UTC #181 properties feedback & recommendations

Markus Scherer & Josh Hadley / Unicode properties & algorithms group, 2024-nov-01

Participants

The following people have contributed to this document:

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

1. Core Spec

1.1 bad advice about composing custom vulgar fractions [#327]

Recommended UTC actions

1. **Consensus**: Re-word the Core Spec, Chapter 6, Section 6.2.9 Other Punctuation – Fraction Slash from "If the fraction is to be separated from a previous number, then a space can be used[...]" to indicate that a separator must be used in this situation and add a table of options for separators with description of their behavior. For Unicode 17.0. See L2/24-224 item 1.1.
2. **Action Item** for Josh Hadley, PAG: Re-word the Core Spec, Chapter 6, Section 6.2.9 Other Punctuation – Fraction Slash from "If the fraction is to be separated from a previous number, then a space can be used[...]" to indicate that a separator must be used in this situation and add a table of options for separators with description of their behavior. For Unicode 17.0. See L2/24-224 item 1.1.

Feedback (verbatim)

Date/Time: Thu Aug 08 21:51:58 CDT 2024
ReportID: ID20240808215158
Name: Marcel Schneider
Report Type: Error Report
Opt Subject: TUS

Hello,

The Unicode Standard misadvises about composing custom vulgar fractions, as it recommends breaking spaces to separate integers and vulgar fractions. It even recommends U+200B:

“If the fraction is to be separated from a previous number, then a space can be used, choosing the appropriate width (normal, thin, zero width, and so on). For example, 1 + thin space + 3 + fraction slash + 4 is displayed as 1¾.”
https://www.unicode.org/versions/Unicode15.0.0/UnicodeStandard-15.0.pdf#page=302&zoom=100,0,400

Although it was intended to be no-break, the Unicode THIN SPACE U+2009 is breaking. So is the ZERO-WIDTH SPACE U+200B, but by design.

The text of TUS is the more inadequate as there is no space between the integer and the precomposed fraction.

I’d suggest changing this to:

A preceding integer part must be separated from the digits composing the fraction. This can be achieved using any of U+200C ZERO WIDTH NON-JOINER, U+2060 WORD JOINER, U+202F NARROW NO-BREAK SPACE, or another no-break character of the appropriate width.

I noted this already on 2023-08-31T0736+0200 and came across it again now while documenting source code and keyboard layouts.

Best regards,

Marcel Schneider

Unicode 16 text location

https://www.unicode.org/versions/Unicode16.0.0/core-spec/chapter-6/#G2001

Proposed wording

Reword from:

> If the fraction is to be separated from a previous number, then a space can be used, choosing the appropriate width (normal, thin, zero width, and so on). For example, 1 + thin space + 3 + fraction slash + 4 is displayed as 1¾.

to

> A separator must be used to distinguish fraction digits from a previous or following digit that is not considered part of the fraction. Any non-decimal-digit character could be used as a separator. Table NN-N lists some possible separators and their typical visual result:

Table NN-N. Fraction-Number Separators

Codepoint	Name	Comment
U+202F	NARROW NO-BREAK SPACE	typically narrower than U+0020 SPACE / same width as U+2009 Thin Space; prohibits line break before and after
U+00A0	NO-BREAK SPACE	typically the same width as U+0020 SPACE; prohibits line break before and after
U+2060	WORD JOINER	no visible space (zero width); prohibits line break before and after
U+2064 [1]	INVISIBLE PLUS	no visible space (zero width); intended for interchange with math-aware programs; `lb=AL`

Note: There are many characters that have some properties similar to Word Joiner but are not recommended for use in this context.

Note: In contexts where it is not certain that a layout engine and font are used which support mixed fractions, a visible space should be used to visually separate the whole number and the numerator. This is not an issue for math-aware programs which support the Fraction Slash and the Invisible Plus according to UTR #25.

[1] note for spec editors: this is recommended by core spec chapter 22 & UTR #25

2. UCD

2.1 Supply more guidance on whitespace [#210]

Recommended UTC actions

1. **Action Item** for Ken Whistler, PAG: Add disambiguating NamesList annotations to U+00A0 No-Break Space, U+2007 Figure Space, U+2008 Punctuation Space, U+2009 Thin Space, U+200A Hair Space, U+202F Narrow No-Break Space. For Unicode 17. See L2/24-224 item 2.1.

PAG input

From Mark Davis

We should supply more guidance on the use of the most common \p{whitespace} characters. The lack of such guidance can cause people to make incorrect choices of characters, and font designers to not structure their fonts correctly. This is particularly important for SPACE, NO-BREAK SPACE, THIN SPACE, and NARROW NO-BREAK SPACE, so that people understand that the appropriate widths need to correspond. The most effective way to do this is in the NamesList.

Related to that, we should surface the character aliases (from NameAliases.txt) in the NamesList (and thereby in the charts).

Good source of information:
https://learn.microsoft.com/en-us/typography/develop/character-design-standards/whitespace

Proposed NamesList additions for consideration by the NamesList editor:

- U+00A0 NO-BREAK SPACE
  - should be the same width as U+0020 SPACE
- U+2007 FIGURE SPACE
  - should be the same width as digit zero (0030)
- U+2008 PUNCTUATION SPACE
  - should be the same width as a full stop (002E)
- U+2009 THIN SPACE
  - should be much narrower than U+0020 SPACE; typically between 1/5 and 1/6 em
  - also known as narrow space
- U+200A HAIR SPACE
  - width 1/10 - 1/16 em
- U+202F NARROW NO-BREAK SPACE
  - should be the same width as U+2009 THIN SPACE
  - also known as no-break thin space

2.2 Linkification of URLs [#281]

Recommended UTC actions

1. **Consensus**: Authorize a Proposed Draft Unicode Technical Standard #xx, Unicode Linkification, based on the working draft in document L2/24-217. See L2/24-224 item 2.2.
2. **Action Item** for Mark Davis, Robin Leroy, PAG: Provide the text of Proposed Draft Unicode Technical Standard #xx. See L2/24-224 item 2.2.
3. **Action Item** for Michelle Perham, PAG: Post the PRI for Proposed Draft Unicode Technical Standard #xx. See L2/24-224 item 2.2.

Document

L2/24-122 “Linkification of URLs” by Mark Davis
L2/24-217 “Working Draft for Proposed Draft UTS #58 Unicode Linkification (revised)”

WD summary: This document specifies a mechanism for performing linkification of URLs containing non-ASCII characters in plain text. It also provides a corresponding mechanism for determining when to escape non-ASCII code points.

2.3 Should the modifier letters from the Phonetic Extensions Supplement have the Diacritic property? (Yes.) [#315]

Recommended UTC actions

1. **Consensus**: Assign the Diacritic property to the modifier letters from the Phonetic Extensions Supplement block, namely U+1D9B..U+1DBE [ᵛᶜᵈᵊᶠᵍᶣᶤᶦʲᶩᶫᶬᶭᶮᶯᶲᶴᶵᶶᶷᶸᶻᶼᶽ], for Unicode Version 17.0. See L2/24-224 item 2.3.
2. **Action Item** for Robin Leroy, PAG: In PropList.txt, assign the Diacritic property to the modifier letters from the Phonetic Extensions Supplement block, namely U+1D9B..U+1DBE [ᵛᶜᵈᵊᶠᵍᶣᶤᶦʲᶩᶫᶬᶭᶮᶯᶲᶴᶵᶶᶷᶸᶻᶼᶽ], for Unicode Version 17.0. See L2/24-224 item 2.3.

PAG input

From Robin Leroy, PAG: While drafting data[1] for L2/24-144, using the existing modifier letters with palatal hook (ᶩ and ᶭ) as reference, I noticed that these do not have the Diacritic property:

Unicode 15.1 characters with gc=Lm and dt=super and sc=Latn, grouped by Age and by Diacritic

Subset without Diacritic:

- Phonetic Extensions Supplement — Modifier letter
  - ᵈ U+1D9B MODIFIER LETTER SMALL TURNED ALPHA .. ᶾ U+1DBE MODIFIER LETTER SMALL EZH
- Superscripts And Subscripts — Superscripts
  - ⁱ U+2071 SUPERSCRIPT LATIN SMALL LETTER I
  - ⁿ U+207F SUPERSCRIPT LATIN SMALL LETTER N

- Latin Extended C — Addition for UPA
  - ᵛ U+2C7D MODIFIER LETTER CAPITAL V
- Latin Extended D — Medievalist addition
  - ꝰ U+A770 MODIFIER LETTER US
- Latin Extended D — Modifier letters for Chatino (México)
  - ꟲ U+A7F2 MODIFIER LETTER CAPITAL C
  - ꟳ U+A7F3 MODIFIER LETTER CAPITAL F
- Latin Extended D — Modifier letter for Japanese phonemic transcription
  - ꟴ U+A7F4 MODIFIER LETTER CAPITAL Q

This is unusual for superscript modifier letters in the Latin script. Of course not all such modifier letters should be Diacritic; for instance ꝰ is clearly an abbreviation, not a diacritic modifying some other character; but the contrast between the Phonetic Extensions and Phonetic Extensions Supplement modifier letters does not seem to have any obvious explanation.

