文字知識に基づく文書処理環境の現状と未来

守岡 知彦

概 要

特定の符号化文字集合に依存しない文字処理技術を開発している CHISE (CHaracter Information Service Environment) プロジェクトの現状と今後について説明する。このプロジェクトは、符号化文字技術の問題点を克服するために、文字に関する諸知識を機械可読な形で記述したデータベースに基づいた文書処理環境を実現することを目指しており、現在、文字属性データベースに基づく多言語文書編集系 XEmacs UTF-2000 と知識表現形式のひとつである Topic Maps 処理系を開発している。また、文字データベース・コンテンツの開発も進めており、ISO/IEC 10646-1,2 に収録された約 7 万字の漢字を対象に部品の組合せ構造をデータベース化した。ここではこうした現状と今後について紹介する。

1 はじめに

計算機における文字表現技術は計算機におけるさまざまなデータ表現の基盤であり、インターネットにおける多彩なコンテンツも文字表現技術に依っている。現在、計算機における文字表現はすべて符号化文字技術に基づいている。これは文字を対応する番号で表現する方法である。この方法では情報交換を行う当事者（即ち、送信者と受信者）が文字と番号の対応規則である符号化文字集合（文字符号）を共有している必要がある。さもなくば文字化けが生じる。また、符号化文字集合に含まれる文字は有限であり、そこに存在しない文字は表現できない。このため、従来、交換可能性をあきらめて外字を用いる方法や、検索などの符号化文字としての利便性をあきらめて画像を用いる方法が採られてきた。また、こうした問題を避けるために、文字符号の規格に多数の文字の追加を行う動きもあるが、これにも技術的・政治的問題がある。

このような問題を抜本的に解決し、日常的な文書のみならず歴史的な文書や将来生じる文書も表現可能で、かつ、各文書の永続性を保証するためには、符号化文字集合に依存しない次世代の多言語文書処理技術の確立が必要である。このためには文書表現におけるあらゆる要素を機械可読な方法で宣言的に記述可能にする必要がある。すでに、文字より大きな単位の要素に対しては、SGML/XML などにもとづくマークアップ手法が用いられているが、文字に対してもその性質を機械可読な方法で記述し、このように表現された文字知識に基づいて文字を処理することは有効な方法であると考えられる。

このような観点に立ち、1999 年から **XEmacs UTF-2000** [12] [1] と呼ぶ多言語文書編集系の開発を開始し、2001 年からは SGML に基づく漢字処理の先駆者の一人である Christian Wittern 氏を交え、文字知識データベースに基づく総合的な文書処理環境の実現を目指して CHISE プロジェクトを開始した。

2 XEmacs UTF-2000

XEmacs UTF-2000（図 1）は 2.1 節で述べる『UTF-2000 方式』の実証を目的に著者が中心となって開発している XEmacs [10] を基にした多言語文書編集系である。

XEmacs UTF-2000 の開発の目的のひとつは Mule [8] 以上に自由に文字が定義・操作できる環境の実現である。XEmacs を含めた GNU Emacs 系編集系の世界において、Mule は高度な多言語機能の基礎を提供したが、現在の Mule は質的・量的な側面で幾つかの問題を抱えていると考えられる。

そのひとつが文字の扱いである。現在の Mule 実装[1]では文字は符号化文字集合と符号位置の対で表さ

[1]現在、内部表現を Unicode 化した GNU Emacs が開発中である。

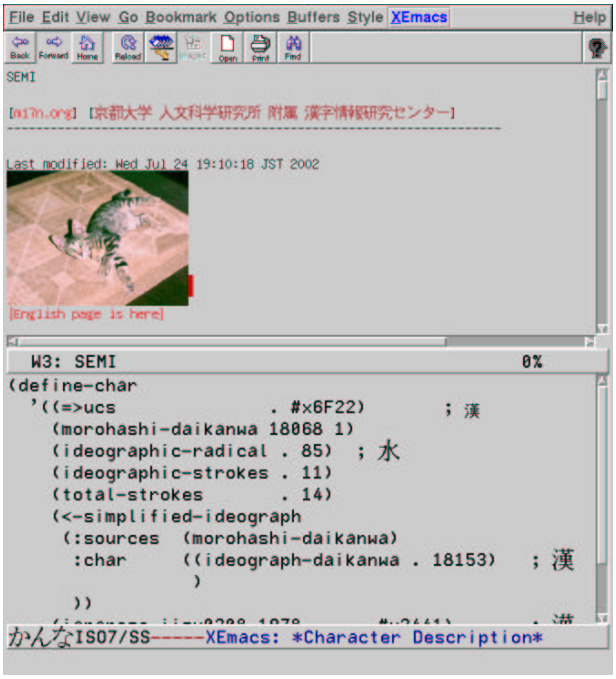

図 1: XEmacs UTF-2000

れる。Mule 実装で利用される符号化文字集合には charset-id という番号を割り振られ、この charset-id と符号位置を加工して文字用の固定長表現と文字列・バッファ用の可変長表現が生成される。これらの表現は、Mule 実装で扱う符号化文字集合が 2 byte 以内の ISO 2022 の図形文字集合の構造を満たすことを前提としているため、Big5 や UCS といった非 ISO 2022 系符号を扱おうとする場合この制約を満たす 1 つまたは複数の内部表現用符号化文字集合に変換（分割）する必要がある。また、文字表現がスパースな構造をしているために、19 bit で 1 文字を表現したとしても実際に使える文字数は 2^{19} = 約 52 万字を大きく下回る。実際、現在、既にコード空間は枯渇しており、事実上、拡張可能性が無くなっている。

