Internationalized Text Manipulation Covering Perso-Arabic Enhanced for Mongolian Scripts

Tomoko I. Kataoka[1], Yutaka Kataoka[1],
Kazutomo Uezono[2] and Hiroyoshi Ohara[2]

[1] Media Network Center, Waseda University
[2] School of Science & Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, 169 Tokyo, Japan

Abstract. Although multi-lingual text processing is required for digital libraries, language education, and natural language processing, it has been difficult to realize for the confusion of a glyph with a character. This paper defines the relation between a glyph and a character, and it describes the processing of Mongolian scripts, an extended case of Perso-Arabic scripts handled by our system in ways which generalize to many complicated scripts. Mongolian scripts have particularly complicated orthographies and are almost impossible to encode. However, separating glyph defining information from encoding position solves some important problems arising from these and other scripts (including mixed languages) which may require multiple direction rendering. The study of many scripts led us to store, attached to the Wide Characters of POSIX, attributes which support not only the information for text manipulation (to be applied to a character) but glyph information as well such as variant and position necessary for display. Moreover, the information which is not available in a character code is provided from the database of our system to be embedded into a WC's attribute.

1 Introduction

Mongolian language is usually transcribed with the traditional scripts, the Cyrillic script and sometimes with the Latin script. Cyrillic script was introduced after the WW-II, and there are needed at least three independent systems with the different numbers of alphabet for Cyrillic writing. Historically, various other scripts were used as shown in Figure 1.[1]

Traditional Mongolian script (hereafter just called the Mongolian script) is written vertically from top to bottom, with left-to-right line feed, because its ancestor, classic Uigur script belonged to the right-to-left Perso-Arabic script family, with top-to-bottom line feed.[2]

[1] An arrow added to each script name shows the direction to which it is written. Paspa script was invented by Paspa, a Tibetan, in the age of the Yuan Dynasty, to be intended as the International Phonetic Alphabet at that time. This script is not discussed here, but we only add that it belongs to the Devanagari Script group. Manchu, the official literal language in the Ching Dynasty, and its descendent Sibo have their base on Mongolian script: Mongolian script family.

[2] When introduced for Mongolian people in 13c., the classic Uigur script, which itself was borrowed from the Sogdians in 8c., turned 90 degrees to the left and was written vertically from the top. For the present Uigur script, see below.

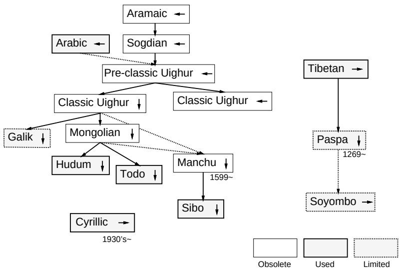

Fig. 1. Mongolian-related Scripts

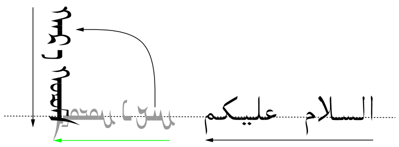

Fig. 2. Mongolian Is a Vertical Perso-Arabic

Like other Perso-Arabic, a character changes its glyph shape according to the position in a word. This means correct glyph changes must be ensured after the operations of insertion and deletion (Figure 7).

Mongolian script has different types of ligatures of Perso-Arabic. The so-called ligatures including ‘l’ or ‘m’ are that just a part of that glyph extends to the preceding/following glyph space. And those with ‘b/k/g/f/p + vowel’ sequence do change shapes, while they have no ‘uncomposed’ counterpart meaningful sequence unlike Arabic (Figure 9). Thus, there should be no codepoint for making ligatures.

Mongolian script is not monolithic, but it is a general term covering three different sets: Script for Mongolia, Scripts for Inner Mongolia and Todo, Oirat Script, according to a dialect and/or a region of writing.[3]

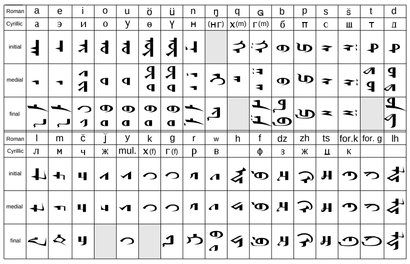

Fig. 3. Scripts for Mongolia

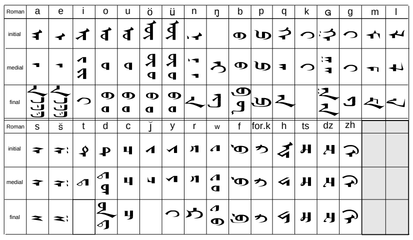

Fig. 4. Scripts for Inner Mongolia

Typical vertical Mongolian shown in Figure 3 and 4 has several problems. Mongolian script is much like Arabic, but it has more complicated orthographies. One cannot always tell which sound is meant just looking at a glyph (Figure 6). Also notice the same is true for ‘a’ and ‘e’, ‘q’ and ‘G’, and many others.

