Subject: **Fixing breaking properties for emoji**
Date: **2016-01-28**
From: **Mark Davis, Emoji Subcommittee**
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: **UAX #29** (grapheme cluster break, word break) and **UAX #14** (line break). 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 customize Unicode segmentation by testing 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).

This document proposes changes to property values and rules in the segmentation UAXes that encompass all the current emoji sequences, and anticipate 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.

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 rows are already handled well enough, and are just presented for comparison.

№	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
	Glue_After_Zwj	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 http://www.unicode.org/L2/L2016/16011-data-file.txt.

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 GCB, WB, and LB property values of undeprecated Tag characters to be the same as those of Variation Selectors, namely: GCB=*Extend*, WB=*Extend*, and LB=*Combining_Mark*.

~~Also change the General_Category to be Mn and the Bidi_Class property to be Nonspacing_Mark (correspondingly), so that they work more like Variation Selectors in other processing. This also affects some additional derived properties:~~

- ~~ID_Continue, Grapheme_Extend will add the tag characters~~
- ~~Other_Default_Ignorable_Code_Point should add the tag characters (to keep them from being removed from Default_Ignorable_Code_Point)~~

Regional Indicator Sequences

The basic support for Regional Indicator sequences is present in TR29 and TR14, which is not to break between Regional Indicator characters: 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>.

Proposal

Change current rules GB8a/WB13c/LB30a to:

> **Break between two Regional Indicators if and only if there is an even number of them before the point being considered:**
>
> ```
> sot   (RI RI)*    RI    ×    RI
> [^RI] (RI RI)*    RI    ×    RI
                    RI   ÷   RI
> ```

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.

Proposal

1. Add new GCB/WB/LB property values E_Base (EB) and E_Modifier (EM)
    - E_Base is the set of characters with Emoji_Modifier_Base=Yes.
    - E_Modifier is the set of characters with Emoji_Modifier=Yes.
2. Add the following rules:

```
GB9c*       E_Base×     E_Modifier
WB13d       E_Base×     E_Modifier
LB30b             EB    ×     EM
```

    * Also insert a different header before 9c, to make it clear that the **9b** header doesn't apply to other rules.
3. Make compensatory rule adjustments:
    - For WB and GCB, the characters with EB and EM had property Other, and thus no other rules need to be changed.
    - For LB, in Unicode 8.0, the Emoji_Modifier_Base characters were LB=ID and the Emoji_Modifier characters LB=AL (but should also have been ID). To account for the change in properties, any rule mentioning ID would be changed to be (ID | EB | EM). This can be done via macros, or directly. For example:

LB22	ID	×	IN	→	(ID | EB | EM)	×	IN
LB23	ID	×	PO	→	(ID | EB | EM)	×	PO
LB24	PR	×	ID	→	PR	×	(ID | EB | EM)

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

Joiner Sequences

The joiners already disallow breaks from the previous characters, but they do allow breaks from the following characters. The proposed change disallows such breaks only between a joiner and characters that actually occur in current cataloged ZWJ sequences. The minimal change to to just deal with the following characters:

Glue_After_Zwj: currently [□-□□□❤]

1F466..1F469	BOY..WOMAN
1F48B	KISS MARK
1F5E8	LEFT SPEECH BUBBLE
2764	HEAVY BLACK HEART

These would be updated with each addition of cataloged emoji ZWJ sequences. Any implementation that supports other ZWJ sequences would need to customize the list of characters in this set in order for the rules described below to work for that implementation.

However, because of the partial overlap between the Glue_After_Zwj and E_Base, the narrowest change would be cause the creation of multiple new properties and complicate the rules. So instead, the proposal is just to the new rules for modifiers in GCB and WB slightly to include the Glue_After_Zwj characters that are not already in them (currently the last 3). This reduces the break opportunities somewhat, but doesn't materially affect the results. The introduced oddities are cases like:

- not word breaking with <❤ EM>
- not line breaking within <X ZWJ ID>

Proposal

- For GCB, WB, LB, add the property value ZWJ, with one character having that value: U+200D ZWJ
- For GCB and WB, add the property Glue_After_Zwj (GAZ), with the above contents.

- For LB, change the property value for U+2764 HEAVY BLACK HEART to ID.
- Add the following rules:

```
GB9d        ZWJ   ×     Glue_After_Zwj
WB3c        ZWJ   ×     Glue_After_Zwj
LB8a        ZWJ   ×     ID
```

- Add compensatory rule adjustments:
  - For WB and GCB in 8.0, the characters with AJ had property Other, and thus no other rules need to be changed. However, with the change for Modifiers, some of them have the property `E_Base`. So change the rules with E_Base to be the union, eg:

```
GB9c*       (E_Base | Glue_After_Zwj)     ×     E_Modifier
WB13d       (E_Base | Glue_After_Zwj)     ×     E_Modifier
```

  - Compensate for the split of ZWJ with the following changes

```
GB9                                       ×     Extend
WB4         X (Extend | Format)*          →     X
LB9         Treat X CM* as if it were X
⇒
GB9                                       ×     (Extend|ZWJ)
WB4   X (Extend | ZWJ | Format)*          →     X
LB9         Treat X (CM | ZWJ)* as if it were X
```