IP Feedback on Devanagari LGR Proposal Variants 2019-03-06

DATE: 2019-04-05

# Introduction

The IP has been reviewing all post-public comment NeoB GP proposals plus Sinhala one final time with special attention to variants. This document reviews the Devanagari LGR Proposal dated 2019-03-06.

All NeoB scripts (plus Sinhala) have code points that have code point contexts. Where variants are defined for them, it may be necessary or advisable to define variant context rules as well. The details depend on the precise circumstances.

Some LGRs have variants defined for sequences and in some cases, these sequences exhibit “overlap”: part of a sequence is also a variant, whether in the same or different variant set (general case: AB <--> C and A <--> F). Such overlapping sequences require special attention to ensure the LGR as a whole is well behaved, even if every variant set is symmetric and transitive.

The scripts affected by these include Devanagari, Gurmukhi, Malayalam, Sinhala and Tamil.

The following are comments for the Devanagari Proposal.

# Aligning Code Point and Variant Contexts

Generally, the IP has been following the principle that if the effective contexts for source and target of a variant mapping cannot be made equal, then a variant context should be defined that covers the intersection between these contexts. Making these adjustments “isolates” a variant from surrounding context, ensuring that the set of variant labels remains well behaved: no matter the surrounding context in any actual label (including its variants), the variant mapping always leads to a valid label.

Absent this guarantee, a weaker measure of well-behaved-ness may be used, namely whether all variant labels from the same set are guaranteed to compute the same index value. That this weaker measure is achieved cannot be demonstrated as easily.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Source | Glyph | Target | Glyph |   | Type | Categories | Variant Context |
| 0902 | ं | 093A | ऺ | ↔ | blocked | D ↔ M | None: set contains cross-script |
| 0902 | ं | 0A02 | ਂ | ↔ | blocked | N/A (cross-script) | None: set contains cross-script |
| 093A | ऺ | 0A02 | ਂ | ↔ | blocked | N/A (cross-script) | None: set contains cross-script |

For this variant set, the explicit code point contexts on the first mapping are not equal

0902: follows-V-or-C-or-N-or-M

093A: follows-C-or-CN

The *implicit code point contexts* are also not equal:  0902 can be followed by Vowels and Consonants only, but 093A can also be followed by 0901, 0902, 0903. Normally, the variant mapping should receive a context rule: when(follows-C-or-CN-and-followed-by-V-or-C). This would match the intersection between these contexts.

However, the code points 0902 and 093A also have a **cross-script variant** each. This variant does not have a context rule (because a context rule for one script could not be satisfied by the symmetric mapping from the other script). Therefore, the variant set cannot be made transitive if a variant context is added, as that would require all contexts to be the same.

Creating a hypothetical label, choosing the contexts so that either 0902 or 093A are valid in the given slot, but not both:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| # | Label |  | Variant + Original | Status | Comment |
| 1 | (V-or-M) **0902** (C) (valid) | 🡪  | (V-or-M) **093A** (C)(V-or-M) **0902** (C) | (invalid)(valid) | index would be original labeloriginal label, and index |
| 2 | (V-or-M) **093A** (C) (invalid)  | 🡪  | N/A | (invalid) | N/A  |
| 3 | (C) **093A** (B-D-X) (valid)  | 🡪 | (C) **0902** (B-D-X)(C) **093A** (B-D-X) | (invalid)(valid) | indexoriginal |
| 4 | (C) **0902** (B-D-X) (valid) | 🡪  | N/A | (invalid) | N/A  |

Here, (V-or-M) indicates a leading context that is either a vowel or a matra, and so on.

This proves, that is possible to create a valid label (#3) that produces an *invalid* index label (the variant label containing 0902 would be the index, because 0902 is the smaller than 093A). Therefore, the variant set is only well-behaved if we allow the index variant to be invalid. We already need to allow the index variant to be invalid because in the general case an index variant could be a mixed script label that is always invalid, so this does present a new problem. An index variant label does not have to be an actual label, as long as all members of a variant label set porduce the same index label.

