23rd Internationalization and Unicode Conference (IUC23)
Unicode, Internationalization and the Web: The Global Connection
March 24-26, 2003, Prague, Czech Republic
http://www.unicode.org/iuc/iuc23/

漢字變體字: 筆謎和謎底

UniHan Variation: Issues and Solutions

Richard S. COOK
Department of Linguistics
University of California, Berkeley
rscook@socrates.berkeley.edu
http://stedt.berkeley.edu/
2003-02-11-17:30

CONTENTS

INTRODUCTION

This presentation is concerned with discussion of issues relating to solutions to the UniHan (CJKV[1] Unified Ideographs) variant problem. Some of the kinds of variation encountered in the encoded CJKV characterset are examined along with its implications for users and developers, and methods for coping with this variation in software implementations are explored. The following three aspects of the CJKV characterset are addressed:

(一) Variant Typology
(二) CJKV Description Language (CDL)
(三) Variant Selectors

Specifically, we intend to propose the use of a formal CJKV Description Language (CDL) in conjunction with Variant Selectors (VS) for resolution of CJKV variant issues. As a first step, we believe that precise descriptions using a formal CDL should be developed for all encoded CJKV entities, including unifications in which basic script elements (see the definitions below) have been deemed non-distinctive. This talk concludes with demonstration of *Wenlin 3.x* software implementing a stroke-based font for Unicode 4.0 UniHan.

[1] Chinese, Japanese, Korean, Vietnamese.

BACKGROUND

The UCS now encodes a large number of variant CJKV character forms of various types, some of which have relative mappings (i.e. traditional-to-simplified mappings), many of which do not. On the basis of IRG member submissions for CJKV Unified Ideograph Extension C1, a very large number of additional encoded variants seem imminent.

Non-systematic encoding of variant character forms presents a great danger to a character encoding, great difficulties to the maintainer and publisher of the standard, great problems for the implementer, and confusion to the end-user. Where unassimilated, non-distinctive character forms creep into the character set, all encoded variants, rare or not, appear to have an equal footing, and information access and exchange are impeded as a result.

One might wonder, "Why has it been permitted to encode character variants in the UCS in the first place, if this is such a problem?" In fact, every character has a story, and different encoded characters have different stories. Sometimes a given variant has been encoded because of legacy encoding issues (*e.g.* Compatibility characters), sometimes because of character property issues (*e.g.* Radicals), sometimes because of irregularities in the traditional organizational systems (Radical/Stroke indices), sometimes because of limitations of the rules and procedures governing the modern encoding process (*e.g.* ISO/IEC 10646 Annex S).

As the previous examples might suggest, the processes driving the encoding of variants are numerous and varied themselves. These processes point to logical classes of variants, and it is to examination of some of these that we should turn, after briefly defining "variant".

VARIANTS

Most broadly, underlying Unicode's minimalist approach to character encoding is something known as the Character-Glyph Model (CGM). In accordance with the implications of the CGM (see ISO/IEC TR 15285-1998E) one seeks to define the optimum set of script elements necessary for the encoding of a given script, based on identification of distinctive features. As historical linguistics tells us, what is distinctive in one time may not be distinctive in another: fine distinctions develop and blur and disappear over time. And so it is that as the Unicode Standard seeks to address the encoding of ancient texts, it is the very locale- and time-dependence of this key notion "distinctive feature" which presents character encoding with its greatest challenge.

