L2/24-057

Title: Proposal of the UNICODE AUTO SPACING
Authors: Koji Ishii (Google), Yasuo Kida (W3C), Fuqiao Xue (W3C)
Date: Feb 28, 2024

UNICODE AUTO SPACING (Proposal)

1 Overview and Scope

East Asian text usually consists of multiple scripts, such as Han ideographs, Kana syllables, and Hangul syllables, along with Latin letters and numeric characters. In their established typography convention, a thin spacing between East Asian scripts and other scripts can improve the readability.

While detailed rules of the spacing can vary across documents, it is important that the choice made by an author for a specific document be clearly established, so that a rendering system can display what the author intended. It is also important that this choice be established independently of the font resources, as the rendering systems may have to use other fonts than those intended or specified in the document. Finally, the expression of the author’s choice should be relatively concise, to facilitate document authoring and minimize document size.

This report describes a Unicode character property which can serve as a stable default rule of inserting the spacing for the purpose of reliable document interchange.

For the purpose of reliable document interchange, this property defines an unambiguous default value, so that implementations could reliably render a character stream based solely on the property values, without depending on other information such as provided in the tables of the selected font.

The intent is that the author should be able to specify where they want to override, and that in the absence of an explicit specification, the spacing is implicitly that defined by the property presented in this report.

The actual choice for the property values should result in a reasonable or legible default, but it may not be publishing-material quality, and it may not be a good choice if used in a specific style or context.

The property values are chosen first to match existing practice in East Asian contexts in their respective environments. For characters that are not generally used in such environments, similarity to existing characters has been taken into consideration. It also takes East Asian characters in non-East Asian texts into account.

2 Conformance

The property defined in this report is informative. The intent of this report is to provide, in the absence of other information, a reasonable way to determine the correct automatic spacing, but this behavior can be overridden by inserting space characters, or by a higher-level protocol, such as through markup or by the preferences of a layout application. This default determination is defined in the accompanying data file [Data], but in no way implies that the spacing is inserted only by this rule.

For more information on the conformance implications, see [Unicode], Section 3.5, Properties, in particular the definition (D35) of an informative property.

3 The Auto_Spacing Property (as)

3.1 Property Values

The possible property values are given in Table 1.

Table 1. Property Values

Value	Description	Examples
W	Characters that are considered as East Asian script for the auto-spacing purpose.	Han ideographic characters and Kana syllables are examples of this value.
N	Characters that need the auto-spacing with adjacent W characters.	Latin letters and digits are examples of this value.
O	Characters that don’t need the auto-spacing.	Most symbols and punctuation characters such as COMMA and FULL STOP are examples of this value.

NOTE: A possible addition of “language conditional” is under discussion; specifically, “Conditional-Chinese N/O”. Please see Symbols and Punctuation Characters for details.

Characters that have the property value N are similar to the “Narrow” characters in UAX#11 EAST ASIAN WIDTH, but most punctuation characters and symbols are excluded. Similarly, characters that have the property value W are similar to the “Wide” characters, but most punctuation characters and symbols are excluded. Also, to follow the existing practice, Hangul characters, circled characters, square characters, and Emoji are defined as O.

3.2 Spacing Algorithm

The auto spacing should be inserted between “W” and “N”, and between “N” and “W”.

The exact amount of the spacing can vary across documents. This property doesn’t define the exact amount. Instead, it should be defined by high-level protocols or applications such as through markup or by the preferences of a layout application.

There are two ways to represent a space: a character space (by the insertion of physical code points, or in a glyph space (like kerning, adjusting the metrics of adjacent glyphs on the device). A glyph space is recommended for high-level protocols or applications that can represent glyph spaces.

3.3 Scope of the Property

3.3.1 Grapheme Cluster

As in all matters of typography, the interesting unit of text is not the character, but a grapheme cluster: it does not make sense to insert the auto spacing between a base character and a combining mark attached to it. Implementations should insert the auto spacing before or after each grapheme cluster.

A possible choice for the notion of grapheme cluster is either that of legacy grapheme cluster or that of extended grapheme cluster, as defined in [UAX29].

The property value for a grapheme cluster as a whole is then determined by taking the property value of the first character in the cluster, with the following exception:

- If the cluster contains an enclosing combining mark (general category Me), then the whole cluster has the Auto_Spacing property value O.

