# Character Processing Based on Character Ontology

MORIOKA Tomohiko

## 1 Introduction

We use characters as a basis for data representation in computers, and as a tool for communication over computer networks. Computer programs are made of characters, we exchange mails that are sequences of characters, and many of the contents available over the Internet are realized in the form of characters.

Currently, in the field of information processing, characters are defined and shared using coded character sets. Character processing based on coded character sets, however, has two problems:

1. Coded character sets do not always contain a necessary character
2. Characters in coded character sets have fixed semantics

To resolve the problems, I proposed “Chaon” model which is a new model of character processing based on character ontology. In Chaon model, characters are defined, represented and processed according to its own character databases. Characters in Chaon model are independent from coded character sets for information interchange, and semantics of the characters stored in the database can be freely added or altered.

To realize the character processing based on Chaon model, I started “CHISE (Character Information Service Environment)” project with some other members. In CHISE project, we have developed some systems and databases to edit/process/print characters and texts. CHISE project is an open source project, so the results are freely distributed. We are realizing character processing environment based on Chaon model. In this paper, I explain an overview of the current state of character processing technology in CHISE project.

## 2 Chaon model

In Chaon model of character representation, a character is not a code point of a coded character set, but a set of the features it has. Characters are represented as character objects, and character objects are defined by character features. Character objects and character features are stored in character databases, and a character can be accessed using its feature as a key.

There are various information related with characters, so we can regard various things as character features, for example, shapes, phonetic values, semantic values, code points in various character codes. Figure 1 shows a sample image of a character representation in Chaon model which indicate a character “吉”.

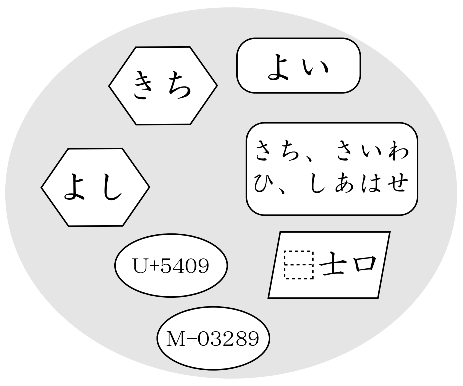


Figure 1: sample of character features to indicate “吉”

In CHISE model, each character is represented by a set of character features, so we can use set operations to compare characters. Figure 2 shows a sample of a Venn diagram of character objects. The diagram indicates that there are common semantic and phonetic values between characters “言”, “云” and “謂” even if they don’t have the same glyph, and characters “云” and “雲” have the common semantic and phonetic values in China.

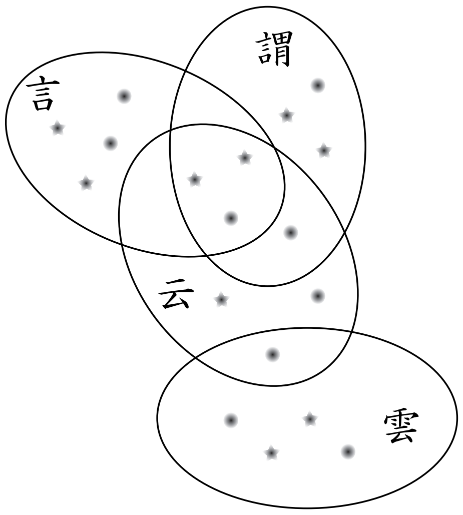


Figure 2: Venn diagram of characters

As we have already explained, a coded character set (CCS) and a code point in the set can also be character features. Those features enable exchanging a character information with the applications that depend on coded character sets. If a character object has only a CCS feature, processing for the character object is the same with processing based on the coded-character model now we are ordinarily using. Namely we can regard the coded-character model as a subset of Chaon model.

## 3 Overview of CHISE system

Character processing based on Chaon model is to represent each character as a set of character features instead of a code point of a coded character set and process the character by various character features. It indicates that character processing system based on Chaon model is a kind of database system to operate character ontology. So the major targets of CHISE project are (1) character database systems, (2) character database contents and (3) CHISE based applications. CHISE Project is working on the targets and provides some results as free software.

