L2/12-250

Review of the Specification of the Unicode Collation Algorithm

Author: **Richard Wordingham**
Date: **23 July 2012**

This review was performed to determine whether the specification is adequate for the production of a compliant collator, and to determine any significant allowed but unpredictable variation in behaviour. The document reviewed is UTS#10 Unicode Collation Algorithm (Revision 24, Version 6.1.0). The outcome of this review is a list of observations that I would like to see properly resolved. Some note has been taken of draft changes already made for Unicode 6.2.0, but this review was not composed with the anticipation that all issues will be addressed in time for Unicode 6.2.0, and I would consider it perfectly reasonable if resolution were deferred until Unicode 7.0.

An earlier version of these observations was shared with Mark Davis and Markus Scherer, and many of the observations have been removed because they are adequately resolved by Draft 4 of Unicode 6.2.0, and yet others have been withdrawn because the relevant text has been moved to the UTS#35, the LDML specification. Some observations have been withdrawn because they were wrong or not worth mentioning, and yet others have been reworded because they were not clear. Finally, several observations are covered by L2/12-223 'Anomalous Level 4 Weights in Tables for UCA'. For their benefit, I have retained the original numbering of the observations, which went up to 133. As a result, the following new observations appear out of order:

No. 134 occurs at the beginning.
No. 135 occurs after no. 15.
No. 136 occurs after no. 45.
No. 137 occurs after no .89.

While the number of observations (about 70) may seem daunting, many of them are related. For example,. the anomalous weighting (or general category?) of U+10A7F OLD SOUTH ARABIAN NUMERIC INDICATOR has given rise to many observations. Other observations result from the specification not having kept up with changes. For example, many observations arose because the relevant text assumes that accented Latin characters have a simple mapping to collation elements, as opposed to the expansion that they have nowadays. A few observations arose because some text appears to date back to when all characters lay in the BMP.

Anomalies in the 3rd and 4th level weights in DUCET have been observed. The significant anomalies in the 3rd level weights are recorded in the table at the end of this review.

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
1 ¶7 Obs. 134. Error	“If a German employee at this French company accesses the data”	French v. German is a bad example, as their CLDR standard orders are identical!	
1 ¶7 Obs. 2. Quibble	“If a German employee at this French company accesses the data, the customer names need to show up in the order that meets this employee's expectations—that is, in a German order”	Dividing tasks alphabetically could be disastrous if the recipients work with different understandings of the ordering behind the division. Give an Englishman A-M to handle and a Lithuanian N-Z, no	

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
		one handles those with names starting with Y, as i << y in Lithuanian. I'm not sure what would happen if you gave a Swede N-Z - would he have the nous to treat that as N-Ö and both the Englishman and the Swede handle Å, or would no-one handle Ö?	
1 ¶8 Obs. 3. Quibble	“For example, Swedish and French have clearly specified, distinct rules for sorting ä ... but neither defines the ordering of characters such as Ж, ש, ♬, ∞, ◊, or ⌂.”	One acquainted with the Hebrew alphabet would expect ת > ש > ר. Similarly, one acquainted with Russian would expect Cyrillic to be ordered as in Russian unless interleaved with another script. Indeed, English users have rather stronger expectations for the sorting of Greek than they do for most of the Latin script.	Perhaps add '(Users may, however, have expectations.)'.
1.3 ¶5 Obs. 6.	“In French and a few other languages, however, it is the *last* accent difference that determines the order”	Is there a contradictory French rule for the sorting of multi-accented characters? An ICU code comment says that there is! (The comment might relate to the French sorting of Vietnamese.) The comment, in tblcoll.cpp of ICU 49.1.1, reads: “To make things even trickier, secondary differences (accent marks) are compared starting at the *end* of the string in languages with French secondary ordering. But when comparing the accent marks on a single base character, they are compared from the beginning.”	
C1 ¶2 Obs. 12. Query	“characters supported by that implementation”	Should this be supported characters or supported strings?	
C1 ¶2 Obs. 13. Minor	“any two canonical-equivalent strings as being equal”	Should add that an optional tailoring may allow this requirement to be ignored.	
C1 ¶3 Obs. 14. Query	“same results as if those strings had been transcoded to Unicode.”	May the transcoding be language-sensitive,e.g. inserting CGJ?	Recommend noting that the transcoding may be language-sensitive.

