UTC #179 properties feedback & recommendations

Markus Scherer & Josh Hadley / Unicode properties & algorithms group, 2024-apr-16

Participants

The following people have contributed to this document:

Markus Scherer (chair), Josh Hadley (vice chair), Christopher Chapman, Elango Cheran, Peter Constable, Mark Davis, Asmus Freytag, Manish Goregaokar, Ned Holbrook, Robin Leroy, Rick McGowan, Roozbeh Pournader, Ken Whistler, John Wilcock

1. Core Spec

1.1 Toward Clarifying the Rules of the Character Name Namespace

Recommended UTC actions

1. **Action Item** for Asmus Freytag, Ken Whistler, PAG: Review L2/24-073 and propose clarifications in TUS docs about the character name namespace, and possibly changes to the restrictions for assigning new character names. For Unicode 17.0. See L2/24-064 item 1.1.

Document

L2/24-073 by Erik Carvalhal Miller

From the doc:

> This document discusses inconsistencies, ambiguities, and loopholes in the Unicode Standard's specifications and policies regarding the character name namespace — as defined in the Character - Encoding Stability Policies, the core specification, and some of its annexes — and suggests remedies for greater consistency and clarity.

Among other findings and suggestions, it points out that we have two definitions of loose name matching and name uniqueness, where the less easily findable one ignores the words "CHARACTER", "DIGIT", and "LETTER".

Background information / discussion

Ken Whistler already added "and code point labels" to core spec chapter 3 definitions D5 & D6.

We should review the related terminology and rules, and consider updating the restrictions for assigning new character names (e.g., also ignoring "LIGATURE"). We should probably not change the loose character name matching rules.

2. UCD

2.1 Are abbreviation marks Terminal_Punctuation? (No.)

Recommended UTC actions

1. **Consensus**: Remove the Terminal_Punctuation property from U+0836 SAMARITAN ABBREVIATION MARK. For Unicode Version 16.0. See L2/24-064 item 2.1.
2. **Action Item** for Robin Leroy, PAG: In PropList.txt, remove the Terminal_Punctuation property from U+0836 SAMARITAN ABBREVIATION MARK. For Unicode Version 16.0. See L2/24-064 item 2.1.

PAG input

From Robin Leroy, PAG.

Recall the definition of Terminal_Punctuation,

> Punctuation characters that generally mark the end of textual units.

Compare Sentence_Terminal,

> Punctuation characters that generally mark the end of sentences. Used in Unicode Standard Annex #29, "Unicode Text Segmentation" [UAX29].

U+0836 SAMARITAN ABBREVIATION MARK has the Terminal_Punctuation property; *The Unicode Standard* has this to say about it:

> U+0836 SAMARITAN ABBREVIATION MARK follows an abbreviation.

Makes sense, the abbreviation is a textual unit, whereof it marks the end.

But then

> U+0970 ॰ DEVANAGARI ABBREVIATION SIGN appears after letters or combinations of letters and marks the sequence as an abbreviation.

Yet that one does not have the Terminal_Punctuation property.

The same applies to the Bengali, Gurmukhi, Gujarati, Avestan, Kaithi, Mahajani, Sharada, Khojki, Newa, Tirhuta, Modi, Takri, and Dogra abbreviation signs or marks.

Should they all be Terminal_Punctuation? Or should the Samaritan one not be? And why?

Note: The same does *not* apply to the Armenian abbreviation mark, which is meant to sit on top of the abbreviation and seems to be a mess as far as implementation is concerned, see https://r12a.github.io/scripts/armn/block.html#char055F and https://en.wiktionary.org/wiki/%D5%9F; nor to the GC=Cf Syriac abbreviation mark (prefix), nor to the Manichæan abbreviation marks (combining).

Background information / discussion

Scripsit Ken:

> As for the matter at hand, abbreviation marks per so do not terminate text elements from any *segmental* process point of view. They just mark that a "word" of some sort has been written in a shortened form. Usually an abbreviation mark occurs at the end of *that* text element ("Dr.", "St.", whatever), but it may not, as in Swedish internal abbreviation, or other abbreviations that don't involve separate marks ("I18n", anyone?). So a terminal abbreviation mark, in a sense, can be seen as terminal for that particular abbreviated word, they don't function like commas, full stops, section marks, etc., etc., which are not *parts* of words, but whose primary purpose is to demarcate text elements from each other.

2.2 Bidi class of Nabla variants

Recommended UTC actions

1. **Consensus**: Change the Bidi_Class of the five ∇ mathematical variants, U+1D6C1 (BOLD), U+1D6FB (ITALIC), U+1D735 (BOLD ITALIC), U+1D76F (SANS-SERIF BOLD), and U+1D7A9 (SANS-SERIF BOLD ITALIC), from Left_To_Right (L) to Other_Neutral (ON). For Unicode Version 16.0. See L2/24-064 item 2.2.
2. **Action Item** for Robin Leroy, PAG: In UnicodeData.txt and derived files, change the Bidi_Class of the five ∇ mathematical variants, U+1D6C1 (BOLD), U+1D6FB (ITALIC), U+1D735 (BOLD ITALIC), U+1D76F (SANS-SERIF BOLD), and U+1D7A9 (SANS-SERIF BOLD ITALIC), from Left_To_Right (L) to Other_Neutral (ON). For Unicode Version 16.0. See L2/24-064 item 2.2.

Feedback (verbatim)

PRI-497

Date/Time: Fri Feb 16 13:51:49 CST 2024
ReportID: ID20240216135149
Name: Charlotte Buff
Report Type: Other Document Submission
Opt Subject: Bidi class of Nabla variants

I propose changing the Bidi_Class value of the following characters to Other_Neutral from their current value Left_to_Right:

U+1D6C1 MATHEMATICAL BOLD NABLA
U+1D6FB MATHEMATICAL ITALIC NABLA
U+1D735 MATHEMATICAL BOLD ITALIC NABLA
U+1D76F MATHEMATICAL SANS-SERIF BOLD NABLA
U+1D7A9 MATHEMATICAL SANS-SERIF BOLD ITALIC NABLA

U+2207 NABLA has Bidi_Class=Other_Neutral, so its font variants should share the same property value. This is how it already works for U+2202 PARTIAL DIFFERENTIAL and its respective font variants, all of which are Other_Neutral.

2.3 Defective decision on ScriptExtensions at UTC-178

Recommended UTC actions

1. **Consensus**: Add all 53 entries to ScriptExtensions.txt as proposed in L2/23-280, including U+0300 and U+0308, for Unicode Version 16.0. See L2/24-064 item 2.3. This decision supersedes consensus 178-C39.

PAG input

Decision UTC-178-C39, per the recommendation of SAH, was to

> Add 51 entries to ScriptExtensions.txt as proposed in L2/23-280 for a future version of the Unicode Standard. [Ref. Section 18 of L2/24-013R]

There are 53 entries proposed in that file. Some UTC participants think the intent was to exclude U+0300 and U+0308, others dissent.

PAG discussed the entries in that document and agreed to add all fifty-three of them provided the test failures involving U+0300 and U+0308 are indeed spurious; see [Script encoding issue #375 comment from Markus 2024-feb-22] and https://github.com/unicode-org/unicodetools/pull/614#issuecomment-1960388135.

Let's have a clean decision so we know where we actually stand.

2.4 Ideographic property value of U+18CFF

Recommended UTC actions

1. **No Action**: PAG recommends no action: the change has been made in PropList.txt.

Feedback (verbatim)

PRI-497

Date/Time: Fri Feb 16 14:01:04 CST 2024
ReportID: ID20240216140104
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 497: Ideographic property value of U+18CFF

U+18CFF KHITAN SMALL SCRIPT CHARACTER-18CFF currently has the property Ideographic=No, but the value should be Yes like with all other Khitan Small Script characters.

Background information / discussion

Indeed. Fixed in https://github.com/unicode-org/unicodetools/pull/715.

2.5 Kirat Rai InSC and InPC data

Recommended UTC actions

1. **No Action**: PAG recommends no action: Indic_Positional_Category and Indic_Syllabic_Category *is* applicable to scripts that use a visual order encoding model; for instance, Thai has both.

Feedback (verbatim)

Date/Time: Wed Feb 14 15:25:32 CST 2024
ReportID: ID20240214152532
Name: Norbert Lindenberg
Report Type: Public Review Issue
Opt Subject: 497 [PAG]

The UCD in Unicode 16.0 alpha defines Indic syllabic and positional categories for the Kirat Rai script. The final proposal for Kirat Rai, L2/22-043R, does not provide such data.

I don't think the omission of Indic data in the proposal was an oversight. The proposal states: "The script does not have the rendering complexity of traditional Brahmic scripts (no reordering, no combining marks, and no conjuncts)." This means, a simple visual encoding model, where spacing characters are encoded in left-to-right order, is sufficient for the script and is intended. Indic data, which implies a phonetic encoding order, should not be added.

The phonetic encoding model used for most Brahmic scripts and the visual encoding model used for most non-Brahmic scripts are fundamentally incompatible and should never be combined. Even for a very simple script like Kirat Rai, there's a slight potential for conflicts between the visual and phonetic encoding order based on the Brahmic cluster model of the OpenType Universal Shaping Engine: Any sequence of characters with gc=Lm (of which Kirat Rai has five) would become part of a single cluster and would have to be encoded in primarily phonetic order.

Any data for Kirat Rai should be removed from IndicSyllabicCategory.txt and IndicPositionalCategory.txt.

2.6 Missing @missing for ArabicShaping.txt

Recommended UTC actions

1. **No Action**: PAG recommends no action: The header of ArabicShaping.txt has been updated editorially, and the pending update of UAX #44 very explicitly recommends for Joining_Type to parse the derived data file.

PAG input

From Mark Davis, PAG.

The definition of Joining_Type in ArabicShaping.txt is unusual, in that in addition to the values listed in the files, it has a complex default depending on the assignment of another property, namely General_Category:

```
Unset
# Note: Code points that are not explicitly listed in this file are
# either of joining type T or U:
#
# - Those that are not explicitly listed and that are of General Category Mn, Me, or Cf
#   have joining type T.
# - All others not explicitly listed have joining type U.
```

This requires special handling if ArabicShaping.txt is used directly.

The PAG discussed a proposal to explicitly list all characters that have joining type T, and to have an invariant test checking the relationship to General_Category. The maintainers of ArabicShaping.txt pointed out that the file would become difficult to review and maintain. An alternative was suggested to extend the `@missing` mechanism to support the reference to another property.

Robin pointed out the following:

> This is hardly the first property for which we have a derivation based on GC combined with a manually-maintained set that is sufficiently large and growing that we want to make it a property: Alphabetic, Uppercase and Lowercase, Grapheme_Extend, (X)ID_(Start|Continue), Math all come to mind.
>
> The way we do that is normally neither to have a fully manually-maintained assignment checked by an invariant nor by having a particularly warty file defining that property directly. Instead it is to have a contributory property, and to use it in the derivation.
>
> So if we were to follow our normal practice, ArabicShaping.txt would define an enumerated [contributory] property Other_Joining_Type with an additional value `Just_Figure_It_Out_From_The_General_Category` which is the default (so `@missing 0000..10FFFF;JFIOFTGC;No_Joining_Group`), and the real Joining_Type, given in DerivedJoiningType, would be derived as « `T` if General_Category = `Mn`, `Me`, or `Cf` and

> Other_Joining_Type=`JFIOFTGC`, `U` if General_Category is something else and Other_Joining_Type=`JFIOFTGC`, and the value of Other_Joining_Type otherwise ».

Robin then proceeded to treat ArabicShaping.txt that way in the Unicode tools, which worked: https://github.com/unicode-org/unicodetools/pull/657. This fiction makes the file and property regular as far as the *production* of the UCD is concerned.

For *consumers* of the UCD however, it is needlessly convoluted to describe things that way and to introduce a new contributory property. Instead we should encourage users to ignore column 1 of ArabicShaping.txt and to read DerivedJoiningType.txt, which is a far more regular file. This was already pointed out in https://www.unicode.org/reports/tr44/#Default_Values.

The header of ArabicShaping.txt has been updated to make that point more clearly: https://github.com/unicode-org/unicodetools/pull/671/files.

2.7 PAG considerations on L2/24-059

Recommended UTC actions

1. **Action Item** for Robin Leroy, PAG: Inform ISO/IEC JTC 1/SC 22/WG 21/SG 16 of the UTC's dispositions on L2/24-059. Inquire about the need for treating non-Emoji_Presentation characters as if they had East_Asian_Width=Wide in [format.string.std] (13.3) and (13.4). See L2/24-064 item 2.7.

