PRI #231:
Bidirectional parenthesis algorithm

1. Introduction

In its current form the UBA (Unicode Bidirectional Algorithm UAX #9) displays instances of parentheses in cases where the boundaries of the parentheses have mixed directionality in a way that will very often not provide the result users expect. 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 desired 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. Furthermore, such a solution is fragile since text may be edited or copied after the placement of the control characters, 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. By altering the logical structure, such workarounds can lead to different problems in subsequent text processing. Even for professional developers and localizers, the problems are time consuming on account of being common, and not always trivial to solve.

A different approach to the handling of parentheses would be to enhance or extend UBA to add logic for handling of paired punctuation marks. Use of a parenthesis algorithm could ensure both logical correctness and display fidelity for common text scenarios without resorting to use of control characters or other workarounds. Such an algorithm might correctly handle runs in either RTL or LRT embedding directions and as such, remain consistent with the dynamic nature of the UBA. Since there are multiple ways such a parenthesis algorithm could be implemented, it would be important to have implementations standardize on a single solution to ensure stability for data interchange.

This document provides details for a solution to the problem—a "bidi parenthesis algorithm" (BPA)—that may be implemented either as a formal amendment to UAX #9 to include the proposed rule N0 (§ 4.1) or by the endorsement of the particular use of higher level protocols described below (§ 4.2).

It is the view of the proposers that UBA can be seen as a heuristic for determining what bidi-layout behavior is most likely desired for any given string, and that the proposed change would comprise an enhancement to that heuristic that would provide significant benefits to implementations and users generally. The BPA is not a panacea: there will always be exceptional scenarios that require use of control characters and specific familiarity with UBA. The expectation, however, is that a BPA-enhanced UBA could eliminate that need in an entire class of common scenarios.

2. 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:

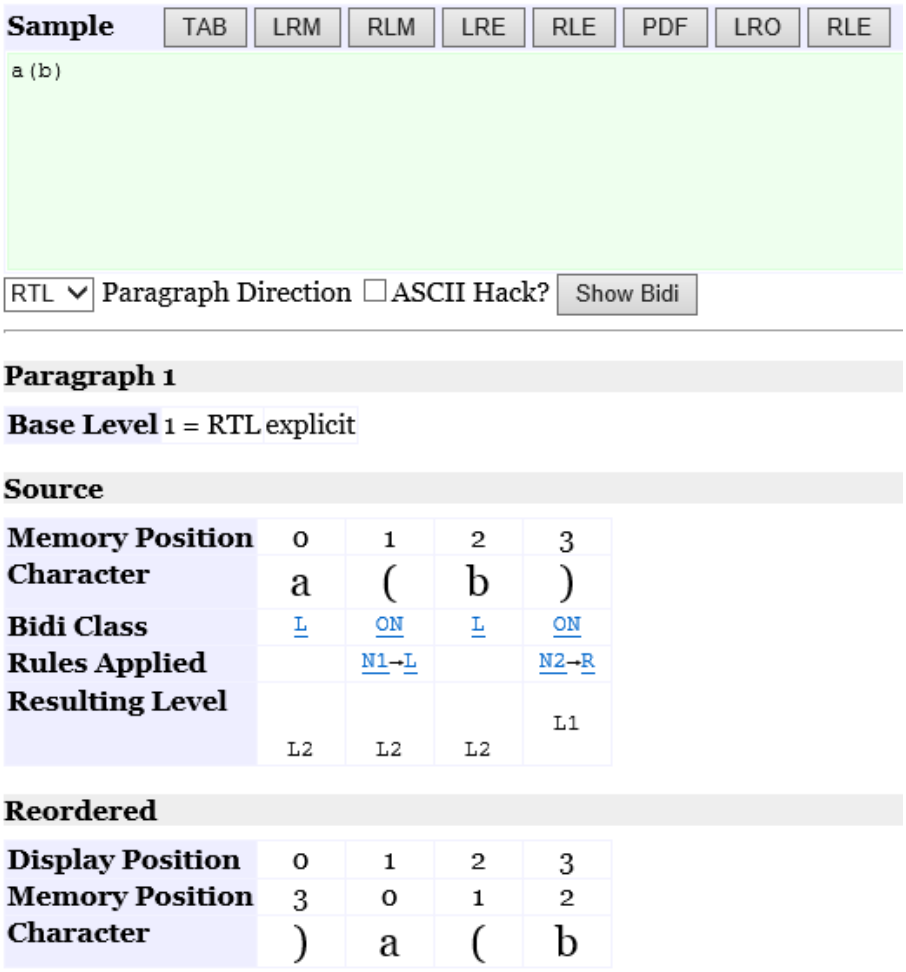

Figure 1. Unicode bidi utility showing output of a(b) in a RTL paragraph

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 non-neutral neighbors to a paired punctuation mark will influence their resolution. Any other neutral types adjacent to parentheses in a run may be ignored when evaluating the level of the paired punctuation marks 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 ON(NON)E will be resolved in the same way as example B above.

