

A Journey from Indian Scripts Processing to Indian Language Processing

R. Mahesh K. Sinha

Indian Institute of Technology, Kanpur

This overview examines the historical development of mechanizing Indian scripts and the computer processing of Indian languages. While examining possible solutions, the author describes the challenges involved in their design and in exploiting their structural similarity that lead to a unified solution. The focus is on the Devanagari script and Hindi language, and on the technological solutions for processing them.

India is a highly multilingual country with 22 constitutionally recognized languages. Besides these, hundreds of other languages are used in India, each one with a number of dialects. The officially recognized languages are Hindi, Bengali, Punjabi, Marathi, Gujarati, Oriya, Sindhi, Assamese, Nepali, Urdu, Sanskrit, Tamil, Telugu, Kannada, Malayalam, Kashmiri, Manipuri, Konkani, Maithali, Santhali, Bodo, and Dogari. Hindi written in the Devanagari alphabet is India's official national language and has the most speakers, estimated to be more than 500 million.

Indian languages belong to the Indo-European family of languages.¹⁻⁴ Languages of the north and western part of India belong to the Indo-Aryan family (spoken by about 74% of India's speakers) while the languages of the south belong to the Dravidian family (about 24% of India's speakers). The Sino-Tibetan, Austric, and some other groups form the other prominent language families. The Sino-Tibetan family is spoken mainly in the northeastern parts of India, and the Austric-Asiatic group of languages is spoken mainly by the tribal people of India's northern belt.

The languages within each family exhibit much structural similarity. In addition, India's languages have undergone significant mixing and cross-fertilization. Interestingly, the English language brought to this subcontinent with British rule is understood by less than 3% of the country's population, although it continues to be the major language to link federal and state communications and is used in the country's higher-education

institutions. Moreover, English is mandated as the authoritative text for federal laws and Supreme court judgments.^{5,6}

In this article, I present an overview of the historical development of the modern Indic scripts' writing system, their mechanization and adaptation to computing, and I examine how this facilitated development of Indian language processing. I concentrate primarily on the Devanagari script and the Hindi language as these are the most popular on the subcontinent. I do not delve into the history of how modern Indic scripts and languages have evolved; instead, I discuss only those features found in current language usage, and explain how the unifying characteristics of the scripts and languages have been exploited to provide solutions applicable to almost all Indic scripts and languages.

Indian scripts: Background

Ten major modern scripts are currently used in India: Devanagari, Bengali, Oriya, Gujarati, Gurumukhi, Tamil, Telugu, Kannada, Malayalam, and Urdu. Of these, Urdu is derived from the Persian script and is written from right to left. The other nine scripts, written from left to right, originated from the early Brahmi script (300 BC)^{7,8} and are also referred to as *Indic scripts*. The early Brahmi script split into two major branches, one consisting of the north Indian scripts (Devanagari, Bengali, Oriya, Gujarati, and Gurumukhi) and the other south Indian or Dravidian scripts (Tamil, Telugu, Kannada, and Malayalam).

Devanagari script is used for writing a majority of the Indian languages such as Hindi, Marathi, Sindhi, Nepali, Sanskrit, Konkani, and Maithali. *Bengali* script is used for writing Bengali and Assamese. *Gurumukhi* is the script for writing Punjabi. Some of these languages have their own script, and some differ by having a few additional symbols to represent the purity of their sound. Several other scripts are in use but are gradually vanishing, primarily from the lack of political and technological support. A detailed description of the *Devanagari* script will be provided in a later section.

The aforementioned nine scripts besides Urdu are commonly used throughout India. It is estimated that all the literate people in India belonging to the different linguistic zones use their regional script in communication. Most of the urban population is also familiar with the roman alphabet and frequently use and mix Indian languages written in romanized form. Such use is more prominent in advertisements, cinema posters, and text messaging. However, romanized text reading is mostly contextual, and only native speakers can read these correctly because no phonetic marker symbols are featured in these writings.

According to the 2001 Indian census, India's literacy rate was 65.38% and the urban population stood at 27.8%. So, approximately 65.38% people use Indic scripts. Although exact figures are not available, the literacy rate in urban India is estimated to be higher than the national average. Thus, we can say that about 18% to 25% of people use both Indic scripts and romanized text. A large population of about 25 million Indians living abroad, however, knows the Indian language but not the necessarily Indic script—these people use the romanized Indian language text. (I found it interesting that when I Web-enabled the Indian Institute of Technology Kanpur's English-to-Hindi translation system, Indians living abroad overwhelmingly requested Hindi translation in romanized form.)

Social transformation

When we examine the pattern of usage of Indian scripts on computers and other devices, we find a chicken-and-egg situation. The language divide significantly contributes to the digital divide.^{9,10} The benefits of advancements in information technology (IT) have yet to percolate down to the grass-roots level; in fact, IT has contributed to the

widening of the social divide.⁹ Although Internet usage has grown tremendously in India (28 million users), it accounts for a meager 2.72% of the Indian population. India, which constitutes 15% of the world population, accounts for only 2% of global Internet searches (<http://www.comscore.com/press/release.asp?press=2400>). In addition to economic factors, this disparity has resulted largely from a lack of Indian language content on the Web and corresponding tools. The increasing availability of these tools, however, corresponds to an increase in computer usage, especially among mid-level businessmen. In a random survey, I found that more than 90% of such businessmen use their local language written in local script. Computerized land records, driving licenses, voter IDs, and so on are some of the other major applications where local script usage is bringing about a social transformation. India is also witnessing a tremendous growth in mobile phone usage. Nonvoice applications via mobile phones—such as text messaging, cash transfers, and online purchases—have emerged as a major alternative to computer-based e-mail in the lower-middle-income economies,¹¹ which has helped drive significant demand outside metro areas for mobile phones that handle native languages.¹² It's clear that the linguistic interfaces to computers and other devices play an important role in providing economic growth to the rural masses and in bridging the social divide.

Although Indian languages and Indic scripts are several centuries old and symbolize humankind's early evolution, their mechanization and computerization has received little attention, for historical and political reasons, compared to the languages of the Far and Middle East such as Chinese, Japanese, and Korean.¹³⁻¹⁷ A major reason behind this neglect has been that the "elite" portion (less than 3%) of the Indian population with whom the international community conducted business knew English because of longtime British rule. This English-speaking Indian community has led India's economic, industrial, professional, political, and social life.⁶

It is only within the past decade or so, as a result of globalization and emerging markets in India, that IT companies have begun investing in Indian language localization. Researchers in India, however, began working on localization in the early 1970s and came up with elegant solutions unifying

A Journey from Indian Scripts Processing to Indian Language Processing

characteristics of the Indic scripts that formed the basis for India's present technological development on localization, as I will explain.

Script differences and similarities

Indic scripts exhibit a lot of similarity in their features and are all phonetic in the sense that they are written the way they are spoken: there is no rigid concept of "spelling" as with western writing systems. However, the same language spoken in different geographical regions can differ in their accents, which can lead to variations in their spellings.

Indian scripts are a logical composition of individual script symbols and follow a common logical structure we can refer to as the "script composition grammar," which has no counterpart in any other set of scripts in the world. Indic scripts are written syllabically and are usually visually composed in three tiers where constituent symbols in each tier play specific roles in the interpretation of that syllable. In one method of mechanizing Indic scripts,¹⁸ the set of these syllables—which number several thousands—has been used like

those applicable to Chinese or Korean languages. Such solutions do work but are cumbersome and unnecessarily burden a computer system because they do not exploit the logical structure of the Indic scripts.

Most Southeast Asian scripts such as Thai, Burmese, Lao, Khmer (Cambodia), and Bali are similar to Indic scripts.¹⁹ Although the work on mechanizing these scripts started in the 1960s by IBM,¹⁹⁻²¹ the scripts' unifying characteristics have not been exploited in finding solutions in terms of devising the computers' internal codes and uniformly rendering script output. Yet no other group of scripts in the world presents such unifying characteristics as found in Indic (South Asian) scripts and Southeast Asian scripts.

Features of Indian scripts

A look at the major features of the Devanagari script^{7,22,23} will help illustrate the complex nature of mechanizing Indian languages; examples are included from other Indian scripts wherever there are variations. The Indic scripts have a number of consonants, each of which represents a distinctive sound. These are arranged in different classes based on the articulatory mechanism used to produce the corresponding sound. At a broad level, these classes (called *varga*) are velar, palatal, retroflex, dental, labial, and a few others. The consonants in each *varga* are further arranged in the order of the voiceless and voiced plosives followed by the corresponding nasal sound. Each voiceless and voiced plosive category is further divided into two parts, unaspirated and aspirated. In the "others" category, we have the fricatives, sibilants, and some other forms. Figure 1 shows the Devanagari consonants and depicts their individual positions. The top row in each *varga* is what is referred to as the "full" consonants. The full consonants have an inherent vowel sound of "a" attached to it. In the second row of each *varga*, the corresponding "pure" consonant form (usually referred to as the "half letter" form) is shown. The half-letter form represents the absence (muting) of the inherent vowel sound.

Visually, we derive the pure consonant form in the Devanagari script from the full consonants by deleting the vertical line at the end (end-bar) or by putting a *halant* sign (see Figure 2) at the bottom of the letter.²⁴ In the case of middle-bar characters, it is shown by straightening of the half loop at the end.

Figure 3 shows the Devanagari vowels. These are also arranged according to the

	voiceless plosives		voiced plosives		Nasal
	unaspirated	aspirated	unaspirated	Aspirated	
velar कवर्ग	क क्	ख ख्	ग ग	घ घ	ङ ङ्
palatal चवर्ग	च च्	छ छ्	ज ज	झ झ	ञ ञ
retroflex टवर्ग	ट ट्	ठ ठ्	ड ड	ढ ढ	ण ण
Dental तवर्ग	त त्	थ थ	द द	ध ध	न न्
labial पवर्ग	प प्	फ फ	ब ब	भ भ	म म्

	palatal	retroflex	dental	labial
Fricative	य र	र र	ल ल	व व
Sibilant	श ष	ष ष	स स	
Aspirate Others	ह ह	ळ ळ		

Figure 1. Ordered list of consonants in full and pure consonant forms.

articulation of sounds and their short and long duration. For each vowel, other than the first denoting an “a” sound, there is a corresponding modifier symbol called a *matra*. A *matra* can be attached to a full consonant or a consonant cluster (also known as a conjunct), imparting its sound to the consonant/conjunct. Only one *matra* can be attached to a consonant/conjunct. Figure 4 shows some of the diacritical marks used in Devanagari script.

A pure consonant, or a sequence of the pure consonants, followed by a full consonant forms a *consonant cluster*—or a *conjunct*. Conjuncts are formed in one of two ways. One is by explicitly using the *halant* symbol (or an equivalent symbol in other scripts), and the other is to graphically combine the two shapes to generate a new glyph. Figure 5 depicts some of the conjuncts along with their constituents. Note that the visual shapes of the conjuncts can be completely different from their constituents. Often, the second consonant glyph is reduced and attached to the first consonant vertically or horizontally. The total number of conjuncts can number as many as 3,000. In early handwriting and typesetting, a large number of conjuncts were frequently used; today, however, people commonly use a much smaller set of conjuncts—usually only 20 to 25.

Conjuncts, regardless of how formed, are all equivalent, and the user can decide which form to use, depending on how elaborate the text is that the user is composing. Even the individual consonant symbols can have different, but equivalent, shapes. Some of the consonants with the *nukta* diacritic behave as an independent consonant with a slightly different sound (see Figure 6 for examples). Further, the conjuncts formed with the consonant corresponding to the *ra* sound yield special symbols attached to the associated consonant. When a pure consonant (half letter) is followed by a *ra* consonant, a symbol called *ra-kar* is attached to the corresponding full consonant. This *ra-kar* symbol is a small left-leaning diagonal line attached to the bottom vertical stem of the consonant. When there is no vertical stem at the bottom of the character as in case of the retroflex class, a small inverted “v” shape is attached at the bottom of the character. On the other hand, if the pure consonant *ra* is followed by a full consonant, a symbol called *reph* (a small c-shaped curve) is attached to the top of the full consonant. Figure 7 gives examples.

