ヒューマンインタフェース 70－8
（1997. 1. 16）

全世界の文字・言語の混在テキストのための表示機能の国際化

上園 一知、大矢 俊夫、片岡 朋子†、片岡 裕†、小原 啓義

早稲田大学理工学部情報学科
† 早稲田大学メディアネットワークセンター
〒169 東京都新宿区大久保3-4-1
E-mail: {uezono, oya, tomoko, kataoka, ohara}@ohara.info.waseda.ac.jp

コンピュータネットワークは国際的な発展を遂げたにも関わらず、コンピュータで同時に扱える文字種は制限されているのが現状である。真の国際化計算機環境を実現するためには、既存の文字コードを超えて全ての文字・言語を同時に混在使用可能であることが必須条件となる。特に出力に関しては、表示される文字図形と文字・文字コードの写像関係が常に言語に依存すると見なされ、更に横書き・縦書きといった文字列の表記方向が混在するために、全世界の文字・言語を全て網羅することは非常に複雑で、一般に不可能と考えられてきた。しかし、世界中の文字・文字コード・正書法を分析した結果、少数の関数で図形決定が可能であり、更にその配置手法が明らかとなった。この関数をハードコーデッドせずに実装することで、全世界の文字・言語が混在したテキストを単一システムで表示可能となった。

I18N Drawing Functions for a Text Containing All Characters/Languages in the World

Kazutomo Uezono, Toshio Oya, Tomoko Kataoka†, Yutaka Kataoka† and Hiroyoshi Ohara

School of Science and Engineering, Waseda University
† Media Network Center, Waseda University
3-4-1 Okubo, Shinjuku-Ku, Tokyo 169 Japan
E-mail: {uezono, oya, tomoko, kataoka, ohara}@ohara.info.waseda.ac.jp

The international networking has already become a common equipment, but usage of characters/codesets is still restricted on computer systems. To realize the international computing environment, it is essential to manipulate any codeset/script/language simultaneously beyond the codesets which are currently defined. The Output for internationalized text was considered impossible by the orthography/language dependencies. But mapping among mb, WC and glyph could be determined by our previous researches. By the analysis of drawing all of scripts, essential information to locate characters with correct shapes were discovered by combinations of several rules. Then, without hard-coding, our system could realize not only one-line drawing but also multi-line drawing of any internationalized text of all writing directions mixtured.

1.はじめに

現在既に計算機ネットワークが世界的に広がり、日常的に計算機を用いた国際的通信が行なわれている。更に、言語学等の研究においてだけでなく、デジタルライブラリ等のデータベース、語学教育等においても特定の言語・文字だけを対象とするのではなく、様々な言語・文字を計算機上で同時に混在使用できることが要求されている。しかしながら、ほとんどの計算機は英語＋1言語の POSIX Locale Model [3] に従うために、同時使用可能な言語・文字は制限されており、情報交換の見地から言えば未だ計算機環境の国際化はなされていない[9]。これは特に、人間と計算機とのインターフェイスである出力機構の国際化の実現が困難であるために生じている。即ち、国際規格や国家規格として定められた文字コードから文字と表示すべき文字図形を確定し、更にこの図形列を二次元座標上に配置する方法が一般化されていないために、特定の言語に依存した形態でしか実現が不可能と考えられている [12, 13]。

印刷装置の制御のために、ISO6429 [2] では、文字図形の指定や図形列の二次元配置の指定のための制御機能が規定されている。しかしこの規格では、文字図形のうち Ligature の図形の指定が不完全であり、また座標軸が機器に依存し二次元配置の指定も曖昧である。従って、この規格だけに従った出力系では全ての文字を正しく混在して表示することは不可能である [11]。

本質的な解決のために、まず、全ての文字・言語・正書法を分析し、テキスト操作の対象である**文字集合**と表示の対象である**最終表示図形集合**を確定し、**文字コード集合**、文字集合、最終表示図形集合間の写像関数を導出した。また、最終表示図形集合の二次元配置手法の分析を行なった。その結果、集合間の写像関数は言語毎に依存せず、少数の関数の組合せで実現されることが明らかになり、図形列表示に必須となる規則を得ることができた。