3. Design

3.1. 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 content inside and outside the enclosed span does not cross the boundaries of the span. The resolution of the pair is intended to provide the most intuitive layout for the context.

3.2. Identification of paired punctuation marks

Paired punctuation marks are pairs of characters A and B, where A has general category Open_Punctuation (gc = Ps), B has general category Close_Punctuation (gc = Pe), and A and B form a mirrored pair (Bidi_Mirrored = Yes for both, and Bidi_Mirroring_Glyph of A is B). See appendix for a complete listing.

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.

3.3. Finding paired punctuation marks

Scan a paragraph from beginning to end looking for characters that meet the definition of paired punctuation marks as defined above (§ 3.2).

Examples of such Open_Punctuation and Close_Punctuation characters are the opening parenthesis (U+0028) and closing parenthesis (U+0029), respectively. For simplicity, the following discussion will use the term *open parenthesis* for the first class of characters and *close parenthesis* for the second.

If an open parenthesis is found, push it onto a stack and continue the scan. If a close parenthesis is found, check if the stack is not empty and the close parenthesis is the other member of the mirrored pair for the character on the top of the stack. If so, pop the stack and continue the scan; else return failure. If the end of the paragraph is reached, return success if the stack is empty; else return failure. Success implies that all open and close parentheses, if any, in a paragraph are matched correctly. Failure implies that there are one or more mismatched paired punctuation marks in a run and therefore the handling under the parenthesis algorithm will not be attempted.

3.4. 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 BPA 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. A drop in the level of the content below the level of either paired punctuation mark would constitute an error in nesting, and therefore, the BPA should be abandoned.

These constraints only apply for the current paragraph.

3.5. 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. For the purposes of assessing which direction to resolve paired marks, the following possibilities exist:

LTR		RTL	
L(L)L → L	L(N)L → L	L(L)L → L	L(N)L → L
L(L)R → L	L(N)R → L	**L(L)R → L**	L(N)R → R
R(L)L → L	R(N)L → L	**R(L)L → L**	R(N)L → R
R(L)R → L	R(N)R → R	R(L)R → R	R(N)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

The highlighted cases are ones which are currently failing under the existing UBA, and are fixed by the BPA.

Sequences with a neutral 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., enclosed within the paired punctuation signs 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 which are resolved, based on context, opposite the embedding direction:

- 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, E(O)O, or O(O)O.
- The enclosed content is neutral and both neighbors have a bidi level opposite to the embedding direction O(N)O. Resolving to opposite to the embedding direction is current behavior under the UBA (N1).

The rationale for following the embedding level in the normal case is that the text segment enclosed by the paired punctuation marks 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.
Other neutral types adjacent to paired punctuation marks are resolved subsequent to resolving the paired punctuation marks themselves, and will therefore be influenced by that resolution.

3.5.1. Examples

Based on an RTL paragraph:

1.	R(L)R	WERBEH (a) CIBARA
2.	R(L)L	book(s) CIBARA
3.	L(N)L	WERBEH hobby(-)horse CIBARA
4.	L(LR)R	WERBEH (CIBARA fabrikam) j. smith

Note that examples 1 and 3 resolve correctly under the current UBA, whereas examples 2 and 4 require the BPA to display correctly.