'halant' symbol signifies muting of inherent vowel sound 'a' (dotted circle denotes a full consonant)	◌̣
---	----

Figure 2. Halant symbol.

vowel	अ	आ	इ	ई	उ	ऊ	ए	ऐ	ओ	औ	ऋ	ॠ
'matra' symbol		ा	ि	ी	ु	ू	े	ै	ो	ौ	ृ	ॄ

Figure 3. Devanagari vowels with corresponding *matra* symbols (dotted circle denotes a consonant/conjunct).

anuswar	'anunashik' / 'chandra-bindu'	chandrama	visarg	Nukta
◌̣	◌̣̣	◌̣̣̣	◌̣̣̣̣	◌̣̣̣̣̣

Figure 4. Devanagari diacritical marks (dotted circle denotes a consonant/vowel).

क + ् + ष = क्ष; त + ् + र = त्र; ज + ् + ञ = ज्ञ;
 श + ् + र = श्र; क + ् + त = क्त; त + ् + त = त्त
 द + ् + द = द्द; द + ् + ध = द्ध; द + ् + व = द्व;
 द + ् + य = द्य; प + ् + त = प्त; ष + ् + ट = ष्ट

Figure 5. Some example conjuncts in Devanagari are shown with their constituent symbols.

ड + ◌̣ = ड़; ढ + ◌̣ = ढ़; क + ◌̣ = क़; ज + ◌̣ = ज़
 फ + ◌̣ = फ़; ग + ◌̣ = ग़

Figure 6. Some examples of Devanagari characters with the *nukta* diacritic attached.

क + ् + र = क्र; प + ् + र = प्र; द + ् + र = द्र;
 ड + ् + र = ड़; र + ् + र = र्र; र + ् + म = र्म;

Figure 7. Some example conjuncts in Devanagari formed with the *ra* consonant.

The *anuswar* and nasalization symbols in the Devanagari script need special mention. When an *anuswar* is used on top of another symbol, the nasalization of the *varga* to which the *following* consonant belongs comes into effect while speaking. Where a *following* consonant is absent, the corresponding associated vowel sound on the consonant to which the *anuswar* is attached is nasalized. Thus, there are two forms of conjuncts with nasalization, one with the *anuswar* symbol and the other that explicitly uses the nasalization character. Both of these forms are equivalent and are frequently used. Unfortunately, many Hindi writers today do not follow this rule that comes

A Journey from Indian Scripts Processing to Indian Language Processing

पंक = प + ङ्क = पङ्क; पंच = प + ञ्च = पञ्च
 पंड = प + ण् + ड = पण्ड ; पंत = प + न् + त = पन्त
 पंप = प + म् + प = पम्प

Figure 8. Some examples showing the use of the *anuswar* symbol in Devanagari and its equivalent conjunct form.

from the restrictions imposed by the articulatory mechanism of the sound. Figure 8 shows examples.

From the description thus far, it is clear that the Devanagari script is a logical composition of its constituent symbols. From a more technical viewpoint, it is possible to define a script composition grammar for the script.²⁵ This also holds true for all other Indic scripts, with minor variations. Figure 9, which is my own formulation, shows this grammar in Backus-Naur Form notation; note that it gives the script composition grammar formulation only at the logical, not visual, level. The visual-level formalism is available elsewhere in a finite state machine I designed for Devanagari OCR work.²⁶

Now, let us examine how the Indic scripts are visually composed. Indic scripts are written from left to right and juxtapose the composite characters as defined in Figure 9; typically, the characters appear to be hanging from a horizontal baseline. With the Devanagari, Bengali, and Gurumukhi scripts, this horizontal line (called a *shirorekha*) is physically drawn and visible; in other scripts this line is virtual.

As I have mentioned, Indic scripts are usually written in three tiers. Figure 10a shows an example word. The middle (core) tier is just below the *shirorekha*; it holds all the main characters (vowels, consonants, and

```
<vowel> := {list of vowels};
<matra> := {list of 'matra' symbols};
<diacritic> := {list of diacritic marks};
<full_consonant> := {list of full_consonants};
<pure_consonant> := {list of pure_consonants};
<conjunct> := <pure_consonant>+ <full_consonant>
<composite_character> :=
    <vowel> <diacritic>* |
    <full_consonant> <diacritic>* |
    <full_consonant> <matra> <diacritic>* |
    <conjunct> <diacritic>* |
    <conjunct> <matra> <diacritic>*
<word>:= <composite_character>+
```

Figure 9. Indic script composition grammar.

(There may be restrictions on the use of certain diacritic marks on symbols that this formulation has not considered).

Devanagari Word: कुर्सियों (English meaning 'chairs');

Constituent Composite Characters(3): कु + र्सि + यों ;

कु = क (consonant in core strip middle tier) + ु (lower 'matra' in lower tier) ;

र्सि = र्स (conjunct aligned with 'shirorekha') + ि ('matra' symbol in middle and upper tier; this gets attached to the left of the conjunct) ;

र्स = र (consonant 'ra') + ्र ('halant' symbol making 'ra' a

pure consonant rendering a 'reph' sign (ँ) in upper tier on

the following character) + स (consonant in middle tier) ;

यों = य (consonant in middle tier) + े ('matra' symbol in middle and upper tier; this gets attached to the right of the consonant) + ं (diacritical mark 'anuswar' attached in the upper tier) ;

(a) an example word showing three tier composition

क + ि = कि(attached to left); क + ी= की(attached to right);

क + ै = कै(attached at top); क + ु = कु(attached at bottom);

ङ + ि = ङि(Gujarati, attached to left)

क + ो = को (Tamil, split into left and right part)

ङ + ु = ङु (Telugu, split into top and bottom part)

(b) examples of variations in positioning of 'matra' symbols

ट + ्र + र = ट्र ('ra-kar' gets attached in the bottom tier in case of a retroflex consonant)

ट + ्र + र + ु = ड्र (the bottom tier has both 'ra-kar' and a lower 'matra' symbol)

(c) example of 'ra-kar' on a retroflex character with lower 'matra'

र + ु = रु (lower 'matra' symbol gets attached in the middle strip modifying the base shape)

र + ्र = र्र (lower 'matra' symbol gets attached in the middle strip modifying the base shape)

(d) lower 'matra' attached to 'ra' consonant

Figure 10. Examples of Devanagari script composition: (a) example word ("chairs") showing three-tier composition; (b) example of *ra-kar* on a retroflex character with lower *matra*—this is a rare combination, however; (c) lower *matra* attached to *ra* consonant; and (d) examples of variations in positioning of *matra* symbols.

conjuncts) and the *aa-kar matra* symbol. The lower tier is exclusively for the lower *matra* symbols, and *halant*, *dot* diacritic, or *ra-kar* sign used with retroflex characters for the Devanagari script. For retroflex characters with the *ra-kar*, the lower *matra* symbol can go in a tier just below the lower tier making it a four-tier composition (see Figure 10b), but such combinations are rare and typically people adjust the height to accommodate the fourth tier into the third. In one exception, the lower *matra* symbol gets attached to the *ra* consonant in the core tier itself with a change in shape (see Figure 10c). The upper tier, above the *shiroreka* line, is used for the

upper *matra* symbols, diacritical marks, and the *reph* sign for Devanagari script. There are four *matra* symbols (*i*-kar, *ii*-kar, *o*-kar, and *au*-kar) that occupy the core tier and extend to the upper tier. Figure 10d shows examples. These examples clearly show that the *matra* symbols get attached to the left, right, top, or bottom of the character. For some scripts, the *matra* symbol may be split into two parts: one may get attached to the left, the other to the right. In some Indic scripts, the shape of the base character or *matra* symbol, or both, changes after the composition.

Early mechanization of Indian scripts

Printing technology arrived with Christian missionaries who came to India in 1556 and who wanted to print the Bible in the Indian languages (http://www.orientalhanne.com/history/news_2007_04_4.htm). Printing did not become popular, however, until the 18th century.²⁷ The earliest type-based Devanagari printing was in 1796 in Kolkata (Calcutta).²⁸ The first publication produced in Devanagari type was developed by Charles Wilkins, an English typographer and noted orientalist who first translated the *Bhagavad-Gita* into English.²⁹ He was also closely involved in the design of the first type for printing Bengali.

The technology for printing the Indian scripts was adapted from Western technology. For type-based printing, a large set of precast conjuncts—the individual characters and symbols running into the thousands, of varying sizes and shapes—were used for manual composing on a three-tier block. An entire page was composed manually with these juxtaposed blocks, but the rest of the process was the same as that for roman-alphabet printing. Printing quality depended on the quality of the typecast used and on the manual layout of the words and the page, as well as on the printing mechanism used.

The first Devanagari typewriter was introduced around 1930.³⁰ Designed by V.M. Atre in Germany and named *Nagari Lekhan Yantra*, the typewriter was built by Remington. In 1964, the government of India's Department of Official Language approved a keyboard layout for Devanagari to which further modifications were recommended in 1969.³¹ The Indian typewriter company Godrej developed the Devanagari typewriter in October 1968 in collaboration with Optima, a German company. L.S. Wakankar designed both the layout and the typefaces for this.³¹

The Devanagari typewriter, an adaptation of the English typewriter, had to accommodate Devanagari symbols in place of the 26 upper- and lowercase roman letters on the keyboard. The typewriting printing mechanism was also modified to allow the multitier composition of the Devanagari script. In summary, the basic mechanisms used for this adaptation are as follows:

- All the Devanagari characters that ended in a vertical line were used with the vertical line removed on the keyboard tops for layout. Recall that this set corresponds to the half-letters (pure consonant forms).
- The vertical bar, *halant* symbol, diacritical marks, nonvertical bar characters, and some of the half-characters such as ँ and ए all had a place on the key tops.
- Among the vowels and the *matra* symbols, only basic shapes were placed on the key tops; the other shapes were composed using a combination of keys.
- If spare unallocated key tops were available, the frequently used vertical bar characters were given a place on them.
- The concept of the “dead” key (overstrike) and the “half-backspace” (move backward by half a character width) were introduced, making it possible to position the lower and some upper *matra* symbols.
- Symbols could be vertically composed by appropriate positioning of the typeface slugs by the typing-striking-hand associated with the key tops.
- The keyboarding method relied on the visual, rather than the logical, order of characters. The typist learned how to generate the script graphics by using the key top symbols; the order creating the script followed the order as seen on paper. The process had no correlation to composite-character composition logic discussed earlier.

Figure 11 shows a mechanical Hindi typewriter and a sample of typed text. These machines found extensive use for producing a low-volume document in the Hindi language. Such typewriters are still widely used, especially in places where electricity is not easily available. It is obvious, however, that the quality of the typewritten Hindi text is poor, with broken lines, broken characters, and bad alignment. The poor quality worsens because of mechanical wear and tear, resulting in inaccuracies in the half-back-spacing and

A Journey from Indian Scripts Processing to Indian Language Processing



हमारा इन्स्ट्रुमेंट देवनागरी कंप्यूटर टर्मिनल 16 बिट माइक्रो कंप्यूटर डेवलपमेंट सिस्टम पर आधारित है। हमने अपना की-बोर्ड, ग्राफिक टर्मिनल बहुदेशीय एडिटर और कर्नल प्रोसेसर बनाया है जो हम सम्मेलन में इसका प्रदर्शन करेंगे। यह एक बहुत ही शक्तिशाली, प्रबुद्ध (इंटेलिजेंट) और बहुदेशीय टर्मिनल है जिस पर रोमन, देवनागरी और भारतीय लिपियों में तथा ग्राफिक्स में बड़ी सुविधा से काम किया जा सकता है। इस टर्मिनल का विकास इलेक्ट्रॉनिक कमीशन के टी.डी.सी. द्वारा दिए गए सहस्रता अनुदान से किया गया है। अब हम इसे एक स्वाधारित (स्टैंड एलोन) टर्मिनल के रूप में विकसित कर पूर्णस्वतंत्र व्यावसायिक प्रतिस्पर्ध देना चाहते हैं। इसके लिए हमने हार्डवेयर मँगवा लिए हैं। यह काम हमें ज्ञाता है कि अप्रैल, 1984 तक पूरा कर लेंगे। साथ ही इस टर्मिनल की बहुदेशीय क्षमता को दर्शाने के लिए हम कोटो कम्पोजिशन मशीन और लेसर प्रिंटर से भी लगाकर इसकी उपयोगिता का परिचय सामने लाना चाहते हैं।

Figure 11. (Top) Mechanical Hindi typewriter and (bottom) a sample of typewritten text.

the dead-key mechanisms. In the 1960s, however, there were few, if any, alternatives.