What might "distinctive features" mean for Han? What exactly is a Han variant? Let's look at the second question first. Consider, for a rather vexing example, the ten forms in the following list (excerpted from line 3767 of my SWJZZ variant mapping table):

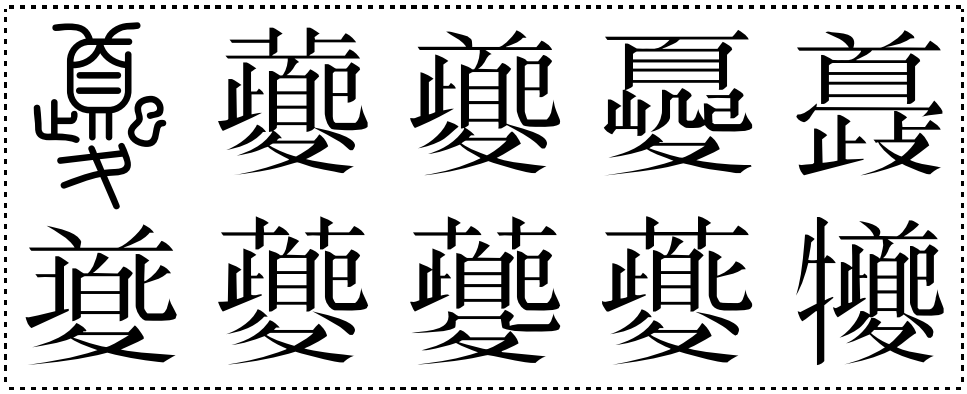


FIGURE 1: *Some Encoded Variants*

Of these, all but the first (a "Small Seal" form in Supplementary Private Use) represent actual encoded characters, as the UCS codepoints listed in FIGURE 2 illustrate:

[U+100eb6]	[U+21582]	[U+5914]	[U+2157f]	[U+2982e]
[U+21578]	[U+270f0]	[U+2f9b2]	[U+270cd]	[U+72aa]

FIGURE 2: *UCS Codepoints for FIGURE 1*

Disregarding certain attributes as non-distinctive (we will discuss what this might mean below), there is a sense in which all of the above forms represent the "same Chinese character", and it is for this reason that FIGURE 1 bears the title (really only an opinion) "Some Encoded Variants". The (121AD) lexicon *Shuowen* says that our Seal form [U+100eb6] is a writing of the name of a malevolent freakish mountain monster, 'like a dragon, but with one foot, horns, hands, and a man's face'. In early inscriptional forms the creature also had claws, a tail, and walked (hopped?) upright; not to be confused with 夔 [U+5912] (猱 [U+7331]). All of the forms in FIGURE 1 stand for (or are derived from, in the case of [U+72aa]) writings of the "mountain monster" syllable, pronounced **kuí** in modern Mandarin.

All forms in FIGURE 1 appear in *Hanyu Da Zidian* (HDZ), a primary lexical source for the IRG's Ext B encoding work, and the nine encoded forms all have (non-PUA) HDZ mappings. Of these nine, all but [U+72aa] (the last listed, with the "Ox" radical on the left) may be regarded as true "graphical variants" of one another (this term is discussed below). In a perfect world, those eight forms now enshrined separately at eight distinct codepoints (only one of which is in the BMP)

	[U+21582]	[U+5914]	[U+2157f]	[U+2982e]
[U+21578]	[U+270f0]	[U+2f9b2]	[U+270cd]	

FIGURE 3: *Eight "retroactive unifications"*

would all have been unified, in strict adherence to the CGM.

The fact that application of the (informative) unification "rules" of ISO/IEC 10646-1 Annex S did *not* result in unification of these eight forms indicates some limitations in those rules, or an over-riding "lexical source separation" rule. In fact, such an over-riding rule is implicit in the Annex S unification rules, which are based on (among other things) such traditional lexical indexing conventions as *stroke types* and *stroke counts*, to the exclusion of "meaning". If one were to insist that the above eight glyphs are all unifiable, one would have to base this opinion on the lexical definitions of each character in a particular lexical source. And different sources might disagree.

VARIANT TYPOLOGY: THE CJKV GRAPHICAL VARIANT

As exclusion of [U+72aa] from the "retroactive unification" list in FIGURE 3 illustrates, variants among the encoded and unencoded CJKV characters may be classed into several types, and not all variant types are relevant to the issues which must be resolved for UCS encoding. In fact, the only kind of variant relevant to UCS encoding is the so-called "CJKV Graphical Variant", and so we should take a moment to try to define this and a few terms relevant to the CKJVUI repertoire.

