2.1 INTRODUCTION

The minimum distinctive unit of the writing system of a particular language is a grapheme. The grapheme [20] has no physical identity. But it is an abstraction based on the different shapes of written signs and their distribution within a given system. This is a linguistic concept. This concept can be extended into the information system to say that the unit of information most appropriate to a given process is language dependent and may or may not correspond to a single character or a single glyph. This concept is extended to coded character sets of the present day usage. For a machine that process information for sorting of text data, it is important that sorting should be done according to language and its culture. The coded character sets that are used in computers have character codes or weights, which are useful for processing information. The question arises here regarding distinct units, which are necessary for encoding process. Three different models are possible here. One is coding the glyphs. The other is coding the syllables and the final one is character-glyph encoding model.
In this chapter general principles of encoding abstract units and the universal standards for character encoding is discussed.

2.2 ENCODING PRINCIPLES

Encoding of glyphs.

In the glyph-encoding model all the distinct glyphs that are necessary to represent characters will be encoded. The famous example is the typewriter keyboard, where only glyphs are available on the keyboard and these glyphs are positional according to the layout of the particular character. This model is useful for presentation of text but not for processing of text information. Text processing requires specification of ordering which is a difficult task. For example the character /KA/ in Telugu requires 2 glyphs and the character /BA/ require 1 glyphs. Here the order of characters is missing, because /KA/ is considered as a lower character than /BA/. Similarly for Arabic characters where the directionality is different, processes will be a tough task. Similarly in Indic scripts, Telugu script requires a glyph set of 111 glyphs and Devanagari requires a glyph set of more than 1000 glyphs. The encoding system will be more complex because there is a large variation in the requirement of code space. Therefore, this model is not useful for presentation and processing of text information in computers.

Language Oriented Encoding Model

In this model all possible text elements are to be coded. The Latin text has 26 lower case and 26 upper case letters, which can be encoded easily. If the accents are considered, then the number of text elements drastically increases to hundreds. Here
encoding require large amount of code space and processing consumes more amount of time. Similarly in Indic scripts the possible text elements are found to be few thousands. Hence, this model also not useful for processing of Telugu text in computers.

**Character-glyph model**

In this model there is a clear distinction between text elements and characters. Text element is nothing but grapheme and characters are abstract units considered for encoding purpose. The inherent design in this model provides a set of code values that allows programmers to design applications capable of implementing variety of text processes in the desired language. Then code values may not map directly to any particular set of text elements. The latest coded character sets ASCII, ISO-8859 series, ISO-10646 and Unicode follow this model for encoding of characters. ISO-10646 and Unicode standards provide multilingual support by encoding different written scripts of the world. These two standards follow 16 encoding schemes, which are discussed in the next section.

**2.3 ISO-10646**

ISO-10646 defines a multi-octet character set called the Universal Character Set (UCS), which encompasses most of the world's writing systems. Two multi-octet encoding methods are defined and they are four-octet per character encoding method called UCS-4 and two-octet per character encoding method called UCS-2. The second method UCS-2 can address only the first 64K characters of the UCS (the Basic Multilingual Plane, BMP). Currently there are no assignments for UCS-4.
The value of any octet is expressed in hexadecimal notation [47] from 00 to FF in ISO/IEC 10646. The canonical form of this coded character set is the way in which it is to be conceived. This standard uses a four-dimensional coding space, regarded as a single entity, consisting of 128 three-dimensional groups. Each group consists of 256 two-dimensional planes. Each plane consists of 256 one-dimensional rows with each row containing 256 cells. A character is located and coded at a cell within this coding space or the cell is declared unused. In the canonical form, four octets are used to represent each character, and they specify the group, plane, row and cell, respectively. The canonical form consists of four octets since two octets are not sufficient to cover all the characters in the world, and a 32-bit representation follows modern processor architectures. The four-octet canonical form can be used as a four-octet coded character set, in which case it is called UCS-4.

