The Multilingual Input Method – Egg V4

Yoshio Katayama
PFU Limited
658-1, Tsuruma, Machida,
Tokyo 194-8510, JAPAN
kate@pfu.co.jp

ABSTRACT

Mule (MULtilingul enhancement for Emacs) is, as its name, the multilingual editor developed by the Electornics Technology Laboratry (ETL) based on GNU Emacs that is one of the most commonly used editor developed by Free Software Foundation (FSF). GNU Emacs is revised as version 20 and took in the Mule feature in the year before last, and GNU Emacs itself become a multilingual editor. The Mule project of ETL is still continuing and developing the multilingual feature of GNU Emacs. Following this renewal, Chinese/Japanese/Korean input method "Egg" is also renewaled as version 4 as a part of the Mule project of ETL. I joined to the Egg V4 project and took charge of the multilingualization. I designed it focusing multilingual text input because there was no input method suitable for multilingual text input. The multilingual text input requires frequently input language switching, and the cost of this operation affects smoothness of the multilingual text input. Egg V3 has restrictions on input language switching and other input methods, to begin with, does not consider about the multilingual text input. So, the multilingual text input was very irritated things. To solved this problem, I developed the mixed language Chinese character conversion technique and implemented it into Egg V4. This technique removes the restriction on input language switching and allows the smooth multilingual text input. This paper reports on the mixed language Chinese character conversion technique.

Keywords

Multilingualization (M17N), input method, multilingual text input, Chinese character conversion, GNU Emacs

1. Introduction

To input the multilingual text (i.e. the text consisting of two or more non-English languages) needs the input method change by mouse operation because ordinary input methods are designed to input text using only the script of specific language (and English words using ASCII character). The most common case of the multilingual text input is writing commentary of foreign language documents, foreign language textbook, and so on. In this case, foreign language words are studded in the base language text. Therefore, frequent mouse operation is needed while typing text. For the smooth multilingual text input, the language change should be done by the same cost as switching to ASCII input (i.e. one or two key strokes).

Although Egg V3 can input Chinese, Japanese and Korean, it still has a restriction about language switching. As you know, shapes of Chinese characters of these languages are different, and the character unifications are done separately in each country. So, Chinese characters of each language should be sometimes considered as different script. This means that Chinese characters of appropriate language should be chosen and Chinese characters of different languages are mixedly used at a time.

To input the languages using Chinese character is done as followings:

- input the reading
- convert it to Chinese characters
- select the aimed one among candidates
  (when converted characters are not appropriate)

The second and third steps are called as the Chinese character conversion together. Usually, this conversion is done by the Chinese character conversion engine because it requires the dictionaries and the sophisticated algorithm to select the most appropriate candidate. Egg is also designed as a frontend of Chinese character conversion engines of Wnn Japanese/Chinese/Korean input system and of SJ3 Japanese input system (user choice). Although, Wnn input system can input Japanese, Chinese and Korean, it has separate conversion engines for each language and Egg V3 was designed to use one conversion engine at a time. To avoid confusion at the Chinese character conversion step, it allows language switching only when input text is empty. So, there is no big difference Egg V3 and other input method at the point of the multilingual text input.

To resolve the multilingual text input problem, I developed the multilingual Chinese character conversion technique using the ordinary (i.e. uni-lingual) conversion engines and implemented it into Egg V4. The key of this technique is using multiple conversion engines simultaneously and selecting proper one. This technique allows the input language switching at any time and makes the multilingual text input very smoothly. I also designed Egg V4 that it can handle the languages without Chinese character conversion (e.g. Thai, Vietnamese, etc.) and allows smoothly input multilingual text consist of languages with and without Chinese character conversion.

I describe about the Mule features briefly in Section 2. In Section 3, the structure of Egg V4 and the multilingual Chinese character conversion technique are described. In Section 4, user operations are explained.

2. Mule Features related to Input Methods

Because Mule features consist of quite many parts, it is impossible to explain all of them in this paper. I explain only the parts most related to input methods.