こうした量的な問題は別にしても、文字の意味が符号化文字集合の定義に委ねられているが故の質的な問題がある。例えば、同じ「あ」であっても、charset-id が異なれば違う『文字』と見做されてしまう。例えば、JIS X 0208:1978 の「あ」と JIS X 0208:1983 の「あ」と GB 2312 の「あ」は別の『文字』として扱われる。[2]

[2]この文字の意味を符号化文字集合の定義に委ねているが故の質的な問題は Mule 固有の問題というよりも符号化文字方式一般の問題であると考えられる。この問題を避けるためには、単一の符号化文字集合を使うのが簡単であり、現状では Unicode 化するのがひとつの現実的な解であるといえよう。しかしながら、現実には Unicode 自体が既存の各種符号の寄せ集めとなっている現状を考えれば、これもまた近似解とならざるを得ない。

文字の個別な定義や文字間の関係に関する情報を扱うための一般的な機構は存在せず、必要に応じて連想リストや char-table などを利用して個別に文字をキーにした表を作ることになる。こうした機構は大規模な文字データベースを作ることを想定しておらず、文字に関する知識を利用したプログラムを書くことは不可能ではないとしてもあまり容易なことではないといえる。

こうしたことを考えると、Mule 以上に自由に文字を定義可能でより高度な文字処理を行うためには、文字を符号化文字集合によって管理し文字の意味を符号化文字集合の定義に委ねる Mule 方式をやめ、1 文字単位に文字を定義・追加可能でかつ文字の性質に関する情報を扱うための機構を追加することが重要であると考えられる。これはいわば文字の知識に関するデータベース機構を基礎に文字処理系を実現することであるといえる。

また、この観点から考えれば符号化文字集合もまた一種の文字データベースであると捉えることができる。そして、符号化文字集合自体をデータとして処理可能であるような文字データベース機構を考えた時、文字を表現するために特定の符号化文字集合を前提にしなくても良いと考えられる。そこで、符号化文字方式に基づかない文字表現モデルとそれに基づく実装を開発することにした。

2.1 文字表現モデル

符号化文字方式に基づかずに文字を表現すること、すなわち、固有の番号で同定することなしに文字を指し示そうとするとしたら、どうすれば良いだろうか？おそらくその一つの方法は指し示したい文字の性質を列挙することだろう。例えば、「あ」を指し示すとするならば

用字系 ひらがな

音価 /a/

のようにすればよいだろう。[3] 漢字の場合には、発音だけでは不十分である。例えば、「字」を指し示したい時に

[3]変体がなを考えた場合に、変体がなでない「あ」に限定したい場合にはこれでは不十分だが…

用字系 漢字

音 /じ/

とすれば、「字」以外にも「時」や「次」など /じ/ という音を持つ全ての漢字が含まれてしまう。総画数を付け

用字系 漢字

音 /じ/

総画数 6

とすれば、「時」や「事」などは除外され、「字」や「次」や「耳」などの総画数が 6 画の /じ/ という音を持つ全ての漢字の集合に限定される。さらに部首「子」を指定すればさらに限定されていくだろうし、文字の構造「『宀』の下に『子』」を指定したり、字義を指定すればさらに限定されるだろう。

このように文字を属性の集合で表現することによって、符号化文字方式に依らずに文字（ないしは文字の集合）を表現することが可能である。また、指定する属性を多くしたり少なくしたりすることによって、表現する文字の包摂規準を細かくしたり粗くしたりすることができる。

このような属性に基づく文字表現においては、文字の性質に関する情報は文字属性として文字表現自体に含まれている。よって、特定の処理に必要な情報は文字属性として文字表現に含めておかなければならない。例えば、表示を行なうには文字の字形を示す情報が必要である。逆に、表示という処理を行なわないことを前提とした文字表現には、字形という属性は不要となる。

このように、操作対象となる文字の性質を文字属性の集合で表現した『文字オブジェクト』として抽象化し、こうした文字属性の集合を格納したデータベースを参照しながら文字を処理する方式を『UTF-2000 方式』と呼ぶことにする。内部処理において文字符号のようなものが必要な場合、そのシステムの内部でのみ通用する ID（文字 ID）を振りそれを使う。システムの外部とは１つまたは複数の文字属性で文字（もしくは文字の集合）に関する情報をやりとりする。文字を表現するためのものは全て文字属性として扱い、各種文字符号での符号位置や辞書での文字番号のようなものも文字属性のひとつとして扱う。

UTF-2000 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。

2.2 実装

著者らは、UTF-2000 方式の実証を目的として、XEmacs UTF-2000 を開発している。これは文書編集系 XEmacs を元に、文字・文字列・バッファの内部表現を変更し文字属性を管理するデータベース機構を付けるなどして UTF-2000 方式に基づく文字処理を実現したものである。

XEmacs UTF-2000 は UTF-2000 方式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。このため、文字データベースを利用しやすいように文字の内部表現を変更している。文字表現空間を 30 bit に拡大したため、同時に利用できる文字数は大幅に増えている。文字は、**文字** *id* と呼ばれる文字オブジェクトの id で内部的に管理され、この文字 id を用いて文字オブジェクトを参照して、処理が行なわれる。

文字オブジェクトは文字属性の集合として定義され、固有の「文字 id」が割り当てられる。文字を定義するために XEmacs UTF-2000 では `define-char` という組込み関数を用意している。この他、文字属性の参照・設定関数、文字属性に対する map 関数、探索関数など文字符号を隠蔽して文字を処理するための関数群を追加しており、文字符号に依存しないプログラムが書けるようになっている。また、従来の XEmacs に対する上位互換性を持っており、XEmacs 用に書かれた多くのプログラムがそのまま動作する。

2.3 外部データベース化

UTF-2000 実装では処理対象とするすべての文字の知識を文字属性として保持しなければならないために、符号化文字モデルに基づく従来型の文字処理系に比べて多くの記憶資源を必要とする。

