社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2006－ＤＤ－56（１）
2006／7／28

インド系言語レガシデータの符号化方式自動識別

クメール語フォントにおける非標準レガシ符号の調査

鈴木　俊哉† 　佐藤　大††

† 〒739-8511 東広島市鏡山 1-4-2　広島大学情報メディア教育研究センター
†† 〒980-8574 仙台市青葉区星陵町 1-1　東北大学病院メディカル IT センター
E-mail: †mpsuzuki@hiroshima-u.ac.jp, ††satodai@sic.med.tohoku.ac.jp

あらまし　Unicode を用いたインド系文字の符号化はレンダリング処理の複雑化を伴う．そのため標準規格外の符号化が広く用いられ，文書データの流通を阻害している．クメール語のレガシ符号フォントを例に網羅的に調査し，フォントのメトリクス情報を用いた容易な符号化方式識別法を提示した．これは汎用的符号化方式への自動変換を可能とする．

キーワード　クメール語，Unicode，TrueType フォント，情報交換，アーカイブ

Auto-Detection of Encoding in Legacy Documents of Indic Script

Non-Standardized Legacy Encodings in Khmer Fonts

suzuki toshiya† and Dai SATO††

† Information Media Centrer, Hiroshima Univ., Kagamiyama 1-4-2, Higashi-Hiroshima-shi, 739–8511 Japan
†† Strategic Informatics Center, Tohoku Univ. Hosp. Seiryo-cho 1-1, Aoba-ku, Sendai-shi, 980-8574 Japan
E-mail: †mpsuzuki@hiroshima-u.ac.jp, ††satodai@sic.med.tohoku.ac.jp

Abstract　Although Unicode text layout systems are introduced into modern text processing softwares, still legacy character encodings are widely used for Indic scripts, to avoid using of intelligent text layout systems. There are scripts which have many fonts encoded by some non-standardized way. Such encodings may inhibit the text data extraction from the document. We investigated encodings in many TrueType fonts for Khmer as an example, and proposed an algorithm to identify the encoding method used in the fonts. By the algorithm, the text extraction of the document including legacy TrueType fonts can be automated.

Key words　Khmer, Unicode, TrueType Font, Information Exchange, Archive

1. 背　　景

ブラフミ文字を起源とするインド系の文字は，基底文字となる子音文字に対し上下左右に母音記号や声調記号を配置し，複雑な合字規則を持ち，さらに配置の順序は発音の順序と一致しないという特徴を持つ．インド系の文字では，子音，母音，声調記号は単体では必ずしも字形が決定できず，一つの音節を形成するよう組み合わせてはじめて字形が決定される．このように字形が決定された一音節分の組み合わせをクラスタと呼ぶ．この特徴のため，インド系の文字の符号化は，ローマ字，アラビア文字，日中韓の言語に比べて困難である．以下，本稿の背景として，インド国内および東南アジアにおける，インド系文字の符号化方式について簡単に整理する．

1.1　インド公用語文字の符号化方式

国際標準規格に基いた情報交換のためには，インド系の文字の文字集合として ISO 10646 [1] を用い，Unicode [2] による符号化を行なうことが，現時点では望ましいとされる．ISO 10646 における文字集合の定義と Unicode における符号化方式は，インドの国家規格 IS 13194:1991 (ISCII) [3] を基本としている．ISCII の文字集合はクラスタの集合ではなく，母音や子音などの音素を指示する抽象的な文字 (音素文字) の集合を定義したものである．そのため，文字集合内の一つのコードポイントは，特定の字形を指示することはできない．また符号化した各音素の順序には，左から右または上から下といった表示順に符号化する方法と，発音に基づき音韻順に符号化する方法がありうるが，ISCII は音韻順の符号化を採用している．以下では，

このような方式を音素文字集合の音韻順符号化と呼ぶ．従って，ISCII に基づくテキストデータの処理には，音素単位のデータから表示用の字形を決定するインテリジェントなテキストレイアウトシステムやインプットメソッドが必須である．インドでは 1983 年頃からこのような処理系が開発されてきた [4].

ISCII の基礎となった ISSCII-1983 は，MS-DOS 上でのインド公用語文字と ASCII 文字との併用を考えた符号化 (マルチリンガル符号化) であった．なお音素文字集合の音韻順符号化は，かならずしもインド系の文字で一般的な発想ではない．事実，インドのタミルナド州でタミル語の符号化方式として提案された TAM 符号化方式 [5] や，バングラデシュのベンガル語の符号化規格 BDS 1520:1995 [6] では，文字集合はクラスタを表示するための図形的な部品の集合として定義されており，符号化も表示順に行なう (図素文字集合の表示順符号化)．また，TAM 符号では 158 個，BDS 1520:1995 では 217 個のコードポイントが定義され，ASCII 文字がわりあてられているコードポイントも利用している．つまりこれらは，ASCII 文字等との併用を考慮しない単一言語のみを対象とした符号化 (モノリンガル符号化) といえる．

1.2 インド系文字の子音表現

インド系文字における文字集合の種類とその大きさには，複合子音の表記方法が大きく影響する．インド系文字の原形となったブラフミ文字は本来サンスクリット語の音声を表記するために作られた文字である．その子音文字には随伴母音が含まれており，子音を随伴母音以外の母音と組み合わせるには，子音文字に母音記号を付加して表記する．このため，独立した母音文字 (孤立形) が使われることは少ない．このようなブラフミ文字の正書法では，サンスクリット語にない複合子音や終子音の表現は困難であった．

このためブラフミ文字には，サンスクリット語と音韻的に異なる言語を表記できるよう，様々な拡張が加えられた．拡張方法を大別すると，特殊な結合文字を使う北方系と，随伴母音の無い子音文字を追加する南方系の二つがある．これらの文字のクラスタ形成例を図 1 に示す．北方系のデヴァナガリ文字の場合，クラスタは結合文字で表記され元の文字に分解することは困難であるが，南方系のタミル文字やテルグ文字の場合は，クラスタの元の文字への分解が容易である．