The variant set discussed here overlaps the following Variant Set A:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Set | Source | Glyph | Target | Glyph |  | Type | Variant Context |
| Variant set A | 0905 0902 | अं | 0973 | ॳ | ↔ | blocked |  |

This particular variant set overlap was already analyzed in an earlier IP report on Devanagari variants as follows:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Initial Label | Partitions | Disp | Variants | Comments |
| 0905 0902  | {0905}{0902}{0905 0902} | validvalid | 0905 09020905 093A0905 0A020973 | valid /original / indexinvalidinvalidvalid |
| \*0905 093A  | {0905}{093A} | invalid | N/A | ignored because source invalid |
| \*0905 0A02  | {0905}{0A02} | invalid | N/A | ignored because source invalid |
| 0973 | {0973} | valid | 0905 09020973 | valid / indexvalid / original |

Invalid source sequences have been marked with (\*).

All valid labels in this set produce the same index variant (0905 0902), but only because label 0905 093A is not the index variant by virtue of being larger compared to 0905 0902. (The overlap with sequence 0905 0902 gives rise to multiple partitions for some labels, but does not change the result.)

This variant definition is therefore ***fragile:*** it depends on the precise details of the index variant calculation chosen. However, as long as index variant calculation proceeds as documented for the Root Zone LGR, it will produce the expected results.

Recommendation:

(1) No change

(2) Alternatively, consider withdrawing the relation between 0902 and 093A

# Overlapped Variants Involving Nukta

|  |  |  |  |
| --- | --- | --- | --- |
| Set | Mapping | Categories | Variant Context  |
| Variant Set B | 093E 0901 <--> 0949 0902 | M1 D 🡨🡪 M B |  |
| Variant Set C | 0906 0901 <--> 0911 0902  | V1 D 🡨🡪 V B |  |
| Variant Set D | 0906 0902 <--> 0974 | V1 B 🡨🡪 V |  |
| Variant Set E | 093B <--> 093E 0902  | M 🡨 🡪 M1 B |  |

Overlapping variant sets involving 0906 and 093E plus Nukta

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source | Glyph | Target | Glyph |   | Type | Variant Context |
| 0906 | आ | 0906 093C | आ़ | ↔ | blocked | not: followed-by-N |
| 093E | ा | 093E 093C | ा़ | ↔ | blocked | not: followed-by-N |

When substituting the variant from the overlapping sets for 0906 or 093E respectively, into the left hand side sequences of Variant Sets B, C, D and E, the result is a valid sequence that displays with a Nukta (for a typical example see “Notes on Variant Set E”). As the presence/absence of Nukta is the basis for several variant sets, the IP feels that these substituted sequences should be part of variant sets B, C, D or E as appropriate. This formally extends their transitivity to account for the effect of overlapping.

Variant set D and E also need a variant context (see the notes).

Recommendations:

(1) Add 093E 093C 0901 as member of variant set B, and add the sequence to the repertoire, with code point context: not-when (preceded-by-H).

 (2) Add 0906 093C 0901 as member of variant set D, and add the sequence to the repertoire, with code point context: not-when(preceded-by-H).

(3) Add 0906 093C 0902 as member of variant set D, and add the sequence to the repertoire, with code point context: not-when(preceded-by-H).

(5) Add a variant context to all mappings of variant set D: when(follows-followed-by-V-or-C)

 (4) Add the sequence 093E 093C 0902 to the variant set E, and add the sequence to the repertoire with context: not-when(preceded-by-H).

(5) Add a variant context to all mappings of variant set E: when(follows-C-or-CN-and-followed-by-V-or-C)

## Variant set B – extended to cover sequence 0906 093C 0901 (all are type “blocked”)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Source | Glyph | Target | Glyph |   | Variant Context |
| 093E 0901 | ाँ | 093E 093C 0901 | ा़ँ | ↔ |  |
| 093E 0901 | ाँ | 0949 0902 | ॉं | ↔ |  |
| 093E 093C 0901 | ा़ँ | 0949 0902 | ॉं | ↔ |  |

