Subject: **Fixing breaking properties for emoji**
Date: **2016-01-19**
From: **Mark Davis, [TBD]**
Working Draft: **https://goo.gl/5p3dLx**

Certain sequences of emoji should be displayed with a single glyph if possible. For historical reasons, we have a number types of such sequences. However, these sequences are not yet fully reflected in the specs and data files for segmentation: **TR29** and **TR14**. That means that various bad effects can occur, such as having what the user normally sees as single characters being broken across lines.

While it is certainly possible for implementations to test for exactly those sequences of characters listed in emoji/latest, that is not very robust, nor very fast, nor very future-proof (that is, where an old implementation recognizes a sequence sent to it by a new implementation).

It would be better to incorporate rules into the segmentation UAXes that encompass all the current emoji sequences, and anticipates to some extent possible future sequences with existing characters. Note that as usual with segmentation, it is not a problem to be broader than just the valid emoji sequences, and even prevent some non-emoji sequences from breaking—as long as those sequences don't occur with an significant frequency, or it doesn't matter that they don't break.

The following provides background on the issue, then includes some specific proposals.

We could include these changes first in CLDR v29 segmentation (ie, customizing the Unicode 8.0 rules), and then roll out in Unicode 9.0. The proposals make use of the Emoji properties defined in emoji/latest. If that is a problem for Unicode 9.0 in terms of timing, then the changes could be listed as optional customizations there, or we could point to CLDR.

Contents

Background

The sequence types are listed below, with trailing characters having the following property values. Sentence break is not included, because none of the sequences would allow a break. The first 2 are already handled well enough, and are just presented for comparison. The Tags need a fix for Grapheme Cluster Break (GCB), and the Flags need an additional note for best practice, while the Modifier and Joiner sequences are the ones giving people headaches.

№	Sequences	Examples	GCBreak	WBreak	LBreak
1	*Non-spacing marks*	*20E3, 20E0*	*Extend*	*Extend*	*Combining_Mark*
2	*Variation*	*FE0E, FE0F*	*Extend*	*Extend*	*Combining_Mark*
3	Tags*	E0020...	Control	Format	Combining_Mark
4	Flags	1F1E6...	Regional_Indicator	Regional_Indicator	Regional_Indicator
5	Modifier	1F3FB..F	Other	Other	Alphabetic
	Modifier_Base	261D...	Other	Other	Ideographic
6	Joiner	200D	Extend	Extend	Combining_Mark
	After_Joiner	1F466..1F469, 1F48B, 1F5E8 (current)	Other	Other	Ideographic Alphabetic (❤)

* Tags are included because of prospective customizations using the TAG characters.

The raw data for these is listed in a separate document.

For reference, the rule sets cited below are:

- Grapheme Cluster Boundary
- Word Boundary
- Linebreak Boundary

An important feature is that the rules GB9, WB4, and LB9 cause characters with properties of GCB=Extend; WB=Extend; LB=Combining_Mark to be "absorbed" into the previous characters. That is, there is never a break between them and the previous character, and subsequent rules ignore them.

The vast majority of the Emoji have LB=Ideographic (835) or LB=Alphabetic (143). If we restrict LB changes to those sets, we'd want to strongly discourage the use of any other LB classes in emoji zwj sequences. That shouldn't be a problem since the remainder is a small number of characters, and much less likely to be used in sequences.

Tag Sequences

The Tag characters are currently GCB=Control, which means that they don't extend Grapheme_Clusters. They also have General_Category=Format. So that they work more properly in customization, they should have the same properties as Variation_Selectors. That will cause them to normally "glue" to the previous characters, and otherwise be ignored by subsequent segmentation rules.

Proposal

Change the General_Category and GCB, WB, and LB property values of Tag characters to be the same as those of Variation Selectors. The Bidi_Class property will thus also change to BC=Nonspacing_Mark.

Flag Sequences

The basic support for flag sequences are present in TR29 and TR14, which is not to break between emoji flag sequences. However, more sophisticated algorithms can do better.

Proposal

Add three notes that more sophisticated implementations should break between two regional indicator symbols if and only if they are preceded by an even number of other regional indicators.

The notes could look something like the following, where "(GB8a/WB13c/LB30a)" would be changed to the appropriate rule for the particular segmentation rule set.

Note: Rule (GB8a/WB13c/LB30a) forbids any break between RI characters. So a sequences <RI RI RI RI> and <RI RI RI> will not break. A more sophisticated mechanism will break between any pairs, starting at the first. Thus there would be the following breaks in those sequences <RI RI | RI RI> and <RI RI | RI>.

The rule syntax expressing that would be something like the following. It would replace the existing rule (GB8a/WB13c/LB30a):

Break between two Regional Indicators if and only if there is an odd number of them before the point being considered:

```
[^RI] (RI RI)*     RI    ×     RI
                   RI    ÷     RI
```

Note

ICU text break support for paired regional indicators is being implemented per ticket #11727 using some behavior that cannot be directly expressed in UAX #29 rules.

