ISO/IEC JTC/1 SC/2 WG/2
Universal Multiple-Octet Coded Character Set (UCS)
Secretariat: ANSI

Title:	**Proposal: ISO/IEC TR 15285 extension**
Doc. Type:	Expert contribution (pre-notice of agenda item for Beijing)
Source:	Takayuki K. Sato -CICC Japan
Project:	02.15285
Status:	Request of slot in an agenda of Beijing WG2 meeting
Date:	2000-01-05
Distribution:	SC2 WG2

There is a plan to propose an addition of new annex for the TR 15285 Character/Glyph model. The experts from Japan like to express the outline and principle of the proposal in the Beijing meeting. As a formality, it will be a project sub-division request (or NP).

Key of the proposal is: Character (coded element) selection guide for the syllabic composition type of script. And method for improvement of user-friendliness of the syllabic character input under the guide.

The preparation process of the proposal is still going on, thus, I am sending a pre-study material for interested people.

Mr. Convener, please register the following agenda item for the Beijing meeting, final contribution will be submitted in later.

----------Attachment----------

Report from "Output Interface Standard Working Group"

Shuichi Tashiro

Electrotechnical Laboratory

1. Introduction

There are advantages to dividing text processing applications into three basic parts, which are input processing, output processing, and content processing. Implementing these three elements as highly independent modules and standardizing the interfaces between them makes it possible to easily configure a multi-vendor system by combining modules even though the modules are developed by different software vendors.

Text processing in multi-lingual data environments is no exception to this. When there is a high degree of independence among the input processing, output processing, and content processing functions, increased language- independence in content processing and the construction of highly compatible multi-lingual documents should be possible.

The combined syllabic scripts that are characteristic of the South-east Asian and South Asian regions require complex input and output processing. Also, there is great diversity in forms of the character codes and the rules for generating characters from the character code strings, and it is, in many cases, difficult to separate input processing, output processing, and content processing.

Also, in the combined syllabic script region, character code systems have, for the most part, conventionally been designed assuming eight-bit bytes because of hardware resource constraints, further complicating the problem.

2. Problems associated with output processing

The output processor, simply speaking, is a system that accepts a character string, selects the appropriate glyphs for it, and arranges them in proper fashion on the display screen. For the Latin alphabet and kanji characters that have been used in computer text processing for a long time, characters and glyphs were, fortunately, more or less the same, so little complexity was required of the output system functions, and there was no deep discussion concerning the definitions of characters and glyphs. When dealing with complex Asian character systems, however, there are many unresolved issues concerning how to define characters and glyphs, and how to set the rules for converting between the two. The ISO IEC JTC1 discussion concerning this problem is entitled "An Operational Model of Characters and Glyphs" and has been published as DTR15285. However, it is still at the stage of defining terms and establishing a starting point for discussion. As for output processor architecture or design, progress in this discussion is indispensable. It is important to investigate conversion rules, external interface specifications, etc., and decide on an overall architecture on the basis of a sound discussion of those matters.

1) Handling combined syllabic scripts

Particularly in Asia, there are many languages, including Thai, Devanagari, and so on, use

character sets that are referred to as combined syllabic scripts. In ISO-10646, the glyph elements that represent the components of which combined syllabic scripts are constructed, that is to say, phonemes, are most often considered to be characters, and character codes are assigned to them. Devanagari script, for example, combines 94 glyph elements to produce over 2400 characters. An exception is the Korean script. Korean script, too, can be called a kind of combined syllabic script in which multiple glyph elements that represent consonants and vowels are combined to form characters, but in ISO-10646, the more than 7000 glyphs that result from combining glyph elements rather than the glyph elements themselves are defined as characters and codes are assigned to them.

In combined syllabic scripts, when codes are assigned to character components, which is to say, glyph elements, the output processor receives from the content processor code strings that are composed of multiple codes of unspecified length and must display the character glyphs that correspond to that code string character.

The problem here is that the substrings that correspond to characters must be extracted from a continuous string of codes, which is the problem of detecting code string boundaries.

If the delimiters are defined within the code, as they are in IS (Indian Standard) -13194, for example, and the boundaries of the substrings that correspond to characters are clearly shown by them, this problem is easily solved. Not all of the current code systems, however, are structured in this way.