Document

L2/24-059 by Jules Bertholet. *Proposal to change the East_Asian_Width property of the Yijing symbols*

The document is a proposal to

> Update the East_Asian_Width property values of all characters in the following ranges from Neutral to Wide:
>
> - U+2630…U+2637 (TRIGRAM FOR HEAVEN (☰) … TRIGRAM FOR EARTH (☷))
> - U+268A…U+268F (MONOGRAM FOR YANG (⚊) … DIGRAM FOR GREATER YIN (⚏))
> - U+4DC0…U+4DFF (HEXAGRAM FOR THE CREATIVE HEAVEN (䷀) … HEXAGRAM FOR BEFORE COMPLETION (䷿))
> - U+1D300…U+1D356 (MONOGRAM FOR EARTH (𝌀) … TETRAGRAM FOR FOSTERING ((𝍖))

Background information / discussion

PAG does not object to the proposed property assignment change; CJK & Unihan Working Group to recommend it to UTC if it is agreeable to that group.

The liaison officer to ISO/IEC JTC 1/SC 22 notes that this would simplify the description of field widths in ISO/IEC 14882; in the current working draft, width computation for `std::format` padding purposes is defined as follows in [format.string.std]:

13 For a sequence of characters in UTF-8, UTF-16, or UTF-32, an implementation should use as its field width the sum of the field widths of the first code point of each extended grapheme cluster. Extended grapheme clusters are defined by UAX #29 of the Unicode Standard. The following code points have a field width of 2:
(13.1) — any code point with the East_Asian_Width="W" or East_Asian_Width="F" Derived Extracted Property as described by UAX #44 of the Unicode Standard
(13.2) — U+4DC0 – U+4DFF (Yijing Hexagram Symbols)
(13.3) — U+1F300 – U+1F5FF (Miscellaneous Symbols and Pictographs)
(13.4) — U+1F900 – U+1F9FF (Supplemental Symbols and Pictographs)
The field width of all other code points is 1.

The proposal would subsume (13.2) in (13.1).

13.3 and 13.4 seem more dubious: East_Asian_Width is aligned with Emoji_Presentation there, and Neutral seems appropriate for the random non-Emoji_Presentation symbols: https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5Cp%7BBlock%3DMiscellaneous+Symbols+and+Pictographs%7D%5Cp%7BBlock%3DSupplemental+Symbols+and+Pictographs%7D&g=ea&i=age%2Cemoji_presentation.

East_Asian_Width is also used in line breaking, currently in rule LB30; we have a proposal to broaden that use in “Line breaking of U+2019 et al. in zh-Hans contexts”.

As far as line breaking is concerned, there are no issues with east-asian-widening the characters at hand: they are neither lb=OP nor lb=CP, so this has no effect on LB30; and for #197, having characters like these adjacent to both sides of an ambiguous quotation mark is not something that we should expect in non-east-asian typesetting (we have a sample of what we expect adjacent to and outside quotation marks in rule LB15b, that is mostly whitespace and punctuation).

An example of a thing that *should not* be east-asian-widened based on such algorithmic considerations, regardless of how wide it physically is, is something like U+2E3A ⸮ DOUBLE QUESTION MARK, currently ea=N; since then the changes proposed in #197 would introduce a most infelicitous break in *Sagt die Katze wirklich „喵÷“⸮*

2.8 PAG review of L2/23-284 Proposal to encode two small form CJK characters for Chinese

Recommended UTC actions

1. **No Action**: PAG recommends no action.

PAG input

From Robin Leroy, staring at the CJK report (L2/23-284)

Robin looked at L2/23-284, *Proposal to encode two small form CJK characters for Chinese*, recommended by L2/24-012 Section 21), and drafted the UCD change in https://github.com/unicode-org/unicodetools/pull/666.

Robin Leroy and Ken Whistler discussed a number of questions:

- Should they be compatibility decomposable to `<small> 513F` and `<small> 5152`? (Absolutely not.)
    - This would set a bad precedent for Han characters.
- Is Extender appropriate for *érhuà*? (Yes.)
    - Conceptually, it is akin to a vowel length mark, except the extension is with [ə˞] rather than just a non-gliding extension of the vowel per se. It isn't analyzed as a segmental /-r/, because Mandarin treats all its vocalic offglides as vowel nuclei, rather than phonemic semi-vowels.
- Should it be Diacritic, since it modifies the preceding character? (No.)
    - It behaves as a phonetic complement, and is also sometimes standing in for another syllable that has become an [ə˞], as in 这里↦这儿, 那里↦那儿, 哪里↦哪儿. It was pointed out that the example 花儿, given in the proposal, is actually in origin the phonological reduction of 花子.
- Is lb=NS right? (Yes.)
    - It should stick to preceding CJKV ideographs in the same way as, *e.g.*, iteration marks do.
- Is gc=Lo right? (Yes.)
    - The alternative discussed was Lm. There was some discussion of the abstract relation between the Ideographic and General_Category properties, but given the analysis in the discussion of Diacritic above, Lo felt more appropriate anyway.

In conclusion, the proposed properties are fine.

2.9 Schematic name of U+1878

Recommended UTC actions

1. **No Action**: PAG recommends no action: This issue has been addressed editorially. New schematic name: "MONGOLIAN CHA WITH 2 DOTS"

PAG input

From Robin Leroy, PAG, while processing L2/24-025 which proposes the next code point.

ArabicShaping.txt has

```
Unset
1878; MONGOLIAN MANCHU CHA WITH 2 DOTS; D; No_Joining_Group
```

U+1878 is called MONGOLIAN LETTER CHA WITH TWO DOTS, emphasis added on the absence of the word MANCHU.

The character was added based on L2/17-007 = ISO/IEC JTC 1/SC 2/WG 2 N4781, which does not contain the word Manchu (language subtag: mnc). Instead it mentions Buryat (language subtag: bua).

Is the word Manchu correct in that schematic name? (No.)

2.10 Should ScriptExtensions.txt be grouped by value? (No.)

Recommended UTC actions

1. **Note**: The UCD file ScriptExtensions.txt is changing the order of lines (but not the format) from grouping by value to simple code point order. The 16.0 beta PRI will include a version of 15.1 scx data for comparison with 16.0 data.

PAG input

From @markusicu and @eggrobin, PAG

While reviewing a change for UTC-178-C39, Markus had this remark:

> I find it hard to figure out what's changing for which character, for those that already had scx, because we group them by scx value which is a set of sc values -- so they move far in the file when the first listed script changes.
>
> I suspect that this file would be easier to read and maintain if we gave up on grouping by value, and instead simply listed by code points and ranges.

Robin drafted a change to switch to such a format, and reënacted all changes to ScriptExtensions.txt since Unicode 15.0 to see if they would have been clearer: https://github.com/unicode-org/unicodetools/pull/723.

Background information / discussion

The following from UAX #44 is relevant: https://www.unicode.org/reports/tr44/#Property_Values_As_Sets

> In the case of Script_Extensions, in particular, the set of sets which constitute meaningful values of the property is relatively small, and could be explicitly evaluated for any particular Unicode version. For example:

```
Unset
{{Adlm, Arab, Mand, Mani, Phlp, Rohg, Sogd, Syrc}, {Arab, Copt}, {Arab, Rohg}, {Arab, Syrc},
{Arab, Thaa}, {Arab, Syrc, Thaa}, {Armn, Geor}, ...}
```

> However, an enumeration of this set of set values is unlikely to be of much implementation value, and would be likely to change significantly between versions of the standard.

Yet we had been formatting the file as if it were an enumerated property.

2.11 UAX #44 clarify that a duplicate property (value) alias can be changed

Recommended UTC actions

1. **Action Item** for Ken Whistler, PAG: In UAX #44 section 5.8 Property and Property Value Aliases, clarify that the duplicate listing of property or value aliases on the same line in the aliases file establishes only one alias, and that either one of the duplicate listings may be replaced when adding a new, unique alias. For Unicode 16.0. See L2/24-064 item 2.11.

PAG input

From Markus Scherer, PAG

Property aliases and property value aliases are stable: See the Alias Stability policy.

During PAG discussion, we found that we don't make it clear what this means for the duplicate listing of aliases.

We sometimes list the same alias multiple times; generally twice for what we advertise as "short" and "long" aliases when we don't want to invent distinct names right away -- simply because the data file format requires at least two fields with alias strings.

For example, PropertyAliases.txt shows the same alias twice for the Hyphen property, and PropertyValueAliases.txt shows duplicate aliases for many values of many properties -- especially many Block names, ccc values, InCB, InPC, InSC, and jg.

“A duplicate listing doesn’t create two aliases.” Listing the same alias (including its precise spelling) more than once still counts as only one alias in the set for the property or value.

In particular, when the same alias is listed twice for the "short" and "long" forms, we may decide to replace either one of the duplicate listings of that same alias (but not both listings!) with a new, unique alias. This would then increase the set of aliases. We may add a new "short" alias when the existing one turns out to be inconveniently long, or add a new "long" one to be more descriptive or less ambiguous. In all of these cases, the existing alias remains absolutely valid and unchanged.

We should clarify this in UAX #44 section 5.8 Property and Property Value Aliases.

This meets all the requirements of the existing alias stability policy.

2.12 [PRI-486] Feedback on proposed stabilization of UAX #42 from asmus: stable aliases

Recommended UTC actions

1. **Action Item** for Ken Whistler, PAG: In UAX #44 section 5.8 Property and Property Value Aliases, document that the spelling of aliases is stable, but not their order, and point to the stability policy. For Unicode 16.0. See L2/24-064 item 2.12.

Otherwise no action. UAX #42 is now planned to continue to be published. Property parsers should normally recognize aliases.

Feedback (verbatim)

PRI-486

Date/Time: Thu Dec 14 16:11:56 CST 2023
ReportID: ID20231214161156
Name: asmus
Report Type: Public Review Issue
Opt Subject: 486
A critical use case for external specifications is the fact that UAX#42 chooses not only the "short" alias for properties and values, but does it in a stable way, whereas the PropertyValueAliases and PropertyAliases are subject to changes in capitalization etc that are within the Loose matching envelope.

In addition, aliases may be augemented by new aliases (sometimes because of corrections). While the old aliases are not removed, they may be moved to a different position on the line. It is therefore not possible to use these files for *stable* keys as they would be needed for DTDs or similar use cases.

There's at least one IETF specification that normatively references UAX#42 for that purpose, and like UAX#42 it is a XML data format that needs to be able to /identify/ unicode properties and values in a stable (but does not need to provide a listing of the actual property data).

Identifying a stable set of keys that do not require loose matching is one feature that is unique to UAX#42 and cannot be replaced by accessing the original UCD. If UAX#42 is to be retired, this functionality should be replaced and linked from the page that documents the stabilization of UAX#42.

3. New Scripts & Characters

Recommended UTC actions

1. **Note**: L2/22-289R “Final Proposal to encode the Tai Yo Script” includes the following one-character typo: In the proposed UnicodeData.txt lines, TAI YO LETTER PHO is listed with the code point U+1E5D3 rather than U+1E6D3 which fits with the adjacent characters. This has been corrected in draft data for Tai Yo.

PAG members reviewed the following proposals, provided feedback to SAH, and the feedback has been addressed.

No further recommended actions from our side.

1. L2/23-280 Request for additions to ScriptExtensions - Pournader
2. L2/24-046 Proposal to encode ARABIC LETTER NOON WITH RING ABOVE -- Evans
3. L2/24-002 Proposal to encode eight more Arabic honorifics
4. L2/24-055 Proposal to add Arabic letter for Swahili -- Denis Moyogo Jacquerye
5. L2/24-077 Proposal to encode twenty-five more Arabic honorifics Roozbeh Pournader, Rikza F. Sh., and Amir Mahdi Moslehi
6. L2/24-004R Proposal for encoding the Beria Erfe (Zaghawa) script in the SMP of the UCS -- Siddick Adam Issa, et al.
7. L2/23-188 Unicode request for compound tone diacritics -- Miller
8. L2/24-021 Do Not Emit (16.0.0) -- Pournader
9. L2/24-048 Review of Proposal to encode 114 Tangut components -- West
10. L2/24-025 Proposal to Encode One Manchu Letter -- CheonHyeong Sim, et al.
11. L2/24-058R Required conjunct forms in extended grapheme clusters -- Lindenberg
12. L2/24-051 Unicode request for affricate ligatures -- Miller
13. L2/24-080 Unicode request for IPA diacritics above and one below Kirk Miller
14. L2/24-050 Unicode request for letters with palatal hook -- Miller
    - Like all 27 existing characters WITH PALATAL HOOK, they do not have decompositions with U+0321 COMBINING PALATALIZED HOOK BELOW. Because of this, DoNotEmit data is provided.
    - U+1DF38 LATIN SMALL LETTER TS DIGRAPH WITH PALATAL HOOK and 1DF2E LATIN SMALL LETTER DZ DIGRAPH WITH PALATAL HOOK are like U+02A6 ʦ and U+02A3 ʣ: they do not have a compatibility decomposition to ts̡ and dz̡ (and note Figures 23–26 which show the hook in the middle of the ligature).
    - No uppercase counterparts.
15. L2/24-052 Unicode request for modifier pre-Kiel click letters -- Miller
16. L2/24-081 Unicode request for modifier capital S -- Miller
17. L2/24-018 Unicode request for one chess symbol -- Bala and Miller
18. L2/23-207 Unicode request for historical asteroid symbols -- Miller & Bala
19. L2/23-218 Proposal to encode Geomantic Figures -- Pollard
20. L2/24-020 Unicode request for shatranj symbols -- Bala and Miller
21. L2/23-246 Proposal to encode 114 Tangut components, L2/23-247 Proposal to encode 114 Tangut components -- West

22. L2/23-024 Proposal to encode Tolong Siki
    - U+11DD9 (sign selā), being a vowel length mark, should be gc=Lm [Jan: rather than proposed Lo] (as well as Diacritic=Yes and Extender=Yes).
    - U+11DDA (sign hecakā) is fine as an Other Letter.
    - Other than that, this looks like a straightforward unicameral alphabetic script.

