Font Forum

Thai fonts

Werner Lemberg

Abstract

This article describes how the Thai script works and how to implement the necessary ligatures for TEX using `afm2tfm`.

1 Some Historic Information

The Thai script has been derived, similar to almost all other southern asian scripts not directly influenced by China, from an ancient version of Indic Sanskrit, Brahmi. Over the years, the original letters have been adapted to the peculiarities of the Thai language, one example would be superscripted digits used for tone marks.

Regarding computers, Thailand followed other principles than those of India in the standardization of the script despite many similarities; all base letters are consonants with an inherent vowel (usually an *a*), and for the special case of Thai, an inherent tone. Vowel and tone can be modified by attaching other letters to the base consonant — before and after the base consonant, but also above and below. The Indic standard (ISCII) uses the logical order to store text, which means that the vowel always comes after the consonant even if it appears before the consonant graphically (such vowels are called independent vowels). Contrary to India's logical order, Thailand defines its industrial standard TIS-620 in that independent vowels must be stored in visual order. Unicode [8] follows TIS-620 in the processing of the Thai script.

In the following, linguistic aspects will be completely ignored, referring to graphical features only.

2 The Structure of Thai Letter Clusters

Since letter clusters are stored in visual order the graphical display of Thai is simplified to base letters, possibly with diacritical signs above and below — no need to reorder vowels. There are five possibilities how diacritical signs can be positioned in Thai.

1. base consonant + vowel above:

   ด + ◌ี = ดี

2. base consonant + tone mark:

   ป + ◌่ = ป่

3. base consonant + vowel above + tone mark:

ป + ◌ี + ◌่ = ปี่

4. base consonant + vowel below:

ห + ◌ู = หู

5. base consonant + vowel below + tone mark:

ฎ + ◌ู + ◌่ = ฎู่

TIS-620 mandates that tone marks come last, but users sometimes ignore this. It is the duty of input methods for Thai to normalize incorrect input. Below, the standardized form is always expected.

It can already be seen in the above examples that diacritical signs change its positions horizontally and vertically dependent on the shape of the base glyph resp. whether another diacritical sign is used.

There are more peculiarities of the Thai script.

1. The vowel *sara am* ำ will be split into the characters *nikhahit* ◌ํ and *sara aa* า if it is appended to a consonant. The character ◌ํ interacts with the preceding character.

   ก + ำ = ก + ◌ํ + า = กำ

   ก + ◌่ + ำ = ก + ◌ํ + ◌่ + า = ก่ำ

   If necessary, *nikhahit* and the tone mark must exchange its positions.

2. The two consonants *yo ying* ญ and *tho than* ฐ drop its lower part if combined with a lower vowel.

   ญ + ◌ู = ญู

   ฐ + ◌ู = ฐู

3. If *sara aa* า follows the independent vowel *ru* ฤ or *lu* ฦ (those two letters are used for Sanskrit), it will be replaced by the sign *lakkhangyao* ๅ.

   ฤ + า = ฤๅ

   ฦ + า = ฦๅ

3 Glyph Classes

To describe the necessary ligatures it is convenient to categorize Thai letters into various graphical glyph classes, ignoring all linguistical aspects. In the author's opinion, even incorrect or unrealistic combinations should be displayed in an optically pleasing way if possible.

base$_{normal}$ Normal base glyphs without special features.

base$_{desc}$ Base glyphs with descender.

base$_{desclike}$ As described above, glyphs of this class consist of two parts, omitting the lower one if combined with a lower vowel.

base$_{asc}$ Base glyphs with an ascender on the right side.

base$_{indic}$ The two independent vowels *ru* ฤ and *lu* ฦ.

base$_{sign}$ The sign *lakkhangyao* ๅ.

base$_{sara\ am}$ The vowel *sara am* ำ.

base$_{sara\ aa}$ The vowel *sara aa* า.

lower Diacritical vowels below.

upper$_{vowel}$ Diacritical vowels above.

upper$_{sign}$ The sign *nikhahit* ◌ํ.

top Tone marks.

