L2/12-173

Bidirectional parenthesis algorithm

Ayman Aldahleh, Gilead Almosnino, Peter Constable, Andrew Glass, Laurentiu Iancu, Dwayne Robinson, Murray Sargent, Robert Steen

Introduction

In its current form the UBA (Unicode Bidirectional Algorithm UAC #9) fails to correctly display instances of parentheses in cases where the boundaries of the parentheses have mixed directionality. A simple example is “a(b)” in an RTL paragraph:

(a(b

Under the UBA in its current form, users, developers, and localizers who wish to obtain the correct display form need to use invisible control characters (Ex: LRE, RLO, PDF) to alter the logical string so that UBA can interpret it correctly. In the simple case above, this could include the following options (not all of which are equally recommended):

[LRE] a (b) [PDF]

[LRO] a (b) [PDF]

a (b) [LRM]

This solution requires users to have detailed knowledge of the way the UBA works to correctly position appropriate invisible control characters. Further, such a solution is fragile since text may be edited or copied after the placement of marks, potentially leading to further problems with the display.

The problem of mismatched parentheses is very common, and end users routinely encounter difficulties. Rarely are users sufficiently informed about the UBA to solve the display problems themselves. On the contrary, users may attempt to fix problems with visual ordering by changing the logical structure of their text in order to achieve the desired output. For example, in place of “a(b)” a user may type “(a(b” in order to achieve the desired display form in a RTL paragraph. Even for professional developers and localizers, the problems are time consuming on account of being common, and not always trivial to solve. For example, problems with mismatched parentheses accounted for almost 13% of the bidirectional localization bugs addressed in Windows 7. Use of the parenthesis algorithm ensures both logical correctness and display fidelity for the text run in either RTL or LRT embedding direction.

Since 2007 Microsoft has shipped a version of the parenthesis algorithm in its Office products. This algorithm has been refined in the upcoming release of Windows 8. By implementing this parenthesis algorithm to display bidirectional text as a supplement to the UBA, users do not need to resort to control characters to fix problems with the display of parentheses. This document provides details on

Microsoft's solution to the problem in its own terms, and in terms of the core rules of the UBA, and in terms of a higher level protocol. Our position is that the parenthesis algorithm described here, fixes a basic flaw in the current UBA, provides value to users, developers, and localizers, and does not cause regressions in existing documents that are themselves well-formed according to the UBA.

Therefore, Microsoft feels that it is appropriate to address these problems at the OS level – i.e., in the implementation of the UBA, in order to benefit users, developers, and localizers. Given that there are multiple ways a parenthesis algorithm could be implemented, we feel it is important to develop a consensus on the solution and have this adopted by the UTC. This should be either a formal amendment to UAX #9 to include the proposed rule N0, or the creation of an annex that endorses the particular use of higher level protocols described here.

Recap of the relevant part of the UBA

Parentheses and other impacted paired signs have the Bidi category ON (other neutral), the resolution of which is treated by rules N1 and N2 in the UBA:

*N1. A sequence of neutrals takes the direction of the surrounding strong text if the text on both sides has the same direction. European and Arabic numbers act as if they were R in terms of their influence on neutrals. Start-of-level-run (**sor**) and end-of-level-run (**eor**) are used at level run boundaries.*

```
 L  N   L  →   L  L   L
 R  N   R  →   R  R   R
 R  N  AN  →   R  R  AN
 R  N  EN  →   R  R  EN
AN  N   R  →  AN  R   R
AN  N  AN  →  AN  R  AN
AN  N  EN  →  AN  R  EN
EN  N   R  →  EN  R   R
EN  N  AN  →  EN  R  AN
EN  N  EN  →  EN  R  EN
```

N2. Any remaining neutrals take the embedding direction.

```
N → e
```

The problem arises when the two paired signs are resolved differently by the above rules. For example, in "a(b)", the opening parenthesis is resolved to L under N1, whereas the final one is resolved to R under N2:

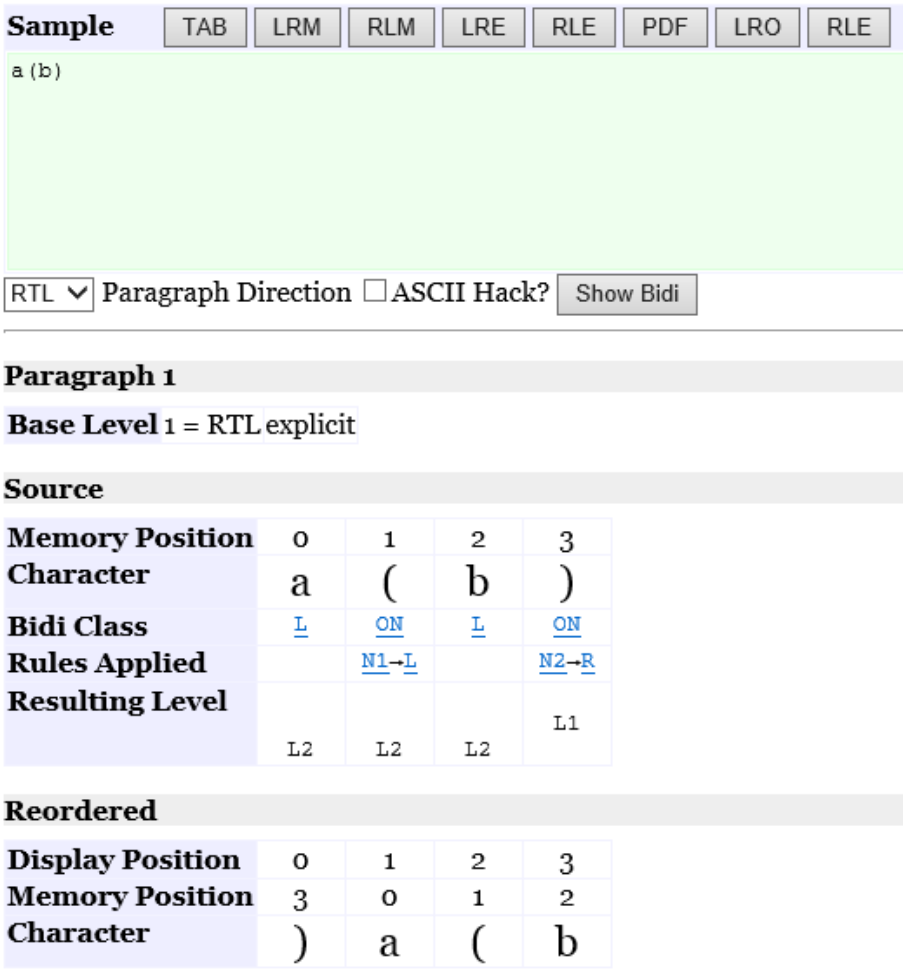

Figure 1. Unicode bidi utility showing output of a(b) in a RTL paragraph

Note that the Unicode bidi utility does not do glyph mirroring. The final output would be (a(b as shown above.

Because there are two possible resolutions under N1, but only one for N2 the possible sequences that give rise to mismatched parentheses are:

A) *N1 and N1:* …O(O…E)E… -OR- …E(E…O)O…
B) *N1 and N2:* …O(O…O)E… -OR- …E(O…O)O…

Where:
O = one or more strong types opposite to the embedding direction
E = one or more strong types of the embedding direction, or start/end of run

Extended complexity within the enclosed text can be ignored since only the neighbors to a paired sign will influence their resolution.

Any other neutral types adjacent to parentheses in a run may be ignored since their resolution is also determined by N1 and N2, and is therefore equal to the individual resolution of the paired punctuation marks. For example, neutrals (N) in a sequence O N (N O N) E will be resolved in the same way as example B above.

Design

Goal

The goal of the parenthesis algorithm is to ensure that paired punctuation marks such as parentheses are always treated as a pair when applying the UBA so that they position and orient correctly, and that the enclosed text never repositions outside of the paired punctuation marks. The resolution of the pair is intended to provide the most intuitive layout for the context.

Identification of paired punctuation marks

Paired punctuation marks are pairs of characters with the first character having general category Open_Punctuation (gc = Ps) and the second character having general category Close_Punctuation (gc = Pe), that are both listed as a Bidi mirrored pair (Bidi_M = Y).

Because the Bidi mirrored characters form a proper subset of the Bidi neutrals (bc = ON), all paired punctuation marks are also Bidi neutral. This definition ensures the inclusion of parenthesis-like marks and the exclusion of quotation marks and presentation forms (e.g., ﴿﴾). It also ensures that the marks of every pair are mirrored characters of each other. As of Unicode 6.1, the set of paired punctuation marks consists of 58 pairs of characters: 55 pairs of script Common, 1 of script Ogham, and 2 of script Tibetan.

Nesting

Paired punctuation marks must be correctly nested in order for the algorithm to run. Incorrectly nested, unbalanced, or mismatched pairs may cause inconsistency in the rules governing the resolution of the paired punctuation marks. Therefore, the parenthesis algorithm should not be applied in these cases. Standard resolution using N1 and N2 should proceed as normal.

Correct nesting requires the paired punctuation marks to be mirror characters of each other, to be at the same embedding level, and to have a lower or equal embedding level as the content they contain.

These constraints only apply for the current paragraph.

