About XeTeX

Jonathan Kew

17 October 2005

Contents

1 Introduction

XeTeX[1] is an adaptation for Mac OS X of the TeX typesetting system developed by Professor Donald E. Knuth of Stanford University. If you are not familiar with TeX, you will need to obtain and study the definitive manual for the system, Knuth's *The TeXbook* (published by Addison Wesley), or one of the many other works on

[1] Normally pronounced as if it were written *zee-TeX*.

TEX, before you will be able to do much with XETEX. The following brief notes discuss only the specific details of the XETEX system, and in particular the ways it differs from standard TEX; much of what follows will make little sense without prior familiarity with TEX.

The XETEX web site, where the latest information and downloads can be found, is at http://scripts.sil.org/xetex.

Questions and comments concerning XETEX may be sent to the author at the email address mailto:nrsi@sil.org, or raised on the XETEX mailing list, hosted by the TEX Users Group. This is a good place to find information and help, as documentation is currently very limited. See http://tug.org/mailman/listinfo/xetex for details.

 It is a successor to the TEXGX program that was developed for earlier versions of Mac OS, using the now-obsolete QuickDraw GX technology.

2 Unicode support

While TEX is an 8-bit system, processing text encoded as a stream of single bytes, and accessing fonts using single-byte character indexes, XETEX is designed to work with Unicode text. The fundamental "characters" that XETEX works with are 16-bit codes that are assumed to represent the UTF-16 encoding form of Unicode (so characters in Planes 1–16 are represented as surrogate pairs, two XETEX code units each).

The most common form of Unicode text used with XETEX is UTF-8; this is also the encoding form used for all text output files (the `.log` file and any `\write` output). XETEX reads UTF-8 input files and converts the byte sequences into Unicode character codes automatically (*without* the use of additional TEX macros such as the LATEX `inputenc` package). It will also automatically read files using UTF-16, should this be necessary.

Because XETEX works with UTF-16 code units, TEX commands that deal with character codes, such as `\char`, `\catcode`, `\lccode`, etc., have been extended to handle 16-bit values (up to 65535, or `"FFFF`). Note that it is *not* possible to assign individual character properties such as `\catcode` to non-Plane 0 Unicode characters, because these are treated as a pair of surrogate codes; however, there is probably little reason to need to do this. Supplementary-plane characters can still be treated as normal text to be typeset.

It is also possible for XETEX to convert external files in non-Unicode encodings into its internal Unicode representation as they are read; see section 5.5 below.

3 Font access

The second main feature of XETEX that sets it apart from TEX is the ability to easily use Unicode-compliant fonts and work with both Mac OS X's advanced typographic services (AAT: Apple Advanced Typography) and OpenType layout fea-

tures. In general, any font installed in the operating system's various Library/Fonts folders is directly available for use in a XeTeX document; there is no need for `.tfm` files as used by standard TeX.

This brings great flexibility in font use; any font available on the Mac OS X system can immediately be used in XeTeX, including whatever "smart" AAT or OpenType features it may offer. However, documents which use such "native" OS X fonts will typically not be portable to standard TeX systems, and even if `.tfm` files for the "same" fonts are available on another system the results will almost certainly not be identical. (XeTeX was created primarily to typeset text in complex non-Roman scripts, implemented using AAT or OpenType fonts, and for this purpose it works well; many "normal" TeX jobs may be better done with a standard version of TeX.)

To provide this flexibility, XeTeX reinterprets TeX's `\font` command slightly. When a `\font` command is read, it attempts to locate an installed font in Mac OS X with the given name. If one is found, then this font will be used in either ATSUI or OpenType mode, making use of layout features that may be included in the AAT/OT font tables, and without reference to any `.tfm` file. (Note that because such font names may include spaces or other characters that would not normally be considered part of a filename by TeX, they can be quoted by enclosing them in single quotes, double quotes, or parentheses.)

In order to provide some compatibility with the "standard" fonts of the TeX world, such as Computer Modern, XeTeX can also read `.tfm` files, and when using a font for which a `.tfm` file is present it should normally provide the same results as any standard TeX. In particular, math mode requires the use of `.tfm`-based fonts.