As character database systems (and language bindings), following implementations are available:

**libchise** is a library to provide fundamental features to operate character database

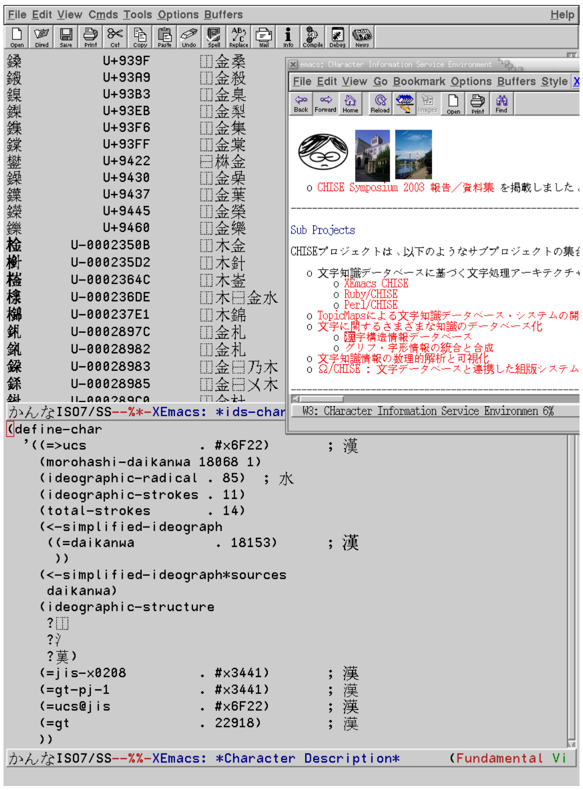
**XEmacs CHISE** is a Chaon implementation based on XEmacs [9] (extensible text editing environment with Emacs Lisp interpreter) [Figure 3]

**Ruby/CHISE** is a Chaon implementation based on Ruby [8] (scripting language)

**Perl/CHISE** is a Chaon implementation based on Perl (scripting language)

For the Chaon implementations, currently two database contents are available as follows:

**CHISE basic character database** is a general character database attached to XEmacs CHISE



```
File Edit View Cmds Tools Options Buffers                          Help
鎟        U+939F        ⿰金桒
鎩        U+93A9        ⿰金殺
鎳        U+93B3        ⿰金臬
鏫        U+93EB        ⿰金梨
鏶        U+93F6        ⿰金集
鏿        U+93FF        ⿰金棠
鐢        U+9422        ⿱楙金
鐰        U+9430        ⿰金喿
鐷        U+9437        ⿰金葉
鑅        U+9445        ⿰金榮
鑠        U+9460        ⿰金樂
     U-0002350B        ⿰木金
     U-000235D2        ⿰木針
     U-0002364C        ⿰木崟
     U-000236DE        ⿰木⿱金水
     U-000237E1        ⿰木錦
     U-0002897C        ⿰金札
     U-00028982        ⿰金札
     U-00028983        ⿰金⿱乃木
     U-00028985        ⿰金⿱乂木
かんなISO7/SS--%*-XEmacs: *ids-char
(define-char
  '((=>ucs                  . #x6F22)        ; 漢
    (morohashi-daikanwa 18068 1)
    (ideographic-radical . 85)  ; 水
    (ideographic-strokes . 11)
    (total-strokes       . 14)
    (<-simplified-ideograph
     ((=daikanwa          . 18153)      ; 漢
      ))
    (<-simplified-ideograph*sources
     daikanwa)
    (ideographic-structure
     ?⿰
     ?氵
     ?𦰩)
    (=jis-x0208          . #x3441)      ; 漢
    (=gt-pj-1            . #x3441)      ; 漢
    (=ucs@jis            . #x6F22)      ; 漢
    (=gt                 . 22918)       ; 漢
    ))
かんなISO7/SS--%%-XEmacs: *Character Description*      (Fundamental Vi
```

```
emacs: CHaracter Information Service Environment
File Edit View Go Bookmark Options Buffers Style

  o CHISE Symposium 2003 報告／資料集 を掲載しました
------------------------------------------------------------
Sub Projects

CHISEプロジェクトは、以下のようなサブプロジェクトの集合

  o 文字知識データベースに基づく文字処理アーキテクチャ
      o XEmacs CHISE
      o Ruby/CHISE
      o Perl/CHISE
  o TopicMapsによる文字知識データベース・システムの開
  o 文字に関するさまざまな知識のデータベース化
      o 漢字構造情報データベース
      o グリフ・字形情報の統合と合成
  o 文字知識情報の数理的解析と可視化
  o Ω/CHISE : 文字データベースと連携した組版システム
------------------------------------------------------------
W3: CHaracter Information Service Environmen 6%
```

