Restoring Canonical Equivalence to Unicode

L2/01-065

Canonical Equivalence and Unicode 3.0.1

According to the Unicode Standard two Unicode sequences are canonically equivalent if they, when correctly rendered, are indistinguishable to the user. The canonical normalisation algorithm described in section 3.6 of the Unicode Standard version 3.0 transforms canonically equivalent Unicode sequences into a common normalised form.

When the ability to encode ligatures with the help of a *zero width joiner* (zwj) was added to the Unicode Standard, with the release of version 3.0.1, this canonical normalisation algorithm was broken. The reason for this is that the change introduced alternative ways to encode those ligatures that had already been given their own code points in the standard. To exemplify this, consider e.g. the following two Unicode sequences:

`U+0061 U+200D U+0065`	`a + zwj + e`
`U+00E6`	æ

When correctly rendered both these sequences are indistinguishably display as æ, but they are not identified as being canonically equivalent by the canonical normalisation algorithm.

One possible way to fix the algorithm would be to give all precomposed ligatures encoded in the Unicode Standard a canonical decomposition so that they in the normalised form would be replaced by the representation containing the zero width joiner. This approach would however violate the stability requirements of the Unicode Standard.

For this reasons it is proposed do the opposite and use the precomposed ligatures in the normalised form. This can be achieved by adding a ligature composition step at the end of the normalisation algorithm. This step consists of scanning the Unicode sequence to be normalised for the presence of zero width joiners (U+200D). If the character immediately preceding a zero width joiner and the character immediately following it can form a ligature that has a precomposed form encoded in the Unicode Standard the three Unicode characters are replaced by the Unicode character representing this precomposed ligature.

There is one potential problem with this approach that arises when ligatures formed from more than two characters are encoded. Consider for example the following Unicode sequences, which all are legal ways to encode a possible fls-ligature and therefore must be normalised to the same Unicode sequence by the algorithm:

`U+0066 U+200D U+006C U+200D U+0073`	`f + zwj + l + zwj + s`
`U+FB02 U+200D U+0073`	`fl-lig + zwj + s`
`U+0066 U+200D U+02AA`	`f + zwj + ls-lig`

With the algorithm described above these three Unicode sequences are not recognised as canonically equivalent. The normalised form of the first sequence is ambiguous and will be equal to either the second or the third sequence depending on weather the scan is started at the start or at the end of the sequence. The second and third form will never be normalised to the same sequence. However, if all ligatures are decomposed into sequences containing zero width joiners before the scan, and if the scan always starts from the beginning of the Unicode sequence, all sequences in the list above will unambiguously be normalised to the form given in the second row.

This decomposition of ligatures into Unicode sequences containing zero width joiners can be done at the same time as the canonical decomposition if canonical decomposition mappings for the ligatures are added to the Unicode Data Base. All ligatures found in any text that does not use the zero width joiner for encoding ligatures (e.g. all texts encoded using versions of Unicode prior to version 3.0.1) will be restored again in the ligature composition step. The normalised forms of such texts are therefore not changed by the proposed change.

Compatibility Mappings

The Unicode Standard does not only define canonical normalisation of Unicode sequences. There is also a compatibility normalisation defined in the standard. In the present version of the Unicode Data Base some of the not so frequent ligatures (e.g. fi-lig and fl-lig) have compatibility mappings to the characters from which the ligatures are formed. These ligatures are therefore replaced by those characters when a Unicode sequence is passed through the compatibility normalisation algorithm. However, the most frequent ligatures (e.g. æ, œ and ß) do not have such mappings and are kept as ligatures, as are the least frequent ligatures that can only be encoded by a sequence containing a zero width joiner (e.g. ct-lig and fj-lig). This behaviour of the algorithm is counter-intuitive and will seem erratic to most users. It is therefore proposed to remove these compatibility mappings so that all ligatures are kept when applying the compatibility normalisation algorithm.

Two of the ligatures encoded in the Unicode Standard have the letter long s (U+017F) as one of its components. Due to the compatibility mapping of long s into a normal s these two ligatures will not be restored during the ligature composition step when the compatibility normalisation algorithm is applied.

For this reason it is proposed that this compatibility mapping is removed from the Unicode Data Base. Such a removal of a compatibility mapping in order to simplify the normalisation algorithm has been done previously in the history of Unicode. The changes introduced in Unicode Data Base version 2.1.9 when the compatibility mappings of some of the Hangul syllable finals were removed in order simplify the Hangul syllable normalisation is one example.

This removal of compatibility mappings will not affect canonical normalisation and any text that has been passed through a previous version of the compatibility normalisation algorithm will still be in normalised form. However, a text containing a long s or a ligature that previously had a compatibility decomposition that has been passed through the new version of the algorithm will not be in normalised form according to the old version. This is however compatible with the stability requirements for Unicode.

