Journal of Zhejiang University SCIENCE
ISSN 1009-3095
http://www.zju.edu.cn/jzus
E-mail: jzus@zju.edu.cn

A simple approach for building transliteration editors for Indian languages

PRAHALLAD Lavanya, PRAHALLAD Kishore, GANAPATHIRAJU Madhavi
(*Institute for Software International, Carnegie Mellon University, Pittsburgh, PA 15217, USA*)
E-mail: lavanyap@cmu.edu; skishore@cs.cmu.edu; madhavi@cs.cmu.edu

Received Aug. 5, 2005; revision accepted Sept. 10, 2005

Abstract: Transliteration editors are essential for keying-in Indian language scripts into the computer using QWERTY keyboard. Applications of transliteration editors in the context of Universal Digital Library (UDL) include entry of meta-data and dictionaries for Indian languages. In this paper we propose a simple approach for building transliteration editors for Indian languages using Unicode and by taking advantage of its rendering engine. We demonstrate the usefulness of the Unicode based approach to build transliteration editors for Indian languages, and report its advantages needing little maintenance and few entries in the mapping table, and ease of adding new features such as adding letters, to the transliteration scheme. We demonstrate the transliteration editor for 9 Indian languages and also explain how this approach can be adapted for Arabic scripts.

Key words: Transliteration editor, Indian languages, Universal Digital Library (UDL)
doi:10.1631/jzus.2005.A1354 **Document code:** A **CLC number:** TP391

INTRODUCTION

Transliteration editors are essential for keying-in Indian language scripts into the computer using QWERTY keyboard. Applications of transliteration editors in the context of Universal Digital Library (UDL) include entry of meta-data and dictionaries for Indian languages. In this paper we propose a simpler approach for building transliteration editors for Indian languages using Unicode and by taking advantage of its rendering engine available in Windows XP and Linux operating systems. We use the transliteration scheme referred to as IT3 developed by IISc Bangalore and Carnegie Mellon University to represent the Indian language scripts.

The Indian language scripts are syllabic in nature and consist of V, CV, CCV and CCCV type of units, where C is a consonant and V is a vowel. The property of these scripts in that a syllable always ends with a vowel makes it easy to identify the syllables using vowels as anchor points.

To render the syllables on the computer screens, we use Unicode, and the Unicode rendering engine available in Windows XP and Linux operating systems. To display a CV unit, we concatenate the UTF-8 sequence of C and V to cause the Unicode rendering engine to render appropriate shape for CV. To display a CCV unit, we need to render the consonant cluster, so a special character called Halant/Viraam ($) is introduced between every two consonants. So to render CCV we concatenate the UTF-8 sequence of C$CV. To display CCCV type of unit, we concatenate the UTF-8 sequence of C$C$CV. In these syllables, if the vowel is of type schwa (short vowel /a/) then it is nullified as the last consonant in the syllable inherits it by default. Every consonant in the Indian language scripts inherits schwa and so Unicode representation too. However, if the vowel is a non-schwa, then the UTF-8 sequence of Maatra of the corresponding vowel is used. A Maatra is a modified shape of a vowel when it is combined with a consonant. Each vowel has only one Maatra.

In this paper, we demonstrate the usefulness of such a simple scheme to train transliteration editors for Indian languages, and report its advantages needing few entries in the mapping table, little maintenance, and ease of adding new features such as new letters to the transliteration scheme. We demonstrate the transliteration for 9 Indian languages and also explain how this approach can be adapted with few modifications for Arabic scripts.

This paper is organized as follows: Section 2 describes the nature of Indian language scripts. Section 3 discusses the shape of an Akshara and its rendering aspects. Section 4 introduces Unicode and UTF-8 representation. Section 5 describes the rendering of Aksharas of type CV, CCV and CCCV using Unicode. Section 6 describes the characteristics of IT3 transliteration scheme and how to key-in the Indian language scripts. Sections 7 and 8 detail the implementation of Indian language editing. Section 9 explains how Unicode and IT3 based editing can be extended to Arabic, Urdu and Persian scripts.