Now the glyph variant forms.

base$_{descless}$ The glyphs of class *base*$_{desclike}$ without the lower part.

lower$_{low}$ The glyphs of class *lower* shifted downwards.

upper$_{vowel_{left}}$ The glyphs of class *upper*$_{vowel}$ shifted to the left.

upper$_{sign_{left}}$ The glyphs of class *upper*$_{sign}$ shifted to the left.

top$_{left}$ The glyphs of class *top* shifted to the left.

top$_{low}$ The glyphs of class *top* shifted downwards.

top$_{low\text{-}left}$ The glyphs of class *top* shifted to the left and downwards.

4 Context Patterns

Using the glyph classes defined in the last section it is easy to describe the context patterns for base glyphs with diacritical signs. Surprisingly, these patterns are quite systematic. Patterns in table 1 which are marked with an asterisk do nothing and are listed for completeness only. As mentioned above, these patterns cover more combinations as existing in the Thai script.

Table 2 covers the ligatures of the character *sara am* ำ. Finally, table 3 describes the letters specific to Sanskrit.

5 Intermezzo 1

A small introduction into the exotic variants of TeX's ligature mechanism which probably many users haven't seen before. Additionally, the documentation in the *METAFONTbook* is very sparse. In the following examples METAFONT's notation is used.

base	*lower*			$\rightarrow$	*base*	*lower*			*
base		*upper*		$\rightarrow$	*base*		*upper*		*
base			*top*	$\rightarrow$	*base*			top_{low}	
base	*lower*		*top*	$\rightarrow$	*base*	*lower*		top_{low}	
base		*upper*	*top*	$\rightarrow$	*base*		*upper*	*top*	*
$base_{desc}$	*lower*			$\rightarrow$	$base_{desc}$	$lower_{low}$			
$base_{desc}$		*upper*		$\rightarrow$	$base_{desc}$		*upper*		*
$base_{desc}$			*top*	$\rightarrow$	$base_{desc}$			top_{low}	
$base_{desc}$	*lower*		*top*	$\rightarrow$	$base_{desc}$	$lower_{low}$		top_{low}	
$base_{desc}$		*upper*	*top*	$\rightarrow$	$base_{desc}$		*upper*	*top*	*
$base_{desclike}$	*lower*			$\rightarrow$	$base_{descless}$				
$base_{desclike}$		*upper*		$\rightarrow$	$base_{desclike}$		*upper*		*
$base_{desclike}$			*top*	$\rightarrow$	$base_{desclike}$			top_{low}	
$base_{desclike}$	*lower*		*top*	$\rightarrow$	$base_{descless}$	*lower*		top_{low}	
$base_{desclike}$		*upper*	*top*	$\rightarrow$	$base_{desclike}$		*upper*	*top*	*
$base_{asc}$	*lower*			$\rightarrow$	$base_{asc}$	*lower*			*
$base_{asc}$		*upper*		$\rightarrow$	$base_{asc}$		$upper_{left}$		
$base_{asc}$			*top*	$\rightarrow$	$base_{asc}$			$top_{low\text{-}left}$	
$base_{asc}$	*lower*		*top*	$\rightarrow$	$base_{asc}$	*lower*		$top_{low\text{-}left}$	
$base_{asc}$		*upper*	*top*	$\rightarrow$	$base_{asc}$		$upper_{left}$	top_{left}	

Table 1: Context patterns for diacritical signs. Here, *base* refers to the union of the subclasses *normal*, *indic*, *sign*, *sara am*, and *sara aa* of *base*; *upper* is the union of the subclasses *vowel* and *sign* of *upper*.

base	$base_{sara\ am}$		$\rightarrow$	*base*	$upper_{sign}$		$base_{sara\ aa}$
$base_{asc}$	$base_{sara\ am}$		$\rightarrow$	$base_{asc}$	$upper_{sign_{left}}$		$base_{sara\ aa}$
base	*top*	$base_{sara\ am}$	$\rightarrow$	*base*	$upper_{sign}$	*top*	$base_{sara\ aa}$
$base_{asc}$	*top*	$base_{sara\ am}$	$\rightarrow$	$base_{asc}$	$upper_{sign_{left}}$	top_{left}	$base_{sara\ aa}$

Table 2: Context patterns for *sara am* ำ. Here, *base* denotes the union of subclasses *normal*, *desc*, and *desclike* of *base*.

