In affirmation of Martin Hosken's proposal for spacing marks, I would like to provide some additional examples comparing our current approach with a potentially new one.

Currently, most OpenType engines ignore the advance width of any glyph classified as a mark in the GDEF table. We can consider this the defacto behavior although the standard does not set any clear expectations for this matter.

We will resort to a realistic example from Burmese script because its geometric shapes convey the idea simply and clearly. In Burmese, it is quite common to see consonant clusters that are rendered as a consonantal symbol below which stands a subscript form of the second consonant.

For example, the sequence of Unicode characters "u1002 u1039 u1002 u1014 u1039 u1001 u1009" renders as follows:

Since u1039 indicates that the successor character should be shown in subscript form, we could rewrite the above expression more clearly as

[u1002 + subscr(u1002)] [u1014 + subscr(u1001)] [u1009]
The square brackets indicate each vertical stack of glyphs.

To further clarify the elements in this graphic, we will apply some color coding where blue indicates a base, while red a combining mark representing subscript glyphs.

In the above case, the size of each subscript form fits within the width of the base above it. Now, had we chosen a different sequence (u1002 u1039 u1002 u1014 u1039 u1010 u1009), we would have obtained the following visual result in which the actual width of the second subscript exceeds that of the base carrying it, resulting in collisions with its right and left neighbors.

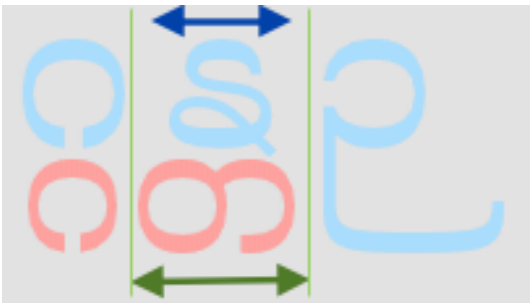
There are several ways of adjusting the inter-character spacing in this example, but they all require increasing the advance width of base glyphs and/or shifting them within their bounding box. To make the next part of the discussion more understandable, we will tag the first two base glyphs in the graphic.

We will use the technique of increasing the advance width of the first base, followed by increasing that of the second base. Here is the result of the first adjustment:

After the second adjustment:

Making these adjustments depends on relatively complex logic in which we utilize contextual rules to detect a wide subscript form that is constrained below the baseline on one or more sides. Once detected, we go on to apply an appropriate change to the advance width of each base glyph.

Now, if we could tell the OpenType engine not to ignore the advance width of the mark attached to base glyph 2, then we would have achieved the necessary width adjustment by simply giving the mark an appropriate advance width. Of course, to achieve this desired result, we must also clarify the semantics of spacing marks and their interaction with base glyphs by defining a 'glyph cluster'. In the above example, base glyph 2, in combination with the subscript mark attached below it, make up a glyph cluster. Currently, OpenType engines are spacing each glyph cluster on the basis of the base glyph in it. For a future version of the OpenType standard, we are proposing new semantics for spacing marks that will calculate the width of a glyph cluster as the aggregate width of all its components.



In the above illustration, the width of the base is shown in blue, while that of the subscript is in green. The extent of the glyph cluster is derived from the furthest reach to the right and left, shown here as green vertical lines.

So, with the new semantics of the glyph cluster, we could altogether avoid any collisions with neighboring glyphs simply by assigning the correct width to the pertinent spacing mark.