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
2 C4 Obs. 15. Query	*“A conformant implementation must specify the version number of this Unicode Technical Standard.”*	My implementation loads UnicodeData.txt and a collation element table. (It's happy to load fake files, which may occasionally be appropriate, e.g. for beta testing.) What is it required to report? Can I read 'conformance claim' for 'conformant implementation'?	Suggest adding, “for the version of the UCA it implements”.
2 C4 Obs. 135. Editorial	The precise values of the collation elements for the characters may change over time as new characters are added to the Unicode Standard.	Irrelevant – there is no requirement to report the version of DUCET used.	
3 (intro) Obs. 16. Error	“a collation element is an ordered list of three or more 16-bit weights”	The fourth element ranges up to $17{\times}2^{16}$. ($15{\times}2^{16}$ for tabulated values). The other elements are indeed 16 bits.	
3.1 ¶5 Obs. 17. Editorial	Where only plain text ASCII characters are available the fallback notation in *Table 10* can be used.	Replace 'can be used' by 'is recommended'. This document does not claim to be defining a notation for general use – see Section 3.1 ¶2. (Revision 2 of UTR#10 did specify a tailoring language, and *LDML* does.)	
3.6 ¶1 Obs. 23. Better information	“Any code points that are not explicitly mentioned in this table are given a derived collation element, as described in *Section 7, Weight Derivation*.”	Should add, 'Some decomposable codepoints have weights based on those of derived collation elements'.	
3.6 ¶4 Obs. 25. Correction	“vowels cannot simply be weighted by their representation order in strings”	VCC combinations are also needed for Lao (and possibly Tai Viet); Thai collation is mechanical.	For second, suggest adding, 'Language tailorings may need to add contractions for vowel plus consonant cluster.'

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
3.6 ¶7 Obs. 26. Editorial	“These get different tertiary weights, accordingly.”	Comma before 'accordingly' is grammatically incorrect.	
3.6 ¶9 Obs. 27. Clarific- ation	“The weightings in the table are grouped by major categories.”	An algorithm for identifying the boundaries should be given for assigning the weights to the significant classes for conformant parametric tailoring. While the algorithm given as an example looks obvious, it is critically dependent on the exclusions. For example, most primary weights for numbers are in the symbol group in DUCET. Similarly, most scripts have a character equivalent at the primary level to '1', and many have script-specific punctuation. I have checked the algorithm restricting my attention to primary weights from solitary collation elements for single characters regardless of whether they are in NFD. This might not be adequate in subsequent versions of DUCET For the CLDR root collation, it appears that a definition of the boundaries of the groups is given by FractionalUCA.txt. It is possible that an algorithm can be given to translate those boundaries for use with DUCET. Any algorithm needs to be checked for each release of DUCET.	The partitioning points of the primary weights are determined by the following weights: 1. First positive primary weight for a gc = punctuation character. 2. First subsequent primary weight for a gc = Symbol character 3. First subsequent primary weights for a gc = Sc character. 4. First subsequent primary weights for a gc = Number character. (For DUCET, this could be specialised to the primary weight of '0'.) 5. First subsequent primary weight for a non-number – this identifies the start of the Latin script outside the 'core' groups. 6. For each of the other unexcluded recommended scripts except Han, the first subsequent primary weight of a character in the script. 7. For Han, the first primary weight is, almost by definition, FB40 8000.

