Low-level Devanāgarī Support for Omega — Adapting devnag

Yannis Haralambous
Département Informatique
École Nationale Supérieure des Télécommunications de Bretagne
BP. 832, 29285 Brest, France
yannis.haralambous@enst-bretagne.fr
URL: http://omega.enstb.org/yannis

John Plaice
School of Computer Science and Engineering
University of New South Wales
Sydney NSW 2052, Australia
plaice@cse.unsw.edu.au
URL: http://www.cse.unsw.edu.au/school/people/info/plaice.html

Abstract

This paper presents tools (OTPs and macros) for typesetting languages using the Devanāgarī script (Hindi, Sanskrit, Marathi). These tools are based on the Omega typesetting system and are using fonts from devnag, a package developed by Frans Velthuis in 1991. We are describing these new OTPs in detail, to provide the reader with insight into Omega techniques and allow him/her to further adapt these tools to his/her own environment (input method, font), and even to other Indic languages.

सारांश

यह लेख देवनागरी लिपि को प्रयोग करते हुए भाषाओं की टाईपसैटिंग या लेखायोजन के लिए प्रयोग किए जाने वाले टूल को प्रस्तुत करता है। ये टूल ओमेगा टाईपसैटिंग पर आधारित है और देवनाग के लेखारुपो को प्रयोग करता है जोकि फ्रांस वेलथुइस द्वारा 1991 मे बनाया गया पैकेज है। हम इन ओ टी पी को विस्तृत कर रहे हैं ताकि पड़ने वालो को ओमेगा तकनीक की जानकारी हो जाए और वो अपने आप को इस टूल से अपने वातावरण और दूसरी भारतीय भाषाओ के अनुरुप बदल सके या ढ़ाल सके।

Introduction

One of the first Indic language support packages for TEX was devnag, developed by Frans Velthuis in 1991. At that time it was necessary to use a preprocessor for converting Hindi or Sanskrit text written in a way legible to humans, into data legible by TEX. This preprocessor allowed the use of an ASCII transcription, and performed the contextual analysis inherent to Devanāgarī script, as well as pre-hyphenation (by explicitly inserting hyphenation points).

Using a preprocessor had many disadvantages, due mainly to the fact that it had to read not plain text, but rather LATEX code. Avoid treating commands and environment names as Devanāgarī text. So the preprocessor should be clever enough to distinguish text from commands, i.e. content from mark up.

It is well known that, in the case of TEX, this is practically impossible, unless the preprocessor is TEX itself (there is a notorious saying: "only TEX can read TEX").

So much for computing in the 20$^{\text{th}}$ century. Nowadays we have other means of processing information, and the concept of (external) preprocessor is obsolete. In fact, the same operations are done inside Omega, the successor of TEX. Processing text internally has the crucial advantage of allowing the processor to distinguish precisely what is content and what mark up (at least as precisely as TEX itself does it).

This makes it much easier to treat properties inherent to writing systems: one only needs to concentrate on the linguistic and typographical properties of the script, and one doesn't need to think of what to "do with LATEX commands" in the data stream.

Furthermore, there is an efficience issue: using Omega there is only one source file, namely the TeX file (and not a pre-TeX file and a TeX file); one doesn't need to care about preprocessor directives; the system will not fail because of a new LaTeX environment which is not known to `devnag`; mathematics and other similar constructions do not interfere with Devanāgarī preprocessing.

Contextual analysis of Devanāgarī script has, at last, become a fundamental property of the system, independent of macros and packages.

Unicode and Devanāgarī

The 20-bit information interchange encoding Unicode (`www.unicode.org`) has tables for all Indic writing systems, based on a common scheme (so that phonetically equivalent letters are placed on the same relative positions in each table). The first of these tables (positions `0900-097F`, see table 1) covers Devanāgarī.

For historical reasons (compatibility with legacy encodings) Unicode approach of Devanāgarī is quite akward: it is partly logical and partly graphical. For example, there are separate positions for independent and dependent versions of vowels: when encoding text one has to choose if a given vowel is dependent or independent, although this clearly derives from contextual analysis, as in Velthuis' transcription where both versions of vowels have excellent the same input transcription.

On the other hand, this method is not applied to consonant *ra*; indeed, placing a *ra* before a cluster of consonants is graphically represented by a mark on the last of the consonants (compare क्त *kta* and र्क्त *rkta*) — this mark is not provided in the Unicode table, and hence application of this feature is left to the rendering engine.