以上の写像関数、図形列表示規則をライブラリとして準備し、文字・文字コード・言語・正書法に依存した情報をハードコーデッドせず、システム外部にデータとして持たせることで、全ての文字・言語が使用可能である国際化環境 (OS + ウィンドウシステム) **System 1** を作成した (図1) [14]。System 1は、全ての文字の混在表示だけでなく、テキスト処理、プロセス間通信も可能としており、これにより真の国際化アプリケーションの開発が可能となった。

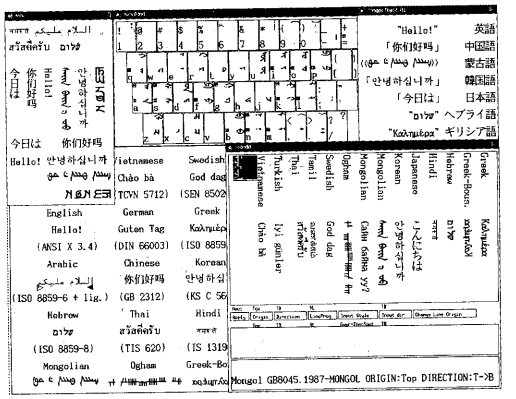

図1. System 1 表示例

2.文字の分析

文字は、情報交換における基本である。現在のテキスト処理システムでは、文字の扱い方法が特定の言語や文字種に制限され、様々な文字を任意に混在使用することができない。従って、全文字を1集合とし、且つ、全文字を対象として一般化した処理を求めなければならない。即ち、全文字を1集合として扱うための分析が必須である。

文字は一般にそれが指し示す対象によって、

1) 音素文字
2) 音節文字
3) 表意文字

に大別できる。しかし、この分類は、文字の構造上の特徴を示すには不十分である。

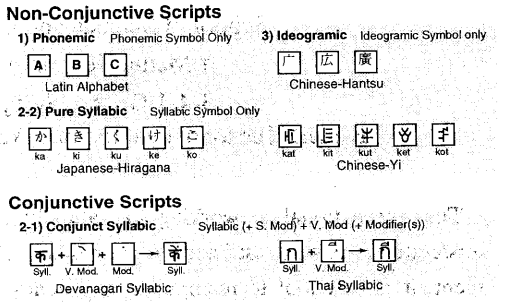

図2. 文字の分類

音節文字には、音韻上更に子音音価と母音音価に分離可能なことから、音節文字のうち、子音音価を主に示す記号と母音音価を主に示す記

号に分離することが可能な文字群がある。これを考慮して、上記の文字分類を更に、

1) 音素文字
2-1) 結合音節文字
2-2) 純音節文字
3) 表意文字

に細分化できる(図2)。

よって、文字は記号がそのまま文字となるものと、記号列をその構成規則により文字を構成するものとに分類可能である。ここで、前者を、構成規則が空である文字、と見なすことで、全ての文字は「記号列とその構成規則」から構成されていると定義することができる(図3)。言語は、その表記に使用する文字集合の要素数を規定し、各文字に意味・音価を与えているのであり、文字を構成する記号、その記号列の構成規則は規定しない。従って、上述の定義によれば、文字は言語に依存しない。よって文字集合は言語に依存せずに BNF で定義可能であり、全世界の文字を単一の集合として定義可能となる。

