4 S − 4

Multilingual I/O and Text Manipulation System (2): The Structure of the Output Method Drawing the World's Writing Scripts beyond ISO 2022

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

1. Introduction

To realize the Multilingual Output Mechanism[4] required by the first paper of this series[5], indepedencies of Locale Model, code set designs, codepoint extension methods and of writing conventions are essential. One codepoint does not always stand for one glyph in Graphic Character Set (GCS)s[1]. For example, in *Thai* in TIS 620-2533:1990, multiple codepoints determine one *final glyph* with Thai writing convention, and in *Arabic* in ISO 8859-6:1987, one codepoint has several glyph candidates, from which the final glyph is determined according to its position of the word. Thus, final glyphs cannot be determined by only ISO 2022[2] and ISO 6429[3].

In X11R5, GCS may be added to a locale table for multilingual drawing, but, X11R5 can draw only the GCSs from left to right only when each codepoint in a GCS is mapped to one glyph in a font file. X11R6 was implemented as multiple *Output Method* (OM)s binding corresponding OM dependent locale tables and as one of the OMs loaded when a locale was specified. Thus, it spends more resources and is fully locale dependent, and one OM in it does not ensure the same drawing manner on different systems because there are no drawing rules for extensions/writing conventions in the locale table.

Based on Multi-locale Model and Global IOTMC Model[5], our OM newly developed satisfies all requirements to be multilingual OM.

2. Definition of the Output and Multilingual OM

Output is defined as to draw final glyphs correctly from given encoding schemes and given codepoints. On a windowing system, *drawing* is an independent event one another, and its extent is one line. *Primitive Drawing Functions* (PDF) draw a string in single direction from a glyph indexes in a font file with coordinate. Thus, multilingual OM processes a string and passes the results to PDF to draw.

To mix all writing scripts, four drawing fucntions in directions left to right, right to left, top to bottom and bottom to up are essential. But usually only left to right drawing function is supported as PDS, thus, the bottom layer of the multilingual OM is an emulator of four direction drawing functions by PDF. The emulator is called *Quad Directional Drawing Function*. Therefore, OM should analyze encoding schemes to specify one GCS, codepoint extension methods to specify candidates of final glyphs, directions of words to be written, origin of line, direction dependency and position dependency on the emulator.

3. Classification of Writing Conventions and Code Set Designs

In order to define mapping between mb and Final Glyph Set, all the writing conventions of scripts and code set designs must be analyzed and classified.

3.1. Analysis of Writing Scripts for Drawing

Writing scripts are classified into 1) scripts having their own writing direction – *Direction Independent* or 2) script drawn to current direction – *Direction Dependent.* Direction dependent is classified into *Variable form* type or *non-Variable form* type[6]. A glyph shape in *Arabic* or *Mongolian* is determined in a position in a word – *Position Dependent.* The forms in position dependent are classified into the following, 1) *Independent form,* 2) *Initial form,* 3) *Medial form,* and 4) *Final form.* But not all of position dependent have the four forms, thus a glyph form is determined by connectivity with the preceding glyph and the following glyph. Therefore, informations, writing direction of a script itself or not, forms for directions, default direction to write, current drawing direction, connectivity rules for position and logical font file names are essential to define one final glyph.

The information of default direction to write is defined in *Relation Tables,* and current direction is determined during drawing, and other informations above are defined in each GCS table.

3.2. Classification of the Code Set Designs

Graphic Character Sets are classified into the following by their designs, 1) *1 codepoint for 1 glyph* (1 by 1), 2) *1 codepoint for Multiple glyphs* (1 by M, eg. ISO 8859-6:1987), 3) *Multiple codepoints for 1 glyph* (M by 1, eg. TIS 620-2533:1990), and 4) *Multiple codepoints for Multiple glyphs* (M by M, eg. IS 13194:1991). Control Character Set (CCS)s contain codepoints that specify final glyph shapes. Also in 3 and in 4, some codepoints having glyphs affect final glyph shapes by specific sequence of codepoints. Those codepoints in GCSs and in CCSs are defined as *Determinative.* Thus, a Determinative is a codepoint having a function to specify new glyph(s) in extra glyph set with specific ranged sequence as its parameter.