The advent of microprocessors in the 1970s made electronic typewriters possible (<http://en.wikipedia.org/wiki/Printing>). The keyboard layout and the keyboarding scheme for Hindi remained the same on these, but output quality improved significantly. The characters and the symbols were stored in ROM, and the words were composed in RAM in bitmap form. These bitmaps were then printed using a dot matrix printer. The 5×7 or 7×9 dot matrix used for roman script was inadequate for representing the complex curves of Indic scripts, so one solution was to print row by row, but this made printing slow. The other solution was to print in the tiers of the 5×7 matrices.

Then came the 24-pin printer, which was a great relief. In addition to the better print quality, referred to as near-letter quality,

some of the electronic typewriters now also provided a small display where users could view the composition before printing and make corrections if needed. Next came the IBM Selectric ball and daisy wheel typewriters. These generated characters by impact printing, and the typewriters' design was similar to mechanical typewriters except that the mechanisms were more rugged and had electronic motor control. Now it was possible to achieve boldface letters by "repeat" printing or by a slightly deviated printing to make the character appear broader. Moreover, it was possible to support different fonts by changing the ball or wheel. The quality obtained was called *letter quality*. These devices, however, were slower than the matrix printers.

In all these adaptations for Indic scripts, vendors tried to support good font quality and to handle ligatures and more-frequent conjuncts. Obviously, it was not possible to cover the set of conjuncts once available with the letterpress machines. Separate conjunct and ligature wheels were provided with the 1970s adaptations, however, and the printer could prompt for a change of the wheel—a cumbersome, slow, and tedious process. In all these developments, few attempts were made to optimize the keyboard layout and the keyboarding process: typists simply learned to adjust to the highly inefficient, somewhat irrational keyboard layout and associated keyboarding scheme.

Before proceeding with the technical details for processing scripts on computers, it will help provide context to take a look, in the next section, at early investigative efforts and IIT Kanpur's role.

Computers, scripts, and early efforts

Although researchers had made several investigations into computer processing of Indian languages using a romanized version of the text, it was only in the 1970s that computer issues specifically involving Indic scripts and computers were first investigated. In 1970, I and other researchers at the Indian Institute of Technology (ITT) Kanpur undertook the task of first analyzing the logic basis of Indic scripts in preparation for mechanizing them.^{32,33}

IIT Kanpur

IIT Kanpur is a leading educational technical institution in India and in the world. It acquired its first computer in 1963 under the Kanpur Indo-American program (KIAP)

and was the first educational computer system established in the northern part of India. Very soon, the institution became a focal point for computer training and awareness. The institute ran a number of short-term programs on computer programming in Fortran for teachers from other engineering and science institutions. Demand was great for acquiring computing skills, and the computing resources were scarce. Soon IIT Kanpur upgraded its computing infrastructure, from an IBM 1620 to a DEC 7044.

In 1969, I joined IIT Kanpur in its PhD program after I had obtained a master's degree at IIT Kharagpur in electronics and communications, with a specialization in industrial electronics. At that time, computer science was not a separate discipline; it was offered only as a specialization in the Department of Electrical Engineering. For my PhD, I started working on fault tolerance in digital circuits. In 1970, one of my professors, H.N. Mahabala, had returned from a visit to the Massachusetts Institute of Technology and described an OCR project at MIT to build a reading machine for the blind. Intrigued, I was motivated to switch from investigating fault tolerance to designing an OCR system for Devanagari script—it was a new topic in uncharted territory and much more challenging than working on OCR for a roman alphabet.

That project was the beginning of any formal exploration on mechanizing Indian scripts. Some of my colleagues expressed ridicule as well as surprise that I should choose to work on Indian languages at a time when it was almost inconceivable that Indian languages could be used on expensive computer systems, which remained within reach of only a very few in India. I had a strong conviction, however, that the benefits of computing technology could truly reach people only through their own language, and therefore that we Indians had to make a beginning in this direction.

While I pursued the design of an OCR system for Devanagari script, Putcha Narasimham, a Telugu-speaking colleague at IIT Kanpur, was working on his master's degree. We regularly had discussions examining features of the scripts of northern and southern India, coming as we did from those two different areas. We soon discovered the unifying patterns of Indian scripts that became the basis for enabling computers to work with Indian scripts. Putcha, who was taking a systems engineering course at IIT Kanpur,

needed a term paper topic, and found this problem of designing a keyboarding scheme for Indian languages to be highly suitable; the results were soon published.³² I myself presented an alternative schema for the same topic that differed in the manner in which the pure consonant forms were derived.³³ These investigations resulted in the later development of a universal keyboarding scheme and a unified internal code for information exchange that was applicable to all Indic scripts.

After completing a PhD in 1973 on Devanagari OCR³⁴ and serving at Banaras Hindu University for a couple years as a Reader, I joined IIT Kanpur as a faculty member (assistant professor) in 1975. It was a good opportunity for developing and continuing research in Indian language technology. Motivating students to work on a problem related to Indian-language technology, however, was difficult at a time when almost all the students were aspiring to go to the US for higher studies. Nonetheless, I encouraged them to tackle the language problem, explaining the challenges and the fact that the problem's solutions must come from us within India and not from others. Moreover, I persuaded them that R&D in Indian-language technology was a necessity for a highly multilingual, multiple-script country like ours. Consequently, I succeeded in forming a core group with some students and research engineers, and in 1983 this finally led to the breakthrough development of the Integrated Devanagari Computer (IDC) terminal and the Graphics and Indian Script Technology (GIST).³⁵ This technology incorporated several desirable features that made it user friendly, such as applicability to all Indian scripts, a natural keyboarding scheme, an internal representation well suited for information interchange and transliteration, and flexibility in script composition. We publicly demonstrated this system at the Third World Hindi Conference (*Tritiya Vishwa Hindi Sammelan*) in New Delhi in October 1983.

After having achieved breakthroughs at the script level,^{25,33,35-43} I turned my attention in 1984 to solving natural-language processing (NLP) problems for Indian languages. I have always felt that the digital divide within the society cannot be bridged without bridging the language divide.¹⁰ Over time, I developed a methodology for machine-aided translation among English and Indian languages,⁴⁴⁻⁴⁹ work that is still ongoing.

A Journey from Indian Scripts Processing to Indian Language Processing

Key events and contributors

Around the time that I focused on NLP, several faculty and research colleagues also began work in this area, many fanning out in different parts of the country, which triggered activities in other Indian languages and scripts. Two events proved particularly noteworthy: in 1988, the Centre for Development of Advanced Computing (C-DAC) acquired the IDC and GIST technology (GIST was now modified to stand for “Graphics and Intelligence-based Script Technology”). C-DAC (<http://www.cdac.in/>) is a scientific society of the Indian government’s Department of Information Technology. Mohan Tambe, who had been working on IDC and GIST with me at IIT Kanpur, joined C-DAC and became instrumental in forming a group devoted to enhancing and commercializing the technology.⁵⁰ Subsequently, C-DAC released a number of commercial products offering printing solutions, word processing, desktop publishing, and font design, spanning most of the Indian languages and southeast Asian languages.⁵⁰

The second noteworthy event occurred in the 1990s. In 1995, while still at IIT Kanpur, I was instrumental in initiating and mentoring NLP activities at a newly established scientific society of the Government of India’s Department of Information Technology (DIT): the Electronic Research and Development Centre of India (ER&DCI) Lucknow. The DIT’s program on Technology Development for Indian Languages (TDIL: <http://www.tdil.mit.gov.in>) sponsored the project on machine-aided translation (MAT) from English to Hindi based on AnglaBharati technology⁵¹ that I developed, and ER&DCI Lucknow was associated with us in this project for productizing the prototype developed. AnglaBharati’s underlying methodology^{45,51} used a pseudo-interlingual approach exploiting the structural commonality of a group of Indian languages. A number of ER&DCI Lucknow’s engineers—when ER&DCI had moved to Noida and became ER&DCI Noida—underwent training with us at IIT Kanpur, which helped them in establishing an NLP center of their own. Subsequently, they acquired the AnglaBharati technology from IIT Kanpur. Under a government reorganization program, ER&DCI Noida eventually became C-DAC Noida. The AnglaBharati technology was also acquired by C-DAC Kolkata and C-DAC Thiruvananthapuram. At these centers, I mentored the machine translation R&D work; IIT Kanpur,

therefore, was directly instrumental in establishing Indian-language technology activities at all these centers.

Meanwhile, Putcha Narasimham—who had been the first to develop a universal key-boarding scheme at IIT Kanpur³²—joined the Computer Maintenance Corporation at Secunderabad and developed an Indian-language terminal;⁵² he also worked on Telugu (personal communication, Putcha Narasimham, Aug. 2008).

Other IIT Kanpur researchers who did not participate actively in our R&D on Indian-language technology but were influenced by our work include Om Vikas, who joined the government’s Department of Electronics after completing a PhD at IIT Kanpur. He persuaded the department to support and fund government-level activities, most notably of which was a national symposium organized on the “Linguistic Implications of Computer Based Information Systems.”⁵³ This symposium, a landmark in the history of Indian language computing, triggered numerous related research projects in India.

Rajeev Sangal, who joined IIT Kanpur’s faculty after completing a PhD in the US, became motivated to pursue research in Indian-language NLP. Vineet Chaitanya, whose PhD at IIT Kanpur was in control systems, joined the Birla Institute of Technology and Science at Pilani and taught Sanskrit at IIT Kanpur in the early 1980s. In those days, we had received a number of Acorn Computers’ BBC microcomputer boards for teaching and training purposes. Chaitanya, who used those boards to teach Sanskrit, worked with Sangal in NLP and developed the Anusaraka project for machine translation.⁵⁴ Later, Sangal moved to IIIT Hyderabad and established research programs in Indian-language technology. T.V. Prabhakar, another researcher, developed Indian-language content and created the Gita supersite (<http://www.gitasupersite.iitk.ac.in>). Three other individuals, who are products of IIT Kanpur, deserve mention: Pushpak Bhattacharya joined IIT Mumbai and continues to work in NLP; B.B. Chaudhuri joined ISI Kolkata and started working on OCR for Devanagari and Bangala; and Harish Karnick, who works on Indian language speech and data mining.

Scripts: Basic design methodology

The Integrated Devanagari Computer (IDC), as I will explain, was developed on the concepts highlighted in this section. I spearheaded the IDC team effort in the

mid-1970s; we developed the standards for it in cooperation with the government of India's Department of Electronics (DOE; now the Department of Information Technology [DIT]). By 1978, the IDC proof-of-concept was ready.^{38,55,56} In 1983, the Indian government sponsored a project for us to develop a Devanagari computer based on these concepts. This was completed in a record time of only eight months.³⁵ We presented most of the major research results at the 1978 Linguistic Implications of Computer-based Information Systems symposium and later published the developments carried out through mid-1984.⁵⁷

While seeking solutions in the early 1980s to the problem of enabling computers to work with Indic scripts, we concentrated on developing the technology indigenously. All of us at IIT Kanpur firmly believed that adapting western equipment and devices designed to deal with roman script would lead to inferior solutions: the Indic scripts formed an entirely separate class and were unique compared to their roman counterparts. Following were our major design considerations:²⁵

- The methodology should be adaptable to almost all Indian scripts and languages; that is, with minor modifications it should be possible to switch to other scripts and languages. This means that the methodology should base itself on the common properties of the scripts and languages.
- The design methodology should assimilate requirements from different application areas and present a unified approach such that, as far as possible, no major modification would be required while switching from one application to another.
- The system should be modularized to the maximum possible extent. It should be possible to configure the system modules appropriately to suit different applications. For software modules, the language-dependent and language-independent parts should be separately modularized; similarly, the device-dependent parts should be kept in a separate module. Portability is also desirable for software modules.

However, meeting these considerations was not easy. Several constraints influenced our design, including these:

- Developments in technology—new microprocessors, new LSI and VLSI

chips—continued to flow from abroad. Therefore, any design exploiting the latest technology had to follow those standards and constraints. This was also true for all imported systems software.

