UTF-2000 — 汎用文字符号に依存しない文字表現系の展望

守岡 知彦

京都大学 人文科学研究所 附属 漢字情報研究センター

概 要

文献や書誌情報の電子化において文字や文書をどのように表現するかは重要な問題である。従来、これらの電子化は符号化文字モデルに基づき何らかの符号化文字集合を用いることにより行われて来た。これは使用可能な文字の集合や各文字の定義を符号化文字集合の定義に委ねる方法であるが、各利用者や用途が異なる以上、万人を満足させる定義を策定する事は容易ではない。また、外字による解決は情報交換の際に問題を生じさせる。こうした問題を解決するために従来の符号化文字モデルに代わる、各文字の定義自体を自由に編集可能な文字表現モデルを提案し、その試験実装である XEmacs UTF-2000 を紹介する。

1 はじめに

電子化文書は作成・加工・転送などが容易であり、非常に低コストで情報を共有することが可能である。また、検索やハイパーリンクなどにより、情報を以前の文書とは異なった視点で加工し提示することを可能にしている。このような点で、文書の電子化技術は情報化社会の基盤技術であるだけでなく、これまでに蓄積されて来た知を新たな視点で広く人類全体で共有可能にする可能性を秘めており、実際、そのような動きが社会のさまざまな局面で起こって来ている。

このような素晴らしい利便性・可能性は計算機で文字を扱えるようにしたことによって起こったことであるが、見方を変えれば、計算機で扱うことが不可能な文字ではこのような利便性は享受できないし、同様に、計算機で扱いにくい文字ではその効果は減少するということになる。実際、（この問題だけが原因ではないと思われるものの）現在もっとも扱うことが容易な ASCII [1] で表現可能な英語を使う社会・分野ではこのような動きがもっとも進んでおり、それに比べれば日本語圏・漢字圏は遅れている。

英語に比べて日本語が、ラテン文字に比べて漢字が計算機で扱いづらい理由の１つは、現在の計算機で文字を扱う仕組みがもともと英語圏でのラテン文字のためにデザインされているからではないかと考えられる。英語で用いるラテン文字は数が少なく、それ故に文字の同定も容易である。タイプフェース上のデザイン差と文字の違いは容易に区別可能であり、概念上の区別を付けやすい。しかし、漢字は文字の数が非常に多く、その同定は容易ではない。字形のデザイン差と字体の差と文字の違いは連続的であり、その区別は恣意的なものにならざるを得ない。一方、文字の表示上の複雑さという観点からいえば、ラテン文字にしても漢字にしても概念上の文字と表示上の文字がほぼ１対１に対応するのに対し、インド系の諸文字では複数の文字が複雑な結合をすることによって表示上の文字が形作られる。文字体系によってはその組合せ方は無数にあり、組合せ規則を明示することが困難である場合がある。

このように世界には異なる性質を持つ文字体系が存在している。しかし、計算機で文字を扱うためのモデルは各種文字用の修正を施したとはいえ、基本的に英語用ラテン文字のためにモデル化さ

れたものであるといえる。そのため、ラテン文字やそれに近い性質を持つ文字体系は扱うのが容易であり、そうでない文字体系は扱うのが難しいという結果になっているといえる。また、各文字体系のラテン文字に近い性質の部分は扱うのが簡単で、遠い部分は難しいということになる。漢字でいえば、フォントを作れば表示は簡単だが、文字の数が多いので、文字の同定や検索は難しくなるということになる。

このような問題を抜本的に解決するには、ラテン文字用にデザインされた文字モデルを各文字体系に無理矢理適用する代わりに、各文字や文字体系の性質を記述可能なメタな文字表現系とそれを処理する文字処理モデルを構築し、さまざまな文字の性質に適した処理を利用可能にすることが重要であると考えられる。即ち、漢字であれば漢字の持つ性質や各文字の性質に関する知識を記述したデータベースを作成し、そのデータベースを参照することによって文字を処理するシステムを構築する訳である。