XEmacs UTF-2000 では文字属性データベースは define-char 形式の Emacs Lisp プログラムとして表現され、文字属性データベース全体を読み込んだ状態の記憶イメージをダンプした実行形式を作り、そのダンプされた実行形式を用いる。このため、XEmacs UTF-2000 のダンプ後の実行形式の大きさと元となった XEmacs-Mule の実行形式の大きさの差は文字属性データベースを保持するための記憶資源の大きさを意味している。

初期の XEmacs UTF-2000 は文字属性データベースの保持するための機構の効率化が十分でなかったこともあり、i386 アーキテクチャ上の Linux において当時の XEmacs-Mule の実行形式が約 10 MB であったのに対し、約 5 万字分の文字データを保持した状態で実行形式が 40 MB を越えるようになった。その後、文字データを保持する機構を改良し、記憶効率を向上したため、最近の XEmacs UTF-2000 では約 10 万字分の文字データを保持した状態で実行形式は約 32 MB となっている。

このように、主記憶上に文字属性データベースを保持する方法は多くの記憶資源を要するという点で問題がある。そして、通常の利用で必要となる文字は数百から数千であり、必要とする文字属性も限られているということを考えると、文字属性データベース全体をダンプするという方法は望ましくないと考えられる。

文字属性データベース全体をダンプする方法のもう一つの問題点は、UTF-2000 実装と文字属性データベースが不可分になってしまうことである。つまり、異なる UTF-2000 実装間で文字属性データベースが共有できないことを意味する。また、UTF-2000 実装の開発と文字属性データベースのメンテナンスは非常に性質の異なる作業であるにも関わらず、両者を統合した形でソースファイルを管理しなければならない。

このような問題点を解決するために、XEmacs UTF-2000 において外部の文字データベースを利用するための機構を開発した。これは外部の文字データベースから文字属性を必要な時に情報を獲得する (lazy-loading) ための枠組と、外部文字データベースの種類毎の実装からなる。現在のところ XEmacs の **database** 機能 (Berkeley DB のような属性値を保持するための単純なデータベースに対する抽象) を利用した外部文字データベースだけが実装されており、Debian GNU/Linux (woody) における Berkeley DB Version 3 でその動作確認を確認した。

この実装 (以下、『lazy-loading 版』と呼ぶ) と文字定義をダンプする方式の実装 (以下、『dump 版』と呼ぶ) の実行形式の大きさを TM 5800 上の Debian GNU/Linux (sid) において比較すると、約 10 万字のの文字定義を持つ XEmacs 21.2.46 UTF-2000 の dump 版の実行形式の大きさが 32 MB (strip 後 27 MB) であるのに対して、lazy-loading 版の実行形式の大きさは 16 MB (strip 後 11 MB) となった。ちなみに、XEmacs 21.2.46 (mule 付き) の実行形式の大きさは 10 MB (strip 後 6 MB) である。

lazy-loading 版の実行形式の大きさがなお XEmacs-Mule よりも 5 MB 程大きいのは、XEmacs-Mule から引き継いだ Emacs Lisp code において、coded-charset を鍵とした char-table が多用されているせいだと考えられる。XEmacs UTF-2000 では char-table は char-id-table で実装されており、coded-charset を鍵にして値を設定した場合、その coded-charset に属するすべての文字に対して値を設定するようになっているため、必要な記憶量が膨らむと考えられる。また、char-table は文字属性と異なり外部データベースに対応付けられていないため、lazy-loading ができないのである。よって、この点を改良すれば lazy-loading 版 XEmacs UTF-2000 の実行形式の大きさを XEmacs-Mule と同程度にすることができると考えられる。

3 文字属性データベース

文字データベースに基づく文字処理システムを活用するためには文字知識のデータベース化が不可欠であり、我々はこうした文字データベース・コンテンツの開発にも力を入れている。2001 年度から漢字の部品の組合せ構造を表現した漢字構造情報データベースの開発をはじめ、現在、ISO/IEC 10646-1:2000 [6] 基本統合漢字 (Unicode の例示字形), 同 統合漢字拡張 A, および ISO/IEC 10646-2 [7] 統合漢字拡張 B に対する入力を一応完了している。

現在のデータベースは

- Unicode Database
- CNS 11643 と諸橋大漢和辞典の対照表
- CDP データベース

- CBETA 外字データベース
- CHINA3 外字データベース
- 開発者がこれまで作成してきたその他の雑多なデータベース

等を統合し、互いの矛盾点を修正するものである。まだ誤りも多く、品質は高くはないが、現時点で約 10 万字分の定義が存在する。

この標準文字データベースでは、非漢字に関してはおおむね Unicode [9] の定義に則っている一方、漢字に関しては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については、`morohashi-daikanwa` という属性の値として、大漢和番号と差異の度合および整理番号を持たせている。また、*The Unicode Standard* [9] の例示字体と同じ字体でないものに対しては、対応する Unicode の符号位置を `=>ucs` という属性の値とする。