If no installed font is found for a `\font` declaration, XeTeX will attempt to load a `.tfm` file and perform standard `.tfm`-based typesetting with this font. Note that for the font to actually be visible in the final output, it must be available as a `.pfb` or `.otf` file where the `xdv2pdf` tool can find it, based on the `dvips.map` font map file and `texmf.cnf` path settings, or else installed as a regular font in OS X.

Note that a document may freely mix the use of `.tfm`-based, AAT, and OpenType fonts.

3.1 AAT fonts

XeTeX can take advantage of most advanced features of AAT fonts, such as contextual variants, complex ligatures, alternate swash forms, variation fonts, etc.

To allow access to the advanced features of AAT fonts, the "font name" in a `\font` declaration for a non-`.tfm`-based font may include additional information. The font name specified should be the full name of a font, as seen by AAT (this may sometimes differ from the name in most Font menus). It may be enclosed in quotes (this also applies to file names, such as in `\input` commands); this allows names to include spaces and other "special" characters. In ATSUI typesetting mode, XeTeX will automatically use the default features of an AAT font; this may include automatic ligatures, contextual swash forms, and so on.

To control the exact set of features used, a `\font` command may include fea-

ture settings after the font name. To do this, you add a colon after the font name, and then a semicolon-separated list of *<feature>=<setting>* pairs. To turn *off* a feature that is enabled by default, put an exclamation mark ("not", in various programming languages) before the setting name. The allowable features and settings are font-dependent; the Typography palette in applications such as TextEdit lets you see what features any given font supports, or the file `AAT-info.tex` shows how XeTeX itself can determine the available features. In the same way, *<variation>=<value>* pairs may be used to control the style of variation fonts such as Skia. (All this is easier illustrated than described; look in the `FontSamples.tex` file to see how it works.)

3.2 OpenType fonts

Starting with release 0.8, XeTeX includes (preliminary) support for complex layout features in OpenType fonts. This support is based on code from the ICU (International Components for Unicode) library.[2] The OpenType feature support is currently incomplete, and the TeX macro interface as well as details of the internal implementation are liable to change in future releases. See the OpenType sample files provided (in the Sample Files archive) for hints on how to work with OpenType fonts in XeTeX.

3.3 Font style options

Any installed font may be accessed by using its full name (the "display name" as shown by Font Book, for example) in a `\font` command. However, it is also possible to access the bold and italic faces within a font family without giving their precise

[2] ICU License - ICU 1.8.1 and later

COPYRIGHT AND PERMISSION NOTICE

Copyright (c) 1995-2003 International Business Machines Corporation and others

All rights reserved.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF THIRD PARTY RIGHTS. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR HOLDERS INCLUDED IN THIS NOTICE BE LIABLE FOR ANY CLAIM, OR ANY SPECIAL INDIRECT OR CONSEQUENTIAL DAMAGES, OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Except as contained in this notice, the name of a copyright holder shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization of the copyright holder.

All trademarks and registered trademarks mentioned herein are the property of their respective owners.

names; if the modifiers `/B` and/or `/I` are added to the name of the regular font, XeTeX will find and load the associated bold or italic style (if available). Similarly, `/BI` (or `/B/I`) will load the bold-italic style.

This feature makes it simpler to load the basic styles of a font family, without having to know the exact names in advance. For example:

Name specified	*Font used*
`"Palatino/B"`	**"Palatino Bold"**
`"Times Roman/B"`	**"Times Bold"**
`"Times Roman/I"`	*"Times Italic"*
`"Hoefler Text/BI"`	***"Hoefler Text Black Italic"***

Of course, in the case of families with additional weights (light, book, semi-bold, bold, ultra, etc.) it is still necessary to use full names to select the exact face desired from among all those available; the `/B` modifier will simply find whatever ATS considers to be the default bold face.