東南アジアのインド系文字は南方系のブラフミ文字から発展したもので，クラスタを元の文字に分解することが容易である．クメール文字でクラスタを形成するために変形させた子音文字は，基底文字に対する配置関係から脚文字と呼ばれる．それに対して，変形前の形である随伴母音付き子音文字は胴文字と呼ばれる．

1.3 インド系文字の文字集合

インド系文字の直感的な図素および音素文字集合の大きさは，実在の Unicode フォント製品に含まれる描画データ (グリフ) 数と Unicode コードポイント数 (表 1) に，それぞれ反映されていると考えられる．(TrueType フォント形式 [8] ではこの二つは同一だが，Unicode フォント製品では，一つの Unicode コードポイントに対して複数のグリフをわりあてたり，連続する Unicode コードポイントのパターンに特別なグリフをわりあてるための拡張が実装されている [9], [10]．分解困難なクラスタを持つデヴァナガリ文字などは結合文字全体を 1 つのグリフとし，分解が容易なクラスタを持つタミル文字やテルグ文字などは各音素に対応した字形をグリフとする傾向があると考えられる．従って，あるフォントのグリフ数とコードポイント数の比は，その文字のクラスタ分解の困難さを反映していると期待される．

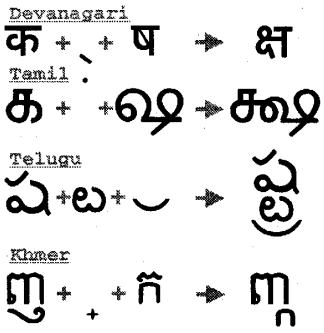

図 1 デヴァナガリ文字，タミル文字，テルグ文字，クメール文字の複合子音を表記する結合文字形成例 [7]．デヴァナガリ文字，タミル文字の例は 2 つの子音文字を脱母音記号を使って結合している．テルグ文字の例は，1 文字目の基底子音文字に対して，2 つの半字形を結合している．クメール文字の例は，2 つの子音文字を脚文字化記号を使って結合している．

北方系の文字のフォントでは，Unicode コードポイント数に対し約 6 倍の 500 ～ 600 個のグリフを含んでいた．これらの文字の図素文字集合の大きさはシングルバイト符号化が可能な 256 個という制限を大きく越えていた．例外的に，グルムキ文字は Unicode コードポイント数 77 個に対して，収録されるグリフ数は 3 倍未満の 181 ～ 211 個であった．従って，グルムキ文字は図素文字集合をモノリンガルにシングルバイト符号化することが可能である．

なお，表 1 のベンガル文字フォントでは収録グリフ数が 600 個前後であったが，先に紹介したベンガル文字の符号化方式 BDS 1520:1995 では文字集合のコードポイント数が 217 個と約 1/3 の大きさである．この違いは，Unicode のベンガル文字フォントにおいては各コードポイント (音素文字集合に対応) に一つないし複数のグリフを定義するのに対し，もう一方の BDS 1520:1995 では，音素文字集合よりもさらに細かくクラスタを分解し，得られた汎用性が高い図形をコードポイントに対応させているためである．

南方系の文字のフォントでは，Unicode コードポイント数に対しグリフ数は 2 ～ 7 倍とばらつきがあった．このうち，タミル文字フォントのグリフ数は 142 ～ 149 個，マラヤラム文字フォントのグリフ数は 207 ～ 248 個であった，従って，これらは図素文字集合をモノリンガルにシングルバイト符号化することが可能である．

以上に示したようにインド公用語文字には，図素文字集合をシングルバイト符号化が可能な文字も存在することが分かった．

系統	文字名	フォントファイル名	N_{glyph}	N_U
北方系	Devanagari	RKRaghuHindi.ttf	625 (623)	106
	Bengali	RVRaghuBengali.ttf	641 (592)	91
	Gurumukhi	RORaghuPunjabi.ttf	181 (211)	77
	Gujarati	RVRaghuGujarati.ttf	689 (1252)	83
	Oriya	RRRaghuOriya.ttf	495 (-)	81
南方系	Tamil	RRRaghuTamil.ttf	142 (149)	71
	Telugu	RORaghuTelugu.ttf	306 (562)	80
	Kannada	RORaghuKannada.ttf	418 (349)	82
	Malayalam	RRRaghuMalayalam.ttf	207 (248)	78
	Khmer	KhmerOS.ttf	307	114
	Thai	Garuda.ttf	119	87
	Lao	phetsarath_OT.ttf	72	65
	Myanmar	Myanmar1.ttf	128	78

表 1 インド系文字の Unicode フォント中グリフ数 (N_{glyph}) と Unicode 中のコードポイント数 (N_U) の比較．ISCII で定められるインド公用語文字 (デヴァナガリ文字，ベンガル文字，グルムキ文字，グジャラティ文字オリヤ文字，タミル文字，テルグ文字，カンナダ文字，マラヤラム文字) のうち，オリヤ文字以外は WindowsXP にフォントが附属しているので，そのグリフ数を括弧内に示した．

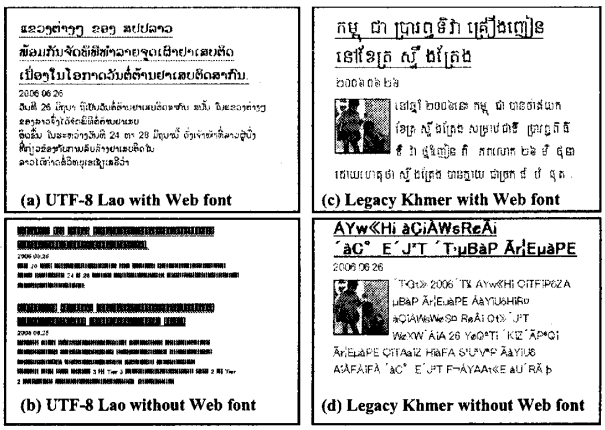

図 2 Web フォントを使ったインド系文字の Web ページの表示例．(a) UTF-8 符号化したラオ語ページを Web フォントにより表示した場合，(b) UTF-8 符号化したラオ語ページをフォントの無い状態で表示した場合，(c) レガシ符号化したクメール語ページを Web フォントにより表示した場合，(d) レガシ符号化したクメール語ページをフォントの無い状態で表示した場合．