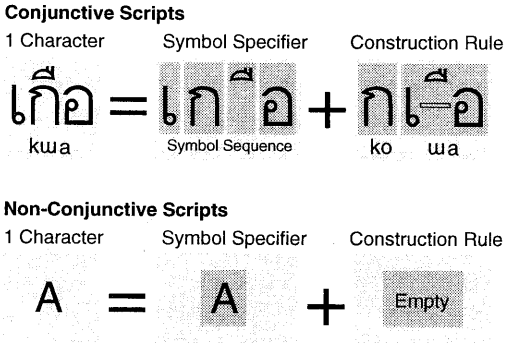

図3. 文字の構造

一般に1文字に1図形が対応していると考えられているが、必ずしも1図形が対応しているとは限らない。文字の表示上の図形は、次の条件によって、複数個ある。

1) 文字の表記方向
2) 文字列中の文字の位置
3) Ligature の形成

1) に関して、ウィンドウシステムやプリンタにおける出力は二次元座標系であり、文字の方向は4方向存在する。例えば句読点に見られるように、文字の方向によって図形の変化するものがあるため、理論上方向毎に4図形持ち得ることになる。2) に関しては、アラビア文字に見られるように、隣接した文字との連結情報を持つ場合、独立形、頭字形、中字形、尾字形の4図形が存在する。従って、1文字に対して、16図形持ち得ることになり (図4)、更に変字体を含めると16図形以上となる。これは即ち、文字は図形集合を表しており、文字名は図形集合のタグであると考えることができる。更に 3) における、複数の文字の組合せにより図形を指し示す場合も考慮すれば、ある特定の文字列も、図形集合を表すと考えることができる。

また、アラビア文字の例から、文字を正しく表記するためには、文字列としての表記が必須であることが自明である。

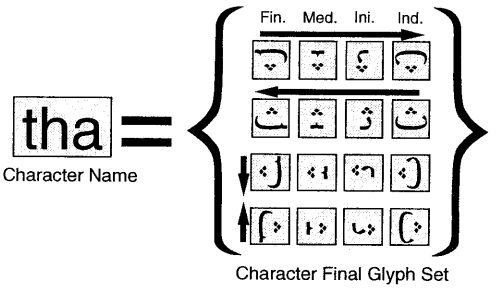

図4. 文字と図形の関係

但し、図5のモンゴル文字に見られるように、異なる文字でも同一の図形を持つことがあるので [10]、単独の図形のみからは、対応する文字を特定することはできない。従って、単純に図形のみから文字コードを作成することができない。

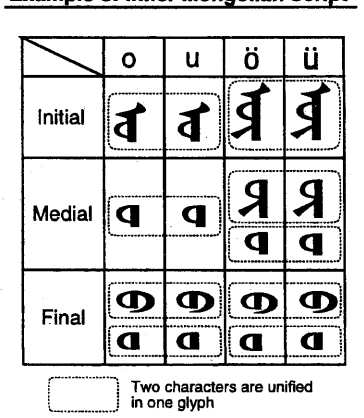

図5. 異なる文字による図形の共有例

3.文字コードと文字・図形との写像関係

文字を通信するために、国際規格・国家規格として、文字コードが規定されている。文字コードのデザインは、図6のように分類できる。

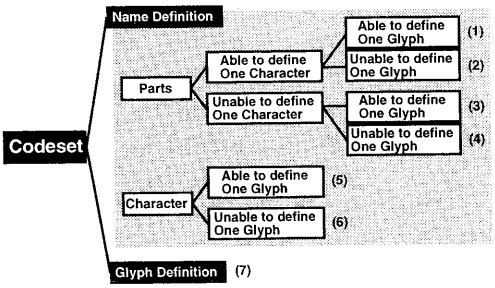

図6. 文字コードの分類

図形から文字は特定不可能なので、図形定義された文字コード (7) は、表示等図形を対象とした処理は可能であるが、文字の編集等、文字操作を行なうことはできない。例えば、GB 8045-87 [8] は文字の部品(不足した)の図形定義であり1文字が決定できず、文字操作と同時に図形を対象とした処理も満足に行なうことができない。従って、文字操作をするためには、文字コードは**名称定義**でなければならない。名称定義は更に、**記号定義**と**文字定義**に分類することができる。計算機で文字操作、図形表示可能な文字コードは、(1)又は(5)でなければならない。但し、全ての文字を包含する目的で規定された、ISO 10646 [5] は、名称定義と図形定義が混在し、かつCJK漢字集合に見られる図形の異なる文字の Unification が行なわれているため、文字操作だけでなく、図形表示さえも正しく行なえる保証はない。