3.4 Optical sizes

With release 0.94 of XeTeX, there is built-in support for optically-sized faces in font families, such as some of the Adobe Pro fonts. If a font family has optically-sized faces, with the OpenType `size` feature tag associating them, then XeTeX will, by default, load the most appropriate face for the font size requested. Thus, with the Brioso Pro Opticals installed:

Name and size requested	*Font used*
`"Brioso Pro" at 7pt`	"Brioso Pro Caption" at 7.0pt
`"Brioso Pro" at 11pt`	"Brioso Pro" at 11.0pt
`"Brioso Pro" at 16pt`	"Brioso Pro Subhead" at 16.0pt

It is possible to override this behavior by adding a `/S=#` modifier to the font name, where # is a specific optical size (in points). This will load the face appropriate to the given optical size, regardless of the actual size to which the font is being scaled. A size of 0 may be used to disable optical sizing altogether, simply loading the named font. Thus:

Name and size requested	*Font used*
`"Brioso Pro/S=7" at 9pt`	"Brioso Pro Caption/S=7" at 9.0pt
`"Brioso Pro/S=10" at 9pt`	"Brioso Pro/S=10" at 9.0pt
`"Brioso Pro/S=18" at 9pt`	"Brioso Pro Subhead/S=18" at 9.0pt
`"Brioso Pro/S=36" at 9pt`	"Brioso Pro Display/S=36" at 9.0pt
`"Brioso Pro Subhead" at 9pt`	"Brioso Pro Caption" at 9.0pt
`"Brioso Pro Subhead/S=0" at 9pt`	"Brioso Pro Subhead/S=0" at 9.0pt

(Incidentally, this also illustrates that some fonts may be known by multiple names; "Brioso Pro" is also called "BriosoPro-Regular", for example, and "Brioso Pro Subhead" is also "BriosoPro-Subh". This seems to occur primarily with OpenType CFF fonts, where Apple Type Services apparently sees the PostScript font name as well as the "full" name. XeTeX will accept either form.)

4 Output

Because XeTeX is so closely tied to the AAT font machinery, the concept of a DVI (device-independent) output file is not very useful—a typeset XeTeX document is inherently system-specific. If any XeTeX features (AAT fonts or graphics inclusion; see below) are used, the source document will not work on a standard TeX system.

XeTeX is implemented as a command-line tool, `xetex`, similar to standard Unix-based TeX tools. The `xetex` tool writes a form of "extended DVI" (`.xdv`) data to describe the typeset pages. A separate tool, `xdv2pdf`, can read this `.xdv` data and convert it to PDF, which may then be viewed or printed on any platform.

By default, `xetex` automatically executes `xdv2pdf` and pipes its `.xdv` output to the converter; therefore, in effect the default output format is PDF. It is possible to suppress the conversion and generate an `.xdv` file instead by specifying the `-no-pdf` option to `xetex`. This is generally much faster, as the PDF generation process is relatively slow; but the `.xdv` file cannot be directly viewed or printed. However, when multiple typesetting runs are required (such as with LaTeX and similar formats), it is more efficient to specify `-no-pdf` for all runs except the last.

4.1 `\special` commands

4.1.1 Color

The `xdv2pdf` driver supports several `\specials` to allow color to be used in XeTeX documents. These are:

```
\special{x:textcolor=color}
\special{x:textcolorpush}
\special{x:textcolorpop}
\special{x:rulecolor=color}
\special{x:rulecolorpush}
\special{x:rulecolorpop}
```

where *color* is an RGB color value expressed as 6 hexadecimal digits. If the color value is omitted, the color is reset to the default (which may not be black, in the case of ATSUI fonts, as `color=`*color* is a possible option in the `\font` command). The `push` and `pop` forms of the `\specials` allow color settings to be saved and restored using a simple stack mechanism, which may be simpler than explicitly restoring the previous color after a local change.

Beginning with release 0.7, color values may also be given as RGBA values with 8 hexadecimal digits, thus allowing transparency effects.

