4S－5

Multilingual I/O and Text Manipulation System (3): Extracting the Essential Informations from World's Writing Scripts for Designing TMC and for the Generalizing Text Manipulation

Tomoko Kataoka*, Tadao Tanaka†, Kazutomo Uezono†, Yutaka Kataoka* and Hiroyoshi Ohara†
* Centre for Informatics, Waseda University † School of Science and Engineering, Waseda University

1. Introduction

The rapid growth of the occasions of both international and national communications among people from various countries urge the computer environments nowadays to ensure the unambiguous interchange or processing of necessary informations. Most current approaches, however, especially the one with the Locale model, are apparently going backward, so impeding the sound development of a computer in terms of its *computability* and *interoperability*.

Suppose you mix unlimitedly all character sets on a computer, which means its *Multilingual* environment [1]. Mb is motivated as a unit for information exchange, as defined in ISO 2022 [2], but its codepoint does not necessarily correspond to one character, because of its extension with the *control sequences* and the *code extension techniques* defined in ISO 2022/6429 [3] and with *determinatives* [5]. Nor is WC a unit to process as a character, although POSIX, X Intrinsics and X Widgets assume so. Posix defines WC as an *implementation dependent* internal code, so it is not appropriate for codeset independent multilingual text manipulation.

In the following sections, the concept of TMC (*Text Manipulation Code*) and the informations necessary for proper text manipulation are presented, a generalization of text manipulation.

2. TMC: Process Unit for Text Manipulation

Proper handling of any text would require an appropriate minimum *unit* that is processed by character/text manipulation functions. A WC codepoint having a glyph is redefined as being mapped to one codepoint of a final glyph (1 to 1 correspondence) [4,5], though there are several cases as TIS 620-2533:1990 that more than one final glyphs constitute one character, so failing to be used for text manipulation. Rather, WC should be user definable and its codepoints must be unique through all locales to avoid locale dependency, but still is not ensured to stand for one character.

The required process code for text manipulation is called TMC. A codepoint of TMC is converted from a codepoint of WC (and via WC from mb) by applying certain explicit conversion rules. As discussed in the first paper of this series [4], there needs a flexible mechanism to allow multiple TMCs, one of which must stand for normalized one character to be processed–TMC-ID 0 provided by our system, satisfying the expected demands from application softwares. Therefore, application softwares can be designed as multilingual by TMC-ID 0 and TMC character/text manipulation functions.

Characters are classified into the following; 1) Phonemic, 2-1) Pure Syllabic, 2-2) Conjunct Syllabic, or 3) Ideogramic. Each type of the characters can be described by BNF. An apparently infinite combinations of symbols in TIS 620-2533:1990 are also proved to be computable based on BNF. It defines one syllabic to be processed, thus enabling our system to convert among mb/WC/TMC-ID 0, just like all the other scripts. Conjunct Syllabic scripts have two layers, the syllabic layer to be pronounced and the phenemic layer, a component of a syllabic. The process unit of *Thai* in TIS is syllabic, while the one of *Devanagari* in IS 13194:1991 is phonemic constructing a syllabic.

One exception is ISO 10646 with *Non-Spacing Character*, which permits the unlimited definitions to specify one glyph. Putting Non-Spacing Character aside, the *unified* codepoints in ISO 10646 make a non-interoperablity among other standard code sets. Thus, there is only one member of the *unified ideographs*, for which correct reverse-conversion to the original codepoint of each GCS would not be supported. Furthermore, it does not specify the multiple different rule sets of extensions according to those writing scripts derived from Brahmi Scripts. Consequently, ISO 10646 is not suitable for the required process unit for TMC nor for communication.

3. Internal Structure of TMC and TMC-ID 0: Normalized One Character

A codepoint of TMC has two bit fields: a) the *Character-ID field* generated from GCS/CCS-ID, font-ID, and glyph-index of the WC codepoint(s), and b) the *Attribute field*. The Attribute field holds informations about what type the character in question belongs to, eg., 'Ideogram/Not' or 'Space/Not', available for the generalized basic text manipulation functions.

Here is provided TMC-ID 0, the normalized 'one character', among other TMCs that are also process units of text manipulation. TMCs are definable codes and may have various types, but at least TMC-ID 0 is ensured to be normalized one character, and is reversible to a codepoint of WC, given necessary informations.

The Attribute field, the bit field for character information, enables the TMC and the text manipulation functions independent from any code sets. Before TMC, any character was only given a codepoint, a digit, and we had no way to know the required information directly, like 'This character is a comma'. Ctype/wctype functions of POSIX, however, are shown to be no valid for code sets

for certain numbers of non-Roman scripts in the world.