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
3.6 ¶10 Obs. 28. Omissions		In DUCET 6.1.0, U+10A7F OLD SOUTH ARABIAN NUMERIC INDICATOR is ordered between numbers within the symbol range. This exception, which causes complications elsewhere, is not listed.	
3.6 ¶10 Obs. 30. Omissions	"Some letter modifiers are grouped with general symbols".	As to Exception 5, it is the modifier letters that are ordered with the other types of letter of their script that are the exception to Table 11, not those ordered with the symbols.	
3.6 ¶10 Obs. 31. Omissions		Characters of the common and inherited scripts are stored within other scripts, not grouped together.	
3.6 ¶12 Obs. 33. Editorial	"So as to maintain the *highest* and *lowest* status, in CLDR these values are not further tailorable, and nothing can be tailored to have the same primary weights."	A cross-reference to within UTS#35 Section 5.14 would be useful.	
3.6.1 Obs. 34. Enhance-ment		There should be a remark that this file format is intended to be usable for expressing collations, even though implementations are not *required* to be able to accept input in this format.	
3.6.1 ¶1 Obs. 35. Correction	"<collationElementTable> := <version> "	The version line does not occur in allkeys_CLDR.txt, which is supposed to have the same format.	Suggest changing '<version>' to '<version>?' and adding that the version line is guaranteed for the DUCET and is optional for other collations (e.g. tailorings). (Or should it be prohibited for other collations?)
3.6.1 ¶2 Obs. 36. Query	"@<version> := <major>.<minor>.<variant> <eol>"	Are the three fields decimal numbers, or are letters also allowed in them? Letters might be appropriate for objects other than DUCET. (Perl includes a test version, and the collation method exports the version string.)	

Section Obs. no Nature	Text (or Topic)	Observation	Ameliorating Actions
3.6.1 ¶3 Obs. 37. Query	"<variableChoice> := 'blanked' | 'non-ignorable' | 'shifted'"	The options 'shift-trimmed' and 'ignoresp' are missing. Should they be present?	
3.6.1 ¶4 Obs. 38. Quibble	"The default is for lines to be forwards."	A *line* cannot be forwards or backwards.	Suggest replacing 'lines to be forwards' by 'levels to be forwards'.
3.6.1 ¶5 Obs. 40. Potential problem	<collElement> := "[" <alt> <char> "." <char> "." <char> ("." <char>)* "]"	There should be a requirement that each collation element have the same number of char fields.	
3.6.1 ¶7 Obs. 42. Clarific-ation	0020 ; [*0209.0020.0002.0020] % SPACE	An example from the Gothic script would be useful. It would demonstrate that the 4th weight is not necessarily expressible in 16 bits.	
3.6.1 ¶7 Obs. 43. Clarific-ation	02DA ; [*0209.002B.0002.02DA] % RING ABOVE; COMPATSEQ	The secondary weights of 002B give a false impression. All primary elements now have a weight of 0020, and decomposable Latin characters now usually get two collation elements.	Suggest that in Section 3.2 ¶2 'not match the weights' be expanded to 'not match the weights or style'.
3.6.1 ¶8 Obs. 44. Error	"For completely ignorable collation elements, the fourth level is set to zero"	Details should be explained by reference to Section 7.3 'Fourth-Level Weight Assignments'. In particular, the statement about completely ignorable elements is either false or has zero content.	
3.6.2 Opt 5 Obs. 45. Query	**IgnoreSP**: This option is the same as **Shifted**, except that only the variable characters that are Whitespace or Punctuation are shifted; the Symbol characters are treated as regular (non-variable) characters.	Presumably no weight at or beyond the lowest positive symbol character weight is then variable. The problem is U+10A7F, a punctuation character that occurs within the number subgroup within symbol group.	
3.6.2 ¶6 Obs. 136. Error	"Rather than using a bit per collation element to determine whether the collation element is variable, the implementation only needs to store the maximum primary value for all the variable elements. All collation elements with primary weights from 1 to that maximum are variables; all other collation elements are not."	The application also needs to store the minimum primary value, for U+FFFE may be tailored to be a non-variable with the lowest primary weight. Similarly, '1' should be replaced by 'that minimum'.	

