UTC #173 properties feedback & recommendations

Markus Scherer / Unicode properties & algorithms group, 2022-oct-27

Participants

The following people have contributed to this document:

Markus Scherer (chair), Josh Hadley (vice chair), Asmus Freytag, Elango Cheran, Ken Whistler, Mark Davis, Ned Holbrook, Peter Constable, Rick McGowan, Robin Leroy

UCD

UCD1: stability policy page should define "domain" etc.

Recommended UTC actions

1. Action item for Asmus Freytag, EDC: Propose changes to the stability policy page to help readers understand "domain" and maybe other terms; by referencing the glossary. See L2/22-244 item UTC1.

Feedback (verbatim)

Date/Time: Thu Sep 8 15:38:26 CDT 2022
Name: Asmus/
Report Type: Website Problem
Opt Subject:

https://www.unicode.org/policies/stability_policy.html

This page should cite definitions of terms such as "domain". This could be done either by citing the location of their formal definition of, perhaps better by making them glossary links and then ensuring that any glossary item always cites the formal definition its based on.

This came up in the context of adding the "domain stability" which introduces the word "domain" which perhaps is not in everybody's active vocab.

New Scripts

Script1: Ol Onal

L2/22-151 Proposal to encode the Ol Onal script

Recommended UTC actions

1. No action. Several PAG members have reviewed the proposal and provided feedback to the SAH.

Late feedback

Markus provided the following feedback shortly before the UTC meeting:

The proposal has been updated to show gc=Lo for the letters, but the text before table 7 still says "30 upper case letters".

This text also says " 1 sign as other letter (Table 8)" although that table contains three "signs" (two Mn, one Lo). None of these tables shows the abbreviation sign (Po).

"This document proposes 41 characters in total" -- but it's really 44 characters.

Is it normal to not unify the dot-above and dot-below with the U+03xx ones?

Collation: I suspect that we will treat the digits, combining marks, and punctuation as usual, rather than all in one primary sequence. The glottal stop can probably remain as the highest-sorting letter.

Script metadata: ID Usage should be "Exclusion", not "Limited_Use".

Script2: Yo Lai Tay

L2/22-208 Final Proposal to encode the Yo Lai Tay Script

Recommended UTC actions

1. No action. Several PAG members have reviewed the proposal and provided feedback to the SAH.

Late feedback

Markus provided the following feedback shortly before the UTC meeting:

“Lai Tay, meaning the script of Tay, is one of the two Brahmic scripts used to write the Tai Yo language. Since Tay or Tai is the endonym of various Tai ethnic groups, the script is hereafter called Yo Lai Tay.”

I don't see how this follows — please rephrase/explain the "Since ... is hereafter called ..." sentence. If Lai Tay is the script of Tay, and we want to be more specific about the script of the Tai Yo language, shouldn't the name be Lai Tay Yo or Lai Tai Yo? Why does the "Yo" move to the beginning?

3.5 Punctuation & 3.6 Numerals -- why does the proposal recommend compatibility characters (fullwidth ASCII U+FFxx) rather than normal ones? The regular characters should render fine in a vertical context.

Script metadata:

- The sample character (LETTER LOW KO) is not very distinct. I suggest LETTER HIGH FO or LETTER HIGH XO.
- ID Usage should probably be "Exclusion" as usual.

Text Segmentation

Seg1: UAX 14 two incorrect examples about applying GCB

Recommended UTC actions

1. Action Item for Robin Leroy, PAG: Make changes to the discussion of possible tailorings in UAX #14 as described in L2/22-244 item Seg1, for Unicode Version 15.1.
2. Action Item for Rick McGowan: Post a Public Review Issue for a proposed update of UAX #14, for Unicode version 15.1.

Changes for “problematic text 1”:

8.1 Types of Tailoring

~~Beyond these three straightforward customization steps, it is always possible to augment the algorithm itself—f~~For example, ~~by providing~~ specialized rules could be added to recognize and break common constructs, such as URLs, numeric expressions, and so on. Such open-ended customizations place no limits on possible changes, other than the requirement that ~~characters with normative line breaking properties~~ non-tailorable line breaking rules be correctly implemented. This means that whatever changes are made must be equivalent to changes to the line breaking assignments of tailorable line breaking rules, and to alteration, removal, or addition of rules applied after rule LB12.