Nevertheless, despite its weaknesses, Unicode is very important because it ensures compatibility between devices all around the world: a text written in Devanāgarī and encoded in Unicode can be processed (read, printed, analyzed) on every machine or software that is Unicode compliant.

Omega fullfils Unicode compliance, and the system we are describing in this paper is designed in such a way that Unicode-encoded texts can be processed equally well as texts encoded in Velthuis' transcription.

Installation and Usage

The Omega low-level support[1] of Devanāgarī consists of eight OTPs (Omega Translation Processes) and a small file with macros:

```
velthuis2unicode.otp
hindi-uni2cuni.otp
hindi-uni2cuni2.otp
hindi-cuni2font.otp
hindi-cuni2font2.otp
hindi-cuni2font3.otp
sanskrit-uni2cuni.otp
sanskrit-cuni2font.otp
odev.sty
```

OTP files have to be converted in binary form (`*.ocp`) and placed in a directory where Omega expects to find them.

To typeset text in Devanāgarī, use the commands `\hindi` or `\sanskrit` (depending on the language of your choice) inside a group, and keyboard the text in Velthuis' transcription (see table 1, taken from Velthuis' `devnag` documentation[2]). For example, `{\hindi kulluu, acaanak, \sanskrit kulluu,`
`acaanak}` will produce कुल्लू, अचानक, कुल्लू, अचानक्.

Description of the OTPs

This description is a bit technical and demands both some knowledge of Omega and of Devanāgarī script. The reader can find more information on the former, on the Omega Web site[3] and on the latter in books about Devanāgarī script. In particular, there is a very nice introduction to the contextual features of the script in the Unicode book[4] (section 9.1).

velthuis2unicode.otp In this OTP we convert Velthuis' input transcription into Unicode. It is a quite short OTP (about 80 lines), with lines of the type

```
'z' => @"095B @"094D ;
'a''a' => @"0906 ;
```

[1] We call it "low-level," because there is no standard LaTeX3-compliant high-level language support interface yet. We don't know yet how languages and their properties will be managed in LaTeX3 and therefore do not attempt to introduce yet another syntax for switching to Hindi or Sanskrit or Marathi. Instead, we — temporarily — use a `devnag`-like syntax: simple commands `\hindi` and `\sanskrit` which have to be placed inside groups, as in the good old days of `plain` TeX...

[2] To be found on CTAN, `language/devanagari/distrib/manual.tex`.

[3] `http://omega.enstb.org`

[4] The Unicode Standard, Version 3.0, Addison Wesley, Reading Massachusetts, 2000.

On the second line, the pair of letters `aa` of Velthuis' transcription is sent to Unicode character `@"0906` (independent vowel "aa"). On the first line, letter `z` is sent to Unicode characters `@"095B` (letter "za") and `@"094D` (virama).

This may seem weird, but indeed the plan is to convert on a later step independent vowels into dependent ones, and to use the virama as a way to find out if a given consonant is part of a consonantic cluster or not. This will be done in the forthcoming OTPs.

(sanskrit|hindi)-uni2cuni.otp In this OTP we deal with virama and dependent vowels. First of all, in the case of Hindi, we remove the (eventual) final virama of the word:

```
{CONSONANT} {VIRAMA} end: => \1 ;
{CONSONANT} {VIRAMA} {NONHINDI} =>
   \1 <= \3 ;
```

In these two lines, we remove virama which is either at the end of the input buffer, or before a non-Hindi character. In the latter case, the non-Hindi character is put back in the stream. The code above is for Hindi. In the case of Sanskrit, we add a (fake) Unicode character which will represent internally the final virama:

```
{CONSONANT} {VIRAMA} end: => \1 @"097F ;
{CONSONANT} {VIRAMA} {NONHINDI} =>
   \1 @"097F <= \3 ;
```

Follow lines of the type:

```
{CONSONANT} {VIRAMA} {INITA} => \1 @"097D ;
{CONSONANT} {VIRAMA} {INITAA} =>
   \1 @"093E @"097D ;
```

Indeed, by placing a virama systematically after each consonant, we have also added viramas between consonants and vowels, which makes no sense. On the first line, the 'short a' vowel is removed together with the (spurious) virama. On the second line, the 'long a' vowel is replaced by Unicode character `"@"093E`, which is the dependent version of vowel 'long a', and the virama is removed. There are such lines for each vowel.

