낌표를 문장 종지 부호로 인식하고 그 뒤에 넓은 공백을 삽입하는데(あっ！ と驚く) 이 명령을 넣어줌으로써 이를 교정할 수 있게 된다(あっ！と驚く). 텍이 문장의 의미 맥락까지 파악하기는 어려우므로 이런 때에는 사용자의 손길을 기다릴 수밖에 없다.

7.2 세로쓰기

1980년대까지만 해도 일상에서 흔히 볼 수 있던 세로쓰기 조판이 이제는 우리 주변에서 찾아보기 힘들게 되어 버렸다. 하지만 여전히 부분적으로는 세로쓰기의 필요성이 남아 있다고 하겠으니 옛 맛을 살리기 위한 특수 목적의 조판이 있을 수 있고 무엇보다 지금도 책등(spine)에서는 세로쓰기가 자주 발견된다. XeTeX은 오픈타입 속성들을 잘 지원하고 있고 오픈타입 속성 중에는 vert 또는 vrt2라는 이름의 세로쓰기 관련 속성이 있으므로,[14] XeTeX을 이용하면 그리고 폰트가 세로쓰기에 관련된 테이블을 가지고 있다면,[15] 그리 어렵지 않게 세로쓰기 조판을 구현할 수 있다.

13. xeCJK는 이미 TeX Live 2009에 포함되었고 pTeX은 올해 출시될 TeX Live 2010에 들어갈 예정이다.
14. vrt2는 vert보다 진보된 속성이고 vrt2가 오픈타입 속성으로 등재되면서 vert는 vrt2에 의하여 대체되었다. 따라서 새로 폰트를 만들 때 vert 대신에 vrt2 속성으로 세로쓰기 지원을 해야 하며 vert는 오래된 옛날 프로그램과의 호환성을 위해서 예외적으로만 사용되어야 한다.
15. 세로쓰기 관련 폰트 테이블 가운데 핵심적인 것은 vhea/vmtx 테이블과 vert/vrt2 속성을 포함하는 GSUB 테이블이다. 오픈타입 폰트의 테이블 스펙에 대해서는 `http://www.microsoft.com/typography/otspec/otff.htm`을 볼 것.

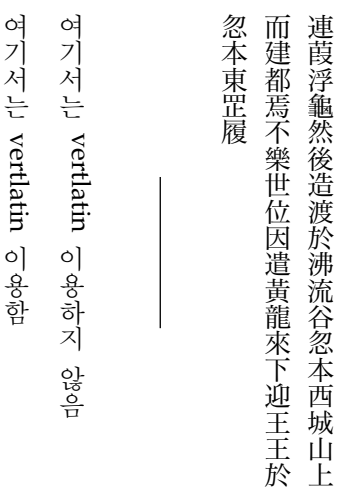

그림 6. 몇 가지 세로쓰기 예시. 오른쪽은 광개토왕릉비의 시작 부분이다. 왼쪽에서는 `\vertlatin`을 사용하여 영문자와 한글의 베이스 라인에 조화를 줄 수 있음을 보였다.

이에 따라 XeTeX-k*o*도 세로쓰기 조판을 지원하고 있다. 하지만 문서 전체를 세로쓰기 하는 경우를 위해서는 따로 매크로를 제공하지 않고 있으니 `\setmainhangulfont` 따위에 `Vertical=RotatedGlyphs` 옵션을 주는 것으로 충분하기 때문이다. 그 대신 XeTeX-k*o*는 문서의 일부만 세로쓰기 할 수 있도록 vertical 환경을 마련해 두었다.[16]

```
\newfontfamily\vertfont
    [Script=Hangul, Vertical=RotatedGlyphs] {Adobe Myungjo Std}
\begin{vertical}{\vertfont}{18em}
...
\end{vertical}
```

vertical 환경은 인자 두 개를 요구한다. 첫째는 세로쓰기에 사용될 폰트 명령인데 미리 `\newfontfamily` 명령으로 설정되어 있어야 한다. 세로쓰기에서는 자간 조정은 전혀 무의미하므로 `\newhangulfontfamily` 따위가 아니라 `\newfontfamily`를 이용하도록 하였다. 그리고 이렇게 지시된 글꼴은 한글 뿐만 아니라 한자를 식자하는 데에도 쓰이도록 해 두었다.[17] 나아가 폰트 가운데는 영문자에 대해서도 세로쓰기 글리프를 따로 정의해 둔 고급 폰트가 있으므로(어도비 명조 글꼴이 그 예이다) XeTeX-k*o*는 기본적으로 라틴 문자들까지도 모두 첫 번째 인자로 지시된 폰트로 식자하도록 하였다. vertical 환경 안에서 영문자를 인자로 지시된 폰트가 아니라 `\setmainfont` 따위의 기본 라틴 폰트로 식자하고자 한다면 해

16. 문서 일부에 대한 세로쓰기 환경은 xetexko-vertical.sty 파일이 제공하고 있다.

17. 사실 vertical 환경의 사용자 인터페이스는 `\newhangulfontfamily` 따위의 매크로가 XeTeX-k*o*에 도입되기 이전에 만들어진 것이고 LuaTeX-k*o*와의 일관성을 감안한 것이어서 다소 불편한 면이 없지 않다. 지금 와서 다시 vertical 환경의 인터페이스를 구상하라면 세로 길이를 지시하는 인자 하나만 받아들이게 하고 글꼴 지정은 사용자가 환경 내에서 fontspec 패키지의 명령에 대응되는 각종 한글 · 한자 글꼴 설정 매크로를 이용하여 이를 자유로이 행할 수 있도록 만들고 싶다. 하지만 이미 기존의 vertical 환경 인터페이스가 이용되고 있는 문서가 있는 까닭에 호환성을 고려하여 그대로 두기로 하였다. 앞으로 사용자의 요청이 있다면 이를 수정할 뜻이 있음을 알려둔다.

