The CJK package for \LaTeX\[2\varepsilon\] —
Multilingual support beyond babel

Werner Lemberg  
Kleine Beurhausstr. 1  
D-44137 Dortmund  
Germany  
a7971428@unet.univie.ac.at

Abstract
With Mule (multilingual Emacs) you can write texts in multiple languages. This editor is especially designed to handle the various encodings and character sets of Asian scripts such as Big 5 and GB for Chinese, JIS for Japanese, etc. Even more, you can use multiple CJK character sets simultaneously which enables e.g. Chinese users to write simplified (ji'àntì 简体字) and traditional Chinese characters (fànìntì 繁體字) at the same time.

The CJK package is the analogue for \LaTeX\[2\varepsilon\] (to be run under standard \TeX). Most of the CJK (Chinese/Japanese/Korean) encodings are implemented; an interface between Mule and \LaTeX\[2\varepsilon\] is provided by an output encoding filter for Mule. CJK is of course not restricted to Mule. Any editor/environment which is able to handle double byte encodings can be used.

If you restrict babel to (7-bit) ASCII as the input encoding it is possible to embed babel into CJK seamlessly. Using Mule’s output filter, you even don’t need enter \LaTeX\-specific accent macros; the accented characters will be converted automatically.

Included in the CJK package are auxiliary programs which can convert CJK TrueType and bitmaps fonts into pk files.

Introduction
\TeX\ (and thus \LaTeX\) is an extremely flexible text formatting system which supports some multilingual features since the final version 3, mainly by allowing 8-bit input and fonts. But for CJK languages (Chinese/Japanese/Korean) this is still not enough because all encodings have more than 256 characters each.

The aim of the CJK package is to provide multibyte encoding support for \LaTeX\[2\varepsilon\] without any specific extensions of \TeX. It contains modules for GB and Big 5 (Chinese), JIS and SJIS (Japanese), and KS (Korean) encoding, to name a few. In section “Unicode” on page 219 you’ll find some notes on the pros and cons of Unicode in relation to \TeX.

As far as I know there is only one freely available editor which is capable to display multiple character sets at the same time: Mule, the multilingual extension of Emacs.\footnote{The next major releases of the various Emacs flavours (GNU Emacs and XEmacs) will merge the features of Mule back into Emacs.} A special output filter for Mule converts the internal encoding of Mule directly into something \LaTeX\ can understand. See the section “The interface between Mule and CJK” on page 220 for more details.

Among other utilities two font converters called hbf2gf and ttf2pk are provided to convert CJK fonts into pk and tfm fonts; both programs are discussed in another paper of the proceedings ([8]).

The latest version of CJK can be found on CTAN in the directory language/chinese/CJK; various basic bitmap font packages are provided in fonts/CJK.

CJK encoding schemes
It is not possible to represent CJK character sets with one byte per character. At least two bytes are necessary, and most of the common CJK encoding schemes (GB, Big 5, JIS, KS, etc.) use a certain range for the first byte (usually OxA1 to OxFE or a part of it) to signal that this and the next byte represent a CJK character. As a consequence, ordinary ASCII characters (i.e., characters between Ox00 and Ox7F) remain unaffected.
The \texttt{CJK} package for \LaTeX\ — Multilingual support beyond \texttt{babel}

<table>
<thead>
<tr>
<th>encoding</th>
<th>1. byte</th>
<th>2. byte</th>
<th>3. byte</th>
</tr>
</thead>
<tbody>
<tr>
<td>GB</td>
<td>0xA1–0xF7</td>
<td>0xA1–0xFE</td>
<td>—</td>
</tr>
<tr>
<td>Big 5</td>
<td>0xA1–0xF9</td>
<td>0x40–0xFE</td>
<td>—</td>
</tr>
<tr>
<td>JIS</td>
<td>0xA1–0xF4</td>
<td>0xA1–0xFE</td>
<td>—</td>
</tr>
<tr>
<td>SJIS</td>
<td>0xA1–0xFE</td>
<td>0x40–0xFE</td>
<td>—</td>
</tr>
<tr>
<td>KS</td>
<td>0xA1–0xFD</td>
<td>0xA1–0xFE</td>
<td>—</td>
</tr>
<tr>
<td>UTF 8</td>
<td>0xC0–0xEF</td>
<td>0x80–0xBF</td>
<td>0x80–0xBF</td>
</tr>
<tr>
<td>CNS</td>
<td>0xA1–0xFE</td>
<td>0xA1–0xFE</td>
<td>—</td>
</tr>
</tbody>
</table>

\begin{figure}
\centering
\begin{tabular}{|c|c|c|c|}
\hline
encoding & 1. byte & 2. byte & 3. byte \\
\hline
GB & 0xA1–0xF7 & 0xA1–0xFE & — \\
Big 5 & 0xA1–0xF9 & 0x40–0xFE & — \\
JIS & 0xA1–0xF4 & 0xA1–0xFE & — \\
SJIS & 0xA1–0xFE & 0x40–0xFE & — \\
KS & 0xA1–0xFD & 0xA1–0xFE & — \\
UTF 8 & 0xC0–0xEF & 0x80–0xBF & 0x80–0xBF \\
CNS & 0xA1–0xFE & 0xA1–0xFE & — \\
\hline
\end{tabular}
\caption{encoding schemes implemented in CJK}
\end{figure}

