L2/16-323

Proposal for consideration by the UTC and ISO/IEC JTC1/SC2

Request to change the level 4 weights in the CTT

Marc Lodewijck (Brussels, Belgium)
November 1, 2016

The Unicode Collation Algorithm has a Default Unicode Collation Element Table (DUCET), which is data specifying the default collation order for the Unicode and UCS characters, and similarly the ISO/IEC 14651 standard has a Common Template Table (CTT), for the same repertoire. Contrary to what is stated in UTS #10 (“Appendix B: Synchronization with ISO/IEC 14651”), the CTT is not constructed with the UCA “Shift-trimmed” option for variable weighting—should this be the case, it would be misleading—but the fact is that any collation elements in this weighting table have a fourth-level weight (the UCA variable-weighting feature is not part of ISO/IEC 14651).

The default assignment of fourth-level weights in the CTT is as follows:

1. The default behavior in ISO/IEC 14651 is to ignore at the first three levels what UCA calls “variable characters”: only when strings compare “tertiary-equal” (equal up to the level 3) may these characters make a difference. Thus, variable collation elements from the DUCET are reset so that all weights are zero (“IGNORE”) at levels one through three, and the *code point* of the character that maps to the collation element is assigned as the fourth-level weight.
2. If a collation element is ignorable at the first three levels and is a control code (with few exceptions), a format control or a variation selector, “IGNORE” is assigned for the level 4 weight.
3. The *code point* of the character itself is assigned as the fourth-level weight in primary (or “non-ignorable”) and secondary (or “primary ignorable”) collation elements.

A critical look at this approach shall be considered, as it is not guaranteed to produce homogeneous and unflawed results. As things stand, the CTT and the DUCET are not built for the same ordering, even though they should (at least this is claimed in UTS #10).

* * *

In the DUCET, “variable collation elements” are collation elements (marked with an asterisk) with low, but non-nil, primary weights. The Unicode Collation Algorithm provides four options for handling variable-weighted characters. Amongst them, the “Shifted” and the “Shift-trimmed” options enable these characters to be treated as level 4 elements. I will come back to this later. For now, the relevant to what will be described hereafter is the “Shifted” option.

1. With the parameter value “Shifted”, variable collation elements are reset to zero at levels one through three, and a new fourth-level weight is appended—that is, the primary weight `L1` of the collation element `[*L1.L2.L3]` of each variable character is shifted right by three levels (to the fourth level). For example:

```
          *L1  .L2  .L3
2009  ; [*0209.0020.0002] # THIN SPACE
          .0   .0   .0   .L4
2009  ; [.0000.0000.0000.0209] # THIN SPACE

          *L1  .L2  .L3    *L1  .L2  .L3
2049  ; [*0260.0020.0004][*0266.0020.0004] # EXCLAMATION QUESTION MARK
          .0   .0   .0   .L4    .0   .0   .0   .L4
2049  ; [.0000.0000.0000.0260][.0000.0000.0000.0266] # EXCLAMATION QUESTION MARK
```

   In the CTT, accordingly, instead of assigning as the level 4 weight the code point of the character itself, the accompanying “first-level assignments” list should be used, and thus we would have for the above illustrative examples:

```
Current:         <U2009> IGNORE;IGNORE;IGNORE;<U2009> % THIN SPACE
Proposed change: <U2009> IGNORE;IGNORE;IGNORE;<S0020> % THIN SPACE

Current:         <U2049> IGNORE;IGNORE;IGNORE;<U2049> % EXCLAMATION QUESTION MARK
Proposed change: <U2049> IGNORE;IGNORE;IGNORE;"<S0021><S003F>" % EXCLAMATION QUESTION MARK
```

2. A *completely ignorable* collation element is ignorable at all levels (except the “Identical” level, if any):

```
          .0   .0   .0
034F  ; [.0000.0000.0000] # COMBINING GRAPHEME JOINER
          .0   .0   .0   .0
034F  ; [.0000.0000.0000.0000] # COMBINING GRAPHEME JOINER

          .0   .0   .0
A670  ; [.0000.0000.0000] # COMBINING CYRILLIC TEN MILLIONS SIGN
          .0   .0   .0   .0
A670  ; [.0000.0000.0000.0000] # COMBINING CYRILLIC TEN MILLIONS SIGN
```

   The number of characters that are fully ignorable at strength level 4 with the parameter value “Shifted” for variable-weighting is 835 in the DUCET. In the current CTT, there are 457 such characters—hence, 378 more should also map to the empty string. There might be a bug or a mismatch in the table generation algorithm—if appropriate or necessary, action should be taken, otherwise it would be a point of clarification.

