1.1 WHAT IS CHARACTER ENCODING

Human being recognizes a character by its shape whereas a machine recognizes the character through bits or bytes. Human being processes the characters through his perception and intelligence whereas computer processes characters through a defined set of codes that represent character. In the age of information explosion, it is humanly impossible to store, retrieve and analyze the data manually. It is even more difficult to do processing accurately since human being is always prone to fatigue.

This is where computers came into existence as a substitution for the human labor. Computers store bundles of information and can process the data at much higher speeds than a human being without any fatigue or errors (if programmed properly). This requires the data to be given in the coded format, which can be understood by the computer. Due to the increase of text based computing, users now require that computers are able to recognize and manipulate text in different languages.

In general character is an abstract concept, represented by a family of graphical shapes, which share some commonality. This means that a bold ‘a’, an italic ‘a’, and a
superscripted 'a' are all the same character, although represented by slightly different shapes. The SGML standard defines a character as an atom of information with an individual meaning, defined by a character repertoire. It also defines a character repertoire as a set of characters that are used together.

Human beings know how to represent characters visually by drawing them, but when it come to machine representation of character digitally is much more difficult task. As the numbers are encoded digitally with binary numbers, it is possible to encode characters in the same way. When the string "Hallow!" is represented as a sequence of binary numbers, code character set plays an important role. Coded character sets assign numbers to each character in the set. In this way a string becomes a sequence of numbers, and by having special routines that interpret these sequences as strings computers can successfully represent text. To display the text, all we need is a table of images with an entry for each character. Such a table is often known as a font, and each individual entry as a glyph. In some cases mapping of characters is more complex phenomena. Storing the strings as an array of numbers allows comparison of two strings. Character encoding is nothing but identifying the characters to be represented and assigning numbers to each character.

1.2 LITERATURE REVIEW

When Samuel F. B. Morse developed the telegraph between 1838 and 1854, he also devised a code to use with it. Each letter in the alphabet corresponded to a series of short and long pulses (dots and dashes). There was no distinction between uppercase and lowercase letters, but numbers and punctuation marks had their own codes. Telex
codes include Baudot [30] and a code known as CTTT #2, are 5-bit codes that incorporate letter shifts and figure shifts.

Early computer character codes evolved from the coding used Hollerith cards, invented by Herman Hollerith and first used in 1890 United States census [10]. A 6-bit character code known as BCDIC ("Binary-Coded Decimal Interchange Code") based on Hollerith coding was progressively extended to the 8-bit EBCDIC in 1960s and remains the standard on IBM mainframes but nowhere else. The American Standard Code for Information Interchange (ASCII) had its origins in the late 1950s and was finalised in 1967. During the development of ASCII, there was considerable debate over whether the code should be 6, 7, or 8 bits wide. Reliability considerations seemed to mandate that no shift character be used, so ASCII couldn't be a 6-bit code. Cost ruled out the 8-bit version. (Bits were very expensive back then.) The final code had 26 lowercase letters, 26 uppercase letters, 10 digits, 32 symbols, 33 control codes, and a space, for a total of 128 codes. ASCII is currently documented in ANSI X3.4-1986, "Coded Character Sets—7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)," published by the American National Standards Institute. ASCII code possesses many positive features. The 26 letter codes are contiguous, for example. (This is not the case with EBCDIC.) Uppercase letters can be converted to lowercase and back by flipping just one bit (5th bit). The codes for the 10 digits are easily derived from the value of the digits by subtracting value of character "0" from the digit. (In EBCDIC, the code for the character "0" followed the code for the character "9", which makes it difficult to extract values from characters). Out of all, ASCII is a very dependable standard. Except ASCII, no other standard is as prevalent or as ingrained in our keyboards, video displays, system hardware, printers,
font files, operating systems, and the Internet. The big problem with ASCII is indicated by the first word of the acronym. ASCII is truly an American standard, and it is not even good enough for other countries where English is spoken. For example, the British pound symbol (£) was not there in 7-bit ASCII.

