UCA DUCET: stop using MAX=1F tertiary weights

Author: Markus Scherer
Date: 2012-aug-30

Some collation elements in DUCET expansions have their tertiary weights set to MAX=0x001F rather than the customary weight according to UTS #10 section 7.2, Tertiary Weight Table.

Proposal

I propose that we remove this from the DUCET generation and instead use the customary weights.

Rationale

- The original reason for setting MAX=1F tertiary weights seems unimportant. While non-overlap encoding is critical for a character encoding, it is not obviously necessary or useful for collation-based sorting and searching.
- Replacing customary tertiary weights by MAX=1F removes the distinctions made by those customary weights.
- Therefore, MAX=1F is not set consistently, avoiding its use in cases where those distinctions are necessary; but that inconsistency requires hacks in the DUCET generation code, the process is hard to describe (and has changed several times), and the resulting collation elements and sort order are hard to predict.

Background

Originally, all DUCET tertiary weights were set to create distinctions such as lowercase vs. uppercase, <font> vs. <circle> decompositions etc., as described in UTS #10 section 7.2, Tertiary Weight Table.

In 2001, someone reported as a “bug” that there were characters X and Y which collation-compare X≠Y but XY=YX. For example, consider the characters

.=U+2024 ONE DOT LEADER
..=U+2025 TWO DOT LEADER
…=U+2026 HORIZONTAL ELLIPSIS

which individually sort differently but .<u>..</u>=<u>..</u>. and ..<u>…</u>=<u>…</u>.. (with the “longer” character underlined for clarity).

This was discussed at UTC #88 (2001-aug-15) for item B.2.3 “Comments on 14651”. Consensus 88-C3 was to "Introduce trailers in compatibility decomposition sequences" and that

was implemented in UTS #10 Version 9 (Unicode 3.1.1, 2002-jul-16).

This change was then proposed in L2/01-330 “US Comments on PDAM 1 to ISO/IEC 14651”, section Algorithmic Consistency Issues, referring to a “separate document … that will describe these consistency problems in detail”.

During work on UCA 6.2, we found that the code for setting the MAX=1F weights had bugs. Some expansions had no collation elements with tertiary=MAX=1F, some had two.

These bugs were fixed so that an intermediate DUCET version consistently set the tertiary weight of the last collation element of a compatibility expansion to MAX=1F.

However, this erased important case distinctions, for example between
LJ=U+01C7 LATIN CAPITAL LETTER LJ
Lj=U+01C8 LATIN CAPITAL LETTER L WITH SMALL LETTER J

There was an additional bug, marking certain expansions with the MAX tertiary although they were intended to have exactly the concatenation of their source collation elements. In particular, the Thai/Lao order-reversing contractions had MAX tertiary weights which resulted in prevowel+consonant≠consonant+prevowel. This is unintentionally different from the original UCA handling which algorithmically reversed prevowel/consontant pairs.

For the final UCA 6.2 DUCET, the generation code was adjusted to set the MAX weight only in certain cases, mostly for non-alphabetic characters but also for Roman numerals with “I”. This also fixed the Thai/Lao contractions.

(We had to adjust the UTS #10 description of how expansion mappings for compatibility decomposable characters can be derived.)

Richard Wordingham noted in a comment on CLDR ticket #5192 that

> The algebraic characterisation of misbehaviour looks nice, but I would remark that "s" sorts before "ss" but concat("s", "ss") and concat("ss", "s") sort the same!

In other words: s<u>ss</u>=<u>ss</u>s

It is not clear that examples involving compatibility decomposable characters pose any more practical problems than Richard’s example.

Without MAX weights we would get:

> I<u>II</u>=<u>II</u>I (U+2160 ROMAN NUMERAL ONE & U+2161 ROMAN NUMERAL TWO)
> ∬∭=∭∬ (U+222C DOUBLE INTEGRAL & U+222D TRIPLE INTEGRAL)

Ken and Richard noted that with the UCA 6.2 DUCET we still see a form of the original “bug”:

… < .… = … (2024+2024+2024 < 2024+2025 = 2026)

III < Ⅲ = ⅢⅢ (2160+2160+2160 < 2160+2161 = 2162)

On the other hand, for the integrals, because U+222B INTEGRAL itself is not a compatibility variant, we get tertiary differences between three single integrals and all of the compatibility variants for that sequence.

∭ < ∫∬ < ⨌ < ∬∫ (222B+222B+222B < 222B+222C < 222D < 222C 222B)

(This corresponds to using U+002E FULL STOP rather than U+2024 in the example above.)

Then, if we do not set the MAX weights we would get

… = .… = … (all equal)

III = Ⅲ = ⅢⅢ (all equal)

and

∭ < ∫∬ < ∬∫ < ⨌ (the triple integral would sort last)

It is not obvious that either order is significantly better than the other, or that the differences matter.

With alphabetic characters, where the tertiary distinctions are often important and must not be erased by setting the MAX weight, “surprising” results are possible. For example, in UCA 6.2, the following sort the same:

- dz U+01F3 LATIN SMALL LETTER DZ
- dz U+02A3 LATIN SMALL LETTER DZ DIGRAPH
- ᵈ U+0369 COMBINING LATIN SMALL LETTER D + U+1DE6 COMBINING LATIN SMALL LETTER Z
- d U+217E SMALL ROMAN NUMERAL FIVE HUNDRED + U+1DE6 COMBINING LATIN SMALL LETTER Z

The general problem is that fixed combinations of character properties map to a small set of tertiary weights. Multiple characters that share such combinations of properties map to the same collation elements.

Switching to a different model with explicitly defined tertiary order, which would allow tertiary differences despite common properties, would be a large change in the generation code and require a lot more manual work for maintenance of the DUCET. This does not seem necessary.

The one issue to consider in the last example might be whether the combining small letters, with their distinct appearance, should sort the same as other <compat> variants; maybe they should have secondary differences instead, or sort like <super>?