   Level 4 weight for U+034F and U+A670, for instance, should be “IGNORE” in the CTT:

```
(C.) <U034F> IGNORE;IGNORE;IGNORE;<U034F> % COMBINING GRAPHEME JOINER
(P.) <U034F> IGNORE;IGNORE;IGNORE;IGNORE % COMBINING GRAPHEME JOINER

(C.) <UA670> IGNORE;IGNORE;IGNORE;<UA670> % COMBINING CYRILLIC TEN MILLIONS SIGN
(P.) <UA670> IGNORE;IGNORE;IGNORE;IGNORE % COMBINING CYRILLIC TEN MILLIONS SIGN
```

3. With the UCA “Shifted” option, characters with primary collation elements, secondary collation elements or tertiary collation elements (there are none in the weighting tables with regard to the

latter) get a high fourth-level weight, higher than that of any variable character—by convention, it is set to FFFF, which is the maximum possible primary weight in the DUCET:

```
          .0   .L2  .L3
0300  ; [.0000.0025.0002] # COMBINING GRAVE ACCENT
          .0   .L2  .L3  .L4
0300  ; [.0000.0025.0002.FFFF] # COMBINING GRAVE ACCENT

          .0   .L2  .L3
0031  ; [.1C3E.0020.0002] # DIGIT ONE
          .0   .L2  .L3  .L4
0020  ; [.1C3E.0020.0002.FFFF] # DIGIT ONE

          .L1  .L2  .L3    .L1  .L2  .L3
2469  ; [.1C3E.0020.0006][.1C3D.0020.0006] # CIRCLED NUMBER TEN
          .L1  .L2  .L3  .L4    .L1  .L2  .L3  .L4
2469  ; [.1C3E.0020.0006.FFFF][.1C3D.0020.0006.FFFF] # CIRCLED NUMBER TEN

          *L1  .L2  .L3    .L1  .L2  .L2    *L1  .L2  .L3
2474  ; [*0317.0020.0004][.1C3E.0020.0004][*0318.0020.0004] # PARENTHESIZED DIGIT ONE
          .0   .0   .0   .L4    .L1  .L2  .L3  .L4    .0   .0   .0   .L4
2474  ; [.0000.0000.0000.0317][.1C3E.0020.0004.FFFF][.0000.0000.0000.0318] # . . .
```

Similarly, in the CTT, the assignment of fourth-level weights for these characters would be as follows:

```
(C.) <U0300> IGNORE;<GRAVE>;<MIN>;<U0300> % COMBINING GRAVE ACCENT
(P.) <U0300> IGNORE;<GRAVE>;<MIN>;<PLAIN> % COMBINING GRAVE ACCENT

(C.) <U0031> <S0031>;<BASE>;<MIN>;<U0031> % DIGIT ONE
(P.) <U0031> <S0031>;<BASE>;<MIN>;<PLAIN> % DIGIT ONE

(C.) <U2469> "<S0031><S0030>";"<BASE><BASE>";"<CIRCLE><CIRCLE>";<U2469> % CIRCLED NUMBER TEN
(P.) <U2469> "<S0031><S0030>";"<BASE><BASE>";"<CIRCLE><CIRCLE>";”<PLAIN><PLAIN>” % . . .

(C.) <U2474> <S0031>;<BASE>;<COMPAT>;<U2474> % PARENTHESIZED DIGIT ONE
(P.) <U2474> <S0031>;<BASE>;<COMPAT>;"<S0028><PLAIN><S0029>" % PARENTHESIZED DIGIT ONE
```

One may wonder why regular characters in the CTT are not, from the outset, recorded with one single <PLAIN> weighting symbol at level 4, as this would meet what is pointed out in paragraph 6.2.2.3 of the International Standard:

> “Subkeys, at the last level, formed with the ‘forward,position’ level processing parameter are formed by forming a subkey as with the ‘forward’ parameter, but for collating elements that are not ignored at all levels but the last one, their last level weighting (list of weights) is replaced by a single weight (call it <PLAIN> here) that is larger than all other weights at the last level in the given tailored table. Collating elements that are ignored at all levels but the last one, retain their weighting according to the given tailored table. . . .”