The first plane (Plane 00 of Group 00) is called the Basic Multilingual Plane. The Basic Multilingual Plane includes characters of general use in alphabetic, syllabic and ideographic scripts together with various symbols and digits. The subsequent planes are regarded as supplementary or private use planes, which will accommodate additional graphic characters. Each character is located within the coded character set in terms of its Group-octet, Plane-octet, Row-octet, and Cell-octet. In addition to the canonical form, a two-octet BMP form is specified. Thus, the Basic Multilingual Plane can be used as a two-octet coded character set identified as UCS-2. Subsets of the coding space may be used in order to give a sub-repertoire of graphic characters.

Up to the present time, changes in Unicode and amendments to ISO/IEC 10646 have tracked each other, so that the character repertoires and code point assignments have remained in sync. The relevant standardization committees have committed to
maintain this very useful synchronism. The UCS-2 and UCS-4 encodings, however, are hard to use in many current applications and protocols that assume 8 or even 7 bit characters. Even newer systems able to deal with 16 bit characters cannot process UCS-4 data. This situation has led to the development of so-called UCS transformation formats (UTF), each with different characteristics. UTF-1 has only historical interest, having been removed from ISO/IEC 10646. UTF-7 has the quality of encoding the full BMP repertoire using octets with the high-order bit clear (7 bit US-ASCII values), and is thus deemed a mail-safe encoding. UTF-8 [66] uses all bits of an octet, but has the quality of preserving the full US-ASCII range: US-ASCII characters are encoded in one octet having the normal US-ASCII value, and any octet with such a value can only stand for US-ASCII character, and nothing else. UTF-16 is a scheme for transforming a subset of the UCS-4 repertoire into pairs of UCS-2 values from a reserved range. UTF-16 impacts UTF-8 in that UCS-2 values from the reserved range must be treated specially in the UTF-8 transformation. UTF-8 encodes UCS-2 or UCS-4 characters as a varying number of octets, where the number of octets, and the value of each, depend on the integer value assigned to the character in ISO/IEC 10646.

This transformation format has the following characteristics:

- Character values from 0000H to 0000H and 000011 to 007F (US-ASCII repertoire) correspond to octets 00 to 7F. A direct consequence is that, a plain ASCII string is also a valid UTF-8 string.

- US-ASCII values do not appear, otherwise, in UTF-8 encoded character stream.

This provides compatibility with file systems or other software that parses based on US-ASCII values but are transparent to other values.
• Round-trip conversion is easy between UTF-8 and either of UCS-4, UCS-2.
• The first octet of a multi-octet sequence indicates the number of octets in the sequence. The octet values FE and FF never appear.
• Character boundaries are easily found from anywhere in an octet stream.
• The lexicographic sorting order of UCS-4 strings is preserved.
• The Boyer-Moore fast search algorithm can be used with UTF-8 data.
• UTF-8 strings are fairly reliable in recognizing as it is by a simple algorithm, i.e. the probability that a string of characters in any other encoding appears, as valid UTF-8 is low, diminishing with increasing string length.

2.4 UNICODE

Unicode is an International standard started from 1991. Unicode is a character coding mechanism that is trying to provide a global character set. It attempts to strike a balance between local users of a script and global or multilingual implementers. This is a 16 bit or double byte encoding standard. Latest programming languages like Java support [5,11,62] Unicode with double byte characters. Even the XML [45] supports Unicode definition of charset for information exchange in WWW. Unicode and ISO 10646 standards are one-to-one synchronized [41] in terms of repertoire and code point assignments. In Unicode code space is allocated for written script in the world. There is no specific language tagging mechanism, but the code space assignment is ordered in such a way that language tagging is easy for the existing protocols to implement language tagging. Another important feature of Unicode is the support of bi-directionality. The bi-directional algorithm [39] of Unicode allow the scripts like Arabic and scripts like English can be written side by side.
The character-encoding model of this standard differs from IAB model by proving two more extra levels. The repertoire is inherent in IAB model where as repertoire is defined specifically in Unicode. The five levels can be summarized as:

Abstract Character Repertoire (ACR) is the set of characters to be encoded, e.g., some alphabet or symbol set.