당 문자열을 \vertlatin 명령의 인자로 사용하여야 한다. 그렇지 않고 만약 \latinalphs 따위를 선언하여 영문자를 라틴 문자로 찍게 되면 영문자의 베이스 라인과 한중일 문자의 베이스 라인이 조화되지 못하는 문제가 생긴다. 세로쓰기의 베이스 라인은 한중일 문자의 폭을 세로로 양분하는 선이 되지만, 영문자에게는 이 베이스 라인이 한쪽으로 너무 치우친 결과가 된다. 따라서 영문자들의 위치를 조절해 줄 필요가 생기는데, \vertlatin이 바로 이를 위한 명령이다.

vertical 환경의 두 번째 인자는 세로쓰기로 식자될 박스의 수직 높이를 길이(dimen) 값으로 요구한다. 가로쓰기가 기본인 문서에서 일부 문단만 세로쓰기로 조판하는 것이므로 수직 높이를 텍이 자동으로 알 수가 없고 따라서 사용자가 따로 지시해주어야 하는 것이다. 사실 vertical 환경이 하는 일은 비교적 단순한데, 박스 속에 세로쓰기 문단을 조판한 다음, 박스를 $-90°$ 회전시키는 것이다. 따라서 사용자가 지시한 수직 높이는 텍에게는 문단의 너비가 되는 셈이니 대체로 전각의 정수배로 이 값을 지정해 주면 무난하다.

7.3 수식 한글

레거시 엔진을 지원하는 예전 k*o*TEX 하에서는 원칙적으로 수식 안에 한글을 쓸 수 없었고 오로지 \hbox 명령의 인자로만 우회적으로 사용할 수 있을 뿐이었다. 텍에서는 수식 폰트와 일반 텍스트 폰트는 전혀 달리 취급되기 때문에 수식 안에서 \hbox를 쓴다는 것은 수식 모드(math mode)에서 텍스트 모드(정확히는 제한적 수평 모드 restricted horizontal mode)로 잠시 옮겨가서 텍스트 폰트로 글자를 식자하라는 의미가 된다 [4, 제13장]. 따라서 수식 안에 가공하지 않은 한글을 그대로 쓰기 위해서는 한글 폰트를 수식 폰트로 따로 설정해줄 필요가 있는 것이다. 하지만 8비트의 한계에 갇혀있는 레거시 엔진 아래에서는 이 작업이 거의 불가능한 것이었기에 어쩔 수 없이 \hbox를 이용하는 불편을 감수할 수밖에 없었다.

그런데 유니코드를 완전히 지원하는 XETEX 엔진에서는 트루타입 · 오픈타입 폰트를 비교적 간편하게 수식 폰트로 설정할 수 있게 되었으니 XETEX은 이와 관련된 몇 가지 편리한 원시명령들을 제공하고 있고 [8], XETEX-k*o*는 이를 이용하여 한글 폰트를 수식 폰트로 간단하게 지시할 수 있는 사용자 매크로를 마련하였다. 따라서 가령

```
\setmathhangulfont{YoonMyungjo Light}
```

이러한 선언을 전처리부에 두면 이후로는 수식 내의 한글이 \hbox에 의지하지 않고도 윤명조체로 식자된다. 이러한 선언이 없다면 수식 한글의 기본값인 은바탕으로 찍힌다. 이 때 fontspec 패키지가 제공하는 글꼴 옵션을 이용할 수 있음은 물론이나, 자간을 설정하는 XETEX-k*o*의 글꼴 옵션은 쓸 수 없다.

현재 수식 내에 \hbox 없이 쓸 수 있는 한국어 문자는 한글 음절 영역(U+AC00부터 U+D7A3까지)에 한정해 두었다. 한국어 문서에서 한글 이외에 한자 따위를 수식에 사용할 일이 거의 없을 것으로 판단하였기 때문이다. 굳이 한자를 써야 할 일이 있다면 사용자는

```
\setmathhangulblock{4E00}{9FC3}
```

```
민사소송의 변호사 선임률을 구하는 공식은 다음과 같다.
$$\frac{원고선임건수 + 피고선임건수 + 쌍방선임건수 \times 2}
      {총처리건수 \times 2}$$
```

민사소송의 변호사 선임률을 구하는 공식은 다음과 같다.

$$\frac{원고선임건수 + 피고선임건수 + 쌍방선임건수 \times 2}{총처리건수 \times 2}$$

그림 7. 수식 한글을 이용하는 예시. 한글을 \hbox 안에 집어넣을 필요가 없다.

이처럼 유니코드 한자 영역까지도 \setmathhangulfont 명령으로 지시한 폰트로 식자하겠다고 명시해 주어야 한다. 인자 두 개는 유니코드에서 한중일 공통 한자 블럭이 시작하는 위치의 문자번호와 끝나는 위치의 문자번호를 16진수로 나타낸 것이다. 반드시 대문자로만 써야 한다. 그밖의 한자 영역뿐만 아니라 기타 기호 영역도 마찬가지 방법으로 지시할 수 있다. 여전히 예전처럼 \hbox를 써도 되고 그것이 크게 불편하지는 않지만, 그래도 이렇게 단순한 설정만으로 수식 내에서 한중일 문자를 편리하게 표현할 수 있는 길이 열렸으니 초심자들이 혼란스러워할 사항이 하나 줄어든 셈이다.

7.4 드러냄표

가로쓰기가 대세인 현대적인 문서에서는 강조의 의미로 폰트를 변경하거나 밑줄을 긋는 것이 일반적인 경향이다. 텍에서 폰트 변경은 아무런 문제가 되지 않으며 밑줄을 긋는 것도 ulem 패키지 따위의 도움을 받아 수월하게 표현할 수 있다. 물론 ulem 패키지를 한국어 조판 구현 환경에 맞추어 에러 없이 작동되게 위해서는 약간의 노력이 필요하지만, 이미 XeTeX-k*o*는 ulem에 대한 패치를 마련하였고 사용자가 ulem 패키지를 로드하면 자동적으로 이를 인식하여 패치가 작동되도록 해두었으므로 사용자 입장에서는 걱정할 거리가 없다.[18]

그런데 전통적으로 한국어 조판에서는 강조의 의미로 드러냄표를 사용해왔으며 오늘날에도 드물지 않게 이용되는 것을 볼 수 있다. 드러냄표는 강조할 글자 위 가운데에 상점(˙)이나 상고리점(˚)을 찍는 형태로 표현한다. 이러한 드러냄표 강조를 XeTeX-k*o*도 지원하고 있는데, 예전 k*o*.TEX의 드러냄표 구현과 아이디어는 동일하지만 엔진이 달라진 탓에 실제 코드는 완전히 다르고 훨씬 복잡해졌다.[19] 하지만 사용자 인터페이스는 달라진 것이 없으니 여전히 예전처럼

```
\dotemph{드러냄표로 강조}
```

이렇게 \dotemph 명령을 이용하여 드러냄표로 강조할 수 있다. 기본값으로 상점을 찍어 강조하게 하였지만 사용자가 상고리점 따위로 설정을 바꿀 수가 있다.