このような目的のために、我々は現在 “UTF-2000” と呼ぶ新たな文字処理モデルを開発している。ここではその概要に関して述べる。

2　符号化文字モデルと UTF-2000 における文字モデル

現在、多くの文字処理システムでは、文字そのものを扱う代わりに、文字に固有の番号を振って、その番号で文字を表す手法を用いている。ここで、文字と番号の対応規則を「符号化文字集合」(Coded Character Set; CCS) もしくは「文字符号」(character code) と呼び、文字に振られた番号を「符号位置」(code point) と呼ぶ。また、このように、有限の文字の集合を定め、各文字に固有の番号を振りその番号で文字を表現する手法のことを、ここでは『符号化文字モデル』と呼ぶことにする。

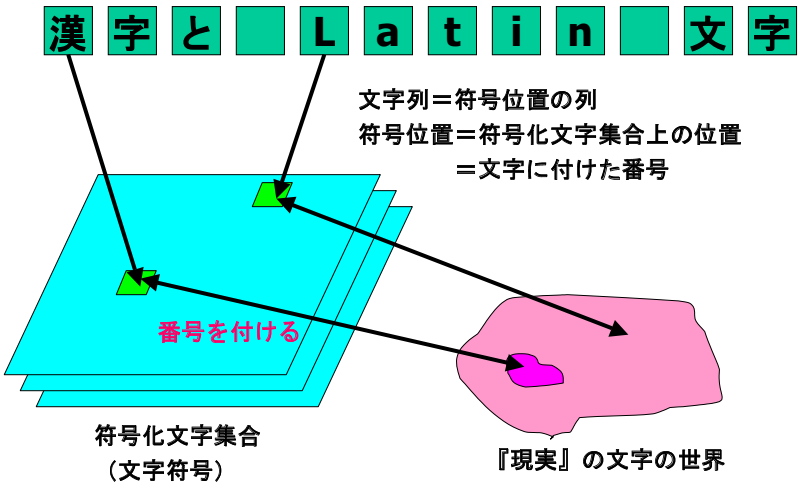

図 1: 符号化文字モデルの概念図

図 1 に示すように、符号化文字モデルでは文字列は文字に振られた番号の列で表現される。文字に関する知識は符号化文字集合の定義の中に存在し、通常、計算機の中には存在しない。このため、計算機は少ない記憶量で文字を表現できる半面、計算機が扱うことができる文字の種類や文字の概念は符号化文字集合の定義に束縛される。

これに対して、図 2 に示すように、UTF-2000 では文字列を文字オブジェクト（への参照）の列で表現する。文字オブジェクトは文字に関するさまざまな属性の集合で表現されるようなもので、文字に関する知識を表現したものである。文字オブジェクトの属性としては、文字の種類、音価・部首・画数などの属性や、その他のさまざまなものが考えられる。既存の符号化文字の世界の文書やフォントなどの資産を利用したり、既存のシステムを利用している人々と情報交換することは需要であるが、これも符号化文字集合の種類と符号位置を文字の属性の一種として捉えることによって可能となる。

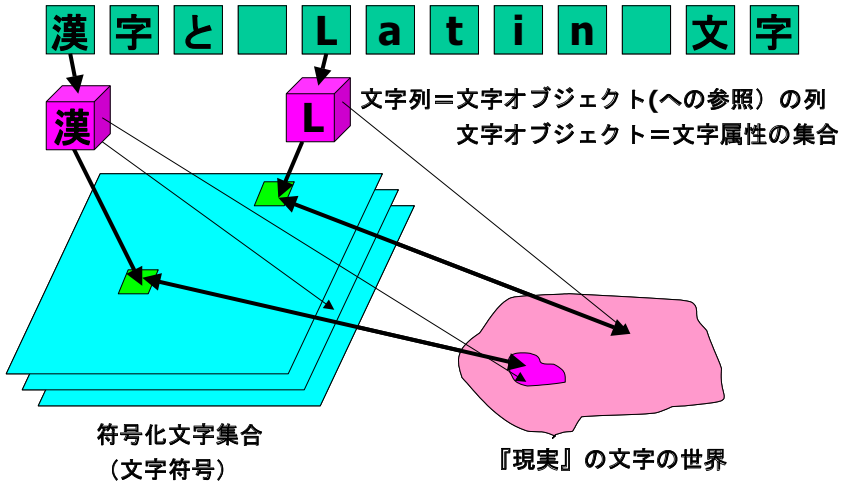

図 2: 文字オブジェクトモデルの概念図

UTF-2000 の方式は文字に関する知識を直接プログラムするのではなく、データベース化してそれを参照する方法であるので、計算機が扱うことができる文字の種類や文字の概念は符号化文字集合の定義に束縛されない。使用する文字データベースを取り換えることによって、容易に文字の種類や概念を変更可能である。このような高度な柔軟性を持つ半面、多数の文字を扱う場合、多量の記憶量が必要となると考えられる。このため、文字データベースを柔軟性を損なうことなく効率的に扱うための機構が必要となる。