The proposal L2/04-132, by Peter Constable, notes the use of the modifier vowels in diphthongs (§ E.4.1); but[2] this does not explain the discrepancy in Diacritic assignment, since the Phonetic Extensions modifier vowels have the Diacritic property.
In any case, such an explanation would not be applicable to the modifier consonants, which are explicitly compared to clearly Diacritic ones such as ʰ or ʷ in § E.4.2.

Background information / discussion

Recall the definition of Diacritic in UAX #44:

Property	Type	Status	Description
Diacritic	B	I	Characters that linguistically modify the meaning of another character to which they apply. Some diacritics are not combining characters, and some combining characters are not diacritics. Typical examples include accent marks, tone marks or letters, and phonetic modifier letters. The Diacritic property is used in tooling which assigns default primary weights for characters, for generation of the DUCET table used by the Unicode Collation Algorithm (UCA).

[1] https://github.com/unicode-org/unicodetools/pull/887 (“Modifier ɖɦɳʂʐ”)

[2] Contra the last paragraph of the background section of “Is gc=Lo *really* right for the two CHINESE SMALL ER? (No.)”, published as L2/24-162 §1.9 (PAG-internal #299)

2.4 UAX #42 Name properties "control" option [#328]

Recommended UTC actions

1. **Action Item** for John Wilcock, PAG: In UAX #42 section 4.4.2 Name properties, remove the long-obsolete alternative `<control>` from the character-name regex. Adjust the syntax example in section 12 accordingly. For Unicode 17.0. See L2/24-224 item 2.4.

Feedback (verbatim)

Date/Time: Fri Aug 09 21:37:02 CDT 2024
ReportID: ID20240809213702
Name: Robert Thomson
Report Type: Error Report
Opt Subject: Unicode Standard Annex #42

With respect to UAX #42 for unicode version 15.1.0 at https://www.unicode.org/reports/tr42/#d1e3008 viewed 2024-08-10, I believe there are a couple of minor errors:

In section 4.4.2 Name properties, the character name has a pattern option of `<control>`. None of the codepoints have that pattern, and I believe that with revision 9 and the introduction of the name alias pattern there is no longer the requirement to include "`|(<control>)`" in the character name pattern.

```
Unset
[name pattern, 12] =
  character-name = xsd:string { pattern="([A-Z0-9 #\-\(\)]*)|(<control>)" }
```

If you should agree with the previous conclusion then Section 12 contains an example fragment that is also in error

```
Unset
<char cp="001F" age="1.1" na="&lt;control&gt;" na1="UNIT SEPARATOR"
            gc="Cc" bc="S" lb="CM"/>
```

Background information / discussion

We have considered whether the latest UCDXML schema should work for validating past versions of the data. We noted several inconsistencies, including in how provisional properties have been handled. (Provisional properties are not stable and have been renamed, redesigned, and removed.)

The PAG had been under the impression that when a property or property value got renamed (that is, when a new alias was made the first or second alias), at least for a normative or informative property, UCDXML kept the old name. Since there is no aliasing mechanism in UCDXML, this would be necessary for implementers to have versionless references to UCDXML, and would facilitate upgrading versioned references. However, this has not consistently been the case; for instance, when `Hamza_On_Heh_Goal` was renamed to `Teh_Marbuta_Goal` (UTC-122-C4), UCDXML instead added a new attribute value `Teh_Marbuta_Goal` (UTC-122-A26). Similarly, `Indic_Matra_Category` is retained in the schema separately from `Indic_Positional_Category`, to which it was renamed as it was made informative (from provisional, UTC-140-C16).

It should suffice for the schema of each version to validate the data for that version. When parsing older versions of the data, the corresponding schemas should be used. We will endeavour to maintain stability of still-relevant features; in particular, any normative and informative properties and values of such properties that get renamed in the future should retain their current names in UCDXML.

2.5 Correction to CJKRadicals.txt [#337]

Recommended UTC actions

1. **Action Item** for Josh Hadley, PAG: Update the description of CJK radical numbers in CJKRadicals.txt to be consistent with the use of apostrophes per the kRSUnicode property. For Unicode 17.0. See L2/24-224 item 2.5.

Feedback (verbatim)

Date/Time: Mon Sep 23 05:13:18 CDT 2024
ReportID: ID20240923051318
Name: Michel Mariani
Report Type: Public Review Issue
Opt Subject: 508

UAX #38 mentions the CJKRadicals.txt data
file https://www.unicode.org/Public/UCD/latest/ucd/CJKRadicals.txt,
which should be updated to be consistent with the use of apostrophes after
the radical number described in the kRSUnicode property.

```
# CJK radical numbers match the regular expression [1-9][0-9]{0,2}\'{0,2}
# and in particular they can end with one or two U+0027 ' APOSTROPHE characters.
```

should be:

```
# CJK radical numbers match the regular expression [1-9][0-9]{0,2}\'{0,3}
# and in particular they can end with one, two, or three U+0027 ' APOSTROPHE characters.
```

2.6 Proposed update to UAX #42 UCDXML [#338]

Recommended UTC actions

1. **Consensus**: Authorize a Proposed Update of UAX #42 UCDXML and its associated data files. For Unicode 17.0. See L2/24-224 item 2.6.
2. **Action Item** for John Wilcock, PAG: Provide a Proposed Update of UAX42 UCDXML and its associated data files. For Unicode 17.0. See L2/24-224 item 2.6.
3. **Action Item** for Michelle Perham, UTC: Publish a PRI for the Proposed Update of UAX #42 UCDXML to close 2025-xx-xx. For Unicode 17.0. See L2/24-224 item 2.6.

PAG input

John Wilcock has been working on an update of UAX #42 and generating the associated data files. PAG should request that a PRI to be opened with the result so it can be reviewed.

Background information / discussion

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

2.7 Numeric annotations and properties for cuneiform signs [#341]

Recommended UTC actions

1. **Consensus**: Assign Numeric_Value=1/2 to U+12226 𒄦 CUNEIFORM SIGN MASH;
   Numeric_Value=1 to U+12038 𒀸 CUNEIFORM SIGN ASH, U+1239 𒉹 CUNEIFORM SIGN ASH ZIDA TENU, U+12079 𒁹 CUNEIFORM SIGN DISH, U+1230B 𒌋 CUNEIFORM SIGN U;
   Numeric_Value=2 to U+1222B 𒈫 CUNEIFORM SIGN MIN and U+12399 𒎙 CUNEIFORM SIGN U U[1];
   Numeric_Value=3 to U+1230D CUNEIFORM SIGN U U U[2],
   and assign Numeric_Type=Numeric to all of these characters, as described in L2/24-239. See L2/24-224 item 2.7.
2. **Action Item for** Robin Leroy, PAG: In UCD file UnicodeData.txt and derived files, assign Numeric_Value=1/2 to U+12226 𒄦 CUNEIFORM SIGN MASH; Numeric_Value=1 to U+12038 𒀸 CUNEIFORM SIGN ASH, U+1239 𒉹 CUNEIFORM SIGN ASH ZIDA TENU, U+12079 𒁹 CUNEIFORM SIGN DISH, U+1230B 𒌋 CUNEIFORM SIGN U; Numeric_Value=2 to U+1222B 𒈫 CUNEIFORM SIGN MIN and U+12399 𒎙 CUNEIFORM SIGN U U; Numeric_Value=3 to U+1230D CUNEIFORM SIGN U U U,
   and assign Numeric_Type=Numeric to all of these characters, as described in L2/24-239. See L2/24-224 item 2.7.
3. **Action Item for** Ken Whistler, EDC: Consider the names list annotations proposed in L2/24-239, §3.1. For Unicode Version 17.0.
4. **Action Item for** Ken Whistler, EDC: Consider the names list annotations proposed in L2/24-239, §3.2, when the characters proposed in L2/24-210 are incorporated into the standard.