If the rules for determining glyphs from code strings are clear, substring boundary detection is not a problem, even without delimiters. If there is ambiguity in those rules, however, multiple different glyphs may correspond to the same code string, and it may be impossible to determine which of the multiple glyphs should be selected. This kind of ambiguity actually has be inherited in ISO-10646, which has been incorporated as-is in the national standards of various countries.

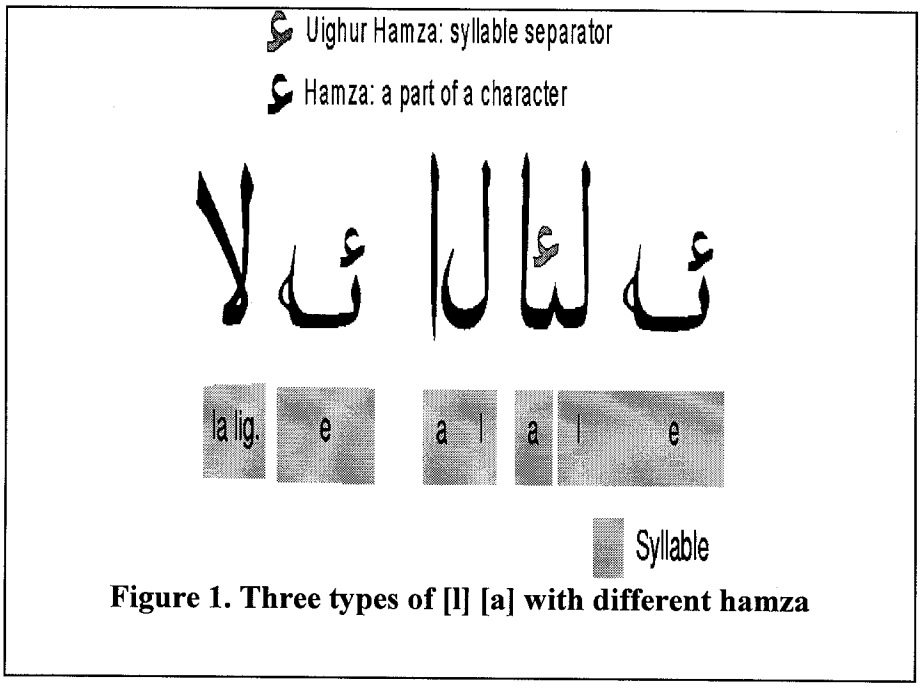

Figure 1. Three types of [l] [a] with different hamza

The Arabic character set is an example of the case in which there can be multiple character glyph

candidates for a single code pointer string. The Arabic character set is used in the scripts of Arabic, Uigur and other languages. As shown in Fig. 1, there are three characters in the Arabic character set for representing [l][a]. In the Arabic language, it is necessary to represent the ligature on the left side of Fig. 1. The two characters on the right in that figure are not used in Arabic, but are

