Invited Paper

Character Code for Japanese Text Processing

Akira Miyazawa*

Japanese standard character set JIS X0208 has been widely utilized in Japanese computer systems for over ten years. This character set has 6877 characters including 6353 *kanji* or Chinese characters, and seems to be nearly sufficient for daily use of Japanese text processing. However, the total number of *kanji* characters is more than fifty thousand, and the missing character problem is inevitable for some application fields. In this paper, problems around character codes are discussed from the following four view points: Japanese writing systems, script of Japanese writing system, inter-change character set, and internal codes of operating systems. At the first, Japanese writing system is briefly introduced, and *kanji* script and variant forms are discussed. Then, topics around the JIS X0208 and the new standard character set are introduced, and the internal code design of JEF, shift-JIS, UNIX EUC-JAE, and TRON TAD are described. Problems are mainly in two points. The vast number of *kanji* characters and the ambiguity of the graphic forms causes problems in the character set control operation. And the increasement of character sets may introduce difficulties into the internal code designs. For the resolution of these problems, efforts on both the linguistic field and computer science field will be required.

1. Introduction

Japanese standard character set JIS X0208 (Code of the Japanese Graphic Character Set for Information Interchange) has been widely utilized in Japanese computer systems for over ten years. Almost all the Japanese systems employ this coded character set at least for their external code. However, recent newly developed application fields, e.g. electric publishing, desk top publishing and database services etc., have brought about new situations in the character code field.

Because of the characteristics of the *kanji* or Chinese character, a character set of six thousands characters can not be complete, though it may be almost sufficient for usual Japanese writings. The most comprehensive *kanwa* dictionary, or Chinese-Japanese character dictionary, named *Daikanwa Jiten* [1] lists more than fifty thousand characters, but most of them are very rare characters. This environment causes a problem of missing characters and private extensions.

In addition, the existence of variant forms makes a more complicated situation. These difficulties come from the script of Japanese writing systems, and they are matters of linguistic studies, essentially.

This large character set induces problems also in the computer system side. For example, we need a system that can handle over fifty thousand characters simultaneously. Popular Japanese systems are made on the assumption that they handle only one or two character sets of 8836 characters. Extension of this size is one of the problems to be solved in the computer system side.

This paper discusses these issues about character set and code of Japanese language processing from the following four aspects:

- natural language or writing systems
- script of Japanese language
- interchange character set
- internal code

and makes comments toward the solution of these problems.

2. Japanese Writing System

Japanese writing system uses two kinds of scripts, i.e. *kana* and *kanji*. And also Arabic figures for numbers and Roman alphabets for proper nouns are commonly used in modern writing system.

Japanese can be written in vertical or horizontal line. Traditional writing is vertical (the first line starts from the right side of a page). News papers, and novels employ vertical writing. Horizontal writing is employed by many office documents and science and technology publishings. They are written from left to right, though it used to be written from right to left until about fifty years ago. Japanese writing system may be one of the most complex system in the world of natural languages.

Kana is the Japanese phonetic alphabet, a letter of which represents one syllable of a consonant and a vowel. *Kana* has two styles of *hirakana* and *katakana*, each has about 80 letters of which 48 are basic letters.

Kanji is Chinese script used in Japanese language. *Kanwa* dictionaries or Chinese Japanese character dictionaries list ten thousands to fifty thousands *kanji* characters in them, including some Japanese origin *kanjis*. Among them, two to several thousands are frequently used. The Ministry of education, science and culture

*National Center for Science Information Systems.

of Japan published the *Joyo kanji hyo* or common use *kanji* table in 1981. It includes 1945 characters and this number seems to be reasonable for frequent used *kanjis* in the modern Japanese writing system.

In usual Japanese writings, like newspapers and books, *kanji* and *hirakana* are mainly used with a little ratio of digits, Roman alphabets and symbols. The ratio of these scripts varies according to the type of the writings, for example, articles in the academic journals use more alphabets than newspapers.