4. Bidi

4.1 UAX #9 rules Xn vs. adjusting levels based on implicit types

Recommended UTC actions

1. **Action Item** for Manish Goregaokar, PAG: Reword the first paragraph of UAX #9 section 3.3.3 referring to the following two subsections instead of using “soon”. For Unicode 16.0. See L2/24-064 item 4.1.

Feedback (verbatim)

PRI-497

Date/Time: Fri Feb 23 14:41:37 CST 2024
ReportID: ID20240223144137
Name: Diggory Hardy
Report Type: Error Report
Opt Subject: TR9

In TR9, version 15.1.0, section 3.3.3 -
http://www.unicode.org/reports/tr9/#Preparations_for_Implicit_Processing

It is implied that rules X1-X8 assign embedding levels to characters based only on the paragraph level and explicit formatting tokens, but that these levels will soon be adjusted based on characters' "implicit bidirectional types".

X9 does not mention adjusting characters' levels.

X10, point 1 does not either. It does however imply that level runs should already have been calculated, and thus that character embedding levels should already have been adjusted.

Furthermore, I do not see any explanation of the calculation of embedding levels, only examples. Is it possible that this part of the specification got lost in a re-organisation?

By the way, I do not find the mixture of prose, algorithms and examples used in this article the easiest to follow, but do not have strong suggestions (only that specifications usually do not bother discussing optimisations which may be applied to implementations).

5. Segmentation

5.1 Ancient UAX #14 errors found relatively recently by Charlotte Buff

Recommended UTC actions

1. **Action Item** for Robin Leroy, PAG: In UAX #14, correct the description of the behaviour of line breaking classes PO and PR when separated from numbers by spaces. For Unicode Version 16.0. See L2/24-064 item 5.1.

PAG input

From Robin Leroy, PAG.

A glance at Standing Document 2 reveals UTC-171-A130 Action Item for Christopher Chapman, Properties & Algorithms Group: Review feedback from Charlotte Buff on PRI #446 [Tue Apr 5 07:14:53 CDT 2022] and provide recommendations.

Chris was at the time the editor of UAX #14, but the editorship has since befallen me, so I thought I might take a look.

The relevant piece of feedback from PRI-446 is

```
Unset
Date/Time: Tue Apr 5 07:14:53 CDT 2022
Name: Charlotte Buff
Report Type: Public Review Issue
Opt Subject: 446

In UAX #14, the descriptions of the line breaking classes Postfix_Numeric
(PO) and Prefix_Numeric (PR) don't match the actual behaviour of the line
breaking algorithm when it comes to the treatment of intervening spaces.

The description of PO states:

    »Characters that usually follow a numerical expression may not be
    separated from preceding numeric characters or preceding closing
    characters, even if one or more space characters intervene. For
    example, there is no break opportunity in "(12.00) %".«

And similarly, the description of PR states:

    »Characters that usually precede a numerical expression may not be
    separated from following numeric characters or following opening
    characters, even if a space character intervenes. For example,
    there is no break opportunity in "$ (100.00)".«
```

```
However, the actual line breaking rules that govern these classes
(LB23a, LB24, LB25, LB27) don't actually contain a special provision for
intervening spaces. As a result, the strings given as examples *do* in fact
contain line breaking opportunities simply due to rule LB18 (Break after
spaces) – before the percent sign in the former and before the opening
parenthesis in the latter. This can be confirmed via the use of Unicode's
online utility tool for breaks and segmentation
(https://util.unicode.org/UnicodeJsps/breaks.jsp).
```

A look at https://eggrobin.github.io/unicode-annotations/alba.html?v=15.1.0&base=3.0.0#p292 reveals that the issue has been there since the dawn of what was then called Unicode Technical Report #14, Line Breaking Properties (revision 6, Unicode Version 3.0.0). We must therefore dig for deeper documents from before the dawn of UAX #14.

Revision 5 (https://www.unicode.org/reports/tr14-5/tr14-5.pdf) is not meaningfully different as far as this issue is concerned; it already has the statements « There is no break in "(12.00) %" » and « There is no break in "$ (100.00)" », and it already has a rule LB 18 that fails to implement that (well, it has *two* rules called LB 18, the second one is the relevant one; there is no rule LB 19).

Revision 4 (https://www.unicode.org/reports/tr14-4/) predates the translation of Michel's pair table into the ancestral rules; the classification found in that revision, and the description of the associated behaviour, is significantly different from what became the Line_Break property. While the examples "(12.00) %" and "$ (100.00)" are nowhere to be found in that revision, we find these sections:

> Postfix characters (XB)
> Characters that usually follow a numerical expression, may not be separated from preceding numeric characters or preceding closing characters, EVEN if space character intervenes.
>
> PERCENT SIGN — 0025

and

> 5.14 Prefix characters (XA)
> Characters that usually precede a numerical expression, may not be separated from following numeric characters, EVEN if space character intervenes.
>
> DOLLAR SIGN — U+0024

I suspect the offending examples are relics of that algorithm that erroneously survived the merger with Michel's work.

It would make sense to prohibit a break in, say, 12 %, though that rapidly falls into the same bucket as units, for which one surely needs a no-break space. In any case that would require a careful review of lb=PO, which seems outside the scope of this action item.

5.2 Ancient feedback from David Corbett and Asmus on Hebrew linebreaking

Recommended UTC actions

1. **Consensus**: In UAX #14, change rule LB21a from `HL (HY | BA) ×` to `HL (HY | BA) × [^HL]` (do not break after the hyphen in Hebrew-hyphen-non-Hebrew) to match the rationale for adding LB21a given in L2/11-141R. For Unicode Version 16.0. See L2/24-064 item 5.2.
2. **Action Item** for Robin Leroy, PAG: In UAX #14, change rule LB21a from `HL (HY | BA) ×` to `HL (HY | BA) × [^HL]`. For Unicode Version 16.0. See L2/24-064 item 5.2.

Feedback

From Robin Leroy, PAG

A glance at Standing Document 2 shows that I have inherited the following action item:

Action	Owners	Description	Documents	Status
UTC-151-A91	Robin Leroy, PAG	Review feedback from David Corbett on PRI #335 of April 29 and May 4, 2017, and make any recommendations for property changes. (linebreak issues)		

Looking at PRI-335, the one from 2017-05-04 was done by UTC-175-C23.
I reproduce here the text of the relevant one.

```
Unset
Date/Time: Sat Apr 29 23:16:48 CDT 2017
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: PRI #335: Breaking between Hebrew and hyphens

The annex says that U+00AD SOFT HYPHEN “is an invisible format character with
no width. It marks the place where an optional line break may occur inside a word.
It can be used with all scripts.” However, because of LB21a, it does not work with
the Hebrew script.

U+05BE HEBREW PUNCTUATION MAQAF, the Hebrew hyphen, has line break class BA. However,
LB21a prevents a line break between HL and BA, making the maqaf essentially a
non-breaking hyphen. If this is intentional, it should have line break class GL, to
make the intent clear; if not, LB21a needs refinement, or even deletion.

It would be nice if the annex explained the rationale for LB21a. See also L2/13-083.
```

L2/13-083 is a document by one Asmus Freytag.

Back in L2/13-083, Asmus had found the rationale to the extent that it was documented in L2, which is not a very great extent: “With <hebrew hyphen non-hebrew>, there is no break on either side of the hyphen.”. (Aside: That document then goes on to suggest we introduce another « treat as » rule. As an implementer, that way madness lies; let me shun that; no more of that.)
In my recent work on the annotated line breaking algorithm, I found that reason as well, but also found an example—the ICU « and » list format—:
https://eggrobin.github.io/unicode-annotations/alba.html?v=15.1.0#p432.2.a.

The feedback from David Corbett pertains to intra-script hyphen usage, and this is also mentioned in Asmus’s document. A cursory Google search finds that a break should be permitted after a maqaf within a Hebrew word, see https://tex.stackexchange.com/a/447275.

This is easy to fix, change `HL (HY | BA) ×` to `HL (HY | BA) × [^HL]` (we could put something like (AL|ID|AS|ever growing list of|spam|spam|spam|eggs|spam) on the right-hand-side of this rule but this will just make our lives worse).

5.3 Error in the derivation of InCB

Recommended UTC actions

1. **Consensus**: Change the derivation of Indic_Conjunct_Break = Extend to exclude only those characters that are Indic_Conjunct_Break=Linker or Indic_Conjunct_Break=Consonant from the set of nonstarters with Grapheme_Cluster_Break=Extend. For Unicode 16.0. See L2/24-064 item 5.3.
2. **Action Item** for Ken Whistler, PAG: Change the derivation of Indic_Conjunct_Break = Extend to exclude only those characters that are Indic_Conjunct_Break=Linker or Indic_Conjunct_Break=Consonant from the set of nonstarters with Grapheme_Cluster_Break=Extend. For Unicode Version 16.0. See L2/24-064 item 5.3.
3. **Action Item** for Robin Leroy, PAG: In DerivedCoreProperties.txt, update the Indic_Conjunct_Break assignments according to the corrected derivation. For Unicode Version 16.0. See L2/24-064 item 5.3.

Feedback

From Robin Leroy, ICU-TC and ICU4X-TC.

UAX #44 defines the definition of InCB as follows:

- Define the set of applicable scripts. For Unicode 15.1, the set is defined as S = [\p{sc=Beng}\p{sc=Deva}\p{sc=Gujr}\p{sc=Mlym}\p{sc=Orya}\p{sc=Telu}]
- Then for any character C:
    1. InCB = Linker iff C in [S &\p{Indic_Syllabic_Category=Virama}]
    2. InCB = Consonant iff C in [S &\p{Indic_Syllabic_Category=Consonant}]
    3. InCB = Extend iff C in
       [[\p{gcb=Extend}-\p{ccc=0}]
       \p{gcb=ZWJ}
       -\p{Indic_Syllabic_Category=Virama}
       -\p{Indic_Syllabic_Category=Consonant}]
    4. Otherwise, InCB = None (the default value)

The derivation in 3. means that some ccc≠0 characters are excluded from InCB=Extend, namely viramas (and reordered consonants, if that is a thing) from inapplicable scripts. An example of that is U+1163F MODI SIGN VIRAMA. Thankfully this only affects degenerate cases, though it is embarrassing.

This was spotted as ICU4X was trying to implement the new rules; comparing it on random strings with the ICU behaviour (which uses the macros that referred to CCC directly) exposed the bug. See discussion in https://github.com/unicode-org/icu4x/pull/4536.

The fix is in principle straightforward, exclude exactly the sets from 1. and 2. in 3., instead of forgetting the &S (intersection with S). For readability, we can just reuse the just-defined sets—this was suggested by Asmus—:

- Define the set of applicable scripts. For Unicode 15.1, the set is defined as S = [\p{sc=Beng}\p{sc=Deva}\p{sc=Gujr}\p{sc=Mlym}\p{sc=Orya}\p{sc=Telu}]
- Then for any character C:
    1. InCB = Linker iff C in [S &\p{Indic_Syllabic_Category=Virama}]
    2. InCB = Consonant iff C in [S &\p{Indic_Syllabic_Category=Consonant}]
    3. InCB = Extend iff C in
       [[\p{gcb=Extend}-\p{ccc=0}]
       \p{gcb=ZWJ}
       -**\p{InCB = Linker}**
       -**\p{InCB = Consonant}**]
    4. Otherwise, InCB = None (the default value)

This moves twenty-one characters from InCB=None to InCB=Extend (the viramas of Gurmukhi, Tamil, Kannada, Sinhala, Balinese, Syloti Nagri, Saraushtra, Javanese, Brahmi, Kaithi, Sharada, Khojki, Grantha, Newa, Tirhuta, Siddham, Modi, Takri, Dogra, Nandinagari, and Bhaisuki).

5.4 L2/24-058R Required conjunct forms in extended grapheme clusters

Recommended UTC actions

1. **Consensus**: Change the derivation of Indic_Conjunct_Break according to L2/24-058R. For Unicode 17.0. See L2/24-064 item 5.4.
2. **Action Item** for Robin Leroy, PAG: In UAX #29, change the derivation of Indic_Conjunct_Break according to L2/24-058R. For Unicode 17.0. See L2/24-064 item 5.4.
3. **Action Item** for Robin Leroy, PAG: Update DerivedCoreProperties.txt according to the new derivation of Indic_Conjunct_Break from L2/24-058R. For Unicode 17.0. See L2/24-064 item 5.4.

