L2/02-267R3

Re:	Property file changes for UCD 3.2.1
From:	Mark Davis
Date:	2002-08-13

While doing some work for the IDN identifier tables, I came across a number of property issues; other ICU team members and users also ran across related property issues that are also included here. The list grew from on and on from there, as I collected other property-related issues that have come up, such as from the editorial committee.

The following are proposed for Unicode 3.2.1.

1. Identifier Revelations

A. **Oversights.** The following two are clear oversights in **PropList.txt**.

`U+034F COMBINING GRAPHEME JOINER` should be in Other_Default_Ignorable_Code_Point.
`U+205F MEDIUM MATHEMATICAL SPACE` should be in White_Space

B. **SHY.** Eric Muller has come up with a *much* cleaner way to view the SHY (and similar characters).

```
00AD;SOFT HYPHEN;Pd
1806;MONGOLIAN TODO SOFT HYPHEN;Pd
```

Essentially, it is that the SHY is a format character that indicates a preferred intra-word line-break position. If the line is broken at that point, *just as if it is broken at some other intra-word position*, then whatever mechanism appropriate for intra-word line-breaks should be invoked. Depending on the language and the word, that may produce different visible results, such as:

- simply inserting a hyphen
- inserting a hyphen and changing spelling in the divided word parts
- not showing any visible change and simply breaking at that point

In line with this, the SHY and similar characters should be changed to be Cf characters and be added to Other_Default_Ignorable_Code_Point.

C. **Other_Default_Ignorable_Code_Point.** We need to clarify the description of this property given in PropList.html. What it says is: "characters that should be ignored in processing (unless explicitly supported)". That is, these are characters that an application should ignore if it does *not* support the characters for the particular process in question (whether it be linebreak, identifiers, rendering, or whatever). We are not talking about characters that the program *does* support for the particular processing.

The idea is that if a program does not support, say, DEVANAGARI LETTER KA, it should still not ignore it in processing,

especially in rendering. Displaying nothing would give the user the impression that it does not occur in the text at all. So we recommend displaying a box or a special last-resort glyph. If we were parsing for identifiers, we would not ignore an unsupported character (like KA); we would break an identifier before it -- and not include it in the identifier.

However, with characters like ZWJ, if the program does not support it, the best approach is to ignore it completely; don't display a box, since the normal display of the character is invisible -- it's effects are on other characters (which we can't show anyway since we don't support the character). In the discussion with Deborah Goldsmith in the UTC meeting, it was also clarified that another way to characterize this property is the set of characters that are normally invisible.

Ken had a good observation: *Just defining a property by enumeration, and then having a two-liner patched into the documentation is evidently not enough. I think we are going to have to require some more explicit criteria put forward for new properties, so that a smart committee of people, conversant in character encoding, can reliably produce the same or similar results when asked to apply the property against some particular character, to make these kinds of determinations.*

D. **Letter Modifiers.** The General Category Sk (Symbol, Modifier) contains characters that are essentially the same as Lm (Letter Modifiers) except that their form is unlike a letter. However, few people take this into consideration in practice, since they were in the symbols section. For example, these were inappropriately excluded from our word and titlecase definition and from our identifier syntax. We have added them back into our word and titlecase definition, but did not make the corresponding change to our identifier syntax.

Unfortunately, Sk also contains some oddities like spacing MACRON, which should not be parts of words or identifiers. Natural languages don't use ^ as part of a word (separately as in "ro^le" -- they do use "rôle")? It is no more natural than "ro◇le", "ro•le", "ro•le" or "ro•le". Nor so we recommend that "ro^le" even sort anything like "rôle".

I recommend that we make the following fixes:

a. ***Move the following into Lm.***

```
02B9..02BA ; Sk # [2] MODIFIER LETTER PRIME..MODIFIER LETTER DOUBLE PRIME
02C2..02CF ; Sk # [14] MODIFIER LETTER LEFT ARROWHEAD..MODIFIER LETTER LOW ACUTE
ACCENT
02D2..02DF ; Sk # [14] MODIFIER LETTER CENTRED RIGHT HALF RING..MODIFIER LETTER
CROSS ACCENT
02E5..02ED ; Sk # [9] MODIFIER LETTER EXTRA-HIGH TONE BAR..MODIFIER LETTER
UNASPIRATED
```

These character are intermixed with letters as a part of words, and should be allowed in both of them; there is also no reason to exclude them from identifiers. Moving them into Lm would reflect that fact, and produce better default behavior for all processes dealing with words and identifiers.

Note: this leaves the following characters in Sk

```
005E ; Sk # CIRCUMFLEX ACCENT
0060 ; Sk # GRAVE ACCENT
00A8 ; Sk # DIAERESIS
00AF ; Sk # MACRON
00B4 ; Sk # ACUTE ACCENT
00B8 ; Sk # CEDILLA
FF3E ; Sk # FULLWIDTH CIRCUMFLEX ACCENT
FF40 ; Sk # FULLWIDTH GRAVE ACCENT
FFE3 ; Sk # FULLWIDTH MACRON
0374..0375 ; Sk # [2] GREEK NUMERAL SIGN..GREEK LOWER NUMERAL SIGN
0384..0385 ; Sk # [2] GREEK TONOS..GREEK DIALYTIKA TONOS
1FBD ; Sk # GREEK KORONIS
1FBF..1FC1 ; Sk # [3] GREEK PSILI..GREEK DIALYTIKA AND PERISPOMENI
1FCD..1FCF ; Sk # [3] GREEK PSILI AND VARIA..GREEK PSILI AND PERISPOMENI
1FDD..1FDF ; Sk # [3] GREEK DASIA AND VARIA..GREEK DASIA AND PERISPOMENI
```

```
1FED..1FEF ; Sk # [3] GREEK DIALYTIKA AND VARIA..GREEK VARIA
1FFD..1FFE ; Sk # [2] GREEK OXIA..GREEK DASIA
309B..309C ; Sk # [2] KATAKANA-HIRAGANA VOICED SOUND MARK..SEMI-VOICED SOUND MARK
```

The above are all special-case spacing symbols that are not used in the interior of words or identifiers in practice, any more that other symbols are. They should be left as Symbols to reflect this.

b. ***Adjust the word/titlecase definitions to remove Sk.***

The only reason for them to include Sk was for the characters in #2, but this also includes the unwanted characters from #1. Once the #2 characters are removed from Sk, then the word/titlecase definitions don't need them any more.

c. ***Other options (not preferred)***

i. Leave Sk the way it is. A useless collection of characters. Add a new property that distinguishes the word-capable characters. Add it to the definitions of titlecase, words, identifiers*.
ii. Move goofy (a technical term) characters like (spacing) MACRON into So. Leave the definition of titlecase, word alone (they use Sk). Add Sk to the definition of identifier*.

E. **Backwards-Compatible Identifiers.** As it turns out, the Unicode definitions of identifiers have been, in general, backwards-compatible. That is, if X was an identifier under some version of Unicode, it is also an identifier under the current version of Unicode (with very few exceptions). We provide guidelines for how people can make their own identifiers backwards-compatible if the Unicode definitions change, but it would make it *far* simpler for implementers if we simply guaranteed that our identifiers were backwards-compatible. The earliest point at which commercial languages started using the Unicode definition was 2.0, so we should start there.

I thus recommend:

a. ***Adding the following exceptional set to each of the identifier properties:***

```
U+2118 # SCRIPT CAPITAL P
U+212E # ESTIMATED SYMBOL
U+309B..U+309C # KATAKANA-HIRAGANA VOICED SOUND MARK..SEMI-VOICED SOUND MARK
```

b. ***Verifying with each release programmatically (with the same program that generated the above) that backwards-compatibility is maintained.***

The only down-side that I can see with this is that we are slightly out of sync with the ISO TR 10176; but we are already out of sync since they are based on old versions of the Unicode standard (currently the one in ballot is based on 3.0, and is already 44K characters out of date!). This is a simple application of "practice what you preach", and makes it far easier for users of our standard to themselves have backwards-compatible identifiers.