ISO 2022 [1]により、1つの文字コード集合を単位とした拡張がなされる。更に ISO 6429 によりコードポイント列から図形の拡張がなされる。しかし、ISO 6429は古典的出力機器の制御を目的とした規格であり、ECMAに登録された文字コードから文字の表示を可能とするための補助機能を規定しているにすぎず、文字を決定することを考慮していない。この拡張機能は、コードポイント: 文字 = 1 : 1の場合のみを想定しているため、構造を持った文字群の文字コードが記号定義である場合、文字の決定はこの規格のみでは不可能であり、ISO を超えた拡張が必要となる。TIS 620-2533 [6] や IS 13194:1991 [7] は記号定義された文字コードであり、外部からその構成規則を与える必要がある。更に TIS 620-2533 では、完全に文字を決定可能とするためには、デリミッタを必要とする。

文字から図形への変換は、文字の方向・位置・変字体の情報の付加で決定可能である。文字の方向・位置は、文字列を構成した場合、必ず文字列が持つ情報であるが、変字体は外部からの指定が必要となる。以上を計算機上で扱うための集合、及びその関係を図7に示す。

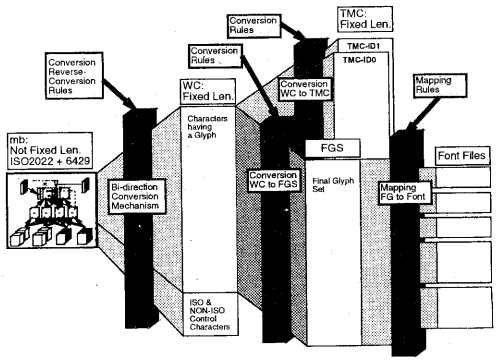

図7. mb・WC・FGS・TMC集合の写像関係

mb (Multi byte character) は、複数の **Octet** で1文字または1機能を表す不定長コードである。従って、mb をそのまま処理するのではなく、文字としての保証された集合(文字集合)へ変換されなければならない。そして、テキスト処理の対象であり、ISO 9899 [4] で規定されている固定長コードの WC (Wide character) が、文字集合でなければ矛盾を生じる。よって、WC (Wide character) は、文字・機能集合のコードと明確に定義される。更に WC は、文字の図形集合である FGS (Final glyph set) に変換され、それに必要な情報(**Attribute**)を持たなければならない。そして、最終的に、FGS は特定の複数のフォント集合の要素列に変換される。WC は言語名情報を持たないため、特定の言語に固有の文字処理をする場合、新たに言語名情報を持った集合を定義する必要がある。これは、TMC (Text manipulation code) として定義し、WC より言語名及び言語固有情報を付加して変換される。

4.文字列の表示手法の分析

古典的表示機構は、単一方向の図形配置のみを考慮している。従って、上下左右の詰めの混在する文字の配置に関し、考慮されていない。

図形列は二次元座標上に配置される。しかし、文字コードに表記方向及び配置情報が完全に含まれるわけではない。この配置のための情報として、

1) 配置開始位置

2) 正書法に定められた図形の配置方向

が必須である。配置開始位置を図形列中で変更した時、その前後の図形列の連続が失われオーバー・ライトされ、改行する必要が生じる。一行とは配置開始位置が同一である図形列、と定義することができる。改行は、定められた領域に図形列を配置するために行うので、配置領域を与えなければ自動的に改行することはできない。従って、表示は改行を含まない一行表示と、改行を含む複数行表示に分離される。

5.一行の表示

一行の表示では、一つの配置開始位置から図形を正書法に従って配置する。複数の文字種を同時に一行で表示する場合、図形の配置方向は左右上下の4方向存在するので、配置方向の混在が生じる。この混在方法として、以下が挙げられる。

