The bidi algorithm is required for Unicode conformance, and, at the same time, has been held very stable. This has reduced the amount of divergence among implementations. Most “changes” to the algorithm, recently, have been in the nature of “clarifications” rather than modifications.

Dealing with URLs, IRIs or filenames exposes some issues which ultimately go back to the problem that the original bidi algorithm (“UBA 1.0”) chooses a single interpretation for punctuation (e.g. “/”) which is used ambiguously.

When the Bidi algorithm was designed, it was decided to select among possible choices, because a fixed selection made the algorithm predictable. Explicit overrides were added to force different display.

In my view, it is of the utmost importance to not give up on this feature of the bidi algorithm. This forbids any “tinkering around the edges”.

However, as I outlined in my post for another bidi-related PRI, this leaves certain higher-level protocols without good choices.

I would favor either the creation of a new version of the UBA (UBA 2.0) or of a formal “extension” of the algorithm, either of which would require a higher-level-protocol to enable it. In the context of such extension, certain restrictions on the original algorithm (such as a limit on the number of bidi classes) could be relaxed, so as to provide the most direct statement of the additional functionality.
In terms of URLs / IRIs or filenames, I would strongly prefer if the detection of these contexts were handled by the required higher-level-protocol, that is, that the UBA 2.0 would be defined without a definition of how to tell when these contexts are entered - but instead, restricted to the rules how to do bidi reordering given that a piece of text is in one of these special contexts.

In other words, the extended bidi algorithm would constitute a building block for use by higher level protocols, but a building block that handles features in a way that simple, permitted overrides of UBA 1.0 fail to achieve.

Doing it this way, makes clear the status of ordinary text data (outside the given protocols). These would continue to be supported by the existing bidi algorithm. In the absence of a higher-level protocol, the extensions cannot come into play, and the higher-level-protocol defines which parts of the text the extensions are to be applied to.

That provides for conformant interchange of data among implementations of the same protocol, and requires (potentially) adjustments of some data when exporting between protocols. However, structured data often needs to be adjusted already on export, so this would not seem to add any new issues.