> • DEFINITION A: ***CJKV Graphical Variants*** (CGV) may here be defined as those entities which are (by consensus or significant opinion) "in some sense the same character" (non-distinctive within a particular usage context), and yet which differ from one another in terms of *basic script elements*.
>
> • DEFINITION B: ***Basic script elements*** such as *stroke type, stroke count, component structure*, or *component position* are those script elements which are usually distinctive in a specific context. In the case of CJKV Graphical Variants, this usual distinctiveness is being over-ridden, such that normally distinctive elements are viewed as non-distinctive for the purpose of building unification/variant classes.
>
> • DEFINITION C: ***Non-basic script elements***, such things as *stroke weight, skew angle, flourish, ink color, resolution*, are those elements which are *never* distinctive, and so can never be used as the basis for disunification.
>
> • DEFINITION D: ***Glyphic Variants*** are therefore distinct from Graphical Variants in that the former involve variation in non-basic script elements.
>
> • DEFINITION E: ***Script entities*** include *basic stroke types* and all complex assemblages of stroke primitives, from dependent *component* up to the level of independent CJKV *character.* [Definitions of stroke primitive, component, and character are given below.]

In terms of the CGM, it appears therefore that the CJKV Graphical Variant is a script entity falling somewhere *between* "character" and "glyph", and this suggests that a CVGM "Character-Variant-Glyph Model" is needed to address CJKV Graphical Variants.

In terms of the unification procedures described in Annex S, the members of a CGV class are sometimes identified and unified, though not always. Disunifications such as those listed in FIGURE 3 result at least in part from the lack of a mechanism for treating the graphic/glyphic difference in some meaningful and consistent way. Similarly, unifications involving glyphs with different stroke counts gloss over distinctions which may be meaningful for certain purposes (*e.g.* for producing a Radical/Stroke index in which real stroke counts for real glyphs must appear).

Within an instance of this "CJKV Graphical Variant" class, one may define a subset including all forms subject to the rules of unification: some forms in this subset may have been unified already, and some may not. One is therefore obliged to clearly state one's unification rules, and make these rules *normative*, if one is to classify forms for the purpose of unification and identifying variants.

FIGURE 4: *Members of the **áo** Graphical Varclass*

In FIGURE 4 above are listed 3 members of the **áo** Graphical Variant Class. This independent character is also a rather productive component in the script, as seen in FIGURE 5 below.

FIGURE 5: *Encoded characters with the **áo** component*

One problem is that in the forms above the solid line in FIGURE 5 the **áo** component sometimes has 10 strokes, and other times has 11 strokes, reflecting IRG unification of the forms given as [U+e109] (PUA) and [U+6556] (BMP) in FIGURE 4 above. That is, variant forms with either 10 or 11 strokes have been unified. To find one of these characters in a Radical/Stroke index, one would not know whether to look under 10 or 11 residual strokes, and the form found in a Unicode R/S index under either of these stroke counts might have one or the other actual stroke counts.

It has sometimes been suggested that one might like to unify, for example, all regular traditional/simplified pairs (that is, cases where "simplified" forms are not "traditional"). The rules for determining regular relations and also for treating exceptions among the distinctive elements of these pairs would need to be formulated. A great deal of consensus on such matters has already been achieved in PRC work. FIGURE 6 below lists some standard PRC Traditional/Simplified pairings. [Roughly, the characters in ROW B are simplified (to A) only as independent characters, never as components. The characters in ROW D are the opposite, simplified (to C) only as components, but never as independent characters, except for the last.]