しかし ISCII では，マルチリンガル符号化の実現と符号化方式の統一のために，これらの文字についても音素文字集合を使用して 128 個以下の文字集合を定義している．

1.4 東南アジアにおけるインド系文字の符号化方式

クメール文字，タイ文字，ラオ文字，ミャンマー文字など，東南アジア各国で用いられているインド系文字は南方系に属す．これらの文字のグリフ数 (表 1) は，最も多いクメール文字でも 307 であった．この結果から，図素文字集合に対応するコードポイント数を 256 個以下に整理できる可能性があると考えた．

Unicode が用いる ISO 10646 文字集合においては，早期に独自の国家規格を定義したタイ [11], [12] を除けば，ISCII と同様に音素文字集合を定義している [1]．ただし，ISSCII-1983 策定と同時期から音素文字集合を扱うテキストレイアウトシステムが実装されてきたインドの場合とは異なり，東南アジアの国々ではこのような技術的な蓄積がなかった．そこで東南アジア諸国においては，文字集合コードポイントとグリフが一対一に対応している欧米用ソフトウェアで自国語を使えるような，シングルバイトの図素文字集合を実装したフォントが広く用いられてきた [13], [14]．これらのフォント製品中で用いられる文字集合や符号化方式のほとんどは名称がなく，私的規格としての定義もされていない．一般に Unicode との互換性がない図素文字集合の表示順符号化をレガシ符号化と呼ぶが，本論文では特に上記のように公開された規格を持たないものを，単にレガシ符号化と表記する．なお，インド国内においても広く使われているレガシ符号フォントが存在する．これは MacOS や Microsoft Windows の標準的なアプリケーション (Web ブラウザやオフィススイートなど) が ISCII に対応していなかったためと考えられる．

1.5 レガシ符号フォントの問題点

一般に文書データを正しく表示させるためには，文書データ作成時の符号化方式に対応したフォントを使用する必要がある．Unicode などの規格が公開された符号化方式の場合には，その規格に準拠したフォントであれば作成時に使用した物と同一である必要はない．仮に作成時の符号化方式について情報がない文書データが存在する場合にも，公開された情報を元にビットパターンから符号化方式を推測することが可能である．

しかし前述したようなインド系文字のレガシ符号化については規格が存在しない，もしくは公開されていないために，このような対応ができない．従って，インド系文字をレガシ符号化した文書データの表示を保証するには，作成時に使用したフォントを常に添付して流通させる必要がある．

例えばある文字で記述された Web ページがある場合，その符号化方式が判別でき，その方式に準拠したフォントがあれば，そのページを正しく表示させることができる．規格が公開されている方式で符号化されている場合には，作成時に使用したフォントがユーザ環境に存在しない場合でも，同一規格に準拠した別のフォントで代用することが可能である．

ユーザ環境に使用可能なフォントが何もない場合でも，CSS などを利用して表示時に必要なフォントをダウンロードさせ (いわゆる Web フォント) [15]，作成した Web ページを正しく表示させることができる．この方法では，レガシ符号フォントで作成された Web ページでも，見読性を維持することが可能である．実際に，この方法で運用されている Web ページがある．

図 2(a) は，Unicode(UTF-8) で符号化されたページを Web フォントで表示させている．Web フォントが欠けた場合，図 2(b) のように表示され，Web フォントを利用して見読性が維持されていることが確認できる．

同様に図 2(c) も Web フォントを使って見読性を維持しているクメール語の Web サイトで，レガシ符号化が用いられている．Web フォントが欠けた場合，図 2(d) のように表示される．

これらの例から，Web フォントが使用可能であれば Unicode でもレガシ符号フォントでも，同じように見読性が維持できる

ことが分かる．

これら二例ではいずれも見読性が維持されているが，その可搬性には大きな差異がある．前者では UTF-8 のラオ語テキストであることが容易に判別可能であり，UTF-8 に対応した検索エンジンのクロールにより，UTF-8 のキーワードでの検索にヒットするようになる．同時に，検索エンジンが他の公開された規格を持つ符号化方式にも対応していれば，その符号化方式での検索にもヒットすることになるだろう．このことを理解するには，日本語のキーワードで検索を行なうと，JIS, Shift-JIS, EUC-JP, UTF-8 で記述されたページが，相互にヒットすることを想起すると分かりやすいだろう．一方後者の場合は，検索エンジンにとって未知の符号化方式であるために，他の符号化方式で記述されたページと相互に変換されることはなく，同一符号化方式で検索しない限り，正しくヒットすることはない．このようにレガシ符号フォントの使用は，文書の見読性を低下させるだけではなく，その可搬性を大きく損ない，結果としてレガシ符合フォントを共有する集団の内部のみでしか流通し得ない文書情報を生み出すことになる．

1.6 クメール文字の符号化の状況

クメール文字は，カンボジアの公用語文字である．その符号化方式については，カンボジアの国家規格が存在しない状態で，国外のクメール語専門家によって ISO 10646 の規格化が行なわれた [16]．当時，ISO 10646 を用いたクメール文字のレンダラの実装は，Yannis によるクメール語 TeX だけであった [17]．

しかし前述の技術的蓄積の不足や国家規格の不在により，Web ブラウザやオフィススイート等のアプリケーションにおける Unicode に基づいたクメール語サポートは大きく遅れ，相互に互換性がない文字集合を用いたレガシ符号フォントが乱立している．これらは，MacOS のクメール語符号化方式を除けば，ベンダによる符号化方式の規格書も公開されていない．

