Response to feedback on draft UCA 6.3 (UTS #10 r27)

L2/13-109R

Author: Markus Scherer
Date: 2013-05-02

This document started with the accumulated feedback as of 2013-May-02.
Responses added begin with "→".
UTS #10 r27 is currently at draft 5. (http://www.unicode.org/L2/L2013/13100-uts10-27-draft.pdf)

Some of the proposed changes to UTS #10 would require fairly significant text changes. I suggest that we postpone those to the next version of UCA, to provide more time for editing and review.

Changes that are specifically proposed for UCA 6.3 are introduced with "*For UCA 6.3*, ...". Some further editorial changes might be appropriate for UCA 6.3, or for the next version of UCA.

Accumulated Feedback on PRI #235

This page is a compilation of formal public feedback received so far. See Feedback for further information on this issue, how to discuss it, and how to provide feedback.

Date/Time: Mon Nov 26 13:17:14 CST 2012
Contact: yoshito_umaoka@us.ibm.com
Name: Yoshito Umaoka
Report Type: Error Report
Opt Subject: UTS#10 - 1.3 Contextual Sensitivity
UTS#10 Unicode Collation Algorithm - 1.3 Contextual Sensitivity
[http://www.unicode.org/reports/tr10/#Contextual_Sensitivity] contains
following description:

Both contractions and expansions can be combined:
that is, two (or more) characters may sort as if they were a different
sequence of two (or more) characters. In the third example, for
Japanese, a length mark sorts like the vowel of the previous syllable:
as an A after KA and as an I after KI.

Also, Table 4. Context Sensitivity contains an example of above:

カー < カイ, but
キー > キイ

The description "as an A after KA and as an I after KI." is correct,
but example is bad. "カー < カイ" is correct, but "キー > キイ" is incorrect.

カー is an contraction and equivalent to カア at level 1, therefore カー < カイ is
correct. キー is also an contraction and its primary weight is equivalent to キイ,
but it is still less than キイ at lower level.

To illustrate the nature of contraction followed by expansion,
the example should be like:

カー < カア < カイ, but
キア < キー < キイ
---

or, simply

カー < カア, but
キア < キー

→ Fixed by draft 5

Feedback above this line was reviewed at UTC #134, Feb 2013.

Date/Time: Wed Feb 13 15:49:25 CST 2013
Contact: richard.wordingham@ntlworld.com
Name: Richard Wordingham
Report Type: Error Report
Opt Subject: UCA conformance test invalid
The UCA conformance test is fundamentally invalid!

The conformance test purports to test an implementation of the UCA by
checking its results when using DUCET. The validity of this test is
presumably based on Conformance requirement C1 Paragraph 1: 'Given a
well-formed Unicode Collation Element Table, a conformant
implementation shall replicate the same comparisons of strings as those
produced by Section 4, Main Algorithm.' However, as stated in Section
3.6.4, DUCET is not well formed. Therefore, a conformant
implementation of the UCA is free to use whatever collation it likes
when instructed to use DUCET!

These problems also apply to Draft 1 of DUCET 6.3.0 and UCA Version 6.3.0 Draft 2.

So far as I am aware, the only well-formedness requirements needed for the collation algorithm to work are WF3 and WF4. I see three options:

1) Fix DUCET to make it satisfy WF5 (already rejected by the UTC).
2) Remove WF5.
3) Reword requirement C1 Paragraph 1 to explain what well-formednesses may not be relied upon by a compliant implementation.

→ Not true. C1 simply makes no statement of conformance in the presence of an ill-formed Collation Element Table. This does not affect the CollationTest* files: Regardless of whether the DUCET is well-formed, a conformant implementation of the Unicode Collation *Algorithm*, together with the DUCET, will yield the ordering of the test files.

