INTERNATIONAL ORGANISATION FOR STANDARDIZATION

ORGANISATION INTERNATIONALE DE NORMALIZATION

ISO/IEC JTC 1/SC 29/WG 3

CODING OF MOTING PICTURES AND AUDIO

ISO/IEC JTC 1/SC 29/WG 3 mNNNN

Online — April 2024

Title: Feature Variations: New LookupVariation Mechanism

Author: Skef Iterum (Adobe Inc., siterum@adobe.com)

Introduction

This introduction is to give context and is not itself proposed text.

This proposal adds an alternative to the FeatureVariationRecord mechanism in the FeatureVariations Table. The initial sketch of the new mechanism was developed by Behdad Esfahbod and Skef Iterum in discussion on GitHub. It also adds negated conditions.

I have substantially restructured the content having to do with feature variations generally, both to improve the flow between topics and concepts when read in order and to prepare for adding conditions to VARC.

The remainder of this document is the restructured text in its entirety, with new versions of sections 6.2.9 and 6.2.10 replacing the existing section 6.2.9. New text is highlighted. Existing text, which in many cases has moved, is plain. Editorial notes are in brackets, with most indicating when and where sections of the current text that are not reproduced should be added.

6.2.9 Conditions and Condition Sets

Condition Table

A condition table describes a particular boolean—that is, either true or falsecondition. Different condition table formats may be defined, with each format used for a particular kind of condition qualifier. There are currently four formats, each of which is parameterized by the same factor: the variation instance of a variable font.

New condition table formats may be added in the future, some of which may be parameterized by different factors. If a layout engine encounters a condition table with an unrecognized format it should treat the condition as false and continue to test other condition sets. In this way, new condition formats can be defined and used in fonts that can work in a backward-compatible way in existing implementations.

Condition Table Format 1: Font Variation Axis Range

[This subsection is unchanged]

Condition Table Format 2: Negated Font Variation Axis Range

The fields in this table after Format are identical to the fields for Format 1, and are interpreted the same way. The difference is that whenever a Format 1 condition is true the Format 2 condition with those field values is false, and vice versa.

The ability to easily negate a condition makes it easy to perform certain kinds of logical analysis and transformation, such as translating an arbitrary boolean expression of conditions into disjunctive normal form.

Type	Name	Description
uint16	Format	Format = 2

Type	Name	Description
uint16	AxisIndex	Same as in format 1
F2DOT14	FilterRangeMinValue	Same as in format 1
F2DOT14	FilterRangeMaxValue	Same as in format 1

Condition Table Format 3: Condition Value

[This section is the same as the existing “Condition Table Format 2”, except with “format 3” replacing “format 2” wherever it occurs.]

Condition Table Format 4: Negated Condition Value

The fields in this table after Format are identical to the fields for Format 3, and are interpreted the same way. The difference is that whenever a Format 3 condition is true the Format 4 condition with those field values is false, and vice versa.

Type	Name	Description
uint16	Format	Format = 4
int16	Default	Same as in format 3
uint16	DeltaSetOuterIndex	Same as in format 3
uint16	DeltaSetInnerIndex	Same as in format 3

ConditionSet Table

A condition set table specifies a set of conditions that are conjunctively related (boolean AND): when all of the specified conditions are true, the set is considered to apply; otherwise it does not apply. A condition set may specify conditions parameterized by various factors; currently, one factor is supported: the variation instance of a variable font.

A condition set does not need to specify conditions for all possible factors or all aspects of a given factor. For example, if the location on one axis of a variable font is not relevant to the use of a condition set, its conditions can simply omit any reference to that axis. Accordingly, when a condition set contains *no* conditions it *always* applies.

The particular use or meaning of a condition set table is context-dependent, but in every case it divides the space of its factors into two: the region or regions in which the condition set applies, and the region or regions in which it does not apply. These are respectively equivalent to the space in which all of its conditions are true and the space in which at least one of its conditions is false.

