Issues in Indic Language Collation

Cathy Wissink
Program Manager, Windows Globalization
Microsoft Corporation

I. Introduction

As the software market for India[i] grows, so does the interest in developing products for this market, and Unicode is part of many vendors' solutions. Microsoft is one of those vendors committed to using Unicode to support the Indic-language and, more generally, the worldwide software market. In our implementation experience, we have found many benefits to creating a single worldwide version of product, and Unicode is the reason many of those benefits are available[ii].

However, there are software vendors who see a barrier to implementing Unicode in products for the Indic-language market. This barrier is the perception that deficiencies in Unicode will keep software developers from creating products that are culturally and linguistically appropriate for the Indian market. This perception manifests itself in a number of ways, but one concern that the Indic language community has voiced is the belief that the Unicode character encoding order is not appropriate for linguistic collation (or sorting). This idea that character encoding order in Unicode must be equivalent to linguistic collation of these same scripts and their respective languages is considered by some developers a blocking point to adoption of Unicode in the Indian market, and is indicative of the concern within the Indic-language community about the feasibility of Unicode for their scripts.

This paper will demonstrate that this barrier to Unicode adoption does not exist and that it is possible to provide properly globalized software for the Indic market with the current implementation of Unicode, using the example of Indic language collation. A brief history of Indic encodings will be given to set the stage for the current mentality regarding Unicode in the Indian market. The basics of linguistic collation and its application to Indic scripts will then be discussed, compared to encoding, and demonstrated as it exists on Windows XP. The other technologies involved to enable properly globalized software will also be briefly discussed as they pertain to the collation example. In conclusion, the paper will show that Unicode *as an encoding* is more than sufficient to support Indic scripts and languages, since it is only one step of many to develop culturally and linguistically appropriate software for India; software vendors must complete the globalization work needed to support Indic scripts and languages. This is the case already with some software vendors, including Microsoft—Unicode is used in tandem with other technologies (e.g., rendering, input, font support and national language functions) to create a product that is linguistically and culturally appropriate for the Indic-language software market.

II. The Indic script development community and Unicode

As developers of Indic-language software begin to consider Unicode as an encoding option for their software, they see a need to refine the repertoire to best represent the scripts. Over the past year, it has become increasingly obvious that some Indic-script software developers are not fully satisfied with the encoding solution that Unicode provides for Indic scripts. One of the concerns in this developer community has been that the Unicode script repertoires for Indic languages are too Devanagari based, having initially been defined from ISCII 1988; developers for non-Devanagari languages have felt that ISCII (and respectively Unicode) do not satisfactorily support their languages. As such, changes to the Indic character repertoire to better represent non-Devanagari languages have been proposed to the Unicode Technical Committee (UTC), including changes to the Tamil block description (which is now being updated for Unicode 4.0, tentatively scheduled for release in later 2003). Like other script repertoires in Unicode, it has taken some time to refine the set of characters, character properties and block descriptions to the full satisfaction of the linguistic community, and this could continue for some time[iii].

However, beyond the updates in character semantics and repertoires, a stumbling block to Unicode acceptance with the Indian development community remains: the perception that character encoding order should be equivalent to linguistic collation, and that incorrect ordering of code points in Unicode will result in incorrect collation. (This belief is reflected in changes implemented in ISCII 1991; some of the changes from ISCII 1988 involved rearranging code points within the encoding, resulting in a more linguistically-correct order.) In the last year, there have been several proposals brought before the UTC involving rearranging code points in the Indic repertoires.

Rearranging characters however runs counter to Unicode Character Encoding Stability Policy #1: once a character is encoded, it will not be moved or removed[iv] and any such proposal to rearrange characters is rejected by the UTC. As a result, the perception persists in some of the Indic language communities that since code points for a particular language are out of order within the Unicode repertoire and will not be rearranged, correct linguistic collation (and by extension, properly globalized software) is not possible using Unicode.