Coded Character Set (CCS) is a mapping from an abstract character repertoire to a set of non-negative integers.

Character Encoding Form (CEF) is a mapping from a set of non-negative integers (from a CCS) to a set of sequences of particular code units of some specified width, such as bytes.

Character Encoding Scheme (CES) is a mapping from a set of sequences of code units (from one or more CEFs) to a serialized sequence of bytes.

Transfer Encoding Syntax (TES) is a reversible transform of encoded data. This data may or may not contain textual data.

In addition, a Character Map (CM) is defined to be a mapping from an abstract character repertoire to a serialized sequence of bytes.

**Abstract Character Repertoire**

A repertoire is defined as the set of abstract characters to be encoded, normally a familiar alphabet or symbol set. The word abstract just means that these objects are defined by convention, such as the 26 letters of the English alphabet, uppercase and lowercase forms. Repertoires are unordered sets that come in two types: fixed and open. For most character encoding schemes, the repertoire is fixed (and often small). Once the repertoire is decided, it can never be changed. Addition of a new abstract
European alphabets and symbols of Latin-1, the IBM host Japanese repertoire. Open repertoires are the Windows Western European repertoire and the Unicode/10646 repertoire.

**Characters vs. Glyphs**

The distinction between characters and glyphs is very important. A glyph is a particular image, which represents a character or part of a character. Glyphs do not correspond one-for-one with characters. For example, a sequence of \( \text{f} \) followed by \( \text{i} \) may be represented with a single glyph, called a fi ligature. On the other hand, the same image as the fi ligature could be achieved by a sequence of two glyphs with the right shapes. The choice of whether to use a single glyph or a sequence of two is up to the font containing the glyphs and the rendering software.

Similarly, an accented character could be represented by a single glyph, or by different component glyphs positioned appropriately. In addition, the separate accents can also be considered as characters in their own right, in which case a sequence of characters can also correspond to different possible glyph representations.

In non-Latin languages, the connection between glyphs and characters may be even less direct. Glyphs may be required to change their shape and widths depending on the surrounding glyphs. These glyphs are called contextual forms. These contextual forms are found in Arabic and Indic scripts.

Glyphs may also need to be widened for justification instead of simply adding width to the spaces. Ideally this would involve changing the shape of the glyph depending on the desired width. On some systems, this widening may be achieved by inserting extra connecting glyphs called kashidas. In such a case, a single character may conceivably correspond to a whole sequence of kashidas + glyphs + kashidas. In other
cases a single character must correspond to two glyphs, because those two glyphs are positioned around other letters. For example two part vowels in Indic scripts fall into this category.

The upshot is that the correspondence between glyphs and characters is not one-to-one, and cannot in general be predicted from the text. The ordering of glyphs will also not in general correspond to the ordering of the characters, because of right-to-left scripts like Arabic and Hebrew. Whether a particular string of characters is rendered by a particular sequence of glyphs will depend on the sophistication of the host operating system and the font.

It is important to note that for historical reasons, abstract character repertoires may include many entities that normally would not be considered as appropriate members of an abstract character repertoire. These may include ligature glyphs, contextual form glyphs, glyphs that vary by width, sequences of characters, and adorned glyphs.

What an end-user thinks about a single character, it may be represented by multiple code points; conversely, a single code point may correspond to multiple characters.

**Subsets**

Unlike most character repertoires, Unicode/10646 is deliberately intended to be universal in coverage. This implies in practice that, in the given complexity of many writing systems, nearly all implementations will implement some subset of the total repertoire, rather than all the characters.