English uses the Latin (or Roman) alphabet. Among written languages that use the Latin alphabet, English is unusual in that very few words require letters with accent marks (or "diacritics"). Even for those English words where diacritics are traditionally proper, such as cooperate or resume, the spellings without diacritics are perfectly acceptable. However, north and south region of the United States and across the Atlantic many countries, the languages use diacritics. These accent marks originally aided in adopting the Latin alphabet to the differences in spoken sounds among these languages. In east or south of Western Europe there are languages that don't use the Latin alphabet at all, such as Greek, Hebrew, Arabic, and Russian (which uses the Cyrillic alphabet). In some Asian countries like China, the ideographic Han characters are used, which were also adopted in Japan and Korea.

The history of ASCII since 1967 is mostly a history of attempts to overcome its limitations and make it more applicable to languages other than American English. In 1967 the International Standards Organisation (ISO) recommended a variant of ASCII with codes 0x40, 0x5B, 0x5C, 0x5D, 0x7B, 0x7C, and 0x7D "reserved for national use" and codes 0x5E, 0x60, and 0x7E labelled as "may be used for other graphical symbols when it is necessary to have 8, 9, or 10 positions for national use."

This is obviously not the best solution to internationalisation because there's no guarantee of consistency. However, it indicates how desperate people were to successfully code symbols necessary to various languages.
By the time the early small computers were being developed, the 8-bit (byte) had been firmly established. Thus, if a byte were used to store characters, 128 additional characters could be invented to supplement ASCII. When the original IBM PC was introduced in 1981, the video adapters included a ROM-based character set of 256 characters, which in itself was to become an important part of the IBM standard.

The original IBM extended character set [61] included some accented characters and a lowercase Greek alphabet (useful for mathematics notation), as well as some block-drawing and line drawing characters. Additional characters were also assigned to the code positions of the ASCII control characters, because the bulk of these control characters were not required.

This IBM extended character set was burned into countless ROMs on video boards and in printers, and it was used by numerous applications to decorate their character-mode displays. However, this character set did not include enough accented letters for all Western European languages that used the Latin alphabet.

MS-DOS 3.3 introduced the concept of code pages [61] to IBM PC users, a concept that was also carried over to Windows. A code page defines a mapping of character codes to characters. The original IBM character set became known as code page 437, or "MS-DOS Latin US." Code page 850 is "MS-DOS Latin 1," which replaces some of the line drawing characters with additional accented letters. Other code pages were defined for other languages. The lower 128 codes are always the same; the higher 128 codes depend on the language for which the code page is defined.

Under MS-DOS, if a user sets the PC's keyboard, video display, and printer to a specific code page and then creates, edits, and prints documents on the PC, all will be
well and everything will be consistent. However, if the user attempts to exchange documents with another user using a different code page or to change the code page on the machine leads to problems. Character codes are associated with the wrong characters. Applications can save code page information with documents in an attempt to reduce problems, but this strategy involves some work in converting between code pages.

Although code pages originally provided only additional characters of the Latin alphabet beyond the unaccented characters, eventually code pages were devised where the higher 128 characters contained complete non-Latin alphabets, such as Hebrew, Greek, and Cyrillic. Such variety makes code page mix-ups potentially worse, of course; it is one thing, if a few accented letters appear incorrect and quite another if an entire text is an incomprehensible jumble.

Code pages proliferated beyond all reason. For example, the MS-DOS code page 855 for Cyrillic is not the same as either the Windows code page 1251 for Cyrillic or the Macintosh code page 10007 for Cyrillic. Code pages in each environment are the modifications of the standard character set for the environment. IBM OS/2 also supports a variety of EBCDIC code pages.

ASCII represents any character set as a set of 256 characters. The ideographs of Chinese, Japanese, and Korean languages are about 21,000 in number. All the Indian languages have many characters and the combination of these characters will be quite high.

There is an important ISO character set standard that fixes this problem for many of the languages in the world, with a series of 256-codes in each set. This standard is known as ISO-8859 and consists of a number of parts, each of which defines a
character set. At the moment there are 14 parts, the most important of these sets is the one often known as ISO-Latin 1 or just Latin 1, which is defined in ISO 8859-1.

ISO 8859-1 includes 128 characters from ASCII (leaving out the control characters), but adds 96 additional characters (thus requiring 8 bits), which are accented characters. Certain special forms of punctuation and some characters are needed in other parts of Europe.