INDIAN LANGUAGE SCRIPTS

Indian language scripts originated from the ancient Brahmi script. The basic units of the writing system are referred to as “Aksharas”. The properties of Aksharas are as follows: (1) An Akshara is an orthographic representation of a speech sound in an Indian language; (2) Aksharas are syllabic in nature; (3) The typical forms of Akshara are V, CV, CCV and CCCV, thus have a generalized form of C*V; (4) An Akshara always ends with a vowel; (5) White space is used as word boundary thus separating Aksharas present in two successive words; (6) The scripts are written from left to right; (7) Roman digits (0...9) are used as numerals. Some of the languages have their own numeric symbols which are rarely used; (8) English Punctuations marks such as comma, full stops are mostly used in writing. Languages such as Hindi have a set of their own punctuation marks which are often used.

Convergence and divergence of Indian language scripts

India is a multi-lingual nation with 17 recognized official languages. These languages are: Assamese, Tamil, Malayalam, Gujarati, Telugu, Oriya, Urdu, Bengali, Sanskrit, Kashmiri, Sindhi, Punjabi, Konkani, Marathi, Manipuri, Kannada and Nepali. Except Urdu and English, all of the remaining official languages have a common phonetic base, i.e., they share a common set of speech sounds.

While all of these languages share a common phonetic base, some of the languages such as Hindi, Marathi and Nepali also share a common script known as Devanagari. But languages such as Telugu, Kannada and Tamil have their own scripts. The property that makes these languages separate can be attributed to the Phonotactics in each of these languages rather than the scripts and speech sounds. Phonotactics is the permissible combinations of phones that can co-occur in a language.

SHAPE OF AN AKSHARA

The shape of an Akshara depends on its composition of consonants and the vowel, and sequence of the consonants. In defining the shape of an Akshara, one of the consonant symbols acts as pivotal symbol. Depending on the context, an Akshara can have a complex shape with other consonant and vowel symbols being placed on top, below, before, after or sometimes surrounding the pivotal symbol.

Ideally the basic rendering unit for Indian language scripts should be Aksharas themselves. However, a language such as Telugu has around 15 vowels and 36 consonants. To render Aksharas as a whole unit, it requires 540 CV units, 19440 CCV units and 699840 CCCV units. It is reasonable to assume that not all combinations of consonant clusters are allowed, but even then nearly more than 10000 Aksharas are needed as rendering units.

Due to this large number of units, an Akshara is rendered by concatenating the consonant and vowel symbols. The following are the symbols used to render Aksharas by a Unicode rendering engine.

Consonant symbol

A consonant symbol in an Indian language represents a single consonant sound and also an inherent vowel (short /a/). Akshara is syllabic so each consonant symbol represents a consonant and the inherent vowel.

Half forms of the consonant

Aksharas of the type CCV or CCCV, have more than one consonant. In these cases, the initial consonant(s) have half-form shape. These half-forms do not have an inherent vowel.

Vowel symbols (independent vowels)

A vowel symbol represents a vowel sound and is used to render a syllable of type V which has no consonants before or after it. These vowel symbols are also referred to as independent vowels.

Maatra (dependent vowels)

Consonants can associate with vowels other than inherent vowel. If a non-inherent vowel is needed, then a diacritical mark corresponding to the non-inherent vowel is added to the consonant symbol. These vowels with their attached diacritical marks are referred to as Maatras or dependant vowels. A Maatra can occupy a place on top, below, before, after, or sometimes surrounding the consonant symbol and thus is often referred to as dependant vowel.

Viraam

Sometimes it is necessary to write consonants without inherent vowels. To remove the inherent vowel from a consonant, a symbol called Viraam is used. The symbol may be an oblique stroke under the consonant symbol, or can change the shape of the consonant if it is on top of the consonant.