- English continued to be the effective link language in the country. Therefore, any Indian-language machine had to also provide facilities for roman script.
- All existing machines were designed with I/O capability only in roman script for which large investments had been made. An Indian-language machine could best be introduced by their adaptation or through add-on modules.

Some of the major characteristics of the Indic scripts our design considered that led to a unified indigenous approach were these:

- All Indic scripts have similar concepts of the full and the pure consonants, and of the vowels and the vowel modifier symbols (*matra*). Their order and categorization are based on the same articulatory mechanism. They differ in number of consonants and number of vowels, some providing finer-grained articulation and some remaining at a coarser level. This observation led us to define a superset of all Indic script symbols. This was referred to as the “enhanced Devanagari script” (*Parivardhit Devanagari Lipi*).
- In all Indic scripts, each consonant has a corresponding pure consonant. Similarly, each vowel has a corresponding modifier (*matra*). Thus it was possible to reduce the entire set of symbols by taking this correspondence into consideration.
- For all Indic scripts, writers use a similar logical order of symbols, which is what children are taught while learning how to write. This order can differ from the visual order (which is graphic oriented) in that the final script composition may not show the symbols in the same order. This led us to develop a uniform keyboarding method for inputting.
- To facilitate the process of editing the individual symbols and the word processing, the script data must be stored in a linearized form, not in the font codes or composed form codes. This observation led us to design the Indian Script Standard Code for Information Interchange (ISSCII) code.
- The manner in which the individual symbols are joined together to form a word

A Journey from Indian Scripts Processing to Indian Language Processing

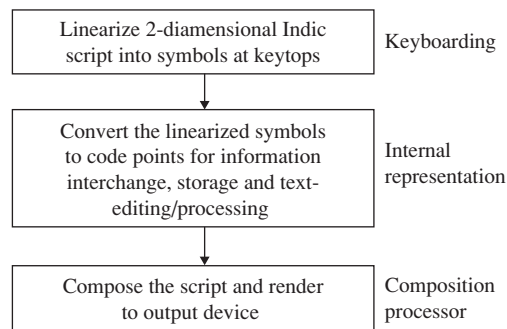


Figure 12. Three basic stages for enabling Indic scripts on computers.

differs from one Indic script to another. This led us to delineate the composition process of the script for the purposes of display and printing from the rest.

These observations led us to split the design process for enabling computers to handle Indic scripts into three basic stages:

- the keyboard layout and keyboarding stage;
- representation of the text for internal storage and text editing; and
- the stage for rendering the script on the output device.

This is diagrammatically shown in Figure 12.

Scripts: Keyboard considerations

Usually a syllable in an Indic script is a two-dimensional composition of the constituent symbols. Therefore, an unambiguous way must be devised to convert it into a linear string of the symbols. This is what we call the “keyboarding problem.”

Keyboard layout design involves the problem of optimally placing all the script’s symbols on the key tops. The placement is done to minimize the number of keystrokes, and to balance the load on the user’s fingers.

The two issues—minimum number of keystrokes and the finger load-balancing—are related. As mentioned earlier, all the pure consonants can be derived from their corresponding full consonants, and each vowel has a corresponding *matra* symbol. Thus, our symbol list could have only the full consonants, the vowels, and the diacritical marks—we could derive all other symbols from this set. There are other alternatives to deciding the symbol list for keyboarding as well.^{25,58-62}

The frequency of occurrence of various symbols plays a dominant role in deciding

the set of symbols for keyboarding. For Hindi, the frequency⁶³ of occurrence of the vowels is about 4.11%; for the *matra* symbols, it is about 35.22%. For the standalone consonants (i.e., the consonant without an attached *matra*), the frequency of occurrence is about 23.87%; for the consonants with the *matra*, it is about 31.84%; for the pure consonants, it is only about 4.94%. From an optimality viewpoint, then, it’s obvious that the pure consonants should not be included on the key tops but should be derived from the full consonants. Note that two keystrokes are needed for this derivation.

Now with the exclusion of the pure consonants (half characters) from the list of symbols, it was possible for us to accommodate all other symbols on the standard QWERTY keyboard layout. For the actual physical layout, we debated, for a considerable amount of time, several proposals. The major debate was whether the layout should be consistent with the logical grouping of the characters, or if instead it should be based on the finger load-balancing determined by the frequency of various symbols’ occurrence. Ultimately, we favored placement according to the logical grouping of symbols—primarily because with electronic touch typing, finger load-balancing had lost its significance. Moreover, the logical grouping would be easy to remember since that is how the script is introduced to learners.

Because the aspirates occur less frequently than the non-aspirates, we kept these with the shift key. Similarly, we kept the *matra* symbols in the normal position and the corresponding vowel in the shift position. Finally, the project team agreed on a universal layout applicable to all the Indic scripts with the symbols of the enhanced Devanagari script (see Figure 13). We named this the InScript keyboard, and it was standardized by the Bureau of Indian Standards (IS 13194:1991). Because space was available to add more symbols, some of the frequent conjuncts were also assigned a place for efficiency; the assignment can differ from one script to another.

The decision on the keyboarding *method* was more vexing. The major debate was whether it should be graphic-oriented (i.e., in visual order, with symbols entered in the same order as they appear on the final output) or in phonetic order (determined by how the word being entered is pronounced). I proposed a third variation in phonetic order—Machine Oriented Devanagari Script,



Figure 13. The InScript keyboard layout with Devanagari symbols. Note that the InScript keyboard layout is an overlay over the QWERTY layout, which lets one easily switch from roman to Indic script and vice versa.

where only the consonants and the vowels were assumed. In MODS, a link operator (Θ) denotes the composition. Figure 14 shows a few examples to illustrate the difference in the three keyboarding schemes.

Hindi typists were accustomed to using the visual order, so there was strong resistance to the phonetic order on a keyboard. The visual order of script symbols, however, has several drawbacks. First, it is script-dependent: the keyboarding sequence differs for different scripts, which effectively loses the universality of the keyboarding scheme we had been seeking. The more problematic situation results when the visual order sequence does not find the corresponding anticipated symbols on the key top (such as \wedge or ^ in Devanagari). Such graphic symbols must be mapped onto a sequence of symbols on the key tops to obtain the required grapheme. Whereas these symbols representing a grapheme are available on the typewriter key top, inserting such symbols on the InScript key tops was another step toward losing a universal solution.

Conversely, however, the phonetic order is the order in which words are spoken, and it does not depend on the script. More

important, children learn a script by the phonetic order; further, the phonetic order provides an easy way for editing and making corrections on a keyboard. Phonetic order makes it easier to implement the script composition grammar and inhibit illegal/nonsensical inputs such as putting two *matra* symbols on a character.

The MODS scheme was a variation on phonetic order and called for the consonants and vowels to be used without *matra* symbols. Because the keyboard layout design had both the vowels and the *matra* symbols, however, we did not pursue this approach.

Ultimately, we decided to use the phonetic keyboarding order as the standard keyboarding scheme. There is still wide resistance to its acceptance, however. Some users, influenced by the roman juxtaposition order, cannot accept that a *matra* symbol like ि, which actually appears *before* the character on output, should be typed *after* the character on a keyboarding scheme designed in the phonetic order. These users fail to understand that phonetically the vowel sound associated with a consonant always appears after the consonant sound. As a consequence, many commercial software products, such as C-DAC's multilingual word processing product i-Leap, give users the option to use the visual order of character entry: through firmware, the input is converted to the phonetic order for further processing.

Coding considerations

The coding scheme we developed had to address the needs of information interchange, storage, and processing.

Coding in terms of conjuncts

In converting Indic script symbols to code points, the simplest coding method is to use the set of conjuncts, or the composite characters (which could number in the thousands),¹⁸ as the atomic code points. Another

आदित्य \rightarrow आ ि द त ्य (visual order)
 आ द ि त ्य (phonetic order)
 आ द Θ इ त Θ य (MODS)

निष्क्रिय \rightarrow ि न ि ष ् क ्र य (visual order)
 न ि ष ् क ्र ि य (phonetic order)
 न Θ इ ष Θ क Θ र Θ इ य (MODS)

कुर्सियों \rightarrow क ु रि स ्र य ो ं (visual order)
 क ु र्स ि य ो ं (phonetic order)
 क Θ उ र Θ स Θ इ य Θ ओ ं (MODS)

Figure 14. Examples illustrating keyboarding schemes.

A Journey from Indian Scripts Processing to Indian Language Processing

method would be to use a font-based coding. Font-based coding was used by almost all vernacular newspapers in the early days of electronic typesetting. The readers of these newspapers have to download their specific fonts for reading the e-paper. Such a situation, however, is good only for the output environment and is of no use for the tasks of text editing and word processing because the logical information of the conjunct or composite character compositions is lost.

Phonetic encoding

Three possibilities for phonetic encoding exist.

Full consonants and vowels. The set of the full consonants, vowels, diacritical marks, and a link symbol operator form the vocabulary for internal representation. The operands of the link symbol operator are converted to their corresponding pure consonant or *matra* symbol forms. For Hindi, the storage requirement is roughly 140.16 bytes per 100 basic symbols.

Pure consonants and vowels. The set of the pure consonants, the vowels, and the diacritical marks form the vocabulary for internal representation. There is no link operator. The full consonants are derived from the corresponding pure consonants by attaching the vowel ॐ. Recall that the pure consonant represents muting of the inherent ॐ sound. If a pure consonant is followed by a vowel, the corresponding *matra* symbol is attached. If it is followed by another pure consonant, a conjunct is formed. For Hindi, the storage requirement for this scheme is roughly 123.87 bytes per 100 basic symbols.

Full consonants, vowels, and matra. The set of all the full consonants, the vowels as well as their corresponding *matra* symbols, the diacritical marks, and the *halant* sign form the vocabulary for internal representation. The *halant* sign converts the preceding full consonant to the corresponding pure consonant. As the *matra* symbols occur more frequently, their redundancy helps in reducing the storage requirement. For Hindi, the storage requirement under this scheme is roughly 104.94 bytes per 100 basic symbols.

Coding using roman characters

Roman characters, with the international phonetic symbols like those dictionaries use to denote pronunciation, have been extensively used by linguists and literary scholars for writing Indian-language texts. In 1984, a

अ आ इ ई उ ऊ ऋ ए ऐ ओ औ

ा ि िी ु ू ृ े ै ो ौ ं ः ॅ ॱ ॲ

a A i I u U q e E o O M H V z Z

क ख ग घ ङ च छ ज झ ञ ट ठ ड ढ ण त थ द ध न

k K g G f c C j J F t T d D N w W x X n

प फ ब भ म य र ल व स श ष ह

p P b B m y r l v s S R h

Figure 15. Devanagari to IITK-Roman code.

roman two-character code with the most common interpretation (based on frequency) was developed for Hindi.⁴⁰ Later, ITRANS (short for *Indian language transliteration*; <http://en.wikipedia.org/wiki/ITRANS>) and INSROT⁶⁴ (short for *Indian Script Roman Transliteration*) have been standardized along similar lines. These use lowercase characters only, which facilitates searching using conventional search engines.

Yet another roman character coding scheme known as IITK-Roman was devised in the mid-1980s that uses both upper- and lowercase roman characters. In this essentially pure-consonant-based coding method, a single roman character code is assigned to each of the vowel and consonant symbols. Figure 15 shows the IITK-Roman assignment table. If a consonant character is followed by a vowel character, the corresponding *matra* symbol is attached to it. If, however, it is followed by another consonant, it forms the corresponding conjunct. The IITK-Roman code provides a convenient way of inputting Hindi using a conventional roman keyboard; text editing and word processing tasks can be easily done with this code. The major disadvantage is that a conventional search engine cannot be used because of the uppercase letters; nonetheless, this coding scheme is still very popular.

Code standardization

Soon after the 1978 symposium, India's Department of Electronics constituted a standardization committee, of which I was a member, for designing codes for the Indic scripts similar to ASCII. After much deliberation with the experts of different Indic scripts, in 1982 we came up with the first version of a 7-bit code, called ISSCII-7 (Indian Scripts Standard Code for Information Interchange).²⁵ In 1983 the first version of the 8-bit code (ISSCII-8) was released.⁶⁵ It was

difficult to incorporate everything that different Indic script users demanded, and it took us quite some time to make users appreciate the concept of universality and the need for delineating the script composition phase from that of internal coding.