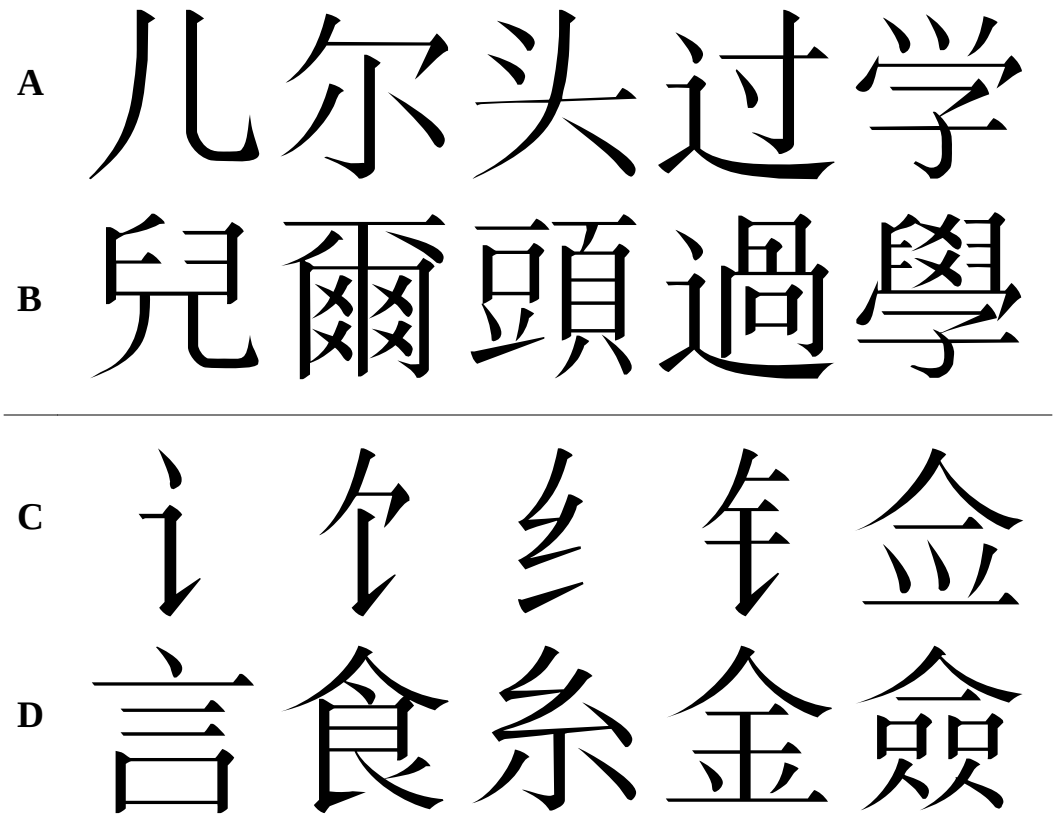
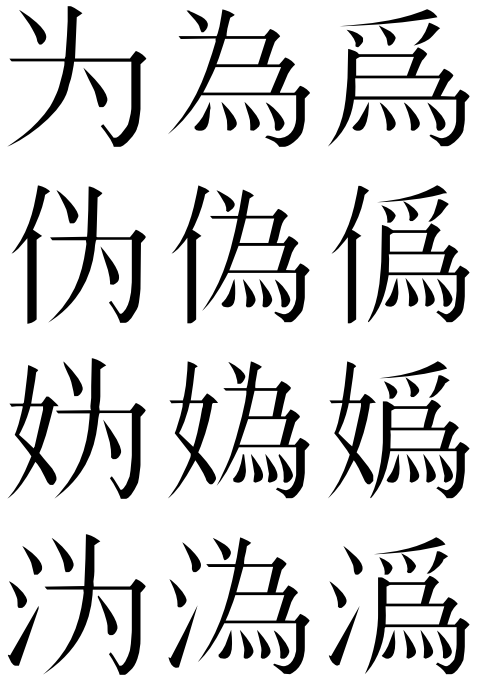


FIGURE 6: *Some Traditional/Simplified pairs*

It is arguable, however, that the bottom S/T pairs in FIGURE 6 are suitable for treatment with Variant Selectors (as dependent Graphical Variants), while most upper pairs are not. But since all characters in FIGURE 6 are encoded, including the Graphical Variants (some in the "Radical" block), any scheme for treating variation in these pairs must be handled with mapping tables (see below).