18. ulem 패키지는 플레인 텍에서도 사용할 수 있다. 하지만 플레인 텍 사용에서는 XeTeX-k*o*의 자동 패치 적용이 이루어지지 않으므로 사용자는 손수 \xetexkoulemsupport 명령을 선언해 준 뒤에만 밑줄을 그을 수 있다.
19. 드러냄표를 구현하는 코드는 xetexko-dotemph.sty 파일에 담겨 있다.

```
\def\dotemphchar{˚}
\def\dotemphraise{0.4em}
\dotemph{상고리점}
```

즉, \dotemphchar 명령을 재정의하여 드러냄표를 다른 모양으로 바꿀 수 있으며, 또한 \dotemphraise 명령을 재정의하여 드러냄표의 세로 위치를 아래 또는 위로 조절할 수 있다. 기본값은 0.4em이니, 특수한 모양의 글자를 이용하지 않는 한 대체로 이 값을 그대로 사용해도 무난한 결과를 얻을 수 있을 것이다.

\dotemphchar를 재정의할 때 드러냄표 문자를 직접 입력하기가 곤란하다면, ^^^^02da 또는 \char"02DA처럼 입력해도 좋음은 두말한 나위도 없다. 다만 전자의 방법을 이용할 때는 유니코드 문자번호(unicode code point)를 16진수 소문자로만 써야 하고 후자의 방법을 이용할 때는 대문자로만 써야 함을 주의해야 할 것이다. 후자의 경우에는 16진수뿐만 아니라 10진수(\char730)나 8진수(\char'1332)로도 표현할 수 있다[4, 제8장].

7.5 매달린 구두점

XeTeX-k*o*는 재미있는 기능 한 가지를 제공하고 있으니 그것은 이른바 '매달린 구두점'(hanging punctuation)에 관한 것이다. 행 끝에 마침표 · 쉼표 · 따옴표 따위의 문장부호가 오는 경우, 이들 글리프는 바운딩 박스가 매우 작은 대신 여백이 커서 문단의 정렬이 가지런히 이루어지지 않는 듯한 느낌을 줄 때가 있다. 그때 이들 구두점을 판면 바깥으로 빼내서 식자함으로써 가지런한 정렬이 이루어진 것처럼 보이게 하는 기술을 매달린 구두점이라 부른다[3].

이러한 매달린 구두점은 pdfTeX에서 제공하기 시작한 마이크로타입(microtype) 기술을 통하여 구현할 수 있다. pdfTeX은 글자 늘이기(character expansion)과 글자 내밀기(character protrusion, margin kerning이라고도 한다)라는 획기적인 신기술을 텍에 도입하였다. 전통적인 텍이 글루를 조금 늘이거나 줄이는 방법만으로 문단 정렬을 수행하였다면 pdfTeX의 마이크로타입 기술 가운데 전자, 즉 글자 늘이기는 글리프까지도 눈에 띄지 않을 정도로 미세하게 좌우로 늘이거나 줄임으로써 더욱 가지런한 판면과 음영을 달성할 수 있게 하였다. 후자의 글자 내밀기는 판면의 좌우 양끝에 걸린 글리프들을 정해진 설정값에 따라 조금씩 판면 바깥으로 내밀어 식자함으로써 역시 가지런한 정렬에 도움을 주는 기술이다. 이들 신기술 가운데 후자의 글자 내밀기가 매달린 구두점과 관련되는 것으로서 특정 문장부호에 내밀기 값을 1000으로 설정하면 우리가 말하는 매달린 구두점과 동일한 효과를 얻을 수 있다. 이들 신기술을 수월하게 사용할 수 있도록 도와주는 패키지가 microtype인데, 그러나 당연하게도 서양에서 만들어진 microtype 패키지의 내밀기 기본값은 한중일 삼국에서 전통적으로 구현했던 매달린 구두점을 염두에 두지 않고 정해놓은 것이다. 따라서 마침표 따위의 폭의 일부가 아닌 전부가 행 끝에 매달리는 '매달린 구두점'을 구현하려면 사용자가 따로 설정값을 수정해 주어야 한다.

어쨌든 pdfTeX 또는 이를 계승한 LuaTeX에서는 이렇게 글자 내밀기 기술을 이용하여 매달린 구두점을 구현할 수 있다. 하지만 XeTeX은 적어도 현재까지는 글자 늘이기나 글자 내밀기와 같은 마이크로타입 기술을 채택하지 않고 있거니와, 다만 최근에 들어서 이들 가운데

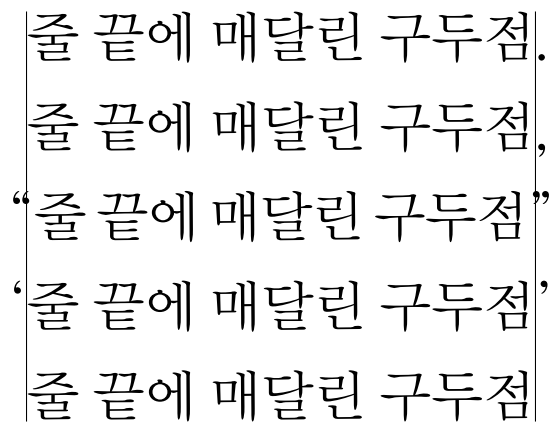

그림 8. 매달린 구두점의 예시. 온점, 반점, 홑따옴표와는 달리 겹따옴표는 글자폭의 일부만이 판면 바깥에 식자된다.

후자만 지원하는 실험적인 코드가 작성되어 테스트 중에 있다고 한다.[20] 이렇게 아직 XeTeX이 글자 내밀기 기술을 정식으로 지원하지 않고 있는 까닭에, 그리고 이를 지원하는 실험적 코드가 작성되기 훨씬 이전부터 XeTeX-k*o*는 일부 문자에 대하여 매달린 구두점 기능을 지원해왔던 까닭에, XeTeX-k*o*의 구현은 글자 내밀기 기술이 아닌 전혀 다른 방법에 기초한 것이 되지 않을 수 없었다. 여기에서 기술적인 내용을 상론할 여유도 필요도 없을 터이니,[21] 사용자 인터페이스에 대해서만 서술하기로 하겠다. 다만 XeTeX의 소스코드 저장소에 올려져 현재 테스트 중인 코드가 장래 정식으로 채택되어 출시된다면 매달린 구두점을 위한 XeTeX-k*o*의 인터페이스도 완전히 달라질 가능성이 있다는 것만 지적해두고 싶다.

