Language-Dependent Ligatures

John Plaice
Département d'informatique, Université Laval, Ste-Foy, Québec, Canada G1K 7P4
plaice@ift.ulaval.ca

Abstract

The concept of *character cluster* is introduced to TEX. Any sequence of letters can be clustered to form a single entity, which can subsequently be used as a single character. The standard example for a cluster is a letter with diacritic marks. Clusters are introduced by extending the TEX input language and by modifying the TEX program. They can be used to define ligatures within TEX, without passing through the `.tfm` files, thereby allowing different languages to be typeset with different ligatures. Clusters can be used in hyphenation tables, thereby eliminating the need to have precomposed characters in fonts to have correct hyphenation. A single 256-character font becomes suitable for typesetting all the world's languages that use the Latin alphabet.

The Latin alphabet is the most widely used alphabet in the world. Although at first glance, it has only twenty-six letters, a more careful look will show that these letters can be used with myriads of different diacritic marks, and that some languages actually use special characters above and beyond the original 26. If we consider all the possible precomposed characters, Haralambous (*TUGboat*, 1989) counts 190 different symbols above the original 26, not including the double-diacritic Vietnamese characters (close to 100 all by themselves). In addition, Jörg Knappen's font for African languages uses another 100 or so characters (Haralambous, 1992a). Finally, the ISO-10646 Universal Character Encoding Standard includes close to 900 precomposed Latin characters.

If all Latin precomposed characters are encoded separately, then several families of 256-character fonts would be needed. This would be an incredible waste of space, as most characters would be reencoded over and over again. Furthermore, most sites would end up stocking only one family of fonts, for the "important" languages (read English and West-European). On top of these considerations, the result would still not be technically desirable. It is standard typographic practice to place diacritical marks differently in different countries; in fact the marks may well look different from one country to another. Let's precompose all *those* combinations!

Another possibility is to use virtual fonts for every language. But, according to the Summer Institute for Linguistics's *Ethnologue*, there are over 6000 languages on this planet, and most of them use the Latin alphabet. To encode all the virtual fonts would literally require thousands of files, and few sites (if any) could afford the Brobdingnagian quantities of disk space required.

The only reasonable solution is to encode, in the fonts, diacritical marks separately from the letters that they are placed on. However, TEX currently only offers two ways to combine characters: ligatures and the `\accent` primitive. Ligatures do not (currently) do vertical placement and `\accent` has a fixed algorithm for character positioning. There is another alternative that can be considered: active characters. However, these can interact in strange ways with macros. Finally, preprocessors have a tendency to do strange stuff with macros.

In fact, what is needed is some sort of "active character" mechanism for *after* macro expansion, i.e., some sort of generalized ligature system that allows vertical, in addition to horizontal, displacement, and that does not require these ligatures to be encoded in the `.tfm` files.

In addition to not allowing vertical displacement, ligatures have their own problems. Jackowski and Ryćko (1992) discuss in detail the problems in using ligatures to access Polish characters with ogoneks. First, because implicit kerns and ligatures are defined one character at a time, it is very difficult, if at all possible, to correctly compose the characters and kern between these newly composed characters and their neighbors. Second, ligatures make it difficult for majuscule letters with diacritical marks to be given the appropriate space factor (`\sfcode`) codes. Third, if ligatures are used in hyphenation patterns, then the `\lefthyphenmin` and `\righthyphenmin` parameters do not work properly, as the individual

characters in the ligatures are counted individually, rather than collectively. The basic problem is the same in all these cases: the generated ligatures are not treated as a single character but, rather, as a sequence of individual characters.

It is not just a system of generalized ligatures that is required, but also the ability to treat these new entities as single characters. This is done in an extension of TEX called Ω, using *character clusters*, which do exactly what is required above. A character cluster is a sequence of letters that can be considered to be a single character. A character cluster can be given an `\sfcode`, `\lccode` and `\uccode`, and can be used in the definition of hyphenation patterns. The typesetting of a character cluster is defined by an arbitrarily complex sequence of TEX instructions. With this added functionality, it becomes possible, to the best of our knowledge, to have a single 256-character font that is sufficient for encoding all the world's languages that use the Latin alphabet.

Character Clusters

A (character) cluster is an ordinary sequence of TEX instructions, which can be considered, for kerning and hyphenation purposes, as a single character, along with a name. The name is just a sequence of characters.

Clusters are defined using the `\defcluster` command. For example,

```
\defcluster{e'}{\'e}
```

defines a possible sequence of instructions to type é (using the dc fonts). Similarly,

```
\defcluster{ij}{\char"BC}
```

gives the Dutch **ij** ligature in the dc fonts.

Clusters are used using the `\cluster` command. For example,

```
\cluster{e'}t\cluster{e'}
```

would give the French word for summer, 'été'. Of course, no one would want to type `\cluster` all the time. To reduce typing problems, we introduce a new syntactic form: $<c_1c_2\ldots c_n>$ is equivalent to `\cluster{`$c_1c_2\ldots c_n$`}`. The above word therefore becomes `<e't<e'>`.