The usage of *kanji* and *kana* is a little bit complicated. Generally, proper nouns and most nouns are written in *kanji*. The stem of the verbs and adjectives are written in *kanji*, with the inflection written in *kana*. Particles, and adverbs are usually written in *kana*. These are general rules and there is no generic answer for individual words. In addition, you can use *kana* for all words, even if they have *kanji* representations.

The most difficult thing for beginners is that a *kanji* character has several readings, and thus there are several way to write them in *kana* letters. Especially, in the case of personal names and geographic names, it is usual to find different *kana* representations for the same *kanji* word. On the other hand, it is also ordinary thing to find different *kanji* representations for the same *kana* word. These are similar to heteronyms in English, but we use two different scripts and have more cases than English heteronyms. This situation originates essentially from the fact that Japanese writing systems borrowed only characters (not language itself) from the Chinese writing systems. However, this complicated writing system has over a thousand year of tradition, in the Japanese history.

3. Kanji Script

3.1 variant Forms

The *kana* script in Japanese has less than one hundred characters and there are little difficulties to handle them. The *kanji* script has difficulties because of the number of the script. In addition, variant forms complicate the situation.

Generally, a variant form is defined as a form of a character, of which readings and meanings are the same as another form. Two forms of the figure 1. have the same readings and meanings, and are called a variant form of each other. The form of figure 1(a) is called new form and 1(b) is called old form. In this example, difference of two graphic forms are rather apparent. Two forms of figure 2 are also called variant forms, and in this case, the graphic form difference is just a stroke on the top. In the case of figure 3, differences are just a direction of a stroke.

There is no established general rules for the usage of these forms. In some cases, forms change with times, or vary in different localities. And sometimes vary with personal tastes.

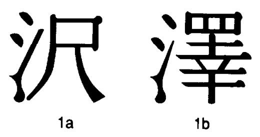

Fig. 1 Example of variant forms (1).

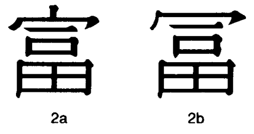

Fig. 2 Example of variant forms (2).

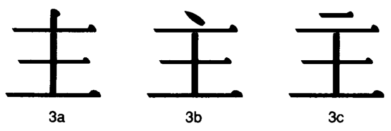

Fig. 3 Example of variant forms (3).

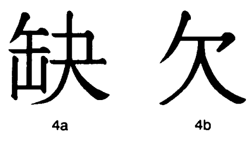

Fig. 4 Example of unclosed grouped character.

Variant forms are shown in *kanwa* dictionaries, which describe meanings, readings and usage of the Chinese characters in Japanese language. However, their way to show these relationships among variant forms is neither explicit nor comprehensive. You can not easily find out all the variant forms of a character by these dictionaries. And often variant form relations are shown one way only. For example, suppose an entry ‘A’ describes that it is an ‘old form’ of ‘B’, still the entry ‘B’ may not describe it is a ‘new form’ of the entry ‘A’. In addition, to list up all the variant forms of the entry ‘A’, you have to search through the whole volume sequentially.

In this paper, we define ‘graphic form’ as an entry in *kanwa* dictionaries. The most comprehensive *kanwa* dictionary has more than fifty thousands ‘graphic forms’ in it, with more than half of them having variant form relationships with other graphic forms.

These graphic forms can be grouped by variant form relationships. We define ‘character’ as such group of graphic forms. There are graphic forms which do not have any other variant form, then, they are characters by themselves. We may call a character of grouped graphic forms as a ‘grouped character’, if distinction is required. A paper reports that the fifty thousand graphic forms of the *Daikanwa Jiten* are grouped into approximately twenty thousand characters. [2]

There are some troublesome cases in the variant form groupings. There are the cases that a graphic form seems to belong to two or more grouped characters. Figure 4 show such a case. The graphic form 4(a) and 4(b) used to be different characters in old times, but in modern Japanese, graphic form 4(a) becomes a variant form (new form) of the graphic form 4(b). (in Chinese, they are still distinguished as two characters). There are not so many such cases, however, a unneglectable numbers exist.

