L2/99-182

1999-06-07

Old comments on LineBreak (P.S. some of these may have been superceded by later changes in the text.)

M. Davis

Data:

- CM should be retitled IG (for ignore--since that's what it really is). Prevents confusion.
- AL should be retitled as something more generic, like Normal.
- Generic quotes should be in the QU bucket.
- Need separate categories for CR and LF for completeness. They are not quite the same a BK.

>The Unicode Standard, Version 2.0, treated the line-breaking behavior of characters largely as self-evident. This technical report intends to discover best practice and capture it via formally assigned line breaking properties. The report assigns normative line-breaking properties to those characters that have a specific function in the process of line breaking. Default, informative line-breaking properties for all other classes of characters are supplied as well.

Change to:

The Unicode Standard, Version 2.0, presents a summary of basic line-breaking behavior, but is not sufficient for implementation. This technical report provides more complete information about best practice. Normative line-breaking properties are assigned to those characters whose line-breaking behavior must be identical across all implementations. Default, informative line-breaking properties for all other classes of characters are supplied as well.

Definitions for "overfull", "underfull", and "optimal". There appears to be a tautological relationship between some of these that should be avoided.

Line Breaking Property

The table shouldn't be in the defintion section.
[start table comments]
Erop the "...".

Add examples of BA, NS

Star the ones that are normative

Group similar items together. e.g. "prevent a line break before" with "prevent a line break before, even across a space"

>break before or after if paired with atomic"
doesn't make sense. Do you mean allow breaks within pairs?

>don't break in front of a numeric expression
including across space?

>are alphabetic characters and symbols.

I disagree that symbols should be treated like alphabetics. In "abc÷cde" for example, there is no reason not to have break opportunities after the symbol, just like slash. Move symbols to SY, reword in a few places.

Drop XX

Where there are correlations between General Categories, include them in the examples, e.g. Alphabets (most Lu, Ll, Lm, Lo, except for ideographs.

Make the acronyms correspond to the title. E.g. Either change Atomic to Ideographs, or ID to AT. Fix some oddities. If BB is break before, I'd expect BA to be break after.

My strong preference is to name the behavior where possible. e.g. BA for Break After, rather than SP.

[end of table comments]

>Hyphenation improves the layout of narrow columns, but is often considered less optimal. It is therefore optional in languages with short words.
This is unnecessarily controversial. Hyphenation is often more optimal than the results if you couldn't. Just say that the choice of the optimal break opportunities, including at hyphenation points, is outside the scope of this document.

>below sometimes assumes that the formatting algorithm ignores the width
remove "sometimes"

1. space-based
   This is not, strictly-speaking, correct and might be better retitled. Publishing practice allows breaking after symbols, etc.

>All line breaking properties are informative, except for the following small subset
Mark these in the table, and remove this section.

>A - the property introduces a break opportunity after in all or some contexts
The phrasing for all these annotations I think could be improved; properties don't introduce. (Also, indent the list.)
A - Break opportunities are allowed *after* characters with this property (in specified contexts)
...
> Unicode equivalents of NL, LF, and CR
Drop "equivalents of".

>NOTE: If SPACE is used to show combining characters in isolation and the line is broken *after* the space character, the next line would start with the combining characters. In this case they are rendered *as if* they followed a space. As a result, it is always possible to maintain the correct rendering for combining character sequences and still process space characters in an optimized way.

No. CM are only ever at the start of a line if they are at the very start of text, or follow a non-base character (e.g. PS). They must never be broken from space.

>If they follow a space character, they still allow a break
A break is permitted between a space character and a glue character. This however contradicts your statement below that
In particular, surrounding SPACE with
ZWNBSP prevents it from acting as a line break opportunity

In general, you have to be careful with the phrasing "X acts as a line break opportunity" That is incorrect. You can say that there is a LBO after X, or before X, but the other phrasing is ambiguous.

>Contingent Break Opportunity
"contingent" and "normative" are contradictory. Remove this from the normative properties. Remove the word contingent, since it is unnecessary.

Important: All the normatives should come before informatives--otherwise the behavior of the latter affects the former, making the former non-normative.

Also, move spaces higher in the list, above "inseparables".

>ZERO WIDTH SPACE (ZWSP) – U+200B
This character does not have width. It is used in a style 2 context analysis to provide additional (invisible) break opportunities.
This is misleading, since you are discussing a combined style 1 and 2 approach. Fix wording here and other places.

>NOTE: In some practice runs of HYPHEN-MINUS are used
In some circumstances,

>OP - Opening characters (XA)
The opening character of any set of paired punctuation must be kept with the following character
Characters of general category Ps in the Unicode Character Database.
You don't seem to describe French punctuation practice, where spaces are permitted.

CL - Closing characters (XB)

The closing character of any set of paired punctuation must be kept with the preceding character
Characters of general category Pe in the Unicode Character Database.
You don't seem to describe French punctuation practice, where spaces are permitted. Also, this should include colon. That would seem to be only reason not to merge this and NS, so I was surprised not to find it.

>QU - Ambiguous Quotation mark Characters (XB/XA)
Some paired characters can be either opening or closing depending on usage. The default is to treat them as both opening and closing.
Smarter systems can deduce the parity of these characters based on surroundings. For example, in abc"]..., the quotation mark can be treated as a closing mark.

