Variant Analysis Discussion - NBGP and IP:

Variant Analysis documents received from the IP provided NBGP a glimpse of things happening behind the scenes at the common LGR level. It provided an in depth analysis of various variant specification mechanisms, their repercussions and the possible solutions. The discussion on possible solutions provided many options, each of which had a different set of challenges. DV LGR team could not formally and affirmatively finalize on one solution without feeling the need for a bit more detailed discussion on the same with the IP. This text tries to put-forth a certain points which the DV LGR team supposes if implemented, may take away the ambiguities that have arisen.

It is understood while writing this text that these inputs may not have comprehensive understanding of the entire gamut of the LGR specification as applicable for all the other scripts under the Root Zone LGR.

Following is the detailed discussion on some of the points.

Transitivity establishment:

Q. Can the transitivity establishment be made automated and compulsory unless otherwise explicitly barred by the specific script LGR?

This will mean that every variant entry needs to be mentioned only once in the XML. Programmatically all the pairs can be made transitive to the maximum extent, unless it is explicitly barred by the specific script LGR (mechanism of specifying the same may need to be worked out if that is a necessary functionality). Given the fact that (at least in the case of blocked variants) full transitivity is anyways required, bringing it in programmatically seems to be a neater solution rather than leaving it to the specific LGR to explicitly mention. This may -

- eliminate the need for manual assessment of the cases whenever any new variant pairs are to be added (every time)

- bring in an uncertainty while debugging any particular issue/bug

Need for hard-coded transitive relation also brings in a factor of typing error / accidental deletion which may bring in even more uncertainty in production time error analysis.

Need for script-boundary establishment:

Q. Script boundaries can be defined by the individual scripts for compensating absence of out-of-repertoire functionality in the common LGR?

The common LGR which will have all the scripts integrated into it will be a super LGR without any out-of-repertoire code-point. This takes away the much-needed functionality of having some out-of-repertoire code-points.

There is a possibility by which this probably can be introduced in the LGR.

E.g. with some script level tag like

<boundary min="0900" max="097F" />

Or

<boundary script="deva" min="0900" max="097F" /> in case it has to be globally mentioned.

The above mechanism can provide two kinds of benefits:

- It may facilitate formalization of boundary analysis even for those scripts which require limited script mixing.

- Will eliminate need for creation of variant context rules only to prevent mix-script labels.

Alternative to Index Variant Mechanism:

Q. Can the index variant mechanism be changed?

The mechanism of index variants is indeed ingenious and solves the problem in a much efficient manner than keeping a track of all the possible variants and having to iterate through all while matching. However, given the specific place where this will be applied, we are actually looking at a few thousands (probably a few lacs, including variants) entries to which this needs to be applied. This too, at the registration evaluation time which is a one-time process in the life cycle of a label. Having to iterate through those many entries should not be that much compute intensive. It can be easily and definitively handled.

On the other hand we have a compute-efficient (index variant) mechanism which brings in certain kind of complexity given the way LGRs are getting defined. Depending on which script takes precedence over the other, in terms of Unicode values, assessment changes. This appears to be a kind of overhead which can be avoided.

If we could go ahead with the mechanism of enumerating all the variants for now, we can have a system which works definitively albeit in a bit compute intensive way. Eventually, index variant mechanism can be introduced after doing due testing on the actual data.

Since the mechanism of index-variant calculation is "value-based", it gives rise to a kind of ambiguity where certain cases simply work out and certain may not, only based on the value of the constituting variant's code-point values. This appears to be non-definitive way of approaching this problem.

Specific Issues:

Issue of Recursive Nukta Variants:

In this context, the suggestion from the IP i.e. 4-2 seems apt and can be incorporated in the current Devanagari LGR specification.

Specific issues that can arise in the absence of script-mixing rule:

By and large, consonants (and vowels in most cases) in any of the scripts under NBGP, do not require any context. If for the sake of completion of the transitivity, a mixed-script label needs to be introduced in a particular LGR, it may give rise to a situation where the label so formed may not appear to the specification to be invalid in any way, despite being mixed-script.

This may necessitate that a rule be introduced e.g. for those particular consonants which will not be a linguistic one but a one required by the XML specification. This may not be one of the best ways of handling this.

Specific issues raised in relation with Devanagari LGR:

If some of these solutions are feasible, many of the options which are currently suggested would go away. Doing nothing may still solve the problems regardless of the precedence of one script over the other in terms of values.