Notice the presence of "fake" Unicode character `@"097D`. This character will be replaced by a soft hyphen at the very last step of our OTP chain.

A special case is the vowel 'short i', where the glyph representing it has to be placed in front of the consonantic cluster. This is done by lines of the type:

```
{CONSONANT} {VIRAMA} {INITI} =>
   @"093F \1 \2 @"097D ;
```

where we have a consonant and virama followed by a 'short i' vowel. In this case we place Unicode character `@"093F` followed by the consonant. On similar lines, we have n-uplets ($n \leq 7$) of consonants and viramas followed by a 'short i' vowel; we replace them by `@"093F` followed by the group of consonants and viramas, except for the last virama.

hindi-uni2cuni2.otp One thing that has not been covered by the previous OTPs is the case of consonantic clusters starting with an 'r' consonant: in this case, a mark is placed on the last consonant ofthe cluster. This mark is not part of the Unicode encoding, and hence we have to use a fake Unicode character. This file contains lines of the type:

```
{RA} {VIRAMA} {CONSONANT} => \3 @"097E @"097D ;
```

On this line we replace a consonant preceded by a 'ra' and a virama, by the same consonant but followed by the fake Unicode character `@"097E` which will be replaced in the next OTP by the TEX code producing the mark we need.

(sanskrit|hindi)-cuni2font.otp In this OTP, which is quite long (328 lines), we start switching from Unicode to font encoding: this specific file — as well as files `cuni2font2.otp` and `cuni2font3.otp` — deals with `devnag` font encoding, but the user can write his/her own files for a different font encoding[5].

First of all we define aliases for all consonants, to make the writing of ligature expressions easier:

```
aliases:

BA = (@"092C) ;
BHA = (@"092D) ;
...
VIRAMA = (@"094D) ;
```

Then we write the ligature expressions, using expressions like the following:

```
{SSA} {VIRAMA} {TTA} {VIRAMA} {YA} => @"00F7 ;
{SSA} {VIRAMA} {TTA} {VIRAMA} {VA} => @"00AB ;
{SSA} {VIRAMA} {TTA} {VIRAMA} {RA}
   {VIRAMA} {YA} => @"00AA ;
{SSA} {VIRAMA} {TTA} {VIRAMA} {RA} => @"0104 ;
```

As the reader can see, the virama is used to insure that these consonants are indeed part of the same consonantic cluster. As in OTP files order of precedence is based on order of expressions in the expression list, the fact that line 3 is before line 4 insures that the consonantic cluster `.s.try` is indeed detected instead of `.s.tr`, which will be matched only if the last letter is not a `y`.

[5] For example for the prestigious *Monotype Devanagari* (`http://www.agfamonotype.com`) which is, IOHO, one of the most beautiful existing fonts, and has even pre-designed glyphs for consonants with dependent short and long 'u' vowels.

Notice that our right-hand expressions are already in the devnag font positions, except for the one of the last line (@"0104), which is a 'fake' glyph position — like we had previously 'fake' Unicode characters — i.e. a byte which will be detected by a forthcoming OTP and converted into something that makes sense.

The devnag system allows preprocessor directives activating and de-activating individual ligatures; we do not have an equivalent feature in our system because we do not consider it to be crucial. Instead, the user has the possibility to add or remove lines as the ones above in the OTP file, save the OTP under a different name and use it as a replacement of the standard one, or in a new OCP list. In the latter case, one can switch on-the-fly from one ligature setup to another.

After the ligature expressions, follow the consonants followed by virama:

```
{BA} {VIRAMA} => @"004E ;
{BHA} {VIRAMA} => @"003C ;
{CA} {VIRAMA} => @"0051 ;
{CHA} {VIRAMA} => "\qq{" @"0043 "}" ;
{DA} {VIRAMA} => "\qq{" @"0064 "}" ;
...
```

If one of these patterns is matched, this means that (a) we are inside a consonantic cluster and (b) all ligatures have been matched. Two options remain: either there is a special half-form of the glyph of the consonant, or an explicit virama is placed under the normal version of the consonant glyph.

Using "halfed" glyphs is, in a sense, intermediate between predefined ligature glyphs and the placement of individual "independent" glyphs next to each other. It is a method to construct arbitrary ligatures using the basic letter part: compare, for example, for the same consonantic cluster "vva," व्व (Sanskrit ligature), व्व (Hindi ligature, where the first व is half-form) and व्‌व (two individual consonants, the first having a virama).

