4 Q − 7

The Worldwide Multilingual Computing (7): Multilingual Programming Language for Advanced Researches

Kenji Maruyama‡, Yutaka Kataoka*, Tomoko Kataoka*, Kazutomo Uezono†, Tadao Tanaka‡, Hidejiro Daikokuya†, Toshio Oya†, Shoichiro Yamanishi† and Hiroyoshi Ohara†

* Centre for Informatics, Waseda University † School of Science and Engineering, Waseda University
‡ Research and Development, Japan Computer Corporation

1. Introduction

The researches on the scripts of the world have made clear the definition of one *character* and the layered structures of the scripts [Talks 1, 4], and more than one process unit must be supported for text processings [Talk 4]. The language/codeset independent basic text manipulation has been segregated from the advanced text processing, which requires the language/codeset/user dependent operations. WC has been normalized to one character for the independent manipulations including the displaying functions [Talks 4, 5]. More advanced processings have been discovered to necessitate the flexible, user-definable unit of *TMCs* (Text Manipulation Code) [1, 2].

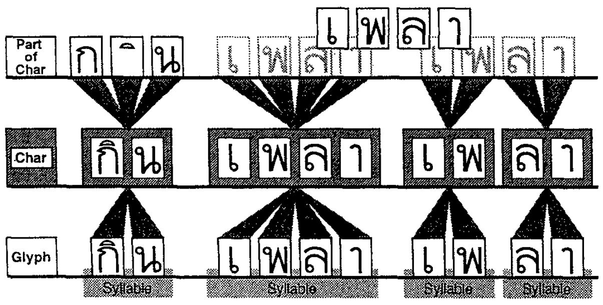

Figure 1. Layers of Syllabic Scripts

Multilingual Applications are supposed to be able to handle both 1) language/codeset independent basic manipulations based on the definite process unit as our WC, and 2) advanced *interactive* processings with dependencies on languages/codesets, or especially on their users' personal purposes. Thus, TMCs and *Interpreter* type of programming languages are essential in this respect.

As a core of such programming languages, the Multilingual FORTH (ML FORTH) System has been developed to realize multilingual text string data handlings on any layer for any use. The ML FORTH compiles *WORDs* (pieces of a program as functions) into a machine language of a virtual CPU by calling the compiler routine – the ML FORTH is faster than standard FORTH. Multilingual Lisp can be provided by the ML FORTH and its program(s). It is easy for the FORTH itself to customize applications for advanced purposes, e.g. Text formatting, parsing, and other natural language processing like WYSWYG.

2. Researches for ML Common Lisp

Common Lisp may come out as one of the first candidates for such a programming language for text string handlings. However, it contains some serious problems. First of all, it is based on *mb* and only one GCS is basically available at one time by its specification. As clarified in Talks 1 and 2, mb codepoints are not for string handlings but for communications; WCs should play the role instead. A few trials for multilingualization, if any, might end up with some variants of Locale model, without revising the specification itself.

The hierarchy of 'characters' of original Common LISP is *nil* → *standard-char* → *string char* → *character* → *t*. To be Multilingual Common LISP, the hierarchy should be extended. The necessary hierarchy of characters is shown in Figure 2. By the extension, mb/WC/TMCs can be processed simultaneously and consistently.

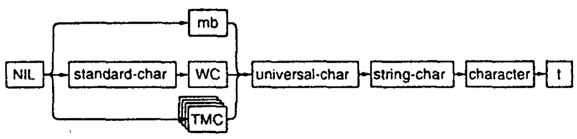

Figure 2. Necessary hierarchy of characters

Here standard-char means current WC. Thus, all 'characters' are WC and other types can be managed consistently by classifying them as *universal-char*.

Yet the specification has another flaw/inadequacy: Common Lisp has no definitions on displaying, although WC has been defined to hold the presentation information in its codepoint in addition to the character ID. There is no way to get informed whether the WC would be displayed/written horizontally or vertically.

Finally, the Common Lisp system can be so huge, and it would be far from the optimal module to be linked to application softwares. To solve those problems we provided the ML FORTH with enough WORDs and functions, from which the efficient ML Common Lisp system can be derived.