As will however be discussed in this paper, encoding order cannot be considered satisfactory collation for just about any language, and the Indic languages are no exception to this rule. The two major reasons for this are:

1. Character encodings are generally script- (or subscript) based[v]; collation must be applied at the language (or language variant) level. This means that any chosen encoding order could be incorrect for a number of languages supported by the script, despite being correct for other languages;
2. Encodings do not always take language-specific sorting elements into account (again, in part due to the fact encodings are script based), and language-based sorting elements (which often are multiple code points) are needed for correct collation.

(A third and less important reason is that there is a long-standing precedent for "disorder" in encodings, if extant code pages or character sets are any indication of implementer expectation concerning order of code points[vi]. It is common knowledge in most user communities that some function outside of the character encoding will be needed to perform linguistic collation; there is no expectation that code point order within an encoding will be sufficient. Many scripts in Unicode already fall into this category.)

To better understand these two major reasons why encoding order cannot serve as a collation order for most languages, including Indic languages, a discussion on the basics of collation follows.

III. Collation: Concepts and Application to Indic Languages.

What is collation? For the purposes of this paper, collation is defined as the culturally[vii] expected ordering of linguistic characters in a particular language[viii]. This culturally expected ordering allows users to define, structure and find data in a way that is consistent for their particular language. For example, a filing system that uses an alphabetical ordering for English will start with A, and continue on to Z in a manner expected by English users. Users of the filing system will expect to find C before D, and R before S; if the filing system uses correct collation, users will be able to file and find data easily. Unfortunately, most languages use an ordering system that is significantly more complicated than this English example. Even within the Latin script, there are considerable differences in how A-Z (and additional characters) sort by language. Some examples include:

- In Danish and Norwegian, Ä and Ö follow the letter Z;
- In Finnish and Swedish, V = W;
- In Polish, Z < Ź < Ż;
- Turkish sorts Ş after S.

In addition, many languages support multiple sorts. For example, German sorting can vary depending on how the umlauted characters are treated (either as variants of their non-umlauted forms, or as expansions); Hungarian supports multiple collations as well. As is seen by this Latin script example, the variance in a single script and even a single language prohibit defining an all-encompassing collation for a script that is acceptable for all languages, and this applies to most scripts.

Collation as a rule is based on language-specific sorting elements. These sorting elements, for the purpose of this paper, refer to discrete elements in a language that carry a primary weight in sorting. Users consider a sorting element a "character" in their language, which impacts collation. Users also expect "groupings" of strings to be collected based on these primary sorting weights. From the previous sorting examples given, some sorting elements include Ä and Ö in Norwegian, and Z, Ź and Ż in Polish. What impacts sorting in one language may not impact sorting in another language in the

same script; in addition, it is very often the case that identical sorting elements used in different languages sort differently. For example, Ö is a sorting element in Turkish, Swedish, and Danish, among other languages. It sorts however in very different ways depending on the language:

Turkish: N < O < **Ö** < P < Q
Swedish: Y < Z < Å < Ä < **Ö**
Danish: X < Y < Z < Ä < **Ö**

In addition, these "characters" can be represented by multiple code points in Unicode. For example, Ö is encoded at U+00D6 as a precomposed form, but can also be created through the composition of U+004F and U+0308; both forms need to sort in a correct manner for the appropriate languages. This is not unusual behavior; many primary (and even secondary) sorting elements of a language can consist of multiple code points[ix]. Because linguistic collation is primarily based on sorting elements in a language, and because a single code point cannot always represent these, it is not possible to create a culturally correct sort based on pure code points (that is, individual characters within an encoding) for many languages.

How do these two concepts discussed above apply to Indic scripts and languages?

The first concept, namely that of a single order not being sufficient for a single script, applies to at least some of the Indic scripts, notably Devanagari. For example: while the research is not complete for Devanagari-script languages other than Hindi, it is apparent that at least Marathi can have a different order than Hindi (Sanskrit and Konkani could possibly differ than Hindi as well). This is seen in the below sample comparing the latter section of consonant ordering within both Hindi and Marathi: Lla (U+0933) sorts between La (U+0932) and Llla (U+0934) in Hindi, but comes after Ha (U+0939) in Marathi. In addition, two different combinations of code points (Ksha and Jnya) are considered conjuncts in Hindi, but are unique characters (graphemes) in Marathi.