[U+4e3a] [U+70ba] [U+7232]
[U+4f2a] [U+507d] [U+50de]
[U+59ab] [U+5aaf] [U+5b00]
[U+6ca9] [U+6e88] [U+6f59]

FIGURE 7: *Encoded "Triplets"*

FIGURE 7 above lists 4 sets of encoded "Triplets". The first column from the left is the "simplified" form, the second form is the "full" Big-5 form, the third is the "old" a.k.a. "ultra-fanti" form.

[U+8aaa]
[U+8aac]

FIGURE 8: *Glyphic vs. Graphical Variation*

FIGURE 8 above is intended to illustrate the difference between *Glyphic Variants* and *Graphical Variants*. The top row lists *glyphic variants* of [U+8aaa], while the bottom row lists *glyphic variants* of [U+8aac]. The characters [U+8aaa] and [U+8aac] are however *Graphical Variants*, in that they differ from one another in *basic script elements* (*i.e.*, basic stroke types).

WHY A CJKV DESCRIPTION LANGUAGE (CDL)?

Why do we need to define a formal CJKV Description Language (CDL) and why apply it to the entire CJKV characterset? What would an adequate CDL do for us? What about other systems such as Unicode's IDC/IDS? Isn't there some existing system that can be applied to the problem? Do we need to invent a new system?

Let's look at these questions here, one at a time.

• First, the reasons we need to define a formal CJKV Description Language (CDL) and apply it to the entire CJKV characterset relate to resolution of the Han Variant Problem. In the CGM, one is content to leave the exact form of the character underspecified. In a "CVGM", one must exactly specify the shapes of all encoded entities, since CJKV Graphical Variants are now to be treated as encoded entities using variant selectors.

• An adequate CDL would provide us with reference glyphs for all encoded entities, storing a great deal of important information in the reference glyphs themselves. Things like "lexical radical", "component structure", "stroke count" and "stroke order" are all specified by the reference glyph.

• Incidentally, though quite significantly, an adequate CDL would provide us with the necessary information to automatically generate computer fonts for production and promulgation of the standard.

• Unicode's IDC/IDS (Ideographic Descriptive Characters and Sequences) are acknowledged as only a glance in the direction of a complete CDL. The IDC/IDS were never intended to serve the purpose here envisaged for a complete CDL. Other such systems, for example, the Big5-based system developed in the InfoTech Lab at Academia Sinica (see my encoding proposal "02221-n2480.pdf" which lays out the system of "operators" used in that system), are rather similar, in that they do not define the necessary basic script elements for a complete description of all Han characters.

• Wenlin Software's highly refined C implementation of its "Stroking Instruction Language" is to our knowledge the only existing, fully implemented, completely adequate CJKV Description Language. Using this language, "Stroking Instructions" have been created for some 30,000 CJKV entities (at last count), both in the BMP and in the SIP, both encoded and unencoded (PUA). It is planned at present to elaborate elements of this system such as may prove relevant to the standardization in future encoding proposals. Without going into the exact details of the specific implementation, we may however sketch the basic outlines of the elements of such a system here.

ELEMENTS OF THE CJKV DESCRIPTION LANGUAGE (CDL)

An adequate CJKV Description Language must be able to describe all uniquely CJKV script entities. The elements of an adequate CJKV Description Language are to be specified in terms of a grid of specific size, and in terms of the entities which may populate that grid. These entities, here termed "Basic Stroke Types", are defined as follows.