Document

L2/24-239 by Robin Leroy.

This document proposes Numeric_Value property assignments for eight characters in the Cuneiform block.
It also proposes informative aliases, cross references, and informative notes,
as well as some adjustments to subheadings, for the character names
lists for the Cuneiform, Cuneiform Numbers and Punctuation, and Early Dynastic Cuneiform blocks.

[1] Readings man and niš.

[2] Reading eš.

3. Characters

3.1 Proposal to add a new Script-Hybrid CJK Ideographs block [#323]

Recommended UTC actions

No action necessary.

Document

L2/24-201 by Gen Kojitani

From the doc intro:

This document is a proposal for adding a new block named “Script-Hybrid CJK Ideographs” to the Unicode Standard. This proposal is a revised version of my previous proposal L2/24-125 following feedback from the UTC-180 meeting (L2/24-165), and is related to my previous proposal L2/23-139R.

From the doc background section:

Most CJK abbreviations are made of the same components as regular CJK characters, but some of the relatively new abbreviations include components derived from non-Han writing systems such as Latin, Katakana, and Hangul, and these abbreviations are used mainly for signboards and other handwritten texts from the viewpoint of ease of writing. ... These abbreviations are not official, but are fairly commonplace in signboards and other handwritten documents. ...

Background information / discussion

PAG defers to CJK+SEW to determine encoding eligibility.
For character properties:

- These would be similar to some CJK ideographs that have the kStrange property and which include elements from Hangul, for example. Therefore, mostly the same properties. Except:
    - Not Unified_Ideograph. These characters would not fit the CJK model of properties and analysis.
    - *Possible* Script_Extensions including Hira or Kana, subject to further discussion. However, no Latn, because that would cause problems for determining script runs.
- gc=Lo (not symbols); Ideographic
- No compatibility decompositions. Not even DUCET `<sort>` decomps.

3.2 PAG review of draft properties for Hiragana and katakana digraphs (1B123..1B125) [#326]

Recommended UTC actions

1. **No Action**: PAG recommends no action; no concerns from our side.

Document

Proposal: L2/24-150
CJK recommendations: L2/24-165 §15

[180-C6] Consensus: Provisionally assign U+1B123 HIRAGANA DIGRAPH KOTO, U+1B124 KATAKANA DIGRAPH TOKI, and U+1B125 KATAKANA DIGRAPH TOTE in the Kana Extended-A block, based on document L2/24-150 (Kojitani) and as amended in Section 15 of document L2/24-165.

Background information / discussion

The Hiragana KOTO is propertywise to こと what 𛀆 is to より;
the Katakana TOKI and TOTE are propertywise to トテ and トキ what 𛀁 is to コト.
These statements are tested as part of the AdditionComparisons invariant tests.
In particular, all are lb=ID, ea=W, and their scripts are according to their names.

4. Proposed new scripts & characters

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

No further recommended actions from our side.

- L2/24-153 Proposal to encode Bengali Sign Combining Anusvara Above -- Jan Kučera [SEW #476]
  - Propertywise like the Bengali sign candrabindu.
  - Also like the similarly-named TELUGU SIGN COMBINING ANUSVARA ABOVE, up to block and script.
  - These statements are tested in the AdditionComparisons invariant tests.
  - In particular, InSC=Bindu, InPC=Top, Other_Alphabetic.
- L2/24-202 Phonetic characters: Greek and Latin? - Denis Moyogo Jacquerye [SEW #486]
  - More modified Greek letters encoded as Latin, more letters with palatal hook; the first letters that fall into both of these categories, but otherwise nothing new. Propertywise like the existing ᶀ and ꭓ, which are alike.
- L2/24-145R Unicode request for modifier psi and omega -- Miller [SEW #485]
  - More modifier Greek small letters, propertywise like ᵝ, in particular, Other_Lowercase and Diacritic.
- L2/24-147 Modifier Sinological extensions to the IPA -- Miller [SEW #493]
  - Propertywise to their non-modifier counterparts [ᴀᴇɿʅʮʯȡȴȵȶ] what ʶ is to ʁ (this is checked in the AdditionComparisons test suite of the invariant tests).
  - In particular, <super>-decomposing to the non-modifier counterparts, and Diacritic and Other_Lowercase.
- L2/24-171 Miscellaneous historical and para-IPA modifier letters - Miller [SEW #494]
  - The barred letters are propertywise like ɇ.
  - The modifier small letters are propertywise like other modifier Latin letters, in particular, Other_Lowercase and Diacritic, and <super>-decomposing to their lowercase counterparts. The modifier ɉ is Soft_Dotted, like ɉ itself—also like the existing ᶨ and ʝ. Note that ɳ and ɉ are part of a case pair, though this does not affect the properties of the modifier letters.
- L2/24-172 Unicode request for 256th, 512th, and 1024th notes and rests --Gavin Jared Bala, Kirk Miller [SEW #392]
  - Propertywise like existing flags and existing rests. In particular, vo=U, Diacritic and Other_Grapheme_Extend for the flags, lb=CM for the flags and lb=AL for the rests.
- L2/24-174 Unicode request for Turkish and Arabic accidentals -- Gavin Jared Bala, Kirk Miller [SEW #445]
  - Propertywise like U+1D130 □; in particular bc=L as noted by SAH, lb=Al, vo=U.
- L2/24-144 Unicode request for modifier letters with palatal hook -- Miller [SEW #443]
  - Propertywise like other modifier letters, Other_Lowercase and Diacritic. Note that the current modifier letters with palatal hook, ᶡ and ᶪ, do not currently have the Diacritic property, but this appears to be an omission; the PAG will report on this separately, see unicode-org/properties#315. Note also ᶎ, unlike ᶁꞕᶇᶊ, is part of a case pair, but this does not affect the properties of its modifier counterpart.
- L2/24-203 On the Indic_Syllabic_Category of vowel carriers -- Robin Leroy [SEW #526]

- L2/24-210 Archaic cuneiform numerals -- Robin Leroy, Anshuman Pandey, and Steve Tinney [SEW #542]
    - The properties are similar to those of characters in the Cuneiform Numbers and Punctation block, and are all tested by such comparisons.
    - As described in the proposed core specification text, the Numeric_Value property assignments follow the same principles, and can in general be straightforwardly tested by comparison with the characters mentioned in cross-references.
    - The fractions of the fourth millennium capacity system all have Numeric_Value=1, as some have unclear relations to the N39 and related units. The third millennium fractions have fractional Numeric_Value like their already-encoded counterparts 𒐗, 𒐙, 𒐚, etc.
    - As far as the Script and Script_Extension properties are concerned, the characters fall into three categories: Script=Script_Extensions=Cuneiform (signs used in the third millennium only), Script=Cuneiform, Script_Extensions=Proto_Cuneiform|Cuneiform (signs used in the fourth and third millennia; as described in the proposal, p. 46, usage in third millennium studies will be more frequent, hence the choice of Script property), Script=Script_Extensions=Proto_Cuneiform (signs used only in the fourth millennium).
    - The characters are vo=R as everything else in the Cuneiform script, notwithstanding the Early Fribergian practice noted in L2/24-210 p. 29 n. 58.
- L2/24-237 Capital R with long leg — Denis Moyogo Jacquerye [SEW #527]
    - A new uppercase counterpart for a pre-existing lowercase letter (ɽ), unproblematic. Propertywise the same as the recent U+A7DC Ɬ LATIN CAPITAL LETTER LAMBDA WITH STROKE.
- L2/24-243 Changing Latin script r glyphs and adding their capital characters — Denis Moyogo Jacquerye [SEW #529]
    - New uppercase counterparts for pre-existing lowercase letters (ꞃꭇ), unproblematic. Propertywise the same as the recent U+A7DC Ɬ LATIN CAPITAL LETTER LAMBDA WITH STROKE.
- L2/24-213 Unicode request for additional tremoli -- Bala and Miller [SEW #447]
    - More combining tremoli, propertywise like the existing tremoli, in particular vo=U and Diacritic.
    - More fingered tremoli, propertywise like the existing tremoli, in particular lb=AL and vo=U.
    - The buzz mark is propertywise like a combining tremolo.
- L2/24-214 Unicode request for triple and quadruple flat -- Bala and Miller [SEW #446]
    - A triple flat, propertywise like the double flat 𝄫. In particular, bc=L, *contra* the proposal which suggests bc=ON (like the single ♭). Similar also to the half sharp etc.
- L2/24-236 Proposal to encode two Tangut ideographs (WG2 N5286) — Eiso Chan et al. [SEW #555]
    - Two more Tangut ideographs in the Tangut Supplement block, propertywise like the others in the same block.
- L2/24-234 Unicode request for barred letters — Kirk Miller, et al [SEW #510]
- L2/24-231 Unicode request for modifier small capital P — Kirk Miller, Denis Moyogo Jacquerye [SEW #554]
- L2/24-219 Unicode request for subscript w y z and ɣ - Miller [SEW #553]
    - More subscripts, propertywise like the existing subscripts in the `Superscripts_And_Subscripts` block.
- L2/24-232 Unicode request for compound tone diacritics III — Kirk Miller [SEW #528]