Some notes on the various encodings given in table 1:

- SJIS, also known as MS-Kanji, consists of two overlayed character sets: the so-called halfwidth Katakana (JIS X0201-1976, 1-byte characters encoded in the range 0xA1 to 0xDF) and the (fullwidth) JIS character set (JIS X0208-1990, mapped to the remaining code points).
- Some encoding schemes (Big 5, SJIS) have gaps in the range of the second byte.
- UTF 8 (Unicode Transformation Format 8), also called UTF 2 or FSS-UTF, is a special representation of Unicode (resp. ISO 10646). It uses multibyte sequences of various length, but only 2-byte and 3-byte sequences are implemented in CJK. ASCII characters will be used as-is — without this property it would be impossible to use UTF 8 with \TeX.
- CNS is defined to have 16 planes with 94 × 94 characters. Currently 7 planes are assigned (CNS 1 to CNS 7, an eighth plane has been said to be under development).
- It’s difficult to input Big 5 and SJIS encoding directly into \TeX\ since some of the values used for the encodings’ second bytes are reserved for control characters: ‘,’ ‘,’ and ‘\’. Redefining them breaks a lot of things in \TeX. to avoid this, preprocessors are normally used which convert the second byte into a number followed by a delimiter character.

For further details please refer to [10]; LUNDE discusses in great detail all CJK encodings which are or have been in use. See also section “Input encodings, output encodings, character sets” below.

The \texttt{CJK} package in detail

\textbf{An example.} Here a small lucullic text:

\begin{verbatim}
\documentclass{article}
\usepackage{CJK}
\usepackage{pinyin}

\begin{document}
\begin{CJK}{Bg5}{fs}
Ңܘ hrsΦʕ埠ඵf
\Wo3 \hen3 \xi3\huan1 \chi1
Zhong1\guo2\fan4.
I like to eat Chinese food very much.
\end{CJK}
\end{document}
\end{verbatim}

I like to eat Chinese food very much.

This example shows that basically only two steps are necessary to write Chinese: loading \texttt{CJK} with \texttt{\usepackage{CJK}} and opening a CJK environment. \texttt{Bg5} selects the Big 5 encoding for Chinese written with traditional characters, \texttt{fs} the font to be used (in this case it is \texttt{f`anqt`ı}). The procedure is slightly different if you use \texttt{cjk-enc.el} and will be described below in the section “The interface between Mule and CJK” on page 220.

\texttt{\usepackage{pinyin}} loads the \texttt{pinyin} package which is also part of \texttt{CJK}. It enables input of \texttt{p`inyin} syllables, the transcription system of Chinese used in Mainland China (see also the section “The \texttt{pinyin} package” on page 221).

\textbf{Basic concepts.} To understand how CJK works behind the scenes some basic concepts have to be introduced.

\textbf{Input encodings, output encodings, character sets.} Since the arrival of NFSS input and output encodings are clearly separated in \LaTeX. With CJK encodings the situation is a bit more complicated because some encodings are input and output encodings \textit{at the same time}. To make things even more confused, they can form a character set also!

Consider as an example Big 5. This is a character set developed by software companies in Taiwan for Chinese written with traditional Chinese characters. It is common practice to describe CJK character sets in rows, usually (but not necessarily!) represented by the first bytes of the output encoding. For Big 5 we have 94 rows of 157 characters each. Not all rows are fully occupied: 94 × 157 = 14 758 characters are possible, but only 13 053 characters
are really defined in the basic form of Big 5. This character set can be split into three parts: 408 symbols (rows 1–3), 5 401 Level 1 hánzi (rows 4–38), and 7 652 Level 2 hánzi (rows 41–89). Level 1 contains the most frequent characters, Level 2 is an extension for rarely used characters occurring mainly in names. With hánzi (漢字, kanji in Japanese, hanja in Korean) all ideographic glyphs derived from the Chinese script are denoted.

Row 1 of the Big 5 character set is mapped to 0xA1 as the first byte in Big 5 encoding, row 2 is mapped to 0xA2 etc.

You may ask: “Why only 157 characters per row? Table 1 would imply that 191 characters are available for each row.” The answer is simple: due to historical reasons the range for the second byte of Big 5 input encoding (we are no longer talking about the Big 5 character set!) is split into two subranges: 0x40–0x7E and 0xA1–0xFE. And for convenience, almost all fonts providing the Big 5 character set can be accessed with an output encoding identical to the Big 5 input encoding.