1) 物理的表記方向を統一した図形列の配置

2) 配置方向を完全に遵守した図形列の配置

3) ISO 6429 の制御機能を使用して配置方向を強制指定した図形列の配置

1) 物理的表記方向を統一した図形列の配置

例として、ラテンアルファベットとアラビア文字の混在配置を考える。同一配置方向を持つ連続した文字列の単位をブロックと定義すると、図8に示すように、ラテンアルファベット列で1ブロック、アラビア文字列で1ブロックを構成する。このブロックをその行の詰め位置から一次元に配置することで、配置方向の混在が可能となる。

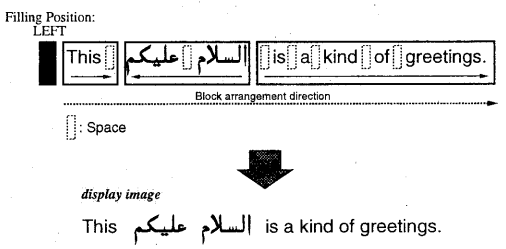

図8. 文字の一次元混在配置

従って、一行の一次元表示では、

a) ブロックの配置開始位置

b) 配置開始位置に対するブロック配置方向

c) ブロック内の図形列の配置方向

が必要となる。

このように、全ての配置方向を統一した物理的表記方向で混在する場合、正書法で定められていない物理的表記方向(ラテンアルファベットなら縦表記)で図形列を配置する必要がある。この時ブロックの配置方向に対し、図形列の配置方向は2方向存在する。図形列は正書法により改行方向を持つので、複数行の表示を考慮し、デフォルトとしてブロック列の改行方向を一致させるように表示されなければならない(図9)。但し、ユーザによりインタラクティブな変更を許可することで、任意のブロックを任意の配置方向で表示可能としなければならない。

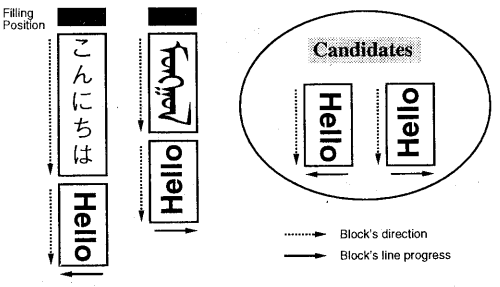

図9. 表示図形列の選択

2) 配置方向を完全に遵守した図形列の配置

正書法で定めた表記方向を遵守して物理表記方向を混在して表記する場合、行内のブロックの配置方向が途中で変更され、二次元配置となる。この配置方法は複数あるが(図10)、ブロック配置の観点からデフォルトとして(a)のような配置を採用することが望ましい。しかし、このような表示では、複数行での可読な配置の保証は不可能である。本システムでは、このような折れ曲がる基本表示関数と物理的な表記方向を揃える基本表示関数の2種類を用意する。

折れ曲がりの場合の配置の決定は、最終的にアプリケーション側のストラテジに依存する。これは、国際化で生じるデフォールトの多様性の例となり、既存の固定的な表記手法では満足しえない。即ち、一例をあげれば、左上が表記の原点で右方向へのみの表記のデフォールトを持つだけのウイジェットでは、国際化を満足することはできない。