Figure 3: XEmacs CHISE

**CHISE-IDS** is a database about shapes of Ideographic characters

As CHISE based applications, we need at least editing system, printing system and character database utilities. As editing system, XEmacs CHISE is available. As printing system, the following programs are available:

**Ω/CHISE** is a multilingual type setting system based on Ω [5] (a multilingual TEX)

**chise2otf** is a converter to process CHISE texts with pLATEX[7] + OTF package [6]

**chise-tex.el** is like chise2otf, but it is implemented as a coding-system of XEmacs CHISE (written by Emacs Lisp)

As character database utilities, there are some emacs lisp programs, Ruby scripts attached to Ruby/CHISE and Perl scripts attached to Perl/CHISE.

In addition, the following system is available:

**Kage** is an automatic Ideographic glyph generating system [10]

Currently Kage is used in Ω/CHISE and chise2otf.

## 4 XEmacs CHISE

XEmacs CHISE (Figure 3), Ruby/CHISE and Perl/CHISE provide operations about character features based on Chaon model. In this paper, I describe a overview of them in the XEmacs CHISE case as an example.

## 4.1 Character representation

XEmacs CHISE represents a character as a character object. A character object is a first class object, just like a character in XEmacs, and completely a different type from the integer.

For programmers of Emacs Lisp applications, a character is represented by a set of character features. The character feature is a pair of a feature key and its value. The feature key must be a symbol and the value can be any Lisp object, that is, a character object itself can be a feature value of other characters, which makes it easy to represent relation network among characters. Namely a character is represented by an association-list of Lisp. In XEmacs CHISE, such kind of association-list which represents a character by its features is named "character-specification (char-spec)".

Each character also has an unique identifier called "character-id" although it is usually hidden from Emacs Lisp users.

## 4.2 Character object related functions

In order to define and handle a character and character features, XEmacs CHISE provides the following built-in functions.

**Function** define-char (*char-spec*)

defines a character object that has a set of character feature *features*, and returns the object. *char-spec* should be an association-list.

**[Example]**

```
(define-char
  '((name               . "CJK RADICAL MEAT")
    (general-category   symbol other)
    (bidi-category      . "ON")
    (mirrored           . nil)
    (ideographic-radical . 130) ; 肉 (radical)
    (ideographic-strokes . 0) ; (body strokes)
    (total-strokes      . 4) ; (total strokes)
    (<-ideographic-component-forms
     ((=ucs               . #x8089)     ; 肉
      ))
    (=ucs               . #x2EBC)       ; 月
    (->subsumptive ; (included variants)
     ((=gt                . 37857)
      (=gt-pj-6           . #x3879)
      (=daikanwa          . 29237)
      )
     ((=ucs@unicode       . #x2EBC)
      )
     ((=big5-cdp          . #x8A73)
      )
     ((=big5-cdp          . #x8958)
      (=gt-k              . 00417)
      (=gt-pj-k1          . #x377D)
      ))
    ))
```

**Function** get-char-attribute (*character feature* ***&optional*** *default-value*)

returns a value of a character feature specified by the key *feature* of a character object *character*.