One might think that extending the notion of identifier could cause problems. But these characters are not an issue. The only possibility of a conflict would arise if you were parsing a file, and encountered something like:

...identifier<syntax character>...

which suddenly got treated as an identifier. However, none of the mentioned characters are treated as syntax characters in any known
programming language, so it would not be an issue.

---

2. UTF-8 in Property Files

I believe the time has come to use UTF-8 consistently in all of our property data files. Currently **Unihan.txt** and **NormalizationTest.txt** are in UTF-8, a couple files are in Latin- 1, and most files are in ASCII. However, importantly:

- UTF-8 and Latin-1 are only used in comments (outside of **Unihan.txt**)
- There is no BOM character in the UTF-8.

This means that parsers that strip comments don't even need to know that the file is UTF-8 (unless they parse **Unihan.txt**); they can just treat it as ASCII. If we continue to follow these two principles, it makes the switchover almost unnoticeable. Initially, this would only matter in the few files that contain some Latin-1 non ASCII. Later, we could add real, readable annotations in comments to some of the files, e.g.:

00DF; 00DF; 0053 0073; 0053 0053; # LATIN SMALL LETTER SHARP S
0130; 0069 0307; 0130; 0130; # LATIN CAPITAL LETTER I WITH DOT ABOVE

could become:

00DF; 00DF; 0053 0073; 0053 0053; # ß; ß; Ss; SS; LATIN SMALL LETTER SHARP S;
0130; 0069 0307; 0130; 0130; # •; i ◌̇; •; •; LATIN CAPITAL LETTER I WITH DOT ABOVE

---

3. Fallback Properties

As a general rule, we should not have the fallback value for a property (the one that we give code points that are not explicitly mentioned) require computation; it should be a single value. Otherwise, it is too error-prone; too easy for programmers to make mistakes when processing the data files.

A. **Bidi_Class (UnicodeData.txt)**

The way that the BIDI class property is handled is very error-prone. We say in UAX #9 that all *unassigned* code points are given the following values

- [0590-05FF, FB1D-FB4F] => R
- [0600-07BF, FB50-FDFF, FE70-FEFF] => AL
- all other unassigned => L

Unfortunately, this is not repeated in UnicodeData-3.2.0.html (where the properties of UnicodeData.txt are documented). Nor are the relevant R and AL code points listed explicitly in DerivedBidiClass-3.2.0.txt. We should address both of these points: document the ranges in the .html file, and add the code points to DerivedBidiClass.txt.

B. **Joining_Type (ArabicShaping.txt)**

The Joining Type T is also not explicitly listed in ArabicShaping-3.2.0.txt. While in this case, at least the formula for computing T is included in the comments in the file, it would be less error-prone if they were listed explicitly. Those values are already given in DerivedJoiningType-3.2.0.txt.

C. **EastAsianWidth.txt**

The data file says:

```
# - Assigned characters that are not listed explicitly are given the value "N".
```

It omits telling what the default is for *unassigned* code points. I assume they are also N, in which case this needs to be changed

to:

```
# - All code points that are not listed explicitly are given the value "N".
```

If they are *not* all N, then the ones that aren't should be explicitly listed!

D. **LineBreak.txt**

The data file says:

```
# - Assigned characters that are not listed explicitly are given the value # "AL".
# - Unassigned characters are given the value "XX".
```

The data file *actually* lists all the characters that are AL, and should. The above should be changed to:

```
# - All code points that are not listed explicitly are given the value "XX".
```

E. **Simple titlecase mapping (field 14 in UnicodeData.txt)**

UnicodeData.html says: "This field is omitted if the titlecase is the same as field 12."

A user noted that "this is apparently not true, except for 01C5, 01C8, 01CB and 01F2." The data should consistently either omit or include the field (when the same as field 12), and the documentation should match.

---

4. Line Break TR

A. The TR has (LB15b) HY ÷ *before* (LB18) HY × NU. Since early rules have precedence, the second rule has no effect, meaning that this doesn't allow -3. This needs to be fixed. The minimal change is to move LB 15b to be LB 18b.
B. I had thought (and reported) that Line Break would incorrectly break within Hangul syllables of the form LLVT. However, on closer examination, that doesn't happen: we are covered by Rule 6. However:
   - While the text of the rule is ok (after a slight amendment), the symbolic statement of the rule is out of date, and needs to be fixed.
   - The assignment of values ID to the leading Jamo and CM to the trailing jamo does *not* produce correct results, and does not account for LV and LVT syllables. These should be all changed to AL (with a note that in non-spacing Hangul environments they would be changed to ID). For more information, see the section on Line_Break_Pair_Tables below.
C. The TR text should reflect more clearly that WORD JOINER (and that semantic of ZWNBSP) prevents line breaks, *including* a break at a hyphenation point in the interior of a word.
D. It should also reflect our position on SHY (see above).
E. The following text is in the rules:

   - Numbers are of the form PR ? (OP | HY) ? NU (NU | IS) * CL ? PO ?
     Examples: $(12.35) 2,1234 (12)¢ 12.54¢
     This is approximated with the following rules.

   The text should be clearer that it is *reasonable* (but *not* required) to use the regular expression, instead of the approximate rules. (It gives better results than the pairwise approach, and for regular-expression-based linebreak engines, is much easier to implement.)
F. The type CB must be resolved before the LB algorithm is invoked, but there is *no* other obvious type that it should normally behave like! B2 doesn't work, since that is specifically quotation dash. Either the text must describe what it is to map to, or a new rule should be added:
   - *LB 14a Break before and after CB*
     CB ÷
     ÷ *CB*

G. The table of rules does not account for the types: SP, BK, CR, and LF. While some of the code implicitly handles SP, it does not account for edge cases, such as a space at the start of text.
H. The text says "(See the Unicode Standard [U3.0] for other rules regarding graphemes.)". This needs to be updated to point at TR29, and use the newer term *grapheme cluster*.
I. The following definitions are supplied below the pair table:
   - "^ denotes a *prohibited break*: Never break here, even if one or more spaces intervene."
   - "% denotes an *indirect break opportunity*: Don't break here, unless one or more spaces intervene."

   However, the term "here" is imprecise. The normal interpretation would be that A ^ B iff A × B *and* A SP* × B. In that case, "here" means just "before the B". However, there are certain cases, such as ZW CL, where ZW SP × CL but ZW *does* break from CL. If the table is right, then the definition needs to be changed to A ^ B iff A × B *and* A SP* × B *and* A × SP* × B. I suggest the following clarification.

   - "A ^ B denotes a *prohibited break* between A and B. Don't break after the A or before the B, even if one or more spaces were inserted between them."
   - "A % B denotes an *indirect break opportunity* between A and B. Don't break between the A and the B, but if one or more spaces were inserted, then do break either after the A or before the B."

J. The table of rules is incorrect for CM. For almost all types A, A ^ CM. This is because A × CM, and A × SP, and SP × CM. (ZW, oddly, is an exception). In addition, CM % PO (by rules 6/7 and 17) and and CM ÷ NU and CM ÷ AL (by rules 6/7 and 20). The reason for this is "Correspondingly, if there is no base, or if the base character is **SP, CM*** or **SP CM*** behave like **ID**."
K. The ordering of the following rules is incorrect:

   LB 12 Break after spaces
   SP ÷

   LB 13 Don't break before or after NBSP or WORD JOINER
   × GL
   GL ×

   The main purpose of WORD JOINER is exactly to *prevent* breaks where they would otherwise occur. The minimal fix is to change the ordering of LB 13, to move it to being LB 11b.
L. At the end of this document (Line_Break_Pair_Tables) are two pair tables. The first represents the current TR results (with the above fixes where the table incorrectly deviates from the rules). The second has the suggested rule movements listed above.
M. The SG section and value should be removed (see text below from the TR); Line Break should be ***not*** be phrased in terms of UTF-16 code units at point in time; and surrogate code points do not have the behavior below, nor are any of them "characters".