A position information is not always enough to specify final glyphs, and there can be some variants even for the same form, e.g. Medial (between the beginning and the

[3] Inner Mongolian style is based on handwriting and some people in Mongolia also favor it over the script in Figure 3.

Fig. 5. Todo Script

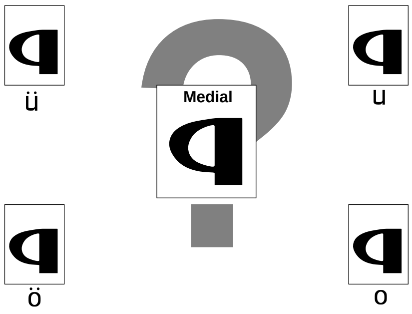

Fig. 6. Ambiguities of Mongolian Script

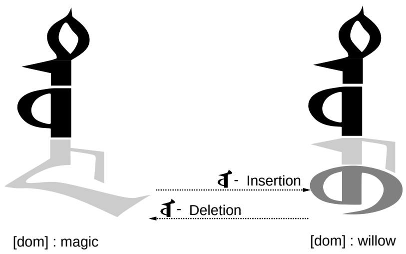

Fig. 7. Correct Glyph Changes Must Be Ensured

end of a word), according to various factors. See the final four variants in Figure 4. That is why these scripts are sometimes called *Hudum*, “not exact” compared to *Todo*, which means “exact” in Mongolian.[4] Users should get output in the variant they desire in some occasions. Hence variant selection mechanism must be introduced as well as that of form selection, which was already required for Perso-Arabic scripts.

It is obvious that character definitions in terms of glyphs fail. Instead, each character should be defined one by one without collapses, and should be given a correct final glyph. The problems arising with GB 8045 will be discussed later.

The present-day Uigur script mainly used in Xinjiang, China is not the direct descendant of the classic Uigur script. Rather, it derived from Arabic script, and under the recent script reformation, it has developed the eight distinct vowel characters. They use hamza as a vowel specifier to mark the syllabic boundary: So, the present-day Uigur forms another branch of the Perso-Arabic family.

Summarizing so far, Mongolian script is a superset of the Perso-Arabic scripts, except for the explicit information of syllabicity of the present-day Uigur script.[5] It means that we can handle all other Perso-Arabic scripts with the text manipulation mechanism to handle Mongolian script.

No present text processing mechanism can edit Mongolian or any other Perso-Arabic type of script texts.[6] In fact, current systems only support ‘one-character-is-one-glyph’ type such as the Latin alphabet. The default writing direction must be horizontal from left to right, with the up-to-down line feed: these factors are fixed, not changeable. Nor is processing of mixed-script text supported.

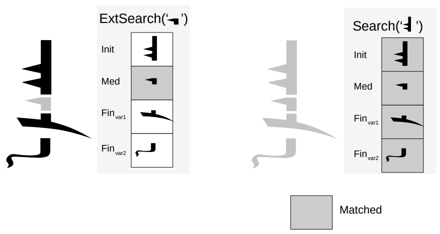

Fig. 8. Required Search Possibilities

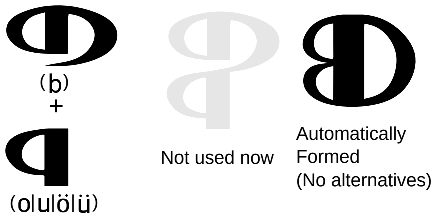

Fig. 9. Curling Type Ligature

4 Todo script is rather exceptional in that it can distinguish almost every sound. Todo was the reformation of Mongolian script in 1648 by Zaya Pandita to make it intelligible to the Oirats. The new alphabet gives distinct glyph shapes to the formerly ambiguous characters. It has been used in Alashan and in Xinjiang with minor modifications. As is clear from the script tables of Mongolia/Inner Mongolia and Todo, there are great differences in the number and glyph shapes of the characters. So it is necessary to make the conversion system among the three.