Based on an LTR paragraph:

5.	L(LR)R	j. smith (fabrikam CIBARA) WERBEH

3.5.2. A special case

The rules for the resolution of paired punctuation marks have been designed to provide the same output as rules N1 and N2 under the existing UBA except in cases that lead to mismatched parentheses. There is, however, a special case that will produce a different resolution for the level of balanced parentheses under the BPA than under the current UBA. This case is:

O(OEO)O

Under the current UBA, both parentheses in this example are resolved to O under N1. However, under the BPA, the mixed directionality of the enclosure will cause the parentheses to be resolved to E:

LTR	UBA & BPA	alpha(bravo CIBARA charlie)delta
RTL	UBA	charlie)delta CIBARA alpha(bravo
RTL	BPA	delta(charlie CIBARA bravo)alpha

Note that in this case, under the current UBA, the text that is logically outside the parentheses is placed between them in RTL context, and the enclosed text has moved outside. Also, the parentheses themselves face the wrong direction. Under the BPA the segments follow the writing direction, so that the enclosed text remains inside the parentheses and the parentheses face each other.

3.6. Bidirectional controls

3.6.1. 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 BPA. 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. Thus, no special handling is needed in the BPA.

3.6.2. 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 BPA by updating the embedding direction. The effect is comparable to that of changing the base paragraph direction. No special handling is needed in the BPA.

3.6.3. 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 BPA. This same is true for ARABIC LETTER MARK (U+061C), which has been accepted for encoding in a future version of the standard.

4. Solutions

There are two alternatives for implementing the BPA: either a new rule should be introduced into the core algorithm immediately before rule N1, or the higher level protocol rules HL4 and HL5 should be used in conjunction with logic to segment text based on the occurrence of paired punctuation marks and insert appropriate directional marks (LRM, RLM) to achieve the desired result. Both solutions use the logic described above (§ 3.2) to identify paired punctuation marks that are properly nested.

4.1. Solution by updating the core UBA

Given that the use of paired punctuation marks such as parentheses is a normal document scenario, we feel that the resolution of paired punctuation marks should be addressed in the core algorithm. The appropriate place to evaluate the paired signs is before the resolution of neutral types, that is, before the application of N1. The solution may be phrased in terms of a new rule N0. In this way, neutral types that are adjacent to paired punctuation marks resolved by N0 may be impacted by the outcome of that resolution. See detailed examples in section 4.1.2.

4.1.1. Proposed rule

*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.

This rule also requires the definition of paired punctuation marks state previously, and an additional qualification regarding the levels:

Paired punctuation marks are pairs of characters A and B, where A has general category Open_Punctuation (gc = Ps), B has general category Close_Punctuation (gc = Pe), and A and B form a mirrored pair (Bidi_Mirrored = Yes for both, and Bidi_Mirroring_Glyph of A is B).

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

4.1.2. Detailed examples

1. RTL									
R(L)R	WERBEH (a) CIBARA								
	Logical sequence	0	1	2	3	4	5	6	
	Text run	ARABIC	Space	(	a	)	Space	HEBREW	
	Bidi Class	R	WS	ON	L	ON	WS	R	
	Rules Applied		N1->R	N0->R		N0->R	N1->R		
	Resulting Level	L1	L1	L1	L2	L1	L1	L1	

2. RTL							
R(L)L	book(s) CIBARA						
	Logical sequence	0	1	2	3	4	5
	Text run	ARABIC	Space	book	(	s	)
	Bidi Class	R	WS	L	ON	L	ON
	Rules Applied		N2->R		N0->L		N0->L
	Resulting Level	L1	L1	L2	L2	L2	L2

3. RTL										
L(N)L	WERBEH hobby(-)horse CIBARA									
	Logical sequence	0	1	2	3	4	5	6	7	8
	Text run	ARABIC	Space	hobby	(	-	)	horse	Space	HEBREW
	Bidi Class	R	WS	L	ON	ON	ON	L	WS	R
	Rules Applied		N2->R		N1->L	N1->L	N1->L		N2->R	
	Resulting Level	L1	L1	L2	L2	L2	L2	L2	L1	L1

4. RTL										
	WERBEH (CIBARA fabrikam) j. smith									
	Logical	0	1	2	3	4	5	6	7	8

L(LR)R	sequence									
	Text run	j. smith	Space	(	fabrikam	Space	ARABIC	)	Space	HEBREW
	Bidi Class	L	WS	ON	L	WS	R	ON	WS	R
	Rules Applied		N2->R	N0->R		N2->R		N0->R	N1->R	
	Resulting Level	L2	L2	L1	L2	L1	L1	L1	L1	L1
5. LTR	j. smith (fabrikam CIBARA) WERBEH									
L(LR)R	Logical sequence	0	1	2	3	4	5	6	7	8
	Text run	j. smith	Space	(	fabrikam	Space	ARABIC	)	Space	HEBREW
	Bidi Class	L	WS	ON	L	WS	R	ON	WS	R
	Rules Applied		N1->L	N0->L		N2->L		N0->L	N2->R	
	Resulting Level	L0	L0	L0	L0	L0	L1	L0	L1	L1

See also additional examples later in this document.