## Variant set C – extended to cover sequence 0906 093C 0901 (all are type “blocked”)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Source | Glyph | Target | Glyph |   | Variant Context |
| 0906 0901 | आँ | 0906 093C 0901 | आ़ँ  | ↔ |  |
| 0906 0901 | आँ | 0911 0902 | ऑं | ↔ |  |
| 0911 0902 | ऑं | 0906 093C 0901 | आ़ँ | ↔ |  |

## Variant set D – extended to cover sequence 0906 093C 0901 (all are type “blocked”)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Source | Glyph | Target | Glyph |   | Variant Context |
| 0906 0902 | आं | 0906 093C 0902 | आ़ं | ↔ | when(followed-by-V-or-C) |
| 0906 0902 | आं | 0974 |  | ↔ | when(followed-by-V-or-C) |
| 0974 |  | 0906 093C 0902 | आ़ं | ↔ | when(followed-by-V-or-C) |

### Notes on Variant Set D

For this variant set, the explicit contexts for both 0906 and 0974 are identical: not-when(preceded-by-H). However, the *implicit code point contexts* on the second and third mappings are not equal:  0902 can be followed by Vowels and Consonants only, but 0974 can also be followed by 0901, 0902, 0903. Therefore, the variant mapping should receive a context rule: when(followed-by-V-or-C). This matches the intersection between these contexts. (Because the explicit contexts match, there is no need to repeat them in the variant context).

## Variant set E – extended to cover sequence 093E 093C 0902 (all type “blocked”)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Source** | **Glyph** | **Target** | **Glyph** |  | **Variant Context** |
| 093B |  | 093E 0902 | ां | ↔ | when(follows-C-or-CN-and-followed-by-V-or-C) |
| 093B |  | 093E 093C 0902 | ा़ं | ↔ | when(follows-C-or-CN-and-followed-by-V-or-C) |
| 093E 0902 | ां | 093E 093C 0902 | ा़ं | ↔ | when(follows-C-or-CN-and-followed-by-V-or-C) |

### Notes on Variant Set E

For this variant set, the explicit code point contexts on the first mapping are not equal

0902: follows-V-or-C-or-N-or-M

093B: follows-C-or-CN

The *implicit code point contexts* are also not equal:  0902 can be followed by Vowels and Consonants only, but 093B can also be followed by 0901, 0902, 0903. Therefore, the variant mapping should receive a context rule: when(follows-C-or-CN-and-followed-by-V-or-C). This matches the intersection between these contexts.

The variant with Nukta (093E 093C) may be substituted for 093E in the sequence 093E 0902 because 0902 may follow a Nukta. Prefixing the sequences by a consonant (0937) to make everything part of a valid label:
0937 093E 0902 (षां) 🡨 🡪 0937 093E 093C 0902 (षा़ं).
These appear to be ordinary variant sequences of each other that follow the principle that presence/absence of a Nukta is reason for a variant set: for that reason the sequence 093E 093C 0902 should be added to the variant set.

## Overlapped Candrabindu Variants

|  |  |  |  |
| --- | --- | --- | --- |
| Set | Mapping | Categories | Variant Context (as recommended) |
| Variant Set F | 0901 <--> 0945 0902 | D 🡨🡪 M B | when(follows-only- C-or-N) |

Technically, variant set F is ***overlapped*** with all the other variant sets, because 0901 is a subsequence of either source or target in all the other four sets, where it occurs following a vowel or matra.

However, because of the variant context rule “when(follows-only-C-or-N)”, none of the sequences (even if partitioned as singletons, e.g. {0906}{0901} can lead to a variant where 0901 is expanded to 0945 0902.

## Other Variants Overlapped With These Sets

Another overlapped variant with these four sets is U+0902 (B, Anusvara) because it is a variant of U+093A (M, Matra). Because U+0902 is a member of several sequences in the variant sets discussed here, there is potential for variant sequences substituting 0902 by 093A. However, they are all invalid as a Matra can only follow C or N, while all sequences including 0902 as second element have V or M as first element. Likewise, the substitution with 093A would also be invalid, so that, for example 0911 093A is invalid and therefore cannot give rise to a variant 0911 0902 and vice versa.

