

UCA: Simplify contraction processing

Authors: Markus Scherer, Mark Davis, Ken Whistler, Peter Edberg

Date: 2012-may-04

It is difficult to write an implementation of the Unicode Collation Algorithm (UCA, [UTS #10](#)) contraction processing that is both conformant and efficient. For performance, implementations need to minimize or avoid modifications of the input string during processing. Certain degenerate cases (which do not occur in normal text) add a lot of complexity or force modifications of the input. These cases involve “interleaved” contractions, such as <0FB2, 0F71, 0F71, 0F80, 0F74>. For full background and details see document [L2/12-108](#), “[UCA: Problems with discontinuous contractions](#)”.

Proposal

We propose a new requirement on contraction mappings in Collation Element Tables and a simplification of the UCA algorithm which together allow significantly simpler implementations that are conformant and efficient.

L2/12-108 offers several alternative proposals. We recommend the following combination (the “Alternative Proposal” with “Option 3”) because it keeps discontinuous-contraction matching and permits all current DUCET and CLDR mappings while eliminating complex code for rare and unusual situations (discontinuous contractions with nested contractions).

Require prefix contractions

1. Change the DUCET:
 - a. Note: This does not modify the ordering.
 - b. Add missing prefix contractions. For example, in UCA 6.1, the following two would have to be added:
0FB2 0F71 ; [.255A.0020.0002.0FB2][.2570.0020.0002.0F71]
0FB3 0F71 ; [.255D.0020.0002.0FB3][.2570.0020.0002.0F71]
 - c. Note: L2/12-108 and the original version of this document claimed that the DUCET needed not be changed. This was a mistake. The “allkeys.txt” file is missing the two contractions above. (However, the CLDR “fractional UCA” files do already contain these contractions.)
2. Change the UCA:
 - a. In section 3.7 Well-Formed Collation Element Tables, require that for a contraction of $n > 2$ code points the n -code point prefix string must also be a contraction. (For example, if "ABC" is a contraction, then "AB" must also be present as a contraction.)
3. Change CLDR:
 - a. Add missing prefix contractions, either manually or via tools. Document the behavior and expectations in the LDML spec.
4. Impact:
 - a. Table builders either auto-generate missing prefix contractions (perhaps with warnings), or report errors when prefix contractions are missing.
 - b. Implementations detect contractions as expected.
 - c. Implementations need not track partial matches nor back up to after the last match.

Note: Auto-generation of missing prefix contractions may not match user expectations. If a user tailors $A+B+n$ as well as $B+C$, and if then $A+B$ is auto-generated resulting in $(CE(A), CE(B))$, then the user would be surprised that $CE(ABC) \neq CE(A), CE(B+C)$.

Limit processing of contractions that start with non-zero combining marks

Ignore contractions that start with non-zero combining marks when they were skipped in discontinuous-contraction matching.

1. No change to the DUCET or to CLDR
2. Change the UCA:
 - a. In section 4 Main Algorithm, specify that for combining marks skipped in Step 2.1, collation elements are fetched by looking up each skipped character in isolation.
3. Impact:
 - a. Only degenerate cases of "interleaved" contractions would be affected: In the modified UCA+DUCET, strings like <0F71, 0F74> sort like in UCA 6.1. However, a degenerate case like <**0FB2, 0F71**, 0F71, **0F80**, 0F74> would sort differently, as *CE(0FB2+0F71+0F80)*, *CE(0F71)*, *CE(0F74)* rather than *CE(0FB2+0F71+0F80)*, *CE(0F71+0F74)*.
(This comparison assumes that the Collation Element Table includes the prefix contraction 0FB2+0F71.)
 - b. Implementations would ignore contraction data for combining marks that are skipped in discontinuous-contraction matching and always fetch the collation elements for each skipped combining mark in isolation. Combining marks that are found in normal processing, rather than skipped in discontinuous contraction, are processed normally, including possible contraction matching.