현재로서 매달린 구두점 기능을 사용하기 위해서는 xetexko 패키지만 불러들이는 것으로는 부족하고 전처리부에서 따로 스타일 파일 하나를 로드해 주어야 한다.

```
\usepackage{xetexko-hanging}
\hangingpunctuation
```

즉, xetexko-hanging.sty 파일을 로드한 후 `\hangingpunctuation` 명령을 내려주어야 매달린 구두점이 동작을 시작한다. 다만 이 명령은 글꼴 설정이 완료된 후에 내려져야지 그렇지 않으면 매달린 구두점이 정확하게 작동하지 않을 수 있다는 점은 지적해 둘 필요가 있겠다. 어쨌든 따로 파일을 불러들이는 등 사용자 인터페이스를 번거롭게 해둔 이유는 이 기능이 얼마나 널리 사용될지 의문이 있기 때문이고, 또한 무엇보다 장차 언제 바뀔지 모르는 임시 코드의 성격이 크기 때문이다.

이 명령에 의하여 판면 바깥에 식자되는 구두점에는 온점(U+002E), 반점(U+002C), 그리고 어포스트로피나 악상그라브에 의해 식자되는 라틴 따옴표(‘ ’, “ ”)만 포함된다. 온점, 반점, 홑따옴표는 그 폭의 전부가 판면 바깥에 식자되지만 겹따옴표의 경우는 그 폭의 일부만 판면 바깥에 식자되도록 하였다. 또한 따옴표들은 행 끝만이 아니라 행 머리에 걸릴 때도

20. http://scripts.sil.org/svn-public/xetex/BRANCHES/microtype/
21. 매달린 구두점은 xetexko-hanging.sty 파일에서 구현하고 있다.

```
\def\page{쪽}    \page\를      ⇒      쪽을
\def\page{page}  \page\를      ⇒      page를
\def\page{面}    \page\를      ⇒      面을
```

그림 9. `\ref` 따위가 아닌 본문 중에 자동조사를 사용한 예. XeTeX-k*o*에서는 자동조사가 완전하게 동작한다.

매달린 구두점이 된다. 본디 매달린 구두점을 구현하려면 일본어 따위에 쓰이는 고리점(。), 모점(、) 등도 그 구현에 포함해야 하겠지만, 한국어 문서에서 이들 문장부호가 쓰일 일이 거의 없다고 판단하였고 또한 임시적 성격의 코드임을 감안하여 제외하였다. 따라서 현재 XeTeX-k*o*의 매달린 구두점 구현 정도는 무척 제한적이지만, 통상적인 한국어 문서에서는 큰 문제가 없는 수준이다.

7.6 자동조사

한국어 문서의 조판에서 자동조사 기능의 필요성은 두말한 나위도 없으니 이전 k*o*.TeX에서와 마찬가지 방식으로 XeTeX-k*o*에서도 자동조사를 쓸 수 있다.[22] 그런데 앞서 말한 바와 같이 XeTeX-k*o*의 자동조사 기능은 이전의 k*o*.TeX보다 더 완벽해졌으니 `\ref`, `\pageref`, `\nameref`, `\cite` 따위의 뒤에서 뿐만 아니라 일반적인 문장 중에서도 잘 작동하게 되었다. 과거에는 일반 문장 중에서는 한글이나 한자 뒤에서만 정상적인 동작이 보장되었지만, XeTeX-k*o*에서는 XeTeX의 새로운 원시명령 덕분에 한글 · 한자뿐만 아니라 영문자나 숫자 뒤에서도 자동조사가 완전하게 동작한다. 이를테면 예전 k*o*.TeX에서는 전처리부에 `\def\page{쪽}`이라고 정의해 두었다가 나중에 `\def\page{page}`라고 재정의하면 본문 중 `\page\를`이라고 사용된 부분에서 자동조사가 제대로 작동할 것이라고 신뢰할 수 없었다. 하지만 XeTeX-k*o*에서는 이런 걱정 없이 자유롭게 매크로를 재정의할 수 있으니, 그 효과가 괄목할 정도까지는 아니겠지만 어쩌다 접할 수 있는 성가신 부담을 조금은 덜 수 있게 된 것이다.

다만, 이미 전술하였지만 jamo-pua 맵핑과 자동조사 기능의 부조화 문제 때문에 옛한글 자모 직후에서는 자동조사가 정상적으로 작동하지 않을 수 있음을 다시 한번 지적해 둘 필요가 있겠다. 이 문제를 해소하려면 jamo-pua 맵핑 대신에 옛한글 오픈타입 렌더링 속성을 지원하는 폰트를 이용하여 옛한글을 식자하면서 `\enablejamoautojosa` 명령을 선언해 두면 된다. 다음은 XeTeX-k*o*가 제공하는 자동조사 명령을 망라한 목록이다.

```
\가 \이      \는 \은      \를 \을      \와 \과      \으로 \로      \이라 \라
```

옛한글과 관련하여 지적할 것은 이러한 `\은`, `\을` 따위의 자동조사 명령은 반드시 유니코드 완성형 음절 코드로 입력하여야지, 옛한글 자모 코드로 입력해서는 안 된다는 점이다. 다시 말해서 `\은`이라는 자동조사 명령은 U+C740의 코드를 가져야 하며 U+110B U+1173 U+11AB의 자모 코드를 잇달아 입력하는 첫가끝 방식은 허용되지 않는다. 옛한글만으로 이루어진

22. XeTeX-k*o*의 자동조사 지원은 주로 xetexko-josa.sty 파일에 의하여 이루어진다. 물론 그 배경에는 xetexko-space.sty 파일을 필요로 한다.

문서를 조판할 일이 거의 없다고 판단하였고 또한 혹시 그럴 일이 있더라도 옛한글 직후에 자동조사를 쓸 일이 거의 없을 것이라 생각하여 굳이 따로 자동조사 명령을 추가하지 않았기 때문이다. 하지만 장차 필요성이 있다고 결정되면 후자의 자모 코드로 이루어진 자동조사 명령도 얼마든지 제공할 수 있을 것이다.