By analysis of Determinatives, Determinatives have summarized to the following essential functions necessary at all; 1) taking adjacent two as its parameters and selecting one glyph-token – named Function Concatenation, 2) taking a ranged sequence and selecting one – Graphic Character Composition (GCC), 3) taking a preceding codepoint and the Determinative itself and selecting one – Conjunct_Form, 4) taking one and selecting three – Form_Selector, 5) taking one and selecting one – Glyph_Selecter, 6) taking itself and selecting one – Discontinuity, 7) taking preceding one and itself and selecting two – Permutation and 8) taking adjacent two and selecting one or two according to value of itself – Halant. And as a special Determinative, Disjoint_Indicator informs the end of a sequence to its paired function.

Generalizing this idea to ordinary codepoints, those codepoints are also functions to return themselves as glyph-token. Thus, all codepoints can be replaced by functions. Consequently, an array of pointers to the functions – the content of an array corresponding to the contents of a GCS Table – can be used to invoke each function in the array with a codepoint as an index. As a result, designation sequences, invocation sequences and control sequences are processed by this method replacing the content of an array without locale/extension rule dependencies by providing all the arrays simultaneously.

The arrays are generated by *Meta Converter Table Compiler*. Each function above has its paired reverse-function from glyph-token(s) to the original sequence. By holding data for reverse-conversion in a data area corresponding to an array, it is possible to do reverse-conversion by specifying the reverse-function index only. Thus, an extended codepoint in WC from multiple codepoints can be stored in a fixed length data type.

4. Structure of the New Multilingual OM

The new multilingual OM loads the *Objects* of the Meta Converter Table Compiler when the multilingual system including the OM is invoked.

For drawing, the *Internal Wide Character* (IWC) is a fixed length code with informations of Origin of line, font-ID and glyph index, and corresponds to final glyph set. The Converter in OM converts from mb/WC codepoints to IWC codepoints and, Quad Directional Drawing Function extracts informations required by PDFs. The conversions to IWC in the OM are done by *Trans-Unit Converter* in the *Meta Converter System*[6]. The arrays of pointers are placed in the Trans-Unit Converter. The Drawing Functions in the OM call mb-IWC or WC-IWC converter to draw a given string.

Mb-IWC String Conversion in the Converter consists of three steps.

1) Control Functions and Determinative processing:
   Automaton for GCS/CCS calls a function described above by the first byte of a given codepoint, and the function generates *Pass 1 Token*(s) from codepoints with given Determinative informations. The Pass 1 Token is a candidate to select one codepoint in the final glyph set. When designation or invocation sequences are processed, the functions switch states of In-use table or Intermidiate table and replace the function set without generating Pass 1 Token. When control sequences affecting Current direction or Origin of line are processed, status areas are changed without generating Pass 1 Token.
2) Direction determination:
   This step generates *Pass 2 Token* from Pass 1 Token. Direction dependency is removed by this step using the information about scripts having direction or not and/or about transformation corresponding to the direction selected here. If a Pass 1 Token is direction dependent, current direction is assigned and corresponding Pass 2 Token is selected.
3) Position determination:
   This step generates IWC from Pass 2 Token. Position dependency is removed by this step using connectivities of two Tokens adjacent to the Token. The rules of connectivities are associated to one GCS with a certain font style. Then the IWC codepoints are passed to the Quad Directional Drawing Function and the Function extracted informations for PDF – finally here, a codepoint is mapped to a final glyph to be displayed – and PDF draws simple string(s).

WC-IWC Converter in Trans-Unit Converter extracts font-ID, GCS-ID and glyph index from given WC codepoints, then it generates IWC.

5. Summary

World's writing scripts, GCSs and CCSs were analized. Then mapping between codepoints and final glyph set, and essential informations for drawing were defined. The new Multilingual OM can draw any GCSs correctly and simultaneously. This system runs 1-1.5 times as fast as X11R5, because the best algorithm could be selected by the analysis for the Meta Converter System.

References

[1] Tanaka, T., et al., Generalized Output System that draws multiple code sets on a windowing environment (in Japanese), Proceedings of the 46th General Meeting of IPSJ, vol. 5, March 1993, pp87-88.
[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., A model for Input and Output of Multilingual text in a windowing environment, ACM UIST'91, November 11-13, pp 175-183.
[5] 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).
[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).