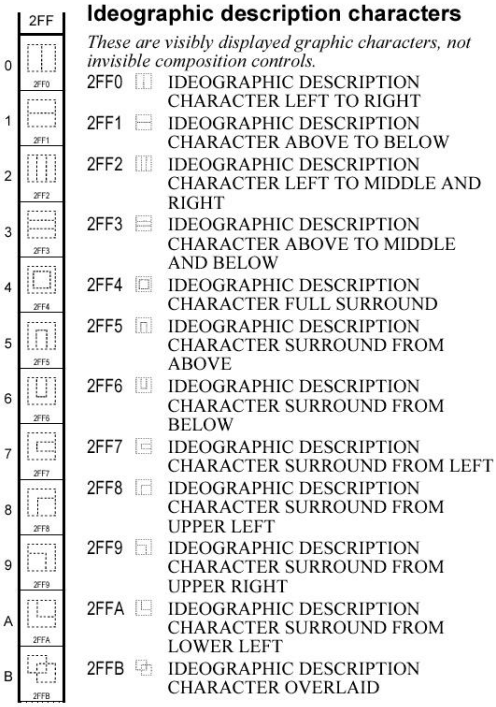

図 2: Ideographic Description Characters

3.1 漢字構造情報データベース

多くの漢字は偏と旁などの部品の組み合わせによって構成されている。こうした漢字の部品の組合せ構造は形の抽象的表現となるだけでなく、字義や音価にも関係しており、字源に基づく文字構造の分析は「解字」と呼ばれ、そうしたデータは重要な辞書記述の１つである。そこで我々は、UTF-2000 基本データベースに収録されている全ての複合（会意・形声）漢字に対し漢字構造情報を付けることを目標に、漢字構造情報データベースの開発をはじめた。

漢字構造情報に基づく符号化手法の試みは 1970 年代に遡るが、漢字符号化の主流とはならず、標準的な記法も確立されて来なかった。その後、ISO/IEC 10646-1:2000 [6] においてはじめて漢字構造情報の標準記法である IDS (Ideographic Description Sequence) とそのためのオペレーター群である IDC (Ideographic Description Characters)（図 2）が定義された。そこで、我々はこの IDS に基づく方法と、これを S 式（Lisp 表現）化し付加情報を許した *ideographic-structure* 形式、および、これを Topic Maps 化したものを用いることにした。

既存の漢字構造情報を含んだデータベースとしては、台湾中央研究院の CDP データベースと台湾の中華電子佛典協會 (CBETA) の外字データベースがある。これらはそれぞれ独自の形式を採っている。なお、これらは GPL で配布されるプログラムで利用可能であるので、可能な限り変換して利用することにした。また、この他、日本でも「今昔文字鏡」があり、検字を目的としたは８万字以上の解字情報を持つが、今の所、自由ソフトウェアで利用することはできない。またこの他、和田研フォントや GT 書体など、フォント合成を目的にしたいくつかの試みが存在するようである。

3.1.1 CDP データベース

CDP (Chinese Document Processing) データベースは、1990 年から台湾中央研究院の謝清俊らが開発している文字データベースで、現在、漢語大辞典に収録されている文字を中心に 55500 字以上を含んでいる。

CDP データベースは Big5 と外字を利用して符号化されており、外字を利用して 14 種類のオペレーターを定義している。そのうち３種類は部品の結合を表現したもので、(1) 縦に並べる、(2) 横に並べる、(3) その他 を表現する。また 8 種類の反復記号がある。こ

れは同じ部品を複数個配置することを表現するものである。また、この他に特殊記号が用意されている。なお、CDP データベースの漢字構造表記では入れ子を認めていない。また、IDS に比べ、結合オペレーターの種類が少なく、結合オペレータ (3) から IDC への変換に曖昧性があるため、機械的に IDS へ変換することはできない。

3.1.2 CBETA 外字データベース

台湾の中華電子佛典協會 (CBETA) は仏典データベースの作成上必要となった外字を対象に文字データベースを作成しており、この外字データベースには現在 13000 字程が収録されている。

CBETA 外字データベースも Big5 と外字を利用して符号化されているが、オペレーターは ASCII 文字を利用する。中置記法を使っており、'(' と ')' を用いることで入れ子表現も可能である。結合オペレーターとしては CDP データベースのものと同様に、縦に並べることを表現する '/' と、横に並べることを表現する '*' と、その他を表現する '@' の 3 種類である。この他に、部品の削除と置換を表現するためのオペレーターが存在する。部品の削除は $A-B$ という式で表現され、これは文字 A から部品 B を削除したものを表現している。例えば、草冠は 草 – 早 で表現できる。部品の置換は $A-B+C$ という式で表現され、これは文字 A 中の部品 B を C に置き換えることを表している。例えば、「花」は 草 – 早 + 化 で表現できる。この削除と置換を用いることで、3 種の結合オペレーターだけでは十分に表現できないような漢字でも、多くの場合において曖昧無く表現可能である。次により複雑な例として $((((瞭-目)-小)-日+(工/十))*支)/皿$ を示す。CBETA 外字表現は少数の規則で驚く程多くの漢字をカバーできる半面、複数の式表現が生じやすいといえる。なお、なるべく '@' を使わずに表現された式表現は、結合漢字の IDS 形式のデータベースが存在すれば、機械的に IDS へ変換することが可能である。

3.1.3 CHISE 漢字構造情報データベース

我々は漢字構造情報表現として、IDS に基づく入れ子上の木構造形式を採用し、プレーン・テキストでは IDS 形式（図 3）、XEmacs UTF-2000 の文字データベースでは *ideographic-structure* 形式に基づく `ideographic-structure` 属性、XML では Topic Maps に基づく形式で扱うことにした。