"Character-graphic forms" hierarchy model is not precise in this point. "Graphic forms and variant form relation" network model may better explain the world of *kanji* script.

3.2 Calligraphic Variants

Usually, three examples of the figures 3(a, b, c) are not called as graphic forms. They are regarded as three *jikei* or calligraphic variants of one graphic form. Thus usual *kanwa* dictionaries do not make three entries for each, but they choose one calligraphic variant for printing and make just one entry to represent this graphic form.

The problem is that there is no definite distinction between the calligraphic variants and the variant forms. For example, suppose that a *kanwa* dictionary prints the form of figure 3(a) for its entry, then, you may find the form of figure 3(c), and think of this graphic form as a missing graphic form in that *kanwa* dictionary, while others may recognize 3(c) is just a calligraphic variant of 3(a). *Kanwa* dictionaries do not mention the range of calligraphic variants of entered (printed) graphic form.

Generally, differences of direction of a short stroke, hook at the end of stroke, length of a stroke are regarded as calligraphic variants. However, opinions may differ by person.

If you become more strict for these differences, you shall find more graphic forms, and vice versa. You may find more than one hundred thousand graphic forms, if your attitude is the most strict one. And if your attitude is the most loose one, you may find many calligraphic variant pairs in the *Daikanwa* dictionary.

4. Interchange Character Sets

4.1 JIS X0208

JIS X0208 (Code of the graphic character set for Japanese information interchange) (formerly named JIS C6226) is widely accepted in Japanese systems. This character set was first established in 1978. The number of characters in this character set are shown in the Table 1.

The 32 "potions of ruled lines" are used to generate ruled lines and boxes for tables. Roman alphabets, digits and 32 symbols of 147 are the same as included in the JIS X0201 (same as ISO 646 or ASCII except for a character), so that only by this JIS X0208 all characters can be represented.

Table 1 The number of characters in the JIS X0208.

symbols	147
digits	10
Roman alphabets	52
hirakana	83
katakana	86
Greek alphabets	48
Cyrillic alphabets	66
kanji	6353
portion of ruled lines	32
total	6877

Kanjis are divided into two grades. The first grade contains 2965 *kanjis* frequently used, and they are arranged by their reading sequence. The second grade has 3388 *kanjis* which are less frequently used, are arranged by the radix-stroke sequence.

Before this JIS X0208 *kanji* character set prevailed, several system venders had developed their own code system and character set. After JIS X0208 was established in 1978, all the Japanese system venders changed their code to this JIS code. Most venders changed their internal code and others accepted it as external code.

4.2 JIS X0208 and Variant Forms

This character set has inconsistent attitude for the matter of characters and graphic forms. Thus, there are at most six graphics forms of a character that have six different code positions in this character set, while most of the characters have only one graphic form and a code position. It is perhaps because in 1970s the discussion about variant forms was not yet fully developed.

This attitude causes some problems for the users. In case the graphic form is not included in the JIS X0208 and the other graphic form of the same character is included, handling of this graphic form may differ by person. One may think that this is a missing graphic form and make new entry in his extension character set. The other may substitute the graphic form by the one already in the JIS character set.

This is the typical case of the variant forms problem around a standard character set. This problem is essentially caused by the inconsistent attitude of the JIS X0208 definition. You can not tell if a code position of the JIS X0208 is associated to a 'character' or 'graphic form'. It is a fact that the explanation attached in this standard says that the code position is associated to a character. However, we can find so many explicit variant form pairs in the character set. And this explanation seems to be a contradiction. This situation causes confusion among some users.

4.3 Private Extensions

Naturally, the character sets developed by Japanese system venders before the JIS X0208 were not the same as the JIS character set. In their old character sets, there were some characters not included in the JIS character set. In some cases the number of such characters exceeds one thousand. When the venders changed their character sets to the JIS character set, they could not discard these JIS-external characters, because they already had users who were using these characters. And they made system vender's private extension character sets including these non JIS characters.