Table 1: Some differences in sorting order between two Devanagari script languages: Hindi and Marathi.

Hindi:		**Marathi:**	
ल Devanagari La	U+0932	ल Devanagari La	U+0932
ळ Devanagari Lla	***U+0933***	व Devanagari Va	U+0935
ऴ Devanagari Llla	U+0934	श Devanagari Sha	U+0936
व Devanagari Va	U+0935	ष Devanagari Ssa	U+0937
श Devanagari Sha	U+0936	स Devanagari Sa	U+0938
ष Devanagari Ssa	U+0937	ह Devanagari Ha	U+0939
स Devanagari Sa	U+0938	***ळ Devanagari Lla***	***U+0933***
ह Devanagari Ha	U+0939	***क्ष Devanagari Ksha****	***U+0915, U+094d, U+0937***
		ज्ञ Devanagari Jnya**	***U+091c, U+094d, U+091e***

**considered a conjunct in Hindi, but the 35th consonant in Marathi*

***considered a conjunct in Hindi, but the 36th consonant in Marathi*

This particular example highlights why a single collation will not work for the Devanagari script; different languages that use the Devanagari script have different expected collation results. Developers for the Indic market (or any language market) should consider it best practices to leverage extant (or develop new) collation technology, rather than depending on character encoding order to get correct sorting results for different languages. Software vendors developing linguistic collation functions conduct research[x] to determine the correct "character" order (where a character actually corresponds to a single code point) for each language within a script and write this into the collation function; these functions should be called for collation, rather than placing any expectation on the encoding that it should be in perfect sorting order.

In comparison to Devanagari, which clearly cannot use a single code point order due to different language collations within the script, there are other Indic scripts which support just a single language (e.g., Gurmukhi, used for the Punjabi language[xi]). Many implementers wonder: can these scripts support linguistic collation for their respective languages using only code point order (provided of course the code points are in the correct order)?

The answer to this question is no; the second concept of collation (primary sorting elements often require multiple code points) applies to monolingual scripts as well as to multiple-language scripts like Devanagari[xii]. In researching collation for Indic languages[xiii], it became apparent very quickly that properly sorting "characters" in many of the Indic languages, including those with a single language per script, often requires treating multiple (two or three) code points as a single sorting element.

For example, in Hindi, consonants with modifier marks, that is the consonants modified by candrabindu (U+0901), anusvara (U+0902) or visarga (U+0903) sort as unique characters[xiv] before the unmarked consonant. In other words, the sorting order for Hindi consonants follows this pattern (using Ka as an example):

कँ (Devanagari Ka + candrabindu)

कं (Devanagari Ka + anusvara)

कः (Devanagari Ka + visarga)

क (Devanagari Ka)

The three variants of Ka with the modifier marks are considered equal from a primary weight perspective (they differ on a secondary weight level), however, all three variants with modifier marks have a lighter primary weight than the version of Ka without a modifier mark. A consonant and one of these modifier marks has a lighter primary sorting weight than one of the same consonants without a modifier mark.

In addition, the nukta in Hindi (U+093c) modifies a consonant in sorting such that this combination has a combined primary weight equivalent to an unmodified consonant, but with an additional tertiary weight. That is:

क (Devanagari Ka)

क़ (Devanagari Ka + nukta)[xv]

This phenomenon is not limited to Hindi. Tamil has an analogous structure in sorting with the virama (U+0bcd), such that a consonant + virama (halant) combination carries a primary weight that is lighter than the consonant by itself (in other words, a consonant + virama combination is a unique sorting element that comes before the consonant without a virama; a consonant + virama combination has a lighter primary weight than a consonant by itself):

க் (Tamil Ka + virama)

க (Tamil Ka)

ங் (Tamil Nga + virama)

ங (Tamil Nga)

ச் (Tamil Ca + virama)

ச (Tamil Ca)

ஞ் (Tamil Nya + virama)

ஞ (Tamil Nya)

Like Hindi, it is often the case in Tamil that multiple code points combine to create a single sorting element[xvi]. This is the situation in other Indic languages as well[xvii], and because of this, using a single code point order within an encoding as linguistic collation is in no way sufficient, even for those scripts which only represent one language. Again, developers should consider using code point order for collation to be against best practices, and they should either use or develop functions that provide linguistic collation. It is important for the development community to consider character encoding order just a characteristic of the encoding, and to not place the burden of linguistic collation on the encoding.