If the value of *feature* is not defined, *default-value* is returned. If *default-value* is omitted, nil is returned.

**[Example]**

```
(get-char-attribute ?\u2EBC 'name)
→ "CJK RADICAL MEAT"
```

**Function** put-char-attribute (*character feature value*)

adds or changes the value of a feature of a character. This function sets a Lisp object *value* to a value of the character feature specified by the key *feature* of a character object *character*.

**[Example]**

```
(get-char-attribute ?あ 'foo)
→ nil
(put-char-attribute ?あ 'foo 1)
→ 1
(get-char attribute ?あ 'foo)
→ 1
```

**Function** remove-char-attribute (*character feature*)

removes character feature *feature* from character object *character*.

**Function** find-char (*char-spec*)

retrieves the character that has specified features *char-spec*.

XEmacs CHISE provides a map function for character features also. This function aims at finding characters with certain character features or processing characters using its character features.

**Function** map-char-attribute (*function feature*)

This function maps *function* over entries in *feature* (an association-list). *Function* is called with two arguments, a key and a value in the list repeatedly, until all the pairs in *feature* is used up.

**Function** char-attribute-alist (*character*)

returns the features of the character *character*. Every feature of a character is retrieved by this function.

**Function** char-attribute-list ()

returns the list of all existing character features except coded character sets.

XEmacs CHISE has on-memory database per each process besides the CHISE character database shared in the CHISE environment. The on-memory database works as a kind of cache memory for the external database. If a character feature is not found in the on-memory database, the feature value is read from the external database and the value is stored into the on-memory database. If a character feature is found in the on-memory database, XEmacs CHISE does not access the external database.

Modification functions for characters or their features of XEmacs CHISE, such as put-char-attribute, work for on-memory database. However it is volatile. So XEmacs CHISE has a function to save the character data.

**Function** save-char-attribute-table (*feature*)

saves each character's value of character feature *feature* into the CHISE character database.

XEmacs CHISE has functions to clear character data in the on-memory database to be able to reread from the CHISE database.

**Function** reset-char-attribute-table (*feature*)

clears character feature *feature* of every character in the on-memory database to be able to reread each value of the *feature* from the CHISE database.

**Function** reset-charset-mapping-table (*coded-charset*)

clears decoding-table of *coded-charset* in the on-memory database to be able to reread from the CHISE database.

XEmacs CHISE has a function to read every character's value of a specified feature from the CHISE database at a burst.

**Function** load-char-attribute-table (*feature*)

reads every character's value of a specified *feature* from the CHISE database at a burst.

In addition, there is a function to register a character feature.

**Function** mount-char-attribute-table (*feature*)

registers character feature name *feature* as a target to read from the CHISE character database.

By the way, Ruby/CHISE and Perl/CHISE don't have on-memory database, so written character data are written into the CHISE database directly. So there are no functions to sync between on-memory database and the CHISE database.

## 5 Character features

### 5.1 Categorization

In the character processing based on Chaon model, it is important to analyze characters and their various properties and behaviors and represent them as character features. We can find sundry properties and behaviors of characters, and we can use infinite kind of character features. However common character database requires a guideline about character features. So we think that it is feasible to regard each character feature as an abstraction of an operation for characters.

In the point of view, character features can be categorized like following:

1. general character properties (such as descriptions of dictionaries)
2. mappings for character IDs
3. relations between characters

For example, radicals, strokes and phonetic values can be classed into the category 1, code points of UCS [2] can be classed into the category 2 and relations between character variants can be classed into the category 3.

Information of the category 2 is used for processing about character codes, such as code conversion. Processing about character codes consists of two kind of operations: encoding and decoding. To encode a character by a CCS is to get the CCS feature's value in the character. To decode a code-point of a CCS is to search a character whose value of the CCS feature is the code-point. Processing about character codes should be fast, so the CHISE character database has special indexes for decoding.[1]

For the processing about character variants, information of the category 3 is used.