Likewise, there is no garantee for functions X{mb|wc}LookupString to return a string for a unit to be processed, which means there would be no *reliable* Text Widget. Thus, one is encouraged to check the behavior of the functions by calling {mb|wc}ToTMC(TMC-ID – 0 here, length, {mb|wc}-String, TMC-string) every input time to know if really one character is read – the function returns T if succesfully converted.

The character informations, which TMC Attribute field holds, are extractable for use by calling the *Character Information Functions* with attribute names and TMC-ID. Those functions include; 1) QueryTMCInformation which returns attribute values, 2) ChangeTMCInformation which changes attribute values, and 3) QueryTMCCtrlCName which returns *Control Function Name*, if given element has such information. Sets of all attribute names associated with a TMC are stored in the Meta Converter System date area, and a set of all names can be aquired by the function of QueryTMCAttribute.

4. TMC Text Manipulation Functions

Generalization of the text manipulation means to make available the code set independent text manipulation operations. Such operations include *Insertion*, *Deletion*, *Comparison*, *Search*, *Replacement* and *Line Separation*. In other words, these functions will be called, one or some of them jointly, when manipulating the texts.

To detect the defining factors, for instance, for line separation and the relating writing convention of 'Prohibition of putting certain characters at the top/end of a line,' three bits of the Attribute field are prepared for TMC-ID 0, ie., 1) Permission of line separation, 2) Permission to put it at the top of a line, and 3) Permission to put it at the end of a line. The bit of 1) plays the role for giving candidates for the point of line separation and also when user/application forces to separate lines at the point. TMC operation of Line Separation makes use of these three bits. Finding Line Separation is essential for a Text Widget to be international.

Giving another example of text manipulation, there are more than one ways to search a certain string from the texts, which means the operations are user-customizable. We prepared two Text Manipulation Functions for Search, one of which refers not only to index values of TMC but to additional character informations.

The TMC Functions are categorized into the *Basic TMC String Functions* and the *Essential Text Manipulation Functions* [6]. The TMC String Functions are; 1) TMCStringLength which returns the length of the given TMC string length, 2) TMCStringCopy which copies a TMC string to a destination. A TMC string concatenation function can be derived from these two.

The essential Text Manipulation Functions are as follows; 1) InsertTMCString which inserts a TMC string into another TMC string, 2) DeleteTMCString which deletes a range of TMC string in a TMC string, 3) CompareTMCString which compares two TMC strings with Character-ID field only, 4) ExtCompareTMCString which compares two TMC strings with Attribute field information(s) and with Character-ID field, 5) SearchTMCString which searches a TMC string in a TMC string with Character-ID field only, 6) ExtSearchTMCString which searches a TMC string in a TMC string with Attribute field information(s) and with Character-ID field, 7) ReplaceTMCString which replaces a sub-string in a TMC string matched to a TMC string-1 by a TMC string-2 with Character-ID field, and 8) ExtReplaceTMCString which replaces a sub-string in a TMC string matched to a TMC string-1 by a TMC string-2 with Attribute field information(s) and with Character-ID field. These functions adjust Position Dependency Attribute fields.

5. Concluding Remarks

The Architecture is conceptually mapping a set to another set with informations, thus all functions are provided by the Meta Converter System. Based on the definition of TMC-ID 0, designing/implementing a Multilingual Widget is now on progress. And a Multilingual LISP is also under way. Various types of combinations of TMCs and the TMC text manipulation functions proposed here will result in many text manipulations for application softwares. One of which might be a collation ordering, which also must be user-definable TMCs.

References

[1] Kataoka, Y. et al., A model for Input and Output of Multilingual text in a windowing environment, ACM Transactions on Information Systems, Vol. 10, No. 4, October 1992, pp. 438-451.

[2] ISO/IEC 2022:1986, Information Processing – 7-bit and 8-bit coded character sets – Code extension techniques.

[3] ISO/IEC 6429:1988, Information Processing – Control functions for 7-bit and 8-bit coded character sets.

[4] Kataoka, Y. et al., Multilingual I/O and Text Manipulation System (1): The Total Design of the Generalized System based on the World's Writing Scripts and Code Sets, Proceedings of the 49th General Meeting of IPSJ, September 1994, 4S-3 (In this volume).

[5] Uezono, K. et al., Multilingual I/O and Text Manipulation System (2): The Structure of the Output Method Drawing the World's Writing Scripts beyond ISO 2022, Proceedings of the 49th General Meeting of IPSJ, September 1994, 4S-4 (In this volume).

[6] Tanaka, T. et al., Multilingual I/O and Text Manipulation System (4): The Optimal Data Format Converter to/from MB/WC/TMC, Proceedings of the 49th General Meeting of IPSJ, September 1994, 4S-6 (In this volume).