3 XEmacs UTF-2000

現在、我々は UTF-2000 モデルに基づく試験実装として **XEmacs UTF-2000** と呼ぶシステムを開発している。

XEmacs-UTF-2000 は **XEmacs** [7] と呼ばれる対話型統合環境を基にしている。XEmacs は **GNU Emacs** [5] と呼ばれる拡張可能なエディタを中心とした対話型統合環境を整理・拡張し、絵なども扱えるようにしたものである[1]。一方、**Mule** (MULtilingual enhancement of GNU Emacsen) [4] は電子技術総合研究所の半田剣一氏が中心となって開発している GNU Emacs の多言語拡張で、現在は GNU Emacs に統合されている。この Mule 機能を XEmacs に統合した XEmacs-Mule の開発も進められており、XEmacs の一部として配布されている。XEmacs UTF-2000 はこの XEmacs-Mule を基に大幅に改変したものである。

[1] GNU Emacs の次の版 (21.1) でも絵も表示することができる。

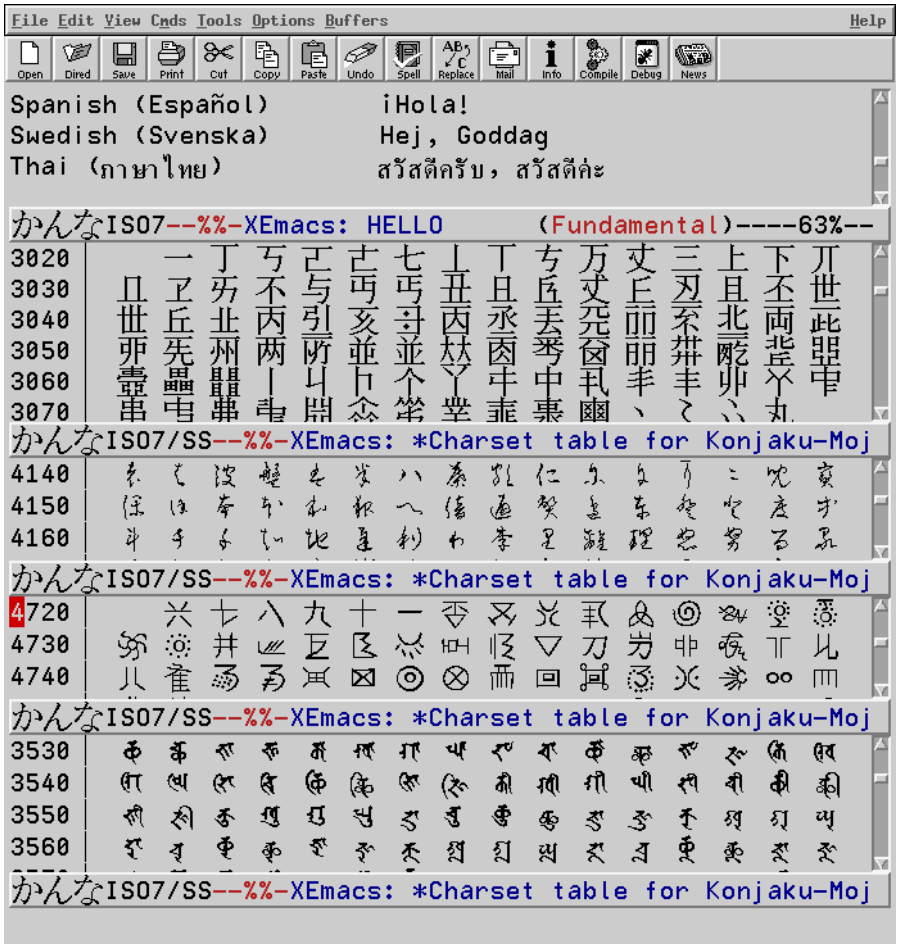

図 3: XEmacs UTF-2000

GNU Emacs および XEmacs は Emacs Lisp 言語を使って機能を拡張することができ、実際に Emacs Lisp で記述されたさまざまなアプリケーションが作成され利用されている。