SG - Surrogates (XP) - (normative)

All characters with General Category Cs. There is no break between a high surrogate and a low surrogate....

---

5. SpecialCasing and TR21

A. Unfortunately, document L2/01-445, Item 3 was overlooked when doing the fixes for Special Casing in 3.2.0. The data lines:

```
# When lowercasing, remove dot_above in the sequence I + dot_above, which will turn
into i.
# This matches the behavior of the canonically equivalent I-dot_above

0307; ; 0307; 0307; tr After_Soft_Dotted; # COMBINING DOT ABOVE
0307; ; 0307; 0307; az After_Soft_Dotted; # COMBINING DOT ABOVE
```

do not match the comment (which is correct). They need to be changed to:

```
# AFTER_I: The last preceding base character was an uppercase I, and
# no combining character class 230 (above) has intervened.
...
# When lowercasing, remove dot_above in the sequence I + dot_above, which will turn
into i.
# This matches the behavior of the canonically equivalent I-dot_above

0307; ; 0307; 0307; tr AFTER_I # COMBINING DOT ABOVE
0307; ; 0307; 0307; az AFTER_i # COMBINING DOT ABOVE
```

Note: This will not have an effect on CaseFolding.

B. In TR21, Definition D1 is as follows:

D1. A character C is defined to be *cased* if it meets any of the following criteria:

- The general category of C is
  - Titlecase Letter (Lt)
- In [CoreProps], C has one of the properties
  - Uppercase, or
  - Lowercase
- Given D = NFD(C), then it is not the case that:
  - D = UCD_lower(D) = UCD_upper(D) = UCD_title(D)

Condition #3 is now redundant, since Uppercase and Lowercase have been 'closed' for #2. It thus does not add any additional characters. Thus #3 should be omitted (although we need to maintain consistency tests to ensure that it is captured in #2).

C. In TR21, there are two places the text needs to be changed to account for edge-cases with subscript-iota.

a. In Section 1.4, change:

For any string X, let Q(X) = NFC(toCasefold(X)). In other words, Q is the result of casefolding X, then putting the result into NFC format...
That is, given R(X) = NFC(toCasefold(X)), there are some strings such that R(R(X)) != R(X).

to:

For any string X, let Q(X) = NFC(toCasefold(NFD(X))). In other words, Q is the result of normalizing X, then casefolding the result, then putting the result into NFC format...
That is, given R(X) = NFKC(toCasefold(NFD(X))), there are some strings such that R(R(X)) != R(X).

b. In section 2.5, change:

- A string X is a canonical caseless match for a string Y if and only if
  NFD(toCasefold(X)) = NFD(toCasefold(Y))
- A string X is a compatibility caseless match for a string Y if and only if
  NFKD(toCasefold(NFKD(toCasefold(X)))) = NFKD(toCasefold(NFKD(toCasefold(Y))))

to:

- A string X is a canonical caseless match for a string Y if and only if
  NFD(toCasefold(NFD(X))) = NFD(toCasefold(NFD(Y)))
- A string X is a compatibility caseless match for a string Y if and only if
  NFKD(toCasefold(NFKD(toCasefold(NFD(X))))) = NFKD(toCasefold(NFKD(toCasefold(NFD(Y))))))

Note: The invocations of normalization before folding in the above definitions are to catch very infrequent edge cases. The only cases where normalization is required before folding is for the character

U+0345 (◌ͅ) COMBINING GREEK YPOGEGRAMMENI and any characters that have it as part of their decomposition (such as U+1FC3 (ῃ) GREEK SMALL LETTER ETA WITH YPOGEGRAMMENI).

In practice, implementations can produce optimized versions that just catch these special cases, and avoid an extra normalization.

---

6. New Properties in PropList.txt

A. There are 3 headings in Section 4.2 of TUS (page 79) which are not reflected in the UCD, and thus cannot effectively be used programmatically. ***If*** those properties are indeed important, they should be reflected in UCD properties. Each of these properties is a subset of the set of characters with canonical combining class = 0 and general_category = mark. They would be:

Split:

```
U+09CB..U+09CC # BENGALI VOWEL SIGN O..BENGALI VOWEL SIGN AU
U+0B48 # ORIYA VOWEL SIGN AI
U+0B4B..U+0B4C # ORIYA VOWEL SIGN O..ORIYA VOWEL SIGN AU
U+0BCA..U+0BCC # TAMIL VOWEL SIGN O..TAMIL VOWEL SIGN AU
U+0CC0 # KANNADA VOWEL SIGN II
U+0CC7..U+0CC8 # KANNADA VOWEL SIGN EE..KANNADA VOWEL SIGN AI
U+0CCA..U+0CCB # KANNADA VOWEL SIGN O..KANNADA VOWEL SIGN OO
U+0D4A..U+0D4C # MALAYALAM VOWEL SIGN O..MALAYALAM VOWEL SIGN AU
U+0DDA # SINHALA VOWEL SIGN DIGA KOMBUVA
U+0DDC..U+0DDE # SINHALA VOWEL SIGN KOMBUVA HAA AELA-PILLA..SINHALA VOWEL SIGN
KOMBUVA HAA GAYANUKITTA
U+17BF..U+17C0 # KHMER VOWEL SIGN YA..KHMER VOWEL SIGN IE
U+17C4..U+17C5 # KHMER VOWEL SIGN OO..KHMER VOWEL SIGN AU
```

Reordrant:

```
U+093F # DEVANAGARI VOWEL SIGN I
U+09BF # BENGALI VOWEL SIGN I
U+09C7..U+09C8 # BENGALI VOWEL SIGN E..BENGALI VOWEL SIGN AI
U+0A3F # GURMUKHI VOWEL SIGN I
U+0ABF # GUJARATI VOWEL SIGN I
U+0B47 # ORIYA VOWEL SIGN E
U+0BC6..U+0BC8 # TAMIL VOWEL SIGN E..TAMIL VOWEL SIGN AI
U+0D46..U+0D48 # MALAYALAM VOWEL SIGN E..MALAYALAM VOWEL SIGN AI
U+0DD9..U+0DDB # SINHALA VOWEL SIGN KOMBUVA..SINHALA VOWEL SIGN KOMBU DEKA
U+1031 # MYANMAR VOWEL SIGN E
U+17BE # KHMER VOWEL SIGN OE
U+17C1..U+17C3 # KHMER VOWEL SIGN E..KHMER VOWEL SIGN AI
```

Subjoined:

```
U+0F90..U+0F97 # TIBETAN SUBJOINED LETTER KA..TIBETAN SUBJOINED LETTER JA
U+0F99..U+0FBC # TIBETAN SUBJOINED LETTER NYA..TIBETAN SUBJOINED LETTER FIXED-FORM
RA
```

B. **NF*_Stable**. There are 4 derived properties that may be useful to add to the UCD, one for each normalization form. They are the set of code points that are always *stable:* never affected by the normalization process in the current version of Unicode. This property is rather useful for skipping over text that does not need to be considered at all when normalizing.

Formally, each stable code point CP fulfills all the following conditions:

a. CP has canonical combining class 0, and
b. CP is (as a single character) not changed by this normalization form, and
   if NKC or NFKC, ALL of the following:
c. CP can never compose with a previous character, and
d. CP can never compose with a following character, and
e. CP can never change if another character is added.

Example: In NFC, a-breve might satisfy all but (e), but if you add an ogonek it changes to a-ogonek + breve. So it is not stable. However, a-ogonek is stable in NFC, since it does satisfy (a-e).

There are pluses and minuses to adding these properties:

- The upside is that it is a bit tricky/time-consuming to compute, so it saves implementers time and avoids mistakes.
- The downside is that the list of characters is rather large, so it bloats the size of the file.