Formal subset mechanisms are occasionally seen in implementations of some Asian character sets, where for example, the distinction between "Level 1 JIS" and "Level 2 JIS" support refers to particular parts of the repertoire of the JIS X 0208 kanji characters to be included in the implementation. Sub setting is a major formal aspect
of ISO/IEC 10646-1. The standard includes a set of internal catalog numbers for named subsets, and further makes a distinction between subsets that are fixed collections and those that are open collections, defined by a range of code positions. The collections that are defined by a range of code positions are themselves open subsets of the repertoire, since they could be extended at any time by an addition to the repertoire which happens to get encoded in a code position between the range limits which define such a collection. For the Unicode Standard, subsets are nowhere formally defined. It is considered, up to the implementation to define and support the subset of the universal repertoire that it wishes to interpret.

**Coded Character Set (CCS)**

A coded character set is defined to be a mapping from a set of abstract characters to the set of non-negative integers. This range of integers need not be contiguous.

An abstract character is defined to be in a coded character set if the coded character set maps from it to an integer. That integer is said to be the code point for the abstract character. The abstract character is then becomes an encoded character. Effectively, coded character sets are the basic objects produced by both ISO and vendor character-encoding committees. They relate a defined repertoire to nonnegative integers, which then can be used unambiguously to refer to particular abstract characters from the repertoire.

The Unicode scalar value of the Unicode concept is explicitly this code point, used for mapping of the Unicode repertoire. A coded character set may also be known as a character encoding, a coded character repertoire, a character set definition, or a code page.
Character Naming

In the JTC1/SC2 context, coded character sets also require the assignment of unique character names to each abstract character in the repertoire to be encoded. This practice is not generally followed in vendor coded character sets or the encodings produced by standards committees outside SC2, where the names provided for characters, if any, are often variable and annotative, rather than normative parts of the character encoding.

The main rationale for the SC2 practice of character naming was to provide a mechanism to unambiguously identify abstract characters across different repertoires given different mappings to integers in different coded character sets. Thus LATIN SMALL LETTER A WITH GRAVE would be seen as the same abstract character, even when it occurred in different repertoires and was assigned different integers, depending on the particular coded character set.

This functionality of ensuring character identity across different coded character sets is handled in the IBM CDRA model instead by assigning a catalogue number, known as a graphic character glyphic identifier (GCGID), to every abstract character used in any of the repertoires accounted for by the CDRA. Abstract characters that have the same GCGID in two different coded character sets are by definition the same character. Other vendors have made use of similar internal identifier systems for abstract characters.

The advent of Unicode/10646 has largely rendered such schemes obsolete. The identity of abstract characters in all other coded character sets is increasingly being defined by reference to Unicode/10646 itself. Part of the pressure is to include every "character" from every existing coded character set into the Unicode Standard results
all of the same length is known as fixed width. A character encoding form whose sequences are not all of the same length is known as variable width.

A character encoding form for a coded character set is defined to be a character encoding form that maps all of the encoded characters for that coded character set. In many cases, there is only one character encoding form for a given coded character set. In some such cases only the character encoding form has been specified. These leaves the coded character set implicitly defined, based on an implicit relation between the code unit sequences and integers.

While interpreting a sequence of code units, there are three possibilities that exist.

The sequence may be illegal. There are two variants of this possibility. First is where the sequence is incomplete. The second variant is where the sequence is complete, but explicitly illegal.

The sequence represents a valid code point, but is unassigned. This sequence may be given an assignment in some future, evolved version of the character encoding. The source sequence is assigned but it represents a valid encoded character. There are two variants of this. First is a standard assigned character. The second variant is a user-defined character.

The encoding form for a CCS may result in either fixed-width or variable-width collections of code units associated with abstract characters. The encoding form may involve an arbitrary reversible mapping of the integers of the CCS to a set of code unit sequences. UTF-8 is a variable-width encoding form that is a mix of one to four 8-bit code units in Unicode and one to six code units in 10646. UTF-16 is another variable-width encoding form, which is a mix of one to two 16-bit code units.
The encoding form defines one of the fundamental relations that internationalized software cares about: how many code units are there for each character. This used to be expressed in terms of how many bytes each character was represented by. With the introduction of UCS-2, UTF-16, UCS-4, and UTF-32 with wider code units for Unicode and 10646, this is generalized to two pieces of information, a specification of the width of the code unit, and the number of code units used to represent each character.