以上のように最も符号化方式が混乱している文字の一つであるクメール文字を例にとり，インターネット上で配布されているフォントの符号化方式について調査した．また，その結果を踏まえて，使用されているフォントの符号化方式を自動識別する可能性について検討した．符号化方式の識別は符号化文字列のビットパターンを対象に行われることが多いが，文書データの内容が不明なまま行なう識別では，偶然の一致や論理的に判別不可能なビットパターンが容易に発生する．これに対してフォントを対象とした符号化方式の識別は，文書データのビットパターンと比べてフォントの種類が圧倒的に少ないため，符号化文字列を対象とするよりも高い精度で実現できると考えられる．

2. クメール文字フォントの調査

クメール文字のレガシ符号フォントは MacOS や Windows の普及にともなって開発されたため，フォント形式としては TrueType フォントである．現在では Windows の方が広く使われているため，今回は Windows 用の TrueType フォントを対象とし，189 種のクメール文字フォントを収集して調査した．

種別	符号化方式	bit	フォント数
Apple Unicode	Unicode-1.0	16	183
Apple legacy	Roman	8	189
Microsoft	Symbol	8	4
	Unicode	16	189

表 2　cmap サブテーブルの種別および符号化方式と，それを含むクメール文字 TrueType フォントの数．いずれのフォントにも含まれていなかった符号化方式については省略した．

2.1 符号化方式の宣言

TrueType フォントの cmap テーブルは，文字符号コードポイントとフォント内のグリフ管理番号の対応表 (cmap サブテーブル) を持つ [18]．cmap テーブルは複数の cmap サブテーブルを持てるが，OS やアプリケーションはその中から適切なサブテーブルを一つ選択し，文字符号コードポイントからフォント内のグリフを選択する仕組になっている．cmap サブテーブルはヘッダにどの文字符号に対する表であるかの宣言を含む．cmap サブテーブルには Apple 用 Unicode，Apple 用レガシ符号，Microsoft 用の 3 種類があり，それぞれ選択できる宣言内容が決まっている．このうちクメール文字のレガシ符号の宣言が可能なのは Apple 用レガシ符号 cmap サブテーブルのみで，選択できる宣言も MacOS 標準レガシ符号だけである．

対象とした TrueType フォントに含まれる cmap サブテーブルの数を，符号化方式の種別と宣言内容ごとにまとめた結果を表 2 に示した．全てのフォントは Apple 用レガシ符号 cmap サブテーブルを含むが，クメール文字の MacOS 用レガシ符号化方式を宣言しているフォントは見当たらず，全てが Roman と宣言されていた．また Apple Unicode 用 cmap サブテーブルや Microsoft 用 cmap サブテーブルでは符号化方式として Unicode が宣言されているが，0x0000-0x00FF(ASCII/Latin-1 領域) のコードポイントにクメール文字が配置されているなど，宣言されている方式とは異なる方法で符号化が行われているフォントがあった．

これらの結果から，クメール文字のレガシ符号化フォントにおいては，cmap サブテーブルで宣言されている符号化方式の情報は信頼できず，TrueType フォントの仕様に基いた演繹的な識別が不可能であることが分かった．

2.2 符号表の作成

クメール文字レガシ符号フォントにおいて，実装されているクメール文字のグリフ数，用いられている符号化方式，符号化方式間での互換性など，それぞれのフォントについて具体的な情報を得るために，印字結果を元にして文字符号コードポイントとグリフを対応させた符号表を作成した．

調査対象とした 189 種全てのフォントは，Apple レガシ用および Microsoft 用 Unicode cmap サブテーブルの両者を持ち (表 2)，183 種のフォントが持つ Apple 用 Unicode cmap サブテーブルの全てが，Microsoft 用 Unicode cmap サブテーブルと同一内容であった．8 ビット用の cmap サブテーブル (Apple レガシ用) では，グリフを割り当てていないコードポイントと空白文字を割り当てているコードポイントの区別が TrueType の仕様上不可能であることと，調査対象が全て Windows 用フォントであ

符号化方式	フォント数	コードポイント数	備考
ABC	46	153	
Zero Space ABC	33	153 - 167	
Theodore Rith Heng	19	151	
XIMPLEX	26	213	
KHEK	2	219	
KHEK1	2	219	KHEK 派生
KHEK US	1	160	
KSCII	1	226	Mac 用
Kh	7	407	Unicode 含む
KSW	23	216	
Limon	18	154	
CDT	7	160	
Cambodia Portal	3	182	
SEA	1	125	言語学教材

表 3　調査したフォントの符号化方式とコードポイント数

ることから，符号表の参照元は Microsoft 用 Unicode cmap サブテーブルとした．

2.3　符号化方式の概観

調査した 189 個のクメール文字フォントから得られた符号化方式の一覧を表 3 に示す．符号化方式の名称が不明なものは，開発者または配布者を符号化方式名とした．表 3 には各符号化方式のコードポイント数のみ示しているが，レガシ符号フォントは Unicode フォントのような拡張を持たないので，各フォントのグリフ数はコードポイント数と同じである．調査したフォントのうち，もっともグリフ数の少ない SEA 符号のフォントは北イリノイ大学の言語学教材として作成されたもので，カンボジア国内では使われていない．

クメール文字の合成規則では，理論上区別されるべきクラスタは 535060 個，日常的に使われているクラスタは 2821 個あり，クメール文字フォントはこの程度の数のクラスタを提供できれば実用的と見積られている [17]．一方今回の調査では，ほとんどのフォントのコードポイント数は 8 ビットで収まる 256 個以下であった (表 3)．参照元とした Microsoft 用 Unicode cmap サブテーブルは 16 ビットの空間を持つが，これらのグリフは ASCII/Latin-1 領域のみで定義されていた．ただし Kh 符号化フォントは例外で，ASCII/Latin-1 領域以外に 0x1780-0x17FF (Unicode のクメール文字領域) にも Unicode 用のグリフが定義されていた．しかし ASCII/Latin-1 領域のみでもレガシ符号化フォントとしては使用可能であり，この領域内のコードポイントは 178 個であった．

