L2/01-098

ISO/IEC JTC/1 SC/2 WG/2 N2324

2001-02-12

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

Title: Information necessary for TR 15285 amendments drafting

Doc. Type: Expert Contribution

Source: Takayuki K. Sato -CICC Japan

Project: 02.18

Status: For review and use by the project editor, and for the discussion at #40 WG2 meeting

Date: 2001-02-12

Distribution: TR 15285 editor, JTC1 SC2 WG2

References: WG2 N2148, N2198, N2199, N2206

Medium

This is a base paper to be used by the project editor.
Also, this paper can be used for review of a principle of the amendments.
This is not a "draft proposal" of the amendments for the TR 15825.
Original Proposal and some technical information are available in documents WG2 N2148, 2198, 2199 and 2206. For better understanding of this paper, please refer to the four documents.

The N2198 proposes two amendments. Namely amendment-X and amendment-Y

Amendment-X.

1. Purpose

This amendment is intending to provide a guideline or hint to select the coded elements to develop a coded character set for a script.

2. Problem statement

In principle, character code to be assigned an “atomic elements” of the target script. Then a sequence of the “atomic element” may represent a text or a data.

The “atomic element” means a minimum symbol of the script, which is normally called a “character”.

Note: This “character” is not same character that is defined in this TR.

By history, the “atomic elements (character)” is defined far before computer in linguistic viewpoint, and the characters are taught in elementary school. For the human writing purpose, the atomic elements fit well because human being uses its intelligence to write by using the atomic elements.

On the other hands, this may not always true for writing by computer. Because computer is not intelligent enough as human being, the same set of atomic elements may not be enough to write and process a text or a data by computer. Computer may make a mistake on the layout or representation of the atomic elements, and may misunderstand the sequence of the atomic elements as different meaning from the original intention of the data creator.

Therefore, the selection of the atomic elements for computer (coded elements) may different from for the human being for some script.

This is not big concern on Alphabet or CJK ideographs. This concern is particularly important for the syllable composition type characters, which normally pick consonants and bowels (and tome marks and so on..) as an atomic element and combine them to form a syllable.

There are many variety of method possible for selection of the atomic elements (coded elements). However, from computer processing purpose, the impact of the selection methods should be as small as possible. If each script selects their special method, then the process to handle the many different scripts might become very complex (and it may not be practical way to select the variety of methods). This is why the guideline for the selection is necessary.

3. Goal of guideline.

There are two goals of the guideline. One is to eliminate any redundancy of the data string for the purpose of data processing and data presentation. And another is to make IT products as independent as possible from script and culture, thus resultant character code should be as similar as possible.

Note: Most of the coded character sets are already taking care of the concerns. For most of the

existing coded character sets are not necessary to be reviewed by this paper. This paper is written for future developments.

There are three kinds of redundancy to be overcame:

- Redundancy in sequence of the coded elements
- Redundancy of processing/rendering unit.
- Language dependency.

4. Redundancy in sequence.

In a computer system, a string of same character codes should be recognized as the same data. Different sequences are, therefore, recognized as different to each other data. However, sequence of atomic elements may not be same always in case of handwriting, even though the sequence is different hand writing result may be recognized as the same data. However, in case of computer different sequence are recognized as different data and may represent different shape.

For example, character code for diacritical marks comes after base character in case of ISO/IEC 10646, besides the base character comes after diacritical marks in case of ISO/IEC 6937. The rules are different each other (working well for each character set), but, within one character set the two rules cannot be mixed each other.

In case of syllabic character, this is important. Syllable is described by sequence of characters (consonant, vowel and marks). For pronunciation of the syllable, the order of consonant and vowel is clearly defined (logical sequence). However, in writing system, there is some flexibility of the sequence. In case of the most South Eastern Asian syllabic scripts, vowel is placed around the consonant (top, bottom, right and left!!). Therefore, in case of writing a syllable, some vowel should come left side (in case of left to right writing, it is in advance to the consonant) of the associate consonant. There are two ways of writing these scripts, one is write left vowel after consonant as if it is pronounced (the most of school teach this method. Backward to the writing direction), and another is to write the left vowel first? then consonant by keeping writing direction constant (mechanical typewriter, many real hand writing). Both do not make any difference on the paper, but in character code sequence, they are different data. This is a problem.