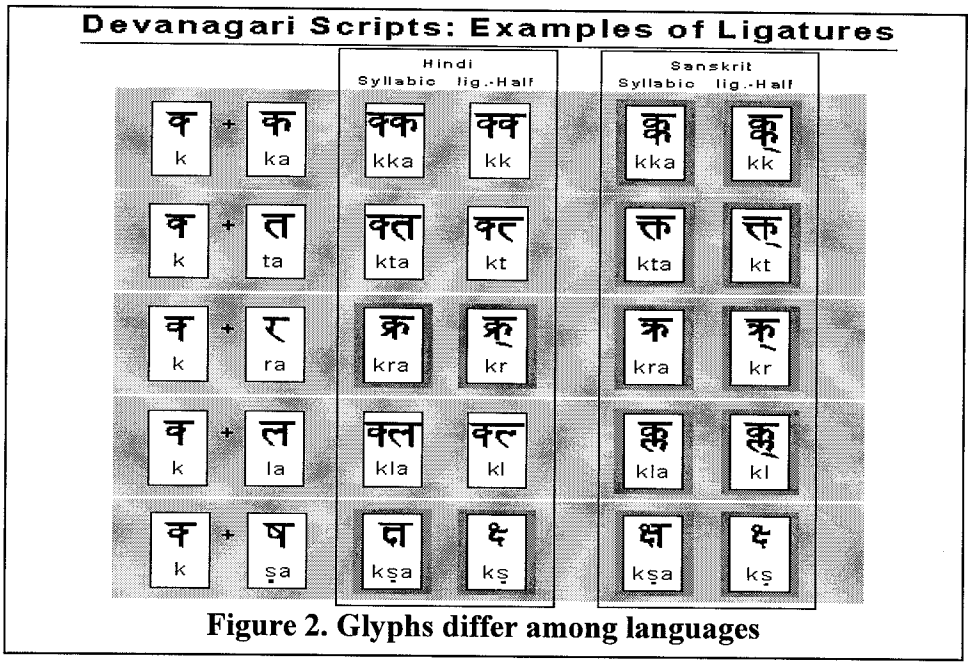

Figure 2. Glyphs differ among languages

used in Uigur and in Arabic character set that serve as script for non-Arabic languages. Accordingly, if the output processing system receives only a code pointer string, it is not possible to determine which glyph should be displayed.

The Devanagari characters are used in the scripts of the Sanskrit and Hindi languages. As shown in Fig. 2, even for the same Devanagari character, the ligature character glyph element differs from language to language, and the glyph cannot be selected from the character code alone.

As we see from these examples, when multiple languages use the same character code set, it is not possible to select the appropriate glyph unless the output processor possesses knowledge of the language being used for the given character string. That is to say, the output processor must be equipped with a database of rules for converting characters to glyphs for each language that is handled, and, so as to be able to select the appropriate database, the language of the text that is currently being dealt with must always be known. There is a need for study of the interface specifications that define how such information, which is referred to as context, should be provided to the output system.

An example of ambiguity in the Thai character set is shown in Fig. 3. In that example, the same character code string, [p][e:][l][a:], can be interpreted as either the two characters [pe:][la:] (axle) as shown on the left side of Fig. 3, or the one character [plaw] (hour) as shown on the right in the figure. This ambiguity can be resolved by determining the meaning from the context, and so the choice between the two-character candidate and the one-character candidate can be made. For the one-character [plaw],

a carriage return cannot be inserted among the four glyph elements, but the output processing system cannot determine this from the code string alone.

The Tibetan character set also provides many examples of the difficulty of determining glyphs from the character code string alone.

The candidate characters shown in Fig. 4 are entirely the same in terms of glyph elements, but they are two different words that have different pronunciations, [ma][ng][s] and [m][nga][s]. Which character is correct is determined by the context. This is because Tibetan characters are a consonant-connected type combined syllabic script, and the basic syllabic characters of Tibetan script are combined with other basic syllabic characters, without vowels and with only consonants remaining. When the vowels are dropped, the form of the character changes in the case of Devanagari script, but for Tibetan script, there are many characters that do not change, so, as the example in Fig. 4 shows, cases occur in which the character cannot be determined from the glyph elements.

In this example, the first character of Type 1 has a vowel and is the core of the glyph; in Type 2, the second character has the vowel and is the core. In the transliteration of Tibetan characters, this is always distinguished as [ma][ng][s] and [m][nga][s].