UNICODE: A UNIVERSAL CHARACTER SET

Computers can only interpret bits and bytes, and hence the representation of a script should be defined in terms of bits and bytes. ASCII—American Standard Code for Information Interchange is an 8-bit code to represent the English character set. Similarly there is Indian Standard Code for Information Interchange (ISCII) that defines an 8-bit character code for Indian language scripts. As these codes overlap, computers using ASCII character set cannot interpret ISCII as a code for Indian language scripts. With an 8-bit code, only 256 unique characters can be defined. To allow computers to represent any character in any language, the international standard ISO 10646 defines the Universal Character Set (UCS). UCS contains the characters to practically represent all known languages in the world. ISO 10646 originally defined a 32-bit character set. Each character is assigned a 32 bit code. However, these codes vary only in the least-significant 16 bits. ISO 10646 and Unicode though started as two projects finally merged their character set around 1991 so that both are now compatible with each other.

In addition to the character set, Unicode standard specifies recommendation for renderning of the scripts, handling of bi-directional texts that mix for instance Latin (left to right writing system) and Hebrew (right to left writing system), algorithms for storage and manipulation of Unicode strings (Alan, 2005; The Unicode Consortium, 2003).

UTF-8 and UTF-16

It has to be noted that Unicode is a table of codes that assigns integer numbers to characters (Markus, 2005). One still has to define its implementation or encoding in the computers. A straightforward encoding of these integers is to store the Unicode text as sequences of 2 byte sequences. This encoding is referred to as UTF-16. An ASCII file can be transformed into a UTF-16 file by simply inserting a 0x00 byte in front of every ASCII byte.

However, operating systems such as Unix/Linux have been written based on ASCII (1 byte code) character set and they expect each byte as a character. For these reasons, UTF-16 may not be an appropriate encoding of Unicode in the case of filenames, text files, environment variables, etc.

The UTF-8 encoding uses Unicode compatibly with operating systems working with 1-byte characters.

UTF-8 has the following properties:

Unicode characters U+0000 to U+007F (ASCII) are encoded simply as bytes 0x00 to 0x7F (ASCII compatibility). This means that files and strings which contain only 7-bit ASCII characters have the same encoding under both ASCII and UTF-8.

All Unicode characters >U+007F are encoded as a sequence of several bytes, each of which has the most significant bit set. Therefore, no ASCII byte (0x00~0x7F) can appear as part of any other character.

The first byte of a multibyte sequence that represents a non-ASCII character is always in the range 0xC0 to 0xFD and it indicates how many bytes follow for this character.

All further bytes in a multibyte sequence are in the range 0x80 to 0xBF. This allows easy resynchronization and makes the encoding stateless and robust against missing bytes.

UTF-8 encoded characters may theoretically be up to six bytes long tio handle 32-bit character set, However for 16-bit characters the UTF-8 encoding is only up to three bytes long.

The bytes 0xFE and 0xFF are never used in the UTF-8 encoding. These bytes are used to denote byte order for UTF-16 codes.

Representation of Indian language scripts

Having known the syllabic nature of Indian language scripts, it is easy to understand the notation followed by the Unicode to represent Indian language characters. The following are the principles used by Unicode to represent the Indian language characters:

(1) A Unicode is assigned to each consonant symbol, i.e., a consonant sound along with the inherent vowel.

(2) Each independent vowel is represented by a Unicode.

(3) Each Maatra is also represented by a Unicode.

(4) Viraam has a Unicode number too.

Unicode rendering recommendations

It should be noted that half-forms of the consonants are not represented by the Unicode. The half-forms are essential to render Aksharas involving consonant clusters. The Unicode recommendation for rendering resolves the issue of half-forms. These rules describe the mapping between Unicode characters and the glyphs in a font. They also describe the combining and ordering of these glyphs.

