Comments on draft of 2024-06-26 from Behdad.

VARC

6.3.10.2 [7.3.10.2]

VARC table header

Only conditionListOffset has an Offset in the name. The rest (coverage, varStore, etc) are all offsets too, without Offset in the name.

Table, *Variable Component Flags*

The last line of the flags table (buts 15-31) now says "RESERVED_MASK", but must also say "Set to 0". That piece of information is not included anywhere. It is important for future extensions, that compilers set reserved to 0. Otherwise we cannot define new flags in the future as they will be misinterpreted.

Page following Variable Component Flags

“The following pseudo-code illustrations the construction”: illustrations should be illustrates.

7.3.10.4 Hinting of Variable Composite Glyphs

I don't like that the CFF hinting comes before “7.3.10.5 Processing of Variable Composite Glyphs”. I highly suggest moving the CFF BS to the end of the VARC section, after Processing.

Coverage

5.6.22 [6.2.5] Coverage table

1. “In a Coverage table, a format field specifies the format as an integer 1 = lists, and 2 = ranges.” Remove it? Doesn’t include formats 3 & 4.
2. In the table CoverageFormat3, make glyphCount be uint24 instead of uint16.
3. In the table CoverageFormat4, make rangeCount be uint24, instead of uint16.

5.6.23 [6.2.6] Class Definition table

1. In the table, ClassDefFormat3, make glyphCount be uint24, instead of uint16.
2. In the table, ClassDefFormat4, make classRangeCount be uint24, instead of uint16..

GSUB

5.7.4.2.1 [6.3.4.3.1] GSUB header, version 1.2

1. There's mention of ScriptList2, FeatureList2, and LookupList2. But there are no such types. The "2" should be dropped from the *type names*. The *field names* are correct. GPOS has it correct.

5.8.4.4.3 Lookup type 3 subtable: alternate substitution

1. In the AlternateSubstFormat2 subtable, alternateSetOffsets[alternateSetCount] should be Offset24, instead of Offset32. This would match MultipleSubstFormat2.

GPOS

6.3.3.1 Lookup Type 1 subtable: Sungle adjustment positioning

1. SinglePosFormat3 is curious. It wasn't in my design AFAIR. It's NOT necessary for 24bit gid support. The only difference it has with SinglePosFormat1 is that it uses Offset32 for coverage instead of Offset16. So the comment "Format 3 is identical to fFormat 1, except that it was modified to support 24-bit glyph ID values." is inaccurate. At this point I don't suggest removing the format though… [Liam notes: SinglePosFormat3 was in our proposal, and we changed coverageOffset to Offset32 in January, supposedly to avoid possible overflow]

6.3.3.3.2 Lookup Type 2 subtable: pair adjustment positioning

1. Page 495: PairSet2 table must refer to a new type PairValue2, that is like PairValue but uses 24bit gids instead of 16. This was in my original design, but seems to have disappeared. [Liam: my bad for that one, it's not in our proposal and we didn't notice in review, new text:]

In Format 3, the PairSet2 table uses 24 bits for the number of pairs, to avoid overflow.

PairSet2 table

Type	Name	Description
uint24	pairValueCount	Number of PairValue records.
PairValue2	pairValueRecords [pairValueCount]	Array of PairValue2 records, ordered by glyph ID of the second glyph.

PairValue2 record

Type	Name	Description
uint24	secondGlyph	Glyph ID of second glyph in the pair (first glyph is listed in the Coverage table).
ValueRecord	valueRecord1	Positioning data for the first glyph in the pair.
ValueRecord	valueRecord2	Positioning data for the second glyph in the pair.

Mark-to-ligature attachment positioning format2

1. Second row of LigatureArray2 table, “Array of offsets to LigatureAttach tables.” should be “Array of offsets to **LigatureAttach2** tables.LigatureAttach2 tables”.

2. Then below it the header “LigatureAttach table” should also be “LigatureAttach2 table”.

SequenceContext / ChainSequenceContext

5.6.24 Common formats for contextual lookup subtables

1. : “(There was no “Chained sequence context formats 6” defined because “Chained sequence context formats 3” encodes only one input sequence pattern and is not likely to overflow.)” There’s a discrepancy here. Apparently we did add SequenceContextFormat6, but no ChainSequenceContextFormat 6. This doesn’t make sense to me. We should either include both or omit both. If anything, ChainSequenceContext3 is more likely to overflow than SequenceContext3. Can you check which ones were in our proposal? I’m leaning towards removing SequenceContextFormat6.

    Liam response: I have no record of adding a SequenceContextFormat6, and we did not add a ChainSequenceContextFormat6. However, I see in the document, Sequence context format 6 is identical to format 3, except that it was modified to support larger offsets to avoid possible overflow. OK with removing SequenceContextFormat6.

SequenceContextFormat4 table

1. make seqRuleSetCount a uint24 instead of uint16.

2. seqRuleSetOffsets points to SequenceRuleSet2 tables, not SequenceRuleSet. The subsequent type should also be called “SequenceRuleSet2 table”, not “SequenceRuleSet table”.

Sequence context format 5: class-based glyph contexts

1. Definition of ClassSequenceRule here is redundant. That struct has already been defined earlier.

Chained sequence context format 4: simple glyph contexts

1. In *ChainedSequenceContextFormat4 table*, let’s make chainedSeqRuleSetCount a uint24 instead of uint16.

2. In *ChainedSequenceRuleSet2 table* make chainedSeqRuleCount a uint16 instead of uint24. Yes.

3. In *ChainedSequenceRule2 table*, lookaheadSequence should be uint24 instead of uint16.

4. *ChainedClassSequenceRule2 table:* In my design, IIRC, was never a ChainedClassSequenceRule2 and we used the existing ChainedClassSequenceRule. This is because this uses class values, not glyph indices, and we didn’t extend class values to go beyond 16bit. So a version with uint24’s is wrong For the same reason, we didn’t add ClassSequenceRule2. [so, r=ChainedSequenceContextFormat5 should use ChainedClassSequenceRule, not a new ChainedClassSequenceRule2, which is not needed]

GDEF

Previously I thought LigCaretList needed no extension. There was an assumption that the font will not have more than 65535 ligatures. However, that argument is broken, because even for 65535 ligature glyphs (or far below it), the Offset16's will overflow. As such, I suggest we add an extended type as follows:

Ligature caret list table

1. Page 469 ("Example 4" LigCaretList table) Make ligCaretListOffset2 point to a LigCaretList2 instead of current "ligature caret list table":

2. After the LigCaretList table, define LigCaretList2 table, that is like LigCaretList, but uses Offset32 coverageOffset, uint24 ligGlyphCount, and Offset24 ligGlyphOffsets[ligGlyphCount]:

LigCaretList2 table

Type	Name	Description
Offset23	coverageOffset	Offset to Coverage table, from beginning of LigCaretList2 table.
uint24	ligGlyphCount	Number of ligature glyphs.
Offset24	ligGlyphOffsets[ligGlyphCount]	Array of offsets to LigGlyph tables, from beginning of LigCaretList2 table, in Coverage index order.

6.3.2.1.3 [6.3.2.4.6] Mark Glyph Sets table

- I'm tempted to add MarkGlyphSets format 2, which uses a uint32 markGlyphSetCount, to allow more than 65535 different mark glyph sets, since we've been removing all kinds of practical limits.

[end]