5 Mongolian script is still phonemic, while present Uigur script is completely syllabic.

6 Any other attempt, [9], [10], and even the Bogda system which has been developed in Xinjiang Autonomous Region, which is quite well-designed by itself, included, is a typesetting system, not the one with text manipulation ability as System 1. Once compiled, a text string cannot be edited, much less the automatic form change nor the extended search by form information in our sense supported.

A text processing system desired should provide two types of search for Arabic and Mongolian scripts: one to find one character, and the other to find a certain form/variant of the character (Figure 8).

2 Definition of One Character: A Processing Unit

Recent development of world-wide computer networks necessitates text processings of script(s) of any kind not only separately but also mixed without inconsistencies. For the latter, internationalized language/codeset independent text processing, a processing unit for each script must be clearly defined to ensure that the same shared text processing functions can operate on any script. Thus, such a processing unit which stores necessary information, the definition of one character, a properly designed codeset, and an output mechanism with clever cursors given, the basic text manipulation like insertion, deletion, and line separation can be realized without language/codeset information at all.

Generally, scripts in the world are classified into 1) phonemic, 2) Syllabic: (2-1) Conjunct Syllabic and (2-2) Pure Syllabic, and 3) Ideogrammic. Our extensive research of writing scripts of the world revealed, however, that it is most appropriate to classify them into the following two categories in terms of the manner of construction, i.e., the way a character is composed: A) Non-Conjunctive scripts and B) Conjunctive scripts ([6], Figure 10). In Thai and Devanagari scripts (Class 2-1)), a character is composed of smaller parts with explicit (expressible using BNF) construction rules, quite different from the Latin script (Figure 11). On the other hand, a character of Non-Conjunctive

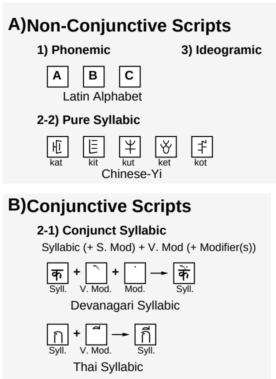

Fig. 10. Categories of Scripts

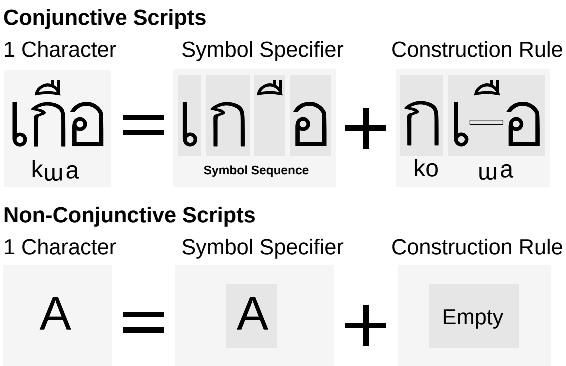

Fig. 11. Construction of Scripts

scripts can be defined as one with empty (zero) construction rules. Thus, the set of Non-Conjunctive scripts is defined as a subset of that of Conjunctive scripts (Figure 11).

It is clear that text manipulation functions as Insertion/Deletion apply to the character unit as a whole, not to a part of the character. Usual text processing presupposes that a fixed-length multi-byte (mb) codepoint corresponds to one character and also to one final glyph. However, an mb codepoint does not always stand for a final glyph, nor for one character (Figure 12). A character is assigned its shape, i.e., glyph, only when it is displayed; so displaying can be defined as one of the basic text manipulation functions ([5]).

Notice that this processing unit, character, does not equal a glyph unit, nor a syllable unit as shown in Figure 13. The character defined here must be the Wide Character (WC) of POSIX. Here it has become clear that WC is not only a unit of text manipulation but also the one for display. WC, normalized one character in this way, makes essential text manipulation language independent.

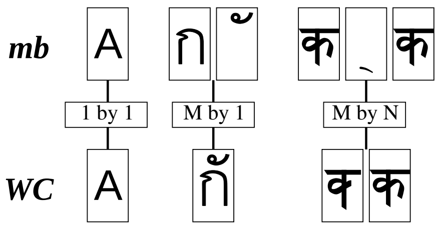

Fig. 12. No 1-to-1 Correspondences

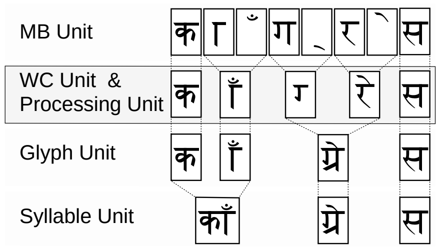

Fig. 13. Process Units in Devanagari

There are so-called logograms as Perso-Arabic and Mongolian scripts, the focus of this paper, whose glyphs are determined according to their positions occurring in a word.