8.2 Examples of Customization, Example 7

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 ~~giving~~ each grapheme cluster the line breaking class of its first code point.

Changes for “problematic text 2”:

An example of a grapheme cluster that would be split by the default line break rules is a ~~Zero Width~~ Space followed by a combining mark.

Feedback (verbatim)

Date/Time: Wed Sep 21 07:53:00 CDT 2022
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #14: Unicode Line Breaking Algorithm

https://www.unicode.org/reports/tr14/#Examples
Version: Unicode 15.0.0
Date: 2022-08-16
Revision: 49

Location: 8.2 Examples of Customization, Example 7

Problematic text 1:

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, giving each grapheme cluster the line breaking class of its first code point.

Explanation:

This tailoring wouldn't be conforming in edge cases. Suppose the text is <CR, LF, LF>. After applying UAX #29, this becomes two grapheme clusters <CR, LF> and <LF>, with first code points <CR> and <LF>, respectively. Then default line breaking rules would prevent a line break between these, contrary to the conformance requirement for a mandatory break.

Problematic text 2:
An example of a grapheme cluster that would be split by the default line break rules is a Zero Width Space followed by a combining mark.

Explanation:
According to the latest version of UAX #29, Zero Width Space followed by a combining mark does not form one grapheme cluster (ZWSP has Grapheme_Cluster_Break=Control).

Background information / discussion

Tailorings cannot override mandatory rules. “Problematic text 1” seems to suggest doing so. Consider a general description of tailorings to be logically applied only after mandatory rules.

About the proposed rewording of section 8.1: This used to be three straightforward steps; but now it is two, and the second one encompasses that « beyond ». The use of the word « normative » to mean non-tailorable is inappropriate since Unicode Version 5.0.0, see 105-C37.

“Problematic text 2”: Change to a different example of a grapheme cluster that would be split by the default line break rules — legacy space behavior of 0020 0338 mentioned as a possible tailoring.

Seg2: GCB & WB inconsistent for RI ZWJ RI RI

Recommended UTC actions

1. No action. This has been reported before (L2/22-019 item F3) and will be handled as part of the work on action item 170-A69a about inconsistencies between different segmentation algorithms.

Feedback (verbatim)

Date/Time: Tue Jul 26 06:12:36 CDT 2022
Name: Oliver Kuederle
Report Type: Error Report
Opt Subject: UAX #29, 14.0.0

In Unicode Standard Annex #29 (Unicode Text Segmentation), v14.0.0, there appears to be an inconsistency between the grapheme cluster boundary rules and the word boundary rules. Specifically, rule GB13 states that a pair of regional indicators may not be broken. If a zero-width joiner precedes a regional indicator, this matches [^RI] and the counting of RI thus starts again. There is no exception for ZWJ in this specific case.

For word boundaries, however, rule WB4 will cause an RI before a ZWJ to maintain its count (WB15/WB16). So the following sequence will break differently for graphemes and for words:

RI ZWJ RI RI

Following the grapheme rules, this will lead to:

RI × ZWJ ÷ RI × RI

And for word rules, this will lead to:

RI × ZWJ × RI ÷ RI

The word rules will therefore break a grapheme cluster which is probably not intended.

Seg3: UAX 14 third style vs. word breaks / syllable boundaries

Feedback for closed PRI #446

Recommended UTC actions

1. Action item for Robin Leroy, PAG: Change UAX #14 section 3.1 “third style of line breaking” as suggested by Norbert Lindenberg. See L2/22-244 item Seg3. For Unicode 15.1.

Feedback (verbatim)

Date/Time: Sun Apr 10 20:12:11 CDT 2022
Name: Norbert Lindenberg
Report Type: Error Report
Opt Subject: UAX 14

Section 3.1 of UAX 14 has the following description of the South East Asian style of line breaking: “The third style is used for scripts such as Thai, which do not use spaces, but which restrict word breaks to syllable boundaries, whose determination requires knowledge of the language comparable to that required by a hyphenation algorithm. Such an algorithm is beyond the scope of the Unicode Standard.”