### 5.2 Description for complex information

For development of a general purpose character database, we may find some cases that there are different kind of usages, purposes, applications, sources, interpretations, theories, etc. so it is hard to chose one feature value and we want to provide alternative values. In that cases, we may want to add metadata, such as sources of the values. To resolve the problem, we have to introduce structured feature value or structured feature name (key).

To represent structured feature values, a format named "character reference (char-ref)" is used in CHISE. It is a kind of property-list of S-expression (Lisp), property name indicates kind of metadata and property value indicates its data. As a special property name, :char is reserved to indicate a character which is added the metadata. Currently :sources is defined to indicate information source.

[1]For the special treatment, we distinguish the category 1 and 2, but it seems that there are no essential differences.

CHISE also has a format to represent structured feature names. In the structured feature names, “domain identifiers” and/or “metadata identifiers” are added to ordinary (base) feature names. The format is defined as following definitions:

```
<concrete feature name>
      := <base feature name> @ <domain identifier>
      |   <concrete feature name> / <domain identifier>

<metadata feature name>
      := <concrete feature name> * <metadata identifier>
```

For example, when total strokes is represented by character feature total-strokes and ucs is used as a domain identifier, concrete feature name is total-strokes@ucs. When source is represented by metadata identifier sources, total-strokes@ucs’s source is represented by metadata feature name total-strokes@ucs*sources.

If there is a correspondence between different kind of features, such as radical and body-strokes, we can represent the correspondence by a domain identifier. For example, when radical is represented by ideographic-radical and body-strokes is represented by ideographic-strokes, two concrete feature names

```
ideographic-radical@ucs
ideographic-strokes@ucs
```

are corresponding.

### 5.3 Inheritance of character definition

If we construct a large scale character database including a lot of character variants, inheritance of character definition is good way to avoid to write a lot of common features. So CHISE introduces four special features to represent parent and child relations:

<–**subsumptive** defined character is a child of each character indicated by its value

<–**denotational** likewise

–>**subsumptive** each character indicated by its value is a child of the defined character

–>**denotational** likewise

## 6 Database contents

Character database is a fundamental part of the Chaon character representation model. Users can, of course, freely define or modify characters by adding new character features, but a rich and accurate database would be a great place to start, and it will also attract new users. We have thus developed a standard character database for Chaon implementations. Currently two database distributions are available:

1. CHISE basic character database
2. Database about structure information of Ideographs (CHISE-IDS)

The former is a basic character database attached in XEmacs CHISE which is realized by a collection of define-chars while the later is a database for Ideographs to represent information about shapes which is represented by “Ideographic Description Sequence (IDS)” format defined in ISO/IEC 10646-1:2000 [2].

Structure information of Ideographs is information about combinations of components of Ideographs. A lot of Ideographs can be represented by a combination of components, so the information is a useful. It is not only representation of abstract shapes, but also related with semantic values and/or phonetic values. So we planned to develop a database about structure information of Ideographs for every Ideograph which consists of combination of components. In the 2001 fiscal year, we realized CHISE-IDS database with supporting of “Exploratory Software Project (未踏ソフトウェア創造事業)” run by IPA (Information-technology Promotion Agency, Japan) [11]. Currently it supports CJKV Unified Ideographs and Extension A of ISO/IEC 10646-1:2000 [2] and Extension B of ISO/IEC 10646-2 [3]. We are also working for representative glyph image of JIS X 0208:1990 and Daikanwa Dictionary.

Before we developed CHISE-IDS database, there are some databases including structure information of Ideographs: CDP database [1] by Academia Sinica, database about gaiji (private used character) used in CBETA and “Konjaku Mojikyo” [4]. These databases use original formats so it is not easy to convert to IDS format. Konjaku Mojikyo is a proprietary software so their data are not opened for the public. In the view of licence, CDP database and CBETA database are available with free software licenced under the term of GPL while Konjaku Mojikyo is not. So we converted CDP database and CBETA database to IDS and integrate them with CHISE database.

