IP Response to Malayalam LGR Update

DATE: 2019-12-16

# Overview

The IP has reviewed the updated Malayalam LGR proposal dated 2019-10-26.

The IP tried to answer the following questions:

1. Does the proposed update remove the inconsistencies noted in the problem report?
2. Were the inconsistencies resolved in a manner that comports with the Principles?
3. Do the proposed fixes introduce other problems?
4. Are there any larger issues that need to be considered?

The following recommendations address issues found. The IP is looking forward to a revised update reflecting these recommendations.

# Recommendations

The following items list the Integration Panel recommendations, with additional background information provided in the last section:

A1: The IP reviewed the text of the revised LGR and found no changes unrelated to the issue being updated. The text of the revised LGR needs some fixes (in some cases, the final text would depend on other recommendations made here, for example item A.2 and A.3). An annotated copy of the LGR document with notes and proposed language in redline is attached.

*The IP requests that the GP review the notes and suggestions in the attached document and make any changes appropriate (and, once resolved, delete any notes from the text).*

*Response: Noted.*

A2: Conflicts due to overlapped variants. The IP notes that the new sequence introduces an effective Null-variant (see section 6.6. in LGR-3 for a description of that). This does not seem to result in any issues. However, all the variant sequences in that set also end in 0D31 റ which introduces an **overlapped variant** with the existing variant sequences starting with 0D31 റ. *The IP has investigated these and determined that additional constraints are needed to ensure that the corresponding variant label sets are well-behaved.* (For details see Section 3 “Variant Issues” below).

***IP Recommendation:***

*(1) remove trailing 0D31 from the sequences defined for variant mappings for “nta”*

*(2) add when(followed-by-0D31) as a context rule on the variants*

*(3) add when(followed-by-0D31) as a context rule on the sequence 0D7B 0D4D (see also A 2 above)*

*Response: Done*

1. *Edits in the document:*
	1. *Used “(0D31)” in Table 7a.*
	2. *Accepted the suggested paragraphs under Table 9.*
	3. *Inserted V1 contextual rule for variant, the former V1, V2 become V2, V3.*
2. *Edits in the XML*
	1. *Removed trailing 0D31 from three sequences*
	2. *Add rule “followed-by-0D31” and updated relevant descriptions*
	3. *Add when(followed-by-0D31) as a context rule on the* ***variants***

*\*Did not apply recommendation (3). The context rule on* ***sequence*** *of 0D7B 0D4D was edited as per recommended in A4.*

A3: New cross-script variants in LGR-4. Once integrated with the other scripts slated for LGR-4, the Malayalam LGR will also have an overlapped cross-script variant issue with Myanmar and Georgian (based on cross-script variants for 0D31 expected from the Myanmar proposal). This issue affects the sequence 0D31 0D31 and its variants. For details see Section 3.1, “ Cross-script overlapped variants” below.

 ***IP Recommendation on new cross-script variants***

*(1) Define cross script variants for 0D31 to match other expected LGR-4 LGRs (1002, 1077 and 10D8)*

*(2) Define equivalent mappings for the pair 0D31 0D31 (and its variant 0D31 0D4D 0D31), e.g. 1002 1002, and so on.*

*Response: Done*

1. *Edits in the document:*
	1. *Added 1077 and 0D31 mapping in section 6.2.3.*
	2. *Added section 6.2.4 for cross-script variant with Georgian.*
	3. *Added section 6.2.5 for overlapped cross-script variant of 0D31. These sequences also added to table 7a.*
2. *Edits in the XML*
	1. *Added 1002, 1077, 10D8 code point and added variant mapping for each code point.*
	2. *Add mapping of 0D31 0D31 and its variant.*
	3. *Add mapping of 0D31 0D31 0D4D 0D31 and its variants.*

A4: WLE rule update: After detailed review of the issue, we now believe that the chosen resolution for the inconsistencies in the formulation of the WLE may be overly broad. Supporting a generic pattern
<0D7B ൻ CHILLU N, 0D4D ◌് VIRAMA, **[any consonant here]**>
when the intent appears to be to support a single exception for
<0D7B ൻ CHILLU N, 0D4D ◌് VIRAMA, 0D31 റ RRA>
seems unnecessary when a focused exception for <0D7B, 0D4D, 0D31> is desired.

By defining a sequence <0D7B, 0D4D, 0D31> the revised LGR already provides an implicit override for the original WLE rule 1. Because a WLE for a code point is never applied to that code point when it appears inside a sequence, the definition of sequence <0D7B, 0D4D, 0D31> would effectively override WLE 1 as it applies to 0D4D in the context of this sequence; therefore, *no modification for the rule* would be needed. The inconsistency in the published formulation of the rules could be resolved by making the XML match the document’s WLE as published, not the other way around.

However, as part of the recommendations for variants, IP suggest a change of <0D7B, 0D4D, 0D31> to <0D7B, 0D4D>, but with a context rule for the code point sequence: when(followed-by-0D31). Even though 0D31 would no longer be part of the sequence, the effect of adding it would still be to allow 0D4D to follow 0D7B if and only if 0D4D is followed by 0D31.