Document

L2/24-058R *Required conjunct forms in extended grapheme clusters* by Norbert Lindenberg

From the document:

> This document proposes to change the definition of the property Indic_Conjunct_Break in UAX 44 Unicode Character Database, which defines values that are used in UAX 29 Unicode Text Segmentation in preventing extended grapheme cluster breaks within conjuncts and conjunct forms of some Brahmic scripts

5.5 Line breaking of U+2019 et al. in zh-Hans contexts

Recommended UTC actions

1. **Consensus**: In Unicode Standard Annex #14, change rule LB19 and add LB19a to treat gc=Pi as opening and gc=Pf as closing when surrounded by characters with East Asian Width F, W, or H, as described in L2/24-064 item 5.5. For Unicode Version 16.0.
2. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, change rule LB19 and add LB19a to treat gc=Pi as opening and gc=Pf as closing when surrounded by characters with East Asian Width F, W, or H, as described in L2/24-064 item 5.5. Consider using a macro throughout the rules for [\p{ea=F}\p{ea=W}\p{ea=H}]. For Unicode Version 16.0.
3. **Action Item** for Robin Leroy, PAG: In LineBreakTest.txt, Change rule LB19 and add LB19a as described in L2/24-064 item 5.5, and add realistic handwritten test cases. For Unicode Version 16.0. See L2/24-064 item 5.5.

Feedback

From Robin Leroy, PAG, based on feedback from Liang Hai, SAH and from Henri Sivonen, ICU4X-TC (in turn from a Firefox user).

SAH, while considering a proposal for VSes disambiguating the EAW of U+2019 and its friends, had questions about the behaviour of those characters in the line breaking algorithm, see https://github.com/unicode-org/sah/issues/352#issuecomment-1762159693.

Around the same time, Mozilla received complaints about the line breaking behaviour of these characters in zh-Hans usage, which were brought to the attention of ICU4X-TC, see https://github.com/unicode-org/icu4x/issues/3255#issuecomment-1771263967.

The issue is that 喵喵‘喵’喵喵 has line break opportunities as follows:
喵÷喵‘喵’喵÷喵

which is missing the break opportunities granted by the zh-Hant quotation marks in 喵喵「喵」喵喵:
喵÷喵÷「喵」÷喵÷喵
We thank Joshua Tsai for pointing out that the relevant standard, GB/T 15834-2011, only forbids the breaks inside of the quotation marks (subclause 5.1.3).

With a language-dependent resolution of QU to OP or CL things would work, but we should do a better job in the language-independent case.

Ideally, in a `[\p{ea=F}\p{ea=W}\p{ea=H}]` context, or when affected by VS-2, we would treat Pi as OP, Pf as CL.
For the sake of the implementer’s rapidly dwindling sanity, this should really be turned into something that does not involve *treat as* rules though, which may make it intractable to do anything about the VS-2. We need to do a better job in the VS-free case anyway, and if we do a good enough job maybe ignoring the VS is OK.

The breaks in question are prevented by https://www.unicode.org/reports/tr14/#LB19: × QU and QU ×; so we could suppress them in an ea=F, W, or H context. It is important to check the context on both sides, because Pi

and Pf are not always initial and final, see http://www.unicode.org/versions/Unicode15.0.0/ch06.pdf#G19995; we do not want to introduce breaks inside the quotation marks in *Die Katze hat „喵“ gesagt.*

Thus, we want to change LB19 to × QU except in [\p{ea=F}\p{ea=W}\p{ea=H}] Pi [\p{ea=F}\p{ea=W}\p{ea=H}], and QU × except in [\p{ea=F}\p{ea=W}\p{ea=H}] Pf [\p{ea=F}\p{ea=W}\p{ea=H}]. The change to LB19 is thus as follows:

> ~~LB19 Do not break before or after quotation marks, such as ‘ ” ’.~~
> LB19 Do not break after non-final unresolved quotation marks such as “‘” or “'”, nor before non-initial unresolved quotation marks such as ‘”’ or “'”.
> ~~× QU~~
> × [QU-\p{Pi}]
> ~~QU ×~~
> [QU-\p{Pf}] ×
> LB19a Unless surrounded by East Asian Characters, do not break either side of any unresolved quotation marks.
> [^\p{ea=F}\p{ea=W}\p{ea=H}] × QU
> × QU ([^\p{ea=F}\p{ea=W}\p{ea=H}] | eot)
> QU × [^\p{ea=F}\p{ea=W}\p{ea=H}]
> (sot | [^\p{ea=F}\p{ea=W}\p{ea=H}]) QU ×

Background information / discussion

Effect

Examples of the added break opportunities—unchanged break opportunities unmarked—:

1. From the zh-hans version of the *Analects* on Wikisource:
   - 子曰：÷“学而时习之，不亦说乎？有朋自远方来，不亦乐乎？人不知而不愠，不亦君子乎？”
   - 子贡曰：÷“贫而无谄，富而无骄，何如？”÷子曰：÷“可也。未若贫而乐，富而好礼者也”。子贡曰：÷“《诗》云：÷‘如切如磋，如琢如磨。’÷其斯之谓与？”÷子曰：÷“赐也，始可与言《诗》已矣！告诸往而知来者。”
2. From the March 2024 recent additions zh-CN Wikipedia:
   - 哪一所中国学校乃÷“为各省派往日本游学之首倡”？
   - 哪个商标以人名为名，因特色小吃÷“五台杂烩汤”÷而入选÷“新疆老字号”？
3. From the March 2024 featured articles on zh-CN Wikipedia:
   - 毕士悌（1901年—1936年）又名÷“杨林”，朝鲜籍红军将领

The heuristic of checking for the width on both sides of the quotation mark will sometimes fail to detect an East Asian context. It is therefore still useful to tailor the line breaking class of the quotation marks based on language if possible. For instance, in this text from the same list of featured articles, the algorithm will fail to find break opportunities outside the quotation marks surrounding “Best Game Boy Strategy”, while it will find break opportunities outside《IGN》just fine,《》being OP and CL:

- 2000年获得了《IGN》的“Best Game Boy Strategy”奖。

This can also happen when the quoted text is East Asian, as in the following example (*ibidem*), where the change to the line breaking algorithm finds the break opportunity after the quoted name, but not before:

- Z-1“莱贝雷希特·马斯”÷号是德国国家海军暨战争海军于1930年代

However, the heuristic should not err in the other direction, and therefore should not introduce unwanted breaks in non-East Asian typesetting (where initial punctuation can be used finally and final punctuation initially, as in German or Swedish). For instance, in this note from a table in the German Wikipedia article on Hong Kong, no break opportunity is introduced within ，白 nor within 人‘.

- Anmerkung: „White“ bzw. ‚白人‘ – in der Amtlichen Statistik

Implementation considerations

The rules are implementable in ICU; in order to avoid chaining over multiple code points with LB15a and LB15b, it is easiest to implement them as break rules overriding the existing LB19, as follows:

```
Unset
$LB18NonBreaks $CM* $QU;
^$CM+               $QU;

[$LB18NonBreaks & [\p{ea=F}\p{ea=W}\p{ea=H}] - [$OP $GL $BA]]           / [\p{Pi} & $QU] $CM*
[ [\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
[$LB18NonBreaks & [\p{ea=F}\p{ea=W}\p{ea=H}] - [$OP $GL $BA]] $CM* $CMX / [\p{Pi} & $QU] $CM*
[ [\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
^[$CM & [\p{ea=F}\p{ea=W}\p{ea=H}]]                    / [\p{Pi} & $QU] $CM* [
[\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
^[$CM & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* $CMX          / [\p{Pi} & $QU] $CM* [
[\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];

$QU $CM* .;

[$LB18NonBreaks & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* [\p{Pf} & $QU]           / [
[\p{ea=F}\p{ea=W}\p{ea=H}] - [$NS $BA $EX $CL $IN $IS $GL $CM]];
[$LB18NonBreaks & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* [\p{Pf} & $QU] $CM* $CMX / [
[\p{ea=F}\p{ea=W}\p{ea=H}] - [$NS $BA $EX $CL $IN $IS $GL $CM]];
^[$CM & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* [\p{Pf} & $QU]           / [
[\p{ea=F}\p{ea=W}\p{ea=H}] - [$NS $BA $EX $CL $IN $IS $GL $CM]];
^[$CM & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* [\p{Pf} & $QU] $CM* $CMX / [
[\p{ea=F}\p{ea=W}\p{ea=H}] - [$NS $BA $EX $CL $IN $IS $GL $CM]];
```

Additional interactions with LB15c/LB15d (proposed in #247) and LB21a require the exclusion of $BA above, and the addition of the following rules:

```
Unset
$SP [$IS & [\p{ea=F}\p{ea=W}\p{ea=H}]]           / [$QU & \p{Pi}] $CM* [
[\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
$SP [$IS & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* $CMX / [$QU & \p{Pi}] $CM* [
[\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
```

```
$SP [$IS & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* [$QU & \p{Pf}]           / [
[\p{ea=F}\p{ea=W}\p{ea=H}] - [$NS $BA $EX $CL $IN $IS $GL $CM]];
$SP [$IS & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* [$QU & \p{Pf}] $CM* $CMX / [
[\p{ea=F}\p{ea=W}\p{ea=H}] - [$NS $BA $EX $CL $IN $IS $GL $CM]];

^$CM*           [$BA & [\p{ea=F}\p{ea=W}\p{ea=H}]]           / [\p{Pi} & $QU] $CM* [
[\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
^$CM*           [$BA & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* $CMX / [\p{Pi} & $QU] $CM* [
[\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];

[$LB20NonBreaks - $HL] $CM* [$BA & [\p{ea=F}\p{ea=W}\p{ea=H}]]           / [\p{Pi} & $QU]
$CM* [ [\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
[$LB20NonBreaks - $HL] $CM* [$BA & [\p{ea=F}\p{ea=W}\p{ea=H}]] $CM* $CMX / [\p{Pi} & $QU]
$CM* [ [\p{ea=F}\p{ea=W}\p{ea=H}] - $CM];
```

Equivalence of the UAX #14 and ICU formulations has been tested by a host of monkeys.

5.6 Line breaking of combining left half marks and continuous lining marks

Recommended UTC actions

1. **Consensus**: Change the Line_Break property of nine characters from Line_Break=Combining_Mark to Line_Break=Glue: U+FE20, U+FE22, U+FE24, U+FE27, U+FE29, U+FE2B, and U+FE2E (seven combining left half marks), as well as U+FE26 COMBINING CONJOINING MACRON and U+FE2D COMBINING CONJOINING MACRON BELOW. For Unicode Version 16.0. See L2/24-064 item 5.6.
2. **Action Item** for [person], PAG: In LineBreak.txt and derived files, change the Line_Break property of the following nine characters from Line_Break=Combining_Mark to Line_Break=Glue: U+FE20, U+FE22, U+FE24, U+FE27, U+FE29, U+FE2B, U+FE2E, U+FE26, and U+FE2D. For Unicode Version 16.0. See L2/24-064 item 5.6.

Feedback

From Robin Leroy, PAG

A glance at Standing Document 2 shows that I have inherited the following action item:

Action	Owners	Description	Documents	Status
UTC-151-A90	Robin Leroy, PAG	Investigate linebreaking properties of double diacritics, based on feedback from David Corbett on PRI #335 of April 29, 2017 22:48:11.		

Looking at PRI-335, the relevant feedback is

```
Unset
Date/Time: Sat Apr 29 22:48:11 CDT 2017
Name: David Corbett
Report Type: Public Review Issue
Opt Subject: PRI #335: More double diacritics

Just as U+035C..0362 have line break class GL, so should the other double diacritics
U+1DCD and U+1DFC. Similarly, it may make sense for the left and conjoining half marks
U+FE20, U+FE22, U+FE24, U+FE26..FE27, U+FE29, U+FE2B, and U+FE2D..FE2E to have line
break class GL.
```

U+1DCD and U+1DFC are lb=GL since Unicode Version 15.0. The reason is decision UTC-172-A60, based on feedback in PRI-453 from… David Corbett.

Making the left half marks lb=GL would probably be sensible, so that the less-preferred representation of double diacritics would behave like the preferred one; see https://www.unicode.org/versions/latest/ch07.pdf#G27052.

Making the conjoining macron and macron below lb=GL also seems reasonable, see https://www.unicode.org/versions/latest/ch07.pdf#G24556.

That briefly raises the question of the other combining continuous lining marks (see Table 4-10, https://www.unicode.org/versions/latest/ch04.pdf#G148813), but since they have no terminator they clearly cannot be made lb=GL.