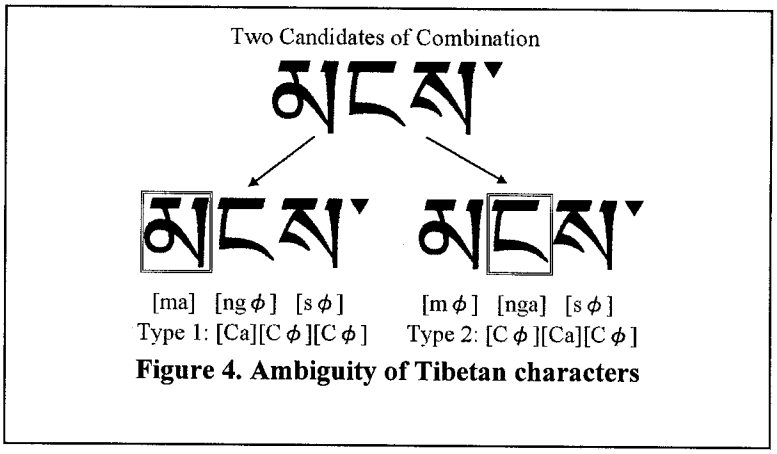

Figure 4. Ambiguity of Tibetan characters

When the glyph is composed at the time of glyph element display in the character output, it may be possible to mitigate this kind of problem (at least for the character codes used in the interface between content processor and the output processor) by an approach in which glyph elements are combined into phonemes and codes are assigned to them. In that case, however, the problem of a very large number of code pointers arises. Also, for the user, there is a need for text manipulation in units of the glyph element in the character input and correction process. To satisfy that requirement, the output system must continue to deal with problems related to glyph composition.

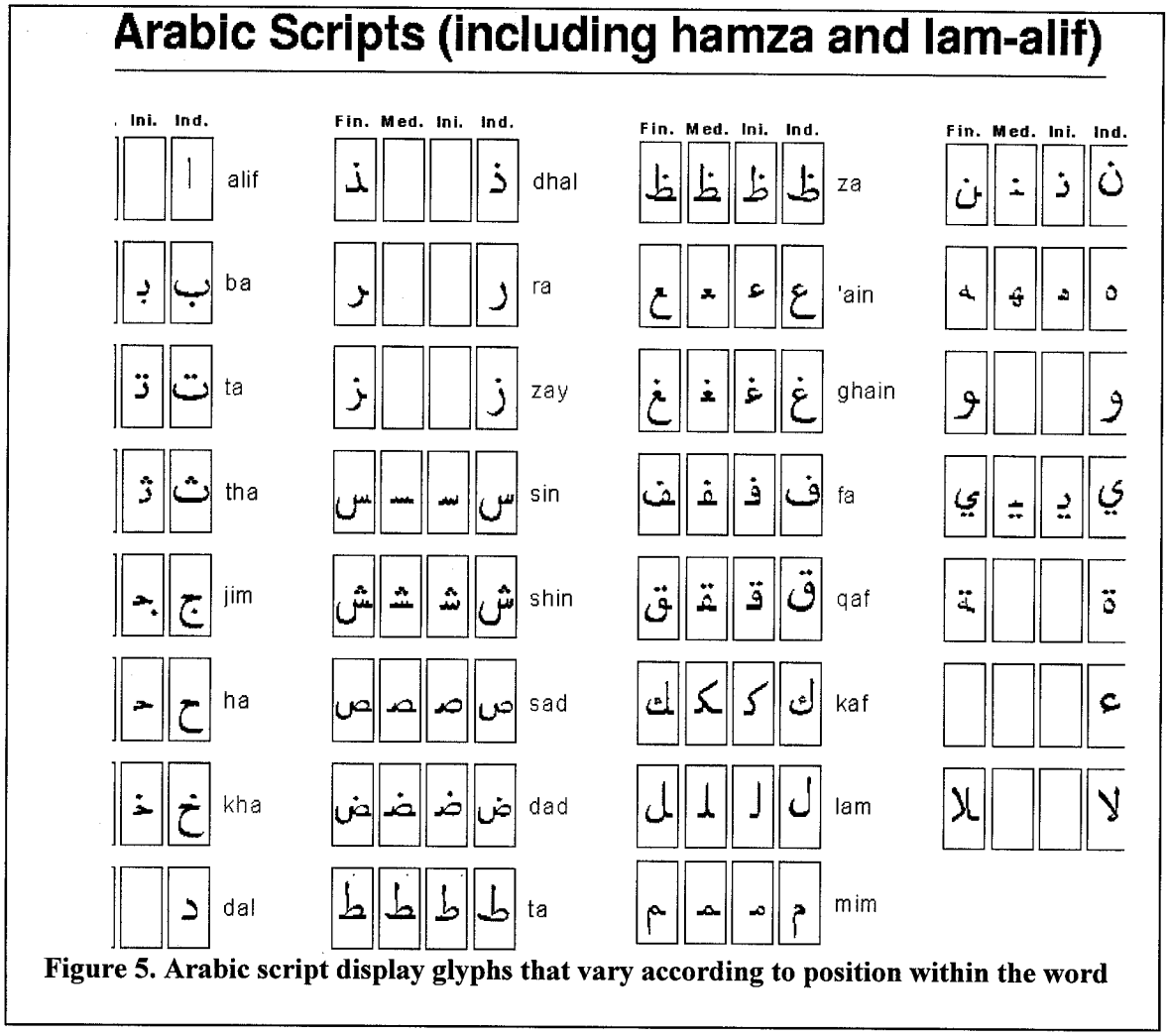

Figure 5. Arabic script display glyphs that vary according to position within the word

2) Handling position-dependent characters and direction-dependent characters

For Arabic characters, Mongolian characters, Tibetan characters and other such character sets, the form of the representation of a character may change depending on its position due to the formation of ligatures and so on.