***IP Recommendation:***

*(1) The IP suggests that the GP remove the “or 0D7B” from both WLE1 and the comment attribute for the XML version of the rule. Also to rename the XML version of the rule to drop reference to 0D7B and finally to fix the XML rule itself (by removing 0D7B).*

*Response: see below, after (3)*

This recommendation assumes that a sequence <0D7B, 0D4D> is defined with context rule when(follows-0D31). However, this sequence already has a context rule, not-when(follows-B-X-or-H). As a result, a new merged context rule needs to be created: when(follows-other-than-B-X-or-H-and-followed-by-0D31). Note that the “follows-other-than-B-X-or-H” part would be equivalent to “follows-C-L-M-or-V” because those four classes complete the repertoire, and the rule would probably best be implemented in the XML using the union of those four sets anyway. Alternatively, all the context rules for chillus could be changed from “not-when” contexts to “when” rules with B-X-or-H changed to C-L-M-or-V. That’s a bit of an editorial choice. Whichever choice the GP makes, all documentation in Section 7 and the <description> need to agree.

***IP Recommendation:***

*(2) Create a context rule “follows-other-than-B-X-or-H-and-followed-by-0D31” and apply it to sequence <0D7B, 0D4D>. (See note above for alternative naming suggestions).
(3) Document this merged rule in the <description>*

*Response:*

1. *Edits in the document:*
	1. *Removed ‘or 0D7B’ from the WLE Rule 1, section 7.1.2.*
	2. *Added Rule 9 and its description*
2. *Edits in the XML*
	1. *Removed ‘or 0D7B’ from the rule name and updated the code for Rule1*
	2. *Added Rule 9, and assigned to 0D7B 0D4D sequence: when (follows-C-L-M-or-V-and-followed-by-0D31*
	3. *Added Rule 9 in the description*

A5: Changes needed in the XML: The XML file needs a couple of fixes as well.

* The <date> element needs to be updated to the current date – Updated to 26 March 2020
* The <version> needs to be set to 4 (because the update will likely be part of LGR 4).- Done
* The <description> in its “Overview” section needs a sentence acknowledging that this is an update. - Done
* The section on “Variants” needs to mention any new sequence. (Any counts of sequences needs to be checked). – Done – Added two paragraphs in Variant section.
* Additional context rules on variants need to be documented in the <description>. - Done
* The change in WLE 1 (naming and content) will need to be documented – Done

And, of course, the changes to the actual specification of the sequences, the context rules and WLE1 and or variants need to be reflected in the <data> and <rules> sections.

*The IP requests the GP to make these updates to the XML.*

A6: Stability of “nta” conjunct encoding: In light of the fact that implementers seem to disagree on how to support this conjunct and the further fact that it is in current discussion in the Unicode Technical Committee, a case could be made asking for removal of all support for the stacked "nta" conjunct in the LGR for the reason that the encoding of that conjunct is "unstable".

***IP Recommendation:***

*The IP is monitoring this issue and has not come to a final conclusion. No action is requested at this time; however, we provide a summary of our best understanding of the issue below (For details see Section 4 “*Background on “nta”*”).*

Response: Noted.

# Overlapped Variant

In the LGR update as proposed, there are three variant sequences ending in 0D31 റ‎:

ന്റ 🡨 🡪 ൻ്റ 🡨 🡪 ൻറ

There are also several sets of variant sequences starting with 0D31, for example:

റ്റ 🡨 🡪 ററ

A conflict arises when a label contains overlapped variants, where the final ‎റ of a variant sequence of the first set is also the starting റ of a variant sequence of the second set, as in this label:

ന്ററ (0D28 0D4D 0D31 0D31)

This label can be partitioned in two mutually exclusive ways depending on which partition contains a possible variant **bold**: [[1]](#footnote-1)

{**0D28 0D4D 0D31**} {0D31}

{0D28 0D4D} {**0D31 0D31**}

When computing variants, the first partition yields variants from the first set, while the second partition yields variants from the second set. Because the partitions are mutually exclusive due to the overlap, it is not possible to generate a variant label that uses variants from **both** sets.

Broadly speaking, it is as if the label belongs to two different variant label sets, which would make the LGR not well-behaved.

Because ***all*** of the sequences in the first set end in 0D31, (and ***all*** of the sequences in the other set begin with 0D31), that code point is not modified by the variant substitution process. Therefore, we can argue that it does not participate in the mapping, but instead, represents an implicit context rule.

In other words, we could redefine the updated variant set as follows:

ന് 🡨 🡪 ൻ് 🡨 🡪 ൻ : when(followed-by-റ)

This substitution would not affect the computed variants fro labels that contain any of these sequences followed by 0D31 റ. However, it would remove the overlap. In our example:

ന്ററ (0D28 0D4D 0D31 0D31)

This label can now be partitioned so that the two variant sequences do not overlap

{**0D28 0D4D**} {**0D31** **0D31**}

In calculating the variant label we can now substitute these variants independently, so that all permutations are possible and the label belongs to a single variant set.