→ TODO (UTC consensus): *For UCA 6.3*, replace “Given a well-formed Unicode Collation Element Table” in C1 with “For a given Unicode Collation Element Table”. Conformance to the algorithm need not be bound to a well-formed CET.
→ *Mark said*: The reason we put that in is that we didn't want to have to describe the behavior for an ill-formed CET, because it could be ill-formed in so many ways.

Date/Time: Tue Mar 12 17:19:18 CDT 2013
Contact: richard.wordingham@ntlworld.com
Name: Richard Wordingham
Report Type: Public Review Issue
Opt Subject: PRI 235 / Problems with UTS#10, UCA
Only the first comment strictly related to PRI 235, except in so far as the other comments could be taken as referring to missed opportunities to correct Version 6.2.0.

1 .If the LDML specification is to be updated at the same time as this document, the reference from Section 5.1 to LDML Section 5.14.3 will break, for that section will be moved to a different file.

→ TODO: CLDR 23 has been released, and the LDML reorganized. *For UCA 6.3*, update the reference to “Section 5.14.13, Case Parameters” appropriately.

---

2. Last sentence of 3.6 'All collation elements with primary weights from 1 to that maximum are variables; all other collation elements are not' is wrong.

→ It is not wrong/no action. This sentence simply rephrases the well-formedness conditions WF3 & WF4.

This is false for the CLDR root collation and all tailorings thereof conforming to CLDR rules. Suggest rephrasing paragraph as,

'Primaries for variable collation elements are *barely* interleaved with other primary weights. This allows for more compact storage of memory tables. Rather than using a bit per collation element to determine whether the collation element is variable, the implementation only needs to store the *minimum and* maximum primary value for all the variable elements. All collation elements with primary weights from *that minimum* to that maximum are variables; all other collation elements are not.'

Changes are marked by *...*.

→ Not false for CLDR/no action for UCA. CLDR adheres to how the UCA handles variable primary weights as a contiguous range from the lowest non-ignorable weight to some maximum. (CLDR does tailor the ordering of some characters, and uses a different “last variable” than the DUCET.) One difference is that CLDR assigns special meaning to U+FFFE which must not be treated as “variable”; I made a note of this in CLDR ticket #5962 “specify CLDR collation algorithm”.

---

3. In proposed 9.1 / old 3.6.1, the text says that there may be lines for 'parametric attributes'. However, the BNF provides no productions for them, and it is not clear to me what these attributes may be. Are they just the forward/reverse ordering at level 2 and the form of variable weighting employed, or do they include the full parametric tailoring? Section 3 list things that an abstract 'collation element table' contains, but is it exhaustive and is it meant to include level 2 ordering and variable weighting scheme?

→ Fixed by draft 5. The section is now explicitly labeled as documenting the format of the allkeys.txt file (only). The text “optional lines for parametric attributes, and” was removed. (Previous versions of this text were hold-overs from when the DUCET was published in the form of several files with more varied syntax.)

---

4. A significant tailoring worth mentioning in Section 5 is removing specific contractions. It is an extremely common optimisation in the search collations of the CLDR.

→ Fixed by draft 5

---

5. In Section 5, changing the secondary level direction and variable weighting options should only be described as changing the collation element table if these settings are part of the abstract collation element table. The former does not change the mapping from NFD strings of codepoints to strings of collation elements, and if the latter does, it then creates mappings for the infinite number of sequences of spaces and punctuation followed by one or more

Latin script diacritics.

→ Fixed by draft 5. The new introductory paragraph of the Tailoring section says that not all forms of tailoring modify the table.

---

6. Contrary to Section 5, turning off normalisation does not transform UCETs. In so far as it changes the collation, the collation has ceased to be an application of the UCA, which requires that canonically equivalent strings collate as equal.

→ Fixed by draft 5 (ditto)

---

7. If applying the LDML parametric tailoring numeric="true" is indeed a tailoring in the UCA sense, then the mapping part of a well-formed UCET may be infinite. This has implications for Requirement C1. When is a conformant implementation of the UCA required to handle infinite UCETs? I'd suggest it only be required to do so when the UCET is an LDML parametric tailoring of DUCET or the root CLDR collation. (Obviously it must handle a well-formed infinite UCET correctly or simply fail to handle it.)