The ISO 8859 character set standards have been restricted to using 8 bits because this exactly corresponds to one byte on modern computers, and most computer systems assume that a character is represented by a single byte. This does not work for all languages, but allowed computer systems to maintain backward compatibility and so this outdated assumption has remained. Today it is going to be abandoned with the introduction of Unicode, but this work is progressing slowly.

ISO-8859 does not only consist of ISO 8859-1, however, but also has thirteen other character sets for different parts of the world, all of which are identical to ASCII in the lower 128 characters and then 128 additional characters correspond to other language. Various ISO-8859 series of standards are along with their coverage of respective languages across the globe is as listed below:

ISO 8859-1 (Latin alphabet no.1)-The western European languages, including Malaysia and Tagalog.

ISO 8859-2 (Latin alphabet no.2) -The central European languages, such as Albanian, Czech, German, Hungarian, Polish, Rumanian, Croatian, Slovak, Slovene and Swedish.

ISO 8859-3 (Latin alphabet no.3) -The southern European languages, such as Catalan, French, Galician, Italian, Maltese and Turkish, and also Afrikaans.
ISO 8859-4 (Latin alphabet no.-4)-The northern European languages, such as Danish, Estonian, Finnish, German, Greenlandic, Sami, Latvian, Lithuanian, Norwegian and Swedish. This character set has now been superceded by ISO 8859-10.

ISO 8859-5 (Latin/Cyrillic alphabet)-The eastern European languages written with Cyrillic script, such as Belarussian, Belorussian, Macedonian, Russian, Serbian and Ukrainian.

ISO 8859-6 (Latin/Arabic alphabet)-Arabic without accents, Farsi and Urdu are also written with Arabic script, but this character set lacks the extra characters needed for these languages.

ISO 8859-7 (Latin/Greek alphabet)-Modern Greek.

ISO 8859-8 (Latin/Hebrew alphabet)-Hebrew and Yiddish.

ISO 8859-9 (Latin alphabet no. 5)-Many western European languages, but most importantly Turkish.

ISO 8859-10 (Latin alphabet no. 6)-Northern European languages, such as Danish, Estonian, Faeroese, Finnish, German, Inuit, Icelandic, Sami, Latvian, Lithuanian, Norwegian and Swedish. Note that the Norwegian Sami community has rejected this character set and uses a character set of their own design instead.

ISO 8859-11 (Latin/Thai alphabet)-Thai. (This character set is the same as the one known as TIS-620).

ISO 8859-15 (Latin alphabet no. 9)-Western European languages. This character set modifies ISO 8859-1 to include the euro symbol and some special characters needed for French.

The approach taken by ISO 8859 is useless for the oriental languages that contain tens of thousands of characters. This led to the development of oriental character sets and

**Problems with exchange**

In the recent period networking of computers is observed as a common phenomena in every organization. Information transfer and visualization is an important application in these networks. Exchange of information is a part of the application layer [4,38] in the Open System Interconnection model of computer networks. The ISO 8859 series can solve the basic problem of enabling different people with different language to write text on their computers. When this text file moves from one computer to the other, the text mapping may be done to wrong abstract character and display a different character when different standards are used on different computers. Similarly, other standards solve a small part of the internationalization problem. The problem associated with case conversion, sorting, searching and layout still remain. Case conversion is highly language-specific. Sorting depends on the language in use. Layout is extremely tricky. Different scripts differ greatly in how they are displayed. Indic scripts are written from left to right, where as Hebrew and Arabic are written from right to left. Similarly many languages have complicated composition rules. These problems are addressed in the International standards like Unicode and ISO 10646.
Internet Explosion

With the development of World Wide Web (WWW), there is tremendous increase in the use of Internet. Public Switched Telephone Network and the latest DSL Technology [50] allow the users to have Internet access from the remote areas. WWW brings the promise of instant access to multinational, multilingual market place. The Standard Generalized Markup Language (SGML.) standard led to Hyper Text Markup Language (HTML) specifications [18.69], which is used for displaying content independent of the platform. Hyper Text Transmission Protocol (HTTP) allows this content to be interchanged between computers. WWW include bi-directional data transfer, which has traditionally been used as a collection of applications exchanging data based on protocols. In the recent period there is a tremendous increase in data transfer among servers, proxies and clients. The data here include non-ASCII characters also. The latest estimated figures of the number of people online in each language zone are shown in table 1.1.