The counter example is the SJIS input encoding. Here we have two distinct Japanese character sets (JIS X0201 and JIS X0208) which will be accessed as two different fonts in the CJK package having a 1-byte and a 2-byte output encoding respectively.

The CJK macro layers. CJK makes all characters above 0x7F active (except 0xFF). The macro level assigned directly to the active characters is called ‘binding’ (stored in files with the extension bdg). The binding decides whether only the current byte, the next byte with the current byte or possibly the next two bytes together with the current byte represent a CJK character (example: JIS encoding has only 2-byte characters, SJIS has 1-byte and 2-byte characters).

The next level chooses the encoding of the CJK font and the output encoding of the subfonts (called ‘fontencoding’: see also the sections “Subfonts” and “CJK font definition files” below for further information); the corresponding macros are stored in files with the extension enc. Here the proper subfont together with some font offsets will be selected and passed as arguments to the next level.

Macros from the third and last level (stored in files with the extension chr) finally select the proper character, check whether it is a special character etc., and print it out.

User selectable are only the encoding and fontencoding, the other levels are chosen automatically.

‘Preprocessed’ mode. Big 5 and SJIS encoding can’t be handled well within \TeX due to some characters in the range of the encodings’ second bytes which interfere with the \TeX control characters ‘\‘, ‘‘, and ‘’. It is possible to redefine them (and the CJK package provides two environments, \Bg5text and \ SJIS\text, which exactly do that), but many commands of \TeX don’t work inside of them. Another annoying fact is that second bytes smaller than 0x80 are affected by case changing commands, altering the CJK characters!

Thus I have decided to program small preprocessors written in C to convert the encodings into a form which won’t cause problems: the second byte of an encoding will be converted into its decimal equivalent, followed by 0xFF as a delimiter character.

This approach works for all encodings except UTF 8. CJK simply checks the presence of the \CJ K\]][preproc command inserted by the preprocessor at the very beginning of the output file: if it is defined, another set of macros connecting the ‘binding’ and ‘fontencoding’ level is used. If it is undefined, \MakeUppercase is disabled for Big 5 and SJIS encoding.

Subfonts. What has been said about output encodings in a previous section is not the truth. It’s a “little white lie”, to cite a famous computer scientist whose name I can’t remember yet. To make large CJK fonts work with \TeX we must split them. I’ve chosen the most compact subfont layout, i.e., 256 characters per subfont, but due to the modular concept of CJK it was not difficult to support other subfont schemes (like poor man’s Chinese which uses one subfont per leading byte).

The number of subfonts per font is large. A JIS encoded font for example needs 35 subfonts, a Big 5 encoded font even 55. See below the section
“CJK font definition files” for the naming scheme of subfonts.

The interaction with NFSS. Today I’m surprised by myself how simple it has been to adapt NFSS to subfont selection; only one internal macro of the \LaTeX{} kernel, namely \texttt{\textbackslash pickup@font}, must be redefined to gain the additional subfont functionality needed by CJK.\footnote{\texttt{\textbackslash selectfont} will also be changed slightly; nevertheless, it is not for the subfont selection but rather for the support of boldface emulation for CJK fonts.} Here is the version as implemented in CJK.sty:\footnote{Please note that almost all files in the CJK package start with \texttt{\textbackslash endlinechar -1} to avoid percent signs at the end of a line if we must suppress the newline character. The other \TeX{} macro code fragments in this document assume the same.}

\begin{verbatim}
\def\pickup@font{
  \ifx\CJK@plane \undefined\else
    \ifx\font@name \relax
      \define@newfont
    \else
      \expandafter\ifx\csname \curr@fontshape/\f@size/\CJK@plane\endcsname \relax
        \define@newfont
      \else
        \xdef\font@name{\csname \curr@fontshape/\f@size/\CJK@plane\endcsname}
      \fi
    \fi
  \fi}
\end{verbatim}

\texttt{\CJK@plane} is only defined inside of a CJK macro on the character level. It will contain the subfont plane to be used. The only purpose of the small extension in \texttt{\textbackslash pickup@font} is to define or check a new \texttt{\font@name} by appending \texttt{\CJK@plane} to the name \LaTeX{} would construct.

CJK font definition files. Font definitions for CJK fonts are basically similar to other fonts. The main difference is that you define classes of subfonts instead of a single font.

Subfont names. The default name of a CJK subfont is the font family name plus a running decimal number (normally consisting of two digits). Example: \texttt{b5ka1201}, \texttt{b5ka1202}, etc. Unicode encoded fonts have two running hexadecimal digits appended instead, and some Japanese and Korean fonts follow other naming conventions. For all declarations in font definition files you have to specify the font family only; in our example this would be \texttt{b5ka12}.