5. East Asian Text

5.1 Working Draft UTR East Asian Spacing [#343]

Recommended UTC actions

1. **Consensus**: Authorize a Proposed Draft Unicode Technical Report #xx, East Asian Spacing, based on the working draft in document L2/24-259. See L2/24-224 item 5.1.
2. **Action Item** for Koji Ishii, Markus Scherer, PAG: Provide the text of Proposed Draft Unicode Technical Report #xx. See L2/24-224 item 5.1.
3. **Action Item** for Michelle Perham, PAG: Post the PRI for Proposed Draft Unicode Technical Report #xx. See L2/24-224 item 5.1.

Document

L2/24-259 by Koji Ishii

East Asian established typography defines that a small amount of visible space between East Asian scripts and other scripts improves readability. This report describes the algorithm and the data which can be used to automatically add visible space.

Background information / discussion

UTC-180 minutes:

F.1.1 Auto Spacing in CJK text / F.1 PAG: UTC #180 properties feedback & recommendations [Markus Scherer, et al, L2/24-162] section 6.4 UNICODE AUTO SPACING (Proposal) [Koji Ishii, et al, L2/24-057] Long discussion.

- [180-A82] Action Item for Koji Ishii, Markus Scherer, PAG: Prepare a working draft UTR for East Asian Auto Spacing based on L2/24-057R, with feedback from UTC #180 discussion, in collaboration with PAG. See L2/24-162 item 6.4.

6. Line Break

6.1 Hyphens and Hebrew again: further adjustments to LB21a and LB20a [#308]

Recommended UTC actions

1. **Consensus**: Add a new Line_Break property value Unambiguous_Hyphen (short alias: HH) and assign this value to the ten characters that have General_Category=Pd and Line_Break=Break_After in Unicode Version 16.0, listed below. Amend rules LB12a and LB21 of the Unicode Line Breaking Algorithm to treat HH like BA, and amend rules LB20a and LB21a to refer to the set of characters with lb=HH instead of singling out a single character or doing set arithmetic on the set of characters with lb=BA. In addition, amend rule LB20a to treat HL like AL. See L2/24-224 item 6.1. For Unicode Version 17.0.
    - U+058A ֊ ARMENIAN HYPHEN
    - U+05BE ־ HEBREW PUNCTUATION MAQAF
    - U+1400 ᐀ CANADIAN SYLLABICS HYPHEN
    - U+2010 ‐ HYPHEN
    - U+2012 ‒ FIGURE DASH
    - U+2013 – EN DASH
    - U+2E17 ⸗ DOUBLE OBLIQUE HYPHEN
    - U+2E40 ⹀ DOUBLE HYPHEN
    - U+2E5D ⹝ OBLIQUE HYPHEN
    - U+10EAD 𐺭 YEZIDI HYPHENATION MARK
2. **Action Item** for Robin Leroy, PAG: In UCD file PropertyValueAliases.txt, add a new Line_Break property value Unambiguous_Hyphen (short alias: HH). For Unicode Version 17.0. See L2/24-224 item 6.1.
3. **Action Item** for Robin Leroy, PAG: In UCD file LineBreak.txt and derived files, assign Line_Break=Unambiguous_Hyphen to the ten characters that have General_Category=Pd and Line_Break=Break_After in Unicode Version 16.0. For Unicode Version 17.0. See L2/24-224 item 6.1.
4. **Action Item** for Robin Leroy, PAG: In Unicode Standard Annex #14, add a description for line breaking class HH, and update rules LB12a, LB20a, LB21, and LB21a as described in L2/24-224 item 6.1. For Unicode Version 17.0.
5. **Action Item** for Robin Leroy, PAG: In UCD files LineBreakTest.txt and LineBreakTest.html, update rules LB12a, LB20a, LB21, and LB21a as described in L2/24-224 item 6.1. For Unicode Version 17.0.
6. **Action Item** for Robin Leroy, PAG: In UCD files LineBreakTest.txt and LineBreakTest.html, add realistic tests exercising the changes to the behaviour of rules LB20a and LB21. For Unicode Version 17.0. See L2/24-224 item 6.1.

PAG input

From Robin Leroy, PAG: In L2/24-162 §5.6 (internal PAG issue #307) an emergency minimal change to LB21a was proposed to ensure the implementability of Unicode 16.0 line breaking. This change consisted in the exclusion of U+3000 from the set of characters considered hyphens for the purposes of that rule. It was noted that the set was still likely far too large, but that further refinement should be done later to minimize risk. This is the proposal for further refinement.

1. We should split out from BA a new Line_Break value HH[1], corresponding to things that are unambiguous hyphens, containing at least HYPHEN and HEBREW PUNCTUATION MAQAF; \p{U16:gc=Pd}&\p{U16:lb=BA} seems like a reasonable and reasonably principled set.
2. As part of the split, we need to update LB12a [^SP BA HY HH] × GL and to add × HH to LB21, resulting in no change to the behaviour of these rules;
3. We should change LB20a to use that instead of singling out U+2010 HYPHEN, and to treat HL like AL: (sot | BK | CR | LF | NL | SP | ZW | CB | GL) (HY | [\u2010] HH) × (AL | HL)
4. We should likewise change LB21 to refer to hyphens, rather than lb=BA: HL (HY | [BA - $EastAsian] HH) × [^HL]

Background information / discussion

Recall that UAX #14 has two rules specific to hyphens[2]:

> LB20a Do not break after a word-initial hyphen.
>
> (sot | BK | CR | LF | NL | SP | ZW | CB | GL) (HY | [\u2010]) × AL
>
> LB21a Do not break after the hyphen in Hebrew + Hyphen + non-Hebrew.
>
> HL (HY | [BA - $EastAsian]) × [^HL]
>
> Note: In the above regular expression, the class [\u2010] contains the single character U+2010 HYPHEN.

The set BA - $EastAsian used in LB21a still includes plenty of characters irrelevant to the reason for this rule[3] and where its application is undesirable, such as a dozen spaces.

LB20a is curious because it includes AL but not HL, even though HL is described as « behav[ing] the same as characters of class AL » except for LB21a and LB21b. This is due to its origin as a Finnish tailoring for ICU. However, there is no reason to retain this discrepancy; just like there should be no line break after the hyphen in « the Akkadian first person possessive suffix *-ī* », there should likewise be none after « the Hebrew first person possessive suffix ׳- ».

A cursory search shows that U+05BE HEBREW PUNCTUATION MAQAF is also used for this purpose: the English Wiktionary redirects from ׳- to ׳־, and this page in Hebrew is full of discussion of suffixes written using a word-initial maqaf: https://hebrew-academy.org.il/category/%D7%A1%D7%99%D7%95%D7%9E%D7%AA-%D7%9D%D6%B4%D7%99/. A broader definition of hyphen (rather than the single U+2010) is therefore appropriate for LB20a.

[1] This name was chosen for a class containing the sole U+2010 when the rule that is now LB20a was first added to CLDR as a Finnish tailoring.

[2] It also has other rules that are described as involving hyphens, LB21 and LB12a, but they treat them in bulk with other non-hyphen characters included in BA and other classes.

[3] For background on LB21a see UTN #54, §432.2 with annotations §432.2.a and §432.2.b.

6.2 UAX #14 CSS normal ≠ default [#316]

Recommended UTC actions

No action. The text "(CSS default)" from the listing of CSS Text Level 3 has been removed in UAX #14 for Unicode 16.