<table>
<thead>
<tr>
<th></th>
<th>Internet access (Millions)</th>
<th>% of people online</th>
<th>Total Population (Millions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>192.1</td>
<td>47.6%</td>
<td>500</td>
</tr>
<tr>
<td>Non-English</td>
<td>211.3</td>
<td>52.4%</td>
<td>5600</td>
</tr>
<tr>
<td>European Languages</td>
<td>118</td>
<td>29.2%</td>
<td>1089</td>
</tr>
<tr>
<td>Asian Languages</td>
<td>93.3</td>
<td>23.1%</td>
<td>4511</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1.1 Number of People Online with language zone

The statistics given in the above table are taken from 'glreach' web site and these estimates are taken up to December 2000. Among Asian languages Japanese language
is used by 9.6% of total population who access Internet. Chinese language is used by 7.2 percent, Korean by 4.2 percent, Arabic by 0.6 percent and Hebrew and Thai languages are used by less than 1.5 percent of total people who will be on the net. Indians using Indian languages on the net is found to be minimal.

Due to this increase of non-ASCII text, international character set standards like ISO-10646 and Unicode were evolved for the written scripts of the world. The Internet Architecture Board (IAB) conducted a workshop on character sets and addressed the problem using the character sets on the Internet. IAB workshop states that successful display of plain text transmitted over the Internet requires a lot of information about the text itself, such as the underlying character set, language and so forth. An additional set of formatting information is needed if the receiving application wishes to use local conventions when it presents the data to the user. This formatting includes information that provides necessary to format certain types of textual data like dates, times, numbers, monitory notations etc. For this purpose IAB defined an architectural model that consists of user interface part and on-the-wire transmission part. IAB defines the user interface part of the architecture as layout, culture, locale and language. The term "language tag" is introduced here to identify the language [1,2,15,21]. This is because Multipurpose Internet Mail Extension (MIME) protocols [14,44] accept 7 bit ASCII characters only. An encoding method [42,43,46] is used with the help of language tagging.

IAB defines the other part of the architecture as coded character set, character encoding scheme and transfer encoding syntax. The default coded character set for Internet is ISO-10646 [28]. IAB also defines that new protocol specifications should use ISO-10646 as the base character set. Language tagging is used for the protocols
that use ASCII and ISO-8859 series of character sets. The other character sets that are not defined in ISO-8859 series should go through a registration process.

The process of Internationalization [9] started with the language specification using charset option and language tagging [3,13] in Hyper Text Transmission Protocol. The concept of Globalization is possible not only with HTTP but also with other protocols, which deal with domain names, remote execution [63] like telnet and MIME extensions. The Internet Engineering Task Force [19] is responsible for the standardization process of protocols on Internet. The charset option for telnet [16] allow language tagging in the present day environment. Internationalization of domain names [48] is possible with ISO-10646 because of the specification of the DNS protocol.

ISO-10646 specifies the Universal Multiple-Octet Coded Character Set (UCS). It is applicable to the representation, transmission, interchange, processing, storage, input and presentation of the written form of the languages of the world as well as additional symbols. Unicode standard [64], which is declared in 1991 is another International character set standard for the languages in the world. Changes in the Unicode and amendments to ISO-10646 have tracked each other, so that the character repertoires and code point assignments have remained in sync.

To encode any character set of written language, three different approaches are followed. One is glyph encoding model, the second is language oriented encoding and the third is character-glyph model. These three models [58] are discussed in the next chapter. On one extreme glyph encoding model require glyph registration process as code points. In language oriented encoding all possible characters are to be coded, which is another extreme. The character-glyph model defines characters and glyphs
separately. This model is a compromise between the two extremes. In this model characters are defined with abstract character repertoire and they are assigned with numbers or code points. Glyphs are derived from the sequence of code points. The selection of repertoire of coded character set is an important aspect in the character-glyph model. ISO technical report on operational model for characters and glyphs specified three important considerations for selecting the repertoire of any coded character set. The first consideration is the characters with same shape but different meaning. The second is the characters with same shape and different meaning. The last consideration is compatibility with the older character set. UCS and Unicode defined their repertoire for the written scripts in the world based on the above considerations. The repertoire for Telugu script is proposed in both the standards based on ISClI standard with a code space of 128 and the language tag as UCII. ISCII is the standard for all Indic scripts, which uses 7-bit and 8-bit code extension technique. This repertoire is based on common canonical structure of all Indic scripts. Even though it resembles the character glyph model, the language-oriented characteristics are considered for commonality of all Brahmi based scripts. For defining the best suitable repertoire the other factors such as layout, culture and locale are necessary. In this thesis an attempt is made to extract these characteristics of Telugu and a suitable repertoire is proposed.