Size functions. A suite of additional font size functions which take care of the subfonts has been defined. Most of them have a ‘CJK’ prefix or postfix to standard \LaTeX{} size function names (CJK, sCJK, CJKsub, \ldots) to indicate a similar behaviour. Additionally some CJK size functions have a ‘b’ postfix to select ‘poor man’s boldface’, as I have called the emulation of bold fonts by printing a non-bold character thrice with small offsets.\footnote{An explanation for the very reason of having extra size functions for bold face emulation is beyond the scope of this (already too long) article. It can be found in the documentation files of CJK.} Providing a boldface emulation proved to be necessary because most of the freely available CJK fonts are available in one series only.

Here a sample entry for a GB encoded pixel font:

\begin{verbatim}
\DeclareFontFamily{C10}{fs}{}
\DeclareFontShape{C10}{fs}{m}{n}{<-> CJK * gsfs14}{}
\DeclareFontShape{C10}{fs}{bx}{n}{<-> CJKb * gsfs14}{\CJKbold}
\end{verbatim}

\texttt{\CJKbold} sets an internal flag to switch on boldface emulation.

NFSS font encodings. Since version 4 of the CJK package all NFSS attributes are supported. To achieve that it was necessary to have an NFSS font encoding for each CJK encoding. Figure 2 shows some of the currently available font encodings; all encodings defined by CJK start with an uppercase ‘C’, followed by two digits.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
language & environment & NFSS encoding \\
\hline
Chinese: & Bg5 & C00 \\
& GB & C10 \\
& CNS1–7 & C31–37 \\
Japanese: & JIS & C40 \\
& JIS2 & C50 \\
& SJIS & C40 (fullwidth) \\
& & C49 (halfwidth) \\
Korean: & KS & C60 (hanja) \\
& & C61 (hangul) \\
Unicode: & UTF8 & C70 \\
\hline
\end{tabular}
\caption{The correlation between CJK encodings, CJK fontencodings, and NFSS font encodings. Only the most important encodings are listed here.}
\end{table}

Some CJK encodings need more than one NFSS font encoding, as can be seen in the table (not listed here is the support for Hir\TeX{} fonts where four NFSS encodings are necessary); the first digit usually represents the CJK encoding, the second digit (except for CNS encoded fonts) the specific subfont layout.
The macro which internally maps the CJK encoding to one or more NFSS font encodings also calls \DeclareFontSubstitution with an empty font name macro to fake the fall-back mechanisms of NFSS.\footnote{The latest version of \TeX has introduced ‘HFSS’ (the ‘H’ stands for Hangul), a font selection scheme to be run in parallel with NFSS, basically having an identical interface. A lot of advantages can be gained, mainly higher speed and better error handling.}

Nevertheless, you never access the NFSS font encodings directly; this will be done automatically if you open a CJK environment or use a \CJKenc command (see below).

**CJK commands and environments.** In this section you will find some important CJK commands not discussed elsewhere in this paper.

Additionally to the CJK environment CJK* will be provided: the starred form suppresses spaces after a CJK character (using \ignorespaces which don’t appear in texts written with Chinese or Japanese as the main language. To ‘switch’ from CJK to CJK* without leaving the environment you can use the \CJKspace command (and \CJKnospace for the other direction).

To access a CJK character directly you can use the \CJKchar command; it takes the first and second byte (represented as a number) of the character code together with an optional encoding string as parameters. This is the most portable form since no input characters larger than 0x7F are used.

Four commands will control encodings and font encodings: \CJKenc, \CJKfamily, \CJKencfamily, and \CJKfontenc. To change the encoding inside of a CJK environment, use \CJKenc. It will always use the font encoding for a certain encoding which has been selected with \CJKfontenc. To change the font family you have two alternatives: The first is to define a family for a specific encoding with \CJKencfamily. If this encoding is chosen, the family defined in this way will be taken. The second is to specify a family for all encodings with \CJKfamily, overriding all \CJKencfamily commands. You must explicitly say ‘\CJKfamily{\ldots}’ to reactivate any font definitions done with \CJKencfamily. Here is a (hypothetical) example:

```
\CJKencfamily{GBt}{hei}
\CJKfontenc{JIS}{dnp}
```

```
\begin{CJK*}{Bg5}{fs}
% this is equal to
% \begin{CJK*}{\{}{\}
% \CJKenc{Bg5}
% \CJKfamily{fs}
```

In case you get overfull \hboxes caused by CJK glyphs the macros \CJKglue and \CJKtolerance will help. The former is used for Chinese and Japanese encodings, defining the glue between CJK glyphs. Its default value is set to \hskip 0pt plus 0.08\baselineskip. The latter makes sense for Korean; the default value for \CJKtolerance is 400.

Only a subset of the commands available has been introduced here. For a complete description please refer to the various CJK documentation files.