These recommendations are used to build Unicode rendering engines in Windows and Linux operating systems for displaying Unicode characters.

RENDERING OF AKSHARAS

As noted in Sections 2 and 3, an Akshara is syllabic in nature, and it is rendered by concatenating a sequence of consonant and vowel symbols. In this section we will examine the rendering of Aksharas of type, CV, CCV and CCCV for the case of Telugu and Hindi.

Rendering a CV unit

If the vowel is the inherent vowel (short /a/), then to render a CV unit we use the Unicode number of the consonant symbol. The following examples show the rendering of two CV units in Hindi and Telugu.

Hindi:

utf-8(2325) → क , where 2325 is the Unicode number of the consonant symbol and utf-8() is a function which converts the integer value to its UTF-8 format.

Telugu:

utf-8(3093) → క

If the vowel is non-inherent vowel then the rendering of a CV unit is a two stage process.

Remove the inherent vowel from the consonant symbol. This can be done by adding a Viraam to the consonant symbol. Now add the Maatra of the corresponding vowel. Please note that a vowel occurring beside a consonant is always a dependent vowel and so Maatra should be used.

The following examples display the rendering of two CV units where V is a non-inherent vowel.

Hindi:

क + ि → कि

utf-8(2325) + utf-8(2367)

Telugu:

క + ి → కి

utf-8(3093) + utf-8(3135)

The Unicode 3093 represents the full consonant symbol, and the Unicode 3135 represents the Maatra. The concatenation of a Maatra to the consonant symbol changes its shape to get the required CV unit.

Any desired CV unit can thus be produced by a simple concatenation of UTF-8 representation of the consonant symbol and the Maatra

Rendering a CCV unit

Let C1 and C2 denote the first and second consonant in a CCV unit. To render the CCV unit:

(1) The consonant C1 has to be rendered in its half-form. Half-form denotes a consonant with its inherent vowel being suppressed. To perform this suppression, the Viraam is added to the consonants symbol of C1.

(2) The remaining syllable C2V has to be rendered as a CV unit as explained in Section 5.1.

The following examples demonstrate the rendering of CCV units in Telugu and Hindi.

Hindi:

क + ् + र + ि → क्रि

utf-8(2325) + utf-8(2381) + utf-8(2352) + utf-8(2367)

Telugu:

క + ్ + ర + ి → క్రి

utf-8(3093) + utf-8(3149) + utf-8(3120) + utf-8(3135)

In the above examples, 2381 and 3149 are the Viraam symbols for Hindi and Telugu respectively. The concatenation of UTF-8 encoding of 3093 and 3149 produces a half-form of the consonant /k/ in Telugu. The concatenation of this half-form with the remaining sequence produces the required CCV unit.

Rendering of a CCCV unit

If we denote the first two consonants as C1 and C2, then the concatenation of the CCCV can be achieved by producing the half-forms of C1 and C2 and then concatenating them with the remaining syllable.

IT3—TRANSLITERATION SCHEME

From the end user point of view, the details mentioned in all of the above sections are masked. He/she is concerned with how to key-in the Indian language scripts. There are many transliteration schemes such as ITRANS to key-in the Indian language scripts. The focus of these schemes was mainly to represent the Indian language scripts and paid less attention on the importance of user readability aspect. IT3 is a transliteration scheme developed by IISc Bangalore, India and Carnegie Mellon University with the primary focus on user readability of the transliteration scheme (Ganapathiraju *et al*., 2005).. The following are the salient points of this transliteration scheme.

