Title: On the Interaction of the Bidirectional Algorithm and Arabic Shaping
Date: 2003-03-01 05:35 IRT (+0330)

Authors: Behdad Esfahbod (`behdad@bamdad.org`) and
Roozbeh Pournader (`roozbeh@sharif.edu`)

Abstract

This document tries to comment on the proposed solutions of document L2/03-064, titled “Unicode BIDI Issue #5”. It establishes some facts and goals, and proves that none of the solutions proposed in the mentioned documents are adequate for the problem. It finally suggests a sixth solution and proves that it achieves the stated goals.

1 Known Facts

Fact 1 It is quite usual to override the direction of a run of some Arabic letters, to embed a piece of visually-ordered text in a logical stream[1].

Fact 2 The output from cases A and D look weird.

Fact 3 The output from cases B and C look less weird, but both appear acceptable to some degree.

Fact 4 Currently, the Bidirectional Algorithm does not specify the position of Boundary Neutral (BN) characters in its output, and applications are even allowed to remove them from the text stream. Zero Width Joiner (ZWJ) and Zero Width Non-Joiner (ZWNJ), are BN characters.

Fact 5 Currently, conformance to the Bidirectional Algorithm does not require the resolved embedding levels of each character to be equal to those computed in the reference algorithm, so using Bidirectional embedding levels in other algorithms should be avoided as much as possible.

Fact 6 The Bidirectional Algorithm can be divided into two phases, namely bidirectional level resolution (rules before L1), and bidirectional reordering (rules L1–L4). These two are seperated by a line breaking algorithm.

2 Goals We Seek

Goal 1 We need a precise, deterministic, non-hueristic, and non-broken Arabic script rendering process. This mainly consists of: a) bidirectional reordering, b) shaping, and c) line breaking.

Goal 2 The shaping control characters, ZWJ and ZWNJ, should behave correctly: they should affect and only affect their neighbors in the logical text stream. (Neighbors are defined to be the first adjacent character that is not transparent with regard to the Arabic shaping algorithm.) In the case this goal cannot be fulfilled, explicit directional formatting codes are allowed to break the connection that defines a neighbor.

[1] HTML 4 also recommends using bidirectional overrides when converting a document from a visual-order character set, which is common in older Persian, Hebrew, and Arabic web pages.

Goal 3 Insertion of an empty pair of matching explicit directional formatting codes, for example ⟨LRO, PDF⟩, in any place in a logical stream, should not affect the way the stream is displayed. Intelligent text editors should be able to add or remove these pairs automatically.

Goal 4 Contextual shaping of bidirectionally-overrided Arabic text should be possible, so that overriding some visually-encoded text (converted from a legacy visual-order character set, for example) would produce a Unicode-conformant text stream representing the original.

Goal 5 Executing any of the three phases of Arabic script rendering (specified in Goal 1) for more than once, should be avoided.

Goal 6 The output of the Arabic rendering process should not look odd, and should be reasonable and acceptable with regard to the underlying mechanisms of the Arabic script.

3 We are Approaching a Proof!

Lemma 1 *Every Arabic script rendering process should execute Arabic shaping at least once before bidirectional reordering (namely the second phase of the Bidirectional Algorithm).*

Proof: As quoted verbatim from UAX#9 “The Bidirectional Algorithm”, the section on “Reordering Resolved Levels”:

- The levels of the text are determined according to the bidirectional algorithm.
- The characters are shaped into glyphs according to their context (*taking the embedding levels into account for mirroring!*).
- The accumulated widths of those glyphs (*in logical order*) are used to determine line breaks.
- For each line, rules L1–L4 are used to reorder the characters on that line.
- The glyphs corresponding to the characters on the line are displayed in that order.

Theorem 1 *None of proposed solutions A, B, C, D, or Z, proposed in document L2/03-064, do fit our goals.*

Proof:

- Solution A breaks Goals 4 and 6.
- Using Lemma 1, solution B breaks Goals 3 and 5. Due to Fact 4, it also breaks Goals 1 and 2.
- Using Lemma 1, solution C breaks Goals 3 and 5. Due to Facts 4 and 5, it also breaks Goals 1 and 2.

- Solution D breaks Goals 4 and 6.
- Solution Z breaks Goals 1, 4 and 6.

Lemma 2 *Assuming the goals stated above, every Arabic **shaping** process should use the results from the UAX#9 rules P1 to X8.*

Proof: Due to the limitation on the maximum valid bidirectional embedding level of the Bidirectional Algorithm (which is about 61), some explicit directional formatting codes may become ineffective (or *invalid*, as per UAX#9 wording). Considering Goal 4, the behaviour of the shaping algorithm is different depending on the effective-ness of an explicit directional formatting code. Thus, the process needs to know which explicit codes are effective. It is UAX#9 rules P1 to X8 that determine this, by definition.

Algorithm X

1. Apply UAX#9 rules P1 to X8 (but not X9 and X10).
2. Assuming the resulting character types from Step 1,
    - 2.1. For each contiguous sequence of characters *not* of type L, shape them according to the original logical order. Assume non-joining boundaries.
    - 2.2. For each contiguous sequence of characters of type L, shape them according to the *reverse* of the original logical order. Assume non-joining boundaries.

Theorem 2 *Algorithm X fits Goals 1 to 6.*

Proof:

- **Goal 1:** This is obvious from the definition of the algorithm.
- **Goal 2:** Algorithm X is shaping characters according to the logical order and the reverse logical order. As both ZWJ and ZWNJ act symmetrically on their previous and next characters, shaping on the reverse logical order does not corrupt their behaviour.
- **Goal 3:** This is achieved by Algorithm X, as explicit codes are ignored in Arabic shaping, specified to be transparent with regard to that.
- **Goal 4:** This can be done. From the fact that all text overrided by LRO...PDF pairs will have characters of type L, and thus will get shaped in the reverse logical order, which produces the desired output. Finally, it is also known that no character with an original bidirectional character type of L is affected or shaped by the Arabic shaping algorithm.
- **Goal 5:** Executing *bidirectional level resolution*, doing Algorithm X after that, then line breaking, and finally bidirectional reordering, is a complete Arabic rendering process, with all the phases running just once. Implementations may share the partial result of phases P1 to X8 of bidirectional level resolution phase of the Bidirectional Algorithm with Algorithm X.

- **Goal 6:** The output is almost the same as the output of proposed solution C (and additionally solving a few other problems) which is the most acceptable among the five solutions proposed in L2/03-064. This part cannot be proven completely, due to the human justice factor. You will be required to trust the authors!

Additional Note: It may also be proved that Algorithm X (also taking into consideration the recommendation in the proof above of the achievement of Goal 5) is the most efficient solution to this problem, in terms of CPU time. We will leave this as an exercise to the reader! (*Hint:* Use Lemma 2.)