2.1 Internal Character Code

Mule internal character code is based on ISO/IEC 2022 [5]. This means that Mule handles 94 or 96 character sets (single byte characters) and 94x94 or 96x96 character sets (double byte characters). The biggest difference between ISO 2022 and Mule code is that ISO 2022 adopts stateful encoding while Mule adopts stateless encodeing for easy handling by computer programs. To put it concretely, each non-ASCII character has prefix byte(s) to distinguish character set and encode to 8 bit scheme (i.e. MSB is set to 1) to distinguish from ASCII character. Table 1 shows the types of prefix bytes.

prefix byte(s)		dim	width	description
	0x81 - 0x8F	1	N/A	basic single byte charset
	0x90 - 0x99	2	N/A	basic double byte charset
0x9A	0xA0 - 0xDF	1	1	extended single byte charset
0x9B	0xE0 - 0xEF	1	2	extended single byte charset
0x9C	0xF0 - 0xF4	2	1	extended double byte charset
0x9D	0xF5 - 0xFE	2	2	extended double byte charset

"Dim" means the number of character bytes (deminsion) and "width" means display width counted by columns. The width of basic character set is not available from the value of the prefix byte because it is provided from another table.

Table 1: prefix bytes of Mule code

Mule feature supports not only standard character sets registered by ECMA (European Computer Manufacture's Association) according to ISO/IEC 2375 [6] but also private chracter sets. Main purpose of private character set is to support languages that does not have the standard of character set. There is one private character set related to Egg — chinese-sisheng. This character set includes the sisheng (four tones) characters of Pinyin (Romanized Chinese script) and Zhuyin-zimu (traditional phonetic symbol; Bo Po Mo Fo) characters. These characters are used for inputting the reading of Simplified and Traditional Chinese characters.

2.2 How to Construct Input method

The Emacs has the LISP processor named "elisp" that is specialized for editor. Almost all features, including input methods, of the Emacs are written in elisp. Usually, each keyboard and mouse event causes a function calling bounded that event by a key map. When an input method is activated, the key map is changed to the key map of the input method and keyboard and/or mouse events are stolen by the input method. Then, the input method converts key sequence to a character string of the specific language. Displaying characters are done by inserting that string into an edit buffer directly. While an input method is converting the input key sequence, the intermediate string is inserted the edit buffer and re-written according to the state change of the input method. At this time, Egg makes a "fence" that surrounds the intermediate string using the fence characters to denote it. This is called as "fence mode" (Figure 1).

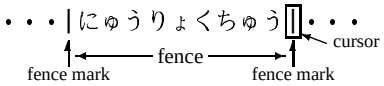

Figure 1: Fence Mode

Input methods for the languages using Chinese characters have two step conversions. The first step is to input reading. For example, to input Japanese Chinese character (Kanji), user types Roma-ji (Romanized Japanese script) and the input method converts it into Hiragana. This conversion is usually called as "translation". The second step is the Chinese character conversion using conversion engines. At this step, input method acts as a mediation between the user and the conversion engine. User commands are translated to operations of the conversion engine and sent to it. Input methods translates user commands to operations of the conversion engine and communicates with it, then shows results to the user. Elisp's IP network feature is used to communicate with conversion engines. This feature is used many elisp programs, mail user agents, netnews readers, ftp, etc.

3. Structure of Egg V4

The structure of Egg V4 can be separated into two parts — Input Translation System and Conversion System. The Input Translation System translates keyboard input to the input string for Chinese character conversion engines. The Conversion System converts that string to Chinese characters communicating with Chinese character conversion engines. Egg V4 has the mechanism to skip Chinese character conversion step for inputting languages using only non-Chinese chracter script.

3.1 Input Translation System

The Input Translation System translates keyboard sequence (i.e. sequence of ASCII characters) into characters of target script using the automaton. Switching the target script is done through switching the state translation table of the automaton. Table 2 shows translation tables currently Egg V4 providing.