In the code above, macro \qq inserts the virama. In version 2 of Omega this macro will be made obsolete, since placement of diacritics will be handled by μ-engines; until then, we use macros, like \qq, taken from devnag package.

Follow two special lines:

```
{RA} {DEPU} => @"007A ;
{RA} {DEPUU} => @"0021 ;
```

where DEPU and DEPUU stand for "dependent short u" and "dependent long u." These cover the special glyphs for consonant 'ra' with these vowels: र + उ → रु, र + ऊ → रू.

Finally, follow lines of the type:

```
{BA} => @"0062 ;
{BHA} => @"0042 ;
{CA} => @"0063 ;
...
```

which simply match consonants (with inherent "short a" vowel) and glyph positions, as well as lines like

```
{CANDRABINDU} =>
   "\llap{{\clearocplists\char32}}" ;
{ANUSVARA} =>
   "\llap{{\clearocplists\char92}}" ;
{DEPAI} =>
   "\llap{{\clearocplists\char123}}" ;
...
```

which map characters candrabindu, anusvāra, dependent "ai" vowel and similar signs with the necessary TEX code to obtain their glyphs. Once again this code will be obsolete in Omega v. 2.

hindi-cuni2font2.otp This OTP, as well as the next one, are provided to deal with cases which could not be handled simultaneously with the previous one. The present file deals with dependent vowels "short u," "long u," "short r," "short l," "English o," which have the common property of being centered under the letter. Until Omega 2 arrives, we need to use macros to place them, and these macros have to be placed *before* the consonant which carries the vowel, so that this consonant can be their argument:

```
(@"0000-@"00FF) {DEPU} => "\qqqa{" \1 "}" ;
(@"0000-@"00FF) {DEPUU} => "\qqqb{" \1 "}" ;
(@"0000-@"00FF) {DEPR} => "\qqqc{" \1 "}" ;
(@"0000-@"00FF) {DEPRR} => "\qx{" \1 "}" ;
(@"0000-@"00FF) {DEPL} => "\qy{" \1 "}" ;
(@"0000-@"00FF) {DEPLL} => "\qz{" \1 "}" ;
(@"0000-@"00FF) {DEPOO} => "\qzz{" \1 "}" ;
```

We could not obtain them in the previous OTP, since that file matched the Unicode characters of consonants and replaced them with font positions. Of course we could include in that file *combinations* of consonants and vowels, but this would make the file unnecessary long: it is easier to match first the consonants and, at a second stage, the vowels.

Since we are now matching font positions, the left-hand expressions use (@"0000-@"00FF). This works only because dependent vowels always follow consonants. Nevertheless it is not very elegant, and this code will return to non-existence as soon as μ-engines are available.

hindi-cuni2font3.otp This very short file (25 lines), deals with the final virama and with some special cases of clusters: combinations of various consonants and consonant "ra." The final virama is used in Sanskrit only (according to Velthuis' convention) and has to be dealt with separately because in our OTP system we have used the (regular) virama as a marker of consonants *inside* consonantic clusters. If we had used a "regular" virama also at the end of the word, then we would obtain consonantic clusters with only half-forms of consonants and no full-form at the end. Instead, we have used a fake Unicode character (`@"097F`) for the final virama and are replacing it by its TEX code only at the very last step, as follows:

```
(@"0000-@"00FF) @"097F => "\qq{" \1 "}";
```

It can happen that the last consonant, although it has no vowel, carries a special sign because the consonantic cluster starts with consonant "ra;" to handle that case we have two extra lines:

```
(@"0000-@"00FF) @"097E @"097F => "\qq{"
   \1 "}\llap{\clearocplists\char13}" ;
(@"0000-@"00FF) @"097F @"097E => "\qq{"
   \1 "}\llap{\clearocplists\char13}" ;
```

which send the two combinations of fake Unicode characters to the same TEX code producing both the "ra" mark and the virama.

Finally there is a line replacing the fake Unicode character `@"097D` used to temporarily stand for the soft hyphen, with the adequate `\discretionary`:

```
@"097D => "\discretionary{\hyph}{}{}" ;
```

If the user does not wish hyphenation, he/she can replace this line by a simpler one, which will "absorb" all `@"097D` characters:

```
@"097D => "" ;
```

Conclusion

The purpose of the previous sections was to illustrate the processing of a given script (Devanāgarī) by Omega Translation Processes. Omega 2 will make these even more efficient since code used to center diacritics under consonants will be replaced by μ-engines. We believe that these methods can be applied to other Indic scripts. Furthermore the fact of having three kinds of files:

1. OTPs for converting input transcription into Unicode;
2. OTPs for handling contextual analysis of Indic scripts;
3. OTPs for converting real or fake Unicode characters into glyphs for a given font,

make this system easily adaptable to any combination of input method and font.