This description is odd in not starting out with line breaking, but with word breaks, whose relevance to line breaking is not explained. The problem statement I usually hear is that Thai, Lao, Khmer, and Myanmar allow line breaks only at word boundaries, but do not mark word boundaries in any way, so that they have to be determined by higher-level algorithms, typically based on dictionaries. See, for example, the W3C layout requirements:

https://www.w3.org/International/sealreq/thai/#h_line_breaking
https://www.w3.org/International/sealreq/lao/#h_line_breaking
https://www.w3.org/International/sealreq/khmer/#h_line_breaking

The comparison with hyphenation algorithms is also questionable, as the complexity of hyphenation algorithms can vary substantially between languages.

Finally, Thai does use spaces to separate phrases.

I propose replacing the text quoted above with "The third style is used for scripts such as Thai, which allow line breaks only at word boundaries, but do not mark word boundaries in any way, so that the determination of line break opportunities requires language dependent text analysis. Algorithms and data for such analysis are beyond the scope of the Unicode Standard."

Seg4: EGYPTIAN HIEROGLYPH V011D should have lb=OP not AL

Recommended UTC actions

1. Consensus: Change the line breaking class of U+1342F EGYPTIAN HIEROGLYPH V011D ( ) from AL to OP, for Unicode Version 15.1.
2. Action item for Ken Whistler, PAG: Change the Line_Break value of U+1342F EGYPTIAN HIEROGLYPH V011D ( ) from AL to OP, for Unicode Version 15.1.

Feedback (verbatim)

Date/Time: Mon Aug 22 14:40:56 CDT 2022
Name: Charlotte Buff
Report Type: Error Report
Opt Subject: Line break class of U+1342F

U+1342F EGYPTIAN HIEROGLYPH V011D currently has Line_Break=Alphabetic (AL) in the preliminary data files for Unicode 15. Because this hieroglyph is the start of a cartouche, it should have Line_Break=Open_Punctuation (OP) instead. This property value is shared by all other hieroglyphs with a similar function (U+13258..U+1325A, U+13286, U+13288, U+13379).

Background information / discussion

These characters were approved by consensus170-C12, which cites L2/21-248, which has U+1342F as AL (so the error is in the proposal, not a clerical error on the way to the UCD). On Charlotte Buff's comment about « other hieroglyphs with a similar function » cf. ibid., Table 13, as well as [:lb=OP:]&[:Script=Egyp:] and [:lb=CL:]&[:Script=Egyp:].

Seg5: UAX 29 GCB regexes missing CR & LF

Recommended UTC actions

1. Action item for Josh Hadley, PAG: In UAX #29 GCB table 1c Regex Definitions change the definition of crlf to "CR LF | CR | LF" as suggested by Rossen Mikhov, for Unicode 15.1. See L2/22-244 item Seg5.
2. Action Item for Rick McGowan: Post a Public Review Issue for a proposed update of UAX #29, for Unicode version 15.1.

Feedback (verbatim)

Date/Time: Fri Sep 16 09:45:20 CDT 2022
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #29: Unicode Text Segmentation

https://www.unicode.org/reports/tr29/#Table_Combining_Char_Sequences_and_Grapheme_Clusters
Version: Unicode 15.0.0

Date: 2022-08-26
Revision: 41

Location: Table 1b. Combining Character Sequences and Grapheme Clusters

Problematic text:
legacy grapheme cluster: crlf | Control | legacy-core legacy-postcore*
extended grapheme cluster: crlf | Control | precore* core postcore*

Possible correction:
legacy grapheme cluster: crlf | CR | LF | Control | legacy-core legacy-postcore*
extended grapheme cluster: crlf | CR | LF | Control | precore* core postcore*

Alternative possible correction:
(In table 1c) crlf := CR LF | CR | LF

Explanation:
Looks like a simple editorial omission.
With this minor correction, the regular expressions exactly correspond to
the specification of the rules GB1-GB999.