In addition to these system venders private extensions, there are user private extensions. Some users are cumulating their own non JIS character sets, having their own character set control systems. Most vender support systems have a function to make the user defined characters by giving the dot patterns for the new characters. Using this function, individual users can define and handle their new character set under their character set control. In some systems, user defined characters were assigned the undefined position in the JIS X0208 character set. As explained in the next section, such treatment caused a problem.

User extensions are generally not large character sets, they usually have several tens to one hundred characters. However, there are a few users that have extension character sets of more than one thousand characters.

4.4 X0208-1983 Revision

JIS X0208 was revised in 1983. Main points of this revision are as follows:
• some of graphic forms were changed according to the *Joyo kanji hyo*:
• following above, some graphic forms of characters exchanged their code position reciprocally.
• accompany with this change, four graphic forms were added:
• some symbols were added.

This revision occasioned hot discussion about graphic forms and characters. Until that time variant forms discussions were not so popular in the computer science field.

In addition, it used the undefined position of the 1978 version, for addition characters. This treatment also caused problems. Because there were many users who already used undefined positions for their private extensions.

The revision also occasioned confusion among users countermoves. Because this revision changed the graphic forms once assigned to a code position. If the change were only addition to the undefined position, such confusions would not be caused.

However, in the present state, the 1983 revision is mostly accepted, as a matter of fact. Perhaps because, the changes of graphic forms were small, from the view point of the most loose attitude.

4.5 New JIS Character Set

The private extensions of each users, obviously, can not be interchanged with each other as they are. This problem became an issue as the application range of Japanese text processing became wider, and the number of users who have private extensions became larger. Especially, with the dissemination of Japanese word processors, printing shops began to accept machine readable manuscripts from authors. In such situations, the existence of private extensions becomes crucial for the printers. This situation caused the request for a new standard character set.

There was another request for the standard character set, that the 1978 version graphic forms should be revived in the new set. In addition, there were requests for alphabets with diacritical marks, used in French and German etc., included in ISO 8859 and ISO 6937.

By the scheduled re-investigation on 1988, these requests were examined and a proposal to make a new character set was approved. This proposal is planned to be voted and will be established and published at the latest in 1990. The main points of this proposal are as follows:
• follows to ISO 2022, new two-byte character set is established:
• this new standard character set will be assigned a new final character:
• about six thousand *kanji* characters not included in X0208 are chosen:
• about 250 alphabets with diacritical marks included in ISO 8859 are chosen:
• user definition area is established.

5. Internal Codes

In this section, internal code designs of four typical Japanese text processing environments, JEF, Shift-JIS, UNIX EUC-JAE and TRON are discussed.

5.1 JEF

JEF (Japanese Extended Feature) is the FUJITSU's Japanese language processing feature on a main frame operating system. JEF is the first vender supplied system which supports Japanese text processing at the operating system level, and had great influence in other mainframe operating systems.

JEF uses one bytes internal code spaces. Figure 5 shows the construction of JEF internal code space. [3]

The EBCDIC is invoked for the one byte space. For the two bytes space, JIS X0208 (with bit 7 on), FUJITSU private extension character set and user defined character set are invoked permanently. This internal code space has 196 one byte positions and 17860 two bytes points.

In the JEF, invocation of the character sets are all locked and can not be changed dynamically. A position or a bit pattern of the code space is assigned a character permanently. An application program can not change the invocations, though different users can invoke different user defined character sets.