5.7 On the Line_Break assignment of three vertical presentation forms

Recommended UTC actions

1. **Consensus**: Change the Line_Break assignment of U+FE10 ﹐PRESENTATION FORM FOR VERTICAL COMMA to Close_Punctuation (CL), and that of U+FE13 ﹓PRESENTATION FORM FOR VERTICAL COLON and U+FE14 ﹔PRESENTATION FORM FOR VERTICAL SEMICOLON to Nonstarter (NS), to match their FULLWIDTH counterparts U+FF0C, U+FF1A, and U+FF1B. For Unicode Version 16.0. See L2/24-064 item 5.7.
2. **Action Item** for Robin Leroy, PAG: In LineBreak.txt and derived files, Change the Line_Break assignment of U+FE10 ﹐PRESENTATION FORM FOR VERTICAL COMMA to Close_Punctuation (CL), and that of U+FE13 ﹓PRESENTATION FORM FOR VERTICAL COLON and U+FE14 ﹔PRESENTATION FORM FOR VERTICAL SEMICOLON to Nonstarter (NS). For Unicode Version 16.0. See L2/24-064 item 5.7.
3. **Action Item** for Robin Leroy, PAG: In UAX #14, update the descriptions of line breaking classes IS, CL, and NS to reflect the change of Line_Break assignment of U+FE10, U+FE13, and U+FE14. For Unicode Version 16.0. See L2/24-064 item 5.7.

PAG input

From Robin Leroy, PAG.

While working on #197, it came to my attention that we have three characters that have lb=IS and ea=W:
U+FE10 ︐ PRESENTATION FORM FOR VERTICAL COMMA
U+FE13 ︓ PRESENTATION FORM FOR VERTICAL COLON
U+FE14 ︔ PRESENTATION FORM FOR VERTICAL SEMICOLON

These have been given lb=IS because their ASCII counterparts are lb=IS; but they are not presentation forms for `,`, `:`, and `;`, as these are ea=Na and vo=R; instead they are presentation forms for the fullwidth U+FF0C comma, U+FF1A colon, and U+FF1B semicolon. See also table 2 of UAX #50: https://unicode.org/reports/tr50/#table_2.

They should therefore have lb=CL for U+FE10, and lb=NS for U+FE13 & U+FE14.

(No-one had noticed previously because these are presentation forms, so they should hopefully not be used all over the place, and because they lb=IS and lb=CL or NS do not differ in relevant ways where you would expect to see these characters; I only spotted that because of extremely obscure interactions between line breaking rules in the optimized ICU implementation.)

Background information / discussion

I looked at the rest of \p{dt=vertical}, and the other line break assignments seem fine: https://util.unicode.org/UnicodeJsps/list-unicodeset.jsp?a=%5B%3Adt%3Dvertical%3A%5D&g=lb&i=.

Ken Lunde was consulted and said this makes complete sense.

5.8 On the exclusion of CCC=0 from InCB=Extend

Recommended UTC actions

1. **Consensus**: Change the derivation of InCB=Extend to [\p{gcb=Extend}\p{gcb=ZWJ}-\p{Indic_Syllabic_Category=Virama}-\p{Indic_Syllabic_Category=Consonant}-[\u200c]]. For Unicode Version 16.0. See L2/24-064 item 5.8.
2. **Action Item for** Robin Leroy, PAG: Change the derivation of InCB=Extend to [\p{gcb=Extend}\p{gcb=ZWJ}-\p{Indic_Syllabic_Category=Virama}-\p{Indic_Syllabic_Category=Consonant}-[\u200c]], and update DerivedCoreProperties.txt accordingly. For Unicode Version 16.0. See L2/24-064 item 5.8.
3. **Action Item for** Ken Whistler, PAG: In Unicode Standard Annex #44, Unicode Character Database, change the derivation of InCB=Extend to [\p{gcb=Extend}\p{gcb=ZWJ}-\p{Indic_Syllabic_Category=Virama}-\p{Indic_Syllabic_Category=Consonant}-[\u200c]]. For Unicode Version 16.0. See L2/24-064 item 5.8.

PAG input

From @eggrobin, PAG, after discussion with @NorbertLindenberg on a SEW issue.

Background information / discussion

Feedback item https://www.unicode.org/review/pri469/feedback.html#ID20230620135108 points out that rule GB9c does not correctly handle Gujarati clusters with a nukta.

The exclusion of CCC=0 in the derivation of InCB=Extend is to blame here, and it is not necessary for the rule to be consistent with canonical equivalence.

My hypothesis for how we ended up with this exclusion is that the proposed rule in L2/18-147 was constructed as follows:

1. `Virama × LinkingConsonant`
2. But wait, that is not consistent with canonical equivalence, because sequence Extend Virama LinkingConsonant could be reordered to Virama Extend LinkingConsonant if the Extend and Virama have nonzero CCC!
3. Alright, so we can have an Extend with nonzero CCC in there, thus `Virama [Extend-\p{ccc=0}]* × LinkingConsonant`.

That is overly specific: the difference between Virama [Extend-\p{ccc=0}]* × LinkingConsonant and Virama Extend* × LinkingConsonant is mostly degenerate. At this point however, there is no real problem.

Then the rule got adapted for reasons yet unclear as part of implementation in ICU into LinkingConsonant [Extend-\p{ccc=0}]* Virama [Extend-\p{ccc=0}]* × LinkingConsonant

and now we are forbidding those Gujarati nuktas.

While getting rid of the leading `\p{InCB=Consonant} [ \p{InCB=Extend} \p{InCB=Linker} ]*`, if possible and if needed, would at this point cause churn for implementers, the fact that we have InCB means we can drop the `-\p{ccc=0}` from the derivation of InCB=Extend, thereby fixing the Gujarati situation. We still need to exclude ZWNJ, which is Extend.

As of Unicode 15.1, all of these 1225 characters will go from InCB=None to InCB=Extend.

They are, by and large, vowels. (The set also includes 256 variation selectors and 96 plane 14 tag characters.)

Alternatively: we could just include those three nuktas (or include InSc=Nukta).

The current definition of `InCB` is

```
Unset
InCB = Linker iff C in [S &\p{Indic_Syllabic_Category=Virama}]

InCB = Consonant iff C in [S &\p{Indic_Syllabic_Category=Consonant}]

InCB = Extend iff C in
[[\p{gcb=Extend}-\p{ccc=0}]
\p{gcb=ZWJ}
-\p{Indic_Syllabic_Category=Virama}
-\p{Indic_Syllabic_Category=Consonant}]

Otherwise, InCB = None (the default value)
```

5.9 Proposal to add a property for auto inter-script spacing

Recommended UTC actions

1. **Note**: PAG has assembled a group of experts that are discussing approaches for auto spacing in East Asian text based on L2/24-057.

Source

L2/24-057 Proposal of the UNICODE AUTO SPACING (supersedes L2/23-283) by Koji Ishii, Yasuo Kida

Summary

This is a proposal to add a property for auto spacing. Initially this property supports inserting inter-script spacing for CJK typography, but it may extend to support other scripts in future, such as French Typographical Rules for Punctuation.

5.10 Remove ancient UAX #14 labels

Recommended UTC actions

1. **Action Item** for Robin Leroy, PAG: In UAX #14, remove the parenthetical labels (A), (B), (P), (XA), (XB), (XP) from the the descriptions of line breaking classes. Consider replacing them with links to relevant line breaking rules. For Unicode Version 16.0. See L2/24-064 item 5.10.

PAG input

From Robin Leroy, PAG.

Should the editor of the line breaking algorithm be so careless as to look at Standing Document 2, he will find the following action item:

UTC-162-A45 Action Item for Steven Loomis, Editorial Committee: Provide texts for UAX #14 based on comments in PRI #406 [Wed Dec 4 15:18:53 CST 2019], for Unicode version 14.0.

Seeing as this action item is unlikely to get done in time for Unicode 14.0, or even for 15.0 to which it has been retargeted in SD-2, that editor will then feel compelled to act upon it.

The feedback from PRI-406 is as follows:

```
Unset
Date/Time: Wed Dec 4 15:18:53 CST 2019
Name: Markus W Scherer
Report Type: Error Report
Opt Subject: UAX #14 line break: confusing naming/behavior of lb=BA eg U+3000
```

```
Feedback on behalf of Javier Fernandez & Florian Rivoal, see
https://unicode-org.atlassian.net/browse/ICU-20843

The last comment there (from Florian) is:

Even though he BA class is described as as “(A)” “providing a break
opportunity” in section 5, if you follow the rules of section 6, you’re
right that the effect is to suppress breaks before, not to introduce them
after. So my comment above was wrong, and there should not be a break
between ID and U+3000, nor between CJ (treated as ID or as NS) and U+3000.
Which means that ICU does not have a bug after all. Sorry for the confusion.

I do wonder if an editorial bug should be open on UAX 14: the informative
text of section 5 in this case is a poor indicator of the normative behavior
of section 6, leading to misunderstanding like the mistake I made above.
```

While the general qualitative description of the line breaking behaviour in words is useful (and indeed lb=BA characters generally tend to allow a break afterwards and none before), the classification in (A), (XA), (B), (XB), (P), and (XP), which has not meaningfully changed since Unicode Version 3.0.0, has long outlived its usefulness.

This classification predates the ancestor of the modern rules, which was a translation of Michel’s pair table: it may be found in https://www.unicode.org/reports/tr14-4/ (the rules first appeared in revision 5). Back then, it was a partial classification of the behaviour of the associated line breaking classes, with additional details, such as the interaction with spaces, provided in the text; effectively, they were the rules.

However, now that there are separate rules, and given the complexity of the current line breaking algorithm, this simplistic classification just leads to confusion; the valiant attempt by Norbert at classifying the new classes VI (XB/XA), VF (XB/A), AK (XB/XA), AP (B/XA), and AS (XB/XA), derived from the rules in which they appear, utterly fails to convey that AK and AS tend to allow breaks before and after, like ideographic characters, unless viramas intervene.

We should get rid of this. One possibly useful replacement would be to list the rules wherein a class is mentioned, thus VI, VF, AK, AP, and AS could link back to LB28a.

5.11 UAX #29: Request to add pointer to Table 1c from note after rule GB999

Recommended UTC actions

1. **No Action**: PAG recommends no action: this issue has been addressed editorially.

Feedback

From email to Josh Hadley from Martin J. Dürst:

I'm working on updating the grapheme cluster implementation in the programming language Ruby. In that context, I have a comment on Unicode® Standard Annex # 29, Unicode Text Segmentation.

The problem is with the first note after rule GB999, which reads: "Grapheme cluster boundaries can be transformed into simple regular expressions. For more information, see Section 6.3, State Machines.".

This is not wrong, but it misleads the reader, because there is already a regex definition for grapheme clusters in Table 1c.

A regex for grapheme clusters and a regex for grapheme cluster boundaries are strictly speaking not the same. But they are very tightly linked; if you use the former to find the grapheme clusters in a text, you should have automatically found the grapheme cluster boundaries, too (or something is really fundamentally wrong with the definitions).

So a pointer from the note mentioned above to Table 1c would be highly appreciated. It would certainly have saved me and others quite some time for the present update.

5.12 UAX #29 remove colons from word break MidLetter

Recommended UTC actions

1. **No Action**: PAG recommends no action. CLDR 46/ICU 76 should have UAX #29 colon behavior again, with the default word breaking treating colon as a MidLetter.

Input from CLDR & ICU

Since CLDR 42 / ICU 72 (2022-oct), their root word break implementations have excluded colons (`[\: \uFE55 \uFF1A]`) from the MidLetter class. The default UAX #29 behavior has been moved to tailorings for just Finnish and Swedish. See CLDR-15910 for background. This was implemented in ICU-22112 https://github.com/unicode-org/icu/pull/2159 (together with a change in `@` behavior which has since been retracted).

Please consider removing colons from UAX #29 WB MidLetter, realigning UAX #29 with CLDR/ICU.

Background information / discussion

PAG and CLDR/ICU discussed this separately and then jointly. We agreed to revert the CLDR/ICU colon behavior to sync with UAX #29 again by default (root locale). There will be a "profile" for technical usage along the lines of the current CLDR/ICU behavior. This is planned for the fall 2024 releases (CLDR 46 & ICU 76). The Swedish and Finnish tailorings will like remain and override the technical profile. (A tailoring for German may be added.)

5.13 Upstream the ICU/CLDR hyphen line breaking tailoring

Recommended UTC actions

1. **Consensus**: In Unicode Standard Annex #14, Unicode Line Breaking Algorithm, add a rule LB20a as described in L2/24-064 item 5.13. For Unicode Version 16.0. See L2/24-064 item 5.13L2/24-064 item 5.13.
2. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, Unicode Line Breaking Algorithm, add rule LB20a as described in L2/24-064 item 5.13. For Unicode Version 16.0. See L2/24-064 item 5.13.
3. **Action Item** for Robin Leroy, PAG: In LineBreakTest.txt, add rule LB20a as described in L2/24-064 item 5.13, and add realistic handwritten test cases. For Unicode Version 16.0. See L2/24-064 item 5.13.

PAG input

From Andy Heninger and Robin Leroy, PAG.

ICU and CLDR have a tailoring to the line breaking algorithm which originated as a Finnish tailoring in 2010 (CLDR-3029), and was upstreamed to the (language-independent) root in 2018 (ICU-8151).

