L2/01-175

Proposal to Extend the Scope of Variations Allowed Under UTS 37, or The Y-Variant Problem and You

John H. Jenkins

7 May 2010

Background

Unicode encodes characters, not glyphs. That is one of the fundamental principals underlying all the work we do. Granted, the line can sometimes be difficult to draw, and granted, we have splitters and we have lumpers, but on the whole this is what we set out to do and what we usually accomplish. This approach separates the content of the text from its representation and keeps text-based processes such as spell-checking and searching free from the tyranny of having to kowtow to every whim of glyph-based processes such as layout.

And yet there is one exception to this rule—and a rather glaring exception, at that, inasmuch as it involves the script which takes up more of Unicode's encoding space than all the others put together. Where Han is concerned, we're neither fish nor fowl, neither a glyph registry nor an actual character encoding standard. Rather, we're something in the middle.

Han is the rare example of a script where we've formally defined the method by which entities are encoded, which consists of three main parts.

First, we have a three-dimensional model. The x-axis represents semantics. Two entities which mean different things occupy separate positions on the x-axis. The y-axis represents abstract shape. Two entities which have the same meaning but different abstract shape are y-variants. Finally, the z-axis represents specific glyphic shape. Two entities with the same abstract shape but distinct glyphs are z-variants.

As examples, 井and 鄭 have different meanings (the former means *well* and the latter is a toponym and surname) and are non-variants. 貓 and 猫 have the same meaning (*cat*) but different abstract shapes (they have different radicals) and so are y-variants. 說 and 説 have the same meaning (*to speak*) and same abstract shape and are z-variants.

Secondly, we have a defined algorithm for analyzing entity pairs to determine whether or not they have the same abstract shape. We split them into pieces and analyze the pieces until either we have atomic pieces or pieces with different abstract shapes.

So 貓 and 猫 are split into 豸 and 苗, and 犭 and 苗, respectively. 苗 is common to both, but 豸 is KangXi radical 153 (badger) and 犭 is a form of KangXi radical 94 (dog) and are considered to have different abstract shapes. Therefore, 貓 and 猫 have different abstract shapes, Q.E.D.

Thirdly, we have a document which brings this together, provides specific instances of "abstract shapes that look different but aren't" (such as 兌 and 兑), exceptions, and so on. This document is 10646's Annex S.

In theory, what we encode in Unihan are entities which are either non-variants (differ on the x-axis) or y-variants (different abstract shape). One's expectation would be to encode only non-variants, but there are a couple of wrinkles.

The first wrinkle is that defining when two entities occupy different positions along the x-axis. In most dictionaries, if you look up 并, you'll see an entry that says something like, "same as 并." That would make 并 and 并 y-variants. But other dictionaries point out that in addition to being a synonym for 并, 并 can mean "bowl of food," which means they have to be distinguished from each other.

Even when two entities have complete semantic overlap, there can be other reasons for keeping them separate in encoding. The historical reason for this in Unicode is maintaining compatibility with earlier character sets, where inconsistent or just plain different criteria were used for determining when to encode and when not to encode. This long-standing practice also means that users have come to expect certain glyphic distinctions to be made, even when there is no semantic reason to do so. This we had a recent complaint that the "What is Unicode?" page in traditional Chinese used three "wrong" glyphs, even though the concession was made that the meaning was unaffected. (There is actually a good reason for this, which I'll get to in a bit.)

Of course, the most persistent case of encoding y-variant pairs comes from simplified Chinese. Not only do we have existing practice to deal with, the fact of the matter is that mappings between simplified and traditional Chinese are on occasion not one-to-one. The exceptions are, however, very few. (The Unihan database has 82 characters with multiple traditional variants and 11 with multiple simplified variants.) Moreover, the overwhelming majority of simplifications are made algorithmically, by substituting a simplified component for a non-simplified one, as when 說 is simplified as 说.

This algorithmic generation of simplified forms means that people can make up new ones on-the-fly without having to consult the lists provided by the PRC government. Anybody familiar with modern Chinese can see the character 緁 and know instantly that its simplified form is 缃, even though 缃 isn't formally listed. So long as 缃 remains a mere platonic ideal with no physical manifestation, it can be ignored; but once someone with a font editor and a moderate amount of ambition decides to use 缃 in an actual publication, it becomes attested and something to encode.

The Unihan Database

The Unihan database inherits a fair amount of variant information from Xerox, RLG, and Apple. We've supplemented this in three ways.

First, the z-variant data was algorithmically extended. Unfortunately, the fact that the algorithm was flawed wasn't discovered until it was too late and nobody has taken the time to clean up the data. The result is that the z-variant data has too many false positives and isn't very reliable.

Secondly, Wenlin provided extensive simplified/traditional variant mapping data.

Third, we took advantage of the fact that many dictionaries list characters they consider variants of one another in the same entry. We were able to use this to significantly extend our y-variant data.