Common:	
down-case	ASCII
up-case	ASCII (convert to upper case)
Simplified Chinese (Chinese-GB):	
pinyin-cn	full stroke PinYin input
erpin-cn	two stroke PinYin input

Table 2: Input Translation Tables (continue)

quanjio-down-cn	fullwidth ASCII	(GB 2312)
quanjio-up-cn	fullwidth upcase ASCII	(GB 2312)
qiama	QianMa input method	
wubi	WuBi input method	
Traditional Chinese (Chinese-CNS):		
pinyin-tw	full stroke PinYin input	
erpin-tw	two stroke PinYin input	
zhuyin-tw	ZhuYin ZiMu (Bo Po Mo Fo)	
quanjio-down-tw	fullwidth ASCII	(CNS 11643)
quanjio-up-tw	fullwidth upcase ASCII	(CNS 11643)
Japanese:		
hira	hiragana	
kata	katakana	
zenkaku-down	fullwidth ASCII	(JIS X 0208)
zenkaku-up	fullwidth upcase ASCII	(JIS X 0208)
han-kata	halfwidth katakana	
Korean:		
hangul	hangul	
jeonkak-down	fullwidth ASCII	(KS C 5601)
jeonkak-up	fullwidth upcase ASCII	(KS C 5601)
Thai (experimental implementation):		
thai	Thai Kesmanee input method	

Table 2: Input Translation Tables (continued)

For the multilingual Chinese character conversion, the translation tables have language tags and the Input Translation System adds it to the result string. The Conversion System switches conversion engines according to the language tag. The reason using the language tag instead of the character sets is some character sets, like ASCII, are shared among conversion engines. The ASCII input translation tables, down-case and up-case, themselves don't have the language tag, but take it from the previously used table to share the tables among language. On the contrary, Simplefied Chinese and Traditional Chinese inputs have very similar tables because respective conversion engines are used and different language tags should be set.

The language tag is realized using elisp's text-property. This feature gives property list to each character in strings like as property list of LISP symbols. So, the Conversion System can find out which part of the string output by the Input Translation System belonging which language and the Input Translation System can switch the input language at any time.

Example of Input Translation

Take Hangul translation that requires the most complicated translation in the translation tables of Egg V4. Hangul characters constructed from two or more Jamo (parts of Hangul character). On Hangul input, Jamo is mapped to respective key and its combination constructs a character (Figure 2). The first character of this example is changed as:

"ㅅ" → "서" → "성" → "서"

The character "성" is once constructed then reconstructed to

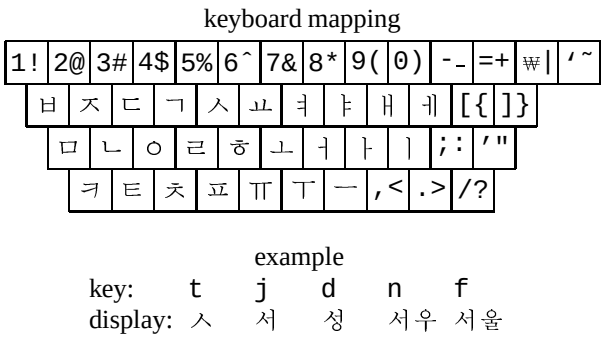

Figure 2: Hangul keyboard and input example

"서" because the automaton uses the fact that all of Hangul characters start with a consonant (Table 3) to detect character boundaries from key sequence.

- consonant vowel [vowel] (고, 과)
- consonant vowel [vowel] consonant (골, 괄)
- consonant vowel [vowel] consonant consonant (곪, 괆)

Table 3: Structure of Hangul character

3.2 Conversion System

The Conversion System converts the string input by the Input Translation System into Chinese characters using Chinese character conversion engines. Conversion engines are external processes separated from the Emacs and they usually use dictionaries for the Chinese charcter conversion. Table 4 shows the conversion engines Egg V4 currently supporting.

Name	System	Language
sj3serv	SJ3	Japanese
cserver	Wnn	Simplefied Chinese
tserver	Wnn	Traditional Chinese
jserver	Wnn	Japanese
kserver	Wnn	Korean

Table 4: Currently supporting conversion engines

To handle different type conversion engine, the abstracted conversion engine is introduced to Egg V4. The gap between real conversion engines and the abstracted conversion engines are filled up by the "backends". Backends translate operations of the abstract conversion engine into operations of the dedicated real conversion engine and communicate with it. Backends can be added dynamically. Backends for other conversion engines are planned.