例えば、GNU Emacs/XEmacs の中で電子メールやネットニュースを読み書きすることができるし WWW 頁を見ることができる。しかし、これらは元々 Latin-1 (ISO-8859-1) などの 1 byte の符号化文字集合でしか利用可能ではなかった。

これに対し、Mule は GNU Emacs/XEmacs の利便性・拡張可能性をそのままに、利用できる符号化文字集合を大幅に拡張したものであり、さまざまな文字を表示する機能と各種言語用の入力機能をはじめとする多言語化機能を提供する。Mule 機能を持った GNU Emacs/XEmacs では GNU Emacs/XEmacs の機能がさまざまな文字・言語で利用可能となる。[2]

Mule では文字は符号化文字集合の種類を表す “charset-id” と符号位置の組で表現される。charset-id は最大 128 個が同時に利用可能である。但し、利用可能な符号化文字集合は ISO 2022 [2] の 94 文字集合、96 文字集合、94 × 94 文字集合、96 × 96 文字集合に限られる。Big5 のような ISO

[2]Mule が行った拡張は本質的に国際化ではなく多言語化であり、さまざまな種類の文字・言語を同時に表示・入力・処理することを可能とするものである。逆に、ある言語環境に特化した設定を行ったり、メニューや説明文等の表示言語を切替えるといった国際化機能は貧弱である。

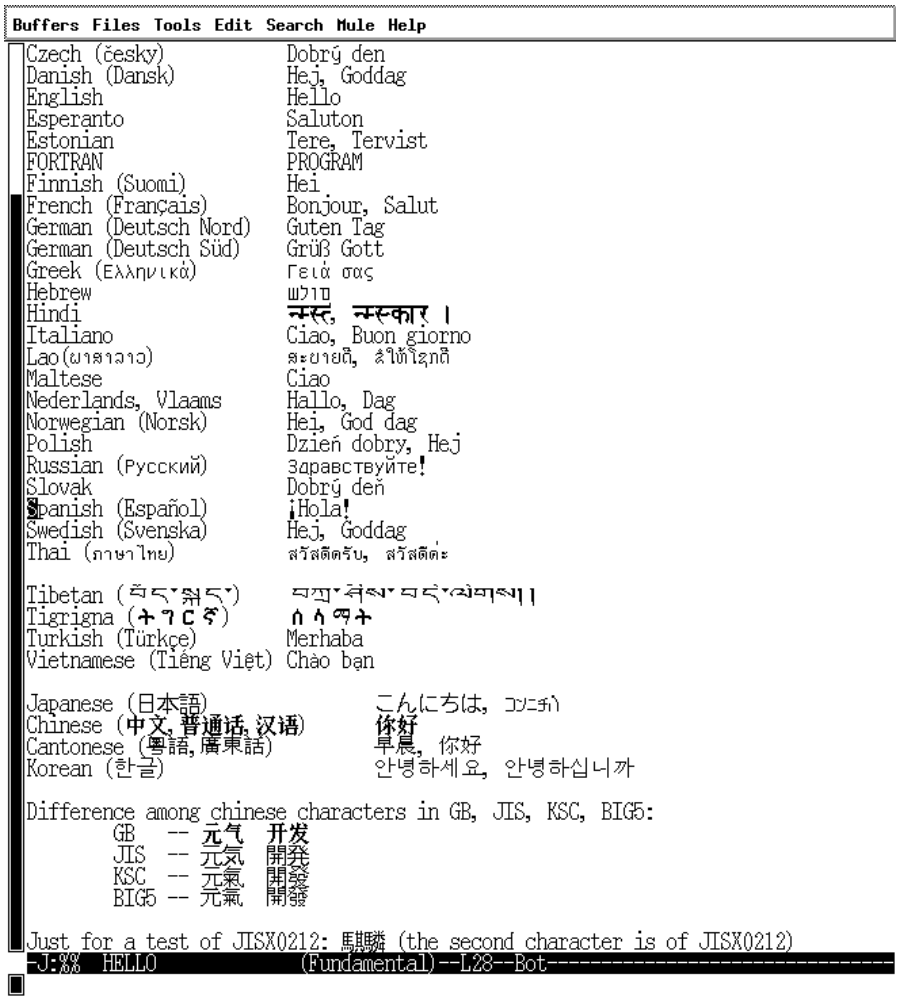

図 4: Mule (GNU Emacs 20.7)