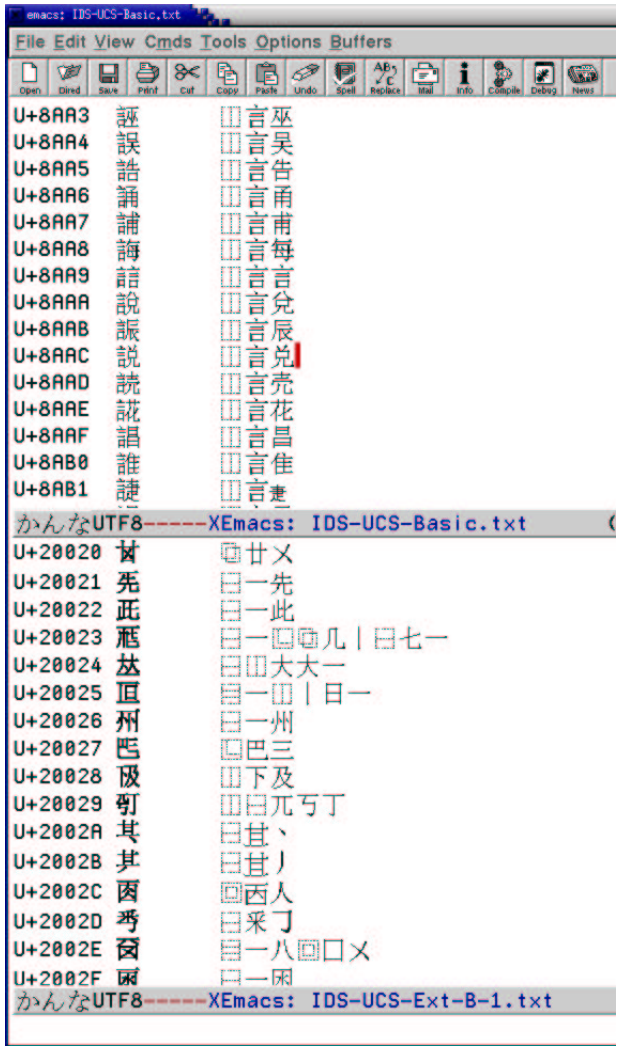

図 3: 漢字構造情報データベース

そして、CDP 形式や CBETA 形式からの変換プログラムを作成し、機械的に変換可能な部分に関しては利用可能な既存のデータを用いることにした。しかしながら、前述のように CDP データベースは機械的に変換することができず、また、形式に則っていないと思われる部分もあり、我々の目的にとっては十分なものではなかった。一方、CBETA 外字データベースは仏典で用いられる特殊な文字が主であり、データソースの点で難点がある。また、IDS に変換するためには結合

漢字を IDS で表現したデータベースが必要となる。そこで、CDP データベースから変換したデータを修正・補完することで、Unicode の基本統合漢字、ISO/IEC 10646-1 の統合漢字拡張 A, ISO/IEC 10646-2 [7] 統合漢字拡張 B, その他のレパートリの順に漢字構造情報データベースを開発することにした。

現在の所、基本統合漢字、拡張 A および拡張 B に関する入力作業はほぼ終了しており、現在、校正作業を行っている。また、この入力作業のために Christian Wittern 氏は CDP 外字や ISO/IEC 10646-1,2 の漢字を含む 7 万字以上の漢字を対象とした quail（GNU Emacs/XEmacs 用の標準的な入力システムの 1 つ）に基づく四角號碼方式の入力システムを開発した。

4 複雑な文字知識の表現

2.1 節で述べた UTF-2000 方式は、特定の符号化文字集合に依らず文字（や文字の集合）を表現することができ、XEmacs UTF-2000 が実証しているように対話型システムのような高速性を要求する場合でも現実的に適用可能な手法であるということがいえる。しかしながら、文字知識の表現方法として考えると、幾つかの点で不十分な点があるといえる。

そのひとつは、幾つかの組込み属性を除き、文字属性の意味が定義されておらず、その解釈はプログラムや利用者の意図に委ねられているということである。このため、ある文字属性を異なるプログラムや利用者が異なる意味を期待して利用する危険性がある。

この問題に関連する問題であるが、文字や文字属性間の関係を記述する一般的方法がないことが挙げられる。もちろん、ある文字属性の値を文字や文字の集合として解釈したり、複数の文字属性を関連するものとして扱うことにより、こうした関係を記述することは可能であるが、前述のようにこれを保証する機構やガイドラインがなければこうした意図を複数の利用者やプログラムの間で共有することはできないといえる。

また、これは UTF-2000 方式の問題とはいえないが、情報交換をどのように行うかという問題がある。現在の状況を鑑みれば、情報交換用形式として XML に基づく文字知識表現形式があった方が良いといえる。

このような文字データベースの表現上の課題を解決するために、現在、2 つの試みを行っている。そのひとつは、文字属性記述に関するガイドラインの作成である。これは著者が中心になって行っているもので、文字属性の命名規則や文字間の関係記述のための形式の定義などからなる。また、もうひとつの試みは XML TopicMaps 形式に基づいて文字や属性間の関係を記述しようというもので Christian Wittern 氏が中心となって行っている。

4.1 文字属性記述に関するガイドライン

現在、XEmacs UTF-2000 附属の文字属性データベースを対象に、文字属性記述に関するガイドラインを作成している。これはまだ完全なものとはいえないが現状について概説する。

4.1.1 文字から番号への写像

文字の集合から番号の集合への写像は $\Rightarrow map$ のような `=>` からはじまる名前で表現する。

例 文字「漢」の UCS での符号位置は、属性 `=>ucs` で表現され、その値は `#x6F22` である。

4.1.2 符号化文字集合

現在の XEmacs UTF-2000 では符号化文字集合 (coded-charset) の名前を属性名とする文字属性は coded-charset における符号位置を表す特別な文字属性として扱われる。しかしながら、ある文字属性が coded-charset の名前であるかどうかは属性名だけから判断することができないため、なんらかのガイドラインが必要であると考えられる。

符号化文字集合は文字から符号位置への写像と符号位置から文字への写像の組であると考えることができるが、前者は 4.1.1 節で述べた $\Rightarrow map$ 形式で表すことができる。そこで後者を表す形式を定義すれば良いと考えられる。そこで `<=` から始まる名前 $\Leftarrow map$ を番号から文字への写像を表すものとして用いることを検討している。また、$\Rightarrow map$ と $\Leftarrow map$ の両方を書くのが繁雑な場合、$= map$ によって両者の組を表すことも検討している。