We invite Omega users around the world to write the necessary code and make it available to other through CTAN servers. We hope that these tools will make processing of Indic languages easier and more efficient and will allow production of high quality documents.

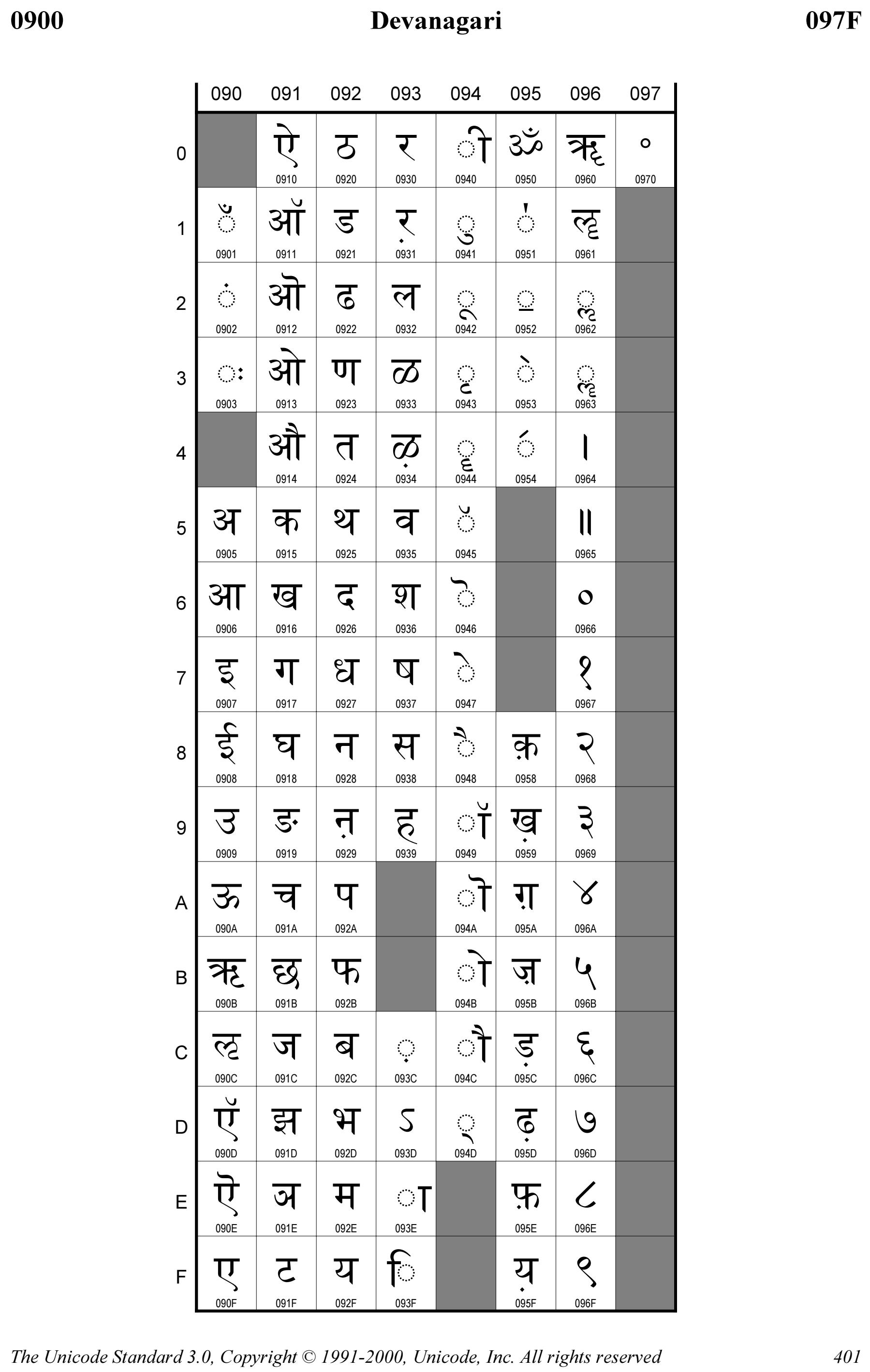
Availability and Thanks

All resources described in this document are free software and are available on CTAN. The OTP and macro files described in this paper can be found on CTAN in

`language/devanagari/omega`

We invite users writing software for Omega typesetting of Indic languages into similar `omega` directories inside the corresponding Indic language directories.