Feedback (verbatim)

Date/Time: Wed Jul 31 03:01:40 CDT 2024
ReportID: ID20240731030140
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #14: Unicode Line Breaking Algorithm

https://www.unicode.org/reports/tr14/#CJ
Version: Unicode 15.1.0
Date: 2023-08-15
Revision: 51

Location:
5.1 Description of Line Breaking Properties
CJ: Conditional Japanese Starter

Problematic text:
CSS Text Level 3 (which supports Japanese line layout) defines three distinct values for its line-break behavior:
• strict, typically used for long lines
• normal (CSS default), the behavior typically used for books and documents
• loose, typically used for short lines such as in newspapers

Possible correction:
Delete "(CSS default)".

Explanation:
In CSS, at least in the current CSS Text Level 3 Candidate Recommendation, and the latest CSS Text Level 4 Working Draft, the default line-break behavior is not "normal". It is "auto", which basically means the browser can do whatever it wants by default. Indeed, my Firefox by default does not break before small hiragana. It does when "line-break: normal" is explicitly specified.

https://www.w3.org/TR/css-text-3/#line-break-property
https://www.w3.org/TR/2024/WD-css-text-4-20240529/#line-break-property

6.3 UAX #14 CGJ should not break a combining character sequence [#317]

Recommended UTC actions

1. **Consensus**: Change the Line_Break assignment of U+034F COMBINING GRAPHEME JOINER from Line_Break=GL (Glue) to Line_Break=CM (Combining_Mark). For Unicode Version 17.0. [Ref. L2/24-224 item 6.3]
2. **Action Item for** Robin Leroy, PAG: In LineBreak.txt and derived files, change the Line_Break assignment of U+034F COMBINING GRAPHEME JOINER from Line_Break=GL (Glue) to Line_Break=CM (Combining_Mark). For Unicode Version 17.0. [Ref. L2/24-224 item 6.3]
3. **Action Item for** Robin Leroy, PAG: In UAX #14, Unicode Line Breaking Algorithm, update the description of line breaking classes GL and CM to reflect the change in Line_Break property from GL to CM. Note in a migration section of the spec that lb=GL was a mistake. For Unicode Version 17.0. [Ref. L2/24-224 item 6.3]
4. **Action Item for** Robin Leroy, PAG: In the core spec, section 23.2.4 Combining Grapheme Joiner, clarify that CGJ does not join graphemes. For Unicode Version 17.0. [Ref. L2/24-224 item 6.3]
5. **Action Item for** Ken Whistler, PAG: In NamesList.txt, express that CGJ is used to affect the collation of adjacent characters for purposes of language-sensitive collation and searching, and also used to distinguish sequences that would otherwise be canonically equivalent. For Unicode Version 17.0. [Ref. L2/24-224 item 6.3]

Feedback (verbatim)

Date/Time: Wed Jul 31 08:12:26 CDT 2024
ReportID: ID20240731081226
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #14: Unicode Line Breaking Algorithm

https://www.unicode.org/reports/tr14/#LB9
Version: Unicode 15.1.0
Date: 2023-08-15
Revision: 51

Location: 6.1 Non-tailorable Line Breaking Rules
[LB9] "Treat X (CM | ZWJ)* as if it were X (where X is any line break class except BK, CR, LF, NL, SP, or ZW)."
[LB12] "GL ×"

Problem:
U+034F COMBINING GRAPHEME JOINER is in Mn, but its line breaking class is GL, not CM.
This causes unexpected behavior when GCJ is used in the middle of a combining character sequence.

Take the following two sequences:
(1) <u, COMBINING DIAERESIS, EM DASH>
(2) <u, CGJ, COMBINING DIAERESIS, EM DASH>
In (1), a line break is allowed before EM DASH (which has line breaking class B2).

In (2), LB9 applies with CGJ taking the place of X, then LB12 kicks in to forbid a line break before the EM DASH.

How I came up with the example: Section 23.2 "Layout Controls" of the Unicode Standard explicitly mentions the use of CGJ in German text to make a distinction between u-umlaut (which is sorted like <u,e>) and u-diaeresis (which is sorted like “u” with a secondary weight). The distinction is purely for collation and it doesn't make sense for such CGJ to affect line breaking behavior after the umlaut/diaeresis.

This is impossible to solve without separating CGJ in a different line breaking class from NBSP (currently both are GL). To see this, observe that in sequence (2) above, if NBSP were used in place of CGJ, the suppression of the line break before EM DASH is exactly the expected behavior.

This is also impossible to solve by tailoring, as CM and GL are non-tailorable classes, and LB9 and LB12 are non-tailorable rules.

While at it, I will also point out a typo:
[LB10] "Treat any remaining CM or ZWJ as it if were AL."
In this definition, the order of "it" and "if" should be reversed.

Background information / discussion

The typo has been corrected in Unicode Version 16.0, no action needs to be recorded for that one.

NamesList.txt version 16:

```
Unset
034F   COMBINING GRAPHEME JOINER
       * commonly abbreviated as CGJ
       * has no visible glyph
       * the name of this character is misleading; it does not actually join graphemes
```

FAQ:

> Q: Does U+034F COMBINING GRAPHEME JOINER join graphemes?
>
> No. Despite its name, the combining grapheme joiner neither joins graphemes together in the way punctuation might, nor does it create new graphemes by combinations of other characters. Especially, it cannot be used to *construct* grapheme clusters out of arbitrary character sequences, or extend the scope of subsequent combining characters. It has no impact on line breaking, except that as for other combining marks, it should be kept with its base when breaking a line.

The Early History of Combining Grapheme Joiner

At the behest of the PAG, the editor of UAX #14 summarizes here the history of CGJ from a time when it was not lb=GL, but should have been, to a time when it was assigned lb=GL, but should not have been.

April 2000–March 2002: Encoding and initial properties

CGJ was originally encoded in Unicode Version 3.2.

On its encoding see L2/00-156 and UTC-83-AI43 where its name was *Zero Width Grapheme Joiner*, UTC-84-M10 which placed it at U+0363, UTC-85-M13 which moved it to its encoded position at U+034F. Of particular interest is the text of this motion:

> **[84-M10] Motion**: The UTC accepts the COMBINING GRAPHEME JOINER with a suggested code point assignment of U+0363. The COMBINING GRAPHEME JOINER will be used to indicate that the adjacent character(s) are part of a single grapheme in terms of grapheme production. It will behave in general like a virama and for line breaking, it will behave like a glue character. It will be a combining mark with a canonical class of zero. The UTC discourages its use for graphical effects, such as for circled numbers. [L2/00-156]
>
> Moved by Mark Davis, seconded by Tex Texin
> **10** for (Basis, Compaq, HP, IBM, Microsoft, NCR, Peoplesoft, Progress, Sybase, Unisys)
> **1** against (Apple)
> **1** abstain (Justsystem)

However, a look at https://www.unicode.org/Public/3.2-Update/LineBreak-3.2.0.txt reveals that it was lb=CM in 3.2, seemingly against the will of the UTC, although a look at UTN54 shows that UAX #14 claimed it was lb=GL from the start: https://www.unicode.org/notes/tn54/alba-1.html?v=4.1.0&base=3.1.0#p219.1.

At that time, the Combining Grapheme Joiner was meant to join graphemes, as shown by this text from UAX #28 Unicode 3.2: https://www.unicode.org/reports/tr28/tr28-3.html#13_2_layout_controls:

> The combining grapheme joiner is used to indicate that adjacent characters belong to the same grapheme cluster. Grapheme clusters are sequences of one or more encoded characters that correspond to what users think of as characters.

However, a note in UAX #28 Unicode 3.2 prefigures the plot twist:

> Note: The rules for default grapheme cluster boundaries, default word boundaries and default sentence boundaries are in the process of being superseded by a new Unicode Technical Report #29, Text Boundaries.