Currently CHISE basic character database (is a part of XEmacs CHISE) and CHISE-IDS package are distributed separately. However CHISE-IDS package provides an installer to integrate CHISE-IDS database files with CHISE basic character database. CHISE-IDS package also have some utility programs to use structure information of Ideographs in XEmacs CHISE:

**ids.el** IDS parser

**ids-read.el** utility to read CHISE-IDS database files into XEmacs CHISE

**ids-dump.el** utility to dump structure information of Ideographs stored in XEmacs CHISE (represented by character feature ideographic-structure) into CHISE-IDS database format

**ids-util.el** utility to convert structure information of Ideographs into other representative glyph images corresponding with specified domains

**ids-find.el** utility to search Ideographs by components

Currently, ids-find.el has two commands to search Ideographs by components named: ids-find-chars-including-components (= ideographic-structure-search-chars) and ids-find-chars-covered-by-components.

If you type

M-x ideographic-structure-search-chars [CR] components [CR]

Ideographs that have every component are displayed. (Figure 4)

If you type

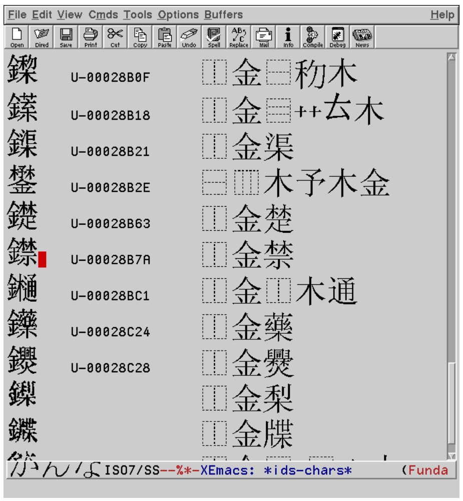
M-x ids-find-chars-covered-by-components [CR] components [CR]

Ideographs that consists of one or more usage of every component are displayed. (Figure 5).

## 7 Conclusion

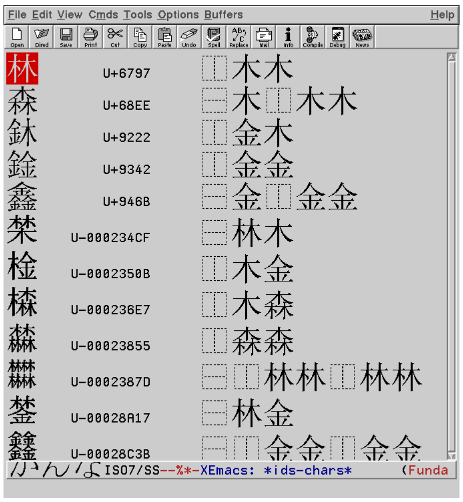
I have described a brand new character processing model Chaon and overview of CHISE project and character processing in CHISE systems such as XEmacs CHISE. Chaon character representation model is powerful and radical enough to solve the problems that the present coded character model has, and the implementation of XEmacs CHISE or other CHISE systems have shown that the model is feasible enough to build a application on.

Chaon model gives users freedom to create, define and exchange characters of their need, as it is easy to change character databases or modify character features dynamically. XEmacs CHISE provides a good framework to experiment the character representation. With the CHISE basic character database, XEmacs CHISE can handle various characters including characters defined in Unicode. Even if a character is not defined in Unicode, users



```
File Edit View Cmds Tools Options Buffers                    Help
鑗     U-00028B0F     ⿰金⿱𥝖木
鏼     U-00028B18     ⿰金⿳++厷木
鐻     U-00028B21     ⿰金渠
鏊     U-00028B2E     ⿱⿲木予木金
鐽     U-00028B63     ⿰金楚
𨭺     U-00028B7A     ⿰金禁
𨯁     U-00028BC1     ⿰金⿰木通
鑅     U-00028C24     ⿰金藥
鑁     U-00028C28     ⿰金爨
鑗                    ⿰金梨
鑠                    ⿰金牒
かん/よISO7/SS--%*-XEmacs: *ids-chars*        (Funda
```