$base_{indic}$	$base_{sara\ aa}$	$\rightarrow$	$base_{indic}$	$base_{sign}$

Table 3: Context patterns for *ru* ฤ and *lu* ฦ.

The usual ligature action of two glyphs a and b is the replacement of both glyphs with another glyph c.

```
a b =: c
```

Another possibility is to retain the left or the right original glyph (before resp. after the ligature) or both.

```
a b |=: c   a b =:| c   a b |=:| c
```

The first rule creates ac, the second cb, and the last acb. In all three cases, the current point after appying the ligature rule is still at the first glyph of the replaced glyphs, and TEX simply restarts there to check ligatures (and kernings). A classical example is

$$\text{f f i} \rightarrow \text{ff i} \rightarrow \text{ffi}$$

To advance the current point to the right, append either `>` or `>>` (the latter is only possible if you retain both input glyphs). Here are the remaining four ligature rules.

```
a b |=:> c   a b =:|> c
a b |=:|> c   a b |=:|>> c
```

For Thai ligatures, the most often needed rule is `|=:` (i.e., retain the left glyph and stay at the same position before applying the next ligature rule). Note that using `|=:>` instead is not a good idea since this would prohibit kerning between the left glyph and the ligature.

6 Ligature Rules

As just explained, TEX can only handle context patterns of length 2, whereas Thai needs patterns of length 3. It was an interesting challenge to find out whether the problem can be solved with TEX's somewhat restricted ligature rules — the gentle reader is invited to find a solution by herself! There won't be any difficulties in understanding ligatures afterwards.

The tables 4, 5, and 6 use the same conventions as tables 1, 2, and 3, respectively. The current point isn't increased in any of the rules.

Most of the ligature rules can be derived easily by handling the patterns sequentially (quite similar to logic puzzles found in various magazines), but at the end there remain two patterns which apparently contradict.

$$\begin{array}{llll} base & lower\ top & \rightarrow base & lower\ top_{low} \\ base_{asc} & lower\ top & \rightarrow base_{asc} & lower\ top_{low\text{-}left} \end{array}$$

After applying ligature rules for the first two glyph classes it is necessary to handle the context '*lower top*', but depending on the previous glyph class *top* must be replaced with top_{left} and $top_{low\text{-}left}$, respectively. With a context pattern length of 3 this would be easy to solve, but TEX doesn't have this feature. What to do?

The context '*base lower top*' must be distinguished from '$base_{asc}$ *lower top*', i.e., two different *lower* classes are needed depending on the previous character since TEX is not able to forward information from one ligature cycle to the next. The idea is now to create an 'alias class', a class which behaves identically to the original one. The glyphs in this alias class are the same, but different glyph indices resp. glyph names are assigned to it. A closer look to table 4 shows that $lower_{left}$ isn't a typo but the alias class of *lower*.

7 Intermezzo 2

`afm2tfm` [6] uses two encoding vectors to create metrics files for TEX. The first maps from the Type 1 font to the *raw* font (converting glyph names to glyph indices):

```
/ps_to_raw [ ... /bar ... /bar ... ] def
```

The second encoding vector used for the *virtual* font (which will contain the ligatures) maps from glyph indices back to glyph names, so we can finally assign different glyph names to identical glyphs:

```
/raw_to_vf [ ... /bar1 ... /bar ... ] def
```

`bar1` and `bar` now both access the same glyph.

Unfortunately, `afm2tfm` can only use glyph names for the virtual font which are already in the base font, so some manual work is needed to overcome this restriction. It is planned to reimplement Thai ligatures using the `fontinst` package [2] which doesn't have this problem.

