UTC Document Register **L2/14-126R**

Improvements requested for Unicode Indic properties

Roozbeh Pournader, Google Inc.
May 8, 2014

Background

This document draws on changes originally suggested in Pournader and Esfahbod's "A bag of suggested improvements to Unicode's provisional Indic properties" (L2/14-065). That document was thoroughly discussed at an informal meeting of people interested in improvements to the OpenType-related font technologies, which took place April 21–25, 2014 in Seattle, Washington. Attendees included Peter Constable, Behdad Esfahbod, Andrew Glass, Greg Hitchcock, Ned Holbrook, John Hudson, Nikhil Kumar, Eric Mader, Sergey Malkin, Eric Muller, Anshuman Pandey, Roozbeh Pournader, and Miguel Sousa. (Not all attendees were present during all discussions.)

The attendees agreed that Unicode's Indic-related properties are extremely useful for technologies used to render present and future Unicode text, especially in detecting the structure of syllables and deciding when to reorder glyphs in displaying Indic texts and where in the text stream to reorder them to.

HarfBuzz, the open source text rendering engine, has already been using a modified version of the properties, and developers working on other existing rendering engines expressed a lot of interest in using the data, assuming the changes specified below would be accepted by the UTC. The attendees reviewed the suggestions in L2/14-065 and arrived at a consensus in the general aspects of the following requests (the specifics are from the author). In order to reduce the potential instability risks, the architectural changes requested have been kept to a minimum.

The author urges the members of the Unicode Technical committee to consider these for Unicode 7.0. These would help in the further development of Unicode-conformant rendering engines, fonts, OCR software, and other similar applications.

The immediate acceptance of these changes for Unicode 7.0 would pave the way for a much faster convergence of text rendering technologies, resulting in improved and interoperable support for existing Indic scripts and those encoded in future, especially including those used for minority and historical languages.

Updated data files with the changes applied are provided as an attachment.

Change requests regarding the Indic Syllable and Matra categories

1. Similar to bidi and Arabic joining properties, where special formatting characters have their own property values, add special property values for ZWJ, ZWNJ, and BNJ in Indic_Syllabic_Category. Such property values would make it possible for algorithms using the property to not look at codepoints directly, and look at character properties instead.

2. Add characters that otherwise participate in Indic syllables to the existing set defined to have the property value Consonant_Placeholder: the Myanmar symbol for “aforementioned” at U+104E is missing from the class, and so do various dashes and the multiplication sign, which are used in Indic scripts as consonant placeholders. For example, according to TUS Core Specification 6.2, section 9.9, page 319:

   > “More generally, rendering engines should be prepared to handle Malayalam letters (including vowel letters), digits (both European and Malayalam), dashes, U+00A0 no-break space and U+25CC dotted circle as base characters for the Malayalam vowel signs, U+0D4D malayalam sign virama, U+0D02 malayalam sign anusvara, and U+0D03 malayalam sign visarga.”

3. Add a new class Number, since they also act as consonant placeholders, but their semantics is different from a typical Consonant_Placeholder (they participate as a group, in case of a two-digit number followed by a vowel).

4. For Brahmi numbers, which can be joined by the Brahmi Number Joiner, add a class of Brahmi_Joining_Number, split from Number.

5. Divide the syllabic category of viramas into three classes: those that are linguistically killers but don’t form visual conjuncts (Pure_Killer), those that have no visual representation and work as control characters joining the next consonant (Invisible_Stacker), and those that could be either (Virama).

6. Divide the Consonant_Repha category into two classes: Consonant_Preceding_Repha, where the Repha is used in logical order (such as Malayalam), as opposed to Consonant_Succeeding_Repha, which is in visual order (Khmer, Javanese, etc). Currently, the general category of the character may be used as a hint for making a distinction, but we believe the distinction is large enough and the hint cannot be assumed to work for future characters.

7. Assign matra category of Top to all Khmer characters at U+17C9 .. U+17D0, and U+17D3 that visually act like top matras.

8. Add Gurmukhi Addak at U+0A71 to matra category of Top.

9. Add matra categories for the reordrant non-matra characters: from Table 4-4 of the Core Specification, these are the Tai Tham U+1A55, the Lepcha U+1C34 and U+1C35, and the Cham U+AA34.

Open Issues

10. As suggested by Ken Whistler, the matra category may better be renamed positional category to better reflect the semantics.

11. As suggested by Andrew Glass, the syllabic property value of Consonant_Placeholder may better be split to normal linguistic placeholders and Other_Carrier or Other_Base, since some of these are really artificial/visual consonant placeholders different from linguistic consonant placeholders.

12. The status of Jihvamuliya and Upadhmaniya characters in Kannada, Vedic, Brahmi, and Sharada is unclear. Which of them need viramas to form conjuncts and which of them don't?

13. Is U+17CB KHMER SIGN BANTOC really a register shifter?

14. What are the best syllabic categories for U+17CD..17D0 and U+17D3?

15. The characters in the Vedic Extensions block are missing Indic properties. There may be more similar characters, scattered through the various blocks.

16. The Lepcha U+1C29 is listed as Top_And_Left in Indic Matra Category, but is listed together with Left matras in Table 4-4 of the Core Specification. This may actually be a left matra, similar to the Devanagari vowel I that extends to the top of the consonant. There are other similar characters potentially misclassified, which can be found below:

    U+0B57: `Top_And_Right`, should be `Right` (Oriya AU Length Mark)
    U+1C29: `Top_And_Left`, should be `Left` (Lepcha OO)
    U+A9C0: `Bottom_And_Right`, should be `Right` (Javanese killer)
    U+111BF: `Top_And_Right`, should be `Right` (Sharada AU)

17. At least seven characters are encoded in Unicode with left and right pieces separately encoded but with no canonical decompositions to the pieces. We should check that there is text in the Core Specification and the NamesList that mention the preferred encoding for each of the cases, as they are ambiguous. We also need to make sure they are added to the list of confusables.

0AC9 => 0AC5 0ABE (Gujarati Candra O) [Confirmed fixed by Eric Muller]
0F77 => 0FB2 0F81 (Tibetan Vocalic RR)
0F79 => 0FB3 0F81 (Tibetan Vocalic LL)
17BE => 17C1 17B8 (Khmer OE)
17C4 => 17C1 17B6 (Khmer OO)
1925 => 1920 1923 (Limbu OO)
1926 => 1920 1924 (Limbu AU)

Bibliography

1. Behdad Esfahbod et al. 2014. *HarfBuzz, an OpenType text shaping engine*. http://www.harfbuzz.org/.
2. Roozbeh Pournader and Behdad Esfahbod. 2014. “A bag of suggested improvements to Unicode’s provisional Indic properties”. UTC Document Register L2/14-065, The Unicode Consortium. http://www.unicode.org/L2/L2014/14065-indic-properties.pdf
3. The Unicode Consortium. 2013. *The Unicode Standard Version 6.2 – Core Specification.*