L2/15-319

Emoji ZWJ Sequence Considered Harmful

Roozbeh Pournader, Google Inc.
November 4, 2015

Background

Experiences with implementing emoji ZWJ sequences sequences in modern software has led into the discovery of several issues which are atypical of how normal text is handled in modern text stacks. Because of that, the author is recommending freezing the existing list, and proposes that UTC adapts a new mechanism based on Edberg et al 2015 (L2/15-252) as early as possible in order to avoid creation of new ones.

Some such problems are listed below (note that this assumes a typical modern text stack, with a lot of the responsibilities separated and run based on Unicode properties and algorithms):

1. The way these sequences are used is not similar to how ZWJ is used with normal text in natural languages. Let's look at these three typical examples of ZWJ in real world:
    a. The Arabic sequence <Heh, ZWJ, Full Stop>, to be rendered as an initial Heh, followed by a full stop. (The sequence is very common in Arabic and Persian text, and is used in abbreviations using the calendar systems starting with the Hegira.) In this case, no ligature is formed, and the natural behavior is treating the sequence as two parts, <Heh, ZWJ> and <Full Stop> for most practical reasons.

    b. The Latin sequence <s, ZWJ, t>, to be rendered as an "st" ligature, if available. In this case, although one glyph may be used to display the sequence, nothing else changes: there's still a potential discretionary hyphenation opportunity between "s" and "t", cursors should be placeable between "s" and "t", backspacing after the sequences should only delete the "t", forward-deleting before the "s" should not delete the "t", etc.

    c. The Devanagari sequence <Ka, Virama, ZWJ, Ssa>, used to form a half-form Ka followed by a SSa. In the sequence, ZWJ is actually used to break the conjunct/akhand typically typically displayed when there is no ZWJ. Although due to some limitations most implementation treat the sequence as one unit, it is arguably better to allow cursoring between the half-form and the second consonant.

(Note that in Indic scripts, ZWJ is almost always used immediately after or before a virama. It's the virama that plays the role of the ligator, so ZWJ can be ignored for most non-display purposes.)

2. Logical ligatures of bidi-neutral characters are not well-defined: In the real world, ligatures tend to have direction and are visual instead of logical. An "fi" ligature is not formed when an "f" precedes an "i", but when an "f" is to left of an "i". Because of this, modern software such as HarfBuzz correctly consider the resolved bidi direction of text to decide if a ligature should be formed.[1]

   The emoji ZWJ sequences throw a wrench in that assumption. Personal communications with participants from existing implementers of the emoji ZWJ sequences has confirmed that they have needed to make various special changes (including a change in font formats?) in their rendering stack in order to get their text stack to handle the sequences properly in RTL text.

   Note that emoji flags, formed by a sequence of two regional indicator symbols, do not have this problem, as the regional indicator symbols are strong left-to-right characters. The tag characters proposed by Edberg et al 2015 are not expected to create such problems either, since they are boundary-neutral. The emoji modifiers are slightly harder, but since they are limited in scope to the character immediately preceding them, they can be treated more gracefully (see also proposal 3 below). The major problems happen when there are arbitrarily-long sequences of bidi neutral characters, which reorder in RTL text.

3. Determining line-breaking opportunities is not well-defined for such sequences. When processing normal text, the existence of a ZWJ in the stream is not a signal for avoiding line breaks. Applications are recommended to ignore ZWJ for line breaking purposes: it's actually valid to break text and insert a discretionary hyphen when a ZWJ is seen in Latin or Arabic text.

   But for the emoji ZWJ sequences, extreme caution should be taken to not break the line around them (but only if they are actually rendered as a unit, which unfortunately needs access to font information).

4. Backspacing is not well-defined for such sequences: In order for software to determine if it should delete the whole sequence, it should consider two things: if the sequence is rendered as a unit by the font, and if the pieces are actually a

[1] The typical exception to the rule is when combining marks are used, when "ligatures" of base+combining marks are formed. Such ligatures are properly handled in bidi environments since the marks are kept with the base character in the bidi algorithm.

part of such a sequence. This is not similar to any existing backspace behavior, when typically either just one character is deleted, or the sequence is deleted only if it has just been entered using one key stroke or it belongs to a limited structured set of limited length (e.g. base+variation selector). Apart from that, the layer that performs backspacing may not have access to rendering-related parts of the data stream.

5. Forward-deleting is not well-defined for such sequences: on top of previous problems with backspacing, since such sequences can be a subset of one another, one needs to do an arbitrary amount of look-forward (while having access to rendering data) in order to delete the cluster properly. Note that while arbitrary amounts of look-forward for forward-deleting is an existing issues for textual data, that can usually be done based on character properties, while one needs access to rendering data for forward-deleting emoji ZWJ sequences.

6. Text selection and cursoring is not well-defined for such sequences: typically, it's better to cursor in the middle of a ligature such as "fi" or "Lam-Alef" (even when the font doesn't provide a cursor position, dividing the width by the number of base characters provides a great heuristic). For the emoji ZWJ sequences it's vastly different: positioning the cursor in the middle of such a ligature should be avoided if a ligature is formed, but not otherwise. The expected logic is contrary to typical behavior of ligatures in normal text. The same applies to selecting text: it's usually OK to let users select a part of the ligature, but not a part of emoji ZWJ sequences.

7. There is no clean way to implement such sequences at different layers of text stacks. Since most Unicode algorithm don't handle the sequences properly, modern text stacks are required to patch these as exceptions in several different layers (text selection, text editing, line breaking, font rendering, bidi, etc). Such workarounds need to be tested vigorously, and are usually prone to missing edge cases, which could cause security or usability issues.

8. Such exceptions are not limited to one typical correct place in the text stack. In modern platforms, there may be several text stacks (such as TextView and WebView in Android) and each of them need character-specific changes at several different layers.

Proposal

The author proposes that:

1. UTC to recommend against creating new emoji ZWJ sequences.
2. UTR #51 to mention the limited list of grandfathered ZWJ sequences, and to suggest avoiding the creation of any such new sequences.

3. All existing emoji ZWJ sequences to be documented in a computer-processable data file in order to enable them to be get hard-coded in various parts of text stacks.
4. The bidirectional class of emoji modifier characters, U+1F3FB to U+1F3FF to be changed from ON (Other Neutral) to NSM (Nonspacing Mark) to make sure they stay with their base emoji.

Bibliography

1. Peter Edberg, Mark Davis, and the Emoji Subcommittee. 2015. “Unicode Customized Emoji (UCE) Proposal.” UTC Document Register L2/15-252, The Unicode Consortium. http://www.unicode.org/L2/L2015/15252-custom-emoji.pdf