Figure 4: result of M-x ideographic-structure-search-chars [CR] 木金 [CR]



```
File Edit View Cmds Tools Options Buffers                    Help
林          U+6797     ⿰木木
森          U+68EE     ⿱木⿰木木
鉌          U+9222     ⿰金木
銓          U+9342     ⿰金金
鑫          U+946B     ⿱金⿰金金
𣓏     U-000234CF     ⿱林木
𣔋     U-0002350B     ⿰木金
𣛧     U-000236E7     ⿰木森
𣡕     U-00023855     ⿰森森
𣡽     U-0002387D     ⿱⿰林林⿰林林
𨨗     U-00028A17     ⿱林金
𨰻     U-00028C3B     ⿱⿰金金⿰金金
かん/よISO7/SS--%*-XEmacs: *ids-chars*        (Funda
```

Figure 5: result of M-x ids-find-chars-covered-by-components [CR] 木金 [CR]

can add it into CHISE database to define it by its features. Users can handle each character based on their point of view or policy. For example, XEmacs CHISE provides some unification rules or mapping policies about Unicode. With the CHISE-IDS database, users can search Ideographs easily. This method is also available for non-Unicode characters.

Currently, CHISE project provides basic elements to process Chaon based text: text editor (XEmacs CHISE), scripting languages (Ruby/CHISE and Perl/CHISE) and type setting system (Ω/CHISE). You can try KNOPPIX/CHISE which is a DVD bootable GNU/Linux system including XEmacs CHISE and Ω/CHISE. Its image is available at `http://kanji.zinbun.kyoto-u.ac.jp/projects/chise/dist/KNOPPIX/`.

CHISE project is an open source project, so its results are distributed as free software. Information about CHISE project is available at:

- `http://cvs.m17n.org/chise/`
- `http://kanji.zinbun.kyoto-u.ac.jp/projects/chise/`

These WWW pages, various programs and data are managed by CVS (a kind of revision control system), so users can get the latest snapshot. There are mailing-lists about CHISE project: for English and Japanese. If you are interested in CHISE project, please join to the lists.

## Bibliography

[1] 漢字庫. `http://www.sinica.edu.tw/~cdp/zip/hanzi/hanzicd.zip`.

[2] International Organization for Standardization (ISO). *Information technology — Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane (BMP)*, March 2000. ISO/IEC 10646-1:2000.

[3] International Organization for Standardization (ISO). *Information technology — Universal Multiple-Octet Coded Character Set (UCS) – Part 2: Supplementary Planes*, November 2001. ISO/IEC 10646-2:2001.

[4] 今昔文字鏡. http://www.mojikyo.com/.

[5] The Omega typesetting and document processing system. http://omega.cse.unsw.edu.au:8080/.

[6] Open Type font 用 VF. http://psitau.at.infoseek.co.jp/otf.html.

[7] Ascii 日本語 TEX(pTEX). http://www.ascii.co.jp/pb/ptex/.

[8] The object-oriented scripting language Ruby. http://www.ruby-lang.org/.

[9] XEmacs. http://www.xemacs.org/.

[10] 上地 宏一. 漢字フォント自動生成サーバ “影 KAGE” の構築 — 文字コードの枠組みを越える次世代漢字処理の提案 —. **漢字文献情報処理研究**, 3:143–147, 2002.

[11] 守岡 知彦 and クリスティアン・ウィッテルン. 文字データベースに基づく文字オブジェクト技術の構築. In **情報処理振興事業協会 平成** *13* **年度 成果報告集**. 情報処理振興事業協会, 2002. http://www.ipa.go.jp/NBP/13nendo/reports/explorat/charadb/charadb.pdf.