C. In a number of cases, we have string transforms, functions that map a string onto a (perhaps) modified string. Thus we speak of NFC(x) as the normalized form of x (according to the definition of NFC). These transforms can also be used to derive useful binary properties: such as isNFC(x), where isNFC(x) is true iff NFC(x) == x. This would be useful to document somewhere.

7. Other Properties

A. I tested Terminal_Punctuation (from **PropList.txt**) with a compatibility closure. The following items are in that set, but not in the original:

```
U+2024 ONE DOT LEADER
U+2025 TWO DOT LEADER
U+2026 HORIZONTAL ELLIPSIS
U+2047 DOUBLE QUESTION MARK
U+FE30 PRESENTATION FORM FOR VERTICAL TWO DOT LEADER
```

It appears that at least the Double Question Mark should be in Terminal_Punctuation (and in the appropriate properties below, if they are accepted). As to the others, who can say...

B. For UTR #29: Text Boundaries, the following should be added to Other_Extend. While we decided to remove combining marks in general, unless these particular marks are included, the boundary specifications are not closed under normalization.

```
U+09BE # BENGALI VOWEL SIGN AA
U+09D7 # BENGALI AU LENGTH MARK
U+0B3E # ORIYA VOWEL SIGN AA
U+0B57 # ORIYA AU LENGTH MARK
U+0BBE # TAMIL VOWEL SIGN AA
U+0BD7 # TAMIL AU LENGTH MARK
U+0CC2 # KANNADA VOWEL SIGN UU
U+0CD5..U+0CD6 # KANNADA LENGTH MARK..KANNADA AI LENGTH MARK
U+0D3E # MALAYALAM VOWEL SIGN AA
U+0D57 # MALAYALAM AU LENGTH MARK
U+0DCF # SINHALA VOWEL SIGN AELA-PILLA
U+0DDF # SINHALA VOWEL SIGN GAYANUKITTA
U+1D165 # MUSICAL SYMBOL COMBINING STEM
U+1D16E..U+1D16F # MUSICAL SYMBOL COMBINING FLAG-1..MUSICAL SYMBOL COMBINING FLAG-2
```

C. For UTR #29: Text Boundaries, it would be useful to add some new properties once it is finalized (so probably in 4.0 instead of 3.2). That way people can use machine-readable properties instead of digging them out of TR text. The possibilities should be

reviewed with the UTC review of the TR. Candidates include:

a. **MidLetter**: Non-letters that normally can occur in the middle of words. This is an informative property, and may be customized for different languages.
   - **Note:** to this class should probably be added:
     U+00B7 (·) MIDDLE DOT
     U+2027 (•) HYPHENATION POINT

b. **MidNumber**: Non-digits that normally can occur in the middle of numbers. This is an informative property, and may be customized for different languages.
   - The TR has three classes: MidLet*, MidNum, and MidNumLet. These are simply to form an enumeration from the overlap of the above sets. * It is called MidLetter, but that should be changed if we adopt the above name.

C. **Sep:** Probably with a different name, it would be useful to have a property for all the characters that break lines.
   - *Open Issue:* should LS break sentences?

d. **Ambiguous_Sentence_Punctuation**: Terminal_Punctuation characters that normally have two usages: they can end a sentence, but can also be used within sentences, such as for abbreviations. This is an informative property, and may be customized for different languages.

e. **Sentence_Punctuation**: Terminal_Punctuation characters that normally end a sentence, and are not normally within a sentence (at the end of words*). This is an informative property, and may be customized for different languages. For example, semicolon should be added when used with Greek.

f. **L, V, T, LV, LVT:** It would be useful to have distinct properties for the first three, and derived properties for the latter two.

g. There are two types of characters that need to be added when considering Katakana, characters that are not marked as being Katakana in Script.txt. We should consider adding the last to the script, and perhaps having a "shared" Katakana-Hiragana script.
   U+30FC # KATAKANA-HIRAGANA PROLONGED SOUND MARK
   U+FF70 # HALFWIDTH KATAKANA-HIRAGANA PROLONGED SOUND MARK
   U+FF9E..U+FF9F # HALFWIDTH KATAKANA SOUND MARKS

D. The properties Grapheme_Link, Grapheme_Base, and Grapheme_Extend should be adjusted to be consistent with TR29.

---

8. PropertyAliases and PropertyValueAliases

A. Markus noticed that Non_Break was in the **PropertyValueAliases.txt** file, but not in **PropList.txt**. It turns out that that name was a holdover from a development version of the property file, and should be deleted in 3.2.1

B. The block property is in from **PropertyAliases.txt**, but the values are missing from **PropertyValueAliases.txt**. We got some complaints about that. Also, the block property is listed as a non-enumerated property, even tho it is enumerated. This should be fixed.

C. **PropertyAliases.txt** do not contain properties from **Unihan.txt**. Pursuant to
   *"**[91-A11] Action Item** for Mark Davis: Add Unihan properties to the UCD."*
   I examined the Unihan tags, and came up with the following categorization. Some of them, while clearly required for IRG work -- should not be considered general-use properties. Others should be, and I propose that they be considered UCD properties, listed in PropertyAliases.txt, and given extracted files. (The latter are very useful when one doesn't want to schlepp around the entire Unihan file.)
   - The numeric values can go into the existing **DerivedNumericType.txt** and **DerivedNumericValues.txt** files, to unify all the Numeric properties.

Recommended to be included as Properties

Numeric:	Completes the other set of numeric properties in the UCD. *Proposed numeric type names:* Han_Primary (hp), Han_Accounting (ha), Han_Other (ho)
Variants	For foldings and comparison. *Proposed property names:* Semantic_Variant (semv), Simplified_Variant (simv), Specialized_Semantic_Variant (specv), Traditional_Variant (tradv), Z_Variant (zv)
kRSUnicode:	For indexing and sorting. *Proposed property name*: Unicode_Radical_Stroke (urs)

Recommended to be excluded as Properties (e.g. left simply as tags in the Unihan file)

Other Radical/Stroke:	Questionable validity; incomplete data
Character Mapping:	Logically a part of character mapping tables, *not* Unicode Properties
Dictionary Position, Definition, Grade:	Applicable only to very specific programs
Frequency, Pronunciations	Questionable validity; incomplete data
Redundant:	derivable from the UCD

Complete list of categorized tags from Unihan

Category	Property Name	Description from Unihan (abbreviated)
Numeric	kAccountingNumeric	The value of the character when used in the writing of accounting numerals.
	kOtherNumeric	The numeric value for the character in certain unusual, specialized contexts.
	kPrimaryNumeric	The value of the character when used in the writing of numbers in the standard fashion.
Variants	kSemanticVariant	The Unicode value for a semantic variant for this character. A semantic variant is an x- or y-variant with similar or identical meaning which can generally be used in place of the indicated character.
	kSimplifiedVariant	The Unicode value for the simplified Chinese variant for this character (if any).
	kSpecializedSemanticVariant	The Unicode value for a specialized semantic variant for this character. A specialized semantic variant is an x- or y-variant with similar or identical meaning only in certain contexts (such as accountants' numerals).
	kTraditionalVariant	The Unicode value(s) for the traditional Chinese variant(s) for this character.
	kZVariant	The Unicode value(s) for known z-variants of this character
Radical/Stroke	kRSJapanese	A Japanese radical/stroke count for this character in the form "radical.additional strokes".
	kRSKanWa	A Morohashi radical/stroke count for this character in the form "radical.additional strokes".
	kRSKangXi	A KangXi radical/stroke count for this character in the form "radical.additional strokes".
	kRSKorean	A Korean radical/stroke count for this character in the form "radical.additional strokes". A ' after the radical indicates the simplified version of the given radical
	kRSUnicode	A standard radical/stroke count for this character in the form "radical.additional strokes". A ' after the radical indicates the simplified version of the given radical
	kTotalStrokes	The total number of strokes in the character (including the radical)
Pronunciations	kCantonese	The Cantonese pronunciation(s) for this character
	kJapaneseKun	The Japanese pronunciation(s) of this character
	kJapaneseOn	The Sino-Japanese pronunciation(s) of this character
	kKorean	The Korean pronunciation(s) of this character
	kMandarin	The Mandarin pronunciation(s) for this character in pinyin
	kTang*	The Tang dynasty pronunciation(s) of this character, derived from _T'ang Poetic Vocabulary_
	kVietnamese	The character's pronunciation(s) in Qu•c ng•
Definition	kDefinition	An English definition for this character
Frequency	kFrequency	A rough fequency measurement for the character based on analysis of Chinese USENET postings
Grade	kGradeLevel*	The grade in the Hong Kong school system by which a student is expected to know the character.
Dictionary Position	kAlternateKangXi	An alternate possible position for the character in the KangXi dictionary
	kAlternateMorohashi	An alternate possible position for the character in the Morohashi dictionary
	kCihaiT*	The position of this character in the Cihai (辭海) dictionary, single volume edition, published in Hong Kong by the Zhonghua Bookstore, 1983 (reprint of the 1947 edition), ISBN 962-231-005-2.
	kCowles*	The index of this character in Roy T. Cowles, _A Pocket Dictionary of Cantonese_, Hong Kong: University Press, 1999.