Another major difference of opinion, between users and the standardization committee, was in the collating order. Therefore, several revisions were done and in 1988 the Department of Electronics published the first official version.⁶⁶ By this time, one S in ISSCII had been dropped and the acronym became ISCII. A further modification was made in 1991, and the Bureau of Indian Standards accepted ISCII-8 as the national standard (IS 13194:1991). The design of ISCII-8 was totally an indigenous effort, addressing India's needs with multiple scripts: in that sense, there was no correlation to what was then being designed by the International Organization for Standardization (ISO) and the newly formed Unicode consortium.⁶⁷

At the international level, ISO came up with a draft framework for a Universal Coded Character Set (ISO/IEC FIS 10646) in 1990. At the same time, the major multinational IT companies formed a consortium for devising character codes to represent all the world's scripts. In particular, the consortium was concerned for business penetration reasons to be able to handle the scripts of Asian countries where English was not used for internal communication. The consortium developed a 16-bit code called Unicode (<http://unicode.org/>) wherein distinct code points were assigned to each character with direct mapping to its rendering on the output device. For the Indic scripts, the Unicode consortium adopted the 1988 ISCII-8 standard version as its base for the pages related to the Indic scripts (for an example, see <http://www.unicode.org/charts/PDF/U0980.pdf> onward). As a result of philosophical differences between the ISCII-8 and Unicode designs, several errors crept into the Unicode—none of the Indian companies or research institutions was a member of the Unicode consortium at that time to address our concerns. Today, India's Department of Information Technology is a consortium member and has made suggestions for making the appropriate changes.

ISSCII-7, ISSCII-8, and Unicode

To help illustrate the three different coding standards' approaches in representing

Devanagari, let us examine the salient features of each.

7-bit internal representation

The 7-bit code has 128 code positions available. The first two columns are reserved for the control characters. If we consider all the special characters and the numerals, we are left with only 64 code positions for assigning Devanagari symbols. In the ISSCII-7 design of 1982, we decided to include the full consonants and the *matra* symbols. The vowels were obtained by attaching the corresponding *matra* symbols to one vowel, ॐ, which was given a code point. The pure consonants were derived using the *halant* symbol. Figure 16 shows the code point outlay for ISSCII-7. The code does give the right collation order and is applicable to all the Indic scripts. It worked in all environments where 7-bit ASCII was being used, so the standard 7-bit communication interface could be directly used. However, the major disadvantage was that it did not provide mixing with the roman script code.

8-bit internal representation

In designing the 8-bit ISSCII code table in 1983, we made the first half of the table the same as for the 7-bit ASCII and used only the latter half of the code space for Devanagari symbol assignment. For the code points of numerals, punctuation marks, and special symbols, we used the code points of the corresponding ASCII code. We left the first two columns of the Devanagari portion intact for the control characters, so that the rest of the 96 code positions were available for placement of the Devanagari symbols. The code used all the full consonants, the vowels, and the *matra* symbols. The “link” symbol ॐ (equivalent to the *halant*) denoted formation of the conjuncts. The *halant* was a printable symbol whereas ॐ was a nonprintable operator symbol. Figure 17 shows the 1983 ISSCII-8 code assignment table. As Figure 17 shows, a special Devanagari space symbol has been provided to aid in the right sorting order. The spare available code points have been used to place some of the common conjuncts (user-defined codes) to reduce the text storage requirement. Thus the 8-bit code provided all the desirable features and was universal, provided that the codes for the conjuncts were not used. However, the ISSCII-8 code was not suitable for those environments where the eighth bit of the byte was being used for some other process-specific

A Journey from Indian Scripts Processing to Indian Language Processing

b7.....				0	0	0	0	1	1	1	1	
b6.....				0	0	1	1	0	0	1	1	
b5.....				0	1	0	1	0	1	0	1	
b4	b3	b2	b1	0	1	2	3	4	5	6	7	
0	0	0	0	0	NUL	DEL	SP1	0		ड	ल	<ऊ>
0	0	0	1	1	SOH	DC1	<अ>	1	ॐ	ढ	ळ	<ऋ>
0	0	1	0	2	STX	DC2	<अ>	2	ॐ	ण	ळ	<अ>
0	0	1	1	3	ETX	DC3	<अ>	3	अ	त	व	<अ>
0	1	0	0	4	EOT	DC4	"	4	क	थ	श	<ए>
0	1	0	1	5	ENO	NAK	%	5	ख	द	ष	<ऐ>
0	1	1	0	6	ECK	SYN	×	6	ग	घ	स	<अ>
0	1	1	1	7	BEL	ETB	!	7	घ	न	ह	<अ>
1	0	0	0	8	BS	CAN	(8	ङ	प)	<अ>
1	0	0	1	9	HT	EM)	9	च	फ	<अ>	<अ>
1	0	1	0	A	LF	SUB	*					
1	0	1	1	B	VT	ESC	+	÷	छ	ब	<अ>	<अ>
1	1	0	0	C	FF	FS	,	<	झ	भ	<अ>	0
1	1	0	1	D	CR	GS	—	=	ञ	य	<इ>	1
1	1	1	0	E	SO	RS	.	>	ट	र	<ई>	2
1	1	1	1	F	SI	US	SP2	?	ठ	ड	<उ>	DEL

SP1 & SP2 are codes for space. SP2 is used for Devanagari Text processing. <v> is the corresponding matra for a vowel. ॐ is used in Malayalam, and ॐ in Tamil and Malayalam.

Figure 16. ISSCII-7 (1982) code assignment table for enhanced Devanagari. Here the *matra* symbols are indicated by writing the corresponding vowel within angular brackets. "SP2" is the Devanagari space, which was introduced to maintain the right collation order.

applications (assuming that ASCII has no use for this bit).

This basic layout was later modified in 1988 and again in 1991; the Bureau of Indian Standards' 1991 ISCII-8 layout can be viewed

at <http://tdil.mit.gov.in/isciiapril03pdf>. The major modification to this was the deletion of the additional Devanagari-space code point. Further, the code points for numerals were added onto the Devanagari portion.

b7=0									b7=1								
	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	
0	NUL	DLE	SP	0	\	P	@	p			ॐ	अ	इ	उ	ए	ॐ	HALANT
1	SOH	OC ₁	!	1	A	Q	a	q			ॐ	ओ	अ	ब	ट	ॐ	
2	STX	DC ₂	"	2	B	R	b	r			ॐ	ए	ट	भ	ॐ	ॐ	LINK
3	ETX	DC ₃	#	3	C	S	c	s			ॐ	ऐ	ठ	म	ॐ	ॐ	
4	ECT	DC ₄	Rs	4	D	T	d	t			ॐ	ऑ	ड	य	ॐ	ॐ	
5	ENQ	NAK	%	5	E	U	e	u			ॐ	आ	इ	र	ॐ	ॐ	
6	ACK	SYN	&	6	F	V	f	v			ॐ	ओ	ढ	ॐ	ॐ	ॐ	
7	BEL	FTB	ॐ	7	G	W	g	w			ॐ	औ	ढ	ल	ॐ	ॐ	
8	BS	CAN	(8	H	X	h	x			ॐ	क	ण	ॐ	ॐ	ॐ	
9	HT	EN	1	9	1	Y	i	y			ॐ	ख	त	ॐ	ॐ	ॐ	
A	LF	SUB	*	:	J	Z	j	z			ॐ	उ	ग	थ	व	ॐ	
B	VT	ESC	+	;	K	[k	{			ॐ	घ	द	श	ॐ	ॐ	
C	FF	FS	,	<	L	\	l	!			ॐ	ड	ध	ष	ॐ	ॐ	
D	CR	GS	—	=	M]	m	}			ॐ	च	न	स	ॐ	ॐ	
E	SO	RS	.	>	N	f	n	~			ॐ	ॐ	ॐ	ॐ	ॐ	ॐ	
F	SI	US	/	?	O	←	o	DEL			ॐ	अ	प	ॐ	ॐ	ॐ	DEL

Figure 17. ISSCII-8 (1983) code assignment table for enhanced Devanagari.

The deletion of the Devanagari space symbol did affect the collation order for the words with some of the diacritical marks. Some of the standardization committee members argued that the universal acceptability of the sorting order was not possible across all Indic scripts. One additional pass on the word-processing part was required to ensure the right sorting order with this change. No place was provided for the code points corresponding to the frequent conjuncts to ensure applicability to all Indian scripts.

In the layout, it should be noted that the *nukta* symbol ण् is not a *matra* but a diacritic mark. When a *nukta* is attached to a consonant, it yields a derived consonant or another consonant. To preserve the sorting order, it was kept following the *matra* symbols and not with the other diacritic marks.

ISCII versus Unicode

The Unicode consortium adopted the 1988 version of ISCII-8 as the base for the 16-bit Unicode for allocating codes to different Indic scripts. Although the consortium tried to preserve the basic characteristics of ISCII coding, ISCII differed significantly from Unicode. The ISCII design exploited commonality of the Indic scripts and allocated code points for the superset of the enhanced Devanagari symbols. The graphical or the compositional aspect of individual characters and the script is not a consideration in its design. Therefore, ISCII applies to all Indic scripts, which makes transliteration among Indic scripts a straightforward task. Unicode, however, is more oriented toward facilitating script composition. It does not reflect in any way what could be common features of a group of scripts that could be dealt with uniformly for text processing. Unicode assigned a separate page for each one of the scripts. Thus, as one perceives more compositional features in the scripts, the demand for including more and more symbols continues. In ISCII, however, the symbols relate to the articulatory aspect of the associated speech, and it remains constant as long as all the articulatory aspects have been considered.

Rendering of Indic scripts

Because Indic scripts vary significantly in terms of how they are composed, the IIT Kanpur project team envisioned a separate composition processor³⁸ for every Indic script. This processor, when fed with an ISCII string, would yield the sequence of composite

characters as desired in the output text. We envisioned this rendering to be dynamic—that is, as the input string is read from left to right, the composition processor must start rendering and modify the earlier rendered part if needed. In other words, the composition processor should not wait for the entire input string before rendering. It was up to the composition processor to choose the appropriate fonts, their features, and the conjuncts, and to provide a variety of users' choices based on the nature of the output device. Separating the rendering stage from the rest of the composition process was a well-regarded decision.

Output was to a dot matrix plotter of varying resolution. A basic resolution of 50 to 70 dots per inch had a matrix size of approximately 15×8 (for Devanagari). Minimum readability required 8 dots for the height of the main character, 3 dots for the lower symbol, and 4 dots for the upper symbol. Medium-to-high quality script could be generated using a dot resolution of 100 to 200 dots per inch with a matrix size of 24×12 or higher.

IDC and GIST: Evolution

The concepts and methodology explained thus far for developing linguistic interfaces were simulated at IIT Kanpur, where we built prototypes during the years 1976 and 1980.^{38,55,56} In 1983, India's Department of Electronics sponsored IIT Kanpur to design and develop the Integrated Devanagari Computer terminal, a project for which I served as chief investigator.³⁵ We developed the IDC using the Intel 8086 processor, with multi-tasking firmware. The Devanagari keyboard was designed in hardware that directly generated ISCII code. The Devanagari character fonts were stored in ROM along with their relative positioning information in the composition frame. To speed up the composition process, the information was stored in multiple partitions. Some of the frequently occurring composite characters were precomposed and stored in ROM. We programmed the composition processor engine to interpret the input ISCII-8 code dynamically and provide the display with the composed sequence of the composite character.

The CRT display dynamically displayed character changes as the input progressed. Display flicker resulted from the script composition time, which affected the refresh time. We reduced the ROM fetch time by logically partitioning the ROM space, by

Related Work and Developments

In 1988, the Graphics and Indian Script Terminal (GIST) terminal evolved into a GIST card that was pluggable into an IBM PC. This allowed all the existing character-oriented software packages to be used with all the Indic scripts. In 1990, the Centre for Development of Advanced Computing (C-DAC) designed an 84-pin PLCC ASIC for GIST called the GIST-9000. It provided an interface for Motorola's 68008 microprocessor with 256 Kbytes of DRAM and an I/O-mapped interface for the IBM PC bus.