The traditional ligatures of TEX are really just alternative presentations of the composing characters. These can be presented as follows: $<c_1c_2\ldots c_n|c_{n+1}c_{n+2}\ldots c_m>$ means that the cluster $<c_1c_2\ldots c_n>$ can be broken up by the hyphenation algorithm into the basic characters $<c_{n+1}c_{n+2}\ldots c_m>$. For example, the '**ffl**' ligature can be represented by `<ffl|ffl>`.

Context Dependent Analysis

Now even the word `<e'>t<e'>` is too much work to type. One should be able to type an ordinary stream of text and, with no special instructions inserted, have the appropriate clusters inserted into the text. Of course, what is appropriate will depend significantly on the language and the family of fonts being used.

In the case of our example, it should be possible to simply type `e'te'` and have the system handle the rest. To do this, the `Chief Executive` routine is modified so that it filters through a deterministic finite state automaton (DFA) all the text that is in horizontal mode. This DFA changes depending on the language being typeset, the typesetting rules in effect and the font families being used.

For example, in French, many words use the œ and æ ligatures that were common for the typesetting of Medieval Latin (Becarri, 1992). A careful perusal of the medium-size *Petit Robert* dictionary (Rey & Rey - Debove, 1984) allowed the definition of a Lex-like set of patterns that can be used to determine when ae and oe should form the ligatures and when they should not:

```
^oe              <oe>
oe/ll            oe
oe/[cilmrstu]    <oe>
^aethuse         aethuse
mae              mae
ae               <ae>
```

This set of patterns only considers common terms. For names and persons, the set is wrong: for example, many Flemish and Dutch proper names have unligatured ae's and oe's that would be ligatured by this set of patterns.

The line

```
oe/[clmrstu] <oe>
```

should be read as if the letters oe are followed by any of `clmrstu`, and then those two letters are replaced by the `<oe>` cluster. Therefore the word `oeil` becomes `<oe>il`. The `^` refers to the beginning of a word.

This same mechanism can be used to handle words with strange hyphenation. For example, the German words `backen`, `Bettuch`, `Schiffahrt` could be written as follows

```
^backen      ba\disc{k-}{k}{ck}en
^Bettuch     Be\disc{tt-}{t}{t}uch
^Schiffahrt  Schi\disc{<ff>-}{f}{ff}ahrt
```

where the `\disc` means a discretionary break. Similarly, the English word `eighteen` could become

```
^eighteen    eigh\disc{t-}{t}{t}een
```

Kerning

Kerning is a purely visual phenomenon, and so is handled after the context-dependent analysis has been made. In this situation, kerning becomes much simpler than the way that it is currently used in TEX and METAFONT. All that is required is to specify the kerning that must take place between any pair of characters or clusters. A Lex-like syntax is also used to specify these pairs.

The example given in the appendix gives the kerning in the `roman.mf` file for the Computer Modern family of fonts (Knuth, 1987), modified so that the letters **a**, **e** and **o** can each receive acute, grave, trema (or umlaut), and circumflex accents.

Hyphenation

Hyphenation patterns are no different from the old ones, except that they also allow clusters. However, it now becomes possible to have a period (`.`) in the middle of a pattern, as is required by some African languages, since the beginning of a word is marked by `^` and the end of a word by `$`. Accented characters are of course represented by clusters. For the purposes of `\lefthyphenmin` and `\righthyphenmin`, clusters are counted as a single character. The following example is an excerpt from the 8-bit hyphenation file `ghyphen3.tex` for German (Schwarz, 1990):

```
.kraf6
.k^^fc5ra
.lab6br
.liie6
.lo6s5k
.l^^f64s3t
```

which is replaced with:

```
^kraf6
^k<u">c5ra
^lab6br
^liie6
^lo6s5k
^l<o">4s3t
```

The second form is more readable and has the advantage of not being tied to a particular font encoding. It is now possible to separate the input encoding from the output encoding.

Handling 16-bit and 32-bit Input

Currently, there are many discussions about how to best handle 8-bit input. There are three major currently-used 8-bit extensions of ISO-646 (ASCII): ISO-8859-1 (Latin-1), Macintosh and IBM-PC. The users of all these systems would like to write using the characters that they have available. How are these needs to be reconciled with Ω? Furthermore, how should ISO-10646 and UNICODE 1.1 be handled? What is the relationship between these character encodings and the clusters?

The answer is quite simple. Internally, if a cluster has a corresponding encoding in ISO-10646, then that number is used for that cluster. However, if a cluster does not have a corresponding encoding, then a negative number is used (remember, ISO-10646 is really only a 31-bit encoding, and 'hi-bit on' is reserved for user-defined characters). The result is to separate input, internal and output encodings.

Implementation

At the time of writing, none of what has been proposed has been implemented, so many of the proposals are not finalized. Nevertheless, some detailed analysis has been completed. There are two parts to the implementation. First, the files defining the context-dependent analysis and the kerning must be translated into finite state automata readable by the `INITEX` program. Second, the TEX program must be changed to include a new data structure for clusters, new syntactic entities, as well as the basic routines which must be written or changed. Most of the work takes place in the `Chief Executive`.