Proposed changes to the Unicode Standard

Change rule D20 in section 3.6 in the Unicode Standard version 3.0 to read:

Compatibility decomposition: the decomposition of a character sequence that results from recursively applying *both* the compatibility mappings *and* the canonical mappings found in the names list of *Section 14.1, Character Names List*, and those described in *Section 3.11, Conjoining Jamo Behavior*, until no characters can be further decomposed, then reordering nonspacing marks according to *Section 3.10, Canonical Ordering Behavior*, and then applying the ligature composition algorithm described in *Section 3.13, Canonical Ligature Composition*.

Change rule D23 in section 3.6 in the Unicode Standard version 3.0 to read:

Canonical decomposition: the decomposition of a character sequence that results from recursively applying the canonical mappings found in the names list of *Section 14.1, Character Names List*, and those described in *Section 3.11, Conjoining Jamo Behavior*, until no characters can be further decomposed, then reordering nonspacing marks according to *Section 3.10, Canonical Ordering Behavior*, and then applying the ligature composition algorithm described in *Section 3.13, Canonical Ligature Composition*.

Add a new section 3.13 to chapter 3 of the Unicode Standard version 3.0

3.13 Canonical Ligature Composition

The purpose of this section is to provide an algorithm to obtain a unique normalized representation of ligatures. In the Unicode Standard, ligatures can be represented using a Unicode sequence containing a U+200D "ZWJ" ZERO WIDTH JOINER (see *Section 13.2, Layout Controls*). Some common ligatures have however been encoded as separate Unicode characters. As an example, the ligature "æ", formed by the letters "a" and "e" can be encoded either as the Unicode sequence U+0061 "a" LATIN LETTER SMALL LETTER A + U+200D "ZWJ" ZERO WIDTH JOINER + U+0065 "e" LATIN LETTER SMALL LETTER E or as U+00E6 "æ" LATIN LETTER SMALL LETTER AE. After applying the canonical mappings found in the names list of *Section 14.1, Character Names List* all ligatures are represented by the form containing the zero width joiner. The normalized form is however given by the Unicode character representing the precomposed ligature if one exists.

To put a Unicode sequence in normalized form the sequence is scanned *from it beginning* for occurrences of U+200D "ZWJ" ZERO WIDTH JOINER. When one is found the names list of *Section 14.1, Character Names List* is searched to see if there exists a Unicode character whose canonical decomposition is equal to the three-character sequence consisting of the zero width joiner and the character immediately before and immediately after it in the Unicode sequence that is being normalized. If this is the case, these three characters are replaced by that Unicode character. The scan then continues with the next character in the sequence until the end of the sequence is reached. Note that if the newly formed ligature itself is followed by a zero width joiner it can be part of an even bigger ligature.

Add canonical mappings for the Unicode characters listed in Table 1 to the names list of *Section 14.1, Character Names List* and to the UnicodeData.txt file, replacing the compatibility mappings where present. Also remove the compatibility mapping for long s (U+017F) in those lists. A delta file showing the changes can be found in UnicodeData-delta.txt

Table 1. Ligatures in Unicode and their canonical decompositions.

Ligature	Canonical Decomposition	Ligature	Canonical Decomposition
U+00C6	U+0041 U+200D U+0045	Æ	A + zwj + E
U+00DF	U+017F U+200D U+0073	ß	long-s + zwj + s
U+00E6	U+0061 U+200D U+0065	æ	a + zwj + e
U+0132	U+0049 U+200D U+004A	IJ-lig	I + zwj + J
U+0133	U+0069 U+200D U+006A	ij-lig	i + zwj + j
U+0152	U+004F U+200D U+0045	Œ	O + zwj + E
U+0153	U+006F U+200D U+0065	œ	o + zwj + e