The goal of the tailoring is to avoid breaks after hyphens that are effectively word-initial—in the sense of space-separated words—; the example given in the CLDR ticket was "Mac Pro -tietokone" (Mac Pro computer).

This tailoring is implemented as follows in ICU rules:

```
Unset
# LB 20.09    Don't break between Hyphens and Letters when there is a break preceding the
hyphen.
#             Originally added as a Finnish tailoring, now promoted to default ICU behavior.
#             Note: this is not default UAX-14 behaviour. See issue ICU-8151.
#
^($HY | $HH) $CM* $ALPlus;
```

where HH is U+2010.

The ICU rules can refer to a preceding break for context; the UAX #14 rules cannot. A translation of the above rules into UAX #14 rules yields something like this:
Before LB9:

```
Unset
CB (CM | ZWJ)* ZWJ ( HY | [\u2010] ) ÷ AL
```

After LB20:

```
Unset
OP SP* ( HY | [\u2010] ) ÷ AL
(sot | BK | CR | LF | NL | OP | QU | GL | SP | ZW) [\p{Pi}&QU] ( HY | [\u2010] ) SP* ÷ AL
(sot | BK | CR | LF | NL | SP | ZW | CB) ( HY | [\u2010] ) × AL
```

Where sot | BK | CR | LF | NL | SP | ZW | CB are generally followed by breaks, and the earlier rules cover cases where they are not actually breaks.

These earlier rules are strange and complex, and they only serve to cause unexpected effects: « the 3ms possessive pronominal suffix (-šu) » ends up having a break before šu, whereas « the 3ms possessive pronominal suffix -šu » doesn't. They are an artifact of reaching for the technically simplest expression in the ICU framework, and should not be included in UAX #14.

Further, in order to avoid the strange situation where turning a space into a no-break space creates break opportunities, the context should match GL as well as SP; this is what we do in the lists of alternatives in LB15a and LB15b.

The proposed rule is therefore:

> LB20a Do not break after a hyphen that follows break opportunity, a space, or the start of text.
> (sot | BK | CR | LF | NL | SP | ZW | CB | GL) (HY | [\u2010]) × AL

Background information / discussion

The changes to simplify the UAX #14-style rules and make the rule less surprising are implementable in ICU; because of interactions with other rules, the updated ICU-style rules become:

```
Unset
^($HY | $HH) $CM* $ALPlus;
$GL ($HY | $HH) $CM* $ALPlus;
# Non-breaking CB from LB8a:
$CB $CM* $ZWJ ($HY | $HH) $CM* $ALPlus;
# Non-breaking SP from LB14:
$OP $CM* $SP+ ($HY | $HH) $CM* $ALPlus;
# Non-breaking SP from LB15a:
($OP $CM* $SP+ | [$OP $QU $GL] $CM*) ([\p{Pi} & $QU] $CM* $SP*)+ $SP ($HY | $HH) $CM*
$ALPlus;
^([\p{Pi} & $QU] $CM* $SP*)+ $SP ($HY | $HH) $CM* $ALPlus;
# Non-breaking SP from LB15a following LB15b:
$LB8NonBreaks [\p{Pf} & $QU] $CM* ([\p{Pi} & $QU] $CM* $SP*)+ $SP ($HY | $HH) $CM* $ALPlus;
$CAN_CM $CM*  [\p{Pf} & $QU] $CM* ([\p{Pi} & $QU] $CM* $SP*)+ $SP ($HY | $HH) $CM* $ALPlus;
^$CM+  [\p{Pf} & $QU] $CM* ([\p{Pi} & $QU] $CM* $SP*)+ $SP ($HY | $HH) $CM* $ALPlus;
```

The equivalence of these rules has been extensively tested ("by innumerable monkeys").

5.14 recommendations for Apparent Sentence_Break miscategorizations

Recommended UTC actions

1. **Consensus**: Update Table 4, Sentence_Break Property Values of UAX #29, categorizing Semicolons and Greek Question Mark as SContinue, U+2CF9..U+2CFB as STerm, and Vertical Forms punctuation (U+FE10..U+FE19) in the same categories as their compatibility equivalents. For Unicode 16.0. See L2/24-064 item 5.14.
2. **Action Item** for Josh Hadley, PAG: Update Table 4, Sentence_Break Property Values of UAX #29, categorizing Semicolons and Greek Question Mark as SContinue, U+2CF9..U+2CFB as STerm, and Vertical Forms punctuation (U+FE10..U+FE19) in the same categories as their compatibility equivalents. For Unicode 16.0. See L2/24-064 item 5.14.
3. **Consensus**: Update SentenceBreakProperty.txt, categorizing Semicolons and Greek Question Mark as SContinue, U+2CF9..U+2CFB as STerm, and Vertical Forms punctuation (U+FE10..U+FE19) in the same categories as their compatibility equivalents. For Unicode 16.0. See L2/24-064 item 5.14.
4. **Action Item** for Josh Hadley, PAG: Update SentenceBreakProperty.txt, categorizing Semicolons and Greek Question Mark as SContinue, U+2CF9..U+2CFB as STerm, and Vertical Forms punctuation (U+FE10..U+FE19) in the same categories as their compatibility equivalents. For Unicode 16.0. See L2/24-064 item 5.14.
5. **Consensus**: Give U+2FC9..U+2CFB the Sentence_Terminal property. For Unicode 16.0. See L2/24-064 item 5.14.
6. **Action Item** for Josh Hadley, PAG: In PropList.txt, add U+2CF9..U+2CFB to Sentence_Terminal. For Unicode 16.0. See L2/24-064 item 5.14.

PAG input

From Markus Scherer: We had an action item
[155-A84] Action Item for Andy Heninger, Mark Davis, Deborah Anderson: Investigate PRI-372 feedback from fantasai on "Apparent Sentence_Break miscategorizations" (Thu Mar 8 02:36:01 CST 2018) and make suggestions for Unicode 12.0, for UTC-157, September 17-21, 2018.

which was done, and resulted in L2/19-294 “Apparent Sentence_Break miscategorizations”

It looks like we have not yet made spec or data changes for that.

5.15 unexpected line break within abc.123

Recommended UTC actions

1. **Consensus**: In Unicode Standard Annex #14, Unicode Line Breaking Algorithm, Change rules LB13 and LB25, and add rules LB15c and LB15d, and make changes to the regular expression for numbers to the examples, as described in L2/24-064 item 5.15. For Unicode Version 16.0. See L2/24-064 item 5.15.
2. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, Unicode Line Breaking Algorithm, Change rules LB13 and LB25, and add rules LB15c and LB15d, and make changes to the regular expression for numbers to the examples, as described in L2/24-064 item 5.15. Add realistic examples to LB15c in UAX #14. For Unicode Version 16.0. See L2/24-064 item 5.15.
3. **Action Item** for Robin Leroy, PAG: In LineBreakTest.txt, Change rules LB13 and LB25, and add rules LB15c and LB15d, and make changes to the regular expression for numbers to the examples, as described in L2/24-064 item 5.15. Add test cases that exercise the rule changes. For Unicode Version 16.0. See L2/24-064 item 5.15.

Document

L2/24-042 by Stanisław Hodur

“While “abc.abc”, “123.123” and “123.abc” are all unbreakable, yet “abc.123” allows break after the dot.

In common language, only 123.123 is used (with the decimal dot in some languages and for dates). Anything like abc.123, 123.abc or abc.abc can be used as identifiers, like Internet address (unicoder.org, 64.182.27.164) or document clause (II.1.A of 2006/42/EC). Breaking inside such an identifier is annoying and can be misleading.”

Background information / discussion

The conformance tests currently published for UAX #14 test a tailored version of the algorithm which avoids breaks within numbers of the form

> (PR | PO) ? (OP | HY) ? NU (NU | SY | IS) * (CL | CP) ? (PR | PO) ?
> The feedback at hand pertains to that tailoring.
> That tailoring is implemented by ICU (and in practice, by anyone who implements the algorithm, since the default is untestable), and ICU further tailors it as follows:
> (PR | PO) ? (OP | HY) ? IS? NU (NU | SY | IS) * (CL | CP) ? (PR | PO) ?

This additional tailoring fixes the issue.

In addition, ICU allows for a break between SP and IS if the IS immediately precedes a number (yielding the proposed LB15c and LB15d), thus allowing a break before a number like .5.

Having conformance tests for a non-default algorithm is a mess; and this non-default has issues which have been fixed in ICU for five years. We should collapse all of that up into the default; we have given up on limiting the default to a pair table since Unicode 9.

Some intermediate steps

So, the exercise is to translate

> Do not break within (PR | PO)? (OP | HY)? **IS?** NU (NU | SY | IS)* (CL | CP)? (PR | PO)?
> (emphasis added on the ICU addition to Example 7) into UAX #14 rules.

First round:

1. (PR | PO) × (OP | HY)? IS? NU
2. (OP | HY) × IS? NU
3. IS × NU
4. NU × (NU | SY | IS)
5. NU (SY | IS)* × (NU | SY | IS | CL | CP)
6. NU (SY | IS)* (CL | CP)? × (PR|PO)

Second round, because the old style monkeys, being quite primitive, prefer simpler regices:

1. —
   a. (PR | PO) × NU
   b. (PR | PO) × (OP | HY | IS) NU
   c. (PR | PO) × (OP | HY) IS NU
2. —
   a. (OP | HY) × NU
   b. (OP | HY) × IS NU
3. IS × NU
4. NU × (NU | SY | IS)
5. NU (SY | IS)* × (NU | SY | IS | CL | CP)
6. —
   a. NU (SY | IS)* × (PR|PO)
   b. NU (SY | IS)* (CL | CP) × (PR|PO)

See https://github.com/eggrobin/icu/commit/179aa25f15413b476c1e341ae2485c14cf0fdf55. A million monkeys were sent to investigate and found no inconsistency between this rule and the ICU behaviour.

The attentive reader will have noticed that this encompasses many rules that are already part of LB25, including IS × NU at hand in 3., but also NU × NU as part of 4., PR × NU in 1., NU × PR in 6., etc. Further, it has plenty of redundant rules or parts of rules, such as (OP | HY) × IS NU (done by × IS in LB13), NU (SY | IS)* × (CL | CP) (done by × CL and × CP ibidem), etc.

Proposal

The proposed changes to the rules are as follows, tested with monkeys beyond counting:
Change LB13:

- × CL
- × CP
- × EX
- × IS
- × SY

Add after LB15b:

- LB15c SP ÷ IS NU
- LB15d × IS

Change LB25:

- NU (SY | IS)* CL × PO
- NU (SY | IS)* CP × PO
- NU (SY | IS)* CL × PR
- NU (SY | IS)* CP × PR
- NU (SY | IS)* × PO
- NU (SY | IS)* × PR
- PO × OP NU
- PO × OP IS NU
- PO × NU
- PR × OP NU
- PR × OP IS NU
- PR × NU
- HY × NU
- IS × NU
- NU (SY | IS)* × NU
- NU × SY
- NU (SY | IS)* × SY

This incorporates L2/19-013, plus hoisting the tailoring into the real LB 25.

In addition the big regex above LB25 should be changed as follows:
(PR | PO) ? (OP | HY) ? IS? NU (NU | SY | IS) * (CL | CP) ? (PR | PO) ?

And the following paragraph

> The default line breaking algorithm approximates implements this with the following rule. Note that some cases have already been handled, such as ‘9,’, ‘[9’. For a tailoring that supports the regular expression directly, as well as a key to the notation see Section 8.2, Examples of Customization.

The tailoring in question needs to go, and the examples under LB25 can probably be removed too.

Additional examples should be added for the regex.

6. IDNA

6.1 Proposals to modify CheckBidi handling (UTS #46)

Recommended UTC actions

1. **Action Item** for Markus Scherer, Mark Davis, PAG: For UTS #46: Review the WhatWG request from Feb 13 07:41:09 CST 2023 for relaxing the CheckBidi validity criterion and recommend a disposition. For Unicode 17.0. See L2/24-064 item 6.1.

Feedback (verbatim)

Date/Time: Mon Feb 13 07:41:09 CST 2023
Name: Anne van Kesteren
Report Type: Error Report
Opt Subject: UTS 46

An issue reported against the URL Standard indicated that the current CheckBidi handling from UTS 46 is rather strict: https://github.com/whatwg/url/issues/543. Namely, domains containing RTL-labels cannot have labels consisting solely of ASCII digits preceding them (such labels are invalid per The Bidi Rule subrule 1). This ends up rejecting a number of domains in the wild and also seems unnecessarily restrictive for RTL users.

In that issue I worked with Harald Alvestrand (one of the editors of RFC 5893: Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)) on a specific set of changes for UTS 46 that would remedy this issue, while still imposing the majority of Bidi-related requirements present in UTS 46 today.