Character Encoding Scheme (CES)

A character-encoding scheme is a mapping of code units into serialized byte sequences. Character encoding schemes are relevant to the issue of cross-platform persistent data involving code units wider than a byte, where byte-swapping may be required to put data into the byte polarity canonical for a particular platform. In particular, most fixed-width byte-oriented encoding forms have a trivial mapping into a CES, each 7-bit or 8-bit quantity maps to a byte of the same value. Most mixed-width byte-oriented encoding forms also simply serialize the sequence of CC-data-elements to bytes. UTF-8, since it is already a byte-oriented encoding form, follows this pattern. UTF-16, which involves 16-bit quantities, must specify byte-order for the byte serialization. This is the difference between UTF-16BE, in which the two bytes of the 16-bit quantity are serialized in big-endian order and UTF-16LE, in which they are serialized in little-endian order.

The difference between CEF and CCS is discussed here.

1. The CEF maps code points to code units, while the CES maps code units to bytes.
2. The CES must take into account the byte-order serialization of all code units wider than a byte that are used in the CEF.

3. The CES may involve two or more CCS's, and may include code units that are not part of the CCS per se, but which are defined by the character encoding architecture and which may require an external registry of particular values. In such a case, the CES is called a compound CES.

4. The CES may also make distinctions such as the number of user-defined characters that are allowable.

**Character Maps**

The mapping from an abstract character repertoire to a serialized sequence of bytes is called a Character Map (CM). A simple character map thus implicitly includes a CCS, a CEF, and a CES, mapping from abstract characters to code units to bytes. A compound character map includes a compound CES, and thus includes more than one CCS and CEF. In that case, the abstract character repertoire for the character map is the union of the repertories covered by the coded character sets involved.

Character Maps are the things that in the IAB architecture get Internet Assigned Numbers Authority (IANA) charset identifiers. The important thing, from the IANA charset point of view is that a sequence of encoded characters must be unambiguously mapped onto a sequence of bytes by the charset. The charset must be specified in all instances, as in Internet protocols, where textual content is treated as an ordered sequence of bytes, and where the textual content must be reconstructible from that sequence of bytes.
Transfer Encoding Syntax (TES)

A transfer encoding syntax is a reversible transform of encoded data which may (or may not) include textual data represented in one or more character encoding schemes. Typically TES's are engineered either to:

1. Avoid particular byte values that would confuse one or more Internet or other transmission/storage protocols.

2. Formally apply various data compressions to minimize the number of bits to be passed down a communication channel.

The Internet Content-Transfer-Encoding tags "7bit" and "8bit" are special cases. These are data width specifications relevant basically to mail protocols and which appear to predate true TES's like quoted-printable. Encountering a "7bit" tag doesn't imply any actual transform of data; it merely is an indication that the charset of the data can be represented in 7 bits, and will pass 7-bit channels -- it is really an indication of the encoding form. In contrast, quoted-printable actually does a conversion of various characters to forms like "=2D", "=20", etc., and should be reversed on receipt to regenerate legible text in the designated character encoding scheme.

2.5 CONCLUSIONS

The ISO-10646 and Unicode standards define their repertoires for the written scripts in the world. Here, defining the repertoire is the first step towards a character-encoding scheme. The repertoires are encoded and finally a mapping of characters to code sequence is done for the purpose of computer processing. At the same time since the implementation is difficult in the present state of art because of the processing deficiencies where a character is treated as a single byte, these standards define a
backward compatibility with the help of variable-width encoding forms called transformation formats. The latest languages like Java, Perl supports character definition as a double byte character, which is a roadmap towards the World as a global village.