以上のように，全てのレガシ符号は 8 ビット符号化方式であることが分かった．従って，それぞれのコードポイントは，クラスタの全体ではなく部分に対応した物であると考えられる．レガシ符号フォントのグリフ数は Unicode クメール文字フォントのグリフ数 (307 個，表 1) に比べて少なく，レガシ符号フォントは表示できる字形に制限があると思われる．

レガシ符号化フォントの文字集合では字形生成を伴うようなインテリジェントな処理を前提とせず，脚文字も個別に符号化していた．また符号化の順序も Unicode の場合と異なり表示順

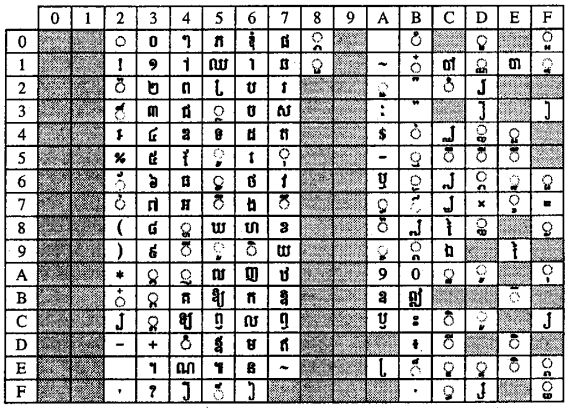

図 3　Zero Space ABC 符号 (使用フォント: text02a.ttf)

符号化が採用されていた．レガシ符号化フォントでは，クラスタの字形は複数のグリフを重ね打ちすることで生成する．このために母音記号や脚文字は，子音文字に対して重ね打ちできるようメトリクスが調整されていた．このような，重ね打ちなどのために印字ごとに描画位置が移動しないようメトリクスが調整された文字をノン・スペーシング・キャラクタと呼ぶ．これに対し，印字すると描画位置が移動する文字をスペーシング・キャラクタと呼ぶ．

なお，今回調査対象としたフォントは 16 ビットコードポイントを使っていた．従って，コードポイント空間としては日常的に使われる 2821 個のクラスタを十分格納可能である．それにも関わらず 8 ビット符号化が行われていたことは，マルチバイト入力を支援するインプットメソッドの使用を避けるような，強い動機付けがあることを窺わせる．

2.4　符号化の詳細

まず，カンボジア政府が使用している二つのレガシ符号，Zero Space ABC 符号と Limon 符号の符号表を例にとり，クメール文字のレガシ符号化の特徴を整理する．図 3 に Zero Space ABC 符号，図 4 に Limon 符号の符号表を示す．この 2 つの符号化方式は符号化された文字列を Unicode に変換するコードコンバータも作成されている [19], [20]．レガシ符号化のクメール文字フォントでは，母音文字や声調記号の描画データにおいて，基底文字の位置を示す点線の丸などの記号は一切含まれないが，本論文では Unicode 符号表との対比のため特に記入した．

どちらの符号化方式でも，ノン・スペーシング・キャラクタ (図中で，点線の○が付加してあるもの) のほとんどは母音記号や脚文字であって，BDS 1520:1995 のように音素よりも細かい単位での図形分解は行なっていなかった．また，スペーシング・キャラクタについても，胴文字か，あるいは基底文字の右側に配置される母音記号・脚文字であった．つまり，レガシ符号化フォントにおける文字集合は，クラスタの図形分解を行なってはいたが，各グリフには音素を対応づけることが可能なレベルに留まっていた．ただし，子音文字に対して左右から囲む形状の母音記号については，右の図形部分と左の図形部分が別々に符号化されていた．これらのグリフに対しては，音素を一対一で対応づけることは難しい．例外的に，母音記号や脚文字と結

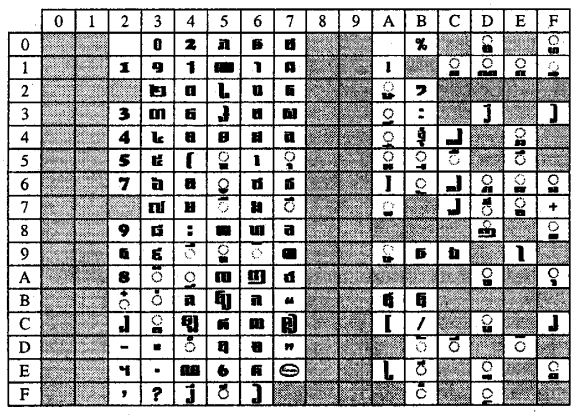

図 4　Limon 符号 (使用フォント: Limon_F4.ttf)

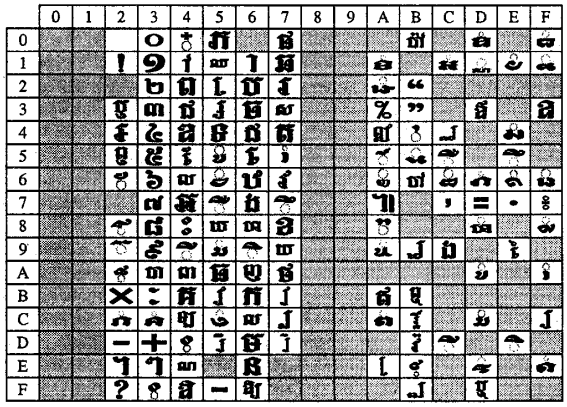

図 5　Theodore Rith Heng 符号 (使用フォント: BARAY_.TTF)

合済の胴文字のグリフがそれぞれ 2 個ずつ含まれている (Zero Space ABC 符号の 0x5B と 0xBB, Limon 符号の 0x5B, 0x7C).