Anyhow, in addition this excerpt suggests that the default assignment of fourth-level weights in the CTT must be regarded as intended (if one sets aside, very likely, the aforementioned issue with characters expected to map to the empty string).

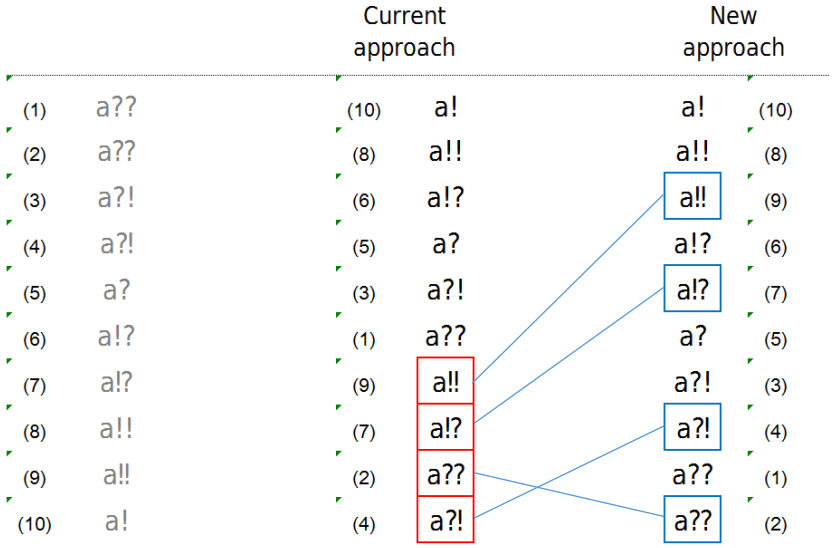
I would like to stress that the current approach, which departs from the rules laid out in UTS #10, can lead to some strange results, which one might truly consider as dubious and that don’t match user expectations, as the examples below will illustrate.

Example #1

Assume the following records, which compare tertiary-equal:

a?? a?? a?! a?! a? a!? a!? a!! a!! a!

We have very different sorting orders:



Following is our detailed data collection, where we just concentrate on the quaternary weights in the sort keys:

Current

		Current CTT	DUCET
(10)	a!	. . . | <PLAIN><U0021>	. . . | FFFF 0260
(8)	a!!	. . . | <PLAIN><U0021><U0021>	. . . | FFFF 0260 0260
(6)	a!?	. . . | <PLAIN><U0021><U003F>	. . . | FFFF 0260 0266
(5)	a?	. . . | <PLAIN><U003F>	. . . | FFFF 0266
(3)	a?!	. . . | <PLAIN><U003F><U0021>	. . . | FFFF 0266 0260
(1)	a??	. . . | <PLAIN><U003F><U003F>	. . . | FFFF 0266 0266
(9)	a\u203C	. . . | <PLAIN><**U203C**>	. . . | FFFF 0260 0260
(7)	a\u2047	. . . | <PLAIN><**U2047**>	. . . | FFFF 0260 0266
(2)	a\u2048	. . . | <PLAIN><**U2048**>	. . . | FFFF 0266 0266
(4)	a\u2049	. . . | <PLAIN><**U2049**>	. . . | FFFF 0266 0260

