L2/13-064R

Feedback on the Proposed Update for UAX#9 (unidode.org/reports/tr9-28.html)

Asmus Freytag

April 28, 2o13 (revised)

Fixed limit for levels

The rationale for fixed limit had been to make sure all implementations “top out” at the same place. Keep fixed limit. Make so high as to make “topping out” rare, but still testable. Could tie it to support or presence of isolates? Otherwise old limits, for compatibility?

Overly “algorithmic” definition BD16

I am a bit concerned with the current wording for BD 16. Notionally, in the spec, it's a definition (the subsection is called “Definitions”) but it is written as a rule (i.e. it describes an algorithm, and, in particular, a rather detailed one.)

The problem with that is that, while on can translate the words into code, one gains no higher level understanding of what that code is supposed to achieve, and has to take on trust that the few examples represent all possible cases.

I also found the original description of the algorithm with a "stack" somewhat misleading, since it works only with those types of stack that let you get access to elements in the stack other than the top element. That makes it impossible, for example, to use simple recursion to get the effect of the "stack".

I believe this approach to BD 16 is unnecessary and the language could be replaced by something like this:

> BD 16. A bracket pair is a pair of characters consisting of an opening paired bracket and the nearest following closing paired bracket such that the Bidi_Paired_Bracket property value of the former equals the latter, subject to the following constraints:
>
> - each character of the pair can belong only to one pair,
> - pairs may nest properly, but their spans may not overlap otherwise, and
> - if two potential pairs overlap, only the one starting at a lower text position is considered.
>
> Note that paired brackets can only occur in an isolating run sequence because they are processed in rule N0 after explicit level resolution. See Section 3.3.2 Explicit Levels and Directions.

If, after consideration the UTC were to conclude that this particular part of the algorithm (i.e. the part implementing the first bullet of N0) is tricky enough to warrant some additional guidance in text, I would not object if a more detailed description of an algorithm was included, for example as part of the

implementation notes. However, the reference implementation should normally address the need for giving such an algorithmic example.

The term “opening paired bracket” is really awkward, because there is need to also talk about such characters that are actually “paired” so you get “paired paired brackets” which is unnecessarily cumbersome.

Corresponding small change to rule N0

Currently, rule N0 says:

- Identify the bracket pairs in the current isolating run sequence according to BD16.
- For each bracket-pair element in the list of pairs of text positions...

Instead it should probably say:

- Starting from the beginning of the run, locate all bracket pairs as defined in BD16 and record their starting and ending text positions.
- Process the bracket pairs in order of increasing starting text position.
- For each pair...

Using this wording, there is no need to introduce an explicit concept of a "list" or "stack" in the description of the logical algorithm - these are artifacts of a particular implementation and therefore do not belong in the specification of the generic algorithm.

Making this change also avoids the undefined term "bracket-pair element".

The biggest benefit in my view is that it collects all the “procedural language” inside rule N0 and keeps it out of the definitions BD16.

Optionally, the first bullet could become its own rule, but that would work only if the UTC were to follow my suggestion to not use the “cute” zero- based numbering scheme for N0. (As I mentioned at the top, I see some usability issues with that, so I would be in favor of a new series, e.g. “P1”, “P2”.).

Corresponding note on BD13

The wording of BD13 in the proposed update also uses algorithmic language, but here the algorithm is merely suggested as a possible way of satisfying the definition (“can be computed”). I suggest that this statement is potentially ambiguous, especially if there should be some discrepancy between the wording of the definition and the result of the algorithm (in some as yet undiscovered edge case).

I therefore suggest, that like the case for BD16, the algorithm be made part of the rule that now cites BD13, by removing “*as specified by BD13*” from X10 and appending the algorithm as an item of the rule.

Nested rule issue in X10

The last item in X10 calls for the application of later rules. This is an unusual and unnecessary "nesting" of rules. It also makes a lie of the subheader of 3.33, which is "preparations for implicit processing", because X10, as written, contains the implicit processing steps themselves by reference.