The backend is not associate with conversion engine itself but associates with the set of the conversion engine and the conversion dictionaries. The conversion engine of Wnn system can do different way conversion by switching the dictionary set. For example, reverse conversion (convert Chinese character into readings) dictionaries, zip code conversion (convert zip code into address) dictionaries, etc., are available for

special conversions. To simplify the abstract conversion engine and to make switching the dictionary set easily, I chosen switching the backends with the dictionary set, not switching the dictionary set itself.
In general, Chinese character conversion operation is done on a clause as a unit. So, clauses are treated as objects and conversion operations are done by sending messages to clauses. The backends provide the method of the messages. Unfortunately, elisp is not an object oriented language, this style is only design philosophy and actual program code is not so clear.
Because features of conversion engines depend on each engine, some backends cannot provide all of conversion methods. Unprovided methods simply return nil. In this case, the Conversion System takes proper actions (e.g. cause an error, no operation, etc.) according to the operation. The most obvious example is the "noconv backend". This is introduced as a mechanism to handle the languages using only non-Chinese characters, like Thai. This provides only one method creating cluase objects.

3.2.1 How to Select Backends

The Conversion System selects backends using language tags set by the Input Transration System as keys for the language-backend associatve list. When input string consists of more than one language, the Conversion System looks up each language parts in the language-backend associatve list and deliver each part to the selected backend separately. Following example, a string consits of two Japanese parts and one Chinese-GB part:

reading いんたーねっとはHù Lián Wǎng といいます
(Japanese ← | Chinese-GB | → Japanese)
Chinese character インターネットは互联网と言います

In this case, Japanese parts "いんたーねっとは" and "といいます" are delivered to the backend associated with Japanese and Chinese-GB part "Hù Lián Wǎng " is delivered to the backend associated with Chinese-GB respectively, then converted to Chinese charcters.
The elements of the language-backend associatve list have following structure:

```
(language
 ;; primary backend set
 ((backend backend-for-reconv...)...)   ;; primary set
 ...)                                   ;; alternative set
```

The primary backend set is used for ordinary Chinese character conversion. The alternative backend sets are used for special conversions mentioned above. They are specified by the prefix command argument of the Emacs. The syntax of a command argument is:

`ESC DIGIT...` or `M-DIGIT [M-]DIGIT...`

Where `ESC` is an escape key, `DIGIT` is a digit key, `M-DIGIT` is a meta qualified digit key (i.e. press digit key while pressing meta key) and `[M-]DIGIT` is an optionally meta qualified digit key. These arguments are passed as integer to the command. The Chineses character conversion command uses this argument for index of the backend set list. The prefix argument "`M-0`" is equivalent to no prefix argument in this case.
The conversion engines of the Wnn system accept any string consisting of any characters of it's language for the input. So, it can convert Chinese character string converted by itself to another string. This feature is the "re-conversion". "backend-for-reconv"s are used for this re-conversion. The selection rule is same as the backend set selection.
The Japanese element of the default language-backend associative list for using the Wnn system is:

```
(Japanese
 ((wnn-backend-Japanese     wnn-backend-Japanese
                            wnn-backend-Japanese-R))
 ((wnn-backend-Japanese-R wnn-backend-Japanese
                            wnn-backend-Japanese-R)))
```

"Wnn-backend-Japanese" is the backend for normal conversion and "wnn-backend-Japanese-R" is the backend for reverse conversion mentioned above. So, the nomal conversion is always taken place for default conversion command and the reverse conversion is always taken place for `M-1` prefixed conversion command. The reverse conversion is not needed 99% for native speakers but very useful for foreigners. As you know, the reading of Chinese character is difficult for foreigners. This reverse conversion feature helps to learn these languages.

3.2.2 Conversion Operation

The conversion operation of the abstract conversion engine is done as Figure 3.

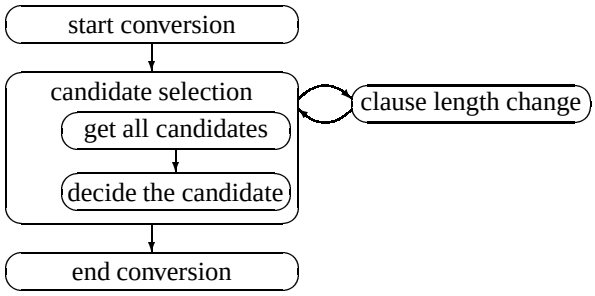

Figure 3: Conversion Flow

The backend receives the string at "start conversion" and separates into clauses that is a unit of conversion, then returns a list of the objects of the converted clause.