From computer processing purpose, it is necessary to have one, only one, rule for the sequence, unless, resultant data have a redundancy.

Requirement: A rule of the sequence of the coded elements should be defined and stated to

avoid the redundancy.

Example:

Thai: <U+0E32> comes before consonant. However, from the surface of code table, this rule is not visible, <U+0E32> might be coded after consonant by someone (this is redundancy). To avoid the redundancy a rule (xxxx) is defined. The rule says proceeding vowel first.

Devanagari: <U+093F> comes after consonant. Because <U+093F> is defined as combining character, it comes after consonants always. This is another type of the rule.

Latin: ISO/IEC 10646-1 defines its own rule.

Note: the smaller number of the rule is the better. Thus the existing rules such as Thai model and Denagari model and Tibetan model might be referred here.

Question to editor:

More samples necessary?

Need some figures to explain above?

5. Redundancy of processing/rendering unit.

In case of those sequence of atomic elements, not only each atomic elements, sometimes a sequence of the atomic elements should be handled as a data processing/representation unit. There should not be a redundancy to pick the unit.

Spelling "KANI" in Japanese ca be either "KAN"+"I" or "KA"+"NI" in syllables.

"KAN"+"I" in Japanese is written as <U+xxx>+<U+xxxx> (Simple), and "KA"+"NI" is <U+xxxx> (crab). Therefore, data and representation are different for same spelling.

This sample case indicates that there are possibility of three kinds of problems.

- Rendering unit separation (selection of font),
- Data processing unit separation
- Word separation (Note that most of Asian text in Asian script do not have word separation like SPACE in Western text)

In Hangul, there are atomic elements called JAMO. There are 24 of (basic) JAMO (14 consonants and 10 vowels). In theory, All Hangul syllables can be composed using the basic Jamo. For human writing, this is almost true, however, it does not work well for computer.

Sequence of JAMO <U+1109><U+1161><U+1109><U+1109><U+1109><U+1161> may mean

either <U+C0BF><U+C30B> or <U+C0C0><U+C0AC>. This is redundancy. Basic JAMO only is not enough for computer. This sample includes both rendering redundancy and syllable separation (for data processing) redundancy. In real application, Introducing double consonant JAMO <U+110A>(= <U+1109>+<U+1109>) resolves the redundancy problem.

To the editor:
It may be better to have some more practical sample by having help by Korean native.
Sato is surveying if there is same kind of sample in Khmer or Lao.

In Mongolian, determination of presentation form for some Mongolian characters is not possible by seeing the code sequence (unlike Arabic). This is representation redundancy.
Because, there is no logical way to pick the presentation form. In case of ISO/IEC 10646, introduction of Mongolian Free Variation Selectors and Mongolian Vowel Separator is a solution of this redundancy problem, Those Selectors or Separator are not introduced by linguistic reason but by computer reason.

In some application, data processing by syllabic unit may be better. For example, in case of text editor, for some script, edition such as deletion might be better by syllable by syllable rather than atomic element by atomic element. Deletion of one atomic element out from regular string of atomic elements (means regular text) might cause irregular string for a syllable, which may confuse an operator. To avoid this kind of problem, it is better to manipulate syllable by syllable such that the target string is regular always.

Requirement:
Review a sequence of conventional (linguistic) atomic elements whether if the sequence can be segmented into a sequence of data processing and/or rendering units (syllable unit in most cases) without ambiguity.
It is important to review special cases. In most script, there is no significant ambiguity in case of simple cases. However, it may happen in special cases.

If there is no ambiguity, the atomic elements are coded elements to be. This is the most easy and likely case. If there is a possibility of ambiguity, then use one of following methods the design of coded elements based on the linguistic atomic elements to avoid the ambiguity of processing ambiguity of the sequence of the coded elements.

a. Define some of the atomic elements as a combining character. or
b. Define some of the atomic elements into two kinds of character, one as a regular character

and another as a combining character.

c. Define some combination of the atomic elements as a coded elements in addition to the linguistic atomic elements.

d. Define (or borrow from other scripts) units (syllable) separator in addition to the atomic elements for the separation.

e. Define (or borrow from other scripts) glyph selector(s). Note: the glyph selector might be used as a separator sometimes.