→ Fixed by draft 5 (ditto)

Date/Time: Wed Mar 13 14:20:55 CDT 2013
Contact: richard.wordingham@ntlworld.com
Name: Richard Wordingham
Report Type: Error Report
Opt Subject: UTS#10 - Additional contractions in DUCET
At the start of paragraph 5 of Section 3.8 of Draft 4 of Version 6.3.0 of UTS#10, the Unicode Collation Algorithm, it says 'Those are the only classes of contractions allowed in the Default Unicode Collation Element Table.'. It has said this for some time. However, it is not a true statement. There are also 6 NFD and 2 non-NFD contractions for the compatibility decompositions of

U+013F LATIN CAPITAL LETTER L WITH MIDDLE DOT,
U+0140 LATIN SMALL LETTER L WITH MIDDLE DOT,
U+0E33 THAI CHARACTER SARA AM,
U+0EB3 LAO VOWEL SIGN AM,
U+0F77 TIBETAN VOWEL SIGN VOCALIC RR, and
U+0F79 TIBETAN VOWEL SIGN VOCALIC LL.

A simple remedy would be to change 'contractions are necessary where a canonical decomposable character requires a distinct primary weight in the table' in paragraph 3 to 'contractions are necessary or desirable where a

decomposable character requires a distinct primary weight in the table'.

→ TODO: *For UCA 6.3*, I suggest we simply remove the sentence “Those are the only classes of contractions allowed in the Default Unicode Collation Element Table.” Otherwise, we would have to put into place a process of updating this section whenever the set of DUCET contractions changes. It is not necessary for this section to describe all DUCET contractions.

Date/Time: Wed Mar 13 17:57:57 CDT 2013
Contact: richard.wordingham@ntlworld.com
Name: Richard Wordingham
Report Type: Public Review Issue
Opt Subject: PRI 235 - UTS#10 (for Unicode 6.3.0)
These comments are made on Draft 4 of Version 6.3.0.

1) There is no upper bound on the values of weights in a collation element table.
Formerly (Version 6.1.0) it was FFFF, though this was breached by the now discarded
fourth weight in DUCET. Consequently, the following changes are needed:

a) In Section 3.6 'Variable Weighting', it should be noted that FFFF denotes an
arbitrarily selected value greater than the primary weight of any variable
collation element.

→ Agreed/TODO: Collation element tables need not store 16-bit weights. Make appropriate changes to the text.

b) In Section 6.1.2 'L2/L3 in 8 bits', there should be a note that the same
number of bytes should be used for each L1 weight.

→ No action. Weights of different levels can be stored using different, independent designs, as long as the ordering behavior is maintained.

c) In Section 6.2 'Large Weight Values', the starting premise is no longer
necessarily true. The opening sentence could be reworded to:

'Some old implementations may not support more than 65,5345 weight values
(or 65,024 values where zero bytes are avoided. A need for more weight values
can still be accommodated...'

The example from DUCET may need to be reworded if the suggestion in point
(e) is taken up.

→ Agreed/TODO: Collation element tables need not store 16-bit weights. Make appropriate changes to the text.

d) In 6.4 'Avoiding Zero Bytes' it should be noted that analogous preprocessing
is available for weight ranges larger than 16 bits.

→ Agreed/TODO (ditto)

The following opportunities present themselves:

e) Implicit weights (Section 7.1.3) no longer need to be generated as two elements. With suitably revised values of WEIGHT, a single primary weight of BASE + CP could be chosen instead.

→ Agreed/TODO: Add a note that this algorithm is used as part of UCA+DUCET, and that implementations need to adjust it to how they encode primary weights.

f) The principle of a fixed range for trailing weights (Section 7.1.4) makes no sense if there are no bounds on primary weights.