3.2.2.1 candidate selection of clauses

The candidate selection is done when the converted string is different from what user expected. There are two type candidate selections, "next (or previous) candidate" and "select

from all candidates”.

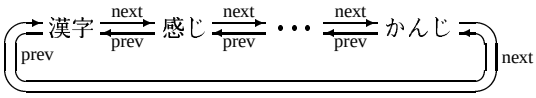

Figure 4: Candidate Selection

In both cases, the messages sent to the clause are “get all candidates” and “decide the candidate”. “Get all candidates” message retrieves the list of the clause’s all candidates represented by the converted strings and the current candidate number. The Conversion System decides the new candidate based on these values and sends “decide the candidate” message with the new candidate number to the cluase. The difference of two type conversions is that “next (or previous) candidate” decides the new one mechanically while “select from all candidates” displays all candidates as menu to the user and decide the new one through user interactions.
Some conversion engines may refer the context (i.e. precede and/or following cluases) at making the candidate list or may affect the context after candidate decision. So, the Conversion System passes the context as arguments of these messages and receive the affected context as a result of the candidate decision.
Usual conversion engines treat only single clause as a subject of candidate selection at a time, but the Wnn conversion engine can treat clause cluster as a subject of candidate selection. This clause cluster is called “major clause” and single clause is called “minor clause”. To support this feature, the predicative message “major-clause-continue-p” is provided. Backends which have no major clause conversion always returns nil for this message to make the Conversion System to treat each clause constructs a major clause of one clause.

3.2.2.2 clause length change

The clause length change is done when separate points are not appropriate. User command of the clause length change is “shrink/enlarge clause”. The Conversion System converts the inrement/decrement value of length into absolute length and sends to the clause. The inrement/decrement value is counted by the number of characters except for Chinese in which counted by the number of syllables. In Chinese conversion, cluases should be separated at the syllable boundaries for meaningful conversion. When the inrement/decrement value is converted to the absolute length, the value is converted to length of string. Clauses following to the target clause are reconverted from that’s readings.

3.2.2.3 end conversion

The conversion operation is finished when the aimed Chinese characters are obtained. At this time, the “end conversion” message is sent to all cluase objects. As a result of this message, finishing conversion operations are directeded to the conversion engines. The conversion engine updates the usage count of dictionary entries and other information.

4 User Operations

In this paper, I describe only the characteristic operations of Egg V4 and omit the same operations with other input methods for the languages using Chinese character.

4.1 Translation Table Switching

As mentioned in section 3.1, the kinds of input characters are switched by switching input translation tables. This operation is bounded to key sequences shoown in Table 5.

Common:	
`C-x C-m q`	down-case
`C-x C-m Q`	up-case
Simplified Chinese (Chinese-GB):	
`C-x C-m C-p`	pinyin-cn
`C-x C-m C-e`	erpin-cn
`C-x C-m C-z`	zhuyin-cn
`C-x C-m C-d`	quanjio-down-cn
`C-x C-m C-u`	quanjio-up-cn
`C-x C-m C-q`	qiama
`C-x C-m C-w`	wubi
Traditional Chinese (Chinese-CNS):	
`C-x C-m P`	pinyin-tw
`C-x C-m E`	erpin-tw
`C-x C-m Z`	zhuyin-tw
`C-x C-m D`	quanjio-down-tw
`C-x C-m U`	quanjio-up-tw
Japanese:	
`C-x C-m h`	hira
`C-x C-m k`	kata
`C-x C-m z`	zenkaku-down
`C-x C-m Z`	zenkaku-up
`C-x C-m x`	han-kata
Korean:	
`C-x C-m H`	hangul
`C-x C-m j`	jeonkak-down
`C-x C-m J`	jeonkak-up
Thai:	
`C-x C-m T`	thai

`C-` means control qualify.