April 2002–April 2003: The end of grapheme joining

The proposed draft of UTR #29 indeed took the CGJ into account. The first draft however no longer did. The modifications section simply states *Simplified grapheme cluster*. The UTC decision approving the progression to draft is only slightly more informative; one presumes that the comments received during discussion prompted this simplification:

> **[91-C9] Consensus:** Advance Proposed Draft Unicode Technical Report #29 Text Boundaries to Draft Unicode Technical Report #29 Text Boundaries after incorporating comments received during discussion and review by the Editorial Committee. [L2/02-164, 175]

At the same time, it was decided that the UTR would be a UAX:

> **[91-M2] Motion:** Draft Unicode Technical Report #29 Text Boundaries is to be placed on track to become a Unicode Annex for 4.0.
>
> Moved by Ken Whistler, seconded by V.S. Umamaheswaran
> **12** for (Adobe, Apple, Basis, HP, IBM, Justsystem, Microsoft, Oracle, PeopleSoft, Sun, Trigeminal, Unisys)
> **0** against
> **2** abstain (RLG, Compaq)

Document L2/03-026[1] offers a hint of the deliberations of UTC-91:

> But in the meantime, the UTC decided to narrow the scope of grapheme clusters to a clear core, basically:
>
> > (<hangul syllable> | <base>) <non-spacing mark>*
> > [and the name is changed to "default grapheme cluster"]

It is striking to note that UTC-91 took place a month after the publication of Unicode 3.2: CGJ had only been joining graphemes for a month when UTC decided that it should not do so.

Document L2/03-026 points out that now that the combining grapheme joiner no longer joins graphemes, other statements made by Unicode 3.2 about marks enclosing sequences joined by CGJ no longer work as stated. The UTC decides as follows:

> [94-M1] **Motion:** When a sequence of default grapheme clusters are linked by a combining grapheme joiner, an enclosing mark may be rendered as enclosing the entire sequence. The target of the enclosing mark is the preceding grapheme cluster or sequence of default grapheme clusters linked by grapheme joiner. The intent of the usage of enclosing marks is on free-standing default grapheme clusters or grapheme clusters linked by grapheme joiner. Clarify this in section 7.7 of the Unicode Standard 4.0. The rendering of enclosing marks in complex cases should have many caveats.[L2/03-026, 027, 028]
>
> Moved by Mark Davis, seconded by Ken Whistler
> **11** for (Adobe, Apple, Basis, HP, IBM, India MIT, Microsoft, PeopleSoft, RLG, Sun, Sybase)
> **0** against
> **2** abstain (Justsystem, Oracle)

Unicode 4.0 was released shortly after UTC-94. However, a look at Section 15.2 of *The Unicode Standard*, Version 4.0 shows a subtlety; the behaviour alluded to by UTC-94-M1 is described as legacy:

> For rendering, the combining grapheme joiner is invisible. However, some older implementations may treat a sequence of grapheme clusters linked by combining grapheme joiners as a single unit for the application of enclosing combining marks.

That version prefigures a use in collation, but does not elaborate, nor does it mention normalization conventions:

> U+034F COMBINING GRAPHEME JOINER is used to indicate that adjacent characters are to be treated as a unit for the purposes of language-sensitive collation and searching. In

language-sensitive collation and searching, the combining grapheme joiner should be ignored unless it specifically occurs within a tailored collation element mapping.

June 2003–July 2006: CGJ in Jerusalem[2]

At the beginning of June 2003, Peter Constable posted L2/03-195, proposing that 14 Hebrew combining marks be duplicated due to CCC issues affecting Biblical Hebrew.

Later that month, this proposal was brought up in a thread on the Unicode mailing list, which had started with Tibetan vowels: https://unicode.org/mail-arch/unicode-ml/y2003-m06/0328.html, https://unicode.org/mail-arch/unicode-ml/y2003-m06/0337.html.
A long discussion ensued on how this duplicate encoding could be avoided; a number of CCC=0 characters were suggested, but many of them had other properties which were troublesome, until the mostly useless CGJ was found[3]. Technical discussion on the Unicode mailing list continued into July, and its conclusions distilled into three documents were presented to the UTC, which decided as follows:

> [96-C20] Consensus: Add text to Unicode 4.0.1 which points out that combining grapheme joiner has the effect of preventing the canonical re-ordering of combining marks during normalization. [L2/03-235, L2/03-236, L2/03-234]
>
> [96-A72] Action Item for Ken Whistler: Draft language for consensus 96-C20 (on the effect of combining grapheme joiner to prevent canonical re-ordering of combining marks during normalization) for inclusion into Unicode 4.0.1 and create a FAQ describing this effect as well. [L2/03-235, L2/03-236, L2/03-234]

Draft text was dutifully presented to UTC-97:

> [97-A36] Action Item for Ken Whistler, Editorial Committee: Update document L2/03-403 on combining grapheme joiner to reflect that this is a mechanism that should be used in specific circumstances and incorporate other comments made during the meeting.

Meanwhile in ISO/IEC JTC 1/SC 2/WG 2, the CGJ was suggested as a way to distinguish Umlaute from trémas in bibliographic collation: https://www.unicode.org/wg2/docs/n2819.pdf.

In UTC, improved documentation of the use of CGJ in collation was requested:

> **[100-C31] Consensus:** Change the collation algorithm so that: [L2/04-311,L2/04-277,L2/04-319]
>
> A. All completely ignorable characters interrupt contractions.
> B. U+0600 ARABIC NUMBER SIGN and U+2062 INVISIBLE TIMES and like characters (U+0600..U+0603, U+06DD, U+2061..U+2063) are not completely ignorable.h
> C. Document in the UCA the general use of combining grapheme joiner to break contractions or in tailoring to have special effects.
>
> **[100-A74] Action Item for** Mark Davis, Ken Whistler, Editorial Committee: Update the Unicode collation algorithm and data for consensus 100-C31 (handling ignorable characters, invisible characters, and the use of combining grapheme joiner to break contractions or to have special effects). and review the description of combining grapheme joiner in the standard.

A similar issue to the Hebrew one came up later regarding Latin, and more text was mandated:

> **[103-A50] Action Item for** Ken Whistler, Editorial Committee: Update the text of the standard for version 5 on the use of combining grapheme joiner in Latin script diacritics as suggested in L2/05-094.

The FAQ entries about CGJ were added to https://www.unicode.org/faq/char_combmark.html at the end of 2004 or at the beginning of 2005[4].

Eventually, the updated core specification was published for Unicode Version 5.0, https://www.unicode.org/versions/Unicode5.0.0/ch16.pdf#G24326

This text has not substantially changed since then:
https://www.unicode.org/versions/Unicode15.0.0/ch23.pdf#G24326
https://www.unicode.org/versions/Unicode16.0.0/core-spec/chapter-23/#G24326

However, between Unicode 4.0 and Unicode 5.0, CGJ had become lb=GL in Unicode Version 4.1. This change was decided by UTC-99-C4, based on a document which pointed out an inconsistency between UAX #14 and the data file:

> [99-C4] Consensus: Change the linebreak class of combining grapheme joiner from combining (CM) to glue (GL) in the data file.[L2/04-123]

> [99-A8] Action Item for Ken Whistler, Editorial Committee: Update the linebreak class of combining grapheme joiner from CM to GL in the Unicode Standard Annex #14: Line Breaking Properties data file. [L2/04-123]

As CGJ no longer joined anything by that point, it is clear that UTC-99 had decided incorrectly, and that the inconsistency should have been fixed in the other direction, by leaving it lb=CM and correcting the UAX.

Later CGJ developments, including its usage in AMTRA, to appear in upcoming volumes of Scherer et al., eds, *Studies in Character Encoding History*.

Line breaking and CGJ today

Line_Break=GL makes no sense here today, as CGJ has not glued anything for twenty years (as we have seen above, it only glued things for a year, or more realistically given that implementers who implement the newest fanciest standardized behaviours also tend to be aware of current developments in standardization, for a month). Indeed since Unicode Version 4.0, the Standard reads, *sub* CGJ and Joiner Characters:

> The combining grapheme joiner must not be confused with the zero width joiner or the word joiner, which have very different functions. In particular, inserting a combining grapheme joiner between two characters should have no effect on their ligation or cursive joining behavior. Where the prevention of line breaking is the desired effect, the word joiner should be used.

While it is used outside of a combining character sequence to break contractions in collation, that 1. has nothing to do with line breaking and 2. is breaking things rather than gluing them anyway.

In fact the FAQ, which, the reader will recall, states that the combining grapheme joiner does not join graphemes, claims (incorrectly since Unicode 4.1) that

> [CGJ] has no impact on line breaking, except that as for other [combining marks] https://www.unicode.org/glossary/#combining_mark), it should be kept with its base when breaking a line.

Let us make that FAQ entry correct again.

[1] The internal date, 2002-01-29, is baffling for a document in the 2003 register; L2/04-001 records its submission on 2003-01-30; we must assume that the internal date of L2/03-026 is erroneous, and that the document is from 2003-01-29.

[2] יְרוּשָׁלַֽם.