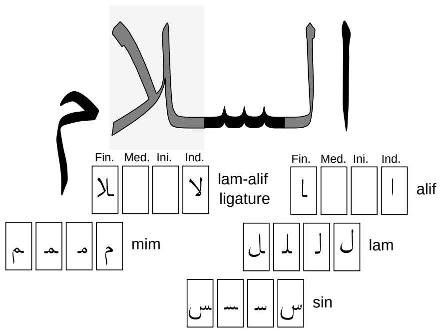

Fig. 14. Arabic Script

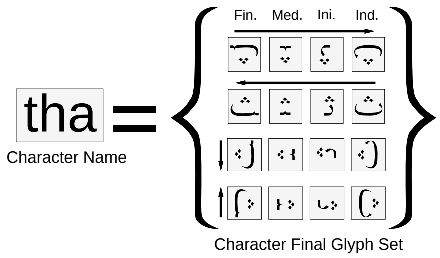

Fig. 15. Character Represents A Set of Glyph(s)

There are also such characters which change their glyphs according to the direction they are written, e.g., the open parenthesis written left-to-right and that written right-to-left. Thus, a character is defined as a name given to a set of multiple glyphs when it is written. In the next section, it will be clarified that all the information required for both text manipulation and display is stored in WC's bitfield to be accessed.

3 WC Information And Essential Functions

Multibyte (mb) codepoints are extended by ISO 2022 [1] and by combination of codepoints in Graphic Character Sets (GCS) and Control Character Sets (CCS) (cf. ISO 6429 [2] and IS 13194 [4]). Potentially repeated processes of character handling and communicating makes it a prerequisite that an mb (multibyte) codepoint string must be uniquely mapped into a WC (wide character) codepoint, and vice versa. A WC codepoint must provide information for presentation and basic text manipulation. It must also maintain language/codeset independence, and ensure reverse-conversion to mb codepoint(s). Thus, a WC is normalized to a Character. It has as a matter of course the character ID field but also has a bitfield to store the necessary information for text manipulation and display.

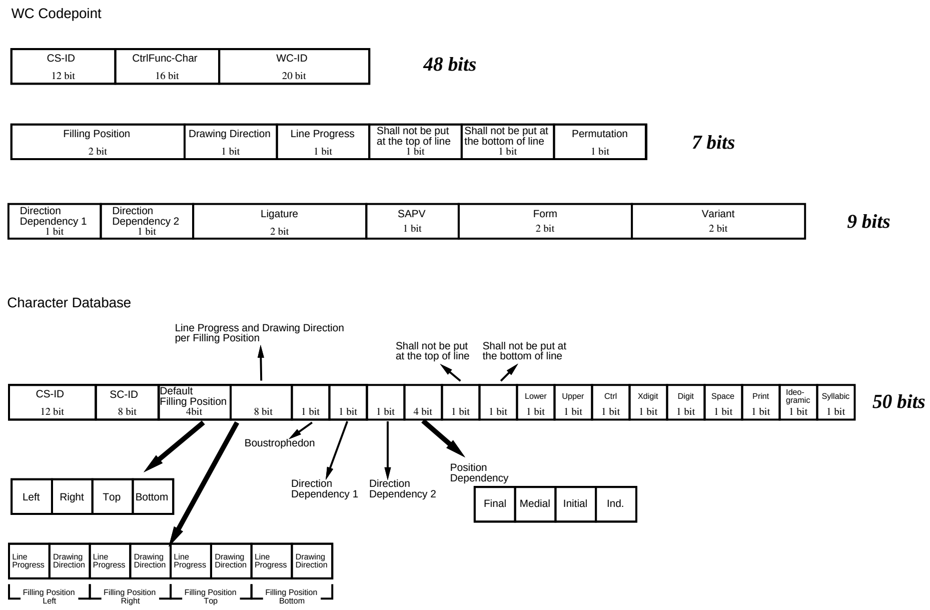

Fig. 16. WC Information

Most importantly for text manipulation of Mongolian script, preparing variant and ligature bits makes possible both the independent text manipulation and displaying the correct glyph. Pieces of character information are divided into changeable and unchangeable ones. The former is placed in the bitfield of a WC codepoint and the latter in the system database.

A TMC (Text Manipulation Code) codepoint, which is prepared for a language specific processing, is made by uniquely converted from WC with language name and language specific information, and there can be more than one TMCs according to users' requirements. Final Glyph Sets (FGS) are mapped from WC. The overall mapping relations are shown in Figure 17.