The new context rule ensures that a label ending in (…0D28 0D4D) or with (0D28 0D4D) not followed by 0D31 does not have a variant. At the same time, the context rule does not interfere with a following variant starting with 0D31 (and for which all mappings also start with 0D31), thus removing the overlap.

The changes would be:

(1) remove trailing 0D31 from the sequences defined for variant mappings for “nta”

(2) add when(followed-by-0D31) as a context rule on the variants

(3) add when(followed-by-0D31) as a context rule on the sequence 0D7B 0D4D (see also A 3 above)

(4) No changes are needed to the existing variant definitions for the several sequences starting with 0D31….

## Cross-script overlapped variants

Since the LGR was originally published, other LGR draft have defined cross script mappings for 0D31 റ, such as Georgian 10D8 ი and Myanmar 1002 ဂ. These cross-script mappings also overlap with the variant mappings defined for 0D31, like

റ്റ 🡨 🡪 ററ

For example, the label ററ can be partitioned in two ways that map differently to the cross-script variants:

{0D31 0D31}

{0D31}{0D31}

The first partition has the sequence (0D31 0D31) which currently has no cross-script variant equivalent. The second partition has two possible cross script mappings, but because of the context rule for singleton 0D31 it is an invalid partition. Therefore, this label does not produce any variants at all, contrary to expectations.

To remedy this problem, the cross-script variants must be defined not only for singleton 0D31 but also for the pair.

# Background on “nta”

The purpose of the updated Malayalam LGR is to deal with reported inconsistencies in handling the following sequences for Malayalam conjunct “nta”:

|  |  |  |
| --- | --- | --- |
| a) | ന്‌ + റ |   0D28 +0D4D +0D31 |
| b) | ൻ + ് + റ  | 0D7B + 0D4D + 0D31 |
| c) | ൻ + റ | 0D7B + 0D31 |
| d) | ൻ + ് + ‍ZWJ + റ | 0D7B + 0D4D + 200D + 0D31 |

The fourth sequence (d) is not allowed in the Root Zone because it contains a ZWJ but needs to be considered in the analysis. (The other three match sequences 1 (a), 1(b) and 1(c) from the revised proposal).

Unicode 5.1.0 added precomposed chillu characters and also defined the sequence (b) as the recommended representation. For various reasons, today, different software packages render some of these sequences differently (stacked or non-stacked) or treat one of them as an unsupported or even ill-formed cluster.

Full details of the analysis can be found in document: <http://www.unicode.org/L2/L2019/19345r-malayalam-nta.pdf>  .

Normally, the fact that renderings are unreliable (depend on the viewing platform) might have argued for dropping LGR support for both “nra” and “nta” altogether. However, as pointed out in <http://www.unicode.org/L2/L2019/19345r-malayalam-nta.pdf>  , there may be a corresponding ambiguity in reading of the non-stacked form. Therefore, making all forms of these sequences variants of each other is perhaps better motivated than it appears at first.

Unicode is currently revising the description of the Malayalam script’s encoding to reflect the fact that the recommended sequence has not become exclusively supported. The issue is sure to be further discussed at the UTC meeting in mid January.

Based on current discussion, Unicode appears unlikely to drop its recommendation for (b) in internal discussions in Unicode it has emerged that the Harfbuzz rendering engine (used by Android but also used on Windows by e.g. FireFox and ThunderBird) does render the stacked version of "nta" if encoded according to Unicode's recommendation, as does Chrome (and, in the near future, Edge). Before that information emerged, there had been some doubt in the IP as to whether any platform supported that sequence at all. It now looks more like an issue of a bug for specific implementations or fonts.

It appears that the alternation of stacked/non-stacked form does not correlate consistently with encoding, while separately and for linguistic reasons, the "reading" of the non-stacked form appears potentially ambiguous as well. Therefore, not all encodings correlate uniquely to phonetics either.

After those clarifications, making all of these sequences blocked variants seems better motivated now, and, if implementations converge (or at least better overlap) in future the user experience may improve further.

Unlike Tamil SHRII, for example, the sequences are not visually identical and do not all have the same range of readings. That does seem to make the choice of blocked rather than allocatable variant more appropriate.

There is a concern that one major rendering engine (Microsoft’s USE) treats this sequence as ill-formed. That platform uses sequence (d) instead, which cannot be part of the Root Zone. It might be appropriate to approach the vendor in question and to raise that issue – perhaps under the umbrella of Universal Acceptability (as it would make some URLs not properly viewable).

1. Please note that in the particular example shown, the top partition would not be valid as it contains a singleton 0D31 following another 0D31; as the overlap in this example is easier to visualize, we are temporarily ignoring the context rule on singleton 0D31. A more complete analysis of all the overlapped labels by the IP demonstrates that accounting for the context rule is not sufficient to prevent the problem. [↑](#footnote-ref-1)