Switching control characters or 'case shifts' are required for a character string, because one byte space

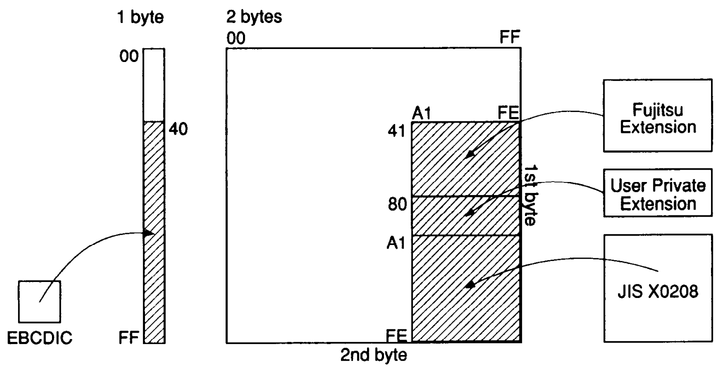

Fig. 5 Internal code space of JEF.

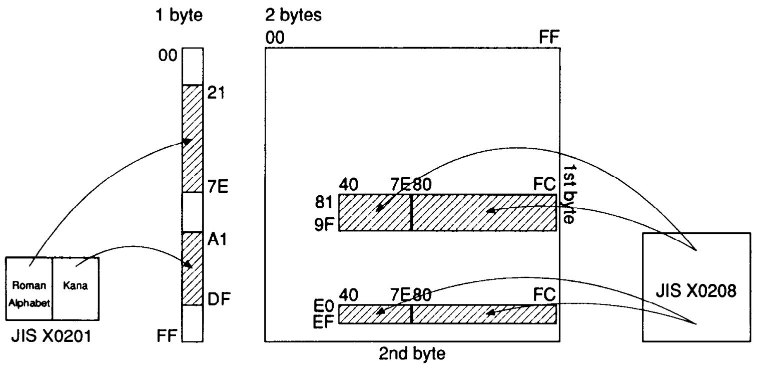

Fig. 6 Internal code space of Shift-JIS.

and two bytes space share the same bit patterns. In JEF there are two one byte control characters which indicate following bit patterns are one byte character or two byte character.

Switching control characters cause complexity in string manipulation algorithms. For example, usual string search algorithm can not be applied if the string contains the switching control characters, and even for concatenation, you have to be sensitive for the switching controls. This complexity is a disadvantage of switching control method.

5.2 Shift-JIS

Shift-JIS is an internal code system widely used with MS-DOS to accommodate Japanese text processing facilities. A certain vender employs shift-JIS internal code in the UNIX operating system. The name 'shift-JIS' comes from the peculiar way to make conversion from the JIS X0208 code to its internal code space. Figure 6 shows the internal code space design of the shift-JIS.

The positions $(E0 \sim EF)_{16}$ of the one byte space are not used. These positions $(E0 \sim EF)_{16}$ and $(81 \sim 9F)_{16}$ are used as the first byte of two bytes space. Second byte of the two byte space uses $(40 \sim 7E)_{16}$ and $(80 \sim FC)_{16}$.

JIS X0201 roman alphabet character set (compatible with ISO 646 but for one character), JIS X0201 *kana* character set, and special graphic symbols are invoked to the one byte space permanently. Because positions $(E0 \sim EF)_{16}$ of JIS X0201 *kana* character set are undefined positions, they can use these positions for other purpose.

The two byte space has 8836 $(31+16)\times(63+125))$ positions. The basic idea of the conversion is to put two rows of JIS X0208 (two 94 positions) into a row of the shift-JIS space $(63+125=188$ positions) with keeping the code order. Next simple algorithm shows the conversion from JIS X0208 to shift-JIS code.

```
Let $c1$, $c2$ the first and second byte of a JIS X0208 character
Let $s1$, $s2$ the result shift-JIS code first and second byte
{$c1$, $c2$, $s1$, $s2$ are all integer of 0 to 255}
$s1=(c1-(21)_{16})/2+(81)_{16}$;
**if** $s1$ is greater than $(9F)_{16}$ **then**
    $s1=s1+(40)_{16}$;
**if** $c1$ is odd number **then**
**begin**
```

```
    s2 = c2 - (21)16 + (40)16;
    if (s2 >= (7F)16) s2 = s2 + 1;
end
else s2 = c2 - (21)16 + (9F)16;
```

There is no need to use switching control characters in shift-JIS unlike JEF. Because one byte space and the first byte of two byte space do not share the same bit pattern, you can tell which space the byte belongs to by inspecting the previous byte. This is an advantage of shift-JIS method.