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
3.6.2 ¶6 Obs. 46. Query	"All collation elements with primary weights from 1 to that maximum are variables; all other collation elements are not."	Does this rule also apply for IgnoreSP? The problem character is U+10A7F.	
3.6.3 ¶1 Obs. 47. Minor	"The unmarked characters will have tertiary weights (such as a_3) equal to MIN_3."	Are all hiragana and katakana marked? Surely not in any linguistic sense.	Suggest adding, "(By convention, all kana are 'marked'.)".
3.7 Cond 2 Obs. 50. Error	All Level N weights in Level N-1 ignorables must be strictly less than all weights in Level N-2 ignorables.	This is a typo for 'All Level N weights in Level N-2 ignorables must be strictly less than all Level N weights in Level N-1 ignorables'. However, even this is not a complete statement.	131-C9, except for second 'Level N'. Reword and change to as 'All Level N weights in Level N-1 ignorables must be different to all Level N weights in Level N-2 and lower ignorables.' For all but the level inserted for alternate weighting, the difference relationship must be 'strictly greater than'. For the level inserted for alternative weighting, the difference relationship must be 'strictly less than'. The required difference relationship may be customised level by level. L2/12-227 contains clearer but less comprehensive wording.
3.7 Cond 4 Obs. 51. Query	For all variable collation elements U, V, if there is a collation element W such that $U_1 \le W_1$ and $W_1 \le V_1$, then W is also variable.	What about IgnoreSP and DUCET 6.1.0 U+10A7F? U+10A7F has gc = Po. The characters before and after it in DUCET order are U+10A7E and U+10917, which have gc = No. Which of the three are variable when IgnoreSP is selected?	

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
6.1.3 Obs. 65. Minor	“the original is to be padded with trailing (not leading) zeros”	Endianist language! Padding an array of bytes with trailing zeroes and reinterpreting as unsigned 32-bit integer would fail spectacularly on a little-endian architecture.	Suggest replacing 'the original...' by 'the original is to be packed into the most significant bits first and padded with zeroes in the least significant bits' (or a better rephrasing if available).
6.1.3 Obs. 66. Clarific-ation	“the original is to be padded with trailing (not leading) zeros”	An example of padding would be helpful.	
6.2 Obs. 67. Editorial		It may be worth mentioning that the technique of large weight values is the basis of the 'fractional weight' scheme used by some implementations.	
6.3.1 ¶1 Obs. 69. Grossly misleading	“Thus all of the secondaries that share a single primary can be renumbered to a contiguous range without affecting the resulting order.”	In DUCET, all secondary weights with a non-zero primary are 0020.	See observation on 6.3.1 ¶2.
6.4 Obs. 73. Omiss-ion	“$x \rightarrow 0101_{16}$ + (x / 255)*256 + (x % 255) “	A similar trick is needed for the identical level (and even more so for DUCET 4th level weights), where values range from 0000 to 0x10FFFF. Simply appending UTF-8 will not work because U+0000 must be supported in strings.	
6.5.2 Obs. 81. Editorial	(Compatibility Decompositions)	Section 6.5.2 belongs better in Section 7, especially as Section 6.5 is entitled 'avoiding normalisation', but this section is talking about *using* compatibility decompositions!	
6.5.2 Obs. 82. Editorial	(Compatibility Decompositions)	There should be a cross-reference at Section 6.3.3 Para 2 for deriving weights via compatibility decomposition. Alternatively, just move Section 6.5.2 to Section 6.3.3 as suggested in UTS#10 6.2.0 Draft 4 notes.	