>These behave like Closing characters, except in relation to postfix characters
This needs more clarification; just exactly why are they different?

General: In all of the descriptions, add an example showing where it would or wouldn't break.

>ID - Atomic characters (B/A)
Do not require other characters to provide break opportunities, can ordinarily break between pairs.

HIRAGANA (except small characters)
KATAKANA (except small characters)

Throughout: Prefix and Postfix characters are badly named: they should be Numeric Prefix and Numeric Postfix

PO includes "preceding closing characters" I assume this is for "(12) ¢". Why is this not with PR, for "$ (12.20)"?

Here again, examples would help immensly to motivate the description.

>**SA** - Complex-context dependent characters (P)
The only purpose for this is for efficiency; to call out to a dictionary when you hit, say, a Thai character. The dictionary then returns the boundaries. As you do with hyphenation, this can be handled by merging this with AL, and having a note in AL that dictionary lookup can map some of these characters to BA in order to allow line breaks.
Failing that, you should say that if dictionary lookup is not available, SA should behave like AL.

> **BA** - Break opportunities after characters (A)
Conflicts with BA meaning Before and After.
All classes used here should be in the main table; this section should only add characters to the categories.

IMPORTANT. This section can't add characters to a normative category; otherwise the category is not really normative.

I like the new section on dictionary usage.

>A position P in the string identifies the space between two characters
I'd prefer the term "offset". Change "the space"; confusion with SPACE. Give example, as follows:
For example, in "ab", offset 0 is immediately before the 'a', offset 1 is between 'a' and 'b', and offset 2 is immediately after the 'b'.

A1. Position N+1 is always a break, position 0 is never a break.
Offset N, not N+1

As you say, 7.1 and 7.2 need to be combined. I will follow up with that while you are digesting this.

>The JIS standard uses 20 classes of which only 14
The JIS standard uses 20 classes, of which only 14
(comma after classes)

>The following example table implements the line breaking behavior described in this Technical Report.
Change to:
The following example table closely <i>approximates<i> the line breaking behavior described in this Technical Report.

The table must include ALL the classes from *Line Breaking Property* Table, including BK, SP, etc.
The table should just use the same notation as the rules, e.g. ^ for allow a break.

>X means: Never break here (^ below)
S means: Allow break here if space intervenes (% below)
Otherwise allow a break (‡ below)

What the heck does S really mean!? This phrasing is very wierd; that's why I was mixed up earlier. Tell me, if A/B is marked with S, what does that mean in these two examples:

A SPACE B
A B

Which of the following does it mean (using ^ for break, $ for no break (since double-dagger isn't 8859)) for each line:

1a. A ^ SPACE ^ B
1b. A ^ SPACE $ B
1c. A $ SPACE ^ B
1d. A $ SPACE $ B

2a. A ^ B
2b. A $ B

>Equivalent Rules

Rather than have a separate set of rules, the Equivalent Rules should be merged in with 7.1/2, with marks to show where a rule is approximated by the table.

>The following two functions demonstrate how the pair table is used.

This code is too obscure: for exposition it should be something like the following.

```
// True pair table

byte breakType[];
boolean pairBreaks[][];

boolean isLineBreak(unichar chars[], int offset, int len) {
 if (offset <= 0 || offset >= length) return true;
 int typeBefore = findType(chars, i-1);
 int typeAfter = findType(chars, i);
 return pairBreaks[typeBefore][typeAfter];
}

byte findType((unichar chars[], int offset) {
 byte result = breakType[chars[offset]];
 if (result != SA) return result;
 return dictionaryLookup(chars, offset);
  // dictionaryLookup returns either AL or BA
}

//═══════════════════════════════

// Changes to handle augmented pair table (with space)
(I really can't tell from your text exactly what SS means, but
```

here is a crack at it)

```
byte pairBreaks[][];

boolean isLineBreak(unichar chars[], int offset, int len) {
 if (offset <= 0 || offset >= length) return true;
 int typeBefore = findType(chars, i-1);
 int typeAfter = findType(chars, i);
 int breakType = pairBreaks[typeBefore][typeAfter];
 if (breakType == SS && offset > 1 && chars[i-1] != SPACE) {
   breakType = NO_BREAK;
 }
 return (breakType == BREAK);
}
```

>A real world line breaking algorithm must be tailorable to some degree. There are three principle ways of tailoring a table based algorithm:

I only saw 2 in my copy. Also change "table" to "pair-table"

I would phrase them as the following classes, in order of easy to hard.

1. change the character breaking types assigned to characters
2. change the table value assigned to pairs of character types
3. add new types
4. augment the algorithm

#1 is the easiest, and probably the most common. #3 requires changing the dimensions of the array

>*7.6 Examples of customization:*
Tell what class (7.5) each of the examples falls under, and give at least one example per class

>Non-spacing marks are often not handled explicitly. If they are applied to a space character, naive algorithms will move them to the next line where they would possibly overhang the beginning margin. However, displaying non-spacing marks in isolation is rare in general purpose text. Applying them

Drop, and replace by "Applying non-spacing marks"

As you say, before coming up for approval by the UTC, this needs to have a human-readable datatable with the precise assignments of classes to characters.