	kDaeJaweon	The position of this character in the Dae Jaweon (Korean) dictionary used in the four-dictionary sorting algorithm.
	kFenn*	Data on the character from _Fenn's Chinese-English Pocket Dictionary_
	kHanYu	The position of this character in the Hanyu Da Zidian (HDZ) Chinese character dictionary (bibliographic information below).
	kHKGlyph*	The index of the character in 常用字字形表 (二零零零年修訂本), 香港: 香港教育學院, 2000, ISBN 962-949-040-4. This publication gives the "proper" shapes for characters as used in the Hong Kong school system.
	kIRGDaeJaweon	The position of this character in the Dae Jaweon (Korean) dictionary used in the four-dictionary sorting algorithm.
	kIRGDaiKanwaZiten	The index of this character in the Dae Kanwa Ziten, aka Morohashi dictionary (Japanese) used in the four-dictionary sorting algorithm.
	kIRGHanyuDaZidian	The position of this character in the Hanyu Da Zidian (PRC) dictionary used in the four-dictionary sorting algorithm.
	kIRGKangXi	The position of this character in the KangXi dictionary used in the four-dictionary sorting algorithm.
	kKangXi	The position of this character in the KangXi dictionary used in the four-dictionary sorting algorithm.
	kKarlgren*	The index of this character in _Analytic Dictionary of Chinese and Sino-Japanese_
	kLau*	The index of this character in _A Practical Cantonese-English Dictionary_
	kMatthews	The index of this character in _Mathews' Chinese-English Dictionary_
	kMeyerWempe*	The index of this character in the Student's Cantonese-English Dictionary
	kMorohashi	The index of this character in the Dae Kanwa Ziten, aka Morohashi dictionary (Japanese) used in the four-dictionary sorting algorithm.
	kNelson	The index of this character in _The Modern Reader's Japanese-English Character Dictionary_
	kPhonetic*	The phonetic index for the character from _Ten Thousand Characters: An Analytic Dictionary_
	kSBGY	The position of this character in the Song Ben Guang Yun (SBGY) Medieval Chinese character dictionary (bibliographic and general information below).
	kCangjie*	The cangjie input code for the character. This incorporates data from the file cangjie-table.b5 by Christian Wittern
Character Mapping	kBigFive	The Big Five mapping for this character in hex; note that this does *not* cover any of the Big Five extensions in common use, including the ETEN extensions.
	kCCCII	The CCCII mapping for this character in hex
	kCNS1986	The CNS 11643-1986 mapping for this character in hex
	kCNS1992	The CNS 11643-1992 mapping for this character in hex
	kEACC	The EACC mapping for this character in hex
	kGB0	The GB 2312-80 mapping for this character in ku/ten form
	kGB1	The GB 12345-90 mapping for this character in ku/ten form
	kGB3	The GB 7589-87 mapping for this character in ku/ten form
	kGB5	The GB 7590-87 mapping for this character in ku/ten form
	kGB7	The "General Use Characters for Modern Chinese" mapping for this character
	kGB8	The GB 8565-89 mapping for this character in ku/ten form
	kHKSCS	Mappings to the Big Five extended code points used for the Hong Kong Supplementary Character Set
	kIBMJapan	The IBM Japanese mapping for this character in hex
	kIRG_GSource	The IRG "G" source mapping for this character in hex. The IRG "G" source consists of data from the following national standards, publications, and lists from the People's Republic of China and Singapore.
	kIRG_HSource	The IRG "H" source mapping for this character in hex. The IRG "H" source consists of data from the Hong Kong Supplementary Characer Set.

	kIRG_JSource	The IRG "J" source mapping for this character in hex. The IRG "J" source consists of data from the following national standards and lists from Japan.
	kIRG_KSource	The IRG "K" source mapping for this character in hex. The IRG "K" source consists of data from the following national standards and lists from the Republic of Korea (South Korea).
	kIRG_KPSource	The IRG "KP" source mapping for this character in hex. The IRG "KP" source consists of data from the following national standards and lists from the Democratic People's Republic of Korea (North Korea).
	kIRG_TSource	The IRG "T" source mapping for this character in hex. The IRG "T" source consists of data from the following national standards and lists from the Republic of China (Taiwan).
	kIRG_VSource	The IRG "V" source mapping for this character in hex. The IRG "V" source consists of data from the following national standards and lists from Vietnam.
	kJIS0213	The JIS X 0213-2000 mapping for this character in min,ku,ten form
	kJis0	The JIS X 0208-1990 mapping for this character in ku/ten form
	kJis1	The JIS X 0212-1990 mapping for this character in ku/ten form
	kKPS0	The KP 9566-97 mapping for this character in hexadecimal form.
	kKPS1	The KPS 10721-2000 mapping for this character in hexadecimal form.
	kKSC0	The KS X 1001:1992 (KS C 5601-1989) mapping for this character in ku/ten form
	kKSC1	The KS X 1002:1991 (KS C 5657-1991) mapping for this character in ku/ten form
	kMainlandTelegraph	The PRC telegraph code for this character, derived from "Kanzi denpou koudo henkan-hyou"
	kPseudoGB1	A "GB 12345-90" code point assigned this character for the purposes of including it within Unihan.
	kTaiwanTelegraph	The Taiwanese telegraph code for this character, derived from "Kanzi denpou koudo henkan-hyou"
	kXerox	The Xerox code for this character
Redundant	kCompatibilityVariant*	The compatibility decomposition for this ideograph, derived from the UnicodeData.txt file.

Background Information

The following lists each Unihan tag, the total number of characters with that tag found in Unihan.txt, the minimum and lengths of the values associated with the tag, and a few sample values (separated by semicolons). Don't worry if some of the less common CJK characters appear as boxes on your machine; they are only examples.