The `xdv2pdf` driver also supports a set of color `\specials` based on those used in drivers such as `dvips` and `dvipdfm`. These begin with the keyword `color`, followed by any of a number of color commands:

```
\special{color push}
\special{color pop}
\special{color rgb R G B}
\special{color cmyk C M Y K}
\special{color hls H L S}
```

```
\special{color hsv H S V}
\special{color hsb H S B}
\special{color gray Gray}
\special{color colorname}
```

The color value can be expressed according to a number of different color models, with one, three, or four values in the range 0.0–1.0, or one of the predefined color names known to the `dvips` driver can be given. In addition, any of these can be followed by the keyword `alpha` and an alpha-channel value in the range 0.0–1.0, for transparency. These `\special` commands set the color for both text and rules together.

4.1.2 Hyperlinks and outlines

While XeTeX does not support the PDF-specific extensions of pdfTeX, some PDF features can be accessed via `\specials` similar to those used by the `dvipdfm` driver. When it generates a PDF file, `xdv2pdf` writes an auxiliary file with the extension `.marks` that contains a record of all `\special` commands that begin with the text `pdf:`. Each such `\special` is written with information on its exact location (page number and (x, y) coordinates). A Perl script `xdv2pdf_mergemarks` is then automatically executed, and has the opportunity to integrate information from the marks file into the main PDF document.

The `xdv2pdf_mergemarks` script attempts to handle the `\specials` used by the `dvipdfm` driver to implement document outlines (bookmarks) and some kinds of hyperlinks, to support the LaTeX `hyperref` package with the `dvipdfm` output option. The `.marks` file is deleted after processing; to retain it for inspection or further processing, modify the script to remove the `unlink` command.

4.1.3 Transforms and other effects

The `xdv2pdf` driver also supports `\specials` for modifying the transformation matrix used in producing the PDF output. This allows rotation, reflection, scaling and skewing of arbitrary TeX output. The available commands are:

```
\special{x:gsave}
\special{x:grestore}
\special{x:scale X Y}
\special{x:rotate D}
\special{x:shadow(X,Y),blur}
\special{x:colorshadow(X,Y),blur,color}
\special{x:backgroundcolor=color}
```

The `transforms.tex` sample file demonstrates use of most of these `\specials`.

5 TEX language extensions

5.1 Identifying the XETEX version

There are several additions to the TEX language in XETEX. Beginning with release 0.5, there are commands `\XeTeXversion` (a number) and `\XeTeXrevision` (a string, beginning with a period) that make it possible for a document to detect which version of the `xetex` processor is running. (This document was typeset using XETEX version 0.997-dev.)

5.2 ε-TEX features

XETEX includes the TeX--XeT extension from ε-TEX, which provides the `\beginL`, `\endL`, `\beginR`, and `\endR` primitives to control bidirectional typesetting (for Arabic, Hebrew, and similar languages). Note that XETEX is also affected by the inherent directionality of Unicode characters; therefore, it is not actually necessary to use `\beginR...\endR` around a word in an Arabic font, for example, to have it appear correctly. However, these commands are required for the overall layout of right-to-left or bidirectional paragraphs to work properly. (Note that `\TeXXeTstate=1` is required to enable these commands.)

Other ε-TEX features such as additional registers (beyond 255) and primitives are also available in XETEX; see the documentation of ε-TEX for details. A short reference manual can be found at http://www.staff.uni-mainz.de/knappen/etex_ref.html.

5.3 Graphics support

To support graphics inclusion, XETEX provides the `\XeTeXpicfile` command. Normally, in other TEX systems, this is done with the `\special` command. However, `\XeTeXpicfile` offers the advantage that XETEX can determine the size of the picture, and thus macros can be written to fit the picture to a certain space or create the needed space for the picture, bypassing the manual picture-measuring process otherwise needed. Specifically, `\XeTeXpicfile` is used as follows:

```
\XeTeXpicfile <filename> [options]
```

where <*filename*> is the name (full pathname, or relative to the main document) of a graphics file. Most graphic file formats are supported (using QuickTime's Graphic Importer facility).

The [options] in the `\XeTeXpicfile` command use the following keywords:

`width` <dimen>
`height` <dimen>
`scaled` <scalefactor>
`xscaled` <scalefactor>
`yscaled` <scalefactor>
`rotated` <degrees>

Scaling and rotation operations are executed in the order specified. If one of `width` or `height`, but not the other, is specified at any stage, the other dimension is scaled

in proportion. If both absolute size and `scaled` are used at the same stage in the transformation sequence, not separated by a rotation, the size is set as requested and the scale factor is ignored.