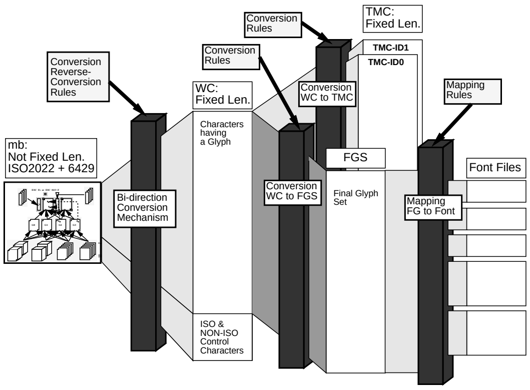

Fig. 17. Mapping Relations

These basic text manipulation functions must be provided: Insertion, Deletion, Search, Input, Output, Cursor Display and Line Separation. Display of cursors must be one of the basic text manipulations provided. Mixing characters with opposite writing directions causes a mismatch of a memory image and its display image. Since there are two possible character directions for a line (Figure 18), a cursor must show 1) direction of a character at the location for which the cursor pointing insertion position, and 2) that there are two possible positions for insertion according to the two directions of character to come at the boundary of character direction change (Figure 19). Also the cursor

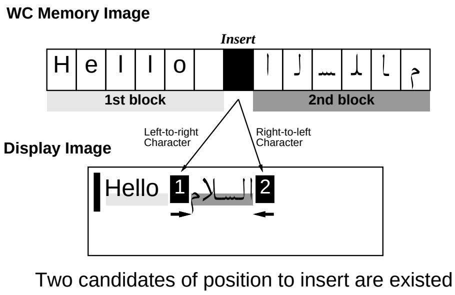

Fig. 18. Two possibilities for Insertion

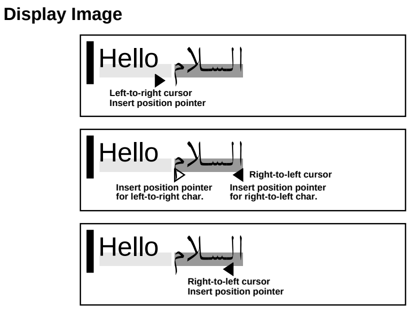

Fig. 19. showing Two Cursors

should be displayed for each character constituting ligatures.

4 Requirements for Internationalized Output Module

Every character either has unchanged glyph or has a glyph which changes according to a) the displaying direction, b) the direction of line feed, and c) its position in a word. At Drawing, the origin of displaying is defined first; next come the definitions of a character, a word, and a direction of line feed; finally, a glyph must be assigned according to a position of the character in a word.

Since a position within a word determines a correct glyph for logograms, minimal length of a character string unit is required for correct drawing. To draw a character for specific glyph of logograms, a character, i.e., a WC must have attributes specifying one glyph from the corresponding glyph set. Notice that a character of Mongolian Script has more that four glyphs. Ligature attributes are also required to specify one glyph from multiple character sequence. Thus, WC must have these attributes to realize the correct character drawings.

Remember a glyph is selected according to the current direction. Proper direction is determined by the relation of the default character drawing direction and the direction of line feed. The default information must be stored in a system, but WC attributes are also needed to ensure the interactive direction change.

Displaying information as a writing direction or a direction of line feed is not necessarily present in ordinary communication or in text files. Note that supporting ISO 6429 requires direction change in one line, although one cannot always read the displayed result.

When the size of a window is detected by a Window Manager, this role of locating blocks should be given by a widget (or an application) instead of the OM. The extent of each block is determined by the default data which are used to determine a block in one line. A line breaking function is also essential.

An internationalized widget should have attributes for origin of line, word direction, and line feed, since it is possible to take the multiple strategies of location of blocks in the proper position in a window. The attributes form default data which must be changed interactively.

The method in which a block itself determines the attributes as fixed data cannot mix multiple directions of words correctly. In other words, letting a premaintained display area define a display direction and a direction of line feed of characters fails to rearrange such blocks. Is is a limitation of widgets and of formatters.

5 Requirements for the Input Module (IM)

Above all, IM must ensure that input is a processing unit (character or word). The input strings of the non-fixed length codepoints of Conjunct Scripts as Thai or Devanagari should not be separated within one meaningful processing unit. Those of Perso-Arabic/Mongolian scripts must form a word, because each form/variant of one character

Fig. 20. Mixing Multi-Line Blocks

cannot be defined unless the whole word is shown. A multilingual IM must communicate with applications in terms of a meaningful processing unit, and should serve both native and non-native users.