[The table-table is unchanged]

6.2.10 Feature Variations

As described in 6.2.5, the GSUB and GPOS tables contain Feature List tables (FeatureList) that enumerate a set of Feature tables. Each Feature table can be identified by its FeatureIndex—its index in the FeatureRecord array of the Feature List table.

Feature variations provides two mechanisms for altering the behavior of a given Feature table based on the value of one or more condition sets. One mechanism is the array of FeatureVariationRecords introduced in version 1.0 of the table. Using this system one can replace the Feature table with a given index with a different, complete table when a condition set is met.

The second mechanism is the array of LookupVariationRecords added in version 1.1. This system is lookup-oriented, allowing individual lookups to be chosen according to when a condition set is or is not met. Lookups that are chosen are then collected into a set which either replaces or supplements the lookups of the original Feature table.

While both systems provide a means of altering the set of lookups associated with a feature, LookupVariationRecords provide a more compact encoding when more than a few lookups need to be changed within a GSUB or GPOS table.

The FeatureVariations Table has two versions, 1.0 and 1.1:

FeatureVariations table

Type	Name	Description
uint16	majorVersion	set to 1
uint16	minorVersion	set to 0
uint32	featureVariationRecordCount	Number of feature variation records.
FeatureVariationRecord	featureVariationRecords[featureVariationRecordCount]	Array of feature variation records.

Type	Name	Description
uint16	majorVersion	set to 1
uint16	minorVersion	set to 1
uint32	featureVariationRecordCount	Number of feature variation records.
FeatureVariationRecord	featureVariationRecords[featureVariationRecordCount]	Array of feature variation records.
uint32	lookupVariationRecordCount	Number of lookup variation records. Added in version 1.1.
LookupVariationRecord	lookupVariationRecords[lookupVariationRecordCount]	Array of lookup variation records (sorted). Added in version 1.1.

NOTE: If minorVersion of the FeatureVariations table is set to 0, then only Condition Table Format 1 can be used. If minorVersion is set to 1, then Condition Table Formats 2, 3, and 4 can also be used.

FeatureVariation Record

The organization of the FeatureVariation record array is as follows: Each record points to a condition set and to a list of alternate Feature tables, each of which is associated with a Feature index. That index is not the offset of the Feature table in its list, it is the index of a Feature table in the main Feature List of the top-level table (GSUB or GPOS) for which this alternate Feature table might be substituted.

The list of FeatureVariation records is searched in order for a condition set that applies to the set of factors matching the current runtime context—currently an instance of a variable font. When the first such condition set is found the search stops, and whatever

alternate Feature tables it points to are substituted for tables in the main Feature List according to the specified Feature indexes.

Note that the records shall be ordered in increasing order of FeatureIndex values, and no two records may have the same FeatureIndex value. This is so that searches for a particular feature index, or for elements of an ordered list of feature indexes, can be optimized, and so that searching can end if a record is encountered with a higher index value.

[The FeatureTableSubstitution and FeatureTableSubstitutionRecord tables are unchanged. The text currently following those tables is omitted.]

Because there is only a single list of condition sets, and only one record containing the first condition set that applies is used, a font with as few as five or six substitutions could need many more condition sets. In such cases the LookupVariation record array is likely to be more efficient in both file size and computation time.

LookupVariation Record

The LookupVariationRecord and its subtables provide an alternative mechanism for changing the lookups associated with a featureIndex. When the index for a feature is present in a LookupVariationRecord, the FeatureLookupsTable at the offset either supplements or replaces the default lookups associated with that index. The subtables of a FeatureLookupsTable are organized around individual lookups rather than whole tables, with individual or groups of lookups activated for the index when given condition sets apply or fail to apply.