まず，両符号表の共通点と相異点を整理する．なお以下では，8 ビット符号化方式のコードポイント空間のうち，0x00-0x1F を C0 領域，0x21-0x7E を G0 領域，0x80-0x9F を C1 領域，0xA1-0xFE を G1 領域と呼ぶ．大きな相異点として，Zero Space ABC 符号は基本的に算用数字を含まなかったが，Limon 符号は算用数字を含んでいた点があった．どちらもクメール数字は含まれていたが (0x30-0x39)，Limon 符号で算用数字に用いていたコードポイント (0x21, 0x23-0x26, 0x28, 0x2A, 0x40, 0x5E) は，Zero Space ABC 符号では母音記号などに用いていたので，符号化方式の互換性以前の問題として，文字集合のレベルで互換性が無いことが分かった．しかし，図 3,4 を比較すると，G0 領域の文字のうち，ASCII でローマ字アルファベットにわりあてられていたコードポイント (0x41-0x5A, 0x61-0x7A の総数 52 個) の文字は，2 つのコードポイント (0x48, 0x5E) でのみ異なり，2 つの符号表で 52 個中 50 個のコードポイントで文字が一致していた．本稿では紙面の都合上一部のレガシ符号表のみ示すが，G0 領域のローマ字アルファベット用コードポイントについては，符号化方式に互換性がない他のレガシ符号化フォントとも同様に一致していた．これは，G0 領域の符号が，ASCII キーボードをクメール文字タイプライタとして使おうとした場合のキー配列に由来すると考えられている [21, p. 84 - 86]．機械式タイプライタではローマ字アルファベットキーとシフトキーの組み合わせのみで基本的な文字を印字しなければならなかったため，タイプライタに由来する符号表でも，G0 領域に全ての胴文字と使用頻度の高い孤立形母音文字，母音記号が配置されたと考えられる．これに対し，G1 領域の文字の配置は，75 個のコードポイント中 33 個しか一致しなかった．

また，上で整理したクメール語の表記に必要とされる 138 文字 [22] のうち，記号類 (ASCII 記号 15 個，その他 22 個) はほとんど省略されており，ASCII 記号が 10 個程度しか含まれていなかった．ここで，文字のメトリクスに注目すると，G0 領域の文字は半分程度が基底文字となるスペーシング・キャラクタであるが，G1 領域はほとんどノン・スペーシング・キャラクタか，スペーシング・キャラクタであっても重ね打ち用のグリフであることが分かった．さらに，G0 領域の文字と G1 領域の文字を比較すると，0x49 と 0xCD, 0x57 と 0xC5, 0x69 と 0xED, 0x77 と 0xE5 のように同一の母音記号についてグリフの微妙な配置の違いを使い分けるために複数のコードポイントで符号化されていた．従って，これらの文字集合は大半の文字に対して音素を対応づけることは可能であるが，図素文字集合として扱うべきである．

2.5　追加される文字の分布

前節で Limon 符号と ABC 符号を例にとり，クメール文字レガシ符号は文字集合レベルで互換性がないにもかかわらず，どちらもクメール文字のスペーシング・キャラクタのほとんどが G0 領域に配置されており，その配置はよく一致していた．このことから，フォント製品の差別化はノン・スペーシング・キャラクタのバリエーションで行なわれていることが推測される．本節では，符号化方式が非常に近い Theodore Rith Heng 符号と XIMPLEX 符号を例にとり，フォント製品がコードポイントを増やした場合に，どのような文字がわりあてられるかを調べる．

まず，カンボジア国内で用いられているフォント製品の中で，もっとも収録グリフ数の少ない Theodore Rith Heng 符号の符号表を図 5 に示す．Theodore Rith Heng 符号も，ABC 符号と同様に，算用数字が無く，記号類がほとんど省略された図素文字集合であった．また，ABC 符号や Limon 符号に含まれていたような結合済のグリフは含まれていなかった．G0 領域の文字，および，G0 領域と G1 領域で重複符号化された母音記号の多くは，ABC 符号の同一コードポイント上の文字と一致していた．

次に字数の多い XIMPLEX 符号のフォントを図 6 に示す．Theodore Rith Heng 符号と比較すると，Ekreach_V3.ttf に追加されている 22 文字の内訳は子音文字が 18 個 (うち脚文字 15 個，結合済胴文字 3 個)，母音記号 1 個，記号 2 個，不明 1 個であった．脚文字の追加が大半を占めていたが，このうち，Theodore Rith Heng 符号にまったく含まれていない文字は 2 文字にすぎず，13 個は微妙に異なるメトリクスを調整するためにコードポイントをわりあてたものであった．文字によっては最大 3 個のコードポイントを用いていた．この結果から，クメール文字のレガシ符号では，図素文字集合を拡張する際，クラスタなど

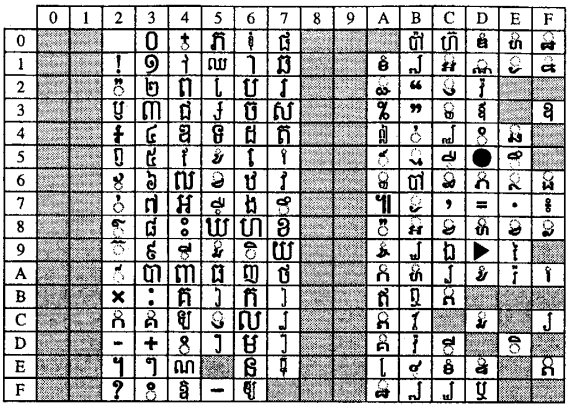

図 6　XIMPLEX 符号 (使用フォント: Ekreach_V3.ttf)

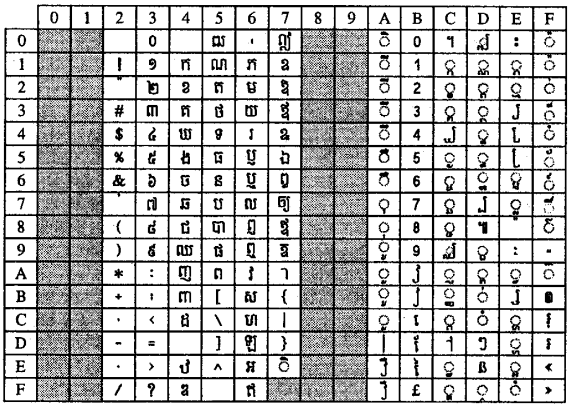

図 7　KSCII 符号 (使用フォント: KMon40K.ttf)