Especially for non-natives, some flexible - programmable - devices should be supported. For example, non-native users need to be informed which key (stroke sequence) corresponds to which character: Keyboard assignment information of our IM.

Another window displaying all candidates for the key stroke sequence would be helpful to get input for ideograms as in Chinese Hantsu, or in the cases like Mongolian, which has many variants to select. The selection of the variants is context-dependent, and users eventually must select one. Since it has some characters which happen to have the same glyph, the sound/phoneme information for keyboard assignment would be also informative for the non-native or unexperienced users.

6 Codeset Designs

For proper text manipulation, character codesets must be definable both for a character and for a glyph, but they cannot always be converted into set of characters, i.e., WCs. As a typical example of uncomputable codeset, GB 8045 for Mongolian Script in China (Figure 21) was designed only for glyph sequences represented for character without any delimiters, and it is impossible to determine one character in the sequence. A certain character is generated by 3/3 *and* by the combination of 3/2 plus 2/10 as well. Different characters which happen to have the same glyph is given the same codepoint (e.g., 4/2).

Below the categorizing character code set was done according to possibilities to determine one character and one glyph to be assigned to the character.

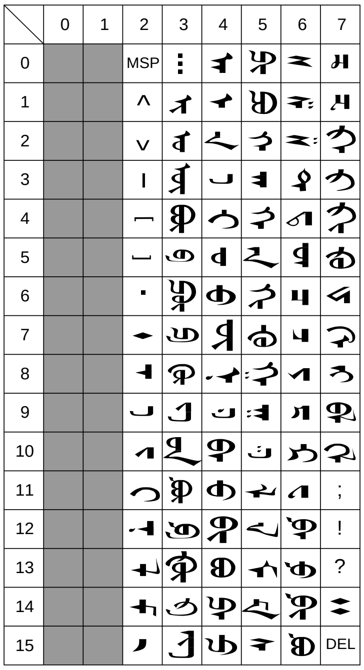

Fig. 21. GB 8045 (Glyph Definition)

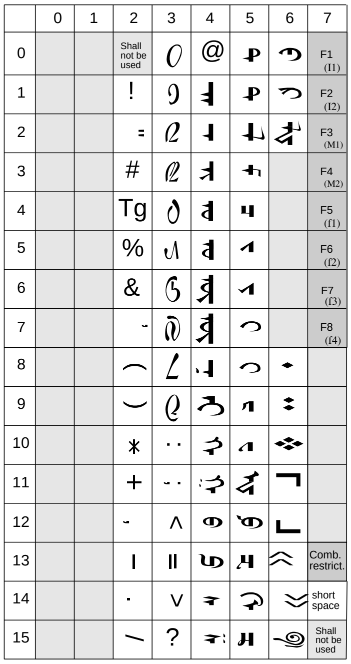

Fig. 22. Waseda Mongolian Code (Character Definition)

As is clear from the discussion above, both a glyph and a character must be definable for a proper text manipulation. Only the cases (1) and (5) in Figure 23 can be converted into WC. In ISO 10646 or UNICODE, cases (2) to (4) and (7) were defined, inabilities to define a character. In the context of Perso-Arabic/Mongolian scripts, different glyphs, forms and variants, of the same character, are and will be assigned different codepoints: No guarantee for the correct glyph after such operation as insertion/deletion.

The proposed codeset for Mongolian script (Figure 22) is a Character definable one for text manipulation. The character ‘a’ and ‘e’, for example, are given distinct codepoints, although they happen to have the same glyph in the medial position. There are general rules; however, there are so many context and user dependent exceptions that it would be inefficient to write such rules for the automaton. Rather, style variation selectors can identify the forms for each character: F1 is used to select the initial form. Likewise, F4 is for the alternative medial form, F5 for the first final form variant, F6 for the second, F7 for the third, and F8 for the fourth final form, respectively.