[3] Ken Whistler suggested ZWJ https://unicode.org/mail-arch/unicode-ml/y2003-m06/0343.html;
Karljürgen Feuerherm suggested pseudo-consonants
https://unicode.org/mail-arch/unicode-ml/y2003-m06/0347.html;
Peter Constable found ZWJ groanable, and pointed out architectural issues
https://unicode.org/mail-arch/unicode-ml/y2003-m06/0358.html;
Ken Whistler suggested U+17B4 KHMER VOWEL INHERENT AQ, as well as ZWNJ and ZWNBSP
https://unicode.org/mail-arch/unicode-ml/y2003-m06/0391.html;
Jony Rosenne suggested RLM https://unicode.org/mail-arch/unicode-ml/y2003-m06/0393.html;
Ken Whistler suggested WJ https://unicode.org/mail-arch/unicode-ml/y2003-m06/0396.html;
finally, Ken came up with the idea of CGJ: https://unicode.org/mail-arch/unicode-ml/y2003-m06/0407.html.

[4] Compare https://web.archive.org/web/20041010040057/https://www.unicode.org/faq/char_combmark.html and https://web.archive.org/web/20050205223246/https://www.unicode.org/faq/char_combmark.html.

6.4 UAX #14 WJ and SY in LB15b but not in LB15a [#320]

Recommended UTC actions

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

Feedback (verbatim)

Date/Time: Thu Aug 01 09:18:31 CDT 2024
ReportID: ID20240801091831
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #14: Unicode Line Breaking Algorithm

https://www.unicode.org/reports/tr14/#LB15b
Version: Unicode 15.1.0
Date: 2023-08-15
Revision: 51

Location: LB15a, LB15b

I found the following document which describes these new rules:
https://www.unicode.org/L2/L2023/23063-break-quot-mark.pdf

Reading through it, it seems that the inclusion of WJ and SY in LB15b (but not in LB15a) might have been accidental, and not really intended by the author. Perhaps it is an artifact of importing the rules from another representation.

Regarding WJ, it seems strange that SP×Pf×WJ, i.e. that WJ should act-at-a-distance across the quotation mark. If somebody actually used WJ after Pf, they probably intended to prevent a line break to the right of Pf, not to the left. Yes, such WJ is redundant in the current version of the algorithm, but implementations deviate (especially Far Eastern implementations tend to allow line breaks much more often), so the WJ might be there in the text for a valid real-world reason. Given that SP×Pf×WJ doesn't seem to have any merit for French (somebody able to type WJ in French could just type <SP,WJ,Pf>, after all), I believe WJ should not be included in LB15b. Including it in LB15b penalizes a user who is mindful about their line breaks (explicitly using WJ), for the sake of somebody who is not careful enough to put the WJ at the correct place.

Regarding SY, the slash »/« is often used in Unix paths, such as »/usr/bin«. I am not familiar with the particulars of French usage, but does it occur « comme ça »/ frequently enough (without a space before the slash) to merit inclusion in LB15b? If it does, then it probably also occurs with the same frequency /« comme ça », so it doesn't make sense to include it in LB15b but not in LB15a.

If WJ and SY are included in LB15b purely for a technical reason (to ease implementations using a particular kind of software), and that reason is important enough to merit complicating the user-facing semantics of WJ, then this should probably be stated in the text.

Background information / discussion

The construction of the rule is documented in page 4 of document L2/23-063 cited by the submitter: the set (WJ | CL | QU | CP | EX | IS | SY) was chosen to cover a final quotation mark occurring before a prohibited break, prohibited breaks being a good heuristic for being somewhere final. This is repeated in the current description of LB15a. In a sense this does mean that they are an artifact of importing the rules from another representation, namely from the description of the rule.

Only theoretical concerns are presented in this feedback, rather than issues with the behaviour of the current algorithm on real text, so no action is required.

6.5 UAX #14 line break via grapheme breaks & lb of first char: does not work [#322]

Recommended UTC actions

1. **Note**: The PAG should consider feedback ID20240805055322 as part of prior action item 160-A73 and the umbrella action item 170-A69a.

Feedback (verbatim)

Date/Time: Mon Aug 05 05:53:22 CDT 2024
ReportID: ID20240805055322
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #14: Unicode Line Breaking Algorithm

https://www.unicode.org/reports/tr14/#Examples
Version: Unicode 15.1.0
Date: 2023-08-15
Revision: 51

Location: 8.2 Examples of Customization, Example 7

Problematic text:

The tailoring can be accomplished by first segmenting the text into grapheme clusters according to the rules defined in UAX #29, and then finding line breaks according to the default line break rules, as follows: After applying the mandatory line break rules, give each grapheme cluster the line breaking class of its first code point.

Explanation:

This text was changed recently to avoid recommending a non-conforming tailoring:

https://www.unicode.org/L2/L2022/22244-utc173-properties-recs.pdf

I agree that with this change the UAX no longer formally contradicts itself, but it still doesn't mean the approach gives sensible results.

Here is an example of misbehavior if the wording of the problematic text is taken at face value:

<U+1112,U+1161,U+11AB, U+1100,U+1173,U+11AF> (literally: 한글)

These are two Korean syllables, each composed of three code points: a leading consonant, a vowel, and a trailing consonant. Segmenting into grapheme clusters will produce two clusters, one for each syllable. If, as the text suggests, we give each cluster the line breaking class of its first code point, this would give each cluster the incorrect line breaking class JL (the class for leading consonants) instead of the correct H3 (the class for three-component syllables). Since the line breaking algorithm does not allow line breaks between leading consonants, there will be no line breaks in the entire sequence.

Now these are just two Korean syllables, so the missed line breaking opportunity between them may not matter, but the same logic holds for an arbitrary long sequence of Korean syllables, potentially forbidding any line breaks in a long run of Korean text.

Another possible example of misbehavior is a sequence of several Emoji flags, e.g. <RI,RI, RI,RI>. Segmenting into grapheme clusters will group together pairs of Regional Indicators, then giving each pair the line breaking class RI will result in prohibition of line breaks between pairs-of-pairs. This is probably not what was intended.

I have not worked out the details for cases of Grapheme_Cluster_Break=Prepend, but they should probably be verified, and then again for each new update of UAX #29, because the segmentation logic tends to get more and more complicated over the years.

In summary, I think it is better not to mislead the reader that it is a simple matter to tailor the line breaking algorithm to work sensibly on grapheme cluster boundaries. Either a complete working solution should be offered, or the reader should be warned of the existence of potential problems.

Date/Time: Mon Aug 05 06:23:35 CDT 2024
ReportID: ID20240805062335
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #14: Unicode Line Breaking Algorithm

https://www.unicode.org/reports/tr14/#Examples
Version: Unicode 15.1.0
Date: 2023-08-15
Revision: 51

Location: 8.2 Examples of Customization, Example 7

I would like to add to the feedback that I submitted on this topic a few minutes ago.

Maybe a workable approach would be:

1. Run both the segmentation algorithm and the line breaking algorithm in parallel, unmodified.
2. Delete the line breaking opportunities that happen to fall within grapheme clusters.

If 2. deletes a non-tailorable line breaking opportunity (produced by rules LB2-LB12), then this means the problem is impossible to solve in the first place.

It would be nice to also verify that it is impossible for 2. to delete too many line breaking opportunities, producing long runs of legitimate text without line breaks.

Background information / discussion

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

6.6 Incoherent documentation of the LB assignment of U+FE10 [#331]

Recommended UTC actions

1. **No Action**: This has been fixed editorially.

Feedback

From Bruno Haible by direct email to the editor:

```
Unset
Hi,

I think there's a mistake in https://www.unicode.org/reports/tr14/tr14-53.html:
U+FE10 is listed as belonging to both class CL and class NS. This cannot be
the case, since any character has only one line breaking class.

The LineBreak.txt lists it in class CL. This means, the mistake is in the
description of class NS.

Best regards,

        Bruno
```

Background information / discussion

Indeed it is CL, and we recommended that it be made CL for Unicode 16 in PAG issue #266 “On the Line_Break assignment of three vertical presentation forms”, and UTC made it CL. This is an editorial issue.

7. Collation

7.1 merge CollationTest.html contents into UTS #10 [#324]

Recommended UTC actions

1. **Consensus**: Merge the contents of CollationTest.html into UTS #10 and omit CollationTest.html from /Public/UCA/. For Unicode 17.0. See L2/24-224 item 7.1.
2. **Action Item** for Markus Scherer, PAG: Merge the contents of CollationTest.html into UTS #10 and omit CollationTest.html from /Public/UCA/. For Unicode 17.0. See L2/24-224 item 7.1.