The authors would like to thank Anish Mehta and Gagan Sharma, *stagiaires* at ENST de Bretagne at the time this paper was written, for their valuable help.



0900 **Devanagari** **097F**

	090	091	092	093	094	095	096	097
0		ऐ 0910	ठ 0920	र 0930	◌ी 0940	ॐ 0950	ॠ 0960	॰ 0970
1	◌ँ 0901	ऑ 0911	ड 0921	ऱ 0931	◌ु 0941	◌॑ 0951	ॡ 0961	
2	◌ं 0902	ऒ 0912	ढ 0922	ल 0932	◌ू 0942	◌॒ 0952	◌ॢ 0962	
3	◌ः 0903	ओ 0913	ण 0923	ळ 0933	◌ृ 0943	◌॓ 0953	◌ॣ 0963	
4		औ 0914	त 0924	ऴ 0934	◌ॄ 0944	◌॔ 0954	। 0964	
5	अ 0905	क 0915	थ 0925	व 0935	◌ॅ 0945		॥ 0965	
6	आ 0906	ख 0916	द 0926	श 0936	◌ॆ 0946		० 0966	
7	इ 0907	ग 0917	ध 0927	ष 0937	◌े 0947		१ 0967	
8	ई 0908	घ 0918	न 0928	स 0938	◌ै 0948	क़ 0958	२ 0968	
9	उ 0909	ङ 0919	ऩ 0929	ह 0939	◌ॉ 0949	ख़ 0959	३ 0969	
A	ऊ 090A	च 091A	प 092A		◌ॊ 094A	ग़ 095A	४ 096A	
B	ऋ 090B	छ 091B	फ 092B		◌ो 094B	ज़ 095B	५ 096B	
C	ऌ 090C	ज 091C	ब 092C	◌़ 093C	◌ौ 094C	ड़ 095C	६ 096C	
D	ऍ 090D	झ 091D	भ 092D	ऽ 093D	◌् 094D	ढ़ 095D	७ 096D	
E	ऎ 090E	ञ 091E	म 092E	◌ा 093E		फ़ 095E	८ 096E	
F	ए 090F	ट 091F	य 092F	◌ि 093F		य़ 095F	९ 096F	

Figure 1: Unicode Table for Devanāgarī script.

Vowels											
a	अ	—	aa, A	आ	ा	i	इ	ि	ii, I	ई	ी
u	उ	ु	uu, U	ऊ	ू	.r	ऋ	ृ	.R	ॠ	ॄ
.l	ऌ	ॢ	.L	ॡ	ॣ	e	ए	े	ai	ऐ	ै
o	ओ	ो	au	औ	ौ	aM	अं	ं	aH	अः	ः

Consonants									
ka	क	kha	ख	ga	ग	gha	घ	"na	ङ
ca	च	cha	छ	ja	ज	jha	झ	~na	ञ
.ta	ट	.tha	ठ	.da	ड	.dha	ढ	.na	ण
ta	त	tha	थ	da	द	dha	ध	na	न
pa	प	pha	फ	ba	ब	bha	भ	ma	म

Semi-Vowels							
ya	य	ra	र	la	ल	va	व

Sibilants					
"sa	श	.sa	ष	sa	स

Aspirate	
ha	ह

Supplementary Consonants															
qa	क़	.kh, .K	ख़	.ga	ग़	za	ज़	Ra	ड़	Rha	ढ़	fa	फ़	La	ळ

Numerals									
0	०	1	१	2	२	3	३	4	४
5	५	6	६	7	७	8	८	9	९

Special Characters											
.o	ॐ	*AUM*	.m, M	ं	*anusvāra*	/	ँ	*candrabindu*	.h, H	ः	*visarga*
.a	ऽ	*avagraha*	@	॰	continuation	*	·	elliptical dot	~r	ऱ	Marathi *ra*
~a	ॅ	English a	~o	ऑ	English o	|	।	*daṇḍā*	..	.	period

Table 1: The Velthuis Transliteration Scheme