Modifier Sequences

These are special because they only combine with specific previous characters (modifier bases). When a Modifier doesn't combine, it should be treated like a stand-alone character. So, for example, there can be a line break before it, etc.

However, such sequences will be extremely unusual cases, and for simplicity and future-proofing, it may be better to assign it properties that simply disallow breaks from most previous characters: for example, by giving it the same GC/W/LBreak property values as Variation.

If we want to limit the range of characters, then we'd introduce more new property values and rules.

Proposal—one of the following options:

A. **Simple but broad**
   - Set the GC/W/LBreak property values to be the same as Variation Selectors

B. **Narrow (all and only emoji modifier sequences)**
   - Add GC/W/LBreak property values Emoji_Base (EB) and Emoji_Modifier (EM), and rules such as:

```
GB9c*        Emoji_Base   ×     Emoji_Modifier
WB13d        Emoji_Base   ×     Emoji_Modifier
LB30b              EB     ×     EM
```

C. **Intermediate**
   - Same as narrow, except limit LB property value changes to only LB=Ideographic (835) & LB=Alphabetic (143).

* Rules after 9b would have a different header, and not only apply to extended grapheme clusters

We don't need to have Emoji Variation Selectors listed in the Narrow rules, because all of these rules are after those are absorbed.

Joiner Sequences

The joiners already disallow breaks from the previous characters, but they do allow breaks from the following characters. For grapheme cluster breaks, we don't really have to do much. The text layout software (like Harfbuzz) should be aware of the boundaries, and not let people click in the middle of a sequence that shows up as as single glyph. However, when hitting a delete button, it might be better to remove the trailing joiner. We don't really specify "delete key behavior" in Unicode, however, although we have talked about doing something in CLDR.

For the other segmentations, we again have a choice. The simplest change would be to disallow breaks between a

joiner and most following characters. The joiners are currently targeted for usage with letters from languages that would not allow word- or linebreaks around them anyway, such as between Arabic letters or within Devanagari words.

This would have the slight disadvantage of disallowing (say) linebreaks within <chinese_character> + ZWJ + <chinese_character>. However, those sequences are unnecessary and probably always just accidental. Thus the chances of a negative impact is extremely small.

Even that requires some changes to rules, since we have to distinguish between Joiners and other extending characters.

Proposal—one of the following options:

- **Simple but broad**
  - Separate ZWJ into its own WB, and LB property values, called ZWJ (ZWJ). Use macros to do this so that the current rules using Word_Break=Extend, and Line_Break=Combining_Mark remain the same.
  - Add the following rules:

    ```
    WB3c        ZWJ   ×
    LB8a        ZWJ   ×
    ```
- **Intermediate**
  - Like Narrow, but future-proof the expressions for most emoji by expanding After_Joiners to encompass all characters with GCB=Other; WB=Other; LB=Ideographic; and Emoji=Yes, thus including 769 other emoji.

    ```
    GB9d        ZWJ   ×     After_Joiners
    WB3c        ZWJ   ×     After_Joiners
    LB8a        ZWJ   ×     After_Joiners
    ```
- **Narrow**
  - Only handle the existing emoji joiner sequences. Add property values for After_Joiners (AJ), using macros. The AJ characters are restricted to the ones appearing after ZWJ in the current sequences, and would need to be updated periodically.
  - ```
    GB9d        ZWJ   ×     After_Joiners
    WB3c        ZWJ   ×     After_Joiners
    LB8a        ZWJ   ×     After_Joiners
    ```
- **Narrower**
  - Only handle the existing emoji joiner sequences. Add property values for Adjacent_Joining (AJ), using macros. The AJ characters are restricted to the ones appearing either before or after ZWJ in the current sequences, and would need to be updated periodically.
  - ```
    GB9d        Adjacent_Joining   ZWJ   ×     Adjacent_Joining
    WB3c        Adjacent_Joining   ZWJ   ×     Adjacent_Joining
    LB8a        Adjacent_Joining   ZWJ   ×     Adjacent_Joining
    ```

Note: it may not be worth changing the GCB for joiners, as remarked in the intro to this section.

BIDI emoji sequence order

BIDI order of emoji sequences is *not* a breaking issue, but is a related issue for having to do with how fonts typically handle bidi. Because characters are typically reordered *before* a font sees them, a sequence of MAN ZWJ WOMAN may be seen as WOMAN ZWJ MAN when processing using the font mapping tables. Now, fonts could be extended to handle that, but the simplest way to handle this in current software would if we specify that:

Proposal

> *There is no a semantic difference between an emoji joiner sequence and its reversal.*
>
> More specifically, an implementation might not support display of the reversal of an emoji joiner sequence as a single glyph, but *if* it does, it must produce the same glyph as the emoji joiner sequence. This means that an implementation can't represent MAN ZWJ WOMAN and WOMAN ZWJ MAN with different (single) glyphs.

Then to deal with the bidi issue for joiner sequences, the font could simply have both directions in its mapping tables. Flag sequences are not a problem for BIDI, since the RI characters are LTR. The modifiers can be easily handled by overriding the general category in the layout software (such as Harfbuzz).