Future Work

Character clusters are not just designed for the Latin alphabet. The same principles could be used to design compact Greek (Mylonas & Whitney, 1992) and Cyrillic fonts. More generally, typesetting Arabic (Haralambous, 1992a-c) or South Asian (Mukkavilli, 1991) scripts requires significant amounts of context analysis to choose the right variant of character and the correct set of ligatures to form a word and, in the case of Arabic, to correctly place vowels. Different solutions have been proposed, either using active characters, ligatures in the fonts or preprocessors, but none of them is sufficiently general. Character clusters should yield an elegant solution for these scripts.

Problems which have not been discussed here include the direction of text. At least four systems are used currently in different languages: left-right - top-down, right-left - top-down (Arabic, Hebrew, Syriac (Haralambous, 1991)), top-down - right-left (Japanese (Hamano, 1990)), and top-down - left-right (Mongolian (Haralambous, 1992a)). Some languages, such as Japanese, Mongolian and Epigraphical Greek, can be written in more than one direc-

tion. Clusters would be useful in implementing these questions.

Acknowledgements

The Ω project was devised by Yannis Haralambous and myself. It attempts to resolve some of the fundamental issues that have been raised during the discussions in the Technical Working Group on Multiple Language Coordination.

Bibliography

Beccari, Claudio. "Computer aided hyphenation for Italian and modern Latin". *TUGboat*, **13**(1), pages 23 - 33, 1992.

Hamano, Hisato. "Vertical typesetting with TEX". *TUGboat*, **11**(3), pages 346 - 352, 1990.

Haralambous, Yannis. "TEX and latin alphabet languages". *TUGboat*, **10**(3), pages 342 - 345, 1989.

Haralambous, Yannis. "TEX and those other languages". *TUGboat*, **12**(4), pages 539 - 548, 1991.

Haralambous, Yannis. "TEX et les langues orientales". Paris: INALCO, 1992.

Haralambous, Yannis. "Towards the revival of traditional arabic typography... through TEX". In *Proceedings of the 7th European TEX Conference*, pages 293 - 305. Brno, Czechoslovakia: Masarykova Universita, 1992.

Haralambous, Yannis. "Typesetting the Holy Qur'ān with TEX", 1992.

International Organization for Standardization. *Information technology — Universal Multiple-Octet Coded Character Set (UCS).* Draft International Standard ISO/IEC DIS 10646-1.2. ISO, 1992.

Jackowski, Bogusław and Marek Ryćko. "Polishing TEX: from ready to use to handy to use". In *Proceedings of the 7th European TEX Conference*, pages 119 - 134. Brno, Czechoslovakia: Masarykova Universita, 1992.

Knuth, Donald. *Computer Modern Typefaces.* Addison-Wesley, 1986, 1987.

Mukkavilli, Lakshmi V. S. TeluguTEX, 1991.

Mylonas, C. and R. Whitney. "Complete greek with adjunct fonts". *TUGboat*, **13**(1), pages 39 - 50, 1992.

Rey, A. and J. Rey - Debove, editors. *Le Petit Robert 1.* Dictionnaires Robert - Canada, 1984.

Summer Institute for Linguistics. *Ethnologue.* SIL, Santa Ana, CA (USA), 1988.

Schwarz, Norbert. "German hyphenation patterns". 1990.

Appendix:

```
as    a<a'><a`><a"><a^>
es    e<e'><e`><e"><e^>
os    o<o'><o`><o"><o^>
%%
k             {as}              \ifsfs-\ka\k\fi
v             {as}              \ifsfs-\ka0\fi
k             c{es}{os}         \k
v             c{es}{os}         \ifsfs\k0\fi
w             {as}c{es}{os}     \k
P             A                 \kb
yP            {as}{es}{os}      \k
yP            \.,               \kb
FVW           {os}{es}ur{as}    \ifsfs\kb\k\fi
FVW           A                 \ifsfs\kc\kb\fi
FKVWX         CGOQ              \k
T             y                 \ifsfs\k\kb\fi
TY            {as}{es}{os}ruA   \kb
OD            XWAVY             \k
hmn           btuvwy            \ifsfs\k0\fi
c             hk                \ifsfs\k0\fi
b{os}p        cd{es}{os}q       -\k
b{os}p        x                 \k
{as}b{os}p    v                 \ifsfs\k0\fi
{as}b{os}p    j                 \ifsfs\ka0\fi
{as}b{os}p    j                 \ifsfs0\k\fi
{as}b{os}pt   y                 \k
{as}b{os}ptu  w                 \k
A             tCGOQU            \k
R             tCGOQU            \ifsfs\k0\fi
AL            TY                \kb
R             RY                \ifsfs\kb0\fi
AL            VW                \kc
R             VW                \ifsfs\kc0\fi
g             j                 -\k
I             I                 -\k
```