**CJK typography rules**

Some special CJK characters should not start or end a line, e.g. various kinds of parentheses and many interpunctuation characters. In Japanese there is additionally a set of Hiragana and Katakana characters which are not allowed to start a line (kinsoku shori 禁則処理); both are supported automatically by CJK (but can be controlled if necessary). The mechanism to achieve this is quite tricky: usually you have some breakable glue (using \hskip) between two consecutive CJK characters (with other words, you need intercharacter spacing for Chinese and Japanese). Every character will be checked against encoding specific lookup tables whether it is a character not allowed to start or to end a line. In the former case, the glue before the character must be made unbreakable, otherwise the glue after the character.

No space will be printed after or before a CJK interpunctuation mark in Japanese and Chinese, but in Korean spaces are used between words\footnote{which are written as a combination with Hanja and Hangul (한글), the Korean syllable script. No intercharacter glue is used but \discretionary{\ldots}{\ldots} instead; additionally \tolerance is increased to \CJKtolerance.} and after interpunctuation marks (which are in most cases the same as in Western languages).
As explained above we have macros for all CJK characters. How can we test whether the previous character was a CJK character, maybe even a special one? It’s impossible to use \futurelet or other such commands because this only works on \TeX’s macro level, not on horizontal lists which are eventually output to a .dvi file.

The solution I’ve found finally uses \lastkern to check the amount of the last kern.\footnote{UN Koang-Hi (敗光熙), the author of \hyperref[label]{HL\TeX} (\cite{12}), for Korean, uses a different solution: he modifies the space factor of specific characters to indicate a break point.} If the current character is an ordinary one, a kern of 1sp is emitted (do you remember \TeX’s smallest unit?), if no break should occur between the current and the next character, a kern of 2sp is used instead. Now the next character will be processed. If it is a CJK character, \lastkern is tested whether it is 1sp or 2sp, and the appropriate action is executed.

Another typographic rule in CJK text processing is the use of different spaces, depending on context. Between a CJK character and a non-CJK character (e.g. an English word to be cited or a number) only a space having the quarter width of a (fullwidth) kanji should be typeset, and between non-CJK characters the default space has to be used. This quarter space is called shibuaki (四分 冊). Some Japanese standards define more sophisticated rules where to print a space, but even the simple problem with having two different space widths can’t be solved automatically in standard \TeX. There exist Japanese adaptations of \TeX which handle this internally, but you can’t use these programs with other CJK languages. Nevertheless, it is solvable within \Omega; see \cite{6} for a short discussion on this topic.

The only way to manage shibuaki is to insert them manually. For this purpose I’ve redefined the tilde character to insert a quarter space instead of them manually. For this purpose I’ve redefined the \CJKtilde on this topic.

The effect of ‘~’ seems to be minimal, but in underfull boxes it is really an optical enhancement.

\textbf{The Chinese Encoding Framework (CEF)}

Christian Wittern, a former employee of IRIZ (\cite{1}), now working at the University of Göttingen, has developed CEF. Its primary aim is to access seldom used CJK encodings (most notably CNS) in a platform independent way using SGML macros of the form ‘&<encoding><code>;’. Examples for valid encoding values are ‘C3’ for CNS plane 3, ‘C0’ for Big 5, ‘U’ for Unicode; the code is given as a hexadecimal number (see \cite{15} for a detailed description).

He has also developed Kanji\text{\textit{JiBase for Windows}}, an input tool for CEF which accesses a large CJK character database (which I consider the very heart of the whole system). It is also described in \cite{15}.

A small example from an old Zen text (\textit{The Records of Zhaozhou} 趙州真際禪師語録, taken from the IRIZ ZenBase CD 1) shows how it works:

\begin{verbatim}
0304a12 古尊宿語録巻第三。

趙州真際禪師語録並行状巻（南嶽下四世顕南泉願）。

師即南泉門人也。俗姓郝氏、本曹州籍人也。講従
訥。鎮府有塔記云、師得七百甲子數、值武王微沐、壹
地焦嶺、木食草衣、僧儀不具。師初隨本師行腳到南
泉。本師先人事了、師方乃為木、師雲在方丈内飲次、
見師來參、便問、近離什麼處。師雲、瑞像院、南泉雲、還

And here the same text with the appropriate CNS character:

\begin{verbatim}
0304a12 古尊宿語録巻第三。

趙州真際禪師語録並行状巻（南嶽下四世顕南泉願）。

師即南泉門人也。俗姓郝氏、本曹州籍人也。講従
訥。鎮府有塔記云、師得七百甲子數、值武王微沐、壹
地焦嶺、木食草衣、僧儀不具。師初隨本師行腳到南
泉。本師先人事了、師方乃為木、師雲在方丈内飲次、
見師來參、便問、近離什麼處。師雲、瑞像院、南泉雲、還

The punctuation marks have been inserted by the editors and are not present in the original text.

\CJK provides a small preprocessor to convert CEF macros into \texttt{\CJKchar} macros.

\textbf{Unicode}

Characters encoded in Unicode can’t be used directly with \TeX because the encoding is 16-bit wide. Instead, you have to use UTF 8 (see table 3 for the relationship between Unicode and UTF 8).