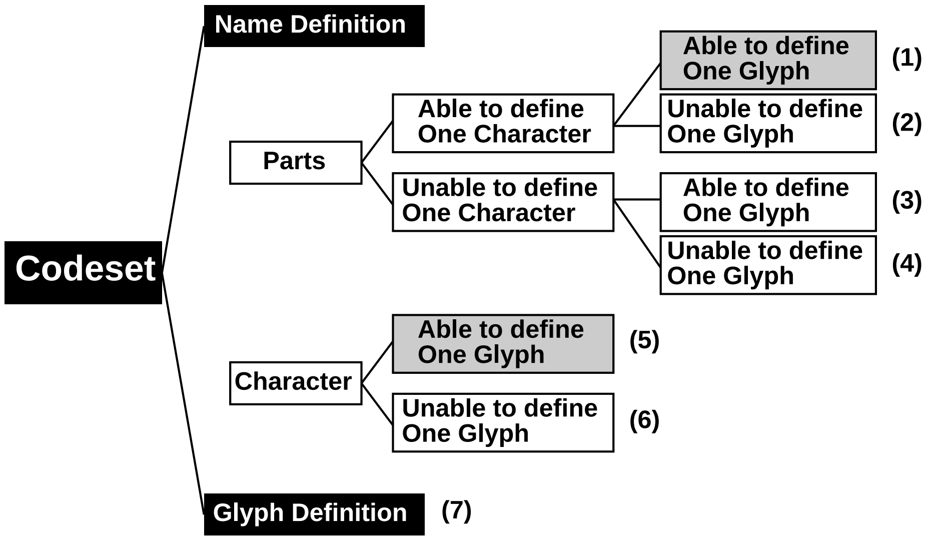

Fig. 23. Codeset Designs

7 System 1

The core of the System 1, Waseda Multilingual Computing Environment, is a combination of converters named Meta-Converter, which performs mb/WC, WC/mb, WC/FGS (Final Glyph Set). Here, FGS is a set of glyphs, and a member of FGS represents number of (Bitmap) Font, Font File Name. In other words, during conversion from WC to FGS, WC can be drawn by X Window System: this part has a very small dependency on X Window System.

To realize the conversions, WC has attributes as discussed. All conversion rules are described in text files, and Meta-Converter Table Compilers compile them and generate Optimized Automaton Bodies. The Automatons are loaded when System 1 library is invoked and shared by all processes. Thus memory usage is also optimally minimized. Over the System 1 Library, portable function layers support compatibility with POSIX and X11R5/6 as their supersets. The compatibility library replaces localized functions. Also new essential basic text manipulation functions and communication functions layers are added. The overall structure of System 1 is shown in Figure 25.

8 Concluding Remarks

The solutions presented here for Mongolian-related scripts text manipulation are not isolated. Rather, they only form a subset of the true Internationalization and multilingual text manipulation. The System 1 of Waseda University gives the total environment of that kind, and both script/character and language processings of any type and any combinations have become possible.