Background information / discussion

See “Table 2. Grapheme_Cluster_Break Property Values”: Control excludes CR and LF.

Seg6: UAX 29 testing GCB does need to look at more than two adjacent characters

Recommended UTC actions

1. Action item for Josh Hadley, PAG: In UAX #29 section 7 Testing, after “Testing two adjacent characters is insufficient for determining a boundary” remove “except for the case of the default grapheme clusters”; for Unicode 15.1. See L2/22-244 item Seg6.

Feedback (verbatim)

Date/Time: Fri Sep 16 09:51:21 CDT 2022
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #29: Unicode Text Segmentation

https://www.unicode.org/reports/tr29/#Testing
Version: Unicode 15.0.0
Date: 2022-08-26
Revision: 41

Location: 7 Testing

Problematic text:
Note: Testing two adjacent characters is insufficient for determining a boundary,
except for the case of the default grapheme clusters.

Possible correction:
Note: Testing two adjacent characters is insufficient for determining a boundary.

Explanation:
Maybe the easiest counterexample is a sequence of many RI characters. There is no
fixed limit to the number of preceding characters needed for context.

Background information / discussion

The statement was true until emoji were encoded…

Seg7: UAX 14 dictionary usage wrongly typeset examples

Recommended UTC actions

1. Action item for Robin Leroy, PAG: In UAX #14 section 5.2 Dictionary Usage, change the examples to use the intended characters, rather than workarounds, and calling out the code points only where they are relevant to the point being made (how to encode the text to get appropriate LB classes); for Unicode 15.1. For details see L2/22-244 item Seg7.
2. Action item for Robin Leroy, Asmus Freytag, PAG: Consider moving much of UAX #14 section 5.2 Dictionary Usage into the core spec, for Unicode 16.0. For details see L2/22-244 item Seg7.

Feedback (verbatim)

Date/Time: Wed Sep 21 02:47:38 CDT 2022
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UAX #14: Unicode Line Breaking Algorithm

UAX #29: Unicode Text Segmentation
https://www.unicode.org/reports/tr29/#Table_Combining_Char_Sequences_and_Grapheme_Clusters
Version: Unicode 15.0.0
Date: 2022-08-26
Revision: 41

UAX #14: Unicode Line Breaking Algorithm
https://www.unicode.org/reports/tr14/#Dictionary
Version: Unicode 15.0.0
Date: 2022-08-16
Revision: 49

Location: 5.2 Dictionary Usage

Problematic text:
BBC English Dictionary: sIləbl where I is <U+026A, U+0332> and ə is U+0259.
The vowel of the stressed syllable is underlined.
Collins Cobuild English Language Dictionary: sIləbə°l where I is <U+026A, U+0332>
and has the same meaning as in the BBC English Dictionary. The ə is U+0259 (both times).
The ° is a U+2070 and indicates the schwa may be omitted.

Explanation:
The typeset examples do not correspond to the explanation text.
Specifically, the examples have the final letter "l" underlined (with an HTML
tag, not with U+0332, so cannot reproduce here). But this is not the stressed vowel.
This should not be underlined and instead the second letter "l" should be underlined.

The typeset examples in this section also deviate from the explanations in other ways
("I" is not U+026A as stated, "°" is not U+2070 as stated, etc.) but those are visually
similar and can be forgiven for lack of fonts or something in the document producing system.

Background information / discussion

Robin: The whole « where X is U+Y » dance should really be read « where X stands for U+Y ». It is a holdover from the Unicode 3.0.0 and 3.0.1 days when the UAX was in CP1252 (https://www.unicode.org/reports/tr14/tr14-6.html and https://www.unicode.org/reports/tr14/tr14-7.html are not in UTF-8, despite what the HTTP headers say).
An attempt was made to patch that up in Unicode Version 4.0.1, but the result is rather messy.

Asmus: This section documents a lot of*valuable* examples for various conventions used in dictionaries, but for most examples there seems to be no explanation what that means in terms of line breaking. With some exceptions.