One of disadvantages of shift-JIS is that it uses control character area $(81\sim9F)_{16}$ for graphic characters, and thus loses compatibility with ISO 2022. For this reason, there may be some MS-DOS application programs that will not work on the shift-JIS coded files.

Another disadvantage is that this internal code space lacks extensionability. Shift-JIS code space has only 8836 positions. Even if you make possible extension to this method the maximum number of positions does not reach 13,000. As a result you can not use two 94×94 character sets in this method, though JEF can use two such sets and possibly be extended to four.

However, these disadvantages comes from the requirement to use *kana* characters in one byte space. There are so many applications which uses JIS X0201 one byte *kana* character set. And without discarding the advantage of no switching control, shift-JIS can not extend its code space.

5.3 EUC JAE

The third example is the Japanese Application Environment (JAE) of the Extended Unix Code (EUC) used in the UNIX operating system. [4] This method is almost compatible with ISO 2022. Figure 7 shows the internal code design of EUC JAE.

In EUC, four character sets (G0, G1, G2, G3) are used. In JAE, JIS X0201 Roman alphabet character set is designated to G0, JIS X0208 to G1, JIS X0201 *kana* character set to G2 and extension two byte character set to G3. These designations are fixed in JAE.

G0 (JIS X0201 Roman alphabet) is invoked to the $(21\sim7E)_{16}$ of one byte space. G2 (JIS X0201 *kana*) is temporary invoked to the $(A1\sim FE)_{16}$ of the one byte space only after the control code SS2 (single shift G2). Similarly, G1 (JIS X0208) is invoked to the $((A1\sim FE)_{16})^2$ of the two bytes space, and G3 (extension two bytes set) is temporary invoked to the same space only after the SS3 (single shift G3) control character. (G2 and G3 single shift invocation is not compatible with ISO 2022 which invokes to $(21\sim7E)_{16}$ space).

Like the shift-JIS, EUC does not need control characters for one byte and two bytes switching. Moreover, JAE handles at most 188 one byte characters and 17672 two byte characters. This number is about the same as JEF's, and thus is more extensionable than shift-JIS.

On the other side, in the EUC JAE, a *kana* character occupies two bytes, even though they are coded in one byte character set, because it always requires preceding SS2. This is a disadvantage of JAE. In addition, existing applications which use $(A1\sim FE)_{16}$ as the one byte *kana* character set do not work in JAE. However, there were not so many such applications on the UNIX environment, it does not cause crucial problems in this point.

EUC realized dynamic invocation of character sets, although it has limitations on ISO 2022 full specifications. In this point, it is new from JEF and shift-JIS method.

5.4 TRON TAD

The last example is the character set handling in TRON TAD (TRON Application Databus). [5] This example is interesting because it is a newly designed operating system after Japanese text processing becomes a popular application. Internal code space of TRON TAD is shown in the figure 8.

Internal code space of TRON uses one byte and two byte space of $(21\sim7E)_{16}$ and $(80\sim FD)_{16}$. This specification is, obviously, not compatible with ISO 2022 as it uses $(80\sim9F)_{16}$ for graphic characters. And invocation

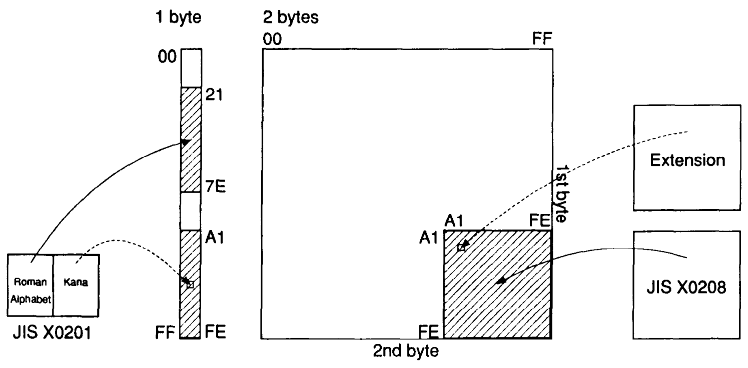

Fig. 7 Internal code space of EUC JAE.