In 1991, C-DAC designed a GIST print spooler that could offload the time-consuming printing task for Indic scripts from the host processor. In 1998, C-DAC developed a GIST-II card and, in 2001, designed a PCI GIST card. During 1990–1992, C-DAC also developed keyboard standards for all the Perso-Arabic scripts and phonetic standards for Thai, Sinhalese, Bhutanese, and Tibetan scripts. The Indian script font code (ISFOC) standards were also developed for all Brahmi-based Indic scripts. During the 1997–2002 period, C-DAC commercially released multilingual word processing software, called LEAP, catering to all Indic scripts.

During the years 1981–1985, the CMC company in Secunderabad, under Putcha Narasimham's leadership, prepared a design document on Telugu¹ and designed LIPI, a multilingual computer system featuring word processing with proportional spacing, and high-quality printing for a large number of Indic scripts (personal communication, Putcha Narasimham, Aug. 2008). This machine was made commercially available. Although LIPI's design was based on a universal coding method, it did not dynamically display composition.

During the years 1978–1980, NCST Mumbai, under the leadership of S.P. Mudur, developed a design document for Devanagari.² This was based on an analysis of graphic strokes and used a visual order for keyboarding.

Between 1980 and 1983, the Birla Institute of Technology and Science in Pilani, under the leadership of Praveen Dhyani and Aditya Mathur, developed a multilingual computer system³ under the government of India's Department of Electronics—sponsored project. It could display text in Devanagari and print text in several other Indic scripts. The computer that Dhyani and Mathur used was a Spectrum/3 from DCM Data Products, connected to an ADM 3A CRT upgraded with a graphics card. This system was called Siddhartha. At the same time, the DCM Data Products company manufacturing computer systems in India also named its computer catering to Hindi word processing as Siddhartha, but the two machines had no correlation (personal communication, Aditya Mathur, Sept. 2008) except that both were based on the Spectrum/3.

Indian Institute of Technology (IIT) Chennai (earlier, IIT Madras), under the leadership of Kalyana Krishnan, developed a method for character generation using cubic splines in 1983 (<http://acharya.iitm.ac.in/history.php>).

In 1988, the first attempt at computing with Indian scripts was made by designing and implementing an interpreter for a Basic-like language written in Tamil or Telugu. The characters were not displayed through fonts but drawn on the screen using the curves. The 16-bit character representation made it possible to quickly identify the strokes needed to generate the character. In 1998, the first version of the fonts-based editor was developed for Microsoft Windows 95, and in 1999, IIT Chennai demonstrated a text-to-speech system and Braille output from Indian language documents.

These works are only a few of the many projects that have been undertaken. Numerous others have involved type and composition design,^{4–15} font design,^{16,17} transliteration schemes for Indic scripts,^{18–20} and on speech processing.^{21–23} Besides these, some other early works on Urdu,²⁴ Farsi,²⁵ and Sinhala²⁶ might be of interest.

References and notes

1. P.V.H.M.L. Narasimham et al., *Design Information Report on Text Composition in Telugu*, Computer Maintenance Corp., Secunderabad, 1981.
2. S.P. Mudur et al., *Design Information Report on Text Composition in Devanagari*, Nat'l Centre for Software Development and Computing Technology, Tata Inst. of Fundamental Research, Bombay, 1980.
3. A. Mathur and P. Dhyani, eds., *Design and Development of a Devanagari Based Computer System*, tech. report, Project Report III, Birla Inst. of Technology and Science, Pilani, Apr. 1983. (Contributors: S. Anand, R. Bagai, V. Dev, P. Dhyani, D. Kumar, and A. Mathur.)
4. A.V. Sagar and S. Chadda, "Composite Character Formation in Indian Scripts with a Small Set of Working Patterns—A PostScript Implementation," *Proc. Workshop Computer Processing of Asian Languages*, Asian Inst. of Technology, Bangkok, Thailand (hereafter, AIT), 1989, pp. 160–167.
5. F.A.V. Donani, "Constructions and Graphic Display of Gujarati Text," master's thesis, Dept. of Electrical Eng. and Computer Science, Massachusetts Inst. of Technology, 1977.
6. H. Ganesh et al., *Design Information Report on Text Composition in Malayalam*, Research Inst. for Newspaper Development (RIND), Madras, 1981.
7. J.B. Millar and W.W. Glover, "Synthesis of the Devanagari Orthography," *Int'l J. Man-Machine Studies*, vol. 14, 1981, pp. 423–435.
8. J.G. Krishnayya, *Stroke Analysis of Devanagari Characters*, quarterly progress report no. 69, Massachusetts Inst. of Technology, 1963, pp. 232–237.

9. M. Vannan, "Structured Approach for the Display of Indian Scripts," *Proc. Workshop Computer Processing of Asian Languages*, AIT, 1989, pp. 147-153.
10. P.K. Ghosh, *An Approach to Type Design and Text Composition in Indian Scripts*, report no. STAN-CS-83-965, Stanford Univ., 1983.
11. S.P. Mudur et al., "An Integrated Software Environment for Localization," *Int'l Conf. Computer and Communication (ICCC 02)*, Int'l Council for Computer Communication, 2002, pp. 828-842.
12. S.P. Mudur et al., "An Architecture for the Shaping of Indic Texts," *Computers & Graphics*, vol. 23, no. 1, 1999, pp. 7-24.
13. T.K. Bhatia, "The Problems of Programming Devanagari Script on PLATO IV and a Proposal for a Revised Hindi Typewriter," *Language, Literature and Society: Occasional Papers*, no. 1, Center for Southeast Asian Studies, Northern Illinois Univ., 1974, pp. 52-64.
14. T. Mukherjee, "The Design of Indian Printing Types: Some Considerations for the Future," *Inside Outside*, 1978, vol. 2,3, pp. 90-93.
15. T.N.V. Reddy, *Design Information Report on Text Composition in Kannada*, RIND, Madras, 1981.
16. A.V. Sagar and G. Muralidhar, "CFONTS—A Font Design System," *Proc. Workshop Computer Processing of Asian Languages*, AIT, 1989, pp. 137-146.
17. C. Muthuvel, N. Alwar, and S. Raman, "A Font Generator for Indian Languages," *Proc. Workshop on Computer Processing of Asian Languages*, AIT, 1989, pp. 154-159.
18. C. Chandrasekaran and S. Chadda, "Transliteration of Persons' Names from English to Hindi," *Proc. Workshop on Computer Processing of Asian Languages*, AIT, 1989, pp. 261-267.
19. E.V. Krishnamurthy, "Automatic Phonetic Transcription of Tamil in Roman Script," *Proc. Indian Academy of Sciences*, Indian Academy of Sciences, 1977, vol. 86, pp. 503-512.
20. S. Goel et al., "LIPYANTARAN—A Computer Aided Transliteration System for English to Devanagari," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 319-320.
21. B. Yegnanarayana et al., "A Continuous Speech Recognition System for Indian Languages," *Proc. Workshop on Computer Processing of Asian Languages*, AIT, 1989, pp. 347-356.
22. D.D. Majumder, A.K. Dutta, and N.R. Ganguli, "Some Studies on the Acoustic Features of Human Speech in Relation to Hindi Speech Sounds," *Indian J. Physics*, vol. 47, no. 10, 1973, pp. 598-613.
23. P.V.S. Rao and N. Bondale, "Syntax and Semantics of Speech Processing," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 219-232.
24. K.S. Mustafa, "On Computerization of Urdu: Problems and Proposals," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 299-302.
25. B. Parhami and F. Mavaddat, "Computers and the Farsi Language," *Proc. IFIP Congress*, Toronto/North Holland, 1977, pp. 673-676.
26. A.K. Kumarsena, "Progress in Sinhala Computing," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 59-73.

pipelining through the buffered output registers, and by skipping over the dark spots (0's representing nonilluminated points). Because the visual screen was a window of the physical page, we provided facilities for panning and zooming. We incorporated built-in intelligence to prevent illegal composition such as attaching two *matra* symbols to the same character.

The backspace and other text editing operations worked on the internal ISCII-8 code and were dynamically reflected on the screen. Automatic transliteration from one Indian script to another was also made possible because the text was stored internally in the ISCII-8 format.

We later extended the IDC project to work on Intel's 32-bit 68000 microprocessor.

The project's new name was the Graphics and Indian Script Terminal (GIST). A number of companies bought this technology for manufacturing multilingual computer terminals. India's Centre for Development of Advanced Computing adapted the GIST technology for the design of an ASIC chip.⁵⁰ Most of the current commercially available software catering to Indic scripts has borrowed ideas from the GIST technology.

The IDC and GIST technology, as I have explained, represented a major breakthrough in solving the complex problem of design of the man-machine linguistic interface for Indian languages. There have been a number of other efforts to develop Indian-language computers that are noteworthy; see the "Related Work and Developments" sidebar.

A Journey from Indian Scripts Processing to Indian Language Processing

Word processing; transliteration

Text editing and word processing of Indic scripts is more complex than for their roman counterparts. Even a function as basic as the backspace is not simple: to use it requires, first, identification of the unit to be erased. There are a few alternative ways to implement this, which offer different levels of user friendliness. The unit to be erased could be a composite character (which is easiest to implement), in which case the minor errors of the *matra* symbol require the user to reenter the whole composite character. Alternatively, a constituent symbol can be erased in the order in which it was entered, in which case the level of user friendliness depends on the keyboarding method used. Yet another alternative is to erase the symbol in the order it was displayed on the screen, in which case the order is dictated by the script composition process and is not easily anticipated by the user. The ISCII-based keyboarding enforced a definite same canonical order both for entry as well as storage, thus the backspace simply deleted the preceding symbols in the order of the code entered. In this case, the composite character composition is dynamically modified, reflecting the deletions.

The tasks of searching and replacing, sorting, and finding the word prefixes and suffixes were easily performed on the ISCII code. The routines for morphological analysis⁴⁹ of the words, and the routines for generating the word aliases as needed in spell checking and correction,⁴¹ required prefix and suffix identification; otherwise, these operations would have been difficult to implement.

Another important deviation from the roman alphabet that affected edit routines was the unequal width of the Indian characters and their compositions. Any deletion, addition, or replacement of a word caused a change in the line width that was not related to the number of symbols involved but was related to their nature. Any editing software had to store the width information of the composite characters. In a string model, such as ours, the lines of a page from a single string and edit could cause the text lines to be readjusted. This was achieved by introducing a soft carriage return for the end of every logical line and a hard carriage return for an actual change of line. The right justification was achieved by our introducing the Indian Language Space (a separate code in ISSCII-8) rather than the typical roman

space. The Indian Language Space had a width equaling the greatest common divisor of all possible widths of composite characters. This served the dual purpose of yielding the right sorting order and for aligning the text.

In screen-oriented editing, the variable width caused another problem. The cursor movements had to vary according to the width of the composite character or word, and the cursor width itself had to vary to indicate the boundary of the word or composite character. The cursor movement in the vertical direction had to similarly adjust to the word boundaries.

The one-to-one correspondence of each Indic script to the symbols of the enhanced Devanagari script with ISCII-8 phonetic encoding provided a natural and easy transliteration method among the Indic scripts, requiring only a few exception rules and a switching of the script composition processor. Figure 18 shows a block schematic of the transliteration schema with a few examples.

OCR for Indic scripts

Researchers have investigated OCR for a number of Indian scripts: Devanagari, Tamil, Telugu, Bengali, and Kannada.^{34,68-71} However, most of this research has been confined to the identification of isolated characters rather than the script. Some systems used a statistical method; others were syntactic and/or heuristic-based. Unlike simple juxtaposition in roman script, the Indic scripts are a composition of the constituent symbols in two dimensions. This meant that researchers first segmented an Indian script word into its composite characters. Each composite character was then decomposed into the constituent symbols or the strokes that were finally recognized.

Further, the Indic scripts posed difficulties to the researchers because of the natural breaks that occurred in a character. There were also natural joins or fusions of the constituent symbols. Additionally, the *matra* symbols often were not attached at precisely the right position and users had to resolve them using context. These difficulties were more often encountered in the handwritten script.

