L2/23-056

Beyond Canonical Equivalence: A discussion on a way forward

Lawrence Wolf-Sonkin

January 2023

Abstract

The Unicode standard provides mechanisms by which to consider distinct sequences of codepoints as equivalent for various purposes. This is very helpful for dealing with the issue of multiply-encodable strings. Unfortunately, for a variety of reasons, many scripts in the standard, but especially Brahmic scripts widespread in South and South-East Asia, contain a plethora of multiply-encodable strings which *are not* formally equivalent by any mechanism in the standard. This leads to a wide variety of downstream issues whereby two strings, in a single script, that seem identical to end users may in-fact have multiple non-equivalent encoded representations. The existing mechanisms of canonical and compatibility equivalence have strong stability requirements which prevent us from alleviating this there. In this paper, we will outline a potential way forward via a more expansive notion of string equivalence which makes a different set of tradeoffs, preferring correctness and updateability over stability.

Background

For a variety of historical and contemporary reasons, the Unicode standard possesses multiple methods by which to encode certain strings, as described in UAX #15[1][2]. Many of cases involve codepoints which were encoded for compatibility reasons (*compatibility equivalence*) with other text encoding standards, but yet many others deal with fundamental ambiguities which we declare to be equivalent (*canonical equivalence*). These mechanisms are absolutely central to a text standard and are used by implementers to avoid the pitfalls associated with visually indistinguishable strings represented by distinct codepoint sequences, for example q̇ as being encoded either as <q, ̇, ̣> or as <q, ̣, ̇>.

It is quite common in large systems for text-at-rest to be normalized into NFC. This works well for cases lucky enough to be captured already by the Unicode Standard's canonical decompositions, but there is an extremely long tail of strings which are (almost) considered equivalent by the standard, but are not formally considered canonically equivalent (nor compatibility equivalent).

Many of these are *described* in the core specification itself, but lack any weight via existing mecha-

[1]The Unicode Standard, Version 15.0 – Core Specification, Section 2.12 Equivalent Sequences
[2]Unicode Standard Annex 15 (UAX #15)

nisms for declaring such equivalences, such as canonical equivalence. In some cases, the core spec discusses multiple representations of a grapheme, but doesn't quite declare them as equivalent in any formal way. For example, the "eyelash-RA" in Devanagari can be encoded <RRA,VIRAMA> or as <RA, VIRAMA, ZWJ>[3]. A similar example is the Tamil Shri ligature, for which the core specification states[4]:

> Prior to Unicode 4.1, the best mapping to represent the ligature shri was to the sequence <U+0BB8, U+0BCD, U+0BB0, U+0BC0>. Unicode 4.1 in 2005 added the character U+0BB6 TAMIL LETTER SHA and as a consequence, the best mapping became <U+0BB6, U+0BCD, U+0BB0, U+0BC0>. Due to slow updates to implementations, both representations are widespread in existing text. Therefore, **treating both representations as equivalent sequences is recommended**.

It is important to note that, even though the core specification describes these cases in a way that makes them quite similar fundamentally to either compatibility equivalence or canonical equivalence, there is no existing mechanism by which to declare them to be such formally since already encoded characters are subject to Unicode normalization stability requirements[5].

Many other such examples of "nearly equivalent" sequences are described via "Do-Not-Use" tables (15 in total) throughout the core spec, which are especially numerous and productive in the Brahmic scripts. These list a grapheme or glyph alongside its preferred encoding as well as a list of dispreferred encodings. These tables suggest a "preference" by the standard, but isn't particularly actionable as to precisely what this preference means. For example, the way that these sequences should be treated by various sorts of systems, such as input methods, rendering engines, lookup, or textual analysis when they do in-fact appear is left unclear. It should also be noted that some of these "Do-Not-Use" tables are often *exemplary* rather than *exhaustive*, such as the Do-Not-Use table on Devanagari consonant conjuncts.[6]

A potential way forward

The current situation of having semi-normative descriptions of sequences which are treated as either "viable-encodings-but-differently-preferred" or "recommended-to-be-treated-as-equivalent", but which ultimately have no formal relation between each other, is an under-specified state whose ambiguity often leads to unexpected issues to users of these scripts and software which tries to process them.

These sorts of issues have consistently come up during Script Ad-Hoc meetings, where we try to steer new scripts away from these encoding models which would lend themselves to these structures in order to avoid the issues that come along with them. With that said, there are still many scripts already encoded in the standard which already have these non-equivalent multiple-encodings, and the Unicode Standard itself seems to be the appropriate place to try to properly define the relations between these sequences.

[3]The Unicode Standard, Version 15.0 – Core Specification, Section 12.1 Devanagari, Rule R5 and R5a

[4]The Unicode Standard, Version 15.0 – Core Specification, Section 12.6 Tamil, Figure 12-24. Tamil Ligatures for *shri*

[5]Unicode® Character Encoding Stability Policies, Normalization Stability

[6]The Unicode Standard, Version 15.0 – Core Specification, Section 12.1 Devanagari, Table 12-3. Devanagari Consonant Conjuncts

The key features of a solution would be:

- Significantly looser stability requirements than canonical decompositions, in order to allow us to update this from version-to-version of the Unicode Standard
- That canonical equivalence would be a *refinement* of this new wider notion of extended canonical equivalence
  - This would mean that each equivalence class under this new extended methodology would be a merger of some existing canonical equivalence classes

I would be interested in getting the view of the Unicode Technical Committee on a way forward to more formally, systematically, and helpfully define these relations in a way that will help users and implementers, and whether the UTC would find solving this to be in-scope, or whether it would be more appropriately solved elsewhere.

Appendix

Further "light-equivalence" examples

Other such examples are described inline in the standard[7]:

> U+0654 ARABIC HAMZA ABOVE should not be used with U+0649 ARABIC LETTER ALEF MAKSURA. Instead, the precomposed U+0626 ARABIC LETTER YEH WITH HAMZA ABOVE should be used to represent a yeh-shaped base with no dots in any positional form, and with a hamza above.

[7]The Unicode Standard, Version 15.0 – Core Specification, Section 9.2 Arabic, Combining Hamza