Proposal

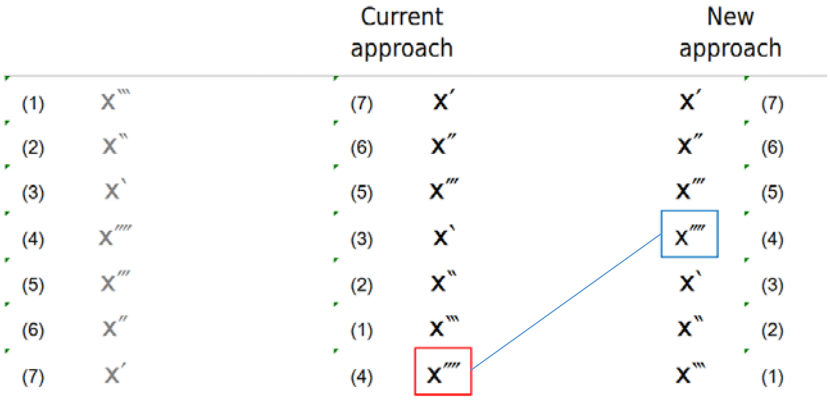
		New CTT	DUCET
(10)	a!	. . . | <PLAIN><S0021>	. . . | FFFF 0260
(8)	a!!	. . . | <PLAIN><S0021><S0021>	. . . | FFFF 0260 0260
(9)	a\u203C	. . . | <PLAIN><**S0021><S0021**>	. . . | FFFF 0260 0260
(6)	a!?	. . . | <PLAIN><S0021><S003F>	. . . | FFFF 0260 0266
(7)	a\u2047	. . . | <PLAIN><**S0021><S003F**>	. . . | FFFF 0260 0266
(5)	a?	. . . | <PLAIN><S003F>	. . . | FFFF 0266
(3)	a?!	. . . | <PLAIN><S003F><S0021>	. . . | FFFF 0266 0260
(4)	a\u2049	. . . | <PLAIN><**S003F><S0021**>	. . . | FFFF 0266 0260
(1)	a??	. . . | <PLAIN><S003F><S003F>	. . . | FFFF 0266 0266
(2)	a\u2048	. . . | <PLAIN><**S003F><S003F**>	. . . | FFFF 0266 0266

```
\u203C: DOUBLE EXCLAMATION MARK
\u2047: DOUBLE QUESTION MARK
\u2048: QUESTION EXCLAMATION MARK
\u2049: EXCLAMATION QUESTION MARK
```

Example #2

Now assume these records:

The sorting results are as follows:



Once again, let's just concentrate on the quaternary weights in the sort keys:

Current

```
                    Current CTT                                    DUCET
 (7)    x\u2032     . . . | <PLAIN><U2032>                         . . . | FFFF 03AA
 (6)    x\u2033     . . . | <PLAIN><U2033>                         . . . | FFFF 03AA 03AA
 (5)    x\u2034     . . . | <PLAIN><U2034>                         . . . | FFFF 03AA 03AA
 (3)    x\u2035     . . . | <PLAIN><U2035>                         . . . | FFFF 03AB
 (2)    x\u2036     . . . | <PLAIN><U2036>                         . . . | FFFF 03AB 03AB
 (1)    x\u2037     . . . | <PLAIN><U2037>                         . . . | FFFF 03AB 03AB 03AB
 (4)    x\u2057     . . . | <PLAIN><U2057>                         . . . | FFFF 03AA 03AA 03AA
```

Proposal

```
                    New CTT                                        DUCET
 (7)    x\u2032     . . . | <PLAIN><S2032>                         . . . | FFFF 03AA
 (6)    x\u2033     . . . | <PLAIN><S2032><S2032>                  . . . | FFFF 03AA 03AA
 (5)    x\u2034     . . . | <PLAIN><S2032><S2032><S2032>           . . . | FFFF 03AA 03AA
 (4)    x\u2057     . . . | <PLAIN><S2032><S2032><S2032><S2032>    . . . | FFFF 03AA 03AA 03AA
 (3)    x\u2035     . . . | <PLAIN><S2035>                         . . . | FFFF 03AB
 (2)    x\u2036     . . . | <PLAIN><S2035><S2035>                  . . . | FFFF 03AB 03AB
 (1)    x\u2037     . . . | <PLAIN><S2035><S2035><S2035>           . . . | FFFF 03AB 03AB 03AB
```

```
\u2032: PRIME
\u2033: DOUBLE PRIME
\u2034: TRIPLE PRIME
\u2035: REVERSED PRIME
\u2036: REVERSED DOUBLE PRIME
\u2037: REVERSED TRIPLE PRIME
\u2057: QUADRUPLE PRIME
```

One could give many more examples, even though collation with the current CTT may in many instances yield perfectly unflawed results.