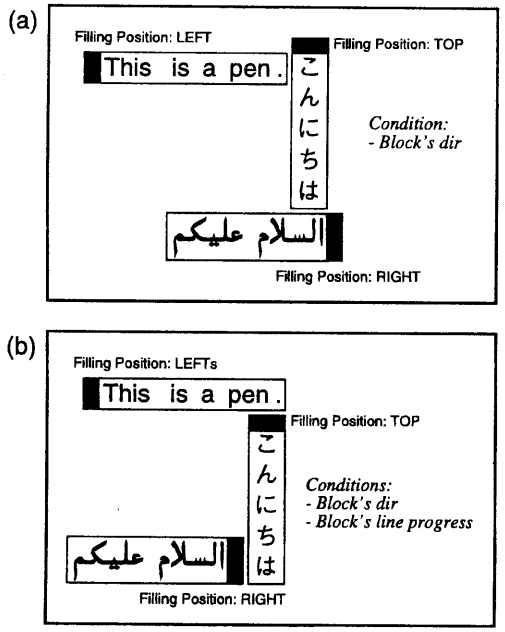

図10. 図形の二次元配置例

3) ISO 6429 の制御機能を使用して配置方向を強制指定した図形列の配置

ブロックの配置とブロック内の図形列の配置方向は、ISO 6429 に規定されている制御機能を用いて強制的に指定することができる。SPD (Select Presentation Directions) は、文字の表示方向と改行方向を変更することができるが、折れ曲がりを指定した時、改行は不可能となり、また折り返した時に重なって配置されることになるので、この制御機能を規格通りに使用することはできない。そこで、ブロック列の配置方向を変更する機能として使用する。しかし、折り返しは許さず、折れ曲がりの際は二次元配置となるため、配置方法は 2) 同様となる。SRS (Start Reversed String) は、ブロック内の文字の配置を逆方向に指定することができる。

6.複数行の表示

前節 1) による一行は、改行により表示上分割され、複数の行を形成することが可能である。改行は、以下の2つに分類される。

1) 表示領域の制限による自動改行
2) 制御機能による強制改行

ほとんどのシステムでは、左から右への表示、上から下への改行に固定されており、その表示領域の横幅に合わせて改行する。複数行が形成され、表示領域の縦幅に収まらない場合、スクロールによって表示したい箇所を表示領域に移動させる。即ち、一行の表示では考慮されなかった、領域の大きさの概念が必要となる。この実際のテキストの領域を仮想配置領域と定義する。ここで、上から下への改行の場合、仮想配置領域は縦に伸長するが、縦書きでは、改行は横方向に行なわれるので、仮想配置領域は横に伸長する。全世界の文章が混在した場合、仮想配置領域は縦と横に伸長するため、スクロール機能は縦と横方向になされなければならない。

全世界の文章を対象とした時、改行方向は4方向存在する。これらを混在配置するために、改行方向が同一の行の列の単位をページと定義し、ページを仮想配置領域に配置する (図11)。

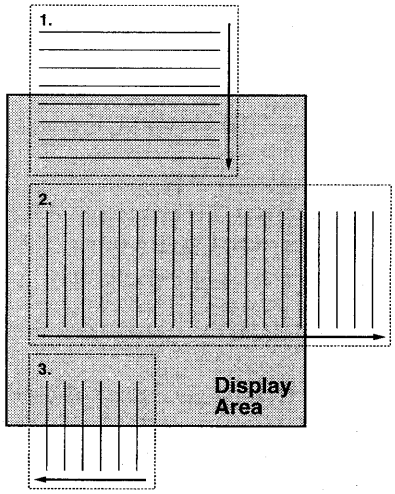

図11. 複数行の配置

ここで、SPD、SRS を使用しての図形列の二次元配置指定をした場合、改行不可能となるため、一行で一ページを構成し、他のページに SPD、SRS の効果は影響しない。

デフォルトとして、縦方向にページを配置し、かつページ幅と表示領域の横幅を一致させておくことで既存のシステムと互換性を保つことが可能である。ページの配置はアプリケーションやユーザによって指定可能でなければならない。従って、ウイジェットでは、仮想ページのXとY座標、及び、配置のストラテジのリソースが必須となる。以上のことから、国際化

は、常に二次元への配置を考慮しなければならず、表示すべきテキストの文字配置は、mb での配置情報と必ずしも同じではない。

7.二次元配置のためにフォントが持つべき情報

実際の出力では、図形に対応したフォントを表示又は印刷する。一般に図形とフォントは1対1に対応しているが、出力装置の特性に合わせて、図形の縦分割、横分割を行なってフォントとし、それを組み合わせて出力させる場合もあり、図形との対応はフォントのデザインによる。よって、最終表示図形からフォント列への写像は1対1ではなく、1対複数となる。