4.1.3 文字指定形式

UTF-2000 実装の中では文字はオブジェクトの一種として符号化せずに扱うことができるが、UTF-2000 実装の外の世界との間ではなんらかの翻訳手法が必要となる。

このとき、もし符号化文字集合が利用可能でかつ表現したい文字を十分に表現可能であるなら、その符号化文字集合の符合位置を使ってその文字を表現することができる。しかしながら、表現したい文字が利用可能な符号化文字集合に収録されていない場合や、対応する文字が収録されていてもそこで規定された抽象文字と表現したい文字との差異が許容できない場合、符号化文字集合を用いることはできない。

そのような問題を解決するためには UTF-2000 方式に基づき文字オブジェクトの性質を列挙するような形式があれば良い。こうしたものとして XEmacs UTF-2000 では**文字指定 (character-specification; char-spec)** 形式を規定している。

文字指定の形式は Lisp の連想リスト (association-list) で、リストの各要素が各文字属性を表現する。連想リストの鍵 (key) 部（各要素の先頭 (car) 部）が属性名を表し、連想リストの値 (value) 部（各要素の残り (cdr) 部）が属性値を表す。

文字指定が表す意味はその性質を有する抽象文字（具象文字（書かれた文字）の集合）である。

なお、この形式は関数 define-char の引数で指定されるものと同じである。

例 「大漢和番号 18153 で表される漢字」という文字（もしくは文字の集合）は、文字指定表現

```
((ideograph-daikanwa .  18153))
```

で表される。

4.1.4 文字参照形式

文字データベースにおいて文字間の関係を記述するような場合、値に記載する文字の他に文字間の関係にも属性を付けたい場合がある。例えば、文字の正規化を行なう場合、アプリケーションによって正規化規則が異なるので、このための異体字データベースを作る場合、どの正規化規則を用いているかを記載する必要がある。また、学術的なデータベースを作る場合において、文字学上の学説が異なる場合に出典やどの学説を用いているかなどを記載する必要がある。この他、知的財産権の管理を行なう場合にもデータの出典や権利情報を記録する必要がある。

このような目的のために、XEmacs UTF-2000 では**文字参照 (character-reference; char-ref)** 形式を規定している。

文字参照の形式は Lisp の属性リスト (property-list) である。ここでは任意の属性が利用可能であるが、幾つかの属性名に対してはその意味が予め規定されている。

以下に意味が規定されている属性について説明する：

:char 参照される文字

型 文字、もしくは、文字指定

:source 出典・典拠等

型 出典・典拠を表すシンボル（出典シンボル）のリスト。

表 1 に XEmacs UTF-2000 の基本文字データベースで用いている出典シンボルを列挙する。我々は漢籍および現代中国文献に対する出典シンボルは国際的に用いられている中国語のピンイン表記を採用することにしたが、歴史的事情から日本語ローマ字表記のものも存在している。この表では、今後用いていく出典シンボル名を「名前」に記載し、歴史的事情から現在用いている日本語ローマ字表記の出典シンボルを「代替名称」に記載している。

例 大漢和辞典を出典とする「大漢和番号 18153 で表される漢字」に対する参照は、文字参照表現

```
(:sources (morohashi-daikanwa)
 :char     ((ideograph-daikanwa . 18153)))
```

で表される。

4.1.5 文字間の関係に関する属性

ある文字 c に対して関係 foo を持つ文字 γ_i が存在する時、文字 c の属性 $\rightarrow foo$ は値の各要素 γ_i が文字 c の foo であることを意味する。ここで、$\rightarrow foo$ の値 $(\gamma_1 ... \gamma_n)$ は文字か文字指定表現か文字参照表現のリストである。

表 1: 出典を表すシンボル

名前	代替名称	内容
	chuuka-daijiten	中華大字典
	doubun-tsuukou	同文通考
	gyokuhen	玉篇
	henkai	篇海
	henkai-ruihen	篇海類編
	inkai	韻会
	inkaiho	韻会補
	jii	字彙
	jiiho	字彙補
jiyun	shuuin	集韻
	kaihen	海篇
kangxi		康熙字典
	kouin	広韻
morohashi-daikanwa		大漢和辞典
	ruishuu-meigishou	類聚名義抄
	seiin	正韻
	seiji-tsuu	正字通
shuowen		説文
	setsumon-tsuukun-teisei	説文通訓定聲
	sougen-irai-zokujifu	宋元以来俗字譜
yuquan		玉泉
	senhen	川篇

同様に、文字 c の属性 $\leftarrow foo$ は文字 c が値の各要素 γ_j の foo であることを意味する。$\rightarrow foo$ と同様に、$\leftarrow foo$ の値 $(\gamma_1 ... \gamma_m)$ も文字か文字指定表現か文字参照表現のリストである。

例 小文字を表す関係を `lowercase` とする時、

1. 文字 `A` の属性 (`->lowercase ?a`) は、文字 a が文字 A の小文字であることを表している。
2. 文字 `a` の属性 (`<-lowercase ?A`) は、文字 a が文字 A の小文字であることを表している。

4.2 TopicMaps

Topic Maps は対象となる情報の構造を topic (『話題』) の集まりとして捉え、topic の定義や topic 間の関係などを記述することで、さまざまな構造を持った各種情報リソースを表現しようとするものである。

TopicMaps に関する標準としては、SGML [2] および HyTime [4] Architectural Forms に基づく形式の DTD (Document Type Definition) 表現による仕様が ISO/IEC 13250:2000 [5] として制定されている。また、独立したヴェンダーのグループによって XML 版も開発され 2000 年 12 月に XTM (XML Topic Maps) 1.0 として公開されている。これは 2001 年 12 月に ISO 標準の修正 (amendment) として承認されている。そこで、CHISE プロジェクトではこの XML 版を採用することにした。