の新しい字形を追加することはなく，既に符号化済のノン・スペーシング・キャラクタに対するメトリクス調整のための図素文字が追加されると考えられる．

2.6 KSCII 符号

ここまで示したレガシ符号では，相互に互換性のない符号化でありながら，G0 領域の符号は類似していた．この理由としてクメール文字タイプライタからの影響が考えられる．

キーボード配列と文字符号を分離してクメール語処理用に再設計した結果，他のレガシ符号と全く異なる符号化方式となった例として KSCII 符号の符号表を図 7 に示す．KSCII 符号は G0 領域は 0x7E 以外をスペーシングキャラクタに割りあて，子音の胴文字はコードポイントが辞書順になるよう並べていた．さらに，G0 領域と G1 領域の関係づけも考慮されており，クメール数字を配置したコードポイント (0x30-0x39) に対して 8 ビット目を立てると算用数字を配置したコードポイント (0xB0-0xB9) になるようにしていた．子音文字についても，G0 領域に配置された胴文字 (0x41-0x5A) のコードポイントに対して 8 ビット目を立てると同じ子音の脚文字のコードポイント (0xC1-0xDA) になるようにしていた．このように，KSCII 符号の設計には情報交換的な観点が反映されていたが，やはりメトリクス調整のために母音記号が複数のコードポイントで重複符号化されており (0xA1 と 0xA2, 0xA3 と 0xA4, 0xA5 と 0xA6, 0xA7 と 0xA8 などは同じ母音記号であった)，図素文字集合の符号化方式であった．

3. 符号化方式の自動識別

以上の調査結果を踏まえて，フォントの符号化方式を識別する可能性について検討する．クメール文字レガシ符号フォントの cmap サブテーブルで宣言されている符号化方式に信頼性がなく，TrueType フォントの規格に基いて演繹的に識別することは困難であった．一方，グリフに対する図形認識を伴う処理は，実装の負荷が大きい．そこで符号化方式識別のための手がかりとして，フォント内部の独立したテーブルに格納されているメトリクス情報に着目した．これを用いれば，図形識別を行わずにスペーシング・キャラクタか否かの判別が可能となる．つまり未使用および (ノン・) スペーシング・キャラクタの分布 (属性分布) から，符号化方式の識別を試みるというものである．

さて，図 3-7 に示した符号表から，レガシ符号化では G0 領域はほとんど使用済みであるが，G1 領域には未使用コードポイントを残している符号化方式が多いことが分かった．表 3 に示したように，調査したフォントのうち，Zero Space ABC 符号を用いる製品では，この未使用コードポイントに追加の文字を配置し，グリフ数を増やしているものがあった．このように，未使用コードポイントでの製品差別化の影響がありうるので，未使用コードポイントの分布よりも，使用済のコードポイントでの属性分布から符号表を識別する方がより正確と期待できる．しかし，Theodore Rith Heng 符号と，それから派生したと考えられる XIMPLEX 符号のように，共通する使用済コードポイントが同一の場合は，未使用コードポイントについて比較せざるを得ない．この場合，G0 領域の未使用コードポイントの比較と，G1 領域の未使用コードポイントの比較のどちらを優先させるかの問題がある．G0 領域へのグリフのわりあてはキーボード配列にも影響するため，フォント製品差別化の観点で見ると，G1 領域へのグリフの追加よりも大きな変更であると考えられる．そこで，本稿では G0 領域の未使用コードポイントに関する比較を優先させた．以上の方針をもとに，収集したレガシ符号化フォントの符号化方式を識別するアルゴリズムの一例を図 8 に示す．

図 8 に示した識別アルゴリズムは，大きな特徴を持っている符号表を先に候補から除外してゆくという順番で組みたてている．仮に誤認識が生じた場合でも，少なくとも先に除外された符号化方式ではないと考えることができる．本稿では全ての符号表を示さなかったので，注目した符号化方式の違いについて以下に説明する．SEA 符号は G0 領域にのみクメール文字を定義しており，G1 領域にはグリフをわりあてない．他のレガシ符号は G0 領域，G1 領域の二つを利用しているので，この特徴はもっとも大きな違いである．これによりまず SEA 符号を除外する．次に，KSCII 符号は G0 領域が 0x7E を除いて全てスペーシング・キャラクタである．他のレガシ符号，特にタイプライタに由来すると考えられる符号化方式では，G0 領域にノン・スペーシング・キャラクタを含むので，KSCII 符号を検出する大きな特徴である．次に，Zero Space ABC 符号だけは 0x20 のコード

