L2/17-434R2

Re: **Emoji Regex & UTS #51 Definitions**
From: **Mark Davis**
Date: **2018-01-25** (originally 2017-12-22)

The Definitions that we have in UTS #51 are fairly complex. Yet in the end we still depend on the validity tests (for flags and tag sequences, for example) to filter the set of emoji sequences. Because of that dependency on validity tests, we could simplify the definitions to make a simple EBNF and Regex for detecting possible emoji characters and sequences. This has advantages for parsing and scanning text for emoji.

Proposal

1. Add a section to the UTS #51 with the following EBNF and Regex expressions and describes their usage in scanning text to identify possible emoji.
2. Remove text_presentation_sequence from emoji sequences.
3. Simplify combining marks in emoji by only allowing the existing keycaps.

Item 1: EBNF and Regex

The following EBNF can be used to scan for *possible* emoji, which can then be verified by performing validity tests as specified in UTS #51. For brevity here, it uses some proposed property aliases, such as EMD for Emoji_Modifier. It is much simpler than the expressions currently in Definitions: you can think of it as the convex hull of what is matched by those definitions. Of course, it also includes some degenerate cases as a by-product of that simplicity, but those are weeded out by validity tests in any event. (We would clarify in the text that the check for valid emoji modifier sequences is also a validity test.)

EBNF	Notes

`possible_emoji :=`
`  flag_sequence`
`| zwj_element (\x{200D} zwj_element)+`
	200D = joiner
	RI = Regional_Indicator

`flag_sequence :=`
`  \p{RI} \p{RI}`	

`zwj_element :=`
`  \p{Emoji} emoji_modification?`	

`emoji_modification :=`
`  \p{EMD}`
`| (\x{FE0F} | \p{Me}) \p{Me}*`
`| tag_modifier`
	Me= Enclosing_Mark
	FE0F = emoji VS
	E00xx are tags
	E007F is the TERM tag.

`tag_modifier :=`
`  [\x{E0020}-\x{E007E}]+ \x{E007F}`	

From this a regex can be generated, as below. While it may seem complex, it is far simpler than what would result from the Definitions, which result in regex expressions which are many times more complicated, and yet still require verification with validity tests.

Regex

```
  \p{RI} \p{RI}
| \p{Emoji} ( \p{EMD}
            | (\x{FE0F} | \p{Me}+) \p{Me}*
            | [\x{E0020}-\x{E007E}]+ \x{E007F} )?
```

```
 (\x{200D} \p{Emoji} ( \p{EMD}
                     | (\x{FE0F} | \p{M}+) \p{M}*
                     | [\x{E0020}-\x{E007E}]+ \x{E007F} )?)+
```

Notes

Note that this **EBNF** shares a characteristic with the UTS #51 Definitions: it is finer-grained than a grapheme cluster. That is, if you have "A<zwj><emoji>", the Unicode 10 grapheme cluster rule GB11 will cause that to be one grapheme cluster, while a scanner based on the **EBNF** will find an emoji *within* that grapheme cluster. (Similarly, a scanner looking for ASCII letters will find an "A" within that same grapheme cluster.)

The proposed Unicode 11.0 GB11 rule is tighter, since it requires an emoji (or emoji-like) character before the ZWJ. So a sequence like "A<zwj><emoji>" will not count as a grapheme cluster under that new rule.

In any event, a subsequent more precise test for valid sequences would also "split" a grapheme cluster like "<RIa><RIb>", by finding it to contain 2 emoji (if "<RIa><RIb>" doesn't form a valid flag). It is a known feature of the stock grapheme cluster rules that they are not built to handle degenerate cases if that makes the rules too complicated.

Item 2: Remove text_presentation_sequence …

The definition currently allows text presentation sequences as part of emoji sequences; yet any sequence containing a text presentation sequence is specifically not presented as emoji, and breaks any enclosing emoji sequence. So change 14b to remove "| text_presentation_sequence"

emoji_combining_sequence :=
(emoji_character | emoji_presentation_sequence ~~| text_presentation_sequence~~)
enclosing_mark*

This change is already reflected in Item 1.

Item 3: simplify combining marks

We have seen no interest in pursuing any enclosing marks other than the keycap, so to reduce complexity we can narrow the definitions 14b and 14c could be narrowed to:

ED-14b. emoji combining sequence — A sequence of the following form:

```
emoji_combining_sequence :=
( emoji_character | emoji_presentation_sequence | emoji_keycap_sequence)
```

ED-14c. emoji keycap sequence — A~~n emoji combining~~ sequence of the following form:

```
emoji_keycap_sequence := [0-9#*] \x{FE0F 20E3}
```

With this change the EBNF and Regex would change from:

from:	to:
`(\x{FE0F} | \p{Me}) \p{Me}*`	`\x{FE0F} \x{20E3}?`