4.2. Solution using rules for higher-level protocols

This approach may be used as a conformant solution under the current UBA since UAX #9 includes additional rules for Higher-Level Protocols that 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 order to ensure consistent implementation the directional control marks LRM and RLM should be applied to paired punctuation marks according to logic described in this section. In this way, neutral types that are adjacent to paired punctuation marks may resolve differently than they would have due to the insertion of LRM or RLM, see detailed examples in section 4.2.2.

4.2.1. Current rules

Properly nested paired punctuation marks may be used to identify segments of text to which the UBA may be applied. The appropriate rule is HL4:

> 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.

The segments are to be identified using the logic described above (§ 3.3).

Where necessary, directional control marks (RLM/LRM) should be inserted at the borders of segments in order to provide correct resolution using the UBA. The appropriate rule is HL5:

> 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 determination of whether directional control marks should be inserted is based on the logic described above (§ 3.5). Once the appropriate marks have been inserted, segment can be processed using the UBA.

4.2.2. Detailed examples

1. RTL										WERBEH (a) CIBARA
R(L)R	Logical sequence			0	1	2	3	4	5	6
	Text run			ARABIC	Space	(	a	)	Space	HEBREW
	Bidi Class			R	WS	ON	L	ON	WS	R
	Segment (HL4)			0	0	1	1	1	2	2
	Artificial context (HL5)					RLM(		)RLM		
	Rules applied				N1->R	N2->R		N2->R	N1->R	
	Resulting Level			L1	L1	L1	L2	L1	L1	L1
2. RTL										book(s) CIBARA
R(L)L	Logical sequence				0	1	2	3	4	5
	Text run				ARABIC	Space	book	(	S	)
	Bidi Class				R	WS	L	ON	L	ON
	Segment (HL4)				0	0	0	1	1	1
	Artificial context (HL5)							LRM(		)LRM
	Rules applied					N2->R		N1->L		N1->L
	Resulting Level				L1	L1	L2	L2	L2	L2
3. RTL										WERBEH hobby(-)horse CIBARA
L(N)L	Logical sequence	0	1	2	3	4	5	6	7	8
	Text run	ARABIC	Space	hobby	(	-	)	horse	Space	HEBREW
	Bidi Class	R	WS	L	ON	ON	ON	L	WS	R
	Segment (HL4)	0	0	0	1	1	1	2	2	2
	Artificial context (HL5)				LRM(		)LRM			
	Rules applied		N2->R		N1->L		N1->L		N2->R	
	Resulting Level	L1	L1	L2	L2	L2	L2	L2	L1	L1
4. RTL										WERBEH (CIBARA fabrikam) j. smith
L(LR)R	Logical sequence	0	1	2	3	4	5	6	7	8
	Text run	j. smith	Space	(	fabrikam	Space	ARABIC	)	Space	HEBREW
	Bidi Class	L	WS	ON	L	WS	R	ON	WS	R
	Segment (HL4)	0	0	1	1	1	1	1	2	2
	Artificial context (HL5)			RLM(				)RLM		
	Rules applied		N2->R	N2->R		N2->R		N1->R	N1->R	
	Resulting Level	L2	L1	L1	L2	L1	L1	L1	L1	L1
5. LTR	j. smith (fabrikam CIBARA) WERBEH									
L(LR)R	Logical sequence	0	1	2	3	4	5	6	7	8
	Text run	j. smith	Space	(	fabrikam	Space	ARABIC	)	Space	HEBREW
	Bidi Class	L	WS	ON	L	WS	R	ON	WS	R
	Segment (HL4)	0	0	1	1	1	1	1	2	2
	Artificial context (HL5)			LRM(				)LRM		
	Rules applied		N1->L	N1->L		N2->L		N2->L	N2->L	

	Resulting Level	L0	L0	L0	L0	L0	L1	L0	L0	L1

See also additional examples given below.