Therefore the bullet in rule X10 should focus on a general statement:

> - *When applying a rule to an isolating run sequence, the rule is applied to all characters of the sequence even if it is discontinuous. In applying a rule to a discontinuous sequence, the last character of each level run in the isolating run sequence is treated as if it were immediately followed by the first character in the next level run in the sequence, if any. The order that one isolating run sequence is treated relative to another does not matter.*

The forward looking statement on which rules apply on a per-isolated run sequence can then become regular text (not a bullet) following X10 and, for example, be worded as such.

> The rules in the following subsections on resolving weak and neutral types and processing implicit levels are applied to each isolated run sequence separately, as noted. While it is immaterial in which order the isolated run sequences are processed, for the same isolated run sequence the rules must, of course, be applied in order.

Each subsub section already notes the limitation to an isolated run sequence, except 3.3.6 where that language needs to be added.

Rewriting the text in this way removes the formal "nesting" with no loss in clarity.

Note that section 3.4 also has a scope statement (per line) for a rule that is not contained in an earlier rule. So the precedent is to place such scope statements with each rule, not a "parent" rule.

Other issue with X10

The start of X10 ("Perform the following steps") is a bit weird. Why not:

> *X10. Compute the set of isolating run sequences based on the the bidirectional types of the characters and the embedding levels assigned by the rules above (X1–X9) and determine the start-of-sequence (**sos**) and end-of-sequence (**eos**).*

With the first two bullets adjusted as follows

> - <algorithm from BD 13 here>
> - *Determine the start-of-sequence (**sos**) and end-of-sequence (**eos**) types, either L or R, for each isolating run sequence. These depend on the higher of the two levels on either side of the sequence boundar… <i.e. as before>*

Minor

1. The subsection “Definitions” could use subheaders to allow locating specific definitions by topic.
2. The number N0 can easily be misread as the word NO.
3. The wording in N0, item c is confusing. Existing:

   > “*Otherwise, if there is a strong type, it is opposite the embedding direction, so test for adjacent strong types as follows:*”

   The problem is the way the “if” is used: by English syntax it applies to the parenthetical corollary, and no longer works as a “flow control” in the algorithm. The attempt to rescue this by the word “so” makes the text confusing (“Otherwise…so test” ??). Here’s a better way:

   > *Otherwise, if any strong type (opposite the embedding direction) is found, test for adjacent strong types as follows:*

4. Beginning of Section 2 “Directional Formatting Characters”

   > *These formatting characters all have the property Bidi_Control, and are divided into three ranges:*

   The number of ranges is incorrect as it does not account for:

   > U+061C ARABIC LETTER MARK

   I would suggest changing “three“ to “following”.

   > *On web pages, the **explicit** directional formatting characters (of all types – embedding, override, and isolate) should be replaced by using the dir attribute and the elements BDI and BDO. For more information, see [UTR20].*

   I would suggest emphasizing the word “explicit” In the sentence quoted above.

   Altogether, because the concept of a e few “*ranges*” is less useful now, esp. with the ALM being in a different location, I would change the entire format of the first listing of codes in section 2 so that there isn’t the duplication of information with the following subsections (2.1 etc) that there is now. I would remove the names, and coalesce the paragraphs into a table. Later in the subsections, each of these is listed anyway with location and full name.

Directional Formatting Characters		
Implicit Directional Marks	200E	RLM
	200F	LRM
	061C	ALM
Explicit Directional Embedding	202A	LRE
	202B	RLE
Explicit Directional Override	202C	LRO
	202D	RLO
Override and Embedding Terminator	202E	PDF
Explicit Directional Isolation	2066	LRI
	2067	RLI
	2068	FSI
Isolate Terminator	2069	PDI

A table like this is much easier to take in at a glance and provides the reader (and implementer) with a succinct overview and listing of locations.