I think this whole section is misplaced and needs to be put into some part in the core spec, as most of it is about which marks are used in dictionary style vs. IPA phonetic notation.

The few bits of line-breaking / word-splitting considerations (like changing hyphens to tilde) can be retained, and the whole section can be referenced from UAX#14 with a brief mention of the issue and "see also".

The tests to apply are

1. is the whole of the information as presented in scope for the scope of LBA?
2. is a reader interested in the topic of "Dictionary Usage" going to look at UAX#14?

I believe the current section fails both of these tests.

Therefore, it should be moved to a location where it is fully in scope and where readers interested in all aspects of the examples will benefit from them.

Collation

Coll1: U+10A7F OLD SOUTH ARABIAN NUMERIC INDICATOR should sort among punctuation

From PAG email discussion.

Recommended UTC actions

1. Action for Ken Whistler, Markus Scherer, PAG: For UCA 15.1 DUCET, move U+10A7F OLD SOUTH ARABIAN NUMERIC INDICATOR to sort among punctuation.

Summary

In the DUCET, U+10A7F (𐩿) [Po] OLD SOUTH ARABIAN NUMERIC INDICATOR sorts among non-digit numeric characters.
The CLDR root collation sorts it among punctuation, as its General_Category suggests.
Ken Whistler agrees that this character should be moved in the DUCET to sort among punctuation.

Regex

Regex1: UTS 18 wrong example with Greek and Basic_Emoji

Recommended UTC actions

1. Action item for Mark Davis, PAG: In UTS #18 section 1.3 Subtraction and Intersection change the last example to `[\P{Script=Greek}&&\P{Basic_Emoji}]`. See L2/22-244 item Regex1.
2. Action Item for Rick McGowan: Post a Public Review Issue for a proposed update of UTS #18.

Feedback (verbatim)

Date/Time: Thu Sep 15 03:28:12 CDT 2022
Name: Rossen Mikhov
Report Type: Error Report
Opt Subject: UTS #18: Unicode Regular Expressions

https://www.unicode.org/reports/tr18/#Subtraction_and_Intersection
Version 23
Date 2022-02-08

Location:
Section "1.3 Subtraction and Intersection", near the end of the section.

Wrong text:
Thus the following matches all code points that neither have a Script value of
Greek nor are in Basic_Emoji:

```
[^[\p{Script=Greek} && \p{Basic_Emoji}]]
```

Possible correction:
Thus the following matches all code points that do not simultaneously have a
Script value of Greek and are in Basic_Emoji:

Suggestion:
There are no Greek emoji, so the example actually matches all Unicode code
points. Perhaps a more illustrative example should be given.

Background information / discussion

Replace the example with one that works and also retains the intersection operator, so that the example fits the section title.

Security

Sec1: SCWG proposals

L2/22-233 “Process report from the source code workinggroup” from Robin Leroy, Source code ad hoc working group

- UTC should review but need not take immediate action.

L2/22-234 “Recommendations of the source code workinggroup for UTC #173” from Robin Leroy, Source code ad hoc working group

- Contains all of the recommended UTC actions.

L2/22-229 “Proposed changes to Unicode properties and reports for source code handling”, from Robin Leroy, Mark Davis, Source code ad hoc working group

- 69 pages
- Changes to Other_ID_Continue, emoji variation sequences, UAXes 9, 14, and 31, UTSes 39 and 51.
- New UTS: Proposed Draft Unicode Technical Standard #55, “Unicode Source Code Handling”

L2/22-230 “Mathematical notation profile for default identifiers”, L2/22-231 “Mixed-script detection in identifier chunks”, and L2/22-232 “Unicode identifier styles”:

- The UTC need not take action about these documents; they are rationales for some of the proposals in L2/22-229.

Recommended UTC actions

1. Discuss the documents in the UTC and adopt the recommendations in L2/22-234.

Background information / discussion

Some PAG members have reviewed much of the documents except the text proposed for a new UTS, provided feedback, and agree with the recommendations.
We agree to the proposed draft UTS #55 and are looking forward to a public review issue.