L2/13-123

A normalization situation and the BPA

Andrew Glass

May 8, 2013

Problem

The current draft of the Bracket-Pair Algorithm in UAX #9 (rules BD16 and N0) assumes that all bracket pairs consist of a pairing of a single code point with another single code point. However, the case of the LEFT- AND RIGHT-POINTING ANGLE BRACKETS, identified by Roozbeh Pournader, complicates the situation. These angle brackets have canonical decompositions. Therefore, unless the effects of normalization are taken into account by the BPA, a situation would occur in which the pairing operation of the BPA would produce different results before and after normalization.

The relevant code points are:

```
2329;LEFT-POINTING ANGLE BRACKET;Ps;0;ON;3008;;;;Y;BRA;;;;
232A;RIGHT-POINTING ANGLE BRACKET;Pe;0;ON;3009;;;;Y;KET;;;;
3008;LEFT ANGLE BRACKET;Ps;0;ON;;;;;Y;OPENING ANGLE BRACKET;;;;
3009;RIGHT ANGLE BRACKET;Pe;0;ON;;;;;Y;CLOSING ANGLE BRACKET;;;;
```

From the book:

Deprecated angle brackets

These characters are deprecated and are strongly discouraged for mathematical use because of their canonical equivalence to CJK punctuation.

2329 〈 LEFT-POINTING ANGLE BRACKET
→ 003C < less-than sign
→ 2039 ‹ single left-pointing angle quotation mark
→ 27E8 〈 mathematical left angle bracket
≡ 3008 〈 left angle bracket

232A 〉 RIGHT-POINTING ANGLE BRACKET
→ 003E > greater-than sign
→ 203A › single right-pointing angle quotation mark
→ 27E9 〉 mathematical right angle bracket
≡ 3009 〉 right angle bracket

Despite being deprecated, U+2329 and U+232A do occur in real text, and could occur in mismatched pairs together with the canonical equivalent of their proper pair. In such a case, under the current draft of the BPA, the mixed pair would not be identified as a bracket pair before normalization, but would be so after normalization. For example:

Before normalization	Example (RTL paragraph)	After normalization	Example (RTL paragraph)
U+2329 and U+3009	〈a〈bc	U+3008 and U+3009	a 〈bc〉

U+3008 and U+232A	⟨a 〈bc	U+3008 and U+3009	a 〈bc〉

The text of UAX #9 (last bullet in section 3.2) makes an implicit stability guarantee that normalization will not impact the outcome of the Bidirectional Algorithm.

As of Unicode 4.0, the Bidirectional Character Types of a few Indic characters were altered so that the Bidirectional Algorithm preserves canonical equivalence. That is, **two canonically equivalent strings will result in equivalent ordering after applying the algorithm. This invariant will be maintained in the future.**

[This stability guarantee should be raised to an appropriate location rather than being buried in the text of the algorithm.]

Options

Therefore, the current situation with the angle-brackets must be addressed. The following options exist:

- Clarify the policy to exclude deprecated characters
  This would have negative consequences for the stability policy. Any future deprecations would have uncertain status as to whether a stability policy is impacted.
- Document this exception
  This would impact the contract of the stability policy and implementations that depend on it.
- Exclude characters having canonical equivalents from Bidi_Paired_Bracket
  This would create inconsistency in the treatment of brackets under the BPA, and would negatively impact real text usage, e.g., mixed Chinese and Uyghur documents.
- Update the property to include relationships
  This would complicate the format of the Bidi_Paired_Bracket property and increase the burden on implementers.
- Update the text of BD16 to include pairing with a canonical equivalent
  This creates an extra burden for implementers, but can be optimized for.

The last option is likely to be preferable. The proposed change may be to add “or a canonical equivalent” to the statement of BD16 as follows:

BD16. A *bracket pair* is a pair of characters consisting of an opening paired bracket and a closing paired bracket such that the Bidi_Paired_Bracket property value of the former or a canonical equivalent equals the latter or a canonical equivalent and which are algorithmically identified at specific text positions within an isolating run sequence. The following algorithm identifies all of the bracket pairs in a given isolating run sequence:

- Create a stack for elements each consisting of a bracket character and a text position. Initialize it to empty.

- Create a list for elements each consisting of two text positions, one for an opening paired bracket and the other for a corresponding closing paired bracket. Initialize it to empty.
- Inspect each character in the isolating run sequence in logical order.
    - If an opening paired bracket is found, push its Bidi_Paired_Bracket property value and its text position onto the stack.
    - If a closing paired bracket or a canonical equivalent is found, do the following:
        1. Declare a variable that holds a reference to the current stack element and initialize it with the top element of the stack.
        2. Compare the closing paired bracket or a canonical equivalent being inspected to the bracket in the current stack element.
        3. If the values match, meaning the two characters form a bracket pair, then
            - Append the text position in the current stack element together with the text position of the closing paired bracket or a canonical equivalent to the list.
            - Pop the stack through the current stack element inclusively.
        4. Else, if the current stack element is not at the bottom of the stack, advance it to the next element deeper in the stack and go back to step 2.
        5. Else, continue with inspecting the next character without popping the stack.
- Sort the list of pairs of text positions in ascending order based on the text position of the opening paired bracket.

In practice this amounts to running the algorithm as before but including the following additional pairs: U+2329 with U+3009, and U+3008 with U+232A.

Next steps

If the latter solution is accepted, the following changes need to be made:

- Update the wording of UAX#9 to include this change
- Update the reference implementation to reflect the new behavior
- Add conformance tests to check for this case