5. Examples with and without the BPA

Paragraph direction	Text	Output	
1. RTL	Text Files (*.txt)	Without BPA	(Text Files (*.txt
		With BPA	Text Files (*.txt)
2. RTL	WWW (World Wide Web) מערכת	Without BPA	מערכת (WWW (World Wide Web
		With BPA	מערכת WWW (World Wide Web)
3. RTL	Office 15 إعداد (Technical Preview)	Without BPA	(Office 15 (Technical Preview إعداد
		With BPA	Office 15 (Technical Preview) إعداد
4. RTL	j. smith (fabrikam עברית (العربية	Without BPA	עברית (العربية j. smith (fabrikam
		With BPA	עברית (العربية fabrikam) j. smith
5. LTR	j. smith (fabrikam עברית (العربية	Without BPA	j. smith (fabrikam עברית (العربية
		With BPA	j. smith (fabrikam العربية) עברית
6. LTR	شركة) السيد محمد موزعين)الإدراك Microsoft Corp))	Without BPA	Microsoft Corp)) محمد السيد (شركة الإدراك(موزعين
		With BPA	((Microsoft Corp محمد السيد (شركة الإدراك(موزعين
7. LTR	מלא צבע [24bpp]	Without BPA	24] מלא צבעbpp]
		With BPA	צבע מלא [24bpp]
8. LTR	From: السيد محمد (الإدراك شركة)	Without BPA	From: (محمد السيد (شركة الإدراك
		With BPA	From: محمد السيد (شركة الإدراك)

6. Stability

Because the BPA proposed here involves a heuristic which determines the level of paired punctuation marks based on the content of the text itself and does not alter the text in any way, well-formed new or existing text will display with desired results under the BPA. This is true whether or not the text contains directional control marks. It is important to stress that current text which has used directional controls in order to obtain correct display will continue to display without change under the BPA. The main stability concern therefore is that text authored using the BPA may display differently when rendered on

a system which has not implemented the BPA. In such a case, the reader of that text is no worse off than they would have been prior to the development of the BPA.

Another stability concern relates to the possibility of there being text which is deliberately contrived to work around the problem of mismatched paired punctuation marks under the current UBA. An example would be a logical pair of nested parentheses which render as a sequence of non-nested paired punctuation marks under the UBA, i.e., logical E(O(OEO)O)E renders as E(O)OEO(O)E under the current UBA, whereas the BPA preserves the logical form. The benefits of the BPA are expected to far outweigh the loss in stability of such sequences.

7. Alternative solutions considered and rejected

7.1. Inserting marks

One suggestion to address this problem is to have edit controls insert the appropriate directional controls automatically. A serious drawback to this suggestion is that the correct display of text with paired punctuation marks would depend on the source application supporting this behavior. This also requires these controls to have an awareness of the UBA in order to insert the correct marks when they may currently be relying on the OS to manage the display of bidirectional text. Given the number of different edit controls, the surface area for this approach is too great to be viable. Moreover, different control implementations might vary significantly in the implementation of this solution, and hence not achieve any overall gain in avoiding user confusion.

Having a tool that inserts the correct marks according to the proposed algorithm might be a useful tool to facilitate cross platform stability during the transition period to widespread adoption of the BPA. However, insertion of marks is not a stable or complete solution to the problem because text that has had marks inserted may be copied and edited in contexts beyond the one in which the marks were applied, and thus, rather than correcting problems, the presence of invisible directional control marks may introduce problems. For example, when the text for example 4 above (§§ 4.2.2 and 5) is updated to include the RLM marks according to the procedure in section 4.2, the text renders correctly in an RTL paragraph:

עברית (العربية fabrikam) j. smith

However, when this text, including the marks, is put in an LRT context the text is distorted:

j. smith (fabrikam עברית (العربية

Only dynamic resolution of the parenthesis under the BPA is able to adapt correctly to changes in context required for resolution to the embedding direction.