> • DEFINITION F: ***Basic Stroke Types***, also called *stroke primitives*, are the finite though extensible set of fundamental script entities, each composed of precisely one "stroke" (uninterrupted movement of the brush or pen), as "stroke" is defined in the traditional Song writing style employed in primary modern sources of lexical stroke-based indices. Each Basic Stroke Type is defined in the CDL in terms of a set of *Segment Types*. Each Segment Type is defined in terms of a set of *Points*, and each Point is defined in terms of a set of Cartesian *Coordinates*. Cf. FIGURE 9.

FIGURE 9: *Selected Basic Stroke Types (arranged according to shape)*

FIGURE 9 lists some representative Basic Stroke Types selected from the currently defined set of approximately 50.

Using a set of Basic Stroke Types, complete descriptions of any CJKV script entity may be built. It is anticipated that a complete inventory of basic stroke types for the CJKV elements of the Unicode 4.0 characterset might require a block of about 64 codepoints. Several rare stroke types are known to us at present, though as yet unsupported.

Certain other classes of CJKV entity require that transformations, rather than additional stroke types, be defined in the CDL. The top row of FIGURE 10 below lists some of these; the bottom row gives forms with usual stroke types:

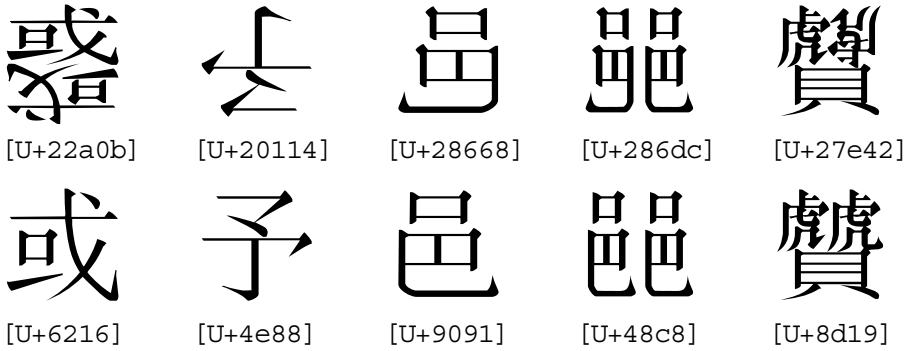


FIGURE 10: *Unusual stroke types in "flipped" components*

The forms given in FIGURE 10 all involve components in which a Basic Stroke Type has been written in some non-standard way, which is to say, the stroke is the inversion or mirror image of a basic stroke type. Other rare stroke types are seen in encoded "cursive" forms: the correct way to treat non-Kaishu forms must be determined. It seems that the best solution involves "kaishu-ification" of such cursive forms. The current reference glyph (on the left in FIGURE 11 below) for [U+201ad] involves elements of this type (kaishu-ified using basic stroke types, on the right).