フォントの配置も図形列の配置同様、二次元座標上に配置されるので、フォントの配置方向も左右上下の4方向存在する。フォントを配置するためには、フォントの原点と、フォントの幅と高さの情報を持たなければならない。ここで、フォントの原点についての例として、左を詰め位置として図形を配置し下に折れ曲がる一行と、右を詰め位置として図形を配置し下に折れ曲がる一行の表示を考える (図12)。

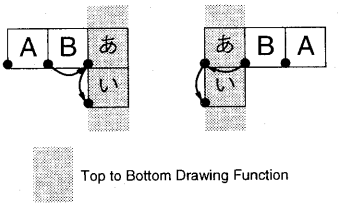

図12. 詰め位置が異なる場合の折れ曲がり表示例

折れ曲がりの指示は折れ曲がり部分の図形の直前に指定されるので、折れ曲がり部分の図形は上から下への表示関数で表示される。フォントの原点を左下に固定した場合、詰め位置が左の表示では、折れ曲がり部分のフォントの配置座標は、直前の座標に直前のフォントの幅を加えて得ることができ、続くフォントは、X座標はそのままで、フォントの高さを直前のフォントのY座標に加えて、次のフォントの配置座標を決定する。しかし、詰め位置が右の表示では、折れ曲がり部分のフォントの配置座標は、直前の座標からそのフォントの幅を引いて得られ、続くフォントは詰め位置が左の表示と同様となる。即ち、詰め位置によって上から下への表示関数の動作が異なるため、詰め位置ごとに表示関数を用意する必要がある。従って、表示関数の一般化のために、フォントは図形の配置方向毎に原点を持たなければならない。

8.カーソルの表示手法

文字と図形の関係が1対1でなく、かつ表記方向の混在したテキストを編集する場合、メモリイメージと表示イメージが一致しない。従って、アプリケーションが、カーソルの位置を決定できないため、出力機構がカーソルの表示に責任を持つ。

メモリイメージと表示イメージの不一致は、Ligature に見られるような、文字数と表示図形数の不一致の際に生じる。文字操作を行なう場合、単に図形を指定するだけではなく、その図形を指定した元の文字を表示し、操作可能とする必要がある (図13)。

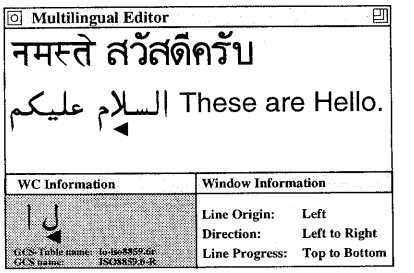

図13. 文字操作のためのカーソル表示

表示方向が混在した一行の文字操作では、例えばブロックの境界に文字を挿入する場合、文字の持つ方向により、挿入された文字の属するブロックが異なり、表示イメージも異なってくる。そのため、ブロックの境界では挿入する文字の方向による、表示位置の相違をカーソルで表示する必要がある (図14)。

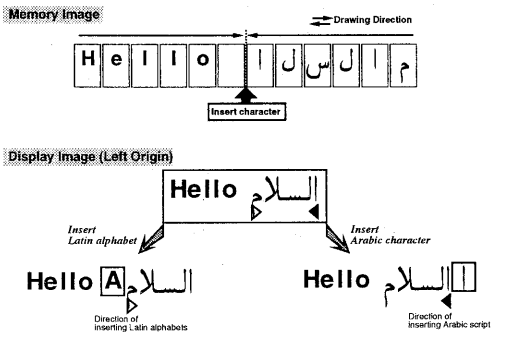

図14. 表示方向混在時の文字の挿入

9.まとめ

文字の構造が一般化され、文字と図形の関係が明らかになったため、文字コード(mb)・文字集合(WC)・最終表示図形集合(FGS)の関係を定義することが可能となった。mbからWCへの写像は、文字コードの分析により、20弱の写像関数により可能である。WCは単なるIDを持つコードではなく、mbへの逆変換、図形表示情報を持つので、mbへの逆写像も可能であり、計算可能性を保証している。WCからFGSへの写像は、文字が図形集合を示しており、文字の表示方向・表示位置の解析と変字体の指定により、文字に対し一意に図形を割り当てることが可能である。変換の概略を図15に示す。