In 1973, I elaborated the techniques and strategies for segmentation, decomposition, and recognition of Devanagari script for the first time.³⁴ Follow-up investigations later focused on segmentation of touching or fused characters, and on contextual

processing for error corrections.⁷¹ The contextual postprocessing stage used script composition information in addition to the individual character confusions as in case of roman OCR.^{26,72}

Indian language processing

The need for providing technological support for NLP tasks has been enumerated elsewhere.^{73,74} Although computers have long been used for studying characteristics of Indian languages, the text form used was only a romanized version. The work on building natural-language systems in Indian languages began only in the late 1970s with some preliminary works on designing a domain-specific question-answering system.⁷⁵ The user-friendly solution to Indic script interfacing with the computers in the 1980s triggered a number of applications in typesetting, printing, publishing, hyphenation, document preparation, spell checkers, and so on.^{25,40,41} Our emphasis then moved to computers as a tool for breaking the linguistic barriers and for language learning.

Work on computer-assisted translation among Indian languages started in the early 1980s.^{25,76} In 1984, I outlined an interlingual approach using Sanskrit as the intermediate base language (see Figure 19).²⁵ A related project was started in 1985 at IIT Kanpur by Rajeev Sangal, but was not pursued due to the system's anticipated complexity. The group instead used an easier, more direct approach by substituting the word-groups of the source language with the corresponding word-groups of the target language. As the Indian languages are structurally similar, this approach did yield an output that could be called a working translation for the language pair. The correctness of the translation very much depended on the closeness of the two languages under consideration. This approach, known as Anusarak,⁵⁴ was described as a language accessor system.

In 1989, the first regional workshop on Computer Processing of Asian Languages⁷⁷ was organized at the Asian Institute of Technology, Bangkok, followed by the second workshop at IIT Kanpur in 1992.⁷⁸ Both of these workshops, supported by UNESCO, provided good forums for exchanging results and made recommendations to UNESCO for future work pertaining to Asian languages.⁷⁹

Between 1990 and 1992, I designed a machine-aided translation (MAT) system—AnglaBharati—for translation from English to Indian.^{45,51} In the AnglaBharati technology,

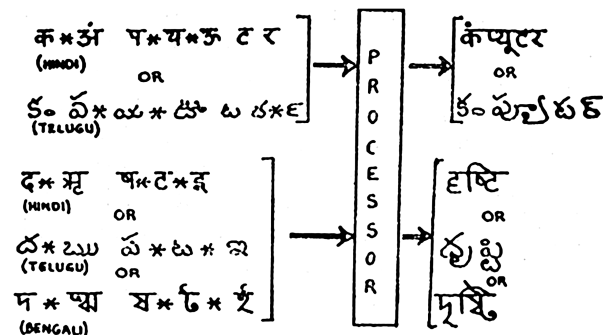
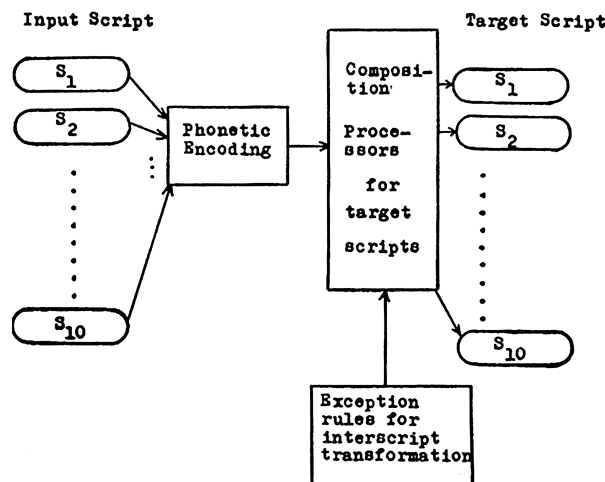


Figure 18. Transliteration using ISCII-8 phonetic encoding and script composition processors.

the input English sentence was transformed into a pseudo-interlingual structure called PLIL (Pseudo Lingua for Indian Languages) using a CFG-like pattern-directed rule base. The PLIL structure had a word order that was

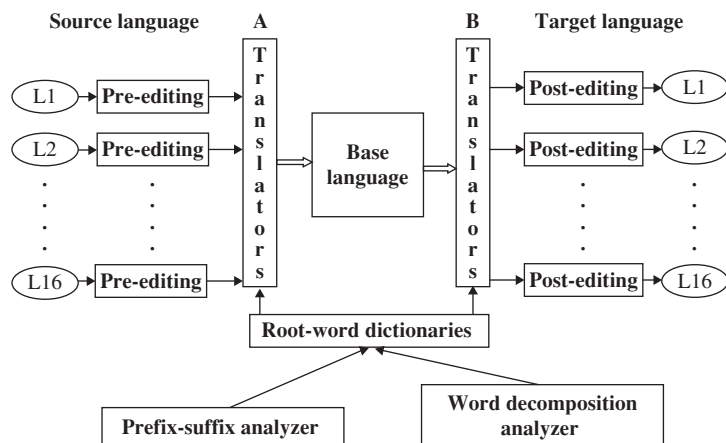


Figure 19. Machine translation among Indian languages using interlingual approach (1984²⁵).

A Journey from Indian Scripts Processing to Indian Language Processing

applicable to a group of Indian languages and carried all the syntactic and semantic information needed to synthesize the target Indian language text belonging to that group.

I based the methodology for developing the target language text generators on the Paninian framework⁴⁴ that was applicable to all the Indian languages. Thus, instead of developing 22 different language translators from English to each one of the official Indian languages, only one translator from English to PLIL was developed. The 22 text generators transforming the PLIL into the target language catering to each one of the Indian languages, however, had to be developed. It was estimated that an additional effort of only about 30% of the total effort of developing the full translation system was needed for developing the text generator from the PLIL. In 2004, the AnglaBharati technology was transferred to a number of organizations: IIT Mumbai, IIT Gwahauti, CDAC Noida, CDAC Kolkata, CDAC Thiruvananthapuram, CDAC Pune, Jawahar Lal Nehru University New Delhi, Utkal University Bhubaneshwara, and TIET Patiala. The development work on the different text generators is currently in progress. The AnglaBharati methodology uses both interlingual and the rule-based machine translation (RBMT) approaches. This has been further hybridized with the example-based approach.⁴⁶

During 2001 and 2002, IIT Mumbai under the leadership of Pushpak Bhattacharya developed a similar interlingual approach using a universal networking language (UNL) for machine translation from English to Hindi⁸⁰ and for information extraction.⁸¹ They have also developed a Hindi word-net⁸² similar to that of Princeton University's English WordNet.

Almost at the same time, C-DAC Pune under Hemant Darbari's leadership developed a machine translation system called MANTRA (<http://www.cdac.in/html/aai/mantra.asp>), specifically tuned to the domain of translation of official documents. The MANTRA was an RBMT system and used tree-to-tree transformation from the source language to the target language.

At IIT Hyderabad, under the leadership of Rajeev Sangal, a machine translation system called Shakti for English-to-Hindi translation is being developed (<http://shakti.iiit.ac.in>).

In 1995–1996, I also designed and developed a hybrid approach to machine-aided translation that was essentially an example-based approach hybridized with a rule-base.⁴⁹

With the availability of a moderate level unilingual and bilingual corpora in Indian languages in the post-2000 period, different corpus-based approaches are currently under investigation. Since 2006, six consortium mode mission-oriented projects have been sponsored by India's Department of Information Technology and are currently in progress. These projects include two on printed text and handwritten text recognition for different Indic scripts, one project on machine translation among Indian languages, two projects on English-to-Indian languages, and one project on cross-lingual information retrieval.

And the journey continues. There is a long way to go, however, before we can truly say that we have overcome the linguistic barrier, with unconstrained speech-to-speech translation being the ultimate goal.

References and notes

1. J. Beames, *A Comparative Grammar of the Modern Aryan Languages of India: Hindi, Panjabi, Sindhi, Gujarati, Marathi, Oriya, and Bangali*, 3 vols., Trübner, 1872–1879.
2. M.B. Emeneu, *Language and Linguistic Area*, Stanford Univ. Press, 1980.
3. S.R. Hill and P.G. Harrison, *Dhatu-Patha: The Roots of Language*, Munshiram Manoharlal Publishers Pvt. Ltd., 1997.
4. S.K. Chatterji, *The Origin And Development of Bengali Language*, Rupa & Co., 2002.
5. B. Kachru, *The Alchemy of English: The Spread, Functions and Models of Non-Native Englishes*, Pergamon Press, 1986.
6. J. Baldridge, "Linguistic and Social Characteristics of Indian English," *Language In India*, vol. 2, no. 4, June-July 2002.
7. *Devanagari through the Ages*, pub. no. 8/67, Central Hindi Directorate, New Delhi, 1967.
8. H. Scharfe, "Kharoṣṭī and Brāhmī," *J. Am. Oriental Soc.*, vol. 122, no. 2, 2002, pp. 391-393.
9. A.S. Mahmud, "Crisis and Need: Information and Communication Technology in Development Initiatives Runs through a Paradox," *ITU Document WSIS/PC-2/CONTR/17-E, World Summit on Information Society*, Int'l Telecommunication Union (ITU), Geneva, 2003.
10. R.M.K. Sinha, "Multilinguality and Global Digital Divide," presented at the *Joint IAMCR/ICA Int'l Symp.: The Digital Divide*, 2001.
11. "ITU's Asia-Pacific Telecommunication and ICT Indicators Report Focuses on Broadband Connectivity: Too Much or Too Little?"; 1 Sept. 2008; http://www.itu.int/newsroom/press_releases/2008/25.html.

12. M. Schwartz, "Fastap Hindi Language Platform Slated to Revolutionise India Mobile Market," 7 Mar. 2008; <http://www.developingtelecoms.com/content/view/1165/26/>.
13. C.A. Arnaldo, "A Holistic Approach to the Computerization of Asian Scripts," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 1-24.
14. Hanzix Work Group, "Open Systems Environment for Hanzi Input Methods," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 49-58.
15. P. Lofting et al., "Handwriting: From Bamboo to Laser," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 93-112.
16. T.C. Chen, "Hanzi Characters and Their Computerizations," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 34-48.
17. Y.S. Moon, "Digital Fonts for Oriental Ideographical Languages," *Proc. Workshop Computer Processing of Asian Languages*, Asian Inst. of Technology, Bangkok, Thailand (AIT), 1989, pp. 168-174.
18. P.H. Noncarrow, "48,000 Characters in Search of a System," presented at *Symp. Linguistic Implications of Computer-based Information Systems*, New Delhi, 1978.
19. K. Hensch, "IBM History of Far Eastern Languages in Computing, Part 1: Requirements and Initial Phonetic Product Solutions in the 1960s," *IEEE Annals of the History of Computing*, vol. 27, no. 1, 2005, pp. 17-26.
20. K. Hensch, "IBM History of Far Eastern Languages in Computing, Part 2: Initial efforts for Full Kanji Solutions, 1970s," *IEEE Annals of the History of Computing*, vol. 27, no. 1, 2005, pp. 27-37.
21. K. Hensch, "IBM: History of Far Eastern Languages in Computing, Part 3: IBM Japan Taking the Lead, Accomplishments through the 1990s," *IEEE Annals of the History of Computing*, vol. 27, no. 1, 2005, pp. 38-55.
22. J. Stevens, *Sacred Calligraphy of the East*, 3rd ed., Shambhala, 1996.
23. L.S. Wakankar, *Ganesh Vidya: The Traditional Indian Approach to Phonetic Writing*, Tata Press, 1968.
24. In Unicode this *halant* symbol has been incorrectly called a *viram*. *Viram* actually represents a full stop, and Devanagari uses a vertical line (a *danda*) for this.
25. R.M.K. Sinha, "Computer Processing of Indian Languages and Scripts—Potentialities and Problems," *J. Institution of Electronics and Telecommunication Engineers*, vol. 30, no. 6, 1984, pp. 133-149.
26. R.M.K. Sinha, "Rule Based Contextual Post-Processing for Devanagari Text Recognition," *Pattern Recognition*, vol. 20, no. 5, 1987, pp. 475-485.
27. S.K. Das, *A History of Indian Literature: 1800–1910*, Sahitya Akademy, New Delhi, 1991.
28. J. Gilchrist, *Grammar of the Hindoostanee Language, or Part Third of Volume First, of a System of Hindoostanee Philology*, Chronicle Press, Calcutta, 1796.
29. W. Franklin, *Introduction to The Bhāgvat-Gēētā; The Hēētōpādēs of Veēshnōō-Sārmā*, C. Wilkins, trans., Ganesha, 2001, pp. xxiv-xxv.
30. B.S. Naik, *Typography of Devanagari*, Bombay, Directorate of Languages, government of Maharashtra, 1971, vol. 2, pp. 636-639; <http://listserv.linguistlist.org/cgi-bin/wa?A2=ind0001&L=indology&D=1&P=20160>. While researching the history of the Devanagari typewriter, I found information on the history of Rudraprayag (currently part of the state of Uttarakhand) noting that the king of Rudrapayag, Kirti Shah, invented a typewriter for Hindi around 1892 and gave the copyright to an unnamed company (<http://rudraprayag.nic.in/history.htm>); further information is not traceable thus far.
31. Vrunda (archivist), *Godrej Archives*; <http://www.archives.godrej.com>, personal communication, Sept. 2008.
32. P.V.H.M.L. Narasimham, B. Prasad, and V. Rajaraman, "Code Based Keyboard for Indian Languages," *J. Computer Soc. of India*, vol. 2, 1971, pp. 33-37.
33. R.M.K. Sinha and H.N. Mahabala, "Machine Oriented Devanagari Script," *J. Institution of Electronics and Telecommunication Engineers*, vol. 19, 1973, pp. 623-628.
34. R.M.K. Sinha, "Syntactic Pattern Analysis and its application to Devanagari Script Recognition," PhD dissertation, Electrical Eng. Dept., IIT Kanpur, 1973.
35. R.M.K. Sinha, principal investigator, *Integrated Devanagari Computer (IDC)*, tech. report IDC-84-1, Dept. of Electrical Eng., IIT Kanpur, 1984.
36. R.M.K. Sinha, "Teaching Script on a Digital Computer," *J. Institution of Electronics and Telecommunication Engineers*, vol. 22, 1976, pp. 720-722.
37. R.M.K. Sinha, "Computer Processing of Indian Languages," presented at *4th Int'l Conf. Computing in Humanities*, 1979.
38. R.M.K. Sinha and A. Raman, "A Modular Indian Language Data Terminal," *Computer Graphics (ACM SIGGRAPH)*, vol. 14, ACM Press, 1980, pp. 39-72.

