Zawgyi と Unicode

普及しすぎた ミャンマーの
オレオレ文字コードと国際化

サークル **ヒュアリニオス**

Zawgyi と Unicode

普及しすぎたミャンマーの
オレオレ文字コードと国際化

にせねこ（@nixeneko）
サークル“ヒュアリニオス”

2024 年 5 月 26 日

目次

第 1 章

はじめに

1.1　まえがき

ミャンマーでは公用語としてビルマ語が使われている。ビルマ語の表記にはビルマ文字を用いるのだが、このビルマ文字のインターネット上での使用は、混迷を極めていた。そしておそらく今もまだ……。なぜか？

それは、Unicode という文字コードの標準がありながら、Zawgyi というものが広く使われていたためである。

1.2　フォント違いによるビルマ文字の文字化け

ビルマ語の表記のためのフォントには Zawgyi フォントと Unicode フォントがある。以前は Zawgyi が広く使われていた。

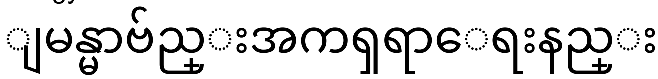
Unicode フォントは Unicode に準拠したフォントのことだが、Zawgyi フォントは Unicode に準拠していない独自の文字コードを採用している。そのため、Zawgyi テキストを Unicode フォントで表示したり、Unicode テキストを Zawgyi フォントで表示すると、正しく表示されず、意味不明な文字列になる。以下にテキストとフォントを変えた場合の見た目の比較を示すので、見比べてほしい。

(1) Unicode テキストを Unicode フォントで表示: OK

မြန်မာဗျည်းအက္ခရာရေးနည်း

(2) Unicode テキストを Zawgyi フォントで表示: NG

(3) Zawgyi テキストを Unicode フォントで表示: NG



(4) Zawgyi テキストを Zawgyi フォントで表示: OK

မြန်မာဗျည်းအက္ခရာရေးနည်း

テキストとフォントが一致している (1) と (4) は文字列の見た目も一致していて、これは正しく表示されている。しかし、テキストとフォントが異なっている (2) と (3) は文字化けしている。

正しく表示できているか確認するためには、ビルマ語がわかる人に見てもらうのが確実である。可能ならネイティブチェックを受けるのが望ましい。

このように互換性のない 2 つの方式が存在しているため、ビルマ語を扱う際に問題となっている。なぜそのようなものが登場し、普及することとなったのか、次章以降で解説する。

1.3　凡例

1.3.1　この記事で使う名称について

ミャンマーという国は、1989 年に公式の英語名称として Burma ではなく Myanmar という語を使うように変更した。Burma と Myanmar はどちらも

ビルマ族の自称（バマー ဗမာ とミャマー မြန်မာ）に由来するもので、それぞれ口語体と文語体のものであり、いずれも現用されている[1]。

本書では便宜的に、「ミャンマー」という国の主要な民族である「ビルマ族」の言語を「ビルマ語」と表記し、ビルマ語表記のための文字を「ビルマ文字」とする。また、ビルマ語以外のミャンマーの諸言語（シャン語、カレン語など）もビルマ語と近い文字を使うが、これらのミャンマー（および近隣）の言語を表記する文字を総称して「ミャンマー文字」と称することにする。

1.3.2　コードポイント

コードポイントは 16 進数 4 桁で示す。Unicode のコードポイント（Unicode スカラ値）は `U+XXXX` の形で示すことがある（`XXXX` には 16 進数が入る）。

1.3.3　ラテン文字表記について

基本的なラテン文字表記は加藤昌彦『ニューエクスプレスプラス ビルマ語』[2] の表記に倣った。また、一部発音を IPA（国際音声記号）で示した部分がある。

そのほかに、文字の名称をイタリック体で書いたものがあるが、これは Unicode で定義された文字名の一部からとったものであり、ビルマ語での発音とは異なる[*1]。現地の呼称と異なる可能性もあるかもしれない。