Table 5: Key Binding of Translation Table Switching

This three key sequence is according to the Emacs key bind convention but user can customize to shorter key sequence. If a prefix argument is supplied to this command, the table switching is temporary and back to previous table is done by a single key `C-g` that is usually used for quit actions or after exit from the input mode (this includes entering to the converion mode) automatically. As mentioned above, the situation in which the multilingual text input is the most important is to input the text studded foreign language words in the basic language, like following:

こんにちはは안녕하세요と言います。

To input sentences like this, the temporary table switching is very convenient. Another way for the temporary table switching is provided using the translation table. Character keys not used in the translation table can be assigned for switching translation table and this switching becomes temporally. This mechanism is used for inputting ASCII characters in the translation tables (e.g. ”q” for Hiragana, ”B” for Pinyin an Hangul). Using this mechanism, user can customize to switch to other language and return typing only one key.

4.2 Reverse Conversion

As mentioned in section 3.2.1, the reverse conversion is useful for foreigner, but main question may be how to input the Chinese character at first? If the user knows at least one reading of that character, (s)he can use it to input then reconvert to know other reading. Another way is using the yank feature of Egg V4. The Emacs provides the yank feature and Egg V4 extends this feature into input method. At first, the user drags the Chinese character and yanks it into input string, then start Chinese character conversion with the prefix argument for reverse conversion.

5. Conclusion and Future Plan

Because ordinary input methods don’t consider about multilingual text input at begin with, it is very irritated and stressful to input multilingual text. The multilingual Chinese chracter conversion removes the restriction of switching input language and make the multilingual text input very smoothly and stress free. There are many situations that require the multilingual text input, like translate or writing commentary on translate documents written in foreign language, it is strange that there isn’t input method for the maultilingual text input up to now.

Finally, I describe the future plan of Egg. Because Egg V4 is very convenient for the multilingual text input, I am planning to extend to the languages using non-Chinese character script. Although Egg V4 has the dummy backend for handling these languages, there is another problem.

Many languages used in Asia use syllabic characters. To input syllabic characters does not require conversion engines, requires only the input translation. Syllabic characters are constructed as the combination of some parts of the character. So, to input these characters, the character parts are usually mapped to keyboards (or few key sequence) and whole character is input as combination of the character parts, like inputting Hangul characters. The rules of the combinations are usually very complex and it is the problem that how to describe the rules. These rules are usually described as the mapping table of from the key sequence to the target character directly (Figure 5).

”rk” → ”가”
”rkr” → ”각”
”rks” → ”간”

Figure 5: Hangul Translation Table (One Stage)

This way is very difficult to write and easy to make mistake. Moreover, to change the key map – this is require when developing a input method of a new language – requires to rewrite whole translation table.

To resolve this problem, it is required to separate the keyboard mapping table and the combination rule table that is described as the mapping table of the sequence of character parts to the target character. For example, previous translation table fragment will be described as Figure 6.

keyboard mapping table:
”r” → ”ㄱ”
”s” → ”ㄴ”
”k” → ”ㅏ”

combination rules:
”ㄱ ㅏ” → ”가”
”ㄱ ㅏ ㄱ” → ”각”
”ㄱ ㅏ ㄴ” → ”간”

Figure 6: Hangul Translation Table (Two Stage)

Some scripts have very complex combination rules and require multiple stage translation tables, like Devanagari. To write translation rules easily, the appropriate description language and the multiple stage translation systems. I am planning to develop such description language and multiple stage translation systems and to implement them into Egg.

ACKNOWLEDGMENTS

I am grateful to Yutaka Niibe, Satoru Tomura and Kenichi Handa for making me to join the Egg V4 project and for useful discussions.

REFERENCES

1. Kenichi Handa, et al., Realization of the Multilingual Environment, Prentice Hall Japan (1996)
2. Jin Sugano, Chinese Input Methods, Asahi Publishing Co. (1991)
3. Daniel Yaqob, Transliteration on the Internet: The Case of Ethiopic, in Proceedings of International Symposium of Multilingual Information Processing (1997)
4. Taichi Kawabata, Prototype Implementation of Devanagari Script on Mule, in Proceedings of International Symposium of Multilingual Information Processing (1997)
5. ISO JTC 1 / SC 2, Information technology – Character code structure and extension techiniques, ISO/IEC 2022 (1994)
6. ISO JTC 1 / SC 2, Data processing – Procedure for registration of Escape sequences, ISO/IEC 2375 (1985)