1.3 OBJECTIVES OF THE WORK

The main objective of this work is to extract best suitable abstract character repertoire of the current Telugu script. Among all Indic scripts, Telugu evolved at a later stage.
Though there are common features with other scripts, the layout, culture and locale are different.

Generally human beings identify the character through glyphs. These glyphs are specific to culture. In the character-glyph encoding model the parameters of layout, locale and culture reflect in glyphs. At the same time pure glyph encoding model is not suitable for processing of characters. Here selection of repertoire is important in the usage point of view.

The complete letter shapes of Telugu script exceed six hundred thousands. These letter shapes are formed with a few glyphs with a specific set of rules. Using this set of rules it is possible to extract specific features of the script and incorporate these features in the repertoire.

Layout is another problem of the script. The character-glyph model deals with not only repertoire and also the glyph rendering method. Here the layout of the character plays an important role. Using the glyph sets of the script and the layout, glyph-rendering engine will be designed. Another objective of this work is to propose the required glyph sets and a complete glyph rendering engine for the current Telugu script.

1.4 PROPOSED METHOD OF ATTACK

The canonical structure of Telugu script is the same as other Indic scripts, which is based on the phonetic principles. But the character glyph formation is based on script-based principles. In this process many character-glyph constructs are possible. In the character-glyph encoding model a consistent approach is necessary to identify the repertoire that will reflect the canonical structure and glyph rendering principles. Here
it is necessary to analyze the canonical structure in detail. Since the analysis is aimed at current script, various text samples that are in current usage are taken for analysis. Different combinations that are possible in the canonical structure will give complete information about the phonetic principles applied on the script. For this purpose four different categories of samples are selected. Samples from children’s science magazine, poem mode samples and indirect speech samples are the first three categories in the increasing order of complexity of the script. Direct speech samples are the fourth category of samples, which gives more information regarding the cultural modifications. After analyzing these text samples the feature of the script related to phonetic principles can be extracted, which are more useful in identifying the abstract character repertoire.

Analyzing the glyph structure is another important part of the locale and culture. Different possible glyph combinations are analyzed and the specific features related to glyphs of Telugu script are analyzed. These features are incorporated into the repertoire and extensions are proposed for the existing repertoire defined by ISClI.

The canonical structure and phonetic principles are related to each other where there is an inherent grammatical construction principles in the formation of complete characters. These grammatical construction principles allow a different direction to define the repertoire, which is proposed in this thesis. Comparison among all these repertoires is done in terms of number of code points per character and memory requirement.

It is necessary to identify the complete set of glyphs for glyph rendering of a character from a sequence of code points that are part of the canonical structure. The layout and glyph structure of Telugu script is analyzed. From this analysis it is possible to extract
different glyph sets required for the glyph-rendering engine. These glyph sets are ordered with reference to the canonical structure. Finally a complete glyph rendering for all possible combinations of Telugu scripts is developed using these glyph sets. With the repertoire and glyph rendering engine it is possible to implement information interchange in the computer networks for Telugu scripts.

The organization of the Thesis is as follows

In Chapter 2, universal character encoding schemes reported in the literature up to December 2000 are reviewed. Chapter 3 presents the nature of Indic scripts, their similarities and dissimilarities. Unicode code allocation for all Indic scripts is also presented in this chapter.

In the next chapter the results of the analysis of canonical structure of Telugu script is presented followed by the discussion on different possible combinations and their contribution to the current text samples.

Chapter 5 presents the extension to the existing repertoire using the analysis on culture and locale. A discussion on new repertoire based on grammatical construction of characters is also presented followed by the comparison among the existing and proposed schemes.

In chapter 6 a complete set of glyphs required for Telugu scripts is presented with a detailed discussion of glyph rendering engine.

The final chapter presents a brief summary of the work followed by a discussion on the future scope of the work.