1. **kAccountingNumeric**
   - count: 23, min length: 1, max length: 5
   - 1; 10; 100; 1000; 10000; 2; 3; 4; 5; 6; 7; 8; 9
2. **kAlternateKangXi**
   - count: 16828, min length: 8, max length: 8
   - 0075.001; 0075.003; 0075.005; 0075.007; 0076.001; 0076.002; 0076.003; 0076.004; 0076.005; 0076.007
3. **kAlternateMorohashi**
   - count: 17919, min length: 5, max length: 6
   - 00001; 00002; 00003; 00006; 00007; 00008; 00010; 00011; 00012; 00013; 00014; 00015; 00019; 00020; 00021
4. **kBigFive**
   - count: 13063, min length: 4, max length: 4
   - A440; A442; A443; A454; A455; A456; A457; A4A1; A4A2; A4A3; A540; A541; A542; C945; C946; C94D; C94F
5. **kCCCII**
   - count: 19698, min length: 6, max length: 6
   - 213021; 213022; 213023; 213024; 213025; 213026; 213027; 21302A; 216421; 216422; 236123; 2D3025; 2D3026
6. **kCNS1986**
   - count: 17258, min length: 6, max length: 6
   - 1-4421; 1-4423; 1-4424; 1-4435; 1-4436; 1-4437; 1-4438; 1-4462; 2-2126; 2-2127; 2-212F; E-2125; E-2126
7. **kCNS1992**
   - count: 17258, min length: 6, max length: 6
   - 1-4421; 1-4423; 1-4424; 1-4435; 1-4436; 1-4437; 1-4438; 1-4462; 2-2126; 2-2127; 2-212F; 3-2125; 3-2126

8. **kCangjie**
    - count: 13056, min length: 1, max length: 5
    - BM; CL; HGI; JK; JU; M; MF; MFM; ML; MLVS; MMM; MN; MOB; MS; MY; MYVS; NEM; NG; OM; PT; TTC; YM; YSM
9. **kCantonese**
    - count: 17379, min length: 2, max length: 33
    - CHAAU4; GUN3 HUNG1 JUNG1; JAAN2; JAAU1 KAAU5; JEUI6; JING4; JIP6; JYU4; KAAU4; LAAU4; NG5; NO6; TIM2
10. **kCihaiT**
    - count: 92, min length: 5, max length: 8
    - 1.101; 10.603; 10.604; 10.605; 11.201; 1398.307; 24.201; 37.103; 6.301; 7.401; 7.402; 952.602; 982.402
11. **kCompatibilityVariant**
    - count: 891, min length: 1, max length: 2
    - 串; 亂; 卵; 句; 喇; 奈; 契; 嵐; 懶; 更; 樂; 欄; 洛; 滑; 烙; 爛; 珞; 癩; 羅; 落; 蘭; 蘿; 螺; 裸; 豈; 賈; 車; 邏; 酪; 金; 駱; 鸞; 龜
12. **kCowles**
    - count: 241, min length: 1, max length: 14
    - 201; 2531; 2577; 3185; 3236; 3305; 3572; 3818; 3895 3896; 4168; 4718; 472; 492; 500; 5133; 906 908
13. **kDaeJaweon**
    - count: 16026, min length: 8, max length: 8
    - 0129.010; 0135.010; 0137.020; 0137.030; 0137.060; 0137.070; 0137.080; 0138.010; 0145.010; 0147.010
14. **kDefinition**
    - count: 19400, min length: 1, max length: 419
    - (an ancient form of 五) five; (corrupted form) to follow, to trust to; to put confidence in; to depend on, to turn around; to turn the body, (interchangeable 隱); (same as 丘) hillock or mound; to lick; to taste, a mat, bamboo bark
15. **kEACC**
    - count: 13244, min length: 6, max length: 6
    - 213021; 213022; 213023; 213024; 213025; 213026; 213027; 213029; 21302A; 216424; 274F22; 275432; 2D332A
16. **kFenn**
    - count: 14, min length: 4, max length: 4
    - 120K; 150C; 299E; 415B; 771B; 777K; 793K; 795C; 799K; 817C; 848I
17. **kFrequency**
    - count: 5089, min length: 1, max length: 1
    - 1; 2; 3; 4; 5
18. **kGB0**
    - count: 6763, min length: 4, max length: 4
    - 1827; 1983; 2201; 3863; 3950; 4093; 4147; 4232; 4582; 4734; 5027; 5175; 5341; 5508; 5602; 5604; 5607
19. **kGB1**
    - count: 6866, min length: 4, max length: 4
    - 1791; 1827; 2201; 3863; 3950; 3980; 4093; 4147; 4232; 4734; 5027; 5341; 5602; 5604; 5607; 8809; 8881
20. **kGB3**
    - count: 4836, min length: 4, max length: 4
    - 1601; 1608; 1610; 1617; 1657; 1660; 1661; 1666; 1667; 1668; 1670; 1671; 1672; 1681; 1855; 1858; 3083
21. **kGB5**
    - count: 2842, min length: 4, max length: 4
    - 1601; 1603; 1628; 1631; 1632; 1633; 1634; 1728; 1738; 1739; 1741; 1742; 1746; 1753; 1755; 1756; 1757
22. **kGB7**
    - count: 42, min length: 4, max length: 4
    - 0101; 0102; 0103; 0104; 0105; 0106; 0107; 0114; 0115; 0116; 0117; 0118; 0119; 0120; 0121; 0142; 0143
23. **kGB8**
    - count: 785, min length: 4, max length: 4
    - 1202; 1203; 1286; 1403; 1404; 1405; 1406; 1591; 9001; 9003; 9012; 9013; 9014; 9015; 9016; 9017; 9024
24. **kGradeLevel**
    - count: 48, min length: 1, max length: 1
    - 1; 2; 3; 4; 5
25. **kHKGlyph**
    - count: 183, min length: 4, max length: 4
    - 0001; 0002; 0003; 0004; 0005; 0006; 0007; 0008; 0009; 0010; 0011; 0012; 0013; 0014; 0015; 0016; 0017
26. **kHKSCS**

- count: 3727, min length: 4, max length: 4
- 89D5; 89DA; 89DB; 89DC; 8ADA; 8BDC; 8F59; 8F5D; 9277; 93CD; 96DF; 96F7; 97DB; 9BDF; 9DAA; 9E53; FA68