A condition for selection the method out from above is "similarity with existing method" and "simple ness". If the unit separation method is complex and totally new, usage of the resultant character code would be very inefficient. (and might be difficult to adopt)

f. If such method is not suitable to add an existing character code, then "pre-composed method" should be selected.

What else?

For editor: do we need a sample case above?

6. Language dependency.

If there is language dependency for selection of the unit or rendering, it is necessary to add information on the language.

For editor: does sample case necessary?

Note: Hindi language and Sanskrit language have different glyph for same Devanagari script.

The language identification might be out of scope of the coded character set; it should be handled at out side character code. But the identification method may use a character code such as tagging character or escape sequences.

--end of amendment-X base document---

Amendment- Y.

1. Purpose

This amendment is intending to explain a need of input assistance and a necessary functional requirements for the input assistance interface.

2. Problem statement

The separated paper describes a need to avoid an ambiguity of a sequence of the coded elements. It requires a normalized sequence and new coded elements in addition to the atomic elements defined in linguistic requirement (which may be taught in elementary school).

The sequence of coded elements should be friendly with computer, but may not be friendly with human being. This is good for computer (that is why it is proposed in the separated paper), but not good for human operator who needs to generate the string. This is a problem.

Traditionally, computer keyboard has a set of keys, which has one to one correspondence with the coded elements. Then a sequence of coded elements has been developed by hitting the keys for coded elements. However, once machine friendly (but not friendly with human) coded elements are provided per proposal in separated paper, input operation would be not friendly with human. This is a problem.

In fact, because there is a need of compromise between elements on keyboard and internal computer convenience, human/machine unfriendly coded elements has been developed in many cases.

To make input operation human friendly and coded elements machine friendly at the same time, it is necessary to make coded elements and input elements independent each other such that the both elements become friendly for each requirements.

The Input Assistance (a kind of extended keyboard driver) software (IA) provides a solution on this requirement. IA bridges keyboard and coded elements (means IA generates target character codes from key board operation).

For example, romanized Japanese word input can be converted into CJK Ideograph code string by using Japanese IME software. In this case, key operation is friendly with human but does not related with the code string directly, but IA converts the input sequence of the Latin characters into CJK ideograph string, which is friendly with computer.

3. Conditions

IA should be different from script to script as well as language-to-language by definition. In addition even if within the same language and/or script, IA should not be only one. There may be variety of Ias according to application and also new IA to be developed as technology progress. To make this happen, standard interface as international (or defacto) standard is desirable.

4. Recommended basic functionality and features of the IA including standard interface .
note: details are in SC2 WG2 N2206.

a. IA internal code should not be same as key code and may be different from IA out put code.
b. IA should have an ability to display the input mode if multiple input modes are available in the IA. And also the mode is selectable by an operator and application (Thus, the interface for this control is necessary).
c. IA should have "by pass" capability for some selected key.
d. Application program has to have a control (on/off etc.) capability of the IA. (Thus, interface for on/off is necessary).
e. Repository of locale (or cultural convention set) information of the system should be available for IA (Thus, the interface for the information is necessary).
f. Application program should have a reset capability of IA (Thus, interface for reset is necessary)
g. IA should have a capability to send mode information to the application. (Thus, interface is necessary)
h. IA may use INPUT WINDOW if it is necessary (Thus, INPUT WINDOW control interface is necessary)
i. Application program should specify that the INPUT WINDOW should be displayed or not.
j. The INPUT WINDOW should have a capability to display a intermediate (while input process) code sequence if necessary.
k. Application should have a control on attribute (font size, typeface…) of the intermediate code sequence.
l. Coded character set for the INPUT WINDOW should be standardized

For editor: do you want to write more details? Do you need more information on above?

---end of Y---and end of all---