7.7 패키지 옵션과 항목 번호

xetexko를 불러들일 때 또는 일반적으로 kotex을 불러들일 때 약간의 패키지 옵션을 줄 수 있다. 예전의 ko.TEX에서는 이러한 패키지 옵션이 상당히 많아서 번잡했는데 XƎTEX-ko는 오직 두 가지 옵션, 즉 `[hangul]`과 `[hanja]`만 제공한다. 전자는 편(part) 제목이나 장(chapter) 제목을 "제 1 편" "제 1 장"의 방식으로 만들어 주며, 후자는 "第 1 篇" "第 1 章"의 모습으로 만들어준다. 주의할 것은 과거 ko.TEX에서와 달리 XƎTEX-ko에서는 절(section) 제목의 모양을 "제 1 절"과 같은 모습으로 바꾸지 않고 영문 조판의 기본값 그대로 절 번호만 사용하도록 하였다는 점이다.

표 4는 `[hangul]` 및 `[hanja]` 옵션에 의해 활성화되는 한글 이름 명령어들을 나열한 것인데,[23] 어쨌든 실무적인 작업에 있어서는 이렇게 기본으로 주어지는 이름이나 양식이 불만족스러운 경우가 많을 것이므로 사용자의 입맛에 맞게 스스로 재정의할 필요가 있을 것이다.

또한 예전 ko.TEX과 마찬가지로 XƎTEX-ko는 나열항목이나 그밖의 번호 형식에 사용할 수 있도록 한국어 기호의 항목 번호로 제공하고 있다. 표 5는 이러한 항목 번호들을 망라한 목록이다.[24] 사용자들은 필요에 따로 적절히 취사 선택하여 이를테면

```
\pagenumbering{hanjanum}
```

처럼 지시함으로써 쪽번호를 한자 형식으로 매길 수 있을 것이다. 물론 표에서 제시된 바와 같이 기본적으로 한자 번호가 24번까지밖에 제공되지 않기 때문에 조판할 쪽수가 많지 않은 경우에만, 아니면 사용자가 매크로를 작성하여 번호를 확장하는 경우에만 유용할 수 있겠다. 어쨌든 이러한 응용은 쪽번호에만 국한되지 않을 것이니 실무적인 사용은 사용자의 몫이다.

한편 표에서 나열된 항목 번호 양식들은 enumerate 패키지나 enumitem 패키지 사용자들에게도 의미가 있다. enumerate를 로드한 후, 또는 그대신, dhucs-enumerate를 불러들이면 jaso, gana 따위의 항목 번호를 enumerate 패키지의 용법에 맞추어 사용할 수 있는 것이다. 마찬가지로 enumitem 대신, 또는 이 패키지를 로드한 후에, dhucs-enumitem을 불러들이면 되는 것이다. 구체적인 사용법은 해당 패키지의 매뉴얼을 참조하여 이를 한국어 항목 번호에 응용하기 바란다.

8 XƎTEX-ko — 혹은 XƎTEX — 의 한계

지금까지 XƎTEX-ko의 여러 가지 기능과 사용법에 대하여 서술하였거니와, 그것은 레거시 엔진을 지원하던 기존 ko.TEX의 조판능력을 분명 모든 면에서 — 마이크로타입 기술을 아직

23. konames-utf.tex 파일에 이들 매크로가 정의되어 있다.
24. kolabels-utf.tex 파일에 정의되어 있다.

표 4. 패키지 옵션에 의해 변경 또는 생성되는 매크로 목록

매크로	hangul 이름	hanja 이름
\today	2011년 2월 5일	2011年 2月 5日
\enclname	동봉물	同封物
\ccname	사본	寫本
\headtoname	받는이	受信人
\seename	\을 참고	\을 參考
\alsoname	\을 함께 참고	\을 參考
\contentsname	차 례	目 次
\listfigurename	그림 차례	그림 目次
\listtablename	표 차례	表 目次
\refname	참고 문헌	參考 文獻
\indexname	찾아보기	索 引
\tablename	표	表
\abstractname	요 약	要 約
\bibname	참고 문헌	著書 目錄
\appendixname	부록	附錄
\KSTHE	제	第
\partname	편	篇
\chaptername	장	章
\sectionname	절	節
\colorlayer	환등판 색깔	幻燈版 色相
\glossaryname	용어집	語 彙

이용할 수 없다는 점만 빼고는—훨씬 능가하는 것이다. 트루타입 · 오픈타입 폰트를 직접 접근하고 이들 폰트의 고급 레이아웃 속성들도 거의 완전히 이용할 수 있음은 물론이고, 한글과 한자와 그밖의 비 한중일 문자에 대해 유연한 글꼴 설정을 할 수 있으며 이들 문자 사이의 정교한 자간 조정도 가능하게 되었다. 더욱이 상당히 만족할만한 일본어 · 고문헌 조판 기능, 세로쓰기 기능, 완전한 자동조사의 구현 등등 기존의 한국어 텍이 가지고 있었던 여러 문제점들을 거의 해소하는 성취를 누릴 수 있게 되었다. 하지만 XeTeX-k*o*는 여전히 많은 해결해야 할 문제점 또는 한계를 안고 있으니 이제 마지막으로 우리는 이러한 한계가 무엇인지, 그것을 과연 극복할 수 있는 것인지 살펴보기로 하자.

이미 앞서 우리는 이러한 한계들에 관하여 단편적으로 언급해왔다. 이를테면 유연한 글꼴 명령의 hyphens 항목과 colons 항목에 결과적으로 동일한 문자인 엠대시(U+2014)가 이중으로 할당되지 않을 수 없었던 점, 자간 설정에서도 마찬가지로 결과적으로 동일한 홑따옴표 · 겹따옴표를 아스키 문자의 글꼴 대체나 리거처로 이루어진 경우와 유니코드 문자(U+201C 따위)를 직접 입력한 경우를 달리 취급하여 전자의 경우에는 quotewidth로 설정하고 후자의