XEmacs UTF-2000 という実装が既に存在している文字属性表現および define-char 形式の場合と異なり、TopicMaps に基づく文字知識表現形式は今のところ処理系が存在していない。そこで、2001 年度から Christian Wittern 氏は Topic Maps エンジンの開発をはじめ、Zope (Zope Object Publishing Environment) [11] を用いたプロトタイプを開発した。しかしながら、数万〜数十万の文字を対象に数十万〜数百万の topic を扱うには現状の Zope は不十分であり、また、XML の処理を行う上で必須といえる UTF-8 の処理にもバグがあり、このプロトタイプはあまり実用的なものとはいえない。このため、今後は PostgreSQL を用いた新たな Topic Maps エンジンや XEmacs UTF-2000 上で動作する Emacs Lisp で記述した Topic Maps 編集システムの開発を予定している。

このような事情から、現在のところ TopicMaps に基づく文字知識表現手法の開発はまだあまり進んでいない。とりあえず現在のところ

- 抽象文字
- 文字インスタンス
- 異体字形
- 文字構造
- 言語
- 読み
- 意味
- 時代
- 空間
- 良く使われる用例
- 符号化文字集合への写像
- 辞書への参照

などの文字属性軸が定義されている。

今後は文字属性記述に関するガイドラインと TopicMaps に基づく文字知識表現形式の双方の開発を進めるとともに、両者間の相互変換を実現し、文字属性と TopicMaps の両方の視点で文字知識を操作できるようにしたいと考えている。

5 今後の展開

文字データベースを XEmacs UTF-2000 の外部に保持する目的のひとつは、2.3 節で述べたように必要な文字の必要な情報だけを保持するようにして記憶資源の効率化を計ることであるが、文字に関する知識をさまざまなアプリケーションから利用できるようにする上でも非常に大きな意味がある。

この観点からいえば、現行の Berkeley DB に基づく文字属性毎の単純なデータベース機構は管理のしやすさやスケーラビリティーの点で問題があると考えられ、現在、CHISE Project では PostgreSQL の採用を検討している。

また、PostgreSQL に基づく文字データベース・サーバーのクライアントとして XEmacs UTF-2000 の他に、Ω に基づく多言語組版系、グリフ・字形情報の管理・合成系、漢字間の関係の視覚化システムの実現を目指している。これらの連携により図 4 に示すような文書処理環境を構成すれば、既存の符号化文字集合に依存せずに文字やテキストを一貫して編集・処理可能な環境が実現できると考えられる。これが CHISE プロジェクトの当面の目標であり、近い将来（2003 年中）における実現を目指している。

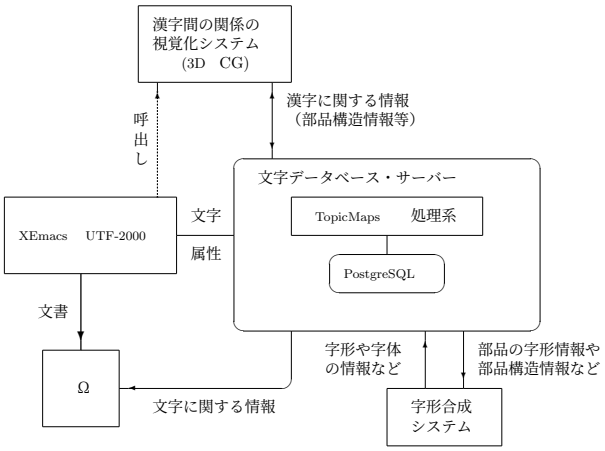

図 4: CHISE System 2003

6 他のシステムとの比較

6.1 Mule

Mule (MULtilingual Enhancement to GNU Emacs) [8] は GNU Emacs の多言語化拡張であり、GNU Emacs という Emacs Lisp 言語による拡張可能性を有した対話型編集環境を多言語化したものである。Mule 機能は本来 GNU Emacs に対する差分として開発されていたが、GNU Emacs 20.1 において GNU Emacs 自体の機能として取り込まれている。XEmacs においても Mule 機能は取り込まれており、ビルド時に `./configure` のオプションとして `--with-mule` を指定することにより Mule 機能を持った XEmacs の実行形式を作成することができる。

Mule の文字表現の概要と問題点は 6.2 節で述べた。GNU Emacs および Mule の開発者達は現在内部表現を Unicode 化する作業を行っており、この新しい実装 (Emacs-Unicode) ではここで述べた Mule の問題点は一応解消されている。しかしながら、XEmacs UTF-2000 が積極的に文字知識を扱おうとしているのに対し、Emacs-Unicode は素直で『軽い』 Unicode 実装を作るという方向を向いているようである。Unicode に収録されていない文字の扱いにも配慮しているが、GNU Emacs における Lisp オブジェクトの表現上の問題から文字を 22 bit で表現する必要があり、XEmacs UTF-2000 程大量に文字を定義するための領域を有していない。また、現在の実装では効率上の理由から 1 文字単位に文字を追加することはできないようである。

Mule は GNU Emacs および XEmacs の文字表現の拡張を行っているが、これ以外にも多言語処理を行うためのさまざまな拡張を行っている。`coding-system` と称するさまざまな文字符号と内部表現を変換する機能により、現在利用されているさまざまな文字符号が利用可能である。また、GNU Emacs の持つ各種モードを併用することにより、さまざまな文書形式の編集を支援する。言い替えれば、Mule は特定の文字符号や文書形式を前提にしたものではないといえる。