Resolution of paired punctuation marks

The next task is to determine whether the paired punctuation marks should be made to match the adjacent context or the paragraph direction. At the point in the UBA where neutrals are resolved, the types present in a run will be strong R, strong L or neutral. For the purposes of assessing which direction to resolve paired signs, the following possibilities exist:

LTR	RTL

L(L)L → L	L(ON)L → L	L(L)L → L	L(ON)L → L
L(L)R → L	L(ON)R → L	L(L)R → L	L(ON)R → R
R(L)L → L	R(ON)L → L	R(L)L → L	R(ON)L → R
R(L)R → L	R(ON)R → R	R(L)R → R	R(ON)R → R
L(R)L → L		L(R)L → R	
L(R)R → R	L(LR)L → L	L(R)R → R	L(LR)L → R
R(R)L → R	L(LR)R → L	R(R)L → R	L(LR)R → R
R(R)R → R	R(LR)R → L	R(R)R → R	R(LR)R → R

Sequences with an ON type outside the parenthesis or mixed with a strong type inside can be ignored since they will be equivalent to one of the above possibilities after resolution using N1 or N2. Similarly, sequences with mixed type content RL, RLR, LRL, etc. are functionally equivalent to the above types with enclosed LR.

Once the paired punctuation marks have been identified, they should be resolved to the embedding direction except in the following cases:

- The directionality of the enclosed content is opposite the embedding direction, and at least one neighbor has a bidi level opposite to the embedding direction O(O)E or O(O)O
- The enclosed content is neutral and both neighbors have a bidi level opposite to the embedding direction O(N)O. This is current behavior in the UBA

The rationale for following the embedding level in the normal case is that the text segment enclosed by the paired punctuation signs will conform to the progression of other text segments in the writing direction. In the exception cases, the rationale to follow the opposite direction is based on context being established between the enclosed and adjacent segments with the same direction.

For example (assuming RTL paragraph level):

R(L)R	WERBEH (a) CIBARA
L(L)R	book(s) CIBARA
L(ON)L	WERBEH hobby(-)horse CIBARA
L(LR)R	WERBEH (CIBARA fabrikam) j. smith

Bidirectional controls

LEFT-TO-RIGHT OVERRIDE (U+202D) and RIGHT-TO-LEFT OVERRIDE (U+202E)

Text containing an explicit directional override (LRO or RLO and PDF) around a sequence that includes paired punctuation marks is not affected by the Bidirectional parenthesis algorithm. This is because the directionality of the content enclosed by the override is already determined to be strong L or strong R (as appropriate) and no neutral ambiguity remains to be resolved.

LEFT-TO-RIGHT EMBEDDING (U+202A) and RIGHT-TO-LEFT EMBEDDING (U+202B)

A span of text that includes explicit directional embedding controls (LRE or RLE and PDF) influences the Bidirectional parenthesis algorithm by updating the embedding direction.

LEFT-TO-RIGHT MARK (U+200E) and RIGHT-TO-LEFT MARK (U+200F)

Explicit directional marks (LRM or RLM) influence the directionality of adjacent neutrals as normal under the UBA; that is they behave like any other strong L or strong R. No special handling is needed in the Bidirectional parenthesis algorithm. This same is true for ARABIC LETTER MARK (U+061C), which has been accepted for encoding in a future version of the standard.

Solution analogy in terms of the core UBA

In order to provide the most consistent, and principled solution, the appropriate place to evaluate the paired signs is before the application of N1, as opposed to between the two N rules or making adjustments after N2. As such, the solution may be phrased in terms of a new rule N0:

**N0. Paired punctuation marks take the embedding direction if the enclosed text contains a strong type of the same direction. Else, if the enclosed text contains a strong type of the opposite direction and at least one external neighbor also has that direction the paired punctuation marks take the direction opposite the embedding direction.*

The rule is applied to those paired punctuation marks that are also correctly nested and occur at the same level without an intervening drop below their level.

For example (assuming RTL paragraph level):

R(L)R								WERBEH (a) CIBARA
	Logical sequence			0	1	2	3	4
	Text run			ARABIC	(	a	)	HEBREW
	Bidi Class			R	ON	L	ON	R
	Rules Applied				N0->R		N0->R	
	Resulting Level			L1	L1	L2	L1	L1
L(L)R								book(s) CIBARA
	Logical sequence			0	1	2	3	4
	Text run			ARABIC	book	(	s	)
	Bidi Class			R	L	ON	L	ON
	Rules Applied					N0->L		N0->L
	Resulting Level			L1	L2	L2	L2	L2
L(ON)L								WERBEH hobby(-)horse CIBARA
	Logical sequence	0	1	2	3	4	5	6
	Text run	ARABIC	hobby	(	-	)	horse	HEBREW
	Bidi Class	R	L	ON	ON	ON	L	R
	Rules Applied			N1->L	N1->L	N1->L		
	Resulting Level	L1	L2	L2	L2	L2	L2	L1
L(LR)R								WERBEH (CIBARA fabrikam) j. smith
	Logical sequence	0	1	2	3	4	5	6
	Text run	j. smith	(	fabrikam	Space	ARABIC	)	HEBREW
	Bidi Class	L	ON	L	WS	R	ON	R
	Rules Applied		N0->R		N2->R		N0->R	
	Resulting Level	L2	L1	L2	L1	L1	L1	L1

However, given that Bidi mirroring is not a core property, this definition and rule may not be suitable for direct inclusion in the UBA.

Solution using rules for higher-level protocols

Additional rules for Higher-Level Protocols may be applied to structured text:

> The following clauses are the only permissible ways for systems to apply higher-level protocols to the ordering of bidirectional text. Some of the clauses apply to segments of structured text. This refers to the situation where text is interpreted as being structured, whether with explicit markup such as XML or HTML, or internally structured such as in a word processor or spreadsheet. In such a case, a segment is [a] span of text that is distinguished in some way by the structure.

In this case, the structure is defined by properly nested paired punctuation marks. The relevant rules are:

> HL4. Apply the Bidirectional Algorithm to segments
> The Bidirectional Algorithm can be applied independently to one or more segments of structured text. For example, when displaying a document consisting of textual data and visible markup in an editor, a higher-level process can handle syntactic elements in the markup separately from the textual data.
>
> HL5. Provide artificial context.
> Text can be processed by the Bidirectional Algorithm as if it were preceded by a character of a given type and/or followed by a character of a given type. This allows a piece of text that is extracted from a longer sequence of text to behave as it did in the larger context.

The structure of a text is defined internally by segments of text defined by the use of paired punctuation marks. The enclosed segments are to be treated separately from the surrounding text. Under HL4 the UBA is applied separately to segments either outside or inside a pair of mirroring punctuation marks. Under HL5, RLM or LRM may be appended to the boundary of an enclosing segment in order to invoke the correct directional attribute for the paired punctuation marks.

For example (assuming RTL paragraph level):

R(L)R						WERBEH (a) CIBARA
	Logical sequence	0	1	2	3	4
	Text run	ARABIC	(	a	)	HEBREW
	Bidi Class	R	ON	L	ON	R
	Segment (HL4)	0	0	1	2	2
	Artificial context (HL5)					
	Resulting Level	L1	L1	L2	L1	L1
L(L)R						book(s) CIBARA
	Logical sequence	0	1	2	3	4
	Text run	ARABIC	book	(	s	)
	Bidi Class	R	L	ON	L	ON
	Segment (HL4)	0	0	1	1	1
	Artificial context (HL5)			(-> LRM(		) -

							>)LRM
	Resulting Level		L1	L2	L2	L2	L2
L(ON)L	WERBEH hobby(-)horse CIBARA						
	Logical sequence	0	1	2	3	4	5	6
	Text run	ARABIC	hobby	(	-	)	horse	HEBREW
	Bidi Class	R	L	ON	ON	ON	L	R
	Segment (HL4)	0	0	1	1	1	2	2
	Artificial context (HL5)			(-> LRM(		) ->)LRM		
	Resulting Level	L1	L2	L2	L2	L2	L2	L1
L(LR)R	WERBEH (CIBARA fabrikam) j. smith							
	Logical sequence	0	1	2	3	4	5	6
	Text run	j. smith	(	fabrikam	Space	ARABIC	)	HEBREW
	Bidi Class	L	ON	L	WS	R	ON	R
	Segment (HL4)	0	0	1	1	1	2	2
	Artificial context (HL5)							
	Resulting Level	L2	L1	L2	L3	L3	L1	L1

Examples

Paragraph direction	Text	Examples	
LTR	From: السـيد محمد (شـركة الإدراك)	Broken	From: (محمد السيد (شركة الإدراك
		Fixed	From: (محمد السيد (شركة الإدراك
LTR	מלא צבע [24bpp]	Broken	24] מלא צבעbpp]
		Fixed	צבע מלא [24bpp]
LTR	السـيد محمد (شـركة موزعـين الإدراك Microsoft Corp))	Broken	Microsoft Corp)) محمد السيد (شركة الإدراك(موزعين
		Fixed	((Microsoft Corp موزعين(شركة الإدراك) محمد السيد
RTL	a(b)	Broken	(a(b
		Fixed	a(b)
RTL	إعداد Office 15 (Technical Preview)	Broken	(Office 15 (Technical Preview إعداد
		Fixed	Office 15 (Technical Preview) إعداد
RTL	WWW (World Wide Web) מערכת	Broken	מערכת (WWW (World Wide Web
		Fixed	מערכת WWW (World Wide Web)
RTL	Text Files (*.txt)	Broken	(Text Files (*.txt
		Fixed	Text Files (*.txt)