표 5. XeTeX-k*o*가 제공하는 항목 번호 양식 목록

매크로	항목 번호
\jaso	ㄱ ㄴ ㄷ ㄹ ㅁ ㅂ ㅅ ㅇ ㅈ ㅊ ㅋ ㅌ ㅍ ㅎ
\gana	가 나 다 라 마 바 사 아 자 차 카 타 파 하
\ojaso	㉠ ㉡ ㉢ ㉣ ㉤ ㉥ ㉦ ㉧ ㉨ ㉩ ㉪ ㉫ ㉬ ㉭
\ogana	㉮ ㉯ ㉰ ㉱ ㉲ ㉳ ㉴ ㉵ ㉶ ㉷ ㉸ ㉹ ㉺ ㉻
\pjaso	⒈ ⒉ ⒊ ⒋ ⒌ ⒍ ⒎ ⒏ ⒐ ⒑ ⒒ ⒓ ⒔ ⒕
\pgana	⒜ ⒝ ⒞ ⒟ ⒠ ⒡ ⒢ ⒣ ⒤ ⒥ ⒦ ⒧ ⒨ ⒩
\onum	① ② ③ ④ ⑤ ⑥ ⑦ ⑧ ⑨ ⑩ ⑪ ⑫ ⑬ ⑭ ⑮
\pnum	⑴ ⑵ ⑶ ⑷ ⑸ ⑹ ⑺ ⑻ ⑼ ⑽ ⑾ ⑿ ⒀ ⒁ ⒂
\oeng	ⓐ ⓑ ⓒ ⓓ ⓔ ⓕ ⓖ ⓗ ⓘ ⓙ ⓚ ⓛ … ⓩ
\peng	⒜ ⒝ ⒞ ⒟ ⒠ ⒡ ⒢ ⒣ ⒤ ⒥ ⒦ ⒧ … ⒵
\hnum	하나 둘 셋 넷 다섯 여섯 일곱 여덟 아홉 열 열하나 … 스물넷
\Hnum	첫 둘 셋 넷 다섯 여섯 일곱 여덟 아홉 열 열한 … 스물넷
\hroman	i ii iii iv v vi vii viii ix x xi xii
\hRoman	Ⅰ Ⅱ Ⅲ Ⅳ Ⅴ Ⅵ Ⅶ Ⅷ Ⅸ Ⅹ Ⅺ Ⅻ
\hNum	일 이 삼 사 오 육 칠 팔 구 십 십일 십이 … 이십사
\hanjanum	一 二 三 四 五 六 七 八 九 十 十一 十二 … 二十四

경우에는 자동적으로 반각으로 식자하도록 함으로써 서로 다르게 취급한 점 따위가 그러했다. 이러한 문제점은 XeTeX이 제공하는 원시명령으로는 기껏해야 토큰 단계의 조작만 가능할 뿐, XeTeX의 내부 깊숙한 곳에서 이루어지는 노드 단계의 조작은 불가능하다는 데서 기인한다.

말하자면 텍을 유기체에 비유할 때 구강에서 음식물을 씹어 식도로 넘기는 과정은 매크로를 이용하여 접근하고 제어할 수 있지만, 저 아래 텍의 복강에서 이루어지는 정교한 소화 과정에는 접근할 수 없다고 표현할 수 있다 [4, 38쪽]. 현재 한창 개발 중인 LuaTeX처럼 텍의 내장 소화기관까지도 접근하여 통제할 수 있도록 허용된다면 사용자가 아스키 영역 하이픈 세 개를 연달아 입력하였든, 아니면 직접 유니코드 U+2014를 입력하였든, 양자 모두 복강에서는 동일한 노드를 가지게 되고 따라서 두 경우를 달리 취급하여 혼란을 야기할 필요가 전혀 없는 것이다. 하지만 XeTeX은—사실 XeTeX뿐만 아니라 pdfTeX을 비롯한 모든 전통적인 텍 엔진에서는—오직 구강 또는 식도 단계의 매크로 접근을 허용할 뿐이므로 매크로 작성자 입장에서는 양자가 서로 다른 문자 코드가 되지 않을 수 없다. XeTeX의 획기적인 새로운 기능, 즉 \XeTeXinterchartokenstate 및 관련 원시명령들은 한 단계 더 깊은 곳에 접근할 수 있도록 허용했지만 결국 기껏 식도 단계에 그치고 있으며 내장기관은 여전히 접근이 봉쇄되고 있는 것이다.

물론 매크로 작성자가 창의력을 발휘하여 불굴의 노력으로 대처한다면 이러한 여러 문제들 가운데 몇몇은 극복 가능할 수도 있겠다. 하지만 문제의 근본은 여전히 해결되지 않은 채로 남게 될 터이니 그것도 일정한 한계가 있다고 하지 않을 수 없다. 다시 말해서 엔진 자체가 변

색깔을넣어본다	색 깔 을 넣 어 본 다
\textcolor{grey}{색깔}을넣어본다	색 깔을 넣 어 본 다
굵은글자로식자	굵 은 글 자 로 식 자
\textbf{굵은글자}로식자	**굵 은 글 자**로 식 자
그룹안쪽바깥쪽	그 룹 안 쪽 바 깥 쪽
그룹{안쪽}바깥쪽	그 룹안 쪽바 깥 쪽
안녕하세요	안 녕 하 세 요
안녕\relax 하세요	안 녕하 세 요
드러냄표강조	드 러 냄 표 강 조
\dotemph{드러냄표}강조	드러냄표 강 조

그림 10. 특수한 토큰을 만났을 때 자간이 정상적으로 동작하지 않는 사례. 한글 자간을 일부러 과장되게 0.5em으로 지정하였다.

화하지 않는 한 매크로로 구현할 수 있는 기능에는 제한이 있을 수밖에 없다는 평범한 진리를 인정할 수밖에 없는 것이다. 어쨌든 이러한 근본적인 한계로 인하여 때때로 일반 사용자는 당혹스러운 결과를 마주하는 사태가 발생할 수 있는데 그림 10에서는 그중 자간에 관련한 몇 가지 사례를 제시하였다.

그림에서는 XeTeX-k*o*—또는 XeTeX—의 한계로 인한 의도하지 않은 결과가 발생할 수 있음을 보여주기 위해 일부러 한글 자간에 0.5em이라는 과도하게 큰 값을 사용하여 눈에 잘 띄도록 하였다. 첫 번째 예시는 글자에 색깔을 입힌 경우로서 자간이 지시된 0.5em이 아니라 0pt의 간격으로 비정상 동작한 예를 보여주고 있다. 글자색은 통상적인 문서에서 그리 자주 발생하는 사례가 아니겠지만 두 번째 예가 보여주는 바와 같이 글꼴을 변경하는 경우는 텍 문서에서 비일비재하게 있는 일이다. 이때에도 한글 자간이 정상적으로 동작하지 못하고 있음을 우리는 눈으로 확인할 수 있다. 세 번째와 네 번째 예시는 단순히 그룹을 열고 닫았을 때나 심지어 `\relax`처럼 아무 일도 하지 않는 원시명령을 삽입했을 때조차 정상적 동작이 보장되지 않음을 보여주고 있다. 다시 말해서 텍스트가 다른 요소의 방해 없이 물 흐르듯 흘러가고 있는 한 지시한 자간 설정이 잘 작동하지만 텍스트 이외의 어떤 다른 요소, 이를테면 박스, 컨, 글루, 페널티, 와츠잇, 그룹열기(`\bgroup`), 그룹닫기(`\egroup`), 심지어 `\relax`에 이르기까지 텍스트 문자 이외의 토큰을 만나면, 그 특수 토큰이 문서 조판에서 별 의미가 없는 것이라 하더라도 텍스트의 흐름을 끊어버리게 되고 결과적으로 우리는 원하지 않는 결과물을 마주하게 되는 것이다.

