L2/11-150R2

Subject: BIDI URL Display
From: M. Davis
Date: 2011-05-05
Live document: http://goo.gl/xuFDh
Status: *Rough draft, for discussion only*

The Unicode Bidirectional Algorithm (UBA) was designed for handling normal text, and predated the rise of the web. Unfortunately, URLs* are not normal text; they are syntactically complex in ways that don't work well with the UBA. That causes URLs to appear jumbled to bidi users (Arabic/Hebrew/...).

> [*Formally speaking, what we are talking about are IRIs, although most end-users know them as "URLs", see idn-and-iri.]

For example, consider the display of the following URL (as per convention, uppercase represents right-to-left characters).

URL (memory): http://a.b.**C.D.**com/**E/F/**g/h

It would be displayed as follows:

Environment	**Display**	**Details**
LTR	http://a.b.**D.C.**com/**F/E/**g/h	http://goo.gl/yvNFq
RTL	g/h**/F/E/**com**.D.C**.http://a.b	http://goo.gl/DuELB

People have been looking for an extension to the Unicode Bidirectional Algorithm (UBA) that handles IRIs in a more consistent way for bidi users. The general goal would be for the "fields" of an IRI to flow in a consistent direction.

This requires consistency in usage across different applications. For example, when someone copies the contents of an address bar into an email, we don't want all the fields in the URL to switch around. Such consistency would require a general extension to the UBA to indicate how IRIs should be displayed, both in the limited context of an address bar, and in other contexts.

The challenges are:

1. There will be a long migration period, so making sure that the negative effects are mitigated as much as possible.
2. For the purpose of the UBA, having a simple, comprehensible way to recognize IRIs in

plaintext.

Contents

Recognizing IRIs in Plaintext

Here are rough thoughts about the second task (recognizing IRIs in plaintext) for use with the UBA.

There are two problems:

1. A formal reading of the spec allows almost anything in fields, so it is hard to test for termination.
2. The URL may not be headed by a scheme in plaintext (eg, google.com should be recognized).

While in theory, almost anything can occur in fields, in practice many Unicode characters never or very rarely occur in these contexts. So one approach is to have a simplified syntax that could be easily recognized. Any characters that were legal, but outside of that syntax would need to be represented with % escapes *if* they are to be handed by the UBA extension.

For the second issue, a list of TLDs would be used (in the appropriate context) if there is no scheme.

Here is the proposed BNF (using Perl-style syntax):

bidi_IRI := ((scheme “://” domain) | domain2) (“/” path)? (“?” query)? (“#” fragment)?

domain := UTS46Chars + (IDNSep UTS46Chars+)* IDNSep?
domain2 := domain IDNSep TLD IDNSep?
path := (char - “?” - “#”)*
query := (char - “#”)*
fragment := char*

IDNSep := [\u002E \uFF0E \u3002\uFF61] // see http://unicode.org/reports/tr46/#Notation

TLD := <list on http://www.iana.org/domains/root/db/>

char := percentEncodedUTF8
| [[:L:][:N:][:M:][:S:][:Pd:][:Pc:][:Cf:] inclusionChar - exclusionChar]

inclusionChar :=
U+0021 (!) EXCLAMATION MARK
U+0022 (") QUOTATION MARK
U+0023 (#) NUMBER SIGN
U+0025 (%) PERCENT SIGN
U+0026 (&) AMPERSAND
U+0027 (') APOSTROPHE
U+002A (*) ASTERISK
U+002C (,) COMMA
U+002E (.) FULL STOP
U+002F (/) SOLIDUS
U+003A (:) COLON
U+003B (;) SEMICOLON
U+003F (?) QUESTION MARK
U+0040 (@) COMMERCIAL AT
U+005C (\ ) REVERSE SOLIDUS
U+00A1 (¡) INVERTED EXCLAMATION MARK
U+00B7 (·) MIDDLE DOT
U+00BF (¿) INVERTED QUESTION MARK

exclusionChar :=
U+003C (<) LESS-THAN SIGN
U+003E (>) GREATER-THAN SIGN

In addition, a final inclusionChar is excluded from the bidi_IRI when parsing (it is possible to capture this with the BNF, but it makes the formulation much less readable).