IV. Unicode: only one part of the globalization solution for Indic scripts and languages

As has been demonstrated in this paper, collation for any language has certain structural needs that cannot be met by an encoding, and Unicode as an encoding is no exception. However, it should also be clear that character encodings were not developed to work as acceptable sorting orders; encodings are for the sole purpose of providing a relationship between a linguistic character and a number.

So if developers cannot count on encoding order to be a viable sorting order for Indic scripts and languages, how can they get culturally correct results? In addition, what else needs to be considered to properly globalize software for India (or any part of the world) if Unicode is only part of a globalization solution?

There are commercial products available today built on Unicode that have implemented culturally correct sorting for Indic languages. For example, Windows 2000 shipped with full language support for Hindi and Tamil, including linguistic collation as described in the previous section. (Windows XP will ship with additional support for other Indic languages, including collation for Telugu, Kannada, Punjabi and Gujarati). As the development team did not expect that encoding order would be sufficient for the Indic languages (since it was not the case with all other linguistic collation support on Windows), it was apparent that collation would need to be researched to add this information to the data tables which support such functions as LCMapString (for sortkey generation) and CompareString (for sorting)[xviii]. For this reason, part of the Windows International development team has been tasked with researching and implementing linguistic collation for all locales (cultural/regional combinations) supported in product. Unicode is considered just an encoding, not saddled with any expectations for collation; collation is to be handled elsewhere (i.e., the above-named functions)[xix]. As a result, it is possible to get linguistic collation (and other =functionality) for supported Indic languages on any version of Windows 2000 or Windows XP, including the English version[xx]; Unicode was the underlying encoding used in both products, but collation was built in elsewhere.

What else needs to be considered for full Indic support in software, if one implements Unicode? As has been discussed in this paper, a character encoding (specifically

Unicode) cannot carry the burden of collation; it also cannot be responsible for all glyph representations of a character (as has also been proposed by certain implementers in the Indian development community)[xxi]. In order to correctly support proper character display and layout of a language, input methods, font support and rendering engines must be properly leveraged or implemented[xxii]. In addition, full NLS (National Language Support) data for formatting of time, dates, numbers, currencies and other locale elements needs to be considered.[xxiii] Again, it should be emphasized that if Unicode is treated *as an encoding*, without the burden of being a catchall for globalization (collation, glyph representation, input, etc.), it is the best encoding solution for software that runs worldwide. After having developed software for many international markets with Unicode, Microsoft as a vendor is even more convinced of the benefits of Unicode for both individual and worldwide markets, and is committed to continuing global development with Unicode into the future.

Software vendors are still in the early stages of developing software that is fully satisfactory for the Indian market on all levels of globalization. It will likely take a few more iterations of the products and on-going refinement of the implementations before the Indic language development community is completely happy with the available products; vendors are working with this community to ensure that the community's and end-users' feedback is integrated into their products. However, as the Windows development team has found over the last ten years, implementing Unicode makes development for the worldwide market (including the Indic languages) not more complicated, but considerably easier; Unicode is the future of worldwide software, including that software for the Indic language market.

---

[i] Note that this market does stretch far beyond the borders of India, and as such, any references to the "Indian market" actually imply "Indic-script/language market".

[ii] For more comprehensive information on these benefits as they apply to Windows, please see the paper Unicode and Windows XP, published in the Unicode proceedings (Washington DC, January 2002, and Dublin, May 2002).

[iii] For more information on the issues surrounding Unicode and Indic script encodings, please refer to Michael Kaplan's paper within these proceedings: Unicode and Indic Scripts: How to Hide a Good Implementation.