2022 の図形文字集合の構造に適合しないものは ISO 2022 の構造に合うように変換して扱う必要がある。また、文字表現空間は 19 bit であり、既に利用可能な空間が枯渇してきている。さらに、Mule では文字は charset-id と符号位置の組で表現されるので、符号化文字集合が異なれば本来同じ文字であっても異なる文字として扱われてしまうという問題もある。

XEmacs UTF-2000 は GNU Emacs, XEmacs, Mule の利点を継承しつつ、これらの問題を解決するために以下に述べるような拡張・改変を行った。

3.1 文字オブジェクトと文字属性

XEmacs UTF-2000 では UTF-2000 モデルに基づき文字データベースを参照することによって文字を処理するようにしている。このため、文字データベースを利用しやすいように Mule の内部表現を変更している。文字表現空間は 30 bit に拡大されており、同時に利用できる文字数は大幅に増えている。Mule における文字の内部表現に相当するものは文字オブジェクトの id で『文字 id』と呼ぶ。この文字 id を用いて文字オブジェクトを参照することにより、文字に関する処理が

行われる。

文字オブジェクトは文字属性の集合で定義される。定義された文字オブジェクトには固有の『文字 id』が割り当てられる。この文字定義のために XEmacs UTF-2000 では define-char という組込み関数を定義している：

関数 **define-char** (*attributes*)

文字属性 *attributes* で表現される文字オブジェクトを定義し、その文字オブジェクトを返す。
文字属性 *attributes* は連想リストである。

各文字オブジェクトの属性は組込み関数 get-char-attribute で参照できる：

関数 **get-char-attribute** (*character attribute*)

文字オブジェクト *character* の属性 *attribute* の値を返す。

また、組込み関数 put-char-attribute で文字オブジェクトの属性の値を追加・変更することもできる：

関数 **put-char-attribute** (*character attribute value*)

文字オブジェクト *character* の属性 *attribute* の値を *value* に設定する。

この他、文字オブジェクトの全文字属性を連想リストとして返す組込み関数 char-attribute-alist や全ての文字属性名をリストとして返す組込み関数 char-attribute-list などを設けている。

3.2 coded-charset

XEmacs UTF-2000 における文字の表現は符号化文字集合に依存していない。このため、Mule におけるような意味での Mule-charset は必要ない。しかしながら、既存の文字符号の世界と情報交換したり、既存のフォントを利用したり、既存の Mule 実装との互換性を実現し既存の Mule アプリケーションを利用するために、何らかの形で Mule-charset に相当するものを実現することは重要である。そこで、XEmacs UTF-2000 では符号化文字集合の抽象として “coded-charset” を設けている。

XEmacs UTF-2000 の coded-charset に関する API は XEmacs-Mule における Mule-charset に関する API の上位互換になるように設計されている。XEmacs-Mule における Mule-charset と同様に、charset 型が存在し、charset 型オブジェクトはシンボルで表現される名前を持つ。Mule では Mule-charset は文字の内部表現に依存しており、その構造上、94 文字集合、96 文字集合、94 × 94 文字集合、96 × 96 文字集合の 4 種類に限定されていたが、XEmacs UTF-2000 ではそのような制約は存在しないので、任意の符号化文字集合を表現できるように、$\{94|96|128|256\}^{1\sim4}$ 文字集合に拡張され、符号位置が最大 4 byte までで表現可能な任意の符号化文字集合を利用できるようになっている。例えば、UCS の基本多言語面 (BMP) [3] は 256 × 256 文字集合で表現可能であり、UCS-4 全体もまた 256^4 文字集合で表現可能である。新設の 128^n 文字集合に対して、関数 charset-chars は 128 を返し、関数 charset-dimension は n を返す。同様に、256^n 文字集合に対しては、それぞれ、256 と n を返す。