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
6.5.2 Step 3 Obs. 86. Enhance-ment		It may be worth noting that categorisations as small or upper case can be adequately extracted from the collation elements of the decomposition.	
6.5.2 End Obs. 88. Minor	“Some characters cannot be computed in this way”	May be worth mentioning that this approach fails for all of the 34 decompositions to two or more characters starting with SPACE.	
6.5.2 End Obs. 89. Enhance-ment	“Some characters cannot be computed in this way”	Is it worth noting here that for CLDR root, but not DUCET, the derivation fails for U+20A8 and U+FDFC so that they may be treated as currency symbols?	
6.9.1 ¶4 Obs. 137. Error	“To find the collation grapheme cluster boundaries in a string...”	Normalisation issues remain. Is <U+0F75 TIBETAN VOWEL SIGN UU, U+0F7A TIBETAN VOWEL SIGN E> one cluster or two?	
6.9.1 ¶5 Obs. 94. External error	For information on the use of collation graphemes, see [UTS18].	The code in UTS#18 Annex B does not appear to be able to handle interleaving discontiguous grapheme clusters.	Updating UTS#18 Version 15 Annex B to UTS#10 Version 6.2.0 Section 6.9 should resolve the matter.
7.1.1 ¶1 Obs. 95. Query	“Implementations of the Unicode Collation Algorithm may choose to treat such ill-formed code unit sequences as error conditions”	How are such implementations to pass the conformance tests? (A modification to the test rubric is promised, but no draft has yet been published.)	
7.1.1 ¶2 Obs. 96. Query	“The first approach is to weight each maximal ill-formed subsequence as if it were U+FFFD REPLACEMENT CHARACTER.”	How are applications that convert them to U+FFFD to pass the conformance tests? (A modification to the test rubric is promised, but no draft has yet been published.)	
7.1.1 ¶2 Obs. 97. Correction	“A second approach, only applicable to UTF-16 strings, is to generate an implicit weight for any unpaired surrogate code point as if it were an unassigned code point, using the method of *Section 7.1.3, Implicit Weights*.”	The second approach is open to all three encodings, not just to UTF-16.	Remove 'only applicable to UTF-16 strings'

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
7.1.1 ¶2 Obs. 98. Remark		The second approach is required to pass the conformance tests. This comes perilously close to treating them as valid Unicode characters!	
7.1.3 ¶5 Obs. 100. Error	“They are set to a non-zero value in the first collation element and zero in the second.”	This conflicts with Step 4 in Section 7.3. For compatibility with the allkeys.txt and allkeys_CLDR.txt, the fourth weight corresponding to the fourth weight in the DUCET tables needs to have the same non-zero value for both elements.	
7.1.3 ¶5 Obs. 101. Omission		Apart from the DUCET 4th weight, the non-zero value must, however, satisfy well-formedness condition 2 of Section 3.7.	
7.1.3 ¶6. Obs. 102. Minor	“Unassigned code points are also excluded from these first two BASE values.”	Excluding unassigned code points from the CJK Ideograph Extensions is distinctly programmer-hostile. What useful function does it serve?	
Table 8 Row 2 Obs. 104. Improve-ment	Unified_Ideograph=True **AND** ((Block=CJK_Unified_Ideograph) OR (Block=CJK_Compatibility_Ideographs))	Actually, the weights for assigned characters in the CJK Compatibility Ideographs block are all recorded in the DUCET.	
7.1.3 ¶7 Obs. 105. Improve-ment	“the explicit primary weights must be shifted so that none are between each of the BASE values and BASE + 34.”	Should remark that the weights of characters sharing values with the CJK ideographs must also be shifted along with the implicit weights.	
7.2 Obs. 106. Minor		The distinction between 'None' and '<compat>' is not rigidly followed. Some non-compatibility decompositions are weighted as though they were decompositions in the obvious fashion. This should be documented. For example, U+1F150 NEGATIVE CIRCLED LATIN CAPITAL LETTER A, weighted as though <circle> 0041, U+1F170 NEGATIVE SQUARED LATIN CAPITAL LETTER A, weighted as though	