This multibyte representation has some important advantages: it is completely transparent for ASCII characters, you can always find the beginning of a multibyte sequence because the leading byte is unambiguously defined, and it is “reasonably compact in terms of number of bytes used for encoding”, to cite from appendix A.2 of \cite{13}.

\footnote{You can find the latest version in \cite{14} which also provides some online access and conversion.}
The interface between Mule and CJK

Handa Ken’ichi (半田 剣一), the main author of Mule, constructed a Lisp code frame for an interface between Mule and CJK which I filled with the needed values. It can be integrated into AUCTeX ([11]) without any great problems, making it a very convenient multilingual environment for \TeX.

The most convenient way to specify CJK font shapes inserted immediately before an encoding change. This means that you load a file into a Mule buffer, change the name of the buffer to the target file name, select \texttt{cjk-coding} as the output encoding and save the file.

Look at figure 5. It shows a small multilingual example where Japanese is mixed with German and Czech.

Another reason is that most of the currently available Unicode encoded CJK fonts provide one glyph shape per code point. However, this is not enough for typographically correct output. In figure 4 you can see the same character in three different shapes. Ken Lunde shows in [9] that for a Unicode CJK font which really satisfies Chinese, Japanese, and Korean users about 40% more glyphs than code points are needed — this would further increase the number of \TeX subfonts to be accessed simultaneously because the order of characters (or glyphs) in the CJK section of a Unicode font does not follow the character frequency but rather the order in a famous Chinese dictionary (Kāngxī zìdiǎn 康熙字典).

The text has been taken from a synchronoptical translation of the libretto of Antonín Dvořák’s opera \Rusalka. The left column is printed in Czech (the original language), the middle column in German, and the right column in Japanese.

As you can see, babel ([2]) is used for the two European languages; a CJK environment together with proper encoding switches is inserted automatically by the output encoding; accentuated characters are translated into \TeX macros also without any additional work.

It’s not necessary to open a CJK environment inside of the document (it can even cause errors). Nor is it necessary to load the CJK package itself. The output filter uses \texttt{\textbackslash RequirePackage} to load CJK and \texttt{\textbackslash AtBeginDocument} to start a CJK environment with empty arguments; proper CJK \texttt{\textbackslash macros} are inserted immediately before an encoding change. The most convenient way to specify CJK font shapes is then to use \texttt{\textbackslash CJKencshape} in the preamble.

Now consider the word ‘Dvořák’ as an illustration of the hidden mechanism of the interface. Mule’s internal representation of this word is ‘Dvo\textasciitilde82\textasciitilde f8\textasciitilde e1k’ (\texttt{\textasciitilde xx} denoting real 8-bit values); ‘82\textasciitilde f8\textasciitilde e1’ is a leading byte representing the Latin-2 character set, ‘\textasciitilde f8’ and ‘\textasciitilde e1’ are the ‘i’ and ‘u’ in Latin-2 encoding. After applying the ‘cjk-coding’ output encoding (as defined in cjk-enc.el), the Czech name looks like this: ‘Dvo\textasciitilde8051\textasciitilde ffr\textasciitilde ff\textasciitilde8020\textasciitilde ffa\textasciitilde ffk’. The active character ‘80\textasciitilde’ has basically the following definition (in MULEenc.sty)

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|}
\hline
Unicode & UTF 8 byte 1 & UTF 8 byte 2 & UTF 8 byte 3 \\
\hline
00000000 & 00b5 0b3b 0b3b 0b3b 0b3b & 00b5 0b3b 0b3b 0b3b 0b3b & \texttt{---} \\
\texttt{(U+0000 – U+007F)} & \texttt{---} & \texttt{---} & \texttt{---} \\
00000b10 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b & \texttt{110b10 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b} & \texttt{10b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b 0b3b} & \texttt{---} \\
\texttt{(U+0080 – U+07FF)} & \texttt{---} & \texttt{---} & \texttt{---} \\
b3b15b14b13b12b11b10b9b8b7b6b5b4b3b2b1b0 & \texttt{1110b3b15b14b13b12b11b10b9b8b7b6b5b4b3b2b1b0} & \texttt{10b5b4b3b2b1b0} & \texttt{---} \\
\texttt{(U+0800 – U+FFFF)} & \texttt{---} & \texttt{---} & \texttt{---} \\
\hline
\end{tabular}
\caption{The UTF 8 representation of Unicode}
\end{table}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure3.png}
\caption{The UTF 8 representation of Unicode}
\end{figure}

\begin{figure}
\centering
\includegraphics[width=\textwidth]{figure4.png}
\caption{The Unicode character U+9038 in Japanese, Chinese, and Korean form (from left to right)}
\end{figure}
The CJK package for \LaTeX\ — Multilingual support beyond babel

\begin{verbatim}
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[german,czech]{babel}
\CJKencoding{JIS}\{song\}

\begin{document}

Palouk na kráji jezera. Kolem lesy, v nich na břehu jezerachalupa
Ježíček. Měsíc svítí. Na staré vrbě, jež se sklání k jezeru, sedí
Rusalka, smutně zamyšlena.