Having said this, the fact of the matter is that the bulk of the variant data is for the original URO. Extensions A, B, and C are at least partly covered in our variant data, but by no means systematically. This is particularly true for Extension B—getting complete variant data for Extension B basically means someone has to read the entire Hanyu Da Zidian. (We won't even mention the quality control problems in Extension B.)

Impact

If I'm dealing with Latin text, because the distinction is strictly maintained between character and glyph and because such variations as exist are well-known and well-defined, I can search for "CAT" and find "cat," or I can search for "deja vu" and find "déjà vu." Collation can similarly be fine-tuned by the user. TTS can manage with data about fewer spellings.

This is not true for Han. Because there is no publicly available, reliable, and extensive body of variant data, one does not expect to be able to search for 無為 and find 無爲 (let alone 无为). My own name, unremarkable though it is, can be spelled in four different ways (井作恆, 井作恒, 井作恆, and 井作恒), three of which I've actually seen used. I prefer the first one, myself.

This is why we have to take seriously complaints that we use 為 instead of 爲 on our traditional Chinese "What is Unicode?" page. There is no widespread support for having the computer equate the two. (Granted, that wasn't the cause of the original complaint. The original complaint was that 爲 "looks wrong" to most people and isn't in Big Five.)

This also means that if we want to support attested y-variants, we have to go for separate encoding. Once 𥳇 is attested, if we want to support it in a standardized way, we have no choice but to push for separate encoding, and that means going through the IRG, both slowing down the ability to support it and adding yet another straw to the back of the IRG's camel.

Variation Seqsuences

We have a process defined, of course, to simplify treatment of variants, with all its attendant blessings. We can register them as variation sequences using the process outlined in UTS 37. This simplifies text processing, and speeds up standardized representation by sidestepping the IRG. Adobe has defined an extension to the TrueType specification to support variation sequences, and this extension is already widely used.

The problem is that right now, the formal limitation on registering variants via variation sequences under UTS 37 is that the variants have to be unifiable under Annex S rules, that is, they have to be z-variants.

The advantage of this is that it allows some control over the registration process. We don't have to worry about some ambitious soul registering 鵪 as a variant of 嘽 just because they want to.

The disadvantage of this is that if we want to be able to represent the newly-attested 绁, we have no choice but to do it via separate encoding.

Proposal

I would like to alter UTS 37 to allow representation of y-variants via variation sequences. That's the bottom line.

Granted, the variation problem in Unihan is pretty nasty and still needs to be systematically addressed, but I don't see any good reason to continue to make things worse especially when we have a good mechanism already defined to deal with currently unencoded variants.

I'm really not sure what the value is of encoding 绁 and characters like it separately. Granted, there are circumstances where it is necessary to distinguish it from 紲. Most of the time, however, it waddles like a 紲, it quacks like a 紲, and it looks like a 紲.

(For purposes of discussion, we can refer to 绁 as "Bob" and 紲 as "Rob." This should simplify things for the UTC. The formal alternative would be "simplified chēn" and "traditional chēn.")

Even within the PRC or Singapore where simplified Chinese is used, a form like 紲 is more likely to show up than 绁. Remember, the formal lists of simplified characters are relatively short. The Unihan database has only 3577 characters with a defined traditional variant. There are likely tens of thousands of Chinese characters which might acquire a new simplified form at any minute. The 言 radical alone has about 1100 and the 糸 radical has nearly 1300. Uncommon characters like 紲 are likely to remain unsimplified simply because few publishers have sufficient ambition to go to the trouble

to create the simplified form—yet so many characters are involved that we get a slow but steady trickle of newly-attested simplifications.

Granted, UTS 37 would have to be modified to support a formal definition of “y-variant.” The best way would probably be to do it via revising Annex S, but I’d rather not go that route. Not only would it take forever to revise Annex S, I think there would be enough FUD that we wouldn’t get the IRG to support it.

Not revising Annex S means that we might have to deal with the possibility of multiple representations of the same form—a problem with which we are all too familiar. Very likely, however, once IRG members see (repeatedly) using IVSs as a viable option to represent the y-variants they want to be able to represent, the FUD is likely to clear from their eyes and a revision to Annex S is likely to become possible.

And honestly, we already have the “multiple representation” problem, even if we never want to search for 綝 and find 彬. It is entirely possible for someone to register a variation sequence, only to find later on that a dictionary treats the two forms as distinct, at which point the non-cognate rule of Annex S kicks in and the two have to be treated as x-variants and separately encoded.

Formally, we could adopt a definition of y-variant along the lines of: “Two characters are y-variants of one another if either one can be derived from the other by application of standard simplification rules, or if the source dictionary for one defines it as a complete synonym of the other.” This would mean that people registering new IVSs should be prepared to document y-variant-ness if questioned. If nothing else, it means that we can stop encoding newly-attested simplified Chinese forms.