8 The Implementation

After solving the problem theoretically now the practical implementation. Only some interesting details are shown. Since it is not possible in `afm2tfm` to collect glyphs in classes the number of all ligature rules is quite big (464 in total). The file `thai.enc`, part of the `CJK` package [3], contains the complete solution. It also contains detailed installation instructions how to use `afm2tfm`.

All glyph names follow the *Adobe Glyph List (AGL)* [1]. There are no predefined Adobe glyph names for Thai, so the prefix 'uni' with attached Unicode value will be used for all glyphs which are encoded in Unicode. Example: The letter *ko kai* ก gets the name `uni0E01`. Glyph variants are identified by a postfix. Example: The left-shifted glyph variant of the vowel *sara uee* ◌ื is called `uni0E37.left`.

The following listing describes some of the ligatures, explaining its function.

$base$	top	$\rightarrow$	$base$	top_{low}
$base_{desc}$	$lower$	$\rightarrow$	$base_{desc}$	$lower_{low}$
$base_{desc}$	top	$\rightarrow$	$base_{desc}$	top_{low}
$base_{desclike}$	$lower$	$\rightarrow$	$base_{descless}$	$lower$
$base_{desclike}$	top	$\rightarrow$	$base_{desclike}$	top_{low}
$base_{asc}$	$lower$	$\rightarrow$	$base_{asc}$	$lower_{left}$
$base_{asc}$	$upper$	$\rightarrow$	$base_{asc}$	$upper_{left}$
$base_{asc}$	top	$\rightarrow$	$base_{asc}$	$top_{low\text{-}left}$
$lower$	top	$\rightarrow$	$lower$	top_{low}
$lower_{low}$	top	$\rightarrow$	$lower_{low}$	top_{low}
$upper_{left}$	top	$\rightarrow$	$upper_{left}$	top_{left}
$lower_{left}$	top	$\rightarrow$	$lower_{left}$	$top_{low\text{-}left}$

Table 4: Ligature rules for diacritical marks.

$base$	$base_{sara\ am}$	$\rightarrow$	$base$	$upper_{sign}$	$base_{sara\ am}$
$base_{asc}$	$base_{sara\ am}$	$\rightarrow$	$base_{asc}$	$upper_{sign_{left}}$	$base_{sara\ am}$
$upper_{sign}$	$base_{sara\ am}$	$\rightarrow$	$upper_{sign}$	$base_{sara\ aa}$	
$upper_{sign_{left}}$	$base_{sara\ am}$	$\rightarrow$	$upper_{sign_{left}}$	$base_{sara\ aa}$	
top_{low}	$base_{sara\ am}$	$\rightarrow$	top_{low}	top	$base_{sara\ am}$
top_{low}	top	$\rightarrow$	$upper_{sign}$	top	
top	$base_{sara\ am}$	$\rightarrow$	top	$base_{sara\ aa}$	
$top_{low\text{-}left}$	$base_{sara\ am}$	$\rightarrow$	$top_{low\text{-}left}$	top_{left}	$base_{sara\ am}$
$top_{low\text{-}left}$	top_{left}	$\rightarrow$	$upper_{sign_{left}}$	top_{left}	
top_{left}	$base_{sara\ am}$	$\rightarrow$	top_{left}	$base_{sara\ aa}$	

Table 5: Ligature rules for *sara am* ำ.

$base_{indic}$	$base_{sara\ aa}$	$\rightarrow$	$base_{indic}$	$base_{sign}$

Table 6: Ligature rule for *ru* ฤ and *lu* ฦ.

- Rule: $base\ top \rightarrow base\ top_{low}$

  This rule needs 225 ligatures (45 base glyphs × 5 tone marks); this is almost 50 % of all rules.

```
% LIGKERN uni0E01 uni0E48 |=: uni0E48.low ;
% LIGKERN uni0E02 uni0E48 |=: uni0E48.low ;
...
% LIGKERN uni0E01 uni0E49 |=: uni0E49.low ;
% LIGKERN uni0E02 uni0E49 |=: uni0E49.low ;
...
% LIGKERN uni0E41 uni0E4C |=: uni0E4C.low ;
% LIGKERN uni0E46 uni0E4C |=: uni0E4C.low ;
```