\selectlanguage{czech}

Wiesengrund am Ufer eines Sees. Ringsum Wälder; unweit des Seeufers
die Hütte der Waldhexe. Der Mond scheint. Auf einer alten Weide, die
sich zum See neigt, sitzt Rusalka in traurigem Sinnen.

\selectlanguage{german}

Wiesengrund am Ufer eines Sees. Ringsum Wälder; unweit des Seeufers
die Hütte der Waldhexe. Der Mond scheint. Auf einer alten Weide, die
sich zum See neigt, sitzt Rusalka in traurigem Sinnen.

Some additional notes:

- Mandarin Chinese has basically four tones (\textit{sì shēng} 四声) but sometimes it is referred to have a fifth, unstressed one. In the Chinese syllable script \textit{zhìyīnhūà} (注音符號) this fifth tone is indicated with a dot, but has no corresponding tone mark in pīnyīn. On the other hand, the first tone will be marked with a horizontal line in pīnyīn but remains unmarked in zhìyīnhūà.

With the pīnyīn package you can write e.g. ‘\texttt{\textbackslash m\textbackslash a5}’ to emphasize that you really mean an unstressed syllable — the result is equal to ‘\texttt{ma}’.

- In [16] a different approach to writing pīnyīn has been described; ligatures are used to compose vowels with tone marks, e.g. ‘\texttt{nu:v\textbackslash e\textbackslash r}’ to get nǚ\textbackslash e\textbackslash r’\textsuperscript{16} (女, the Chinese word for daughter). The advantage is the avoidance of any macros to produce the accented letters. The disadvantage is that you need virtual fonts to

\textsuperscript{16} The quote character is used to denote the syllable boundary in ambiguous cases.

\end{document}
\end{verbatim}

which will always be \texttt{\textbackslash input} at the very beginning
of the output file):

\begin{verbatim}
\def\^\texttt{80#1}\texttt{ff#2}\texttt{ff}{{...}}
\end{verbatim}

The first parameter is an index to an accent
macro, the second is the letter (or character macro)
to be modified. A number is used as the index;
this has the advantage that it works in verbatim
environments equally well as in case modifying com-
mands. After expanding the macros \texttt{\textbackslash mule\textbackslash 051} and
\texttt{\textbackslash mule\textbackslash 020} (defined with \texttt{\textbackslash csnname}) are called which
finally expand to \texttt{\textbackslash v} and \texttt{\textbackslash \textbackslash v}. Great care has been
taken to assure that really only expansion occurs to
retain kerning.

Besides the common Latin and CJK character
sets the Mule interface supports Vietnamese ([7]); it
is planned to extend it to Thai and Russian soon.

Other tools

The pīnyīn package. This style file (which can be
also used with plain \TeX) enables the input of pīnyīn
syllables with tones. An example of its usage was
given on page 215.

\textbf{Figure 5:} A screen snapshot of Mule
realize the ligatures which are not as easily in-
stalled as a macro package because you have to
create them first if you want to use a new font.

The ruby package. To cite Martin Dürst\footnote{His email address is mduerst@ifi.unizh.ch.} who wrote a proposal for ruby in HTML documents:

Ruby are small characters used for annotations of a text, at the right side for vertical text, and atop for horizontal text, to indicate the reading (pronounciation) of ideographic characters.

The name ruby is the name of the 5.5 point type size in British terminology; this was the size most used for ruby.

Ruby are in most cases set at half the size of the main letters, resulting in a possible two ruby characters per main character, and taking up half of the width of the main characters. However, at least up to five ruby characters per main character are possible (an example is \texttt{u-ke-ta-ma-wa-ru} (承\ruby{\texttt{る}}, to listen respectfully), and so various solutions, from leaving white space in the main text to having the ruby overlap the next characters of the main text, are possible (the latter is possible in Japanese especially because in many cases, the characters around an ideograph with ruby are syllabic, and therefore the assignment of ruby to main characters poses no problems for the reader).

Ruby are particularly frequent in Japanese, because of the way CJK ideographs are used in Japanese. Ideographs can have many different readings (pronounciations) because different readings were taken over from different regions of China and at different times when the characters where adopted in Japan. Also, these characters are used to write indigenous Japanese words, and many readings may be possible because the ideograph might cover many different concepts distinguished in the Japanese language. \[\text{\ldots} \]

The main use of ruby today is in magazines of all levels, and of course in educational material. Ruby are also used in educational material in China and Taiwan.

In Japan, the term \texttt{furigana} (ふりがな) is also used instead of ruby. ‘Furigana’ is composed of the verb \texttt{furu} (付\ruby{\texttt{る}}, to attach, sprin-
kle, \ldots) and \texttt{gana} (仮名, either hiragana or katakana, one of the two Japanese syllabaries usually used for ruby).