For the characters of the Arabic script, glyphs are selected according to the position in which they are written, i.e., word initial, word medial, word final, and independent. The word initial, word medial, word final, and independent glyphs for the Arabic font are shown in Fig. 5.

Also, the forms in which some characters appear may differ according to whether the script runs is vertical or horizontal. The parenthesis symbol, "(", is an example. In such cases, too, opinion is sometimes divided as to whether these different forms should be dealt with as different characters or should be treated as the same character and have the appropriate glyph selected automatically for display in the output processing. Position dependence rules and direction dependence rules also vary from language to language, so the output system must have a function for selecting the database to be used for the language that is being handled from among the various databases that are available to it.

3) Representation of transitional forms in the input process

For input of a combined syllabic script, the input style in which glyph elements that correspond to phonemes are arranged on the keyboard, and characters are composed by pressing the respective keys in order is said to be user-friendly. With this input style, each keystroke produces an incomplete character on the display screen, and it is desirable for that process of gradual building-up of characters to be displayed as it is. Even if the completed character glyph is a forbidden combination, it is displayed in the input process and it is necessary to prompt for correction. The output processor must have a function for representing such transitional forms. Also, internal representation code for communication of the input processor, the content processor, and the output processor, too, must be able to cope with the representation of those transitional forms and the processing for them.

4) Deciding whether to regard several characters that differ slightly in form as different characters or as variations of the same character

In ISO-10646, for example, the numeral, "1", and the circled numeral one, "①", are considered to be different characters and are given different codes (0031 and 2460, respectively). This approach, however, means that there is not necessarily a single definition for the character and glyph. Another approach is to consider "1" and "①" to be modified forms of the same character rather than different characters. There are probably times when, in the content processor, it is more convenient not distinguish between these and to have a single character code. Sorting and automatic translation processing are examples of that. In this case, the character corresponds to the concept of the numeral "one", and for the glyph, "1" or "①" may be selected according to the case. The recent word processing software that runs on computers that employ bit-map displays routinely select among various different forms for the same character, such as bold or italic typefaces. Considering that, it is natural to think that the distinction between "1" and "①" can be made by selecting among character types in the same way.

3. Requirements for the output processing system interface

The previous section described various factors that are involved in the inability to determine the glyph that should be output when the output processing system receives only a character code string that is based on current character code specifications. It was pointed out that rather than displaying the completed character glyph, particularly for combined syllabic characters, it is necessary for the output processing system to compose the character, displaying the transitional incomplete character glyphs as the person presses the keys on the keyboard that correspond to the element signs.

As a result of considering these matters, the requirements for the specifications of the interface between the output processing system and other modules are listed below.

I. When a continuous character code string is received, it is possible to uniquely determine which part of that string (from where to where) corresponds to a single glyph.

II. Which glyph should be selected for one code string does not depend on the language, or the means of conveying the linguistic context of the dependence is standardized.

III. Which glyph should be selected for one code string does not depend on the direction in which the script is written, or the means of conveying the directional context of the dependence is standardized.

IV. Which glyph should be selected for one code string does not depend on the position in which the glyph is written, or the means of conveying the positional context of the dependence is standardized.

V. For incomplete transitional character glyphs such as arise during the input process of compound syllabic scripts, too, there is a standardized means of determining them.

Along with further study of these requirements, there is a need for the design and standardization of an inter-module data exchange character code and an accompanying control code for it.

This standard specification should be designed with emphasis on efficient internal text processing in information systems, with the possibility that the data exchange specifications for communication between remote systems and the data exchange specifications for long-term storage in files and so on are separate specifications.