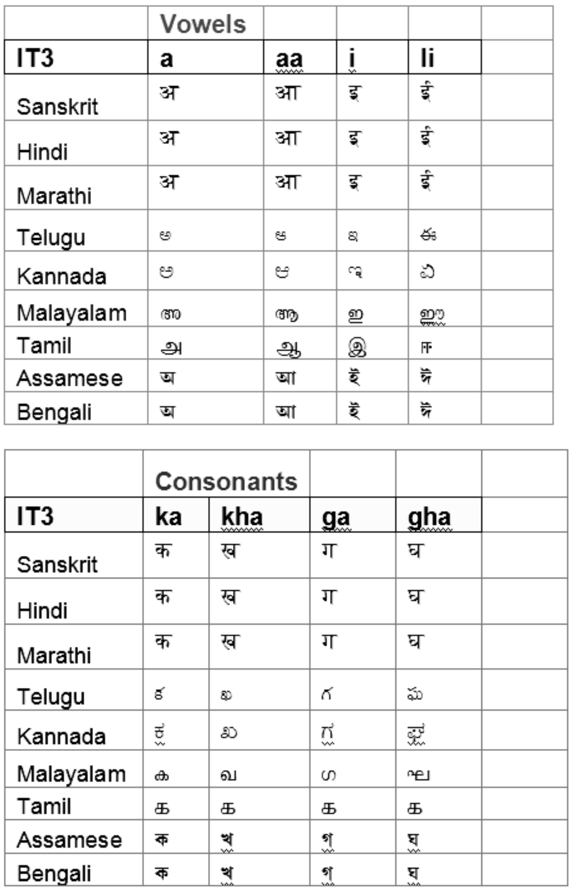
(1) It is case-insensitive.

(2) This scheme is phonetic in nature, the characters corresponds to the actual sound that is being spoken. Thus a single transliteration scheme is used for all the Indian languages, as they share the same set of sounds.

(3) Each character (corresponding to a phone/sound) should be not more than three letters length.

(4) There should be minimal use of punctuation marks in the composition of a character

In Fig.1, a section of the transliteration scheme is shown.



	Vowels				
IT3	a	aa	i	ii	
Sanskrit	अ	आ	इ	ई	
Hindi	अ	आ	इ	ई	
Marathi	अ	आ	इ	ई	
Telugu	అ	ఆ	ఇ	ఈ	
Kannada	ಅ	ಆ	ಇ	ಈ	
Malayalam	അ	ആ	ഇ	ഈ	
Tamil	அ	ஆ	இ	ஈ	
Assamese	অ	আ	ই	ঈ	
Bengali	অ	আ	ই	ঈ	

	Consonants				
IT3	ka	kha	ga	gha	
Sanskrit	क	ख	ग	घ	
Hindi	क	ख	ग	घ	
Marathi	क	ख	ग	घ	
Telugu	క	ఖ	గ	ఘ	
Kannada	ಕ	ಖ	ಗ	ಘ	
Malayalam	ക	ഖ	ഗ	ഘ	
Tamil	க	க	க	க	
Assamese	ক	খ	গ	ঘ	
Bengali	ক	খ	গ	ঘ	

Fig.1 A section of the transliteration scheme

MAPPING IT3 to UNICODE

Using the IT3 notation and the Unicode characters, one can build a simple transliteration editor in a short amount of time. Any new language can be added to it with minimal effort. To add a language, a mapping table has to build to map an IT3 character to the corresponding Unicode character.

However, attention should be paid to the following issue in building this table. IT3 being phonetic in nature, a character 'k' represents a consonant sound /k/ and there is no notion of inherent vowel (short /a/) associated with the consonant sound. Thus to get the syllable /ka/ one has to write a sequence of two characters 'ka'.

In Unicode a consonant symbol has a consonant as well as the inherent vowel. The issue is whether the IT3 character 'k' should be mapped to the half-form of the consonant /k/ or to the consonant symbol, i.e., the syllable /ka/ which is a CV unit.

Let us say, we map 'k' to half-form of the consonant. This can be achieved by concatenating the consonant symbol i.e., /ka/ and a Viraam. Given this mapping stored as a table, to obtain /ka/ we need to write 'ka', and during rendering concatenate the half-form of 'k' and the Maatra of /a/. But unfortunately Unicode defines Maatra for all other vowels except for /a/.