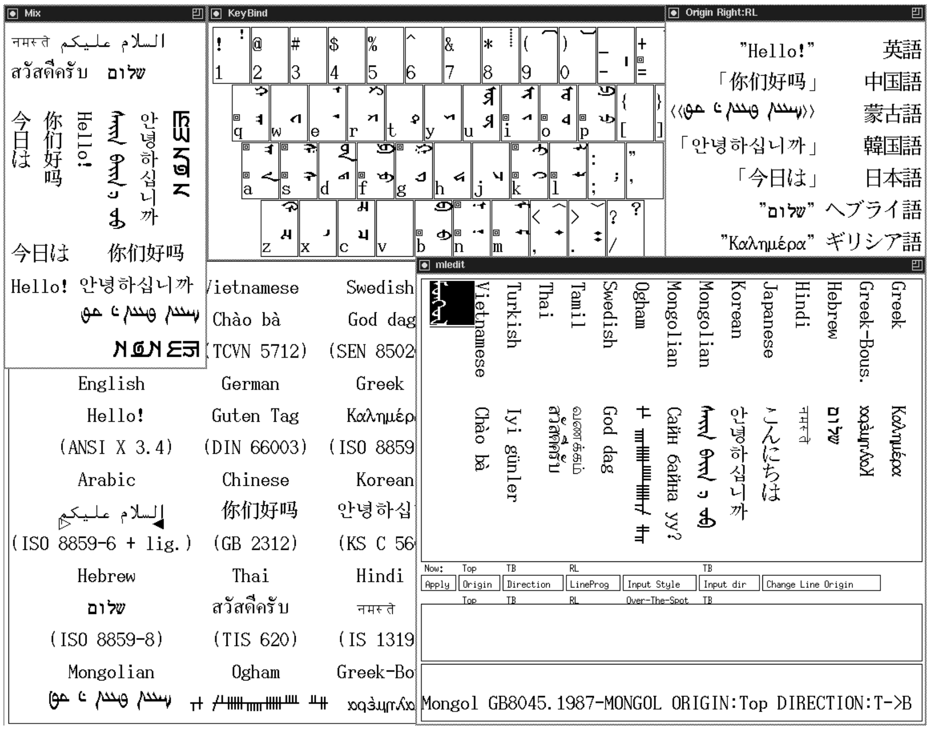

Fig. 24. Running View of System 1

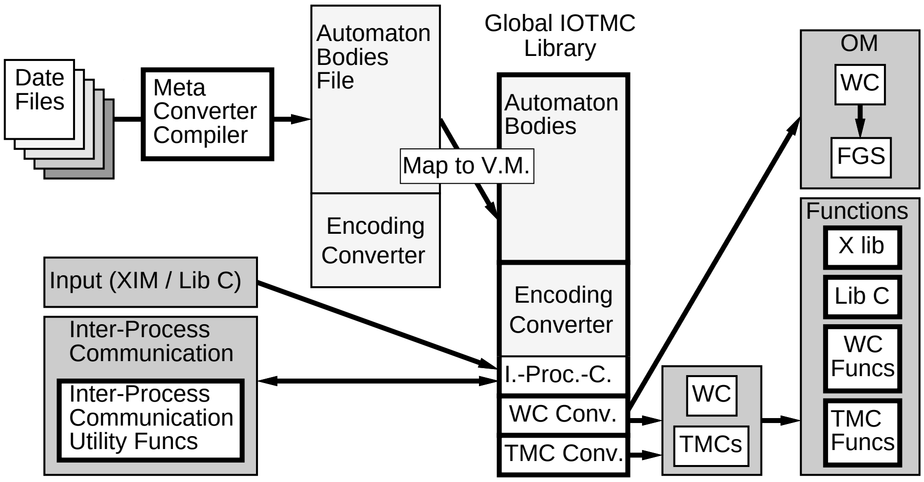

Fig. 25. Modules of System 1

The research above also gave us insight into natural language processing in general and into influences among languages and among scripts. Studying this diversity of writing systems also has given us insight into sorting orders and database accessing codes.

An Internationalized (I18N) Browser cannot be available unless the problems be solved of all writing directions and of the opposite directions of line feed. As a presupposion, the region of display which shares the same direction of line feed must be specified and rearranged as a prior condition. Simple extensions on the current browser systems cannot realize it, and as the project continues, we have already started such researches ([7]).

Acknowledgments

Many scholars and native speakers have helped us. To list a few alphabetically: Arkin Ablimit, Osamu Inoue, Dawa Yidemucau and Jun-ichi Yoshida. Some have taken the heavy task of developing and implementing of System 1: Tadao Tanaka, Hidejiro Daikokuya, Toshio Oya, Kenji Maruyama and Kazue Hoshino. To the Program Committee and the Conference Secretariat, especially to Professor Bob Morris and Dr. Jaques André, goes our deep gratitude for providing invaluable suggestions and comments and helping us enhance the readability of an earlier draft.

References

1. ISO/IEC 2022: 1986, Information processing 7-bit and 8-bit coded character sets Code extension techniques
2. ISO/IEC 6249 (1992), Information processing Control functions for coded character sets
3. TIS 620-2533 (1990), Thai Character Codes for Computers, Thai Industrial Standards Institute, Ministry of Industry, Thailand
4. IS 13194 (1991), Indian Script Code for Information Interchange ISCII, Bureau of Indian Standards, India
5. Kataoka, Tomoko, I., et al. Problems and Solutions for Mongolian Related Scripts Text Processing. *Proceedings of the Symposium of Humanities and Computers '96*, 1996, 81–96
6. Kataoka, Yutaka et al. The Essentials for Developing Multilingual Computer Environment. *Proceedings of the 2nd Workshop on Multilinguality in software Industry: The AI Contribution (MULSAIC'97)*, August, 1997
7. Uezono, Kazutomo et al. I18N Drawing Functions for a Text Containing All Characters/Languages in the World. *IPSJ SIG Notes, Human Interface, 97-hi-70*, 1997, 55–62
8. Haramlambous, Yannis. "Typesetting Khmer", *EP-odd Journal*, Vol. 7, No. 4, Dec. 1994, 197–215
9. Becker, Zeev and Daniel Berry, "Triroff, an Adapatation of the Device-Independent Troff for formatting tri-directional text," *EP-odd Journal*, Vol. 2, No. 3, Oct. 1989, 119–142
10. Srouji, Johny and Daniel Berry, "Arabic Formatting with Ditroff/Ffortid," *EP-odd Journal*, Vol. 5, No. 4, Dec. 1992, 163–208