XEmacs UTF-2000 の coded-charset は文字の内部表現と直接関係しないので、coded-charset に収録された各文字とその coded-charset における符号位置の対応表を設けて文字と符号位置の関係を表現している。この対応表として、符号位置から文字を得るための “decoding-table” と、文字から符号位置を得るための “encoding-table” の 2 種類を設けている。このうち、後者は文字属性の一種として扱っている。即ち、coded-charset の名称を属性名とする文字属性は “encoding-table” の要素と見倣されるようになっている。関数 define-char もしくは関数 put-char-attribute で coded-charset の名称を持つ文字属性を指定した時、その値は指定された coded-charset の符号位置と見倣され、指定された coded-charset の decoding-table も同時に設定される。このような仕組みにより、陽に変換表を指定することなく、また、内部表現に関する言及なしに、文字オブジェクトモデルだけでさまざまな符号化文字集合を表現可能である。

表 1, 2, 3, 4, 5, 6, 7 に第 4 節で述べる UTF-2000 用標準文字データベースでの文字定義で用いられている coded-charset の符号位置を表す属性名の一覧を挙げる。

表 1: 94 文字集合での符号位置を表す文字属性

ascii	ISO 646 国際基準版の符号位置
katakana-jisx0201	JIS X 0201-カタカナ での符号位置
latin-jisx0201	JIS X 0201-Latin での符号位置
lao	Mule-Lao での符号位置

表 2: 96 文字集合での符号位置を表す文字属性

latin-iso8859-1	ISO 8859-1 での符号位置
latin-iso8859-2	ISO 8859-2 での符号位置
latin-iso8859-3	ISO 8859-3 での符号位置
latin-iso8859-4	ISO 8859-4 での符号位置
cyrillic-iso8859-5	ISO 8859-5 での符号位置
arabic-iso8859-6	ISO 8859-6 での符号位置
greek-iso8859-7	ISO 8859-7 での符号位置
hebrew-iso8859-8	ISO 8859-8 での符号位置
latin-iso8859-9	ISO 8859-9 での符号位置
thai-tis620	TIS 620 での符号位置
latin-tcvn5712	TCVN 5712 (VSCII 2) での符号位置
latin-viscii-lower	Mule-VISCII-lower での符号位置
latin-viscii-upper	Mule-VISCII-upper での符号位置
ipa	Mule-IPA での符号位置

表 3: 256 文字集合での符号位置を表す文字属性

latin-viscii	VISCII での符号位置

表 4: 94^2 文字集合での符号位置を表す文字属性

japanese-jisx0208-1978	JIS X 0208:1978 での符号位置
chinese-gb2312	GB 2312 での符号位置
japanese-jisx0208	JIS X 0208:1983 での符号位置
korean-ksc5601	KS C 5601 での符号位置
japanese-jisx0212	JIS X 0212 での符号位置
chinese-isoir165	CCITT 拡張 GB 2312 での符号位置
japanese-jisx0208-1990	JIS X 0208:1990 での符号位置
chinese-cns11643-1	CNS 11643 第 1 面での符号位置
chinese-cns11643-2	CNS 11643 第 2 面での符号位置
chinese-cns11643-3	CNS 11643 第 3 面での符号位置
chinese-cns11643-4	CNS 11643 第 4 面での符号位置
chinese-cns11643-5	CNS 11643 第 5 面での符号位置
chinese-cns11643-6	CNS 11643 第 6 面での符号位置
chinese-cns11643-7	CNS 11643 第 7 面での符号位置
japanese-jisx0213-1	JIS X 0213 第 1 面での符号位置
japanese-jisx0213-2	JIS X 0213 第 2 面での符号位置