The above limitation of Unicode forces the IT3-Unicode mapping table to map the IT3 character 'k' to the syllable /ka/. Let us say we map 'k' to /ka/. But as IT3 is phonetic in nature, the user would want to type 'ka' to get /ka/. To resolve the conflict, we should allow the user to type 'ka', but nullify the /a/ before looking for an entry in the mapping table. As another example, if the user types /ki/ then we can get half-form of /k/ and concatenate it with the Maatra of /i/.

BUILDING THE EDITOR

To build the Indian language editor, the following is the pseudo code which is followed:

(1) Given an IT3 word, parse it into sequence of characters. The maximum length of an IT3 character is three letters with each character denoting a speech sound.

(2) Aksharas are obtained from the sequence of characters. An Akshara is of the type C*V, thus every vowel marks the end of an Akshara. This step can also be treated as syllabification.

(3) An Akshara can be V, CV, CCV or CCCV. Rendering of the Aksharas of type CV, CCV and CCCV is done as discussed in Section 5. As IT3 is phonetic in nature, users type the inherent vowel (/a/). This has to be nullified for the reasons discussed in Section 7.

In Fig.2 we show some of the screen shots of the editor built using this simple scheme.

EXTENDING THE APPROACH TO ARABIC, URDU AND PERSIAN

As the extension to the Indian language Transliteration Editor, we have included Middle East languages such as Arabic, Urdu and Persian.

The Middle East languages like Arabic, Persian and Indo-European language Urdu have some common characteristics.

All these three languages are written from right to left. Persian and Urdu are derived from Arabic. The characters of Arabic, Persian and Urdu characters are called alphabets. Each alphabet corresponds to a phone (Library of Congress, 1997).

Arabic:

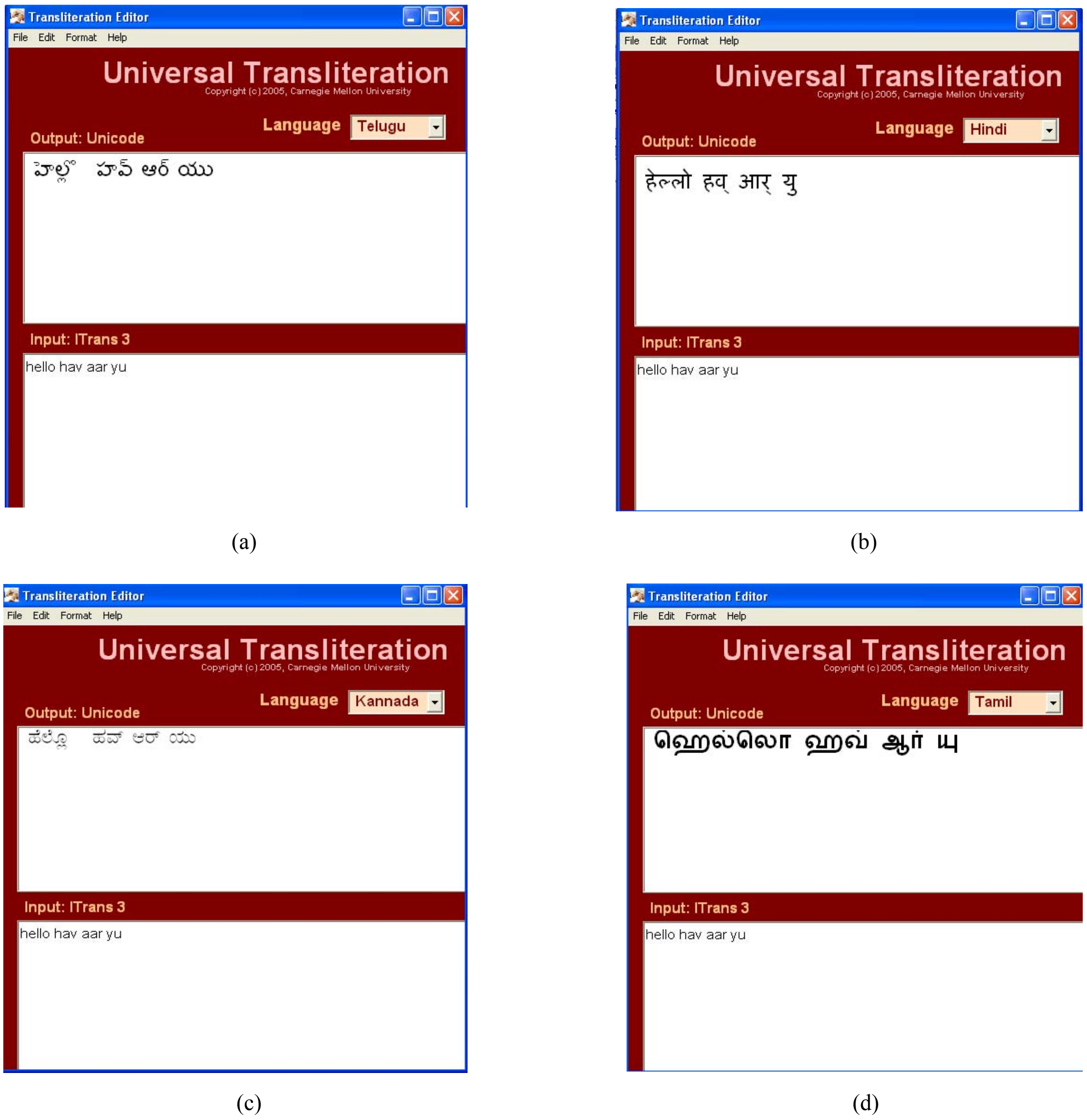
Each alphabet corresponds to a phone. Arabic has 9 vowels and 28 consonants (Library of Congress, 1997; Qur'an Transliteration, 2005). In Arabic it should be noticed that the vowels do not appear independently as seen in many Indian languages, they occur only with a consonant. All the Arabic characters are written in different ways by the occurrence of the same, i.e. the alphabet is written differently if it occurs in the initial position and it is written differently if it occurs in the middle or end of a word. This rule is same for Persian and Urdu.

Persian:

Persian also has 9 vowels and 32 consonants. Modern Persian uses a modified version of the Arabic alphabet. Persian adds four extra alphabets due to the fact that four sounds that exist in Persian do not exist in Arabic. The additional four alphabets are shown in Table 1 (Omniglot, 2005).

Table 1 Extended Persian set

Sound	Shape	Unicode name
[p]	پ	Peh
[tʃ] (ch)	چ	Tcheh
[ʒ] (zh)	ژ	Jeh
[g]	گ	Gaf



Transliteration Editor

Universal Transliteration

Copyright (c) 2005, Carnegie Mellon University

Output: Unicode — Language: Telugu

హెల్లో హవ్ ఆర్ యు

Input: ITrans 3

hello hav aar yu

(a)

Transliteration Editor

Universal Transliteration

Copyright (c) 2005, Carnegie Mellon University

Output: Unicode — Language: Hindi

हेल्लो हव् आर् यु

Input: ITrans 3

hello hav aar yu

(b)

Transliteration Editor

Universal Transliteration

Copyright (c) 2005, Carnegie Mellon University

Output: Unicode — Language: Kannada

ಹೆಲ್ಲೊ ಹವ್ ಆರ್ ಯು

Input: ITrans 3

hello hav aar yu

(c)

Transliteration Editor

Universal Transliteration

Copyright (c) 2005, Carnegie Mellon University

Output: Unicode — Language: Tamil

ஹெல்லொ ஹவ் ஆர் யு

Input: ITrans 3

hello hav aar yu

(d)