LookupVariation record

Type	Name	Description
uint16	featureIndex	The feature table index to match (this is the sort key)
Offset32	featureLookupsTable	Offset to a FeatureLookupsTable

FeatureLookupsTable

Type	Name	Description
uint16	majorVersion	set to 1
uint16	minorVersion	==set to 0 ==
uint16	flags	FeatureLookups qualifiers see below
uint32	lookupConditionCount	Number of LookupCondition records.
LookupCondition-Record	lookupConditionRecord[conditionCount]	Array of LookupCondition records.

The FeatureLookupsTable provides offsets to a list of LookupConditionRecords that affect

the associated featureIndex. Because all LookupConditionRecords are evaluated, they can be in any order.

Flags can be assigned to indicate certain uses or behaviors for a given FeatureLookups table. The following flags are defined.

Mask	Name	Description
0x0001	ADD_DEFAULT_LOOKUPS	When this bit is “on the lookups in the default feature table for the index are added to the lookup set. Otherwise only the lookups specified by the LookupConditionRecords are included in the set.
0xFFFE	Reserved	Reserved for future use set to 0.

LookupCondition record

Type	Name	Description
Offset32	conditionSetOffset	Offset to a condition set table
Offset32	trueLookupIndexListOffset	Offset to a LookupIndexList table to add to the set when all conditions are true (0 if unused)
Offset32	falseLookupIndexListOffset	Offset to a LookupIndexList table to add to the set when at least one condition is false (0 if unused)

LookupIndexList Table

Type	Name	Description
uint16	lookupIndexCount	Number of LookupList indices in this table.
uint16	lookupIndices[lookupIndexCount]	Array of indices into the lookup list.

The LookupConditions record encodes the equivalent to an if/else structure. When when the condition set applies all lookups from the trueLookupIndexList are added to the set of lookups corresponding to the feature index. When it does not apply the lookups from the falseLookupIndexList are added to that set. Either entry (but not both) can be disabled by setting its offset to 0. As with condition sets generally, a 0 offset indicates the set always applies, and therefore the entries from the trueLookupIndexSet will be added.

A typical FeatureVariations Table will contain either FeatureVariationRecords or LookupVariationRecords but not both. The primary reason for having both would be if LookupVariationRecords are used to determine the lookups that are applied for a given feature index but the featureParams (see 6.2.5) of one or more feature tables also need to change in certain regions of the font’s design space. When a given feature index is listed in both a FeatureVariationRecord subtable and a LookupVariationRecord subtable, the featureParams are taken from the former and the set of lookups is taken from the latter.

Because LookupVariationRecords specify the set of lookups for the default instance, it is strongly recommended that that set match the list of lookups in the Feature table of the Feature index it corresponds to.

Explanation of FeatureVariations table processing

Given a sorted array of Feature indexes for selected features obtained from the LangSys table, substitution of alternate feature tables and lookups can be done as follows:

1. Process the featureVariationRecords:
    a. Evaluate the condition set of each FeatureVariationRecord in order until one applies (e.g. every condition in the set evaluates to true.)
    b. If there is such a record, find the intersection of Feature indexes in the array with Feature indexes in the FeatureTableSubstitution records. For each of those Feature indexes, associate that Feature table with the offset of its alternate.
2. Find the intersection of the Feature indexes in the array with the Feature indexes among the LookupVariation records. Then for each such record:
    a. Allocate an empty feature table structure to use for that index.
    b. Copy any FeatureParams from the "current feature table (either the alternate from step 1 if there is one, or the original Feature table for this index).
    c. If ADD_DEFAULT_LOOKUPS is set, copy list of lookups from current feature table into the set for this feature.
    d. For each LookupCondition record:
        i. If all conditions are true set o = trueLookupIndexSetOffset
        ii. Otherwise set o = falseLookupIndexSetOffset
        iii. Copy each lookup in the LookupIndexSet at o into the set for this feature
3. For each Feature index in the array without a corresponding LookupVariationRecord:
    a. Use the current Feature table for that featureIndex (either the alternate from step 1 if there is one, or the original Feature table).