Mule は「多言語化」を標榜するものの、実際にはさまざまな言語を組織的に扱うための枠組は十分に整備されているとはいいがたく、概ね各種文字符号の優先順位や入力法などを切替える『言語環境』と称する機能が用意されているに留まる。言語と用字系 (script), 書記系 (writing system), 文字符号等の区別が希薄であり、これらが混然一体となって『言語環境』として扱われる。また、国際化機能も弱い。

XEmacs UTF-2000 は XEmacs-Mule から派生しており、文字表現を除けば、こうした Mule の性質は利点・欠点とも継承している。CHISE プロジェクトではこうした Mule の問題点を解決するために、文字に関する知識のデータベース化の次の段階として、各種言語や用字系、書記系の性質のデータベース化を進めたいと考えている。

6.2 超漢字

「超漢字」は BTRON 仕様に基づく OS である。BTRON は「実身・仮身」モデルに基づくハイパーテキスト支援機能をシステムの根幹に備えており、複雑な情報を操作するための枠組として優れた点を持っているといえる。一方、文字表現は ISO 2022 [3] を単純化したような（Mule の内部表現をステイト・フルにしたような）ものであるといえ、コード空間の問題を除けば 節で述べた Mule 方式の問題点が当てはまるといえる。超漢字ではこうした問題点の克服のため、各種文字データベースの開発が行われているようである。こうしたデータベースは検字や検索を目的としたものであり、文字表現とは不可分ではない。文字は構造を持たないものとしてとらえられ、内部表現も隠蔽されない。CHISE ではいわば全ての文字がオントロジーを持った外字のように扱われるのに対し、BTRON は外字が存在しないシステムであるといえる。

CHISE システムや Mule が特定の文字符号や文書表現を前提としない『開いた』システムを指向しているのに対し、「超漢字」は独自の文字符号体系やデータ表現を前提とした『閉じた』システムを指向していると考えられる。CHISE システムにとって、文字データベースや XML といった開放系を指向する技術に基づいて『閉じた』システムの持つ使いやすさや一貫性を実現するという点において BTRON はひとつの目標といえるかも知れない。一方、文字表現やデータ表現の点では超漢字はやや古いシステムということができる。利用できる文字の数においても、トンパ文字や i モード絵文字などを除いて、XEmacs UTF-2000 の部分集合になっている。

7 おわりに

CHISE プロジェクトの現状と今後の展開に関して述べた。XEmacs UTF-2000 が実証しているように文字データベースに基づく文字処理技術は現在の計算機環境において十分な実用性を有していると考えられる。また、文字に関する知識を積極的に利用することで、さらなる可能性があると考えられる。

こうした可能性を現実のものとするためには、プログラム・データ双方のさらなる整備が必要であり、多言語化技術、国際化技術、データベース、分散化技術など各方面のハッカーや、言語学、文字学、文献学など各種言語や字書やテキストなどを扱う人文科学者など、多方面に渡る研究者の幅広い協力が不可欠である。

こうしたことを鑑み、CHISE プロジェクトはオープン・ソースで開発されており、その情報は

- http://cvs.m17n.org/chise/
- http://kanji.zinbun.kyoto-u.ac.jp/projects/chise/
- http://mousai.as.wakwak.ne.jp/projects/chise/

で公開されている。これらの WWW 頁群を含め、各種成果物は CVS で管理されており、最新の開発状況を知ることができる。日本語用と英語用の 2 つのメーリングリストも用意されており、参加方法は上述の WWW 頁で説明されている。CHISE プロジェクトに興味を持たれた方は、是非、お気軽に御参加願いたい。

謝辞

本論文で述べた CHISE プロジェクトの研究の一部は 2000 年度に旧通商産業省工業技術院電子技術総合研究所（現 独立行政法人 産業技術総合研究所）からの受託研究として行われ、2001 年度には情報処理振興事業協会の「未踏ソフトウェア創造事業」の助成を受けた。

また、忙しい中 CHISE プロジェクトに主要メンバーとして御参加頂いた Christian Wittern 氏、江渡浩一郎氏、上地宏一氏、鈴木泰博氏、苫米地等流氏、藤原義久氏、宮崎泉氏、師茂樹氏に感謝する。また、2001 年度に未踏ソフトウェア創造事業のプロジェクトマネージャーとして、その後も、プロジェクト遂行において貴重なご助言と多大な御助力を頂いた g 新部裕氏に感謝する。

また、横田裕思氏やしおざきかずひこ氏をはじめとする UTF-2000 mailing list の参加者に感謝する。また、XEmacs UTF-2000 の開発にあたって貴重なご助言と御助力を頂いた産業技術総合研究所の戸村哲氏、半田剣一氏、錦見美貴子氏、高橋直人氏に感謝する。

参考文献

[1] bit 別冊「インターネット時代の文字コード」, 第 9 章「文書編集系における文字コード」. 共立出版, 2001.

[2] International Organization for Standardization (ISO). *Information processing — Text and office systems — Standard Generalized Markup Language (SGML)*, 1986. ISO 8879:1986.

[3] 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」）.

[4] International Organization for Standardization (ISO). *Information processing — Text and office systems — Hypermedia/Time-based Structuring Language (HyTime)*, 1997. ISO 10744:1997.

[5] International Organization for Standardization (ISO). *Information technology — SGML Applications — Topic Maps*, January 2000. ISO/IEC 13250:2000.

[6] 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.

[7] 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.

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

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

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

[11] Zope. http://www.zope.org/.

[12] 守岡知彦. UTF-2000 — 汎用文字符号に依存しない文字表現系の展望. アジア情報学のフロンティア — 全国文献・情報センター人文社会学学術セミナーシリーズ No.10, 全国文献・情報センター人文社会学学術セミナーシリーズ, 第 10 巻, pp. 13–24, 2000.