A Journey from Indian Scripts Processing to Indian Language Processing

39. R.M.K. Sinha, "Computers for Indian Languages," *Proc. Ann. Convention of Computer Soc. of India*, Computer Soc. of India, Madras, 1982, pp. 163-174.
40. R.M.K. Sinha and B. Srinivasan, "Machine Transliteration from Roman to Devanagari and Devanagari to Roman," *J. Institution of Electronics and Telecommunication Engineers*, vol. 30, no. 6, 1984, pp. 243-245.
41. R.M.K. Sinha and K.S. Singh, "A Program for Correction of Single Errors in Hindi Words," *J. Institution of Electronics and Telecommunication Engineers*, vol. 30, no. 6, 1984, pp. 249-251.
42. R.M.K. Sinha, *Data Representations for Indian Language Databases*, tech. report TRCS-84-22, Dept. of Computer Science, IIT Kanpur, 1984.
43. R.M.K. Sinha, "Non-Latin Information Systems: Some Basic Issues," *Proc. Conf. Information Processing*, H. Kugler, ed., Elsevier Science, 1986.
44. R.M.K. Sinha, "A Sanskrit Based Word-expert Model for Machine Translation among Indian Languages," *Proc. Workshop Computer Processing of Asian Languages*, AIT, 1989, pp. 82-91.
45. R.M.K. Sinha et al., "AnglaBharti: A Multilingual Machine Aided Translation Project on Translation from English to Hindi," *IEEE Int'l Conf. Systems, Man and Cybernetics*, IEEE Press, 1995, pp. 1609-1614.
46. R.M.K. Sinha, "Hybridizing Rule-Based and Example-Based Approaches in Machine Aided Translation System," *2000 Int'l Conf. Artificial Intelligence (IC-AI 2000)*, CSREA Press, 2000, pp. 1247-1252.
47. R.M.K. Sinha, "An Engineering Perspective of Machine Translation: AnglaBharti-II and AnuBharti-II Architectures," *Proc. Int'l Symp. Machine Translation, NLP and Translation Support System (ISTRANS-2004)*, Tata McGraw Hill, 2004, pp. 10-17.
48. R.M.K. Sinha and A. Thakur, "Machine Translation of Bi-lingual Hindi-English (Hinglish) Text," *MT Summit X, Proc.: The Tenth Machine Translation Summit*, Phuket, Thailand, 2005, pp. 149-156.
49. R.M.K. Sinha, "A Hybridized EBMT System for Hindi to English Translation," *CSI J.*, vol. 37, no. 4, 2007, pp. 3-9.
50. M. Kulkarni, personal communication; <http://www.cdac.in/html/gist/about.asp>.
51. K. Sivaraman, "AnglaBharati: A Machine Aided Translation System from English to Indian Languages—English to Tamil Version," M.Tech thesis, Dept. of Computer Science & Eng., IIT Kanpur, 1993.
52. P.V.H.M.L. Narasimham et al., *Design Information Report on Text Composition in Telugu*, Computer Maintenance Corp., Secunderabad, 1981.
53. O. Vikas, "Summary Report of the Symposium on Linguistic Implications of Computer Based Information Systems," *Electronics Information & Planning*, New Delhi, government of India, 1978, pp. 801-804.
54. A. Bharati et al., *Anusaaraka: Machine Translation in Stages*, report no. IIIT/TR/1997/1, IIIT Hyderabad, 1997.
55. A.K. Pathak, "An Input/Output Terminal for Indian Languages," M.Tech thesis, Dept. of Electrical Eng., IIT, Kanpur, 1978.
56. M.P. Sastri, "A Universal Script Generator for Indian Languages," M.Tech. thesis, Dept. of Electrical Eng., IIT Kanpur, 1978.
57. *J. Institution of Electronics and Telecommunication Engineers*, vol. 30, no. 6, 1984, special issue on computer processing of Indian languages and scripts, R.M.K. Sinha, guest ed.
58. A. Mathur and F. Fowler, "Design of a Dynamically Reconfigurable Keyboard," *Proc. Int'l Conf. Chinese and Oriental Language Computing*, IEEE CS Press, 1987, pp. 20-23.
59. B. Nag, "Information Technology for Indian Scripts: Problems and Prospects," *Proc. Workshop Computer Processing of Asian Languages*, AIT, 1989, pp. ks-2-15.
60. K.P.S. Menon, "High Speed, Visual Direct Indian Language Data Entry," *Indian Linguistics*, 1974, vol. 35, pp. 97-111.
61. N. Mate, "Keyboard Overview—An Accommodative Approach for Devanagari Keyboard," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 291-298.
62. S.P. Mudur, *An Alphabetization Procedure for Devanagari Words*, tech. report, Nat'l Centre for Software Development and Computing Technology, 1978.
63. J.N. Tripathi, "Statistical Studies of Printed Devanagari Text (Hindi)," *J. Institution of Telecommunication Engineers*, 1971.
64. O. Vikas, "Standardizing Representation of Indian Languages for Information Processing," *Proc. Int'l Symp. Machine Translation, NLP and Translation Support System (ISTRANS-2004)*, Tata McGraw Hill, 2004, pp. 313-314.
65. *Standardisation of Indian Script Code for Information Interchange and Keyboard Layout*, Dept. of Electronics, government of India, 1983.
66. "Report of the Committee on Standardization for Indian Scripts and Keyboard Layout," *IPAG J.*, Ministry of Communication and Information Technology, New Delhi, Oct. 1986.
67. R.M.K. Sinha, "Standardizing Linguistic Information—An Overview," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 272-290.
68. M. Chandrasekaran, "Machine Recognition of the Modern Tamil Script," PhD dissertation, Univ. of Madras, India, 1982.

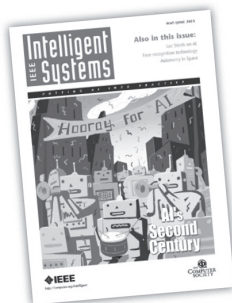
69. R. Chandrasekaran, "Computer Recognition of Certain Ancient and Modern Indian Scripts," PhD dissertation, Univ. of Madras, India, 1982.
70. U. Pal and B.B. Chaudhuri, "Indian Script Character Recognition: A Survey," *Pattern Recognition*, vol. 37, 2004, pp. 243-245.
71. V. Bansal, "Role of Knowledge in Document Recognition—A Case Study for Devanagari Script," PhD dissertation, Dept. of Computer Science and Eng., IIT Kanpur, 1999.
72. R.M.K. Sinha, "Methodology for Computer Recognition of Devanagari Scripts," *IEEE-SMC Int'l Conf.*, IEEE Press, 1984, pp. 1220-1224.
73. H. Nomura, "Role of AI in Natural Language Processing for Asian Languages," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 147-152.
74. R. Narasimhan, "Technology Support for Asian Language Studies and Applications," *Computer Processing of Asian Languages: CPAL-2 Proc.*, R.M.K. Sinha, ed., Tata McGraw Hill, 1992, pp. 25-33.
75. R.M.K. Sinha and G.C. Pathak, "A Heuristic Based Question Answering System in Natural Hindi," *IEEE-SMC Int'l Conf.*, IEEE Press, 1984, pp. 1009-1013.
76. P.C. Ganeshsundaram, "The P-Structure C-Structure Grammar (PCG) for the Contrastive Study of Two or More Languages," *J. Indian Inst. of Science*, 1978, pp. 167-191.
77. *Proc. Workshop on Computer Processing of Asian Languages*, AIT, 1989.
78. R.M.K. Sinha, ed., *Second Regional Workshop on Computer Processing of Asian Languages: CPAL-2 Proc.*, Tata McGraw Hill, 1992.
79. *Report and Recommendations of Second Regionalo Workshop on Computer Processing of Asian Languages: CPAL-2*, IIT Kanpur, India, pp. 19-21.
80. S. Dave, J. Parikh, and P. Bhattacharyya, "Interlingua Based English Hindi Machine Translation and Language Divergence," *J. Machine Translation (JMT)*, vol. 17, Sept. 2002, pp. 251-304.
81. P. Bhattacharyya, "Knowledge Extraction into Universal Networking Language Expressions," *Proc. Universal Networking Language Workshop*, 2001.
82. D. Narayan et al., "An Experience in Building the Indo WordNet—A WordNet for Hindi," *Proc. Int'l Conf. Global WordNet (GWC 02)*, 2002.



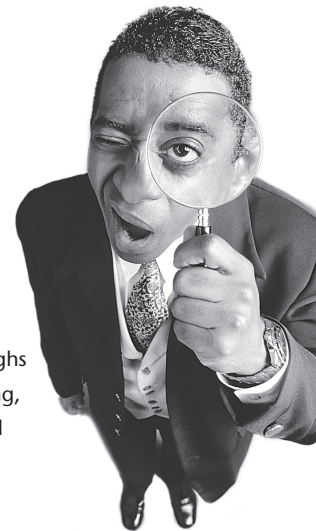
R. Mahesh K. Sinha is a professor of computer science and engineering (CSE) and electrical engineering (EE) and has been on the faculty of CSE and EE at IIT Kanpur since 1975. He is the originator of the well-known multilingual GIST technology, AnglaBharati and AnuBharati machine translation technology, and ISCII, among others. He has been a visiting faculty member at Michigan State University, Wayne State University, the University of Quebec (INRS), and AIT, Bangkok. Sinha obtained a PhD from IIT Kanpur in 1973. Contact him at rmk@iitk.ac.in.

For further information on this or any other computing topic, please visit our Digital Library at <http://computer.org/csdl>.

See the Future of Computing Now in *IEEE Intelligent Systems*



Tomorrow's PCs, handhelds, and Internet will use technology that exploits current research in artificial intelligence. Breakthroughs in areas such as intelligent agents, the Semantic Web, data mining, and natural language processing will revolutionize your work and leisure activities. Read about this research as it happens in *IEEE Intelligent Systems*.



SUBSCRIBE NOW! <http://computer.org/intelligent>

**IEEE
Intelligent
Systems**