[iv] http://www.unicode.org/unicode/standard/policies.html

[v] The Unicode Standard defines a script as: "A collection of symbols used to represent textual information in one or more writing systems." (http://www.unicode.org/glossary/)

[vi] A clear example of this is in the Windows code pages, as seen at http://www.microsoft.com/globaldev/reference/WinCP.asp. The ordering of 1252 for example is not acceptable for any language or culture. In addition, a speaker of any of the Latin-script languages covered by Unicode would not be pleased by the default code point ordering within the standard, since the Latin character repertoire stretches far across the Basic Multilingual Plane, completely out of linguistic order for Latin languages.

[vii] Some people argue that collation can be based on individual expectations (an individual might have different expectations than someone else who speaks the same language), or even that of expected results within an application. Regardless of the type of expectation, the constants here

are that the results should be consistent over time, generally not random (at least from a human language perspective), and usable.

[viii] The Unicode Standard defines collation as: "The process of ordering units of textual information. Collation is usually specific to a particular language. Also known as alphabetizing or alphabetic sorting…" (http://www.unicode.org/glossary/)

[ix] Other examples here include LJ and NJ compressions for Croatian, the NG compression for Vietnamese, and combinations of base character and combining mark in many languages (like the Ö above).

[x] This research involves linguistic analysis of ordered data in the language (indices, lists, dictionaries) as well as input from language experts and native speakers.

[xi] Singh Gill, Harjeet, 1996. "The Gurmukhi Script." In *The World's Writing Systems*, ed. Peter T. Daniels and William Bright, pp. 395-398. New York: Oxford University Press.

[xii] You can see this concept in practice with the Ksha and Jnya consonant examples for Marathi listed above.

[xiii] The languages we have worked on include Hindi, Marathi, Tamil, Punjabi, Gujarati, Kannada, Telugu, with more Indic scripts and languages planned for future versions. The research on Indic collation is challenging (compared to many other scripts Windows supports) and as such sorting continues to be refined, but we are committed to the on-going research and feedback this work entails.

[xiv] Informants referred to these as "letters", "consonants" or "graphemes" in Hindi.

[xv] Note that U+0915 and U+093c should sort equivalently to U+0958, which is क़ (Devanagari Qa = Ka + nukta). This is the precomposed vs. composite version of Ka + nukta (analogous to the Ö example listed previously), and shows yet again how code point order cannot work for sorting, since in this case, two code points should sort equivalently to a single, unrelated code point.

[xvi] Tamil has two primary sorting elements that are created from *three* code points, namely the Ksha (U+0b95, U+0bcd, U+0bb7) and the Shri (U+0bb7, U+0bcd, U+0bb0).

[xvii] Initial research shows that this type of structure will also be needed for other Southeast Asian languages (e.g., Khmer, Dzhongkha), perhaps with a four- or five-to-one (!) code point to sorting element relationship.

[xviii] An interesting side note regarding the development of Indic sorting: the structures for collation in Windows NT were developed in 1991; when we started work on Hindi and Tamil *6 years later*, we were able to work with the extant collation architecture without any changes to code. By virtue of working with Unicode, which is—for lack of a better term—character based, we were able to add Indic languages to only the data tables and CompareString worked correctly.

[xix] Of course, the Unicode Collation Algorithm is part of Unicode; however, that is an algorithm independent of the specific placement of characters within the standard.

[xx] This document was originally written using Windows XP and Office 2000 (English versions of both), with full Indic support enabled.

[xxi] Michael Kaplan will cover this topic in his Indic encoding paper (previously referenced).

[xxii] For an explanation of the relationship between Indic writing systems, rendering and fonts, please see Mudur, S.P., et al. 1999. "Computer Graphics in India: An architecture for the shaping of Indic texts." In *Computers & Graphics* 23: 7-24.

[xxiii] For more comprehensive information on globalization on Windows, please see the paper Unicode and Windows XP, published in the Unicode proceedings (Washington, DC, January 2002, and Dublin, May 2002), as well as Kevin Gjerstad's paper on the Windows Text Services Framework (available in the Unicode proceedings, San Jose 2001)