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
		<square> 0041, U+1F197 SQUARED OK weighted as though <square> 004F 004B and U+1F1FB REGIONAL INDICATOR SYMBOL LETTER V weighted as though <compat> 0056. In the reverse direction, we have examples where all hint of a compatibility decomposition is ignored, such as U+0F77 TIBETAN VOWEL SIGN VOCALIC RR <compat> 0FB2 0F81, which has tertiary weight 2.	
7.2 Obs. 107. Enhance-ment		'h' has a very wide range of decompositions to it – it might be worth using it for samples.	
7.2 Obs. 108. Minor		Some tertiary weights are far from obvious. See the table at the foot of this review.	
7.2 Obs. 109. Error	“<small> small hiragana”	Decomposition type should be NONE for small hiragana. Alternatively, it should be made clear that 000F is used for any small non-Hiragana characters, e.g. much of the Small form Variants block.	
7.3 Obs. 110. Error		There are many strange deviations in both DUCET and CLDR root. They were reported together separately to Unicode as L2/12-223.	
7.3 ¶1 Obs. 111. Omission	The assignment of fourth-level weights in the Default Unicode Collation Element Table is done as follows:	This does not explain how to assign weights to contractions. While it may be deduced for most contractions from Rule 2 and the principle of canonical equivalence, there are some contractions that are not covered. There is no obvious rule for vowel swapping for the Indic scripts with user-friendly storage order. Fortunately, at least at Version 6.1.0, the 4th level weights are	The following rules cover many cases: 1) If the key is canonically equivalent to a character, that character is used as the 4th weight. 2) If the key is canonically equivalent to the compatibility decomposition of character, and the key and the character have

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
		irrelevant for the sorting of the characters employed, and UCA-compliant searching is also unaffected by their values.	the same weights for the first three levels, the character is used as the 4th weight. This gives the DUCET 4th weights for: 004C 0387 006C 0387 0E4D 0E32 0ECD 0EB2. 0FB2 0F71 0F80 0FB3 0F71 0F80
7.3 Step 2 Obs. 112. Error	If a character is weighted as an expansion based on a compatibility decomposition or a synthetic expansion, then assign the code point of the character itself as the fourth-level weight for each element of the expansion.	This does not work for CJK characters in DUCET 6.1.0. For all compatibility decompositions yielding a character in a CJK block, the fourth level weight is the code point of that CJK block character. Thus U+2F17 KANGXI RADICAL TEN and U+3038 HANGZHOU NUMERAL TEN are quaternary equal in allkeys.txt. By contrast, the process is followed in the generation of allkeys_CLDR.txt, so there they are only tertiary equal.	
8 (intro) Obs. 114. Editorial		The text in Section 8 before Section 8.1 is too long, and it contains two numbered lists, which further makes referencing difficult.	
8 ¶7 Obs. 117. Clarific-ation	So in a language where "ch" is a contraction, "bac" would not match in "bach" (given the proper user setting).	The conclusion all depends on the definition of grapheme clusters.	Add 'and grapheme cluster definition' to 'user setting'.
8 ¶13 Obs. 120. Clarific-ation	“Whole Word Search, as defined in [UAX29].”	Should alert user here, rather than only in UAX29, that this may depend on further customisation or even, as for Thai, extensive dictionaries.	

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
8 ¶14 Obs. 121. Booby-trap for users	“The parameter *match-boundaries=whole-character* requires that the start and end of a match each be on a grapheme boundary.”	Could add a note that in general the definition of grapheme boundaries will be tailored, and *should* be consistent with the collation. Above all, do not leave the impression that UAX29 gives a complete definition of grapheme clusters.	
8 ¶17 Obs. 123. Query	**DS2.** The pattern string P *has a match at Q[s,e] according to collation C* if C generates the same sort key for P as for Q[s,e], and the offsets *s* and *e* meet the boundary condition B. One can also say P has a match in Q according to C.	So in Danish, do both “å” and “ä” have a match in “aa\u0308”? Are there two Danish matches for “å” in “baaab”?	
8 ¶18 Obs. 124. English	“*canonical* match at Q[s,e]”	That should be '*in* Q[s,e]', not '*at* Q[s,e]'. Compare the preposition use in the previous paragraphs, where 'at' indicates that the whole string matches and 'in' indicates that a substring matches. For example, one should say “cat” has a canonical match *in* “catalogue”, not *at* “catalogue”.	
8 ¶21 Obs. 126. Error	“Note that Whole Word Search as defined in [UAX29] is grapheme complete.”	Sanskrit word boundaries often cross not only aksharas, but even split vowel marks! Final Pali anusvara can appear as the initial nasal in a cluster with the first consonant of the next word. I can find nothing that makes whole word search grapheme-complete. The statement is true if one only uses default word boundary and legacy or extended grapheme clusters, but would then be meaningless for most SE Asian scripts, as they cannot use default word boundaries.	