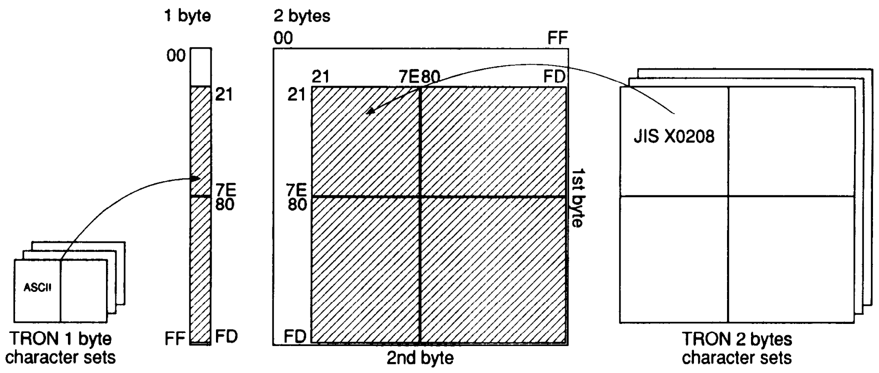

Fig. 8 Internal code space of TRON TAD.

(and designation) of the character sets is made by the language specification code. The language specification code invokes a character set (of either one byte or two bytes) associated to the language, and also invokes the language specific environment, such as input method, typesetting rules, etc.

The internal code space of the TRON can provide 220 positions in one byte space and 48,400 positions in two bytes space. It can acquire such large character code space, because it abandons the compatibility with ISO 2022. However, as for $(21 \sim 7E)_{16}$ region of its character code space, it is intended to use ISO 2022 compatible character sets of 94 or 94×94 positions, such as JIS X0208. But for non compatible regions, the TRON has to develop its own character sets.

In addition, still there remains the problem of missing characters. Even if it provides over 48,000 characters, it can not be the complete character set for the Japanese writing system. Because, at least it can not handle whole entries of the *Daikanwa* dictionary, which has over 50,000 entries.

However, with these unsolved problems, the TRON TAD is the first operating system that handles dynamic invocation of character sets fully. Though its application programs are not yet developed fully, it seems to have a bright future in the field of text processing.

6. Future Prospects

6.1 Variant Forms

As introduced in the chapter 3 and 4, the complexity of variant forms causes problems. For example, when a person recognizes a new graphic form not in JIS X0208, the other person may consider the same graphic form as a calligraphic variant of existing graphic form. Moreover, if that one is a new character not in JIS X0208, someone has to decide which graphic form should be chosen from its possible calligraphic variants. There is no authority to refer to for such work.

This is a problem for creating new character set, and also a problem for the control operation for a character set.

It is linguistic study field to answer this problem. We need a new complete character dictionary which lists all the graphic forms grouped into the characters, with all possible calligraphic variants of them. Of course, the compilers of existing *kanwa* dictionaries did similar studies, in the course of their compiling process. However, we can get only the result of character dictionaries, in which we can not or hardly find information we need.

If we had such a dictionary, character set specification should be changed to refer to this dictionary. A position, or a bit pattern, of a character set should be associated to a character or a graphic form or graphic forms in the new dictionary. If the character set definition is specified in that way, anyone becomes able to tell whether a 'new' graphic form is included in the character set or not referring to the dictionary and the character set specification. Thus the operation of the character set control becomes much easier.

Such a dictionary is a new requirement from the computer science field to linguistic study field. And the study from such view point has not been developed in the linguistic field. New cooperative project of computer scientists and linguistic sholars may be required for the development of such field.