27. **kHanYu**
    - count: 55817, min length: 9, max length: 29
    - 10009.060; 10015.030; 10019.020; 10031.040; 10036.020; 10038.080; 10053.130; 10056.020; 10263.070; 42813.010
28. **kIBMJapan**
    - count: 360, min length: 4, max length: 4
    - FA61; FA68; FA69; FA6A; FA6B; FA6C; FA6D; FA6E; FA6F; FA70; FA71; FA72; FA73; FA74; FA75; FA76; FA77
29. **kIRGDaeJaweon**
    - count: 16024, min length: 8, max length: 8
    - 0129.010; 0135.010; 0137.020; 0137.030; 0137.060; 0137.070; 0137.080; 0138.010; 0145.010; 0147.010
30. **kIRGDaiKanwaZiten**
    - count: 17864, min length: 5, max length: 6
    - 00001; 00002; 00003; 00006; 00007; 00008; 00010; 00011; 00012; 00013; 00014; 00015; 00019; 00020; 00021
31. **kIRGHanyuDaZidian**
    - count: 55812, min length: 9, max length: 9
    - 10009.060; 10015.030; 10019.020; 10031.040; 10036.020; 10038.080; 10053.130; 10056.020; 10263.070; 42813.010
32. **kIRGKangXi**
    - count: 70205, min length: 8, max length: 8
    - 0078.010; 0078.030; 0078.101; 0079.020; 0079.021; 0081.180; 0083.011; 0084.051
33. **kIRG_GSource**
    - count: 57623, min length: 2, max length: 6
    - 3-3024; 3-302B; 3-3032; 5-3024; 5-3044; 5-3076; 5-334D; HZ; KX
34. **kIRG_HSource**
    - count: 1106, min length: 4, max length: 4
    - 8A4D; 8B6A; 8EF5; 8F5D; 90B7; 9253; 92A1; 92C9; 9455; 957A; 97DC; 9844; 9C5D; 9E64; A0E8; FB43; FC65
35. **kIRG_JSource**
    - count: 13119, min length: 5, max length: 6
    - A-2121; A-2122; A-2123; A-2124; A-2125; A-2126; A-2127; A-2128; A-2129; A-212A; A-212B; A-212C; A-212D
36. **kIRG_KPSource**
    - count: 23957, min length: 8, max length: 8
    - KP1-3451; KP1-345F; KP1-346A; KP1-348C; KP1-34B5; KP1-34CD; KP1-34F3; KP1-3502; KP1-350C; KP1-3555
37. **kIRG_KSource**
    - count: 17392, min length: 6, max length: 6
    - 3-2121; 3-2122; 3-2123; 3-2124; 3-2125; 3-2126; 3-2127; 3-2128; 3-2129; 3-212A; 3-212B; 3-212C; 3-212D
38. **kIRG_TSource**
    - count: 54988, min length: 6, max length: 6
    - 3-2323; 3-2741; 3-286C; 3-343B; 3-396D; 4-2157; 4-2224; 4-2336; 4-2835; 6-2123; 6-2130; 6-222C; F-216C
39. **kIRG_VSource**
    - count: 9841, min length: 6, max length: 6
    - 0-3034; 0-3047; 0-3048; 0-304E; 0-304F; 2-6E49; 2-6E4B; 2-6E65; 2-6E7B; 2-6F57; 2-8874; 2-8875; 2-8876
40. **kJIS0213**
    - count: 3625, min length: 7, max length: 7
    - 1,14,03; 1,14,51; 1,14,59; 2,01,13; 2,01,18; 2,01,19; 2,01,54; 2,01,62; 2,01,94; 2,03,11; 2,03,15; 2,84,72
41. **kJapaneseKun**
    - count: 11259, min length: 1, max length: 96
    - HINOTO ATARU YOBORO; HITOTSU HITOTABI HAJIME; NANATSU NANATABI; SAMATAGERARERU; SHIMO; UE KAMI; YOROZU
42. **kJapaneseOn**
    - count: 13139, min length: 1, max length: 35
    - ICHI ITSU; JOU CHOU; JOU SHOU; KA; KA GE; KI GI; KOU; MAN BAN; SAN; SHICHI SHITSU; SHOU; TEI CHOU TOU
43. **kJis0**
    - count: 6356, min length: 4, max length: 4
    - 1676; 1715; 1828; 1978; 2154; 2716; 2823; 3069; 3070; 3204; 3590; 4152; 4392; 4531; 4802; 4803; 5034
44. **kJis1**
    - count: 5801, min length: 4, max length: 4
    - 1601; 1602; 1603; 1604; 1605; 1606; 1607; 1608; 1609; 1610; 1611; 1612; 1613; 1614; 1615; 1616; 1617
45. **kKPS0**
    - count: 4653, min length: 4, max length: 4

- D0DF; DAB9; DDF9; DFBE; DFC9; E1B5; E1C2; E4EF; E5F9; E6DD; E8B9; EAB2; ECA8; EEC9; EFA6; F2BA; FCD6

46. **kKPS1**
    - count: 19301, min length: 4, max length: 4
    - 3451; 345F; 346A; 348C; 34B5; 34CD; 34F3; 3502; 350C; 3526; 3555; 355D; 356C; 357D; 3580; 3582; 359D
47. **kKSC0**
    - count: 4888, min length: 4, max length: 4
    - 4688; 5618; 6016; 6084; 6164; 6318; 6330; 6506; 6710; 7673; 7759; 7943; 8173; 8306; 8568; 8650; 8927
48. **kKSC1**
    - count: 2856, min length: 4, max length: 4
    - 5608; 5762; 5941; 5966; 6174; 6485; 6517; 6582; 6612; 6618; 7002; 7314; 7315; 7460; 7742; 7779; 8511
49. **kKangXi**
    - count: 21158, min length: 8, max length: 8
    - 0075.010; 0075.030; 0075.050; 0075.070; 0076.010; 0076.020; 0076.021; 0076.030; 0076.040; 0076.050
50. **kKarlgren**
    - count: 2560, min length: 1, max length: 5
    - 103; 115; 126; 149; 150; 151; 180; 189; 201; 232; 233; 280; 285; 33; 331; 334; 35; 375; 506; 554; 77
51. **kKorean**
    - count: 9034, min length: 1, max length: 17
    - CANG; CENG; CHIL; CHWUK CHWU; HA; IL; KAL KAY; KI; KYO; MAN MWUK; MYEN; PWU PWUL; SAM; SANG; YE
52. **kLau**
    - count: 37, min length: 3, max length: 9
    - 1130; 1378 1499; 1498; 1549; 1683; 1915 1942; 2366; 2454; 2892; 3332; 3462; 3548 3549; 934
53. **kMainlandTelegraph**
    - count: 7085, min length: 4, max length: 4
    - 0001; 0002; 0003; 0004; 0005; 0006; 0007; 0008; 0009; 0010; 0011; 0012; 0013; 0018; 1413; 5280; 8001
54. **kMandarin**
    - count: 25383, min length: 2, max length: 42
    - CHOU2; DAI4; LIN3; LIU2; NUO4; QIU2; SI4 YI2; TIAN3 TIAN4; WU3; XIN4; XING2; XU4; YE4; YIN3; ZHEN3
55. **kMatthews**
    - count: 8988, min length: 1, max length: 6
    - 1666; 1726; 2753; 2946; 300; 4078; 4279; 5143; 5575; 6273; 6662; 7038; 7044; 7140; 7186; 7187; 7194
56. **kMeyerWempe**
    - count: 224, min length: 1, max length: 9
    - 1; 1458; 2; 23; 30a; 30d; 3315; 3450; 39a; 52a; 54; 63; 68c; 7; 70a; 765a; 79; 84a; 92; 93; 94; 98a
57. **kMorohashi**
    - count: 21204, min length: 5, max length: 6
    - 00001; 00002; 00003; 00006; 00007; 00008; 00010; 00011; 00012; 00013; 00014; 00015; 00019; 00020; 99999
58. **kNelson**
    - count: 5399, min length: 3, max length: 24
    - 0028; 0084; 0127; 0133; 0265; 0677; 1278; 1408; 1476; 1487 1491; 2399; 2453; 2459; 2809; 3629; 5343
59. **kOtherNumeric**
    - count: 3, min length: 1, max length: 2
    - 2; 20; 30
60. **kPhonetic**
    - count: 9887, min length: 1, max length: 5
    - 1038; 1343; 155; 208; 247; 266; 284; 312; 415; 475; 539; 65; 779; 828; 916; 925; 928; 954; 963; 985
61. **kPrimaryNumeric**
    - count: 19, min length: 1, max length: 58
    - 1; 10,000,000,000,000,000 ten quadrillion (American); 10000; 100000000; 1000000000000; 2; 3; 5; 7; 9
62. **kPseudoGB1**
    - count: 153, min length: 4, max length: 4
    - 9232; 9258; 9301; 9304; 9318; 9329; 9331; 9347; 9352; 9365; 9369; 9374; 9375; 9376; 9380; 9390; 9391
63. **kRSJapanese**
    - count: 198, min length: 3, max length: 6
    - 14.3; 22.8; 26.5; 26.7; 30.11; 30.14; 30.19; 30.6; 30.7; 32.7; 32.9; 4.6; 42.6; 52.8; 9.11; 9.15; 9.6
64. **kRSKanWa**
    - count: 157, min length: 3, max length: 6
    - 10.5; 15.4; 18.11; 19.18; 22.18; 27.10; 27.8; 3.2; 3.7; 37.3; 4.5; 4.6; 9.10; 9.13; 9.14; 9.4