[*1] おそらく Unicode の文字名はビルマ文字のラテン文字転写[3] に由来するものと思われるが、ミャンマーで公式に採用されているラテン文字転写はパーリ語向けのものを元に作られているようで、ビルマ語の音韻構造を表してはいない。そのため、ある程度文字の区別が可能である一方で、転写から発音を想像することが困難になっている。

第 2 章

Zawgyi の概説と歴史

2.1　Zawgyi とは

Zawgyi は、ビルマ語表記のためのフォントであり、ゾージーと読む。Zawgyi font, Zawgyi-One などとも書かれる。おそらくビルマ文字では ဇော်ဂျီ と書き、発音が /zɔ̀d͡ʑì/ である語だと思われる。「ヨガをする人」の意らしい。

Zawgyi はフォントではあるものの、Unicode に準拠せず独自に文字を割り当てている。そのため、それ自体が文字コードのようになっている。

2.2　Zawgyi のダウンロード

次の URL から `ZawgyiOne2008.ttf` がダウンロードできる。

`https://code.google.com/archive/p/zawgyi/downloads`

2.3　Zawgyi 誕生・普及の経緯

この節はブログ記事*Battle of the fonts | Frontier Myanmar*[4] を元にしている。

2.3.1 複雑なビルマ文字

ビルマ文字はインド系文字の一種であり、複雑な用字系[*2]である。複雑な用字系とは、ある 1 つの文字が前後の文字に応じて様々に形を変える、文字が発音される順番に並ばない場合がある、などの要素を持つ文字体系である。

2.3.2 ビルマ文字と Unicode

Unicode には、1999 年のバージョン 3.0 でビルマ文字が収録された。

Unicode 以前にビルマ文字の符号化標準が作られたことはなく、そのため、それ以前はコンピューター上で扱うのは困難だったか、可能でもビルマ文字テキストデータを不特定の人とやりとりすることは不可能に近かったと思われる[*3]。

Unicode 標準に入ったため、ビルマ文字の文字列を一通りのコード列にすることは可能になったものの、Unicode のビルマ語を正しく表示するための技術は、2005 年まで存在しなかった。これは、複雑な用字系を扱うための技術的困難に起因している。

一方では西欧諸国による制裁などもあり、自国で複雑な仕組みに対応するシステムを開発する能力もなかったことから、当面の間は自分の国で何とかなる仕組みでミャンマー文字をコンピュータで使えるようにしようと、暫定的な回避策が開発された。実用できない標準より、とりあえず表示はできる非標準の方がいいだろう、ということだと思われる。これは Unicode やコンピュータの Unicode 対応が成熟するまでの繋ぎのつもりだったようだ。

[*2] Complex script. 「用字系」は script の訳で、文字の体系のことを示している。script は単に「文字」と訳されることがある（例: ビルマ文字/Burmese script）が、個々の文字（letter）と区別するために用字系と訳されていると思われる。

[*3] 当時のパソコンでビルマ文字を扱う場合、8 ビットコードの範囲内（ASCII や、Latin-1 の範囲）にビルマ文字を割り当てたフォントが使われていたらしい。ただし、フォント毎に文字の配列が異なるので、フォントを変えると文字化けしたとのこと[5]。

2.3.3 回避策としてのビルマ文字フォントの登場

まず、Ko Ngwe Tun 氏が Myazedi[*4] というビルマ文字用フォントを作り、販売した（2003 年ころと思われる）。これのフォントは、Unicode のミャンマー文字ブロックの位置にビルマ文字を収録していたが、一つの文字に形状のバリエーションがある場合、それらを別々のコードポイントに割り付けていた。部分的には Unicode と共通している部分はあるが、互換性はなかった。

非標準ではあるものの「とりあえず表示できる」フォントの最初期の実装だったが、プロプライエタリであり高額（ユーザーライセンスが 100 米ドル、コンテンツを作る会社には 1,000 米ドル）だったため、広くは普及しなかった。

Zawgyi-One（あるいは単に Zawgyi）フォントは 2006 年にフリーウェアとして登場した。Myazedi をパクったとみられ、最初のバージョンには Ko Ngwe Tun 氏の著作権表記の一部がそのまま含まれていたらしい。無料であることから、