→ Agreed/TODO (ditto)

Indeed, it makes sense for the implicit weights, trailing weights and weights reserved for special collation elements to be aspects of the collation element, especially if 2nd level order and variable weighting are part of the collation element table.

→ No action. I think this is a misunderstanding based on the old introduction to the Tailoring section.

2) There are a few faults in the code example in Section 6.10:

a) The code assumes that primary values FFE0..FFFF are not allowed. However, U+FFFD has primary weight FFFD and the CLDR assigns primary weight FFFF to U+FFFF.

→ TODO: This is true. In general, as stated, the code is incomplete and overly simplistic. It has had bugs before, and I don't think it is particularly valuable. *For UCA 6.3*, I suggest we delete section 6.10 "Flat File Example" and its subsections, including the Sample Code. (http://www.unicode.org/reports/tr10/#Flat_File_Example)

b) Variable weights have an upper and a lower limit, not just an upper limit.

→ No action. The exclusive lower limit is 0, implied by WF3, other text, and by the sample data structure documenting "Every CE below this value that has a *non-zero primary* is variable." (my emphasis)

Date/Time: Wed Mar 13 18:06:54 CDT 2013
Contact: richard.wordingham@ntlworld.com
Name: Richard Wordingham
Report Type: Error Report
Opt Subject: UTS#10 - Error in 'Escape Hatch'
The fault reported below has been present in the UCA since at least Version 6.1.0.

The section entitled 'Escape Hatch' suggests one can encode a large number of secondary differences by varying z and n in 2-element collation keys of the form

[.yyyy.00zz.00ww] [.0000.00nn.0001]

However, the relative orderings of single collating elements would not be stable if the French accents (= level 2 ordering) setting were toggled. To provide stability, it is necessary to use a palindromic 3-element collation key of the form

[.yyyy.00zz.00ww] [.0000.00nn.0001] [.0000.00zz.0001]

→ TODO: *For UCA 6.3*, add a note that secondary-backwards processing needs to compare complete weights, with most significant bits first, even if they are spread out over multiple data structure CEs. Not for UCA 6.3: Consider merging this with the Large Weight Values section.
→ Rename the idiomatic “Escape Hatch” to “Large values for secondary or tertiary weights”.

Date/Time: Sat Mar 16 07:51:47 CDT 2013
Contact: richard.wordingham@ntlworld.com
Name: Richard Wordingham
Report Type: Error Report
Opt Subject: UCA (UTS#10) Conformance Requirements
The definition of the 'standard UCA parametric tailoring' is subcontracted to the LDML specification (UTS#35). An implementation claiming conformance to it should therefore specify the relevant version numbers – LDML and possibly CLDR for data held in FractionalUCA.txt, which is no longer identified by UCA version. For example, there is a proposal to change the meaning of the variableTop parameter (http://unicode.org/cldr/trac/ticket/5016 Comment 8), and I believe it will be accepted. I'm not sure whether this belongs in Requirement C4 or C6.

→ Possible action: Since UTS #10 itself does indeed not specify parametric tailorings any more in detail, I suggest we delete C6.
→ *Mark says*: I don't think so. I remember some other spec referencing these. It could be cleaned up a bit to point directly at LDML.
→ TODO (UTC consensus): *For UCA 6.3*, remove C6. Add a note there, and a statement in the Migration section: A former claim to conformance to C6 is to be interpreted as a claim of conformance to LDML parametric tailoring.

The standard UCA parametric tailoring parameter 'reorder' is in general undefined – see last paragraph of http://unicode.org/repos/cldr/trunk/specs/ldml/tr35-collation.html Section 3.12 or, for the latest released version, Section 5.14.12 of UTS#35 Version 22.1. The text of the LDML specification says, 'The reordering groups for the DUCET are not specified here'. For the LDML, this has also been raised as http://unicode.org/cldr/trac/ticket/5813 .

→ TODO: *For UCA 6.3*, add a note that an implementation that supports reordering based not on CLDR should specify the reordering groups.