The proposed changes are:

1. Remove step 8 of https://unicode.org/reports/tr46/#Validity_Criteria as Validity Criteria only operates on a single label. (Although it somehow claims to have knowledge about the domain_name string as well...)
2. Add a new step 5 to https://unicode.org/reports/tr46/#Processing. (Note that due to step 4 we will have U-labels.)

The new step 5 would as follows:

- If CheckBidi, and the domain_name string is a Bidi domain name, record there was an error if neither of the following conditions is true:
    - All labels in the domain_name string satisfy the 6 subrules of The Bidi Rule of RFC 5893, Section 2.
    - RTL labels in the domain_name string are immediately followed by an LDH label whose first code point is not of class EN and all labels in the domain_name string are either LDH labels or satisfy the 6 subrules of The Bidi Rule of RFC 5893, Section 2.

Thank you for your consideration. This is probably the final IDNA-related issue from the URL Standard. Once all of them have been resolved I'll work with browser implementers to ensure the changes (if any) get implemented so we can finally declare victory on IDNA interoperability.

Background information / discussion

Markus: The problem description sounds reasonable, and just at a glance the proposed changes sound plausible. I have not tried to think through all of the effects yet. Needs more time & discussion.

(We would remove item 9, not item 8 of the Validity Criteria: In 15.1 we inserted item 4 about `xn--` so the following items got renumbered.)

Mark: This looks like an improvement, but would differ from the IDNA2008 spec. Consider adding a flag and making this conditional.

6.2 UTS #46 5 Han characters allowed directly but not in Punycode

Recommended UTC actions

1. **Consensus**: In UTS #46, the base exclusion set has been derived from differences between IDNA2003 data and UTS #46 principles. Replace this base exclusion set with a smaller set. This will change the IdnaMappingTable.txt Status for some characters from disallowed to ignored/mapped/valid. For details see the proposal in L2/24-064 item 6.2. For Unicode 16.0.
2. **Action Item** for Markus Scherer, PAG: In the UTS #46 spec, the base exclusion set has been derived from differences between IDNA2003 data and UTS #46 principles. Replace this base exclusion set with a smaller set. For details see the proposal in L2/24-064 item 6.2. For Unicode 16.0.
3. **Action Item** for Markus Scherer, PAG: In the UTS #46 tools code, the base exclusion set has been derived from differences between IDNA2003 data and UTS #46 principles. Replace this base exclusion set with a smaller set. This will change the IdnaMappingTable.txt Status for some characters from disallowed to ignored/mapped/valid. For details see the proposal in L2/24-064 item 6.2. For Unicode 16.0.

Feedback (verbatim)

Date/Time: Mon Feb 05 04:40:33 CST 2024
ReportID: ID20240205044033
Name: Henri Sivonen
Report Type: Error Report
Opt Subject: UTS 46

UTS #46 section 4 Processing step 1. Map says of "disallowed" characters: "Leave the code point unchanged in the string. Note: The Convert/Validate step below checks for disallowed characters, after mapping and normalization."

Step 2. Normalize then turns compatibility ideographs into their singleton decompositions.

Therefore, if a label does not start with "xn--", the characters U+2F868, U+2F874, U+2F91F, U+2F95F, and U+2F9BF will no longer occur in the domain after Step 2. Normalize.

However, if the label starts with "xn--" and Punycode decoding yields U+2F868, U+2F874, U+2F91F, U+2F95F, or U+2F9BF, these characters fail section 4.1 Validity Criteria item 7., which denies characters that don't have the status "valid" or for Nontransitional Processing "deviation".

It seems to me that these five characters are the only ones for which it makes a difference not to do "disallowed" processing before normalization already in 4 Processing step 1. Map.

ICU4C's UTS 46 normalization yields the REPLACEMENT CHARACTER for these five characters, which is logically equivalent to performing "disallowed" processing already in 4 Processing step 1. Map.

Please call out the implications for these five characters explicitly so that it's clear to the reader whether the intent is that these characters are only prohibited in already-Punycode-form labels but not in Unicode-form labels (the outcome of the current spec text) or whether the intent is for these characters to be prohibited both in already-Punycode-form labels and in Unicode-form labels (as suggested by ICU4C's UTS 46 normalization).

Furthermore, today it appears to be the case that all "disallowed" characters either decompose to themselves or have a singleton decomposition in NFD. It would be useful to have a remark about whether this is a guarantee that is expected to hold for future versions.

Background information / discussion

These five CJK compatibility ideographs are the only characters which are "disallowed" in UTS #46 but are mapped by NFC to "valid" characters. Henri is correct in that they should be mapped in the "Normalize" step when they occur directly in a label. Handling them correctly in an optimized implementation like ICU's, which fuses the "Map" and "Normalize" steps, would require special code or data.

This problem with these 5 Han characters, and with the ICU implementation, is new since Unicode 15.1. The "Map" step used to immediately report an error for disallowed characters, but we removed that because it caused other inconsistencies. See the UTS #46 15.1 proposed update and the UTS #46 15.1 Modifications.

Almost all of the other CJK compatibility ideographs (except for the twelve that do not decompose) have a Status value of "mapped", and their decomposition mapping is applied directly in the UTS #46 "Map" step. All of these CJK compatibility ideographs, including the special five, are disallowed when they occur in a Punycode-encoded label.

The five are inconsistent with the others because of UTS #46 section 6 Mapping Table Derivation / Step 3: Specify the base exclusion set.

- This part of the data derivation disallows characters which would normally be "valid", "mapped", or "ignored", but existed in Unicode 3.2 and were either disallowed in IDNA2003 or were mapped differently there.
- These five CJK compatibility ideographs had their decomposition mappings corrected in Unicode 4.0 (Corrigendum #4: Five CJK Canonical Mapping Errors); they are the last characters whose existing normalization behavior changed before normalization was more strictly stabilized.

Henri suggests that we explicitly call out the exceptional behavior, so that implementers take extra care to handle it properly.

Proposal

After PAG discussion, we propose to remove the inconsistent behavior, and to regularize and simplify the UTS #46 data file derivation, which will then simplify implementation requirements.

IDNA2008 was published in 2010. UTS #46 was intended to bridge the differences between IDNA2003 and IDNA2008 by providing a mapping table (still relevant), dealing with "deviation" characters (which we deprecated in 2023 / Unicode 15.1), and disallowing other characters that would behave differently in old vs. new implementations.

IDNA2003 was superseded fourteen years ago. We believe that differences with IDNA2003 implementations are no longer relevant. Therefore, we propose to no longer disallow characters because they behave differently from IDNA2003.

Instead, we propose to define the base exclusion set as a smaller set of characters that we think should remain disallowed:

- U+FFFC OBJECT REPLACEMENT CHARACTER
- U+FFFD REPLACEMENT CHARACTER
- U+E0001..U+E007F tag characters

Compared with the current base exclusion set, this changes the following characters from "disallowed" to a different Status. As a result, some unusual input domain names would be mapped to valid ones rather than rejected. Domain names that already map to valid ones are unaffected.

Changing from "disallowed" to "ignored"

- U+115F..U+1160 HANGUL CHOSEONG FILLER..HANGUL JUNGSEONG FILLER
- U+17B4..U+17B5 KHMER VOWEL INHERENT AQ..KHMER VOWEL INHERENT AA
- U+180E MONGOLIAN VOWEL SEPARATOR
- U+2061..U+2063 FUNCTION APPLICATION..INVISIBLE SEPARATOR
- U+206A..U+206F INHIBIT SYMMETRIC SWAPPING..NOMINAL DIGIT SHAPES
- U+3164 HANGUL FILLER
- U+FFA0 HALFWIDTH HANGUL FILLER
- U+1D173..U+1D17A MUSICAL SYMBOL BEGIN BEAM..MUSICAL SYMBOL END PHRASE

"Ignored" means that these characters map to the empty string; in other words, they are removed from the label before NFC normalization and further processing.

Removing these from the base exclusion set makes them behave like many other characters that have the Default_Ignorable_Code_Point property. Other "ignored" characters include U+00AD SOFT HYPHEN, U+034F COMBINING GRAPHEME JOINER, U+200B ZERO WIDTH SPACE, U+2064 INVISIBLE PLUS, and the variation selectors.

In "xn--" Punycode, they remain disallowed.

Changing from "disallowed" to "mapped"

- U+04C0 CYRILLIC LETTER PALOCHKA
- U+10A0..U+10C5 GEORGIAN CAPITAL LETTER AN..GEORGIAN CAPITAL LETTER HOE
- U+2132 TURNED CAPITAL F
- U+2183 ROMAN NUMERAL REVERSED ONE HUNDRED
- U+2F868 CJK COMPATIBILITY IDEOGRAPH-2F868
- U+2F874 CJK COMPATIBILITY IDEOGRAPH-2F874
- U+2F91F CJK COMPATIBILITY IDEOGRAPH-2F91F
- U+2F95F CJK COMPATIBILITY IDEOGRAPH-2F95F
- U+2F9BF CJK COMPATIBILITY IDEOGRAPH-2F9BF

These characters were "disallowed" because their Case_Folding, or, for the five CJK their Decomposition_Mapping, changed between Unicode 3.2 and when those properties were stabilized (many years ago now for both properties). Case_Folding changed when lowercase variants were added; since Case_Folding was stabilized, Unicode generally encodes both case variants at the same time.

Removing these from the base exclusion set makes them behave like other characters that have Case_Folding or Decomposition_Mapping values. In "xn--" Punycode, they remain disallowed.

Changing from "disallowed" to "valid"

- U+1806 MONGOLIAN TODO SOFT HYPHEN

This one character would become "valid" both when it occurs directly in a label as well as in "xn--" Punycode.

This character has General_Category=Dash_Punctuation, which makes it DISALLOWED in IDNA2008 regardless.

No changes to the idna2008derived data file

None of these changes affect the idna2008derived data file. In terms of IDNA2008 (as opposed to UTS #46), these characters remain DISALLOWED.

---

https://www.unicode.org/reports/tr46/#Processing

https://github.com/unicode-org/unicodetools/issues/687 UTS #46 IdnaTestV2.txt: add 5 normalization corrections

https://unicode-org.atlassian.net/browse/ICU-22658 UTS #46 incorrectly disallows 5 Han characters

> Furthermore, today it appears to be the case that all "disallowed" characters either decompose to themselves or have a singleton decomposition in NFD. It would be useful to have a remark about whether this is a guarantee that is expected to hold for future versions.

We do not want to make that promise, and it seems too specific to say anything about it in UTS #46.

6.3 UTS #46 VerifyDnsLength vs. the empty root label

Recommended UTC actions

1. **Action Item** for Mark Davis, PAG: Review IdnaTestV2.txt and its generator code for setting appropriate error codes for VerifyDnsLength=true. For Unicode 16.0. See L2/24-064 item 6.3.

Feedback (verbatim)

Date/Time: Mon Mar 18 08:41:27 CDT 2024
ReportID: ID20240318084127
Name: Henri Sivonen
Report Type: Error Report
Opt Subject: UTS 46

https://www.unicode.org/reports/tr46/#ToASCII says "When VerifyDnsLength is true, the empty root label is disallowed." Yet, it appears that IdnaTestV2.txt is meant to be run with the flags set to the more restrictive options but the test input "a.b. c。d。" and similar inputs following it are expected to pass without errors despite (after normalization) ending with the empty root label dot.

It's unclear to me if this is a test suite bug or a spec bug. I observe that the VerifyDNSLength check in the Rust `idna` crate allows the trailing dot (agreeing with the test suite but appearing to disagree with the spec).

Background information / discussion

Use this section for any notable additional information to add to the public report (delete otherwise).

7. Collation

7.1 DUCET and UCD disagree on the size of the Tangut_Supplement block

Recommended UTC actions

1. **No Action**: PAG recommends no action. This has been fixed in the UCA 16.0 data files.

Feedback from ICU4X

Via Manish Goregaokar (मनीष गोरेगांवकर)

Blocks.txt claims:

```
Unset
18D00..18D7F; Tangut Supplement
```

Whereas allkeys.txt claims:

```
Unset
@implicitweights 18D00..18D8F; FB00 # Tangut Supplement
```

I assume this is a typo in allkeys, and the block should be made to be 18D00..18D7F.

Background information / discussion

Ken Whistler:

Not a typo, but an oversight in following through on the decision as of Unicode 14.0 to size down the Tangut Supplement block.

Unicode 13.0:

```
Unset
18D00..18D8F; Tangut Supplement
```

Unicode 14.0:

```
Unset
18D00..18D7F; Tangut Supplement
```

This history of that was complicated, but came down to wanting to provide more contiguous room for the Tangut Components supplement block now starting at 18D80.

The oversight in allkeys.txt is of no actual consequence for handling real data, because there won't be any code points in 18D80..18D8F until Unicode 17.0, but yeah, we should fix the inconsistency now for 16.0.

7.2 The sorting order of Bopomofo ㄒ (U+3112) and ㄏ (U+312C) is the wrong way around

Recommended UTC actions

1. **Consensus**: In the DUCET, change U+312C ㄏ to sort before U+3112 ㄒ. For Unicode 16.0. See L2/24-064 item 7.2.
2. **Action Item** for Ken Whistler, PAG: In the DUCET, change U+312C ㄏ to sort before U+3112 ㄒ. For Unicode 16.0. See L2/24-064 item 7.2.