3. The ML FORTH Overview

The ML FORTH is a compact, interactively programmable language with a set of WORDs, which are functions, to handle multilingual texts. All the operations are simply stack-based with *Reverse Polish Notations*, so leading to a high programming flexibility and modularity. It can handle mb/WC/TMCs strings simultaneously with causing no inconsistencies. Conversions among the three units are supported by the *Meta Converter*. The strings in mb/WC/TMCs are stored and manipulated by optimal WORDs; thus, it is quite easy to manipulate text like LISP. Allocation of memory for string and deallocation is done by the WORDs automatically. Thus, it is quite easy to manipulate strings in any types. All data types to satisfy the multilingual Common LISP were provided. It is able to call all OS functions, i.e., functions in the standard libraries are available.

The *standard-char* of the initial WORDs of the ML FORTH is internally mapped to WC from ISO 646 and C0 set of ISO 646 i.e., initial WORDs of the ML FORTH were described in ISO 646 in mb. But the default character sets can be specified to save locales. By the selection, one of ISO 646 IRVs is mapped to ISO 646 space in WC.

The ML FORTH gets input a string in mb and the string is converted into WC. Then the string is compiled. And strings in all character sets by ISO 2022 [3] may be used to describe the ML FORTH WORDs. Thus, all characters in WC can be used as the character set of the ML FORTH. To correctly process an encoding scheme which has default designations and invocations, default of standard-char may be changed. The memory image that is generated by the compiler can be saved into a file outside.

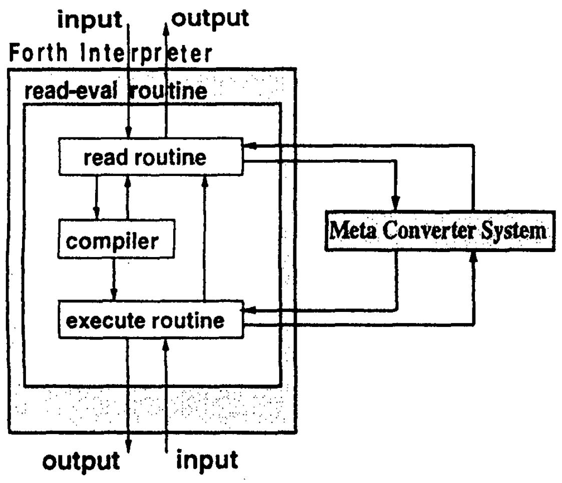

Figure 3. The Structure of the ML FORTH

When a string in WC is gotten output to stdout, the ML FORTH converts the string into mb quoted by '<' and '>'. When a string is in TMC, the string is converted into mb quoted by '[n' and ']', where n means TMC ID number. But direct output in WC or TMCs can be selected.

Adding to ML FORTH interpreter, there is a version ready to link to applications as a *Runtime Library*. Note that all the Meta Converter Functions are also callable as WORDs for the advanced text processing. The structure of the ML FORTH is shown in Figure 3.

4. Generalization of Processing Texts in TMCs

Even for the advanced text processing with language/codeset dependencies and personal requests – category 2), it is possible to perform the processing by a set of finite functions. Each TMC has customizable and name-specifiable bitfields to hold information which may be highly specific. Specifying a combination of names of a bitfield in a TMC codepoint, operations and TMC number for one of the basic processing functions, it is possible to perform infinite processings [2, Talk 4].

5. Summary and Further Remarks

Category 1) of multilingual applications noted above can be defined as the Multi-Codeset Application and 2) as the true Multilingual Application, in that the latter involves several kinds of specificities lurking behind the languages/scripts, which TMCs absorb. Given the ML FORTH, handy text processings have become possible in the areas as multilingual language education or library databases with suitable syntax sugar. The existing applications became easy to multilingualize by linking the Runtime Library of the ML FORTH.

Essential information embedded in orthographies has been made clearer by the analysis of written languages and spoken languages embedded in orthographies have been made clearer [2, Talk 4]. Thus, the information can be sufficiently utilized to define the structural rules for *Conjunct Syllabic* character combinations, where the semantic unit can be found out which is the substructure of a word. By TMCs with the ML FORTH associated, new approaches to natural language processing can be taken.

References

[1] Kataoka, Y. et al., 1995. Codeset Independent Full Multilingual Operating System: Principles, Model and Optimal Architecture, IPSJ SIG System Software & Operating System, 68-4, pp. 25-32.

[2] Kataoka, T. et al., 1994. 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, Proceedings of the 49th General Meeting of IPSJ, Vol. 3, September 1994, pp. 303-304.

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