6.2 Standard Character Sets

In the former section, I mentioned that a new standard character set is scheduled to be published. This extension of the standard may introduce new difficulties into existing systems.

Database servers who have their private extension with their character set control systems, have to make effort to change their code (at least of their products) to the new standard. For example the Japan MARC database, which is a book catalog database produced and distributed by the National Diet Library (NDL), has over a million bibliographic records coded in JIS X0208 and the NDL private extension character code. It is considerable work to change this database to a new code.

Moreover, they can not expect to find there are no more missing characters, even if they introduce the new standard character set. Missing characters may decrease in number, but still they exist. And they have to maintain their private extensions. These private extensions will grow again in future and the interchange of them will be the next problem. Thus it is possible that the request for the third standard character set arises. And if the third standard will be established, it will generate the same situation.

Such a situation is, essentially, inevitable for the scripts like *kanji* characters, until we have the complete dictionary and the complete character set. At most we can expect that in future we shall have sufficient character set which scarcely has missing characters. Though we desire that the new standard will bring such a stage, no one can tell until it comes into use.

6.3 Internal Code

The second standard character set introduces difficulties also in the architectures of internal code of operating systems.

Shift-JIS does not have the internal code space for the new set. It seems impossible to extend internal code space of the shift-JIS code, without destroying its advantages. Although JEF has the space for the second set, it is already used for the FUJITSU extension and the user defined character set. Extension of the internal code space, presumably second byte to $(41 \sim 9E)_{16}$, will be required to accommodate the new standard.

This is a task for computer scientists. Requirement for new characters creates new character sets. And existing operating systems can not handle so many character sets simultaneously. Thus extension of internal code space, or dynamic invocation (such as in TRON) technic becomes invitable for these new character sets.

We can not tell that two 94×94 character sets are enough for Japanese *kanji* text processing. On the contrary, we can tell that they will find missing characters in some special fields like text critics of classic literatures. In addition, information interchange between Japan and China or Korea brings many new requests for *kanji* or Chinese character.

Internal code space extension, or dynamic invocation technic supported by operating system will be important issue in the near future.

7. Conclusion

In this paper, character sets and codes of Japanese text processing are introduced and problems around them are discussed from the four view points. They are writing system, script, interchange character set and internal codes of operating systems.

It should be emphasized that these problems have become issue mainly from the widening of application fields. For the daily use of Japanese text processing, such as office documents editing and printing, the existing character set and systems seem to be nearly sufficient, with slight confusions. And such confusion was not rare even in the traditional writing systems.

However, new application brings new problem. Especially, applications for printings and database productions require rare characters to be coded and also special care for variant forms is necessary in such fields. These requirements produce the new character set, and it requests large internal code space or dynamic invocation technic to operating systems.

These problems extend across the computer science field and the linguistic study field. For the solution of these problems, cooperative efforts of both fields are necessary. It is linguistic scholars task to construct a complete character dictionary. On the computer side, we should make efforts to increase the number of characters simultaneously handled.

References

1. Morohashi, T., *Daikanwa, Jiten. Rev. ed., Daishukan*, Tokyo (1984).
2. Tajima, K. The Structure of a KANJI character Set: the problem of open character set and code structure, *IPSJ SIG Reports*, **87-FI**-4-1 (1987) (in Japanese).
3. Kanda, Y., Nishimura, M. design Philosophy of the FUJITSU Japanese Character Information System, *FUJITGSU*, **31**, 5 (1980), 649-660 (in Japanese).
4. Ono, Y. An Implementation Methodology of Japanese Facilities Provided for the UNIX System, *J. IPSJ*, **27**, 12 (1986), 1393-1400 (in Japanese).
5. Sakamura, K. Multi-language character sets handling in TAD, *Real time architecture TRON working group, Institute of Electronics, Information and Communication Engineers*, Tokyo (1987) (in Japanese).

(Received June 26, 1989)