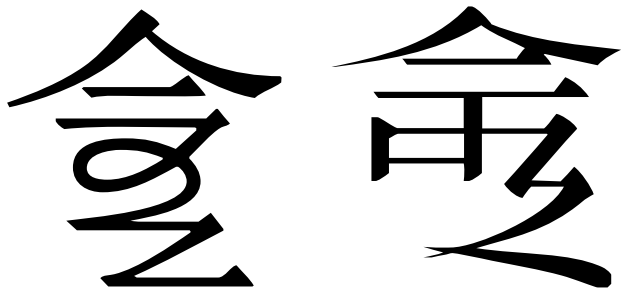


FIGURE 11: *[U+201ad]: Unicode 3.2 reference glyph, and its "kaishu-ification"*

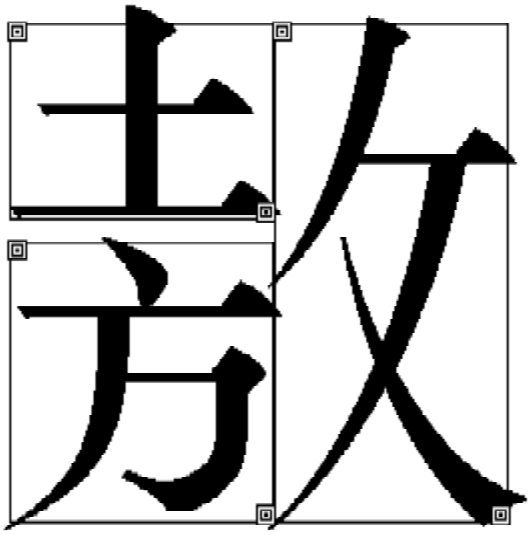


FIGURE 12: *CDL for PUA graph [U+e109]*

The CDL description of `[U+e109]` (PUA graphical variant of `[U+6556]`; cf. FIGURE 4 above) specifies the components `[U+571f]`, `[U+65b9]`, and `[U+6535]`, each with the (x, y) coordinates of its bounding rectangle within the bounding square of the composite character. This description is not self-contained: in order to display the composite character, the language interpreter uses the CDL descriptions of each of the three components. In general, components can be any characters that are themselves defined as sequences of basic strokes and/or simpler components.

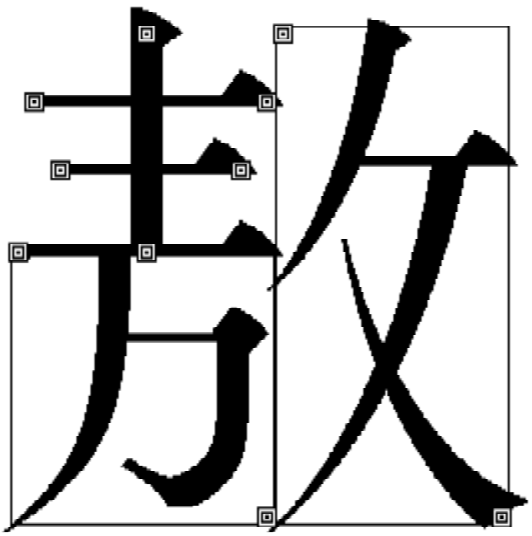


FIGURE 13: *CDL for BMP graph [U+6556]*

The CDL description of `[U+6556]` specifies the first three strokes as basic stroke types, each with the coordinates of its starting and ending points (and possibly stroke modifiers), and then specifies the two components `[U+4e07]` and `[U+6535]` with their bounding rectangles.

VARIANT SELECTORS (VS)

Unification classes (as these have been set up for the CJKVUI so far) are simply special cases of *variant classes* (which remain to be established). With unification classes, the variation among the class members has in some regard been deemed irrelevant to encoding, and in effect discarded from standardization (as abandonment of the multi-columnar format for the printing of Ext. B illustrates). As discussed above, there are indeed some reasons to reconsider the ultimate utility of a unification model which does not distinguish basic and non-basic script elements, and which does not standardize (preserve) such information.

In the model which we would put forth for application of variant selectors to CJKV encoding, variant classes would be comprised of all unified (or unifiable) forms involving basic script elements, both encoded and unencoded. That is, unified entities distinguishable in terms of basic script elements (e.g. stroke count) would in effect be disunified employing the variant selector scheme. Script entities suitable for encoding in the VS scheme would be all basic script entities (as these were defined above), both independent and dependent, including contextually dependent (that is, compositionally deformed) forms. It is argued that the distinction between independent and dependent basic script elements is not relevant to encoding. Rather, "graphical component" is defined as follows:

> • DEFINITION G: ***Graphical Components*** are any productive assemblage of basic stroke types, where productivity is gauged in terms of frequency of occurrence. That is, a component is identified in terms of its appearance (as a recurrent pattern of basic stroke types) in two or more non-unified glyphs. A *Graphical Component* may or may not be encoded, depending upon its degree of productivity (or importance, according to some authoritative source). Highly productive or highly salient components should most definitely be encoded immediately. Less productive components may be encoded on an as-needed basis.