The ruby\footnote{His email address is mduerst@ifi.unizh.ch.} of the above citation has been input as \texttt{\ruby{\texttt{る}}\ruby{\texttt{れる}}}; the first parameter is the base character and the second the ruby itself.

To avoid lines sticking together the ruby package sets \texttt{\lineskiplimit} to 1 pt. It may be necessary to increase this value for larger font sizes.

Whether a ruby overlaps with the surrounding characters or not can be controlled with the overlap and nonoverlap options. There are a number of possibilities how ruby can interact with other CJK characters in both cases.

\begin{itemize}
  \item The ruby has a smaller width than its base character: the behaviour is identical to an ordinary CJK character.
  \item The ruby has a greater width than its base character:
    \begin{itemize}
      \item Overlapping ruby:
        \begin{itemize}
          \item If the previous or next character is a CJK character (ordinary or punctuation), insert unbreakable glue between.
          \item If the previous or next character is a ruby, handle both ruby as non-overlapping and insert unbreakable glue between.
          \item A ruby at the beginning of a paragraph will be treated as if the nonoverlap option had been set. To force an overlapping ruby you have to start the paragraph with a \texttt{\leavevmode} command.
        \end{itemize}
      \item Non-overlapping ruby: if the previous or next character is a CJK character (ordinary or punctuation), insert unbreakable glue between.
    \end{itemize}
\end{itemize}

\texttt{ruby.sty} introduces a third variation of a small kern (3 sp) to inform the next CJK or ruby macro that the previous character was an overlapping ruby with the ruby’s width greater than its base character. The global variable \texttt{\ruby@width} then contains this width.

The interface to the koma-script package

One of the greatest deficiencies of the current implementation of the standard document classes is the inflexibility in handling captions which follow non-English conventions. But even English captions can cause trouble if they are non-standard. Consider “Chapter Two” vs. “Second Chapter”: the former

\[\text{\ldots} \]
is supported, the latter isn’t. With ‘supported’ I mean that it is not necessary to rewrite any internal \TeX commands, and that even a novice user can change it. Besides this, many languages, especially in Asia, follow different conventions how to numerate sections, figures etc. A Chinese example: ‘Chapter 5’ is ‘第五章’, ‘第’ (di) is a prefix which converts the following Chinese number from a numeral into an ordinal, ‘第五’ (diwù) thus means ‘the fifth’. ‘章’ (zhāng) means chapter.¹⁸

Markus Kohm, the maintainer and developer of the koma-script package ([5]), has extended the package’s document classes with a flexible hierarchical captioning model which consists of three levels, one level deeper than in standard \TeX.

Level 1 are the well known standard macros \tablename etc. Language specific packages or options usually replace the English names with the right ones.

Level 2 is the modification of sectioning counters like \thesetitle. In the above example Chinese numbers should be used instead of Arabic digits.

Level 3 finally enables full control over the exact placement of spaces, counters, and other text in captions. All macros of this level have ‘format’ as postfix, e.g. \chapterformat; they are directly used by \chapter, \section etc.

A simplified definition for a Chinese chapter heading macro would be:

\newcommand\CJKnumber[1]{
  \ifcase#1\or
  一\or二\or三\or四\or五\or
  六\or七\or八\or九\or十\fi
}\newcommand\prechaptername{第}
\newcommand\postchaptername{章}
\renewcommand\thechapter{
  \CJKnumber{\value{chapter}}
  \prechaptername
  \postchaptername
}\renewcommand\chapterformat{\thechapter}

The CJK package supports this interface and provides caption files for Chinese, Japanese, and Korean. To activate, say \CJKcaption{xxx} inside of a CJK (or CJK*) environment; then the language module xxx.cap will be loaded. The names of the modules usually mirror the encoding, e.g., the Chinese caption file in Big 5 encoding is named Bg5.cap.

**Conclusion**

The CJK package works best if you write a document in a non-CJK language as the main language. Many typographic features needed for native CJK script support can’t be handled automatically due to limitations in \TeX itself (the abovementioned shibukai problem, vertical typesetting,¹⁹ and others).

Another not yet mentioned problem is speed. Due to the many (sub)fonts you have to change the font for almost each character—if your document consists entirely of, say, Chinese, you have more than enough time to drink a cup of coffee to format 50 pages on a moderately fast computer.

The future for CJK multilingual text processing with \TeX is definitely Ω, but until someone will have found time to provide CJK support (it is highly probable that this person is me again), it may not be the worst choice to use the CJK package meanwhile.

**References**


¹⁸ Often it is written like ‘第五章’ (第\五\章) in chapter headings, but the form without spaces is used in the table of contents.

¹⁹ Something which I’ve almost forgotten to say: CJK contains an experimental package for vertical typesetting with Big 5 encoded characters.