8. Appendix – List of paired punctuation marks

U+0028	U+0029	(	)	LEFT PARENTHESIS	RIGHT PARENTHESIS
U+005B	U+005D	[	]	LEFT SQUARE BRACKET	RIGHT SQUARE BRACKET
U+007B	U+007D	{	}	LEFT CURLY BRACKET	RIGHT CURLY BRACKET
U+0F3A	U+0F3B	༺	༻	TIBETAN MARK GUG RTAGS GYON	TIBETAN MARK GUG RTAGS GYAS
U+0F3C	U+0F3D	༼	༽	TIBETAN MARK ANG KHANG GYON	TIBETAN MARK ANG KHANG GYAS
U+169B	U+169C	᚛	᚜	OGHAM FEATHER MARK	OGHAM REVERSED FEATHER MARK
U+2045	U+2046	⁅	⁆	LEFT SQUARE BRACKET WITH QUILL	RIGHT SQUARE BRACKET WITH QUILL
U+207D	U+207E	⁽	⁾	SUPERSCRIPT LEFT PARENTHESIS	SUPERSCRIPT RIGHT PARENTHESIS
U+208D	U+208E	₍	₎	SUBSCRIPT LEFT PARENTHESIS	SUBSCRIPT RIGHT PARENTHESIS
U+2329	U+232A	〈	〉	LEFT-POINTING ANGLE BRACKET	RIGHT-POINTING ANGLE BRACKET
U+2768	U+2769	❨	❩	MEDIUM LEFT PARENTHESIS ORNAMENT	MEDIUM RIGHT PARENTHESIS ORNAMENT
U+276A	U+276B	❪	❫	MEDIUM FLATTENED LEFT PARENTHESIS ORNAMENT	MEDIUM FLATTENED RIGHT PARENTHESIS ORNAMENT
U+276C	U+276D	❬	❭	MEDIUM LEFT-POINTING ANGLE BRACKET ORNAMENT	MEDIUM RIGHT-POINTING ANGLE BRACKET ORNAMENT
U+276E	U+276F	❮	❯	HEAVY LEFT-POINTING ANGLE QUOTATION MARK ORNAMENT	HEAVY RIGHT-POINTING ANGLE QUOTATION MARK ORNAMENT
U+2770	U+2771	❰	❱	HEAVY LEFT-POINTING ANGLE BRACKET ORNAMENT	HEAVY RIGHT-POINTING ANGLE BRACKET ORNAMENT
U+2772	U+2773	❲	❳	LIGHT LEFT TORTOISE SHELL BRACKET ORNAMENT	LIGHT RIGHT TORTOISE SHELL BRACKET ORNAMENT
U+2774	U+2775	❴	❵	MEDIUM LEFT CURLY BRACKET ORNAMENT	MEDIUM RIGHT CURLY BRACKET ORNAMENT
U+27C5	U+27C6	⟅	⟆	LEFT S-SHAPED BAG DELIMITER	RIGHT S-SHAPED BAG DELIMITER
U+27E6	U+27E7	⟦	⟧	MATHEMATICAL LEFT WHITE SQUARE BRACKET	MATHEMATICAL RIGHT WHITE SQUARE BRACKET
U+27E8	U+27E9	⟨	⟩	MATHEMATICAL LEFT ANGLE BRACKET	MATHEMATICAL RIGHT ANGLE BRACKET
U+27EA	U+27EB	⟪	⟫	MATHEMATICAL LEFT DOUBLE ANGLE BRACKET	MATHEMATICAL RIGHT DOUBLE ANGLE BRACKET