3.3.2 Space Characters

The property values for space characters (General Category Zs) are O. This is to avoid inserting the auto spacing around space characters, which can lead to undesirable double spacing.

It also allows authors to override the algorithm when high-level protocols or applications don’t provide a way for authors to express their intentions to override this algorithm, such as in plain text files. For example, in East Asian contexts, U+0020 SPACE should usually represent a wider space than the auto spacing, indicating a semantic boundary stronger than the auto spacing, while U+2006 SIX-PER-EM SPACE should usually represent a thin space similar to the auto spacing for the readability. Inserting U+2006 SIX-PER-EM SPACE to where the algorithm doesn’t insert the auto spacing should indicate that the auto spacing is desired there. Likewise, inserting U+200B ZERO WIDTH SPACE to where the algorithm inserts the auto spacing should prevent the auto spacing from being inserted by rendering systems.

3.3.3 Symbols and Punctuation Characters

In some existing practices, symbols and punctuation characters insert the auto spacing, while they don't insert the auto spacing in other existing practices.

The motivation to insert the auto-spacing is that words such as "20%", "$20", or "C#" would look *unbalanced* if the auto-spacing is inserted to the letters and digits but not to the symbols and punctuation characters. It also matches the existing practice of widely adopted style guides for East Asian plain text defining to insert U+0020 SPACE between any wide and narrow characters with some exceptions. Large amount of existing plain text files in East Asian writing systems follow this convention for over decades, and one of existing implementations as well.

On the other hand, traditional printing typography often accepts such *unbalanced* spacing as good results, from a perspective that the spacing is to prevent characters from being too close together, not to highlight words like parentheses do [JLREQ-TF]. Another existing implementation follows this convention for over decades too.

Please see the "Open Issues" section for more information and possible options on this point.

3.3.4 Vertical Text Layout

In vertical text layout, a character may be displayed upright or sideways rotated, as defined in [UAX#50].

If a character that has the Auto_Spacing property value N is displayed upright, the rendering system should handle it as if it has the property value O instead.

3.3.5 Right-to-Left Scripts

This property has a current limitation in that the handling of right-to-left scripts is not specified. This includes scripts that are predominantly written right to left, such as Arabic, along with right-to-left scripts that are meant to be written vertically, such as Chorasmian.

4 Data File

https://github.com/kojiishi/unicode-auto-spacing/blob/main/auto-spacing.txt

Currently. the property values of this data file is derived from existing properties using the following algorithm (see NOTEs below):

- A code point has the property value W if it's in the following set:
    - Include if the script property is one of the following values, or if the script_extension property contains one of the following values:
        - Han, Tang, Kits, Nshu, Hira, Kana, Bopo
    - Excluding if the General Category property is one of following values:

        - P*, S* except Sk, No
    - Excluding if the East_Asian_Width property is H.
- A code point has the property value N if it's in the following set:
    - Include if the General Category property is one of following values:
        - L*, M*, Nd
    - Excluding the set for the value W.
    - Excluding if the script property is Hang, or if the script extension property contains Hang.
    - Excluding if the East_Asian_Width property is F or H.
- A code point has the property value O if it's not in either set above.

There is a python code that generated the data, for reference.

NOTE: This algorithm isn't final yet.
NOTE: There is a possible addition of code point lists, making this not a fully derived property. Please see the Open Issues below.

5 Open Issues

1. One existing implementation defines some symbols and punctuation characters as N. An example is U+0025 PERCENT SIGN, so that a word such as "200%" can have the auto spacing on both sides, and there are more such code points than "%". See also "3.3.2 Symbols and Punctuation Characters" and #11.
    a. Not to include symbols and punctuation characters. This matches one existing implementation and the JLREQ proposal.
    b. Take the CLREQ proposal. This inserts the auto space in a lot more cases than all existing implementations. The current differences in the data file are listed here.
    c. Define a language conditional value to accept both CLREQ and JLREQ proposals. That value will work as N for Chinese and O for other writing systems. Those characters having N will impact English text with Chinese characters.
    d. Find a list of "code points that have less side effects" which both parties can reach consensus on.
2. A suggestion to use the Ideographic property is under discussion. The diff is now very small as the algorithm evolves. See also #1.