- Rule: $base_{desclike}\ lower \rightarrow base_{descless}\ lower$

  Here the left glyph will be replaced.

```
% LIGKERN uni0E0D uni0E38 =:| uni0E0D.descless ;
% LIGKERN uni0E10 uni0E38 =:| uni0E10.descless ;
...
```

- Rule: $base\ base_{sara\ am} \rightarrow$
  $base\ upper_{sign}\ base_{sara\ am}$

  The context pattern

$$a\ b \rightarrow a\ c\ d$$

  has to be transformed to the following for TeX (as shown in table 5):

$$a\ b \rightarrow a\ c\ b$$
$$c\ b \rightarrow c\ d$$

  The first ligature is of interest:

```
% LIGKERN uni0E01 uni0E33 |=:| uni0E4D ;
% LIGKERN uni0E02 uni0E33 |=:| uni0E4D ;
...
```

9 The Font Encoding

The real encoding of the virtual font is irrelevant for ligature rules because glyph names have been used exclusively. Nevertheless, it has practical advantages to use TIS-620 as a font encoding also, filling unused positions with glyph variants (this is similar to the Unicode++ font encoding of Ω [5]). Especially users of Plain TeX will benefit if input and output encoding are identical. Care must be taken in LaTeX documents to avoid the use of `\uppercase` and `\lowercase` commands so that Thai letters aren't modified due to incorrect `\lccode` and `\uccode` values.

Table 7 shows the used encoding of the Thai glyphs.

10 Problems

From a typographical point of view, all problems are solved. To believe that it is now possible to simply enter Thai for getting correct output is an error, though. Two serious obstacles must be mastered first: finding word breaks and insertion of intercharacter glue.

Words in Thai are *not* separated with spaces, and they aren't hyphenated either.[1] A space, usually much bigger than a space for the Latin script, has a similar function to an em-dash or to a semicolon; its primary use is to structure a sentence.

Correct recognition of words in Thai is a very complex problem which can be solved without errors by sentence analysis only. The CJK package uses a relatively simple algorithm developed by Vuthichai Ampornaramveth วุฒิชัย อัมพรอร่ามเวทย์ which basically searches for the longest words in a dictionary (this is implemented as a Lisp package for Emacs — it is assumed that the next major version of Emacs will contain this module directly). Due to missing context analysis it can't guarantee error-free results in all cases.

Another complication is that Thai tends to very long words, making the search for good break points in justified paragraphs difficult. A legitimate solution is to moderately apply some intercharacter glue (cf. figure 1). An even better solution would be the use of *Multiple Master* fonts or similar fonts to enable small variations of the typeface. Newer versions of pdfTeX have experimental support for stretching and squeezing of fonts [7].

11 Thai Support In The CJK Package

The results presented in this paper will be part of the next version of the CJK package (daily snapshots of the development archive are available from `ftp://ftp.ffii.org/pub/cjk/devel/cjk-current.tar.gz`). Included are (in addition to `thai.enc` and other auxiliary files) encoding, metrics, and font definition files for the freely available Thai font families DBThai and Norasi [4]. The latter uses glyphs created by Yannis Haralamous and Tereza Tranaka and are still under development; all Thai examples in this article have been typeset with it.

Intercharacter glue and word break points will be inserted automatically by the Emacs interface cjk-enc. This package could be considered as a generalized inputenc package which is able to handle multiple character sets in Emacs simultaneously, and which does correct translation to LaTeX transparently to the user.

12 Acknowledgements

Most of the development work on Thai ligatures has been done during a two-week research visit in Tsukuba (Japan) on invitation of the Electrotechnical Laboratory (ETL).

[1] For narrow-column printing, hyphenation is used, but it isn't considered as good typography.