The final result of a `\XeTeXpicfile` command is an object which can be included in the text being typeset just like a character, albeit usually a rather large one. It could be thought of as a box, though it cannot be unboxed. It can, however, be put inside a normal TEX box register, and in fact this is often desirable. After saying, for example,

```
\box0=\hbox{\XeTeXpicfile "my-pic.jpg"
      scaled 750 rotated 45}
```

you can find the dimensions of the graphic by examining the width and height of box 0. (The depth will always be zero.) Your macros can then arrange to place the picture suitably on the page. (See the `PicFileSample.tex` file.)

There is also a `\XeTeXpdffile` command. This is similar to `\XeTeXpicfile`, but recognizes an additional option:

`page` <number>

This specifies the page to be included from the PDF file (default is the first page). Negative numbers can be used to indicate pages relative to the end of the file; thus `page -2` means the second to last page. The `page` option, if used, must be specified as the *first* keyword after the filename, before any transformations.

Note that PDF files can also be included by the `\XeTeXpicfile` command, but this has two significant drawbacks: they will be rendered as lower-resolution graphics, and only the first page is available.

In versions before 0.5, these commands were named `\picfile` and `\pdffile` respectively, and were always enabled. Beginning with release 0.5 of XHTEX, the command names have been changed to reduce the likelihood of clashes with names used in macro packages or other extensions. In addition, these commands (like other ε-TEX extensions) are only available when the XHTEX engine is running in "extended" mode (activated by an initial `*` when the format file is created). Note that the format files included with XHTEX have extended mode active, so these commands will normally be available.

For use with LATEX, there is a "driver" file `xetex.def` that maps features from the standard `graphics` packages onto these XHTEX primitives.

5.4 Font information primitives

New in XHTEX 0.7 are a set of commands that allow authors or macro writers to query AAT fonts to determine the available font features, their names, and default settings. Pending real documentation, see the sample file `AAT-info.tex`, which uses these commands to generate a page listing the features supported by a given font. There is also a `\XeTeXglyph` command that allows an arbitrary glyph, identified by glyph ID number, to be rendered from any AAT font.

Similarly, release 0.84 and later includes commands that can determine the available script, language, and feature tags in OpenType fonts. These are illustrated by the `OpenTypeInfo.tex` sample file, which iterates over and prints all the tags

supported by a given font.

5.5 Input encodings

Although XeTeX is designed as a Unicode-based system, and access to standard OS X fonts is via Unicode character code values, it is possible to use input text files in other legacy encodings. By default, XeTeX examines each input file and reads it as either UTF-16 (if a UTF-16 Byte Order Mark is found, or if the presence of null byte values makes the text look like UTF-16 data), or else as UTF-8. Plain ASCII text is of course also valid UTF-8, so this can be read equally well with no further work.

However, if the input text is 8-bit data in an encoding such as Mac OS Roman, ISO Latin 1, etc., it will not be read correctly in this way; byte codes above 127 will be taken as beginning UTF-8 sequences, not as individual codes in their own right. And because this happens at the file-reading level, it is not possible to reinterpret the codes at the TeX macro level, as is done with legacy packages such as `inputenc` for LaTeX.

To overcome this, and allow existing 8-bit text to be read without requiring conversion to Unicode before running XeTeX, it is possible to switch the input encoding using:

`\XeTeXinputencoding "`*charset-name*`"`

(introduced in release 0.9). The *charset-name* here is the name of a character set recognized by the ICU library; see http://www.iana.org/assignments/character-sets for a large registry of names. Most of the expected forms such as `"mac"` (Mac OS Roman), `"cp1252"` (Windows codepage 1252, Western European), `"shift-jis"` (e.g., for Japanese), and many other codepage names and numbers should be recognized.

In addition, there is a special "encoding" called `"bytes"` which can be used for legacy support. After `\XeTeXinputencoding "bytes"`, the byte values 0–255 will be mapped directly to the internal XeTeX character codes 0–255. This will not generally be correct Unicode, but it should allow existing TeX macro packages that expect this range of character codes to be used for further interpretation.