Fig.2 Screen shots of the transliteration editor in different Indian languages. (a) Telugu; (b) Hindi; (c) Kannada; (d) Tamil

Urdu:

Urdu is derived from Persian which in turn is derived from Arabic. Urdu uses more complex and sinuous Nastaliq script. It is said that Arabic is a subset of Urdu. Urdu has 11 vowels in addition to the 9 vowels of the Arabic alphabet and 35 consonants (alphabets) (Hugo's Website, 2005; U-TRANS, 2002). Urdu language has two noon (n), one is noon and the other is noon gunna, where as Persian and Arabic has one only noon.

Transliteration*:*

As IT3 is sound based scheme, we used most of the existing IT3 codes for these languages, which we have used for the developing Indian languages Transliteration. But there are additions that are made for some of the sounds which do not exist in the Indian languages. For example the vowels a, aa, i, ii, u, uu, e, o, ai, au have the same notation for Indian and as well as for middle-east languages. But additional IT3 codes such as ain (') and zheh (z') are added to accommodate middle-east languages. The following steps are followed to develop the Transliteration:

(1) Once all the alphabets are assigned IT3 codes, then each IT3 code should be mapped to the corre-

sponding Unicode number.

(2) The above languages have explicit Unicode number assigned to each of the alphabet.

(3) Unlike Indian languages, a consonant alphabet represents a consonant alone and a vowel alphabet represents a vowel. For example in Indian languages "k" is mapped to the Unicode representing /ka/. But in Middle East languages "k" is explicitly mapped to /k/ (kaf). There is no syllabification required in these languages.

CONCLUSION

In this work, we have described the process of building Indian language editors using a simple scheme based on Unicode. This simple approach has the following advantages:

(1) Lesser number of entries in the mapping table. There are only 92 entries (71 characters and 21 punctuation marks) for Telugu language. In the case of ASCII font based Eenadu, a similar mapping table consists of 635 entries; Hindi Language has 93 entries including punctuation marks and Assamese has 89 entries with punctuation marks.

(2) Automatic rendering of the Unicode characters by the Unicode rendering engine in Windows XP/Linux. Using Unicode based approach, a single module can render all the languages. The mapping table changes, but the parsing of IT3 sequence and syllabification are the same across all of the Indian languages. Whereas in the case of ASCII based fonts, one may have to write separate modules for each of the fonts to handle exceptions to render some of the complex consonant clusters.

(3) Easy to adapt for new languages. Unicode based approaches require minimal knowledge to work in new languages, whereas ASCII font based approach requires a better understanding of the language to handle exceptions related to rendering of consonant clusters.

References

Alan, W., 2005. Unicode Resources. http://www.alanwood.net/unicode/index.html.

Ganapathiraju, M., Balakrishnan, M., Balakrishnan, N., Reddy, R., 2005. Om: One Tool for Many (Indian) Languages. Proceedings from the International Conference on Universal Digital Library (ICUDL), Hangzhou, China. *Journal of Zhejiang University SCIENCE*, **6A**(11):1348-1353.

Hugo's Website, 2005. Urdu & Arabic Pages. http://users.skynet.be/hugocoolens/.

Library of Congress, 1997. ALA-LC Romanization Tables: Transliteration Schemes for Non-Roman Scripts. http://www.loc.gov/catdir/cpso/roman.html.

Markus, K., 2005. UTF-8 and Unicode FAQ for Unix/Linux. http://www.cl.cam.ac.uk/~mgk25/unicode.html.

Omniglot, 2005. Persian. http://www.omniglot.com/writing/persian.htm.

Qur'an Transliteration, 2005. Qur'an Transliteration. http://transliteration.org/quran/Home.htm.

The Unicode Consortium, 2003. The Unicode Standard, Version 4.0. Addison-Wesley, Boston, MA.

U-TRANS, 2002. Urdu. http://tabish.freeshell.org/u-trans/.