	0	1	2	3	4	5	6	7
20	ุ	ู	ฺ	่	้	๊	๋	์
21	่	้	๊	๋	์		๎	ํ
22	ญ		ั	็	ิ	ี	ึ	ื
23	่	้	๊	๋	์			ฐ
24		ก	ข	ฃ	ค	ฅ	ฆ	ง
25	จ	ฉ	ช	ซ	ฌ	ญ	ฎ	ฏ
26	ฐ	ฑ	ฒ	ณ	ด	ต	ถ	ท
27	ธ	น	บ	ป	ผ	ฝ	พ	ฟ
30	ภ	ม	ย	ร	ฤ	ล	ฦ	ว
31	ศ	ษ	ส	ห	ฬ	อ	ฮ	ฯ
32	ะ	ั	า	ำ	ิ	ี	ึ	ื
33	ุ	ู	ฺ	‘	’	“	”	฿
34	เ	แ	โ	ใ	ไ	ๅ	ๆ	็
35	่	้	๊	๋	์	ํ	๎	๏
36	๐	๑	๒	๓	๔	๕	๖	๗
37	๘	๙	๚	๛	ุ	ู	ฺ	

Table 7: The Thai encoding of the CJK package. Range 241–373 (0xA1–0xFB) without 333–336 (0xDB–0xDE) is TIS-620, the rest are glyph variants. 241–316 are consonants. 320–332, 340–344, and 347 are vowels. 337 is the Thai currency symbol, Baht. 350–353 are tone marks. 360–371 are the digits 0 to 9. The alias class to *lower* (at position 330-332), $lower_{left}$, is at position 330–332.

รายการ FAQ นี้สร้างขึ้นเพื่อสรุปคำถามที่ถามกันบ่อยครั้งและคำ
ตอบคำถามในรูปแบบทีสะดวก. โครงสร้างของรายการ FAQ นี้เปลี่ยนไ
ปมากตั้งแต่รุ่นที่แล้ว. **ดูรายละเอียดสำหรับโครงสร้างใหม่ได้จากช่วง
“โครงสร้างและวิธีการอ่าน FAQ.”**

รายการ FAQ นี้สร้างขึ้นเพื่อสรุปคำถามที่ถามกันบ่อยครั้งและคำ
ตอบคำถามในรูปแบบทีสะดวก. โครงสร้างของรายการ FAQ นี้เปลี่ยนไ
ปมากตั้งแต่รุ่นที่แล้ว. **ดูรายละเอียดสำหรับโครงสร้างใหม่ได้จากช่วง
“โครงสร้างและวิธีการอ่าน FAQ.”**

Figure 1: The same text, with and without intercharacter glue. To suppress warnings and error messages for the above variant, `\tolerance` had to be set to 8000 and `\badness` to 10000. `\baselinestretch` has the value 1.2.

References

[1] The Adobe Glyph List. `http://partners.adobe.com/asn/developer/typeforum/unicodegn.html`.

[2] Allan Jeffrey et al. The fontinst package. Available from CTAN and its mirrors, e.g. `ftp://ftp.dante.de/pub/tex/fonts/utilities/fontinst`.

[3] Werner Lemberg. The CJK package. `http://cjk.ffii.org`.

[4] Surapant Meknavin and Theppitak Karoonboonyanan. The thailatex package. `ftp://opensource.thai.net/pub/linux-tle/updates/SOURCES/thailatex-0.2.1.tar.gz`. The implementation for Thai in this package is incompatible to the one described in this article. For this reason, the Babel module of the CJK package is called ‘thaicjk’ and not ‘thai’.

[5] John Plaice and Yannis Haralambous. The Ω system. `http://www.gutenberg.eu.org/omega`. Almost all modern TEX distributions contain support for Ω.

[6] Tomas Rokicki. The `afm2tfm` program. Part of the dvips package which is available from virtually all TEX distributions.

[7] Han The Thanh. pdfTEX. `ftp://ftp.cstug.cz/pub/tex/local/cstug/thanh/pdftex/latest`. pdfTEX is, similar to Ω, already part of most modern TEX distributions. The given URL specifies the primary address of pdfTEX since it still in development, sometimes with incompatible changes.

[8] The Unicode Standard. `http://www.unicode.org`.

◇ Werner Lemberg
Kl. Beurhausstr. 1
44137 Dortmund
`wl@gnu.org`