The `\XeTeXinputencoding` command changes the interpretation of the *current input file* only; to be exact, it affects how text is read from the file, beginning at the *following* line. For a legacy-encoded file to be read correctly, the appropriate encoding command should be included near the beginning, with only standard ASCII characters preceding it.

An additional command `\XeTeXdefaultencoding` works similarly, except that instead of changing the mapping used to read the *current* input file, it changes the *initial* mode that will be used for newly-opened files. This includes files opened with `\openin` as well as `\input`. The default behavior may be restored with `\XeTeXdefaultencoding "auto"`.

5.6 Line-breaking without spaces

Normally, TEX will only break lines at inter-word spaces, or according to its hyphenation rules. A new feature introduced in XETEX version 0.91 is support for line-breaking in scripts such as Chinese that are written without inter-word spaces. Additional line-break positions in such scripts will be found if a *line-break locale* is specified (the rules may differ between locales). The XETEX primitives used are:

`\XeTeXlinebreaklocale "`*locale-id*`"`
`\XeTeXlinebreakskip =` *glue-specification*
`\XeTeXlinebreakpenalty =` *integer*

The locale ID is a POSIX standard identifier recognized by the ICU library; for example, `en_US` for the US English locale, or `th_TH` for Thai, or `zh_HK` for Hong Kong Chinese. Most locales simply use the default Unicode line-break rules based on character properties, so in practice even `\XeTeXlinebreaklocale "en"` is sufficient for East Asian line-breaking to work. Note that Thai, however, has additional rules that will only be activated if a Thai locale is specified.

The parameters `\XeTeXlinebreakskip` and `\XeTeXlinebreakpenalty` (both zero by default) control what happens at the locale-dependent intercharacter line-break locations. A typical `\XeTeXlinebreakpenalty` setting might be 100, so that there is a slight preference for line-breaks at spaces if possible. If the text is to be justified (and there are no "real" spaces available for stretching), then `\XeTeXlinebreakskip` should be given some stretchability; for example, `\XeTeXlinebreakskip=0pt plus 1pt`.

`\XeTeXlinebreaklocale` is a global setting, not affected by TEX's grouping. To disable such line-breaking and revert to standard TEX behavior, set the locale to an empty string: `\XeTeXlinebreaklocale ""`.

6 Limitations

The marriage of TEX and ATSUI is not without its problems; a few issues are mentioned here, though there are doubtless many others to be encountered. This is a "work in progress" and many features may change in future versions.

Versions of XETEX before version 0.3 (like all versions of the earlier TEXGX), did not support automatic hyphenation when using AAT-based fonts. This feature is now operational, using the standard TEX hyphenation algorithm. (Note, however, that non-English hyphenation tables will need to be converted to Unicode before they can be used.) This means that version 0.3 is liable to format paragraphs differently from earlier versions, even when using the exact same fonts, unless hyphenation is suppressed using TEX commands.

Math typesetting works as expected, but as math symbol and extension fonts must have additional metric information, math mode can only be expected to work with `.tfm`-based fonts. There has been further discussion of these issues on the XETEX mailing list.

Currently, line breaking and hbox construction in XETEX with non-`.tfm` fonts is based on the width of each word, measured individually by ATSUI. This means

that it is not perfect in cases where the width of a word changes significantly depending whether it is at a line end or in the middle of a line. We have not found this to be a serious problem in actual practice, but this would depend on the characteristics of the fonts used and the kind of material being typeset. In general, XeTeX's output is best when TeX is able to pass relatively long runs of text, rather than individual characters or words, along to ATSUI for output, so that ATSUI's spacing and justification can work with enough text at a time.

For standard TeX documents using only `.tfm`-based fonts, XeTeX should provide the same results as a standard version of TeX. However, it will normally be considerably slower than a standard TeX, so it is not particularly recommended unless you will be using its special features—in particular, the AAT capabilities. For completely standard TeX documents, a standard TeX implementation such as teTeX will be more efficient.