65. **kRSKangXi**
    - count: 63687, min length: 3, max length: 7
    - 1.0; 1.1; 1.2; 1.3; 203.3
66. **kRSKorean**
    - count: 20, min length: 3, max length: 6
    - 118.8; 122.9; 125.5; 26.7; 30.11; 57.9; 58.13; 61.11; 64.7; 72.9; 74.7; 85.9; 86.9; 9.15; 9.8; 93.9
67. **kRSUnicode**
    - count: 71098, min length: 3, max length: 7
    - 1.4; 1.5; 2.2; 4.1; 4.5; 5.2; 5.3; 5.4; 5.5
68. **kSBGY**
    - count: 19511, min length: 6, max length: 48
    - 066.03 279.38; 275.10; 310.04 424.03; 328.25; 380.30; 416.56; 442.07 444.28; 446.22; 459.08; 474.39
69. **kSemanticVariant**
    - count: 25, min length: 1, max length: 3
    - •; • 龝; •; •; •; •; 棊 棋; 棊 碁; 棋 碁; 烟; 煙; 猫; 着; 秋; 翥; 著; 葯; 薯; 藥; 藷; •; 衰; •
70. **kSimplifiedVariant**
    - count: 2628, min length: 1, max length: 3
    - •; •; •; •; 个; 么 幺; 乱; •; •; •; •; •; •; •; •; •; •; •; 侠; •; •; •; 俣; •; •; 干; 并; 来; 杰; 系; •; •; •
71. **kSpecializedSemanticVariant**
    - count: 10, min length: 1, max length: 3
    - •; 一; 八; 壹; 捌; •; 烟 煙; 舒; •
72. **kTaiwanTelegraph**
    - count: 9041, min length: 4, max length: 4
    - 0001; 0002; 0003; 0004; 0005; 0006; 0007; 0008; 0009; 0010; 0011; 0012; 0013; 0014; 0015; 8002; 8003
73. **kTang**
    - count: 121, min length: 3, max length: 15
    - biêt4; deng1; djhiäng2; ha2 ha3; jrha3; kiou1; pyi1; qit4; shiëi3; säm1; tsia2; tsit4; zhiäng3 zhiäng2
74. **kTotalStrokes**
    - count: 27786, min length: 1, max length: 2
    - 11; 2; 3; 4; 5; 6; 7; 8; 9
75. **kTraditionalVariant**
    - count: 2553, min length: 1, max length: 7
    - •; •; •; 丟; 個; 兩; 叢; 喎; 喪; 嚴; 專; 撝; •; 東; 業; •; •; 筴; 紬; 絧; 絲; •; •; 與; 萬; 訢; 醜; •; •; 鵁; •; •; •; •
76. **kVietnamese**
    - count: 4516, min length: 1, max length: 6
    - b•i; ch•m; hôn; h•t; khía; l•p; l•ng; mau; m•; nhe; nhái; nh•i; n•m; phay; sùm; u•ng; vòi; v•n; êu
77. **kXerox**
    - count: 9747, min length: 7, max length: 7
    - 241:042; 241:044; 241:052; 241:113; 241:120; 241:345; 246:341; 250:132; 253:120; 253:252; 316:252; 317:164
78. **kZVariant**
    - count: 5758, min length: 1, max length: 1
    - 丄; 丅; 上; 下; 世; 丗; 丟; •; •; 井; 兩; 其; •; 叢; 喪; 嚴; •; 東; 業; 為; 爿; 絲; 臨; 與; 萬; 豐; 酉; 醜; 麗; 龝; •; •; •

Note: I have also run across some problems in some of the 'provisional' data; I have filed bugs directly with John on those.

---

9. Line Break Pair Tables

The tables below match the TR14 ordering, and use the notation described in the TR. The table is extended by also including SP..CB, and the values L, V, and T for Hangul Jamo. If your browser is enabled for tool-tips, then hovering over the cell reveals the Rule number that determines the breaking status in the case in question. Sometimes there are multiple rules, when a case has to be tested with and without intervening spaces. The differences between the two are marked in yellow.

This does not imply that the current layout of the table in the TR should be changed to be as large as the one below. The more complete table below is simply provided to illustrate the effects of the recommended changes.

Current:

	OP	CL	QU	GL	NS	EX	SY	IS	PR	PO	NU	AL	ID	IN	HY	BA	BB	B2	ZW	CM	SP	BK	CR	LF	CB	L	V	T
OP	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^
CL	_	^	%	%	^	^	^	^	_	%	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
QU	^	^	%	%	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	^	^	^	^	^	%	%	^	^
GL	%	^	%	%	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	^	^	^	^	^	%	%	^	^
NS	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
EX	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
SY	_	^	%	%	%	^	^	^	_	_	%	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
IS	_	^	%	%	%	^	^	^	_	_	%	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
PR	%	^	%	%	%	^	^	^	_	_	%	%	%	_	%	%	_	_	^	^	^	^	^	^	_	%	^	^
PO	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
NU	_	^	%	%	%	^	^	^	_	%	%	%	_	%	%	%	_	_	^	^	^	^	^	^	_	_	^	^
AL	_	^	%	%	%	^	^	^	_	_	%	%	_	%	%	%	_	_	^	^	^	^	^	^	_	_	^	^
ID	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	^	^
IN	_	^	%	%	%	^	^	^	_	_	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	^	^
HY	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
BA	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
BB	%	^	%	%	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	^	^	^	^	^	%	%	^	^
B2	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	^	^	^	^	^	^	^	_	_	^	^
ZW	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	^	_	^	^	^	^	_	_	_	_
CM	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	^	^
SP	_	^	_	_	_	^	^	^	_	_	_	_	_	_	_	_	_	_	^	^	^	^	^	^	_	_	^	^
BK	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
CR	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	%	_	_	_	_
LF	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
CB	_	^	%	%	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	^	^
L	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	%	^	^
V	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	^	^
T	_	^	%	%	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	^	^

Recommended:

	OP	CL	QU	GL	NS	EX	SY	IS	PR	PO	NU	AL	ID	IN	HY	BA	BB	B2	ZW	CM	SP	BK	CR	LF	CB	L	V	T
OP	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^	^
CL	_	^	%	^	^	^	^	^	_	%	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_
QU	^	^	%	^	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	^	^	^	^	^	%	%	%	%
GL	%	^	%	^	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	^	^	^	^	^	%	%	%	%
NS	_	^	%	^	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_
EX	_	^	%	^	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_
SY	_	^	%	^	%	^	^	^	_	_	%	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_
IS	_	^	%	^	%	^	^	^	_	_	%	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_
PR	%	^	%	^	%	^	^	^	_	_	%	%	%	_	%	%	_	_	^	^	^	^	^	^	_	%	%	%
PO	_	^	%	^	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_
NU	_	^	%	^	%	^	^	^	_	%	%	%	_	%	%	%	_	_	^	^	^	^	^	^	_	_	_	_
AL	_	^	%	^	%	^	^	^	_	_	%	%	_	%	%	%	_	_	^	^	^	^	^	^	_	_	_	_
ID	_	^	%	^	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	_	_
IN	_	^	%	^	%	^	^	^	_	_	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	_	_
HY	_	^	%	^	%	^	^	^	_	_	%	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_

BA	_	^	%	^	%	^	^	^	_	_	_	_	_	_	%	%	_	_	^	^	^	^	^	^	_	_	_	_
BB	%	^	%	^	%	^	^	^	%	%	%	%	%	%	%	%	%	%	^	^	^	^	^	^	_	%	%	%
B2	_	^	%	^	%	^	^	^	_	_	_	_	_	_	%	%	_	^	^	^	^	^	^	^	_	_	_	_
ZW	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	^	_	^	^	^	^	_	_	_	_
CM	_	^	%	^	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	_	_
SP	_	^	_	^	_	^	^	^	_	_	_	_	_	_	_	_	_	_	^	^	^	^	^	^	_	_	_	_
BK	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
CR	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	%	_	_	_	_
LF	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
CB	_	^	%	^	_	^	^	^	_	_	_	_	_	_	_	_	_	_	^	^	^	^	^	^	_	_	_	_
L	_	^	%	^	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	%	%	_
V	_	^	%	^	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	%	%
T	_	^	%	^	%	^	^	^	_	%	_	_	_	%	%	%	_	_	^	^	^	^	^	^	_	_	_	%