Ligature	Canonical Decomposition	Ligature	Canonical Decomposition
U+025A	U+0259 U+200D U+02DE	schwa with hook	schwa + zwj + rhotic hook
U+025D	U+025C U+200D U+02DE	rev open e with hook	rev open e + zwj + rhotic hook
U+026E	U+006C U+200D U+0292	l-ezh-lig	l + zwj + ezh
U+02A3	U+0064 U+200D U+007A	dz-lig	d + zwj + z
U+02A4	U+0064 U+200D U+0292	d-ezh-lig	d + zwj + ezh
U+02A5	U+0064 U+200D U+0291	dz-curl-lig	d + zwj + z-curl
U+02A6	U+0074 U+200D U+0073	ts-lig	t + zwj + s
U+02A7	U+0074 U+200D U+0283	t-esh-lig	t + zwj + esh
U+02A8	U+0074 U+200D U+0255	tc-curl-lig	t + zwj + c-curl
U+02A9	U+0066 U+200D U+014B	f-eng-lig	f + zwj + eng
U+02AA	U+006C U+200D U+0073	ls-lig	l + zwj + s
U+02AB	U+006C U+200D U+007A	lz-lig	l + zwj + z
U+03DA	U+03A3 U+200D U+03A4	STIGMA	SIGMA + zwj + TAU
U+O3DB	U+03C3 U+200D U+03C4	stigma	sigma + zwj + tau
U+04A4	U+041D U+200D U+0413	cyr-NG-lig	cyr-N + zwj + cyr-G
U+04A5	U+043D U+200D U+0433	cyr-ng-lig	cyr-n + zwj + cyr-g
U+04B4	U+0422 U+200D U+0426	cyr-TTS-lig	cyr-T + zwj + cyr-TS
U+04B5	U+0442 U+200D U+0446	cyr-tts-lig	cyr-t + zwj + cyr-ts
U+04D4	U+0410 U+200D U+0415	cyr-Æ	cyr-A + zwj + cyr-E
U+04D5	U+0430 U+200D U+0435	cyr-æ	cyr-a + zwj + cyr-e
U+0587	U+0565 U+200D U+0582	ech-yiwn-lig	ech + zwj + yiwn
U+05F0	U+05D5 U+200D U+05D5	vav-vav-lig	vav + zwj + vav
U+05F1	U+05D5 U+200D U+05D9	vav-yod-lig	vav + zwj + yod
U+05F2	U+05D9 U+200D U+05D9	yod-yod-lig	yod + zwj + yod
U+0675	U+0627 U+200D U+0674	alef with high hamza	alef + zwj + high hamza
U+0676	U+0648 U+200D U+0674	waw with high hamza	waw + zwj + high hamza
U+0677	U+06C7 U+200D U+0674	arab-u with high hamza	arab-u + zwj + high hamza
U+0678	U+064A U+200D U+0674	yeh with high hamza	yeh + zwj + high hamza
U+0EDC	U+0EAB U+200D U+0E99	ho no	ho sung + zwj + no
U+0EDD	U+0EAB U+200D U+0EA1	ho mo	ho sung + zwj + mo

Ligature	Canonical Decomposition	Ligature	Canonical Decomposition
U+FB00	U+0066 U+200D U+0066	ff-lig	f + zwj + f
U+FB01	U+0066 U+200D U+0069	fi-lig	f + zwj + i
U+FB02	U+0066 U+200D U+006C	fl-lig	f + zwj + l
U+FB03	U+FB00 U+200D U+0069	ffi-lig	ff-lig + zwj + i
U+FB04	U+FB00 U+200D U+006C	ffl-lig	ff-lig + zwj + l
U+FB05	U+017F U+200D U+0074	long-st-lig	long-s + zwj + t
U+FB06	U+0073 U+200D U+0074	st-lig	s + zwj + t
U+FB13	U+0574 U+200D U+0576	men-now-lig	men + zwj + now
U+FB14	U+0574 U+200D U+0565	men-ech-lig	men + zwj + ech
U+FB15	U+0574 U+200D U+056B	men-ini-lig	men + zwj + ini
U+FB16	U+057E U+200D U+0576	vew-now-lig	vew + zwj + now
U+FB17	U+0574 U+200D U+056D	men-xeh-lig	men + zwj + xeh
U+FB4F	U+05D0 U+200D U+05DC	alef-lamed-lig	alef + zwj + lamed

Table 1 contains all Unicode characters that have the words ‘ligature’, ‘ligation’, ‘ligated’ or ‘digraph’ in their names or in the comments in the names list with the exception of

- Croatian digraphs (U+ 01C4 – U+01CC),
- Latin letter inverted glottal stop with stroke (U+ 01BE),
- ligatures having *canonical* decompositions (U+06C0, U+06C2, U+06D2),
- non-spacing marks (U+06D6, U+06D7) and
- Arabic presentation forms.

To this set the l-ezh-ligature and small and capital stigma have been added. I am not an expert on every script encoded in Unicode so I might have missed some ligatures that cannot be found by this simple search for certain words in the Unicode Data Base.

Summary

The canonical equivalence algorithm was broken with the release of version 3.0.1 of the Unicode Standard. By introducing an additional step in the normalisation algorithms and minor changes to the Unicode Data Base canonical equivalence can be reestablished without violating the stability requirements of the standard. Since canonical equivalence is such a central concept in the Unicode Standard, these changes should be introduced in the standard as soon as possible.

Mattias Ellert

mattias.ellert@tsl.uu.se