물론 사용자가 쓸데없이 그룹을 열었다 닫거나 공연히 `\relax` 명령을 삽입하거나 하지는 않을 것이다. 하지만 이러저러한 조판 효과를 보기 위해 이러저러한 패키지가 제공하는 이러저러한 명령을 사용자가 사용하는 순간, 그 명령의 매크로가 사용자도 모르는 사이에

내부적으로 저러한 특수한 토큰들을 입력 토큰열 사이에 삽입하게 되고 결과적으로 의도에 없는 자간의 왜곡이 유발된다. 이 문제를 근본적으로 해결할 수 있는 방법은 거의 없다고 해야 할 것이다.[25] 이 세상에는 수없이 많은 패키지가 있고 수없이 많은 매크로 명령들이 있으니 이들을 일일이 XeTeX-k*o*의 한국어 조판 구현에 맞추어 수정할 수는 없는 노릇이기 때문이다. 따라서 혹 저러한 비정상적인 사태를 만나게 된 사용자 스스로가 구체적인 상황에서 맞게 적절한 대응책을 강구하는 것 이외에는 따로 XeTeX-k*o*의 입장에서 일괄적으로 해결할 수 있는 것은 거의 없는 셈이다.

XeTeX-k*o*가 한글 자간의 기본값을 0pt로 잡은 것도, 한글 · 한자와 영문자 사이 간격인 hu의 기본값을 0.06em이라는, 16분각도 안 되는 아주 미세한 값으로 잡은 것도 다 이러한 예기치 못한 부작용을 최소화하고자 한 데 기인한다. 일반 사용자들도 개인적으로 자간을 설정할 때 지나치게 과도한 값을 지시하지 않도록 주의할 필요가 있다. 미세한 부작용은 어쩔 수 없이 항상 발생하는 것이지만 자간 값을 과도하게 설정하면 이런 부작용이 눈의 띄게 두드러지게 되기 때문이다. 사실 XeTeX-k*o*는 이런 종류의 부작용을 오히려 적극적으로 이용하기도 한다. 전술한 `\inhibitglue` 명령은 실은 원시명령 `\relax`로 확장되게 해 두었으니, 텍스트의 흐름이 `\relax`에 의해 끊어지면 그 지점에 미리 지시된 자간이 아닌 0pt의 공백이 삽입되는 성질을 남용(?)한 사례이다.

XeTeX-k*o*의 이러한 한계는 사실 XeTeX-k*o* 스스로가 제공하는 명령에서도 나타난다. 위 그림 10의 마지막 예에서 보듯이 드러냄표를 이용하면 모든 자간 설정이 작동하지 않는 문제가 있다. 박스로 둘러싸여 피강조문자 위에 올려지는 드러냄표가 텍스트 흐름을 끊어 놓기 때문이다. 물론 이는 XeTeX-k*o*가 직접 통제할 수 있는 매크로이기 때문에 문제에 대처하려고 마음만 먹는다면 어쩌면 해결이 전혀 불가능하지는 않을 것이다. 하지만 피강조문자가 한글인지 한자인지, 피강조문자 주변에 영문자나 숫자가 있는지 없는지 여부를 모두 고려하여 매크로를 작성하여야 하므로 생각보다 지난한 작업이 되지 않을 수 없고 과연 그만큼 가치가 있는 것인지 의문스러워지게 된다. 따라서 현재 상황으로는 단순히 0pt의 공백만 삽입하는 데 만족하고 있다.

그밖에 XeTeX-k*o*의 한계로서 반드시 언급해 두어야 할 것으로 이른바 '고아글자'의 문제가 있다. 고아글자란 한글 한 글자—그 뒤에 마침표나 쉼표가 붙어있는 경우도 포함한다—가 외따로 한 줄을 차지한 채 문단이 바뀌어 행의 여백이 지나치게 많이 남는, 그리하여 음영 비율이 적절치 못하게 되는 때를 일컫는다. 이러한 고아글자를 회피하는 코드가 기존의 k*o*.TeX에서는 부분적으로 구현되었지만 XeTeX-k*o*에서는 그것이 빠져버렸다. 한국어 조판 구현의 근본적인 방법이 달라져 고아글자 회피를 구현해내기가, 불가능하지는 않더라도 대단히 어렵게 되어버렸기 때문이다. 결국 XeTeX-k*o*의 현 상태에서는 고아글자가 발생하는 문단에 대하여 사용자가 일일이 `\nobreak` 명령을 마지막 글자 앞에 넣어주는 방법밖에는 없다. 다만 `\nobreak`라는 명령을 주는 것이 번거로운 측면이 있어 XeTeX-k*o*는 "`\다`" 명령을 마련해두

25. XeTeX 하에서 이런 한계를 근본적으로 극복할 수 있는 방법이 없는 것은 아니다. 그것은 폰트를 잘 만드는 것, 또는 잘 만들어진 폰트를 사용하는 것이다. 커닝을 멋지게 구현한 GPOS 테이블이 들어간 폰트가 있다면 굳이 매크로를 이용하여 자간 조정을 시도할 필요도 없이 이 오픈타입 속성을 사용하라고 지시하면 족한 것이다. 하지만 현실은 그렇지 못함을 이 글의 서두에서 지적하였다.

고 있다. 현대 한국어에서는 문단의 마지막 글자가 "다" 인 경우가 대부분이므로 이를 통하여 입력 부담을 조금이나마 덜 수 있게 하기 위함이다. 물론 이러한 명령은 `\penalty10000`으로 확장되기 때문에 해당 지점에서 자간이 작동하지 않게 된다. 따라서 사용자는 모든 문단의 마지막 글자에 대해 이 명령을 내려주기보다는 눈으로 고아글자임이 확인된 때에 한해서 적용하는 것이 바람직하다.