Feedback (verbatim)

Date/Time: Sun Jan 21 03:42:45 CST 2024
ReportID: ID20240121034245
Name: Jaycee Carter
Report Type: Error Report
Opt Subject: DUCET allkeys.txt

The sorting order of Bopomofo ㄒ (U+3112) and ㄏ (U+312C) is the wrong way around. ㄒ is equivalent to pinyin "x", while ㄏ is now obsolete in Mandarin, but was originally included in Bopomofo to represent the initial /ɲ/ in Old National Pronunciation (老國音). It is no longer usually included in Bopomofo tables. However, sources which do include ㄏ list it before ㄒ, as part of the sequence of alveolo-palatal initials:

1. Page 24 of a copy of《註音漢字》from 1936, column 4 (https://en.wikipedia.org/w/index.php?title=File:CADAL11100176 注音漢字.djvu&page=24).

2. A table in《校改國音字典》from 1920 (https://commons.wikimedia.org/wiki/File:《校改國音字典》注音符號.jpg).

3. Two (unfortunately unsourced) copies of what look to be period books on Wikipedia (https://en.wikipedia.org/wiki/File:Bopomofo.gif; https://zh.wikipedia.org/wiki/File:Bopomofo_in_Regular,*Handwritten_Regular*%26_Cursive_formats.jpg).

As such, their primary weights should be switched in the UCA: U+312C should have the value [.4573.0020.0002], and U+3112 should have the value [.4574.0020.0002].

7.3 UCA 16: virama variants primary vs. tertiary

Recommended UTC actions

1. **No Action**: PAG recommends no action.

PAG input

From Ken Whistler & Markus Scherer, PAG

For Kirat Rai and other scripts that have multiple virama or "vowel killer" characters, it is not obvious whether to sort them as primary-different or merely tertiary-different from each other. For a shape variant, tertiary seems appropriate; for a functionally different character, primary seems better. However, it is not always clear how different such characters are from each other.

This is probably not very visible; it would only affect strings that have a common prefix followed by one or the other such character.

For Unicode 16, we ended up leaving Kirat Rai virama vs. saat primary-different, but making Tulu-Tigalari looped virama only tertiary-different from virama.

Background information / discussion

Ken gave Kirat Rai virama & saat characters separate primary CEs.

Markus made saat a tertiary variant of virama:

```
Unset
# L2/22-043R: Difference between Sign Virama and Sign Saat:
# Both the signs are used to mute the inherent vowel sound. [...]
# SIGN SAAT is only used to mute the inherent vowel of the first letter of the word;
# all other places are represented by SIGN VIRAMA.
# Both the signs are represented in Devanagari by virama U+094D.
16D6B;KIRAT RAI SIGN VIRAMA;Lm;;;;;;
16D6C;KIRAT RAI SIGN SAAT;Lm;<sort> 16D6B;;;;;
```

Ken responded:

In the past I have used tertiary variants for viramas if they were just formal shape variants. You can see this in Malayalam, for example:

```
Unset
0D4D;MALAYALAM SIGN VIRAMA;Mn;;;;;;

# Sort the two alternate form viramas as variants of virama.
```

```
0D3B;MALAYALAM SIGN VERTICAL BAR VIRAMA;Mn;<sort> 0D4D;;;;;
0D3C;MALAYALAM SIGN CIRCULAR VIRAMA;Mn;<sort> 0D4D;;;;;
```

But I'm not sure we have a precedent for two different characters that consistently are used in distinct word positional contexts, as opposed to just being stylistic variants.
Another potential exceptional case we have in new 16.0 characters where a virama might be just a presentation variant is in Tulu-Tigalari:

```
Unset
113CE;TULU-TIGALARI SIGN VIRAMA;Mn;;;;;;
113CF;TULU-TIGALARI SIGN LOOPED VIRAMA;Mc;;;;;;
113D0;TULU-TIGALARI CONJOINER;Mn;;;;;;
```

The looped virama may just be a presentation variant of 113CE, so we could consider giving that particular one a tertiary weight distinction. The conjoiner is the invisible character that conjoins horizontally *or* vertically, depending, whereas the virama is basically a vowel killer. I didn't jump on this one already, because the situation for the regular virama and the looped virama is also a bit different than just being a presentation style -- they may also occur in somewhat different contexts, so I'd like to have a discussion about the details before committing on that one.

7.4 re-align the DUCET & CLDR: order of groups below letters

Recommended UTC actions

1. **Consensus**: In the UCA DUCET, move the non-decimal-digit numerics to sort right after decimal digits. For Unicode 16.0. See L2/24-064 item 7.4.
2. **Action Item** for Ken Whistler, PAG: In the UCA DUCET, move the non-decimal-digit numerics to sort right after decimal digits. For Unicode 16.0. See L2/24-064 item 7.4.

PAG input

From Markus Scherer, PAG

UTS #10 (UCA) defines a default sort order (DUCET). CLDR tailors that sort order for its default (root) sort order. (ICU implements the CLDR order.)

The biggest difference from this tailoring is among characters with non-zero primary weights that sort below letters. CLDR moves some characters so that they better fit into a small number of distinct “reordering groups” for parametric reordering at runtime. In effect, CLDR swaps the relative orders of two groups of characters.

These sort orders have been different since 2010. Since then we have generated two sets of files with sort order data and test data, and many users have experienced a slightly different sort order from what UTS #10 provides. We want to re-align them.

After much discussion, we have agreed to change both sort orders by moving the non-decimal-digit numeric characters (ones with values 10, 20, 30, ... or 11, 12, 13, ...; fractions; counting rods; etc.) to sort *after* the decimal digits. It puts the "larger" numbers after single digits which makes sense for some series of related characters. And it puts the gc=Letter_Number characters between digits and letters. For other non-digit numerics, basically any order seems fine, and they move as part of the group.

With this one move, the DUCET and the CLDR root sort orders become very nearly the same.

Also, in the DUCET, the non-digit numerics become non-variable. In CLDR, by default, they have been non-variable already.

Background information / discussion

The CLDR root order has been different from the DUCET since CLDR 1.9 in 2010: PRI #175 & L2/10-275R

This was motivated by national standards to “sort digits after letters” and user requests to “ignore punctuation”, together with creating/naming/documenting a small set of “reordering groups” that have been built into APIs.

Groups of characters below letters:

- DUCET:
    - First variable-weighted non-script-specifics:
        - whitespace, punctuation, general symbols, **non-decimal-digit numerics**
    - and then non-variable-weighted non-script-specifics:
        - extenders, currency symbols, decimal digits
    - (Most of the extenders do seem to be script-specific)
- CLDR: whitespace, punctuation, general symbols incl. extenders, currency symbols, and numbers (**non-decimal-digit numerics** & digits)
    - (The “variable top” is tailorable; its CLDR default is between punctuation & general symbols.)

DUCET	CLDR	comments	proposed
whitespace	whitespace	same	whitespace
punctuation	punctuation	same	punctuation
general symbols	general symbols	same (last: U+FFFC OBJECT REPLACEMENT CHARACTER)	general symbols
	35 extenders	CLDR: part of symbols	
	60 currency symbols	CLDR: between gen. symbols & numbers	
505 non-decimal-digit numerics	non-decimal-digit numerics	U+09F4 BENGALI CURRENCY NUMERATOR ONE.. U+1D371 COUNTING ROD TENS DIGIT NINE	
35 extenders		U+02D0 MODIFIER LETTER TRIANGULAR COLON.. U+30FE KATAKANA VOICED ITERATION MARK	extenders
60 currency symbols		U+00A4 CURRENCY SIGN.. U+20C0 SOM SIGN	currency symbols
decimal digits	decimal digits	same	decimal digits
		-	non-decimal-digit numerics

The 35 UCA “extenders” are *similar to* the 38 `[:Extender:]&[:Lm:]`.

With this one move, the DUCET and the CLDR root sort orders become very nearly the same.

Remaining exceptions:

- Different threshold for Non-variable symbols.
- Additional contractions for Tibetan
- Tailored noncharacter weights — U+FFFE and U+FFFF have special tailorings

(Until CLDR 45, U+20A8 RUPEE SIGN and U+FDFC RIAL SIGN were also tailored to sort differently from the DUCET, but CLDR has agreed to align these in CLDR 46 / fall 2024.)

8. Security

8.1 Strange Identifier_Type combinations

Recommended UTC actions

1. **Consensus**: Change the Identifier_Type of U+A9CF JAVANESE PANGRANGKEP to Limited_Use Uncommon_Use, removing Exclusion. For Unicode 16.0. See L2/24-064 item 8.1.
2. **Action Item** for Markus Scherer, PAG: Change the Identifier_Type of U+A9CF JAVANESE PANGRANGKEP to only Limited_Use Uncommon_Use, removing Exclusion. For Unicode 16.0. See L2/24-064 item 8.1.
3. **Action Item** for Markus Scherer, PAG: In UTS #39, add a sentence to the Uncommon_Use description along the lines of “May be combined with Exclusion or Limited_Use for characters that are less common than the main characters of their scripts.” For Unicode 16.0. See L2/24-064 item 8.1.

PAG input

From Markus Scherer, PAG

Identifier_Type is a UTS #39 (security) property with data in https://www.unicode.org/Public/security/latest/IdentifierType.txt .

Overview: UTS #39 Table 1. Identifier_Status and Identifier_Type

Characters can have certain combinations of multiple Identifier_Type values.

The types Exclusion and Limited_Use look mutually exclusive with each other. From the spec:

- Exclusion: Characters with Script_Extensions values containing a script in Table 4, Excluded Scripts from [UAX31], and no script from Table 7, Limited Use Scripts or Table 5, Recommended Scripts, other than “Common” or “Inherited”.
- Limited_Use: Characters from scripts that are in limited use: with Script_Extensions values containing a script in Table 7, Limited Use Scripts in [UAX31], and no script from Table 5, Recommended Scripts, other than “Common” or “Inherited”.

However, there is one character that has both of these restrictions:

```
Unset
A9CF          ; Limited_Use Exclusion          # 5.2        JAVANESE PANGRANGKEP
```

U+A9CF has these Script_Extensions:

```
Unset
A9CF          ; Bugi Java # Lm       JAVANESE PANGRANGKEP
```

Buginese is in Table 4. Excluded Scripts and Javanese is in Table 7. Limited Use Scripts.

Therefore, U+A9CF should have Limited_Use and not Exclusion.

8.2 UTR #36 has been dead for a decade and should be mothballed

Recommended UTC actions

1. **Consensus**: Stabilize UTR #36. See L2/24-064 item 8.2.
2. **Action Item** for Mark Davis, PAG: Stabilize UTR #36 with a cover page that points out that some material may still be useful while other material has been superseded by UTS #39 and UTS #55. See L2/24-064 item 8.2.

PAG input

From a discussion on the PAG list regarding overannuated action items for UTR #36.

UTR #36 has not been maintained for a decade. Some of the material may still be useful, but more recent material can be found in UTS #39 and UTS #55. Some of the material has been superseded by those UTSes, and would be very misleading for readers. However, the PAG has a full plate as it is, and performing the work to sift through the material is of lower priority. Some of the useful material in a stabilized UTR #36 could in the future be incorporated into UTS #39 in the future, if resources become available.

Given the situation, there was consensus in the PAG that UTR #36 be stabilized (and associated action items be closed), and the stabilization page should describe the situation.

8.3 short names for Identifier_Status & Identifier_Type

Recommended UTC actions

1. **Consensus**: In UTS #39 and its data files, define & document short property aliases ID_Status for Identifier_Status and ID_Type for Identifier_Type. For Unicode 16.0. See L2/24-064 item 8.3.
2. **Action Item** for Markus Scherer, PAG: In UTS #39 and its data files, define & document short property aliases ID_Status for Identifier_Status and ID_Type for Identifier_Type. For Unicode 16.0. See L2/24-064 item 8.3.

PAG input

From Markus Scherer, PAG

We currently don't have short property names for Identifier_Status & Identifier_Type, nor for their values -- deliberately. I have been pushing to not invent short names for everything.

This generally helps with readability and documentation.

For these two properties, especially since they are enumerated and set properties, I think it could make sense to create what seem like obvious and still-readable shorter names.

Consider a regex character class:

- `[[:Identifier_Type=Recommended:][:Identifier_Type=Inclusion:][:Identifier_Type=Limited_Use:]]`
  vs.
- `[[:ID_Type=Recommended:][:ID_Type=Inclusion:][:ID_Type=Limited_Use:]]`

Shorter names like ID_Status and ID_Type would follow the pattern of the long names ID_Start and ID_Continue.

I am not suggesting short names for the *values* of these two properties.

9. Authorize proposed updates

Recommended UTC action

1. Consensus: Authorize proposed updates of UAX #9 and UTS #46, for Unicode 16.0.