In application of this Variant Selector scheme to CJKV script entities, use of VS is completely optional, and so we foresee two kinds of text: text *not* employing variant selectors ("text"); text employing variant selectors ("VS text"). In VS text every basic script entity would be identified in the form

`[U+X]([U+V])? => <CDL>`

FIGURE 14: *Variant Selector scheme for CJKV script entities*

where "[U+X]" represents a CJKV character codepoint, ([U+V])? indicates the optional Variant Selector, <CDL> indicates that there is a precise description (stroking instructions) of the entity using the CDL, and "=>" indicates the mapping relation between the variant selector/codepoint pair and the <CDL>. Though the VS is optional, any block of VS text will necessarily include at least one VS. In this scheme, all script entities in VS text are treated as variants. In cases of VS text in which the VS is not used, that is, in cases in which a variant form has not been explicitly chosen, the system would be expected to fall back to some locale-specific default among the variants associated with a particular Unicode Scalar Value.

Why use Variant Selectors at all, rather than simply continue encoding variants? Why not simply develop and maintain variant mappings as needed in Unihan.txt? There are several reasons why VS might be viewed as preferable to simple mapping tables, though we do not believe that VS and

mapping tables are mutually exclusive. On the contrary, VS *and* variant mappings will both be necessary to handle encoded and unencoded variants. A VS scheme simplifies searching, since search algorithms do not fail in a VS scheme with addition of variants: the VS provides an extensible means of dealing with variants, whereas the non-VS mapping table approach requires mapping table maintenance without an adequate organizational mechanism. Most importantly, it is the tying together of VS with CDL which provides the most important long-term benefits. Variant tables would also use the VS scheme:

```
([U+X]([U+V]) => <CDL>) => ([U+Y]([U+V]) => <CDL>)
```

FIGURE 15: *Variant Selectors in Variant Mapping*

SUMMARY

In summary, it may be said that we intend to propose the use of a formal CJKV Description Language (CDL) in conjunction with Variant Selectors (VS) for resolution of CJKV variant issues. Ultimate resolution of variant issues will require that the unification procedures outlined in Annex S be refined and made normative. As a first step, we believe that precise descriptions using a formal CDL should be developed for all encoded CJKV entities, including unifications in which basic script elements (see the definitions above) have been deemed non-distinctive. Such unifications would then be the initial test cases for application of VS to CJKV.

CONCLUSION

As might be suspected from this brief discussion, the "Han variant problem" is deep and difficult, and will not be resolved quickly. On the contrary, seeking to solve the problem without first working out a careful methodology built on well-defined principles must only compound future difficulties. Since the characters derive from multiple lexical sources, and since these sources are not always in agreement, the UCS is now a new super-lexical source. One may view the encoded characterset in its present state as the first adequate statement of the problem. One is obligated now to take this mass of partially assimilated or completely unassimilated data and evaluate it scientifically, for the purpose of refining the method to be used in future encoding work.

ACKNOWLEDGMENTS

This paper paper was composed in association with Thomas BISHOP <wenlin@wenlin.com>, inventor of the CDL discussed here. Thanks to Tom for prescient thinking, indefatigable programming, numerous suggestions, and for providing text for the discussion of FIGURES 12 and 13. All figures containing CJKV strokes and graphs were produced using Wenlin software; descriptions of Wenlin's CDL are used here by permission. Detailed examples of Wenlin's CDL notation are excluded from this paper, pending publication. For more information on Wenlin software, please visit <http://www.wenlin.com/>.

The writing of this paper was supported in part by STEDT grants from:

- The National Science Foundation (NSF), Division of Behavioral & Cognitive Sciences, Linguistics, Grant No. BCS-9904950;
- The National Endowment for the Humanities (NEH), Preservation and Access, Grant No. PA-23353-99.