끝으로 한 가지 지적해두어야 할 것이 있다. 한국어 입력과 관련하여 그동안 필자는 XeTeX은 유니코드와 EUC-KR만 입력 인코딩으로 받아들이고 윈도 운영체제에서 사용되는 이른바 CP949 인코딩으로는 입력할 수 없는 것으로 알고 있었다. 그래서 XeTeX-k*o* 매뉴얼 [6] 에서나 한국텍학회 학술대회에서 이것을 XeTeX의 한계 중의 하나라고 쓰고 또 그런 취지로 강연도 했었다. 하지만 최근 XeTeX의 소스를 들여다보고는 이런 생각이 잘못이었음을 깨달았다. XeTeX은 엔진 바이너리 파일에 정적 링크된 ICU 라이브러리[26]를 이용하여 입력 인코딩을 실시간 유니코드로 바꿔주는 기능을 수행하고 있는데, ICU 라이브러리에는 EUC-KR은 물론이고 CP949 인코딩에 관한 데이터도 들어있었던 것이다. 따라서 설정만 잘 하면 CP949 입력 인코딩도 아무 문제 없이 처리할 수 있으니

```
\XeTeXinputencoding="korean"
\XeTeXdefaultencoding="korean"
```

이라고 선언하면 된다. 윗줄은 이 선언이 포함된 파일의 입력 인코딩을, 아랫줄은 원본 파일에서 `\input` 명령으로 불러들이는 파일들의 입력 인코딩을 지시한다. 따라서 하나의 파일로 이루어진 문서라면 윗줄 하나로 충분하다. 어쨌든 이러한 설정으로 EUC-KR과 CP949 입력을 모두 처리할 수 있으므로, CP949 입력이 불가능한 한계가 있다는 필자의 과거 언급은 이 글을 빌어 모두 수정하는 바이다.

9 맺음말

몇 가지 한계에도 불구하고 XeTeX-k*o*는 기존의 한국어 텍 시스템이 가지던 많은 문제점들을 해결하고 새로운 가능성을 열어 놓은 획기적인 한국어 조판 텍 매크로 패키지이다. 그것은 XeTeX이라는 탁월한 텍 엔진이 있었기에 가능한 일이었다. 하지만 아직도 많은 할 일이 남아 있거니와, 글꼴 분리와 할당에 관해서도 세부적인 차원에서 문자 범주를 수정하거나 확장해야 할 것이 적지 않을 터이고 자간 조정 기능, 자동조사 기능, 드러냄표 강조 기능, 매달린 구두점 기능 따위에서도 아직 다듬고 개선해야 할 사항이 많다. 더욱이 XeTeX 엔진 자체가 앞으로도 계속 발전해 나갈 것이므로 그에 발맞추어 XeTeX-k*o*도 진보를 거듭하지 않을 수 없는 것이다.

하지만 한국의 텍 매크로 개발자의 인력 풀이 그리 크지 못하다는 이유 때문에 아마 앞으로 XeTeX-k*o*의 발전은 더디게 이루어질 수밖에 없을 것 같다. 더구나 현재 LuaTeX 엔진이 왕성하게 개발되는 중이고 이와 관련된 베이스 매크로 패키지들이 하나 둘 그 모습을 드러내고 있는 상황에서 실험적으로 만들어진 기존의 LuaTeX-k*o* 패키지도 그에 따라 전면 재검토되

26. `http://icu-project.org/`. XeTeX은 오픈타입 레이아웃 속성 지원도 ICU 라이브러리에 의존하고 있다.

고 수정되지 않으면 안 된다는 과제가 주어져 있어 XƎTEX-k*o* 개발에 할애할 시간과 노력은 더욱 줄어들 전망이다. 이러한 난국을 타개하기 위해서는 매크로 개발자가 더 많이 출현해서 한국어 텍 패키지의 개발을 서로 분담하는 협업체제가 요구된다고 하겠다.

텍 매크로 개발, 특히 한국어 조판을 구현하는 매크로의 개발은 텍의 저수준 원시명령들의 의미와 역할을 잘 이해하는 데서 출발한다. 이를 위해서는 무엇보다 [4]의 저서를 숙독하지 않을 수 없는데, 이는 XƎTEX이나 LuaTEX 매크로를 개발하는 경우에도 여전히 타당하다. 크누스의 오리지널 텍이 출현한 이후 등장한 모든 종류의 텍 엔진들은 오리지널 텍 소스를 기반하여 이를 약간씩 확장 또는 수정한 것에 지나지 않기 때문이다. 따라서 어떤 텍 엔진을 위해서건 조판 매크로를 개발하는 데는 아직도 텍의 바이블인 크누스의 저서를 참고하는 것이 필수적이며 그 위에 각각의 텍 엔진의 매뉴얼에 따라 이를 응용하게 되는 것이다. 바라건대 텍 매크로 개발자가 한국에서도 많이 출현하여 어떤 다른 조판 시스템도 따라 올 수 없는 질 좋고 사용하기 편리한 최고급 한국어 조판을 XƎTEX이든 LuaTEX이든 또는 그 무엇이건 텍으로 구현할 수 있기를 기대한다. 물론 폰트 없는 텍은 상상할 수도 없거니와 다른 나라 폰트의 모범이 되는 훌륭한 한국어 폰트의 개발도 같이 이루어지면 더 이상 바랄 나위가 없겠다.

참고 문헌

1. 김강수, 한글 문장부호의 조판관행에 대하여, The Asian Journal of TEX **1** (2007), no. 1, 17–30.
2. 이기황, Hangul-ucs를 이용한 옛한글 조판, The Asian Journal of TEX **1** (2007), no. 1, 31–43.
3. 이주호, 출판 현장에서의 텍의 활용, The Asian Journal of TEX **3** (2009), no. 1–2, 51–79.
4. Donald E. Knuth, *The TEXbook*, Addison-Wesley, 1986.
5. Haruhiko Okumura, *pTEX and Japanese Typesetting*, The Asian Journal of TEX **2** (2008), no. 1, 43–51.
6. 김도현, XƎTEX-k*o* 간단 매뉴얼, 2010. `http://ftp.ktug.or.kr/KTUG/texlive/texmf-dist/doc/xelatex/kotex-dev/xetexko/xetexko-doc.pdf`
7. Will Robertson, *The fontspec package*, 2008. `CTAN:macros/xetex/latex/fontspec/fontspec.pdf`
8. ———, *The XƎTEX reference guide*, 2009. `CTAN:info/xetexref/XeTeX-reference.pdf`