L2/12-287

Title: Comments on bidi parentheses and other similar problems

Authors: Roozbeh Pournader and Behdad Esfahbod (Google)

Date: 2012-07-30

Action: For UTC's information and potential action

Live link: http://goo.gl/D2ROQ

Regarding Microsoft's proposed Bidi Parentheses Algorithm, we agree with Microsoft that it's not always best to just take a keystroke stream, convert it to a character stream character-by-character, apply the UBA to that stream, and show it to users just like that. We support more intelligent solutions and hope that more innovative and user-oriented solutions are implemented by others too.

We are also concerned about the inadequacy of UBA to handle some very common scenarios, if it's used as the sole mechanism for text entry and editing. Here are some such extremely common scenarios:

1. RTL sentences ending in neutral punctuation end up with the punctuation mark displayed on the right side instead of the left side when the paragraph direction is LTR (as happens very frequently with input boxes in LTR web applications). For example, the logical string "A B." is displayed as "B A." while it should be displayed as ".B A".
2. In LTR paragraphs, RTL sentences containing LTR words are displayed broken. For example, the logical string "A B c D E؟" would be displayed as "B A c ؟E D" instead of "؟E D c B A".
3. Neither the plus sign (U+002B) nor either of the minus signs (U+2212 MINUS SIGN and U+002D HYPHEN-MINUS) are usable in Persian text for showing signed numbers: signs that logically precede numbers incorrectly appear to the right of such numbers. The logical text "LIKE -1" is rendered as "1- EKIL" while it should be rendered as "-1 EKIL". The problem exists regardless of the use of ASCII numbers or Extended Arabic-Indic ones. Similar problems exist for text in Arabic, Urdu, etc.
4. European numbers starting an Arabic RTL paragraph are treated differently than when they are in the middle of the same paragraph, making plus and minus signs very hard and confusing to use in bidi contexts. For example, the logical sequence "1-2 NEQ 2-1" is displayed as "1-2 QEN 1-2".
5. Phone numbers appear broken in RTL contexts. For example, the logical "PHONE: +1 (234) 567-8900" will appear in RTL contexts as "8900-567 (234) 1+ :ENOHP" while it should appear as "+1 (234) 567-8900 :ENOHP".
6. Commas and semicolons (both ASCII and European versions) separating LTR phrases break the sentence. For example, the logical string "LIKE a, b, AND c." is displayed as ".c DNA ,a, b EKIL" instead of the correct ".c DNA ,b ,a EKIL".
7. RTL sentences starting with LTR words get categorized as LTR paragraphs. For example, the logical RTL paragraph "1. html IS NICE" is treated as LTR since it starts with an LTR

character. So it is displayed as “1. html ECIN SI.” instead of desired “ECIN SI html .1”. This would be the case even if “1” is an Arabic-Indic or Extended Arabic-Indic digit, since digits are skipped over for the purposes of rule P2.

8. File names and file paths containing RTL characters are displayed in a confusing way. For example, the Unix file path “/home/user/DIR/FILE.txt” is displayed as “/home/user/ELIF/RID.txt” instead of “/home/user/RID/ELIF.txt”.
9. Trademark signs appear at the wrong side of the trade mark name. For example, the phrase “Windows™” incorrectly appears as “™Windows” in RTL contexts.
10. Product names such as “Google+” get disrupted in RTL context, incorrectly appearing as “+Google”, which would be understood as a reference to Google’s plus page or bringing Google into a discussion, instead of referring to the Google+ social network itself.

On the other hand, while BPA reduces some parentheses-related issues, bidi formatting codes would still be needed for rendering very simple text. For example, it’s impossible to get to the visual sequence “401(k) RUOY” in an RTL paragraph without using bidi formatting codes, with or without BPA. The logical sequence “YOUR 401(k)” would result in “(k)401 RUOY” in both standard UBA and UBA+BPA. Trying to modify the logical sequence to something like “YOUR (k)401” doesn’t work either: it will show as “k)401) RUOY” under both UBA and UBA+BPA. The only working solution is using bidi formatting codes, like “YOUR <LRM>401(k)<LRM>” or “YOUR <LRE>401(k)<PDF>”.

Stability and interoperability concerns

We are concerned about stability and interchangeability of information, and we would appreciate it if all conforming implementations of the Unicode Standard make sure the text they send out to other platforms (or to applications developed by others running on their platforms) can be understood and displayed properly, even if the other systems are not intelligent enough, or are intelligent in a different way.

This especially becomes a problem with older systems, where updating them may be difficult or impossible. Old software also stay with us for years: for example, a very powerful Windows tool for examining bidi issues, SC UniPad, has not been updated since 2006, but is still extremely useful to bidi developers, as most important Unicode characters were added before then.

Older systems encountering text created on newer systems would not know about the update to the bidi algorithm: although they have been supporting all the character in the text they have received. If BPA is implemented invisibly, text that looks OK to the content author will now appear in a broken way on other systems. For example, if BPA is silently applied to all text entered in Windows 8, a simple email sent from a Windows 8 machine to a Windows 7 machine may appear broken to the Windows 7 user.

Some other suggested improvements to BPA

1. The implementations that support BPA would depend on actual character codepoints and not just character properties, requiring heavy changes to some existing implementations. We suggest that the number of characters that can pair be reduced to only those that commonly appear in bidi contexts, so those implementations can implement BPA using a limited set of bidi classes without the requirement to look at actual codepoints.
2. Surprises will happen if BPA is used for displaying text created without BPA. For example, assume that a computer typist working at a magazine has been given a handwritten document containing the right-to-left visual sentence

   G (f) e D c (b) A

   and is asked to type it for an article. Not knowing about the bidi controls (like most bidi speakers) the text has been typed as logical

   A (c (b D f) e) G

   after some moving the cursor around, switching the keyboard layout a few times, and playing until the text is displayed right. It was then posted on a web page and printed on paper too, nobody having any problem with it, as it looks OK everywhere. It may not be properly searchable, but no one wants to search for a long sentence, and no one wants to sort individual sentences either. Now, with a browser using the modified algorithm suggested by Microsoft, that same text will be displayed as

   G (c (b D f) e) A

   which would be totally unreadable. We suggest that BPA would not be applied to documents that are not specified to be requiring BPA.
3. Nested matching doesn't appear much in normal text (but only when computer source code or mathematical formulae are mixed with bidi text, which usually calls for markup anyway) and support for nested parentheses may be unnecessarily expensive to implement and test. (Nested bidi embeddings is not comparable, as the bidi embedding controls are invisible and are usually auto-generated.) Removing the nesting feature will also reduce some of the problems like the example specified in issue 2, above. We suggest that the nesting of parentheses be limited to one level.
4. BPA still doesn't catch some logically matching parentheses, like the "YOUR (k)401" example above, resulting in parentheses that don't match visually. BPA may need to be updated to catch more cases.

Acknowledgments

Mark Davis, Aharon Lanin, and Luke Swartz reviewed this document, contributed their insight, and helped the authors with earlier drafts of this document. They may not agree with all the points raised in this document, or not agree as strongly as we do.