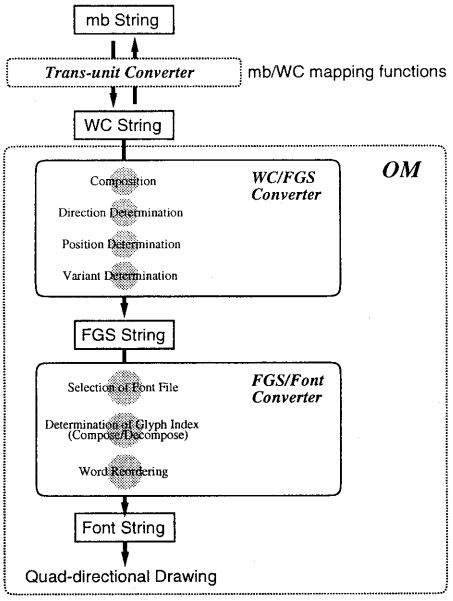

図15. mb - WC - FGS の変換過程

基本的なOS及びWindowing Systemに関しては、ライブラリとして実装され、POSIXとX Window Systemを置換して、完全な国際化を実現している。

表示手法の分析により、表示方向の混在したテキストの表示が可能であることが明らかとなったため、上記のライブラリを用いて、二次元配置可能な上位の表示機構を実装中である。これらの使用により、国際化したテキストフォーマッタ、テキストエディタ等のアプリケーションの開発が極めて容易となった。

参考文献

[1] ISO/IEC 2022: 1986, Information processing – 7-bit and 8-bit coded character sets – Code extension techniques.

[2] ISO/IEC 6429: 1992, Information processing – Control functions for coded character sets.

[3] ISO/IEC 9945-1: 1990, Information technology – Portable Operation System Interface (POSIX) Part 1: System Application Program Interface (API) [C Language].

[4] ISO/IEC 9899: 1990, Programming language – C.

[5] ISO/IEC 10646: 1992, Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architechture and Basic Multilingual Plane.

[6] TIS 620-2533 (1990), Thai Character Codes for Computers, Thai Industrial Standards Institute, Ministry of Industry, Thailand.

[7] IS 13194: 1991, Indian Script Code for Information Interchange – ISCII, Bureau of Indian Standards, India.

[8] GB 8045-87 (1987), 信息処理交換用蒙古文七位和八位編碼図形字符集, 国家標準局, 中国.

[9] Kataoka Y. et.al., Codeset Independent Full Multilingual Operating System: Principles, Model and Optimal Architecture, IPSJ SIG Notes, System Software & Operating System, Vol. 95 No. 36, Mar. 1995, pp. 25-32.

[10] Kataoka, T. Inagawa et al., Definition of the Mongolian Character Codesets Enabling Multilingual Text Manipulation, IPSJ SIG Notes, Computer and Humanities, 96-ch-29, pp. 61-66.

[11] Uezono, K. et al., The Worldwide Multilingual Computing (2): Functions, Models, Design and Architecture of Multilingual I/O TM/C System, Proceeding of the 51st Genaral Meeting of IPSJ, Vol. 3, September 1995, pp. 247-248.

[12] Oya, T. et al., The Worldwide Multilingual Computing (5): Multilingual Text Manipulation and Text Widget, Proceeding of the 51st General Meeting of IPSJ, Vol. 3, September 1995, pp. 255-256.

[13] Oya T. et al., Sequential Talks on the Multilingual Text Processing (3): The Basic and Advanced Multilingual Text Widgets, Proceedings of the 53rd General Meeting of IPSJ, Vol. 4, September 1996, pp. 127-128.

[14] Uezono, K. et al., Sequential Talks on the Multilingual Text Processing (5): Extension of POSIX to Multilingual Processing, Proceedings of the 53rd General Meeting of IPSJ, Vol. 4, September 1996, pp. 131-132.