Section Obs. no Nature	**Text (or Topic)**	**Observation**	**Ameliorating Actions**
8 ¶29 Obs. 130. Quibble	If, for example, the condition is Whole Grapheme, then the matches are restricted to "abc¸|-°|d", thus discarding match positions that would not be on a grapheme cluster boundary. In this case the minimal match would be "abc¸|-°d"	Grapheme clusters under the tailoring don’t have to be default grapheme clusters	
8.2 (intro) Obs. 133. Error	“a character is unmarked if it has the lowest collation weight for that level”	The definition of unmarked is wrong. In DUCET, the lowest tertiary weight is 2, but no primary element for a BMP kana 'letter' has a weight of 2!	We need a definition based on collation elements with the same primary for the secondary weight indicating unmarkedness, and possibly based on same primary and secondary for the tertiary weight. The definition of the latter may have to be worded very carefully; in particular, it must encompass the exceptional treatment of kana.

Unusual Tertiary Weight Assignments

Code-point	General Category	Decom-position	General Category of First Character of Decomposition	Actual Tertiary Weights	Expected Tertiary Weight	Name
309D	Lm	NONE	N/A	02	0E	HIRAGANA ITERATION MARK
30FD	Lm	NONE	N/A	02	11	KATAKANA ITERATION MARK
1D2D	Lm	super	Lu	14 14 1F	1D 1D 1F	MODIFIER LETTER CAPITAL AE
214D	So	NONE	N/A	0A 04 1F	02	AKTIESELSKAB
210F	Ll	font	Ll	02 02	05	PLANCK CONSTANT OVER TWO PI
A7F8	Lm	super	Lu	14 14	1D 14	MODIFIER LETTER CAPITAL H WITH STROKE
1D4E	Lm	NONE	N/A	14	02	MODIFIER LETTER SMALL TURNED I
1F16A	So	super	Lu	14 14	1D 1D	RAISED MC SIGN
1F16B	So	super	Lu	14 14	1D 1D	RAISED MD SIGN
2120	So	super	Lu	14 14	1D 1D	SERVICE MARK
2122	So	super	Lu	14 14	1D 1D	TRADE MARK SIGN
A728	Lu	NONE	N/A	0A 04	08	LATIN CAPITAL LETTER TZ
037A	Lm	compat	Zs	04	02	GREEK YPOGEGRAMMENI
03CF	Lu	NONE	N/A	0A 04 1F	08	GREEK CAPITAL KAI SYMBOL
2D6F	Lm	super	Lo	02	14	TIFINAGH MODIFIER LETTER LABIALIZATION MARK
1A59	Mn	NONE	N/A	04	19	TAI THAM CONSONANT SIGN FINAL NGA
1B000	Lo	NONE	N/A	02	11	KATAKANA LETTER ARCHAIC E
1B001	Lo	NONE	N/A	02	0E	HIRAGANA LETTER ARCHAIC YE
31B3	Lo	NONE	N/A	16 16	2	BOPOMOFO LETTER INNN

U+214D and U+A728 are only peculiar in that the weights can only be deduced from the character shapes, and cannot be deduced from their Unicode properties. It accords perfectly well with its appearance, and is thus similar to many other cases with weights corresponding to a non-standard <compat> decomposition.

U+1D4E is arguably not unusual – perhaps the tertiary weight 0x14 should be deduced from the 'modifier' in its name.