PAG input

From Markus Scherer, PAG

We usually document data files and their formats, including test data for segmentation and IDNA, in the respective UAX/UTS together with varying degrees of details in the data files themselves. For the collation test data, we have a separate file, CollationTest.html, with a brief description. This looks like an anachronism, and adds some friction to the release process.

I propose that we merge the contents of this file into UTS #10 section 12 Data Files.

Background information / discussion

The collation test data, and this separate documentation page, goes back to 2002:
https://www.unicode.org/reports/tr10/tr10-9.html#Test

8. Regex

8.1 UTS #18 misleading about Any/Assigned/ASCII vs. General_Category [#340]

Recommended UTC actions

1. **Action Item** for Mark Davis, PAG: In UTS #18, change the discussion of Any/Assigned/ASCII to clarify that these are not General_Category values. See L2/24-224 item 8.1.

Feedback (verbatim)

Date/Time: Mon Oct 21 14:42:36 CDT 2024
ReportID: ID20241021144236
Name: Huáng Jùnliàng
Report Type: Error Report
Opt Subject: UTS #18

In section 1.2.5, there is a table containing General Category Property values and three star entries, Any, Assigned and ASCII. Although there is a note that starred entries in the table are not part of the enumeration of General_Category values, it may still be a little bit confusing as one browser engine maintainer interprets [1] that ASCII belongs to General Category:

Yes, but that means that they are not part of the enumeration of values and not that they don't belong to that category. I.e. they are not listed as being part of that categories in UnicodeData.txt.

Can we we improve the text and/or the table layout to clarify that Any, Assigned and ASCII are not a General_Category property value?

[1]: https://issues.chromium.org/u/0/issues/373759990#comment5

Background information / discussion

https://www.unicode.org/reports/tr18/#General_Category_Property
“The General_Category property values are listed below.”

While for real gc values one can do either `[:Lo:]` or `[:gc=Lo:]` the latter does not work for Any/Assigned/ASCII.

PAG suggests moving these three out of the General_Category property values table and inserting another heading (1.2.5.1) (maybe titled “Other Useful Categories”) between that table and the explanation of these special pseudo-properties.

This text

> Starred entries in the table are not part of the enumeration of General_Category values. They are explained below.

could be changed to something like

> The following table contains other categories that are useful in regular expressions but not directly enumerated in the UCD.

The row for ASCII could benefit from a note like this:

> This category includes all ASCII control codes including newline.

9. Emoji

9.1 Is “component” a value of the RGI_Emoji_Qualification property? [#336]

Recommended UTC actions

1. **Consensus**: In UTS #51 ED-28, add a new property value with long name "Standalone_Component" and short name "component" corresponding to the "component" field value in the associated data file. For Unicode 17.0. See L2/24-224 item 9.1.
2. **Action Item** for Mark Davis, ESR: In UTS51 ED-28, add a new property value with long name "Standalone_Component" and short name "component" corresponding to the "component" field value in the associated data file. For Unicode 17.0. See L2/24-224 item 9.1.
3. **Action Item** for Mark Davis, ESR: In the emoji-test.txt header comments, make the appropriate changes for the new property value Standalone_Component=component. For Unicode 17.0. See L2/24-224 item 9.1.

PAG input

From Markus Scherer, PAG

See https://unicode.org/reports/tr51/#def_rgi_emoji_qualification

> This is an enumerated property of strings, defined by the emoji-test.txt file [...]. It assigns one of the three values [...] Fully_Qualified, Minimally_Qualified, Unqualified

vs. https://www.unicode.org/Public/emoji/latest/emoji-test.txt
which has data with *four* Status values, including

```
Unset
#       component           – an Emoji_Component,
#                             excluding Regional_Indicators, ASCII, and non-Emoji.
```

emoji-test.txt has 9 characters (no strings) with Status=component: skin-tone U+1F3FB..U+1F3FF and hair-style U+1F9B0..U+1F9B3.

Emoji_Component in UCD emoji-data.txt has 146 code points including those 9.

For someone implementing the RGI_Emoji_Qualification property, should they ignore the Status=component entries?
If so, then we should document this clearly in UTS #51 and in a future version of UTS #18.

Or should we modify the definition of the property to include everything that emoji-test.txt has?

10. Math

10.1 MathClass of U+22A5 ⊥ UP TACK is R=Relation, should be N=Normal [#334]

Recommended UTC actions

1. **No Action now**: This will be addressed in a future revision of UTR #25.

Feedback (verbatim)

Date/Time: Thu Sep 19 09:19:51 CDT 2024
ReportID: ID20240919091951
Name: Malo
Report Type: Error Report
Opt Subject: MathClass

As of Unicode 15, in MathClass documents (https://www.unicode.org/Public/math/revision-15/*), the character U+22A5 ⊥ UP TACK is classified as a Relation (R). This is contradictory with its use as a value (class N for Normal) in many fields such as logic and type theory (where it is often referred to as "bot," or "bottom"). In fact, U+22A4 ⊤ UP TACK ("top"), which is used along with top in those fields, is classified as Normal (N).

This is likely due to a confusion with the homoglyphic perpendicular symbol (U+27C2 ⊥ PERPENDICULAR), which is correctly classified as a Relation (R). It is this exact difference between bot being used as a value and the perpendicular sign being used as a relation that lead to the introduction of those two distinct characters in Unicode, according to this 2003 draft: https://www.unicode.org/L2/L2003/03194-math-letterlike.pdf.

As a final note, bot was initially properly classified as Normal (N) in Unicode 9 (https://www.unicode.org/Public/math/revision-09/MathClass-9.txt), but this changed with Unicode 11. If this change was intentional, I think this oddity deserves a comment in the MathClass files to inform the reader that this is not a mistake, and a short explanation.

Background information / discussion

UnicodeData.txt

```
Unset
22A4;DOWN TACK;Sm;0;ON;;;;;N;;;;;
22A5;UP TACK;Sm;0;ON;;;;;N;;;;;
```

https://www.unicode.org/Public/math/revision-15/MathClassEx-15.txt

```
Unset
22A4;N;T;top;ISOTECH; top;DOWN TACK
22A5;R;⊥;bottom;ISOTECH; bottom ;UP TACK
```

The feedback represents what would be done in an "ideal" world, where each character is cleanly related to a single operator. However, historically 22A5 has been mapped to both and 27C2 was not used in some entity sets. The question remains, what should we put in the mathclass.txt file, given that we do have the disunification. Anything we decide will have to be part of a larger discussion of our plans to update UTR #25.

From a reply by **David Carlisle** to a request for comments on this issue (lightly edited/formatted):

> Classic tex fonts use the same glyph for \perp and \bottom (but with different math spacing) so some conflict here is inevitable
>
> StackExchange answers by David Carlisle:
>
> https://tex.stackexchange.com/questions/102184/difference-of-perp-and-bot/102187#102187
>
> https://tex.stackexchange.com/questions/118605/is-there-a-difference-between-bot-and-perp-when-they-are-used-in-exponent/118620#118620
>
> The first sentence of the second one is the main answer:-)
>
> However, Unicode does offer two codepoints so there is a possibility of separating them
>
> The primary support for OpenType Unicode math fonts in latex is the unicode-math package which assigns
>
> - \UnicodeMathSymbol{"022A5}{\bot }{\mathord}{bottom}%
> - \UnicodeMathSymbol{"027C2}{\perp }{\mathrel}{perpendicular}%
>
> so
>
> - U+22A5 is \bot with no math spacing (N in mathclass-15 notation)
> - U+27C2 is \perp with math relation spacing (R in mathclass-15 notation)
>
> For historical reasons HTML/MathML entity set define & bottom; & bot; & perp; & UpTee; all to be U+22A5 and assigns no spacing to it so it is \mathord (N)
>
> No html entity name or mathml spacing is assigned to U+27C2
>
> ...
>
> So in an ideal world we would have
>
> - U+22A5 would be \bot and have no math spacing
> - U+27C2 would be \perp and have R spacing
>
> But that isn't quite the world we live in.

11. Authorize proposed updates

Recommended UTC action

1. Consensus: Authorize proposed updates of UAX #14, UTS #10, and UTS #51, for Unicode 17.0.