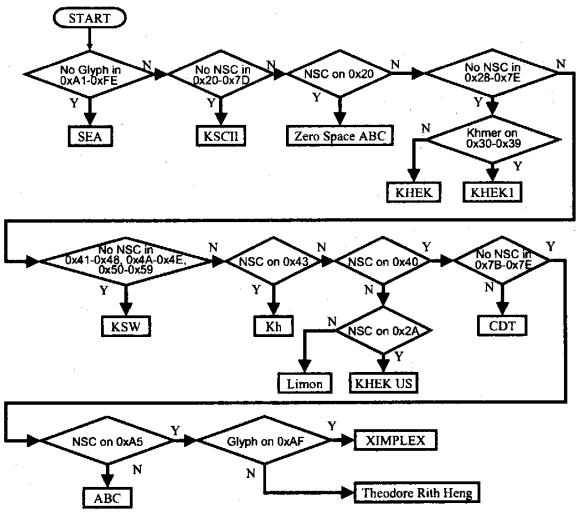

図 8　レガシ符号表の識別アルゴリズム

ポイントはゼロ幅のスペース文字がわりあてられているが，他のレガシ符号は 0x20 は通常の有限幅のスペース文字であるので，これにより Zero Space ABC 符号を検出できる．この 3 つは，符号化方式の設計に関わるものであるため，大きな特徴として現われた．これらを除外した後の符号化方式の識別は，設計方針を反映している部分が明らかではなく，十数個 ～ 数個のコードポイントでのノン・スペーシング・キャラクタの分布で識別する．これ以降の識別手順においては，未使用コードポイントに製品差別化のためにノン・スペーシング・キャラクタがわりあてられる可能性を考慮し，連続したコードポイント群にスペーシング・キャラクタが配置されていることを調べる．これにより除外できるのは，KHEK, KHEK1, Kh, KSW などである．ここで，KHEK と KHEK1 は同一ベンダによる符号化方式で，クメール数字を使う (KHEK) か算用数字を使う (KHEK1) かのみが異なるので，この識別には図形認識が必要である．ただし，クメール数字と算用数字の書法は同じであり，この違いがクメール語テキストの見読性に及ぼす影響は他の符号化方式との誤認にくらべれば小さい．これ以降の符号化方式の識別 (Limon, KHEK US, CDT, ABC, XIMPLEX, Theodore Rith Heng) は，数点のコードポイント上での属性から識別せざるを得ないので，ここで示したものとは異なる順序で除外してゆくアルゴリズムもありうる．

4. ま と め

本稿では，インド系の文字合成を行なう文字のうち，Unicode 以外の規格化がされていない文字系の例としてクメール文字をとり，レガシ符号化による TrueType フォントについて図形認識によらずに符号化方式を識別する可能性について検討した．調査の結果，インターネット上で配布されているクメール文字 TrueType フォント 189 個から 14 種類の符号化方式が見つかった．同一ベンダにより算用数字とクメール数字を置換したものを除き，文字のメトリクス種別により自動識別が可能であることを示した．

この結果から，レガシ符号化された文書データからのテキスト抽出について，符号化データのビットパターンからの符号化方式推測や，図形認識的な手法に比べて，より精度の高い抽出方法の可能性が示された．しかし，文書データを文字コードの変換なしにレンダリングするにはレガシ符号フォントが必要なことには変わりがない．レガシ符号フォントを Unicode でエミュレートする可能性についての検討を今後の課題とする．

文　献

[1] ISO/IEC JTC1/SC2: “ISO Standards: Information Technology – ISO/IEC 10646:2003, Universal Multiple-Octet Coded Character Set (UCS)”, ISO (2003).

[2] The Unicode Consortium: “The Unicode Standard, Version 4.0.0”, Addison-Wesley (2003). ISBN 0-321-18578-1.

[3] Bureau of Indian Standards: “Indian Script Code for Information Interchange”, BIS (1991).
http://www.cdac.in/HTmL/GIsT/down/iscii_d.asp.

[4] C-DAC: “GIST ISCII layout engine”.
http://www.cdac.in/html/gist/localization.asp.

[5] K. Kalyanasundaram and Muthu Nedumaran: “A Glyph-Based Font Encoding Scheme - TSCII as a candidate for Tamil Computing”, Tamilnet (1999).

[6] Bangladesh Standard and Testing Institution: “BDS 1520:1995 Bangla coded character set”, BSTI (1995).

[7] Microsoft: “TrueType Open Specification”, Microsoft (1995).
http://www.microsoft.com/typography/
SpecificationsOverview.mspx|.

[8] D. Weise and D. Adler: “TrueType and Microsoft Windows Version 3.1”, Microsoft, MSDN Library (1992).

[9] Adobe Systems Inc.: “OpenType Specification”, Adobe Systems Inc., San Jose, 1.4 edition (2002).
http://partners.adobe.com/public/developer/opentype/
/index_spec.html.

[10] Apple Computer: “Inside Macintosh: Quickdraw GX Typography”, Addison-Wesley, Massachusetts (1994). ISBN: 0-201-40679-9,
http://developer.apple.com/documentation/mac/
/GXTypography/GXTypography-2.html.

[11] NECTEC: “TIS 620:2533, Thai Character Code for Computers” (1994).
http://www.nectec.or.th/it-standards/std620/std620.htm.

[12] T. Karoonboonyanan: “Thai vs Indic Encoding Scheme” (2004).
http://linux.thai.net/~thep/thai-vs-indic.html.

[13] Y. Mikami and Z. Pavol: “Writing systems and character codes in the world (1)”, IPSJ Magazine, **46**, 8, pp. 919–924 (2005).

[14] Y. Mikami and Z. Pavol: “Writing systems and character codes in the world (2)”, IPSJ Magazine, **46**, 9, pp. 1046–1052 (2005).

[15] H. W. Lie, B. Bos, C. Lilley and I. Jacobs: “”Descriptor for Referencing: ’src’” in Cascading Style Sheets, level 2”, W3C (1998).
http://www.w3.org/TR/CSS2/fonts.html\#referencing.

[16] A. Daniels: “Unicode Technical Report #1 - Burmese, Khmer, and Ethiopia” (1992).
http://www.unicode.org/unicode/reports/tr1.html.

[17] Y. Haralambous: “Typesetting khmer”, Electronic Publishing, **7**, 4, pp. 197–215 (1994).

[18] Apple Computer: “The TrueType Reference Manual” (1996).
http://developer.apple.com/fonts/TTRefMan/index.html.

[19] KhmerOS project: “Limon2Unicode.exe, ABC2Unicode.exe” (2005).
http://www.khmeros.info/drupal/?q=en/download/others.

[20] M. Bauhahn: “Procedures/tools for Transcoding Cross-Platform Legacy Text to Unicode Using TECkit” (2004).
http://www.bauhahnm.clara.net/Khmer/TECkit.html.

[21] 三上：“文字符号の歴史 -アジア編-”, 共立出版 (2002). ISBN 4-320-12040-X.

[22] Cambodia: “Report by Higher Education Task Force” (1996).
http://www.bauhahnm.clara.net/Khmer/Page1.JPG ～ Page4.JPG.