Besides, should the present proposal be accepted and implemented as designed here (in strict accordance with UCA + DUCET, it must be said again), one caveat should be clear and kept in mind: some “quaternary characters” which currently (with the current CTT) don’t compare as equal will share an identical collation element. For instance, the 17 Unicode characters in the “Separator, Space” (“Zs”) General Category:

```
<U0020> IGNORE;IGNORE;IGNORE;<S0020> % SPACE
<U00A0> IGNORE;IGNORE;IGNORE;<S0020> % NO-BREAK SPACE
<U1680> IGNORE;IGNORE;IGNORE;<S0020> % OGHAM SPACE MARK
<U2000> IGNORE;IGNORE;IGNORE;<S0020> % EN QUAD
<U2001> IGNORE;IGNORE;IGNORE;<S0020> % EM QUAD
<U2002> IGNORE;IGNORE;IGNORE;<S0020> % EN SPACE
<U2003> IGNORE;IGNORE;IGNORE;<S0020> % EM SPACE
<U2004> IGNORE;IGNORE;IGNORE;<S0020> % THREE-PER-EM SPACE
<U2005> IGNORE;IGNORE;IGNORE;<S0020> % FOUR-PER-EM SPACE
<U2006> IGNORE;IGNORE;IGNORE;<S0020> % SIX-PER-EM SPACE
<U2007> IGNORE;IGNORE;IGNORE;<S0020> % FIGURE SPACE
<U2008> IGNORE;IGNORE;IGNORE;<S0020> % PUNCTUATION SPACE
<U2009> IGNORE;IGNORE;IGNORE;<S0020> % THIN SPACE
<U200A> IGNORE;IGNORE;IGNORE;<S0020> % HAIR SPACE
<U202F> IGNORE;IGNORE;IGNORE;<S0020> % NARROW NO-BREAK SPACE
<U205F> IGNORE;IGNORE;IGNORE;<S0020> % MEDIUM MATHEMATICAL SPACE
<U3000> IGNORE;IGNORE;IGNORE;<S0020> % IDEOGRAPHIC SPACE
```

Consequently, these characters will sort the same:

```
<U0020> ≡  <U00A0> ≡  <U1680> ≡  <U2000> ≡   . . .  ≡  <U3000>
```

Which should be compared with the notation for the current root collation data (in binary order):

```
<U0020> <4 <U00A0> <4 <U1680> <4 <U2000> <4  . . .  <4 <U3000>
```

However, should the order of any level 4 characters of this type be relevant and thus not be sorted the same, the fact remains that implementations may still choose to specify arbitrary mappings for tailoring of the collation data, if desired. That said, implementers are also free, ultimately, to assign as the fourth-level weight the code point of the character itself (this can easily be achieved algorithmically), although it must not be recommended—to state otherwise would render the present proposal meaningless.

* * *

Before the sort keys can be compared with a binary comparison, one must choose how to handle the level 4 subkey. There are two options available in the International Standard, and the change requested to be made to the current CTT would not affect them.

Assume the following three records:

```
(1)    ab@    <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<PLAIN><PLAIN><S0040>
(2)    ab     <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<PLAIN><PLAIN>
(3)    a#b    <S0062><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<PLAIN><S0023><PLAIN>
```

NOTE: <S0040> < <S0023> < <PLAIN> (which would read 038E < 0398 < FFFF if the DUCET were used).

There are two possible options for “quaternary characters”:

1. With the “<forward,position> level processing parameter”, the *trailing* high-quaternary weights (the <PLAIN> symbol, here) are removed from the sort keys, and the sorting result is as follows:

```
(2)    ab     <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0
(3)    a#b    <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<PLAIN><S0023>
(1)    ab@    <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<PLAIN><PLAIN><S0040>
```

   Inserting a character only making a level 4 difference anywhere in a string makes it sort *after* the string without it—the string having such a character in the lowest position makes it sort *before* the other string.

2. With the “<forward> level processing parameter” (“position” is set aside), *all* the high-quaternary weights are backed out of each level 4 subkey:

```
(2)    ab     <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0
(1)    ab@    <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<S0040>
(3)    a#b    <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<S0023>
```

   NOTE: <S0040> < <S0023>

   In strings which compare tertiary-equal, the differences are still assessed from the start to the end of the last level, though whithout regard to the position of the level 4 characters in the string—it is only their weights that matter.

   With the current CTT, the outcome would be:

```
(2)    ab     <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0
(3)    a#b    <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<U0023>
(1)    ab@    <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0<U0040>
```

   NOTE: <U0023> < <U0040>

The first option (“<forward,position>”) behaves like the “Shift-trimmed” option provided in the Unicode Collation Algorithm for variable-weighted characters, whereas the second parameter setting isn’t supported by the UCA.

In UTS #10, three other options for variable-weighted characters are described. Amongst them, the “Shifted” option also enables these characters to be treated as level 4 elements. It yields the following ordering:

```
(3)    a#b    [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF 0398 FFFF |]
(2)    ab     [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF |]
(1)    ab@    [1C47 1C60 | 0020 0020 | 0002 0002 | FFFF FFFF 038E |]
```

   The trailing FFFFs are not trimmed from the sort keys. This means that inserting a variable character makes a string sort *before* the string without it, while appending a variable character makes a string sort *after* the string without it.