表 5: 256^2 文字集合での符号位置を表す文字属性

ideograph-daikanwa	大漢和番号（ダッシュありを除く）
chinese-big5	Big5 での符号位置

表 6: 256^3 文字集合での符号位置を表す文字属性

mojikyo	文字鏡番号

表 7: 256^4 文字集合での符号位置を表す文字属性

ucs	UCS での符号位置

3.3 組込み文字

XEmacs UTF-2000 では、文字は関数 define-char で文字属性の集合を指定することによって定義されるが、最初に文字を定義する時に何も文字が存在しないとすれば、文字の定義プログラムを文字列で表現できないことになる。これは不便であるので、関数 define-char で文字を定義する前からあらかじめ定義されている文字を設けている。これを『組込み文字』(builtin-character) と呼ぶ。

組込み文字としては ISO 8859-1 の文字が存在すれば十分なのであるが、大量の文字定義がなくても少なくとも字形が確認できるように、UCS の文字や ISO 2022 の図形文字集合の文字、文字鏡の文字などを組込み文字としている。

表 8: 組込み文字

U+000000 〜 U+00FFFF	UCS 基本多言語面 (BMP)	約 6 万字
U+010000 〜 U+10FFFF	0 群 1 面以降の Unicode	（約 100 万字）
U+E00000 〜 U+E0C3B4	大漢和辞典	約 5 万字
U+E0C3B5 〜 U+E816EF	文字鏡 (50101 〜 530159)	約 58 万字
U+E90940 〜 U+E9269F	94 文字集合（1 周目）	約 7 千字
U+E926A0 〜 U+E9449F	96 文字集合（1 周目）	約 7 千字
U+E944A0 〜 U+E961FF	94 文字集合（2 周目）	（約 7 千字）
U+E96200 〜 U+E97FFF	96 文字集合（2 周目）	（約 7 千字）
U+E98000 〜 U+E99D5F	94 文字集合（3 周目）	（約 7 千字）
U+E99D60 〜 U+E9BB5F	96 文字集合（3 周目）	（約 7 千字）
U+E9BB60 〜 U+E9D8BF	94 文字集合（4 周目）	（約 7 千字）
U+E9D8C0 〜 U+E9F6BF	96 文字集合（4 周目）	（約 7 千字）
U+E9F6C0 〜 U+F4BFFF	94 × 94 文字集合（1 周目）	約 70 万字
U+F4C000 〜 U+FFFFFF	96 × 96 文字集合（1 周目）	約 72 万字

4 UTF-2000 用標準文字データベース

XEmacs UTF-2000 用に関数 define-char による文字定義の形式で表現される UTF-2000 用標準文字データベースを開発中である。これは Unicode Database や川幡氏による CNS 11643 と大漢和辞典の対照表、拙作の幾つかのデータベースなどのさまざまな文字データベースを統合し、互いの矛盾点を修正することによって改良作業を行っている。

この UTF-2000 用標準文字データベースでは非漢字に関しては概ね Unicode [6] の定義に則っているが、漢字に関してはかなり微小な字体差も区別している。漢字の各文字（字体）の内、大漢和辞典と同じ字体でないものは ‘morohashi-daikanwa’ という属性名に 3 要素以上の整数のリストで対応する大漢和番号と差異の度合および整理番号を振っている。また、*The Unicode Standard* [6] の例示字体と同じ字体でないものに対しては ‘->ucs’ という属性名の値に対応する Unicode の符号位置を入れている。同様に、文字鏡に同じ字体の文字が存在しない時に、対応する文字鏡番号を ‘->mojikyo’ という属性名の値に記している。

表 1, 2, 3, 4, 5, 6, 7 および表 9, 10 に属性名の一覧を挙げる。

表 9: 漢字関連の文字属性

->bopomofo-letter	（漢字に）対応する注韻字母（UCS の符号位置）
cns-radical	CNS 11643 での部首
cns-total-strokes	CNS 11643 での総画数
daikanwa-radical	大漢和辞典での部首
daikanwa-strokes	大漢和辞典での画数
ideographic-radical	漢字の部首
ideographic-strokes	漢字の画数
japanese-kun-yomi	漢字の訓読み
japanese-on-yomi	漢字の音読み
japanese-radical	（Unicode の）日本出典の部首
japanese-strokes	（Unicode の）日本出典の画数
jisx0208-1978/4X	JIS X 0208 1978 年版 第 4 刷より前の規格票に用いられ、第 4 刷附属の正誤表で置き換えられた字形
jisx0208-difference	JIS X 0208 での差異の種類
kangxi-radical	康熙部首
kangxi-strokes	康熙画数
->mojikyo	対応する文字鏡番号
morohashi-daikanwa	対応する大漢和番号（ダッシュありを含む）
shinjigen-1	新字源（初版）番号
shinjigen-2	新字源（改定版）番号
total-strokes	漢字の総画数
unicode-radical	Unicode での部首
unicode-strokes	Unicode での画数