For the full IRI syntax, see: http://rfc-ref.org/RFC-TEXTS/3987/chapter2.html

- ***TBD****: add userinfo, port, IP, etc.*
- ***TBD:*** *include some statistics on the character usage in URLs based on data collected at Google.*

Termination and Continuation

Thus the bidi IRI basically *terminates* with:

1. Unassigned, surrogates, private-use, control codes
2. Whitespace

3. Open, close or most ‘other’ punctuation, plus special cases < and >.

These characters can be included in an IRI, but they would need to be % encoded for the specialized bidi display to kick in.

The bidi IRI basically *continues* with:

1. Letters, Marks, Numbers
2. Dash and connector punctuation
    - See http://unicode.org/cldr/utility/list-unicodeset.jsp?a=%5Cp{Pc}%5Cp{Pd}-%5Cp{ascii}
3. Symbols
    - See http://unicode.org/cldr/utility/list-unicodeset.jsp?a=%5Cp{s}-%5Cp{ascii}

For ASCII and Latin1, the terminating punctuation* in #3 are:

U+003C (<) LESS-THAN SIGN
U+003E (>) GREATER-THAN SIGN
U+0028 (( ) LEFT PARENTHESIS
U+0029 ( )) RIGHT PARENTHESIS
U+005B ([ ) LEFT SQUARE BRACKET
U+005D ( ]) RIGHT SQUARE BRACKET
U+007B ({) LEFT CURLY BRACKET
U+007D (}) RIGHT CURLY BRACKET
U+00AB («) LEFT-POINTING DOUBLE ANGLE QUOTATION MARK
U+00BB (») RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK

Open issues

- The inclusionChars are from ASCII/Latin1. Do we want to tweak any of the ASCII/Latin1 exceptions to move them to terminating characters?

Applying the UBA Extension

Given the above parsing, any recognized bidi IRI would be ordering according to a (proposed) extension to the UBA. The goal is to get a single, uniform ordering of fields, no matter whether a field has RTL or LTR characters.

If URLs were consistently “big-endian”, it would be useful to have the ordering depend on the direction of the paragraph. However, URL are not consistently big-endian; the most important part (the domain) is backwards. Because the ordering is fairly arbitrary, this proposal is to have a consistent ordering always, no matter what the environment is.

This is just a strawman, and needs more discussion. An alternative is discussed under Open Issues below.

Given a bidi_IRI:

A *separator* is defined as any instance of the quoted strings in the bidi_IRI BNF:

- right after a scheme: “://”
- in a domain: IDNSep
- right after a domain: “/” , “?”, “#”
- in a path: “/”
- right after a path: “?”, “#”
- in a query: “=” or “&”
- right after a query: “#”

A *field* is defined as any text between separators, or at the front or end.

Each bidi_IRI is displayed with fields from left to right. Thus the following will always appear with the same display, whether in a RTL or LTR environment.

Environment	**Display**
LTR	http://a.b.**C.D.**com/**E/F/**g/h
RTL	http://a.b.**C.D.**com/**E/F/**g/h

Note that *within* each field, the UBA still applies. So inside of C, for example, RTL characters are still displayed from right to left.

This can be handled in the UBA extension by behaving as if:

- the entire bidi_IRI is embedded in <LRE>...<PDF>
- each field is surrounded by LRMs.

Open Issues

Alternatively, the display could be subject to the environment (whether the current embedding level is RTL or LTR). In that case, the display would be something like:

Environment	**Display**
LTR	http://a.b.**C.D.**com/**E/F/**g/h
RTL	h/g**/F/E/**com**.D.C**.b.a//:http