The “Shifted” option is not conformant to the requirements of the International Standard, even though it could be easily achieved.

The UCA “Blanked” option is the same as the “Shifted” option, except that there is no fourth level. In an implementation of the International Standard, as the UCA variable characters are nil by default on levels one through three (that is, recorded as such in the CTT), the behavior of the option “Blanked” can be obtained simply by setting the collation strength at level 3:

```
                UCA: Strength=4, Variable=Blanked
(1)     ab@     [1C47 1C60 | 0020 0020 | 0002 0002 | |]
(2)     ab      [1C47 1C60 | 0020 0020 | 0002 0002 | |]
(3)     a#b     [1C47 1C60 | 0020 0020 | 0002 0002 | |]

                ISO/IEC 14651: Strength=3
(1)     ab@     <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0
(2)     ab      <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0
(3)     a#b     <S0061><S0062>\0<BASE><BASE>\0<MIN><MIN>\0
```

And finally, with the UCA “Non-ignorable” option, variable collation elements are not reset to be quaternary collation elements:

```
(3)     a#b     [1C47 0398 1C60 | 0020 0020 0020 | 0002 0002 0002 | |]
(2)     ab      [1C47 1C60 | 0020 0020 | 0002 0002 | |]
(1)     ab@     [1C47 1C60 038E | 0020 0020 0020 | 0002 0002 0002 | |]
```

Considering that the UCA variable collation elements are already reset to zero at the first three levels in the CTT and that you can’t recover what you don’t know is lacking, an option which would perfectly mimic the “Non-ignorable” option in an implementation of the International Standard is virtually impossible, unless you have undergone special preliminary preparation.

Appendix: Incorrectness in the text of UTS #10, Appendix B

For consideration by the UTC

In UTS #10, under the heading of “Appendix B: Synchronization with ISO/IEC 14651”, the text reads as follows:

> For each version of the UCA, the Default Unicode Collation Element Table (DUCET) [Allkeys] is constructed based on the repertoire of the corresponding version of the Unicode Standard. The synchronized version of ISO/IEC 14651 has a Common Tailorable Template (CTT) table built for the same repertoire and ordering. The two tables are constructed with a common tool, to guarantee identical default (or tailorable) weight assignments. The CTT table for ISO/IEC 14651 is constructed using only symbols, rather than explicit integral weights, and with the Shift-Trimmed option for variable weighting.

Specifically,

> . . . with the **Shift-Trimmed** option for variable weighting.

… should instead read:

> . . . with the **Shifted** option for variable weighting.

The records stored in the weighting tables (DUCET and CTT) are collation element mappings. The "Shift-trimmed" option provided for handling variable-weighted characters applies to sort keys, not to collation elements. "The CTT . . . is constructed with the Shifted option" would make sense: it means that a fourth-level weight is appended to any collation elements—the primary weight of each variable collation element is *shifted* right to the 4th level, while primary and secondary collation elements get a high fourth-level weight, which, with the "Shift-trimmed" option (in ISO/IEC 14651 implementations: with the "<forward,position> level processing parameter"), will be removed *from the sort key* if it is a trailing weight.

What I have just written, however, is not consistent with the current default assignment of fourth-level weights in the CTT…

Thus,

> The CTT table for ISO/IEC 14651 is constructed using only symbols, rather than explicit integral weights, and with **the Shift-Trimmed option for variable weighting**.

… should instead read something like:

> The CTT table for ISO/IEC 14651 is constructed using only symbols, rather than explicit integral weights, and with **level 4 weights**.

Actually, a correction to the text can wait. A request would need to be submitted at a later stage.

Voilà.

mlodewijck@gmail.com

References

International Organization for Standardization. *Information Technology—International String ordering and comparison—Method for comparing character strings and description of the common template tailorable ordering.* ISO/IEC 14651:2016 Amd 1 + ISO14651_2016_TABLE1_en.txt and ISO14651_2016_TABLE1_fr.txt (a.k.a. Common Template Table [CTT]).

Unicode Technical Standard #10: Unicode Collation Algorithm, unicode.org/reports/tr10/ + allkeys.txt (a.k.a. Unicode Collation Element Table [DUCET]), www.unicode.org/Public/UCA/latest/allkeys.txt