表 10: その他の文字属性

script	script を表すシンボルのリスト
->ucs	対応する UCS の符号位置
->decomposition	（結合文字の完成系に）対応する文字（UCS の符号位置）もしくは文字修飾記号の列
->ideograph	（注韻字母に）対応する漢字（UCS の符号位置）
->lowercase	対応する小文字（UCS の符号位置）
->titlecase	対応する語頭文字（UCS の符号位置）
->uppercase	対応する大文字（UCS の符号位置）
bidi-category	（Unicode での）横書き双方向表示での種類
comment	コメント
decimal-digit-value	10 進数での数価
digit-value	数価
general-category	（Unicode での）文字種
iso-10646-comment	ISO 10646:1993 でのコメント
mirrored	横書きでの書字方向による鏡像化の有無
name	文字名称
numeric-value	数価

現在、約 7 万字分の定義が存在するが、まだ誤りも多く、品質は高くない。

5 おわりに

文字データベースに基づく文字処理モデルである『UTF-2000 モデル』と、このモデルに基づく実装である **XEmacs UTF-2000** に関して述べた。

XEmacs UTF-2000 は文字データベースを取り換えることにより、容易に異なる文字表現や文字概念を利用可能であり、符号化文字モデルに基づく実装に比べて格段の柔軟性を持っているといえる。

XEmacs UTF-2000 は符号化文字モデルに基づく従来の XEmacs-Mule に比べればより多くの記憶量を要求するが、現在の豊富な記憶量を鑑みればこのことはそれほど問題ではないと考えられる。また、現在は文字定義を全て XEmacs UTF-2000 が管理する主記憶上に格納しているが、これを外部のデータベースに持ち、必要な時に必要な分だけ主記憶に取り込むようにすることで、必要とする記憶量を削減することが考えられる。

なお、XEmacs UTF-2000 は現在

http://www.kanji.zinbun.kyoto-u.ac.jp/ tomo/comp/emacsen/utf-2000/

で配布しており、自由に利用することができる。また、cvs.m17n.org で CVS による分散共同開発も行っており、ここから常に最新の source を入手することができる。

ところで、実際にさまざまな文字の扱いを利用するにはその基礎となる標準的な文字データベースの存在が不可欠である。現在、利用できる大規模文字データベースは Unicode 用のものが大部分であり、Unicode とは異なる文字統合・分離規則を用いるのは困難である。また、Unicode に含

まれない文字に関する情報を系統的に集めた自由に利用可能な文字データベースは現状ではほとんど存在しない。このため、このような標準的データベースを作成し、自由に利用できる形で配布することは重要であると考えられる。また、こうしたものを拓本や印刷文字などの一次資料に基づいて作成すればその利用価値ははかり知れないといえる。そこで、このような字形の出典に関する情報も含めた文字データベースを作成することは、符号化文字モデルの問題点の克服するために大変重要なことだと考えられる。

参考文献

[1] American National Standard Institute. *Coded Character Sets – 7-bit American National Standard Code for Information Interchange (7-bit ASCII)*, March 1986. ANSI X3.4.

[2] International Organization for Standardization (ISO). *Information technology – Character code structure and extension techniques*, 1994. ISO/IEC 2022:1994 （= JIS X 0202,「情報交換用符号の拡張法」）.

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

[4] Mikiko Nishikimi, Ken'ichi Handa, and Satoru Tomura. Mule: MULtilingual Enhancement to GNU Emacs. In *Proc. INET '93*, pages GAB–1–GAB–9, 1993.

[5] Richard M. Stallman et al. GNU Emacs version 20.7. ftp://ftp.gnu.org/gnu/emacs-20.7.tar.gz, June 2000.

[6] The Unicode Consortium. *The Unicode Standard, Version 3.0*, February 2000.

[7] XEmacs. ftp://ftp.xemacs.org/pub/xemacs/.