This is fortunate, because U+0902 also has a cross-script variant in the Gurmukhi code point U+0A02, and any context rule defined for 0902🡨🡪 093A would by transitivity apply to that cross-script mapping as well. However, it would not have been possible to set a variant context on the mapping 0902 🡨🡪0A02 (the context rule would have to apply symmetrically to U+0A02 and such variant would always be undefined for any whole-label Gurmukhi variant).

Recommendation: overlap prevented by context rules: no change to sets B, C or D.

# Overlapped Sets Not Implicated with Nukta or Candra Variants

|  |  |  |  |
| --- | --- | --- | --- |
| Set | Mapping | Categories | Variant Context  |
| Variant Set G | 092A 094D 091F 0947 <--> 0A0F | N/A (out-of-script) | N/A (out-of-script) |

Overlapped variant set involving 0947

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source  | Glyph  | Target  | Glyph  |   | Type  | Variant Context |
| 0946  | ॆ | 0947  | े | ↔  | blocked  | (none) |

(This set also contains out-of-script members that do not affect further analysis)

The trailing 0947 can be substituted and gives rise to a variant 092A 094D 091F 0946 (प्टॆ). The appearance between the sequences ending in 0946/0947 varies on the same principle as between 0946 and 0947. This is incidentally the same principle that leads to an unrelated variant set between 090E and 0910. Therefore, there is no reason to restrict this particular variant substitution; instead the variant set should be made transitive under overlap by including 092A 094D 091F 0946.

Recommendations:

Based on 0946 🡨🡪 0947 the variant set F: 092A 094D 091F 0947 (प्टे) 🡨🡪 0A0F (ਏ) should be extended by:

(1) Defining sequence 092A 094D 091F 0946 (प्टॆ) with no code point context

(2) Adding it to Variant set G with no variant context required.

Variant F set as extended:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Source  | Glyph  | Target  | Glyph  |   | Type  | Variant Context |
| 092A 094D 091F 0946 | प्टॆ | 092A 094D 091F 0947  | प्टे | ↔ | blocked  | (none) |
| 092A 094D 091F 0947  | प्टे | 0A0F  | ਏ | ↔ | blocked  | (none) |
| 0A0F | ਏ | 092A 094D 091F 0946 | प्टे | ↔ | blocked  | (none) |

## Other sequences starting with 092A

Note that the other two sequences starting with 092A are not affected, because there are no in-script variants that overlap. For example, for the sequence 092A 094D 091F 093F the trailing 093F could be substituted by 09BF, which is Bengali, leading to a mixed script label that can never be valid and thus can be ignored in this analysis.

Variant sequence (092A 094D 091F 093F) overlaps with (091F), (092A) and (093F) from different variant sets. The variants for 091F, 092A and 093F are out-of-script, substituting any, or all would lead to a mixed script label (because 094D would remain unchanged).

Variant sequence (092A 094D 091F 0940) overlaps with (091F), (092A) and (0940) from different variant sets. The variants for 091F, 092A and 0940 are out-of-script, substituting any, or all would lead to a mixed script label (because 094D would remain unchanged).

Recommendation: no overlap with in-script variant sets: no change needed.

# Cross-script variants

For variant sets with cross script variants, there can be no *variant* contexts, because context definitions must apply equally to symmetric mappings. Any context definition that depends on the nature of one of the scripts would fail for the reverse mapping.

For in-script variants, we like to see the set of variant *labels* fully transitive. This is never possible for cross-script variants as many variant labels would be mixed-script and therefore invalid. Ultimately, we only care about whole-script cross-script variant labels.

By ignoring the potential variants created for any invalid label and by allowing the index variant itself to be an invalid label we believe we can guarantee that index label computation is unambiguous in the cross-script case, whether or not the code point or sequence in the source script has a *code point* context or not.

Effectively this holds if all valid variant labels generate a subset of the full variant set valid or not, and that subset always includes the index variant).

The one remaining issue is overlapped variants, because they offer a potential way for variant label sets to overlap. Therefore, this analysis includes all variant sets with cross-script variants that overlap or are overlapped by another set. Sometimes, this overlap produces sets that are fragile. (See Section 2 above).