U+27EC	U+27ED	⟬	⟭	MATHEMATICAL LEFT WHITE TORTOISE SHELL BRACKET	MATHEMATICAL RIGHT WHITE TORTOISE SHELL BRACKET
U+27EE	U+27EF	⟮	⟯	MATHEMATICAL LEFT FLATTENED PARENTHESIS	MATHEMATICAL RIGHT FLATTENED PARENTHESIS
U+2983	U+2984	⦃	⦄	LEFT WHITE CURLY BRACKET	RIGHT WHITE CURLY BRACKET
U+2985	U+2986	⦅	⦆	LEFT WHITE PARENTHESIS	RIGHT WHITE PARENTHESIS
U+2987	U+2988	⦇	⦈	Z NOTATION LEFT IMAGE BRACKET	Z NOTATION RIGHT IMAGE BRACKET
U+2989	U+298A	⦉	⦊	Z NOTATION LEFT BINDING BRACKET	Z NOTATION RIGHT BINDING BRACKET
U+298B	U+298C	⦋	⦌	LEFT SQUARE BRACKET WITH UNDERBAR	RIGHT SQUARE BRACKET WITH UNDERBAR
U+298D	U+2990	⦍	⦐	LEFT SQUARE BRACKET WITH TICK IN TOP CORNER	RIGHT SQUARE BRACKET WITH TICK IN TOP CORNER
U+298F	U+298E	⦏	⦎	LEFT SQUARE BRACKET WITH TICK IN BOTTOM CORNER	RIGHT SQUARE BRACKET WITH TICK IN BOTTOM CORNER
U+2991	U+2992	⦑	⦒	LEFT ANGLE BRACKET WITH DOT	RIGHT ANGLE BRACKET WITH DOT
U+2993	U+2994	⦓	⦔	LEFT ARC LESS-THAN BRACKET	RIGHT ARC GREATER-THAN BRACKET
U+2995	U+2996	⦕	⦖	DOUBLE LEFT ARC GREATER-THAN BRACKET	DOUBLE RIGHT ARC LESS-THAN BRACKET
U+2997	U+2998	⦗	⦘	LEFT BLACK TORTOISE SHELL BRACKET	RIGHT BLACK TORTOISE SHELL BRACKET
U+29D8	U+29D9	⧘	⧙	LEFT WIGGLY FENCE	RIGHT WIGGLY FENCE
U+29DA	U+29DB	⧚	⧛	LEFT DOUBLE WIGGLY FENCE	RIGHT DOUBLE WIGGLY FENCE
U+29FC	U+29FD	⧼	⧽	LEFT-POINTING CURVED ANGLE BRACKET	RIGHT-POINTING CURVED ANGLE BRACKET
U+2E22	U+2E23	⸢	⸣	TOP LEFT HALF BRACKET	TOP RIGHT HALF BRACKET
U+2E24	U+2E25	⸤	⸥	BOTTOM LEFT HALF BRACKET	BOTTOM RIGHT HALF BRACKET
U+2E26	U+2E27	⸦	⸧	LEFT SIDEWAYS U BRACKET	RIGHT SIDEWAYS U BRACKET
U+2E28	U+2E29	⸨	⸩	LEFT DOUBLE PARENTHESIS	RIGHT DOUBLE PARENTHESIS
U+3008	U+3009	〈	〉	LEFT ANGLE BRACKET	RIGHT ANGLE BRACKET
U+300A	U+300B	《	》	LEFT DOUBLE ANGLE BRACKET	RIGHT DOUBLE ANGLE BRACKET
U+300C	U+300D	「	」	LEFT CORNER BRACKET	RIGHT CORNER BRACKET
U+300E	U+300F	『	』	LEFT WHITE CORNER BRACKET	RIGHT WHITE CORNER BRACKET

U+3010	U+3011	【 】	LEFT BLACK LENTICULAR BRACKET	RIGHT BLACK LENTICULAR BRACKET
U+3014	U+3015	〔 〕	LEFT TORTOISE SHELL BRACKET	RIGHT TORTOISE SHELL BRACKET
U+3016	U+3017	〖 〗	LEFT WHITE LENTICULAR BRACKET	RIGHT WHITE LENTICULAR BRACKET
U+3018	U+3019	〘 〙	LEFT WHITE TORTOISE SHELL BRACKET	RIGHT WHITE TORTOISE SHELL BRACKET
U+301A	U+301B	〚 〛	LEFT WHITE SQUARE BRACKET	RIGHT WHITE SQUARE BRACKET
U+FE59	U+FE5A	﹙ ﹚	SMALL LEFT PARENTHESIS	SMALL RIGHT PARENTHESIS
U+FE5B	U+FE5C	﹛ ﹜	SMALL LEFT CURLY BRACKET	SMALL RIGHT CURLY BRACKET
U+FE5D	U+FE5E	﹝ ﹞	SMALL LEFT TORTOISE SHELL BRACKET	SMALL RIGHT TORTOISE SHELL BRACKET
U+FF08	U+FF09	（ ）	FULLWIDTH LEFT PARENTHESIS	FULLWIDTH RIGHT PARENTHESIS
U+FF3B	U+FF3D	［ ］	FULLWIDTH LEFT SQUARE BRACKET	FULLWIDTH RIGHT SQUARE BRACKET
U+FF5B	U+FF5D	｛ ｝	FULLWIDTH LEFT CURLY BRACKET	FULLWIDTH RIGHT CURLY BRACKET
U+FF5F	U+FF60	⦅ ⦆	FULLWIDTH LEFT WHITE PARENTHESIS	FULLWIDTH RIGHT WHITE PARENTHESIS
U+FF62	U+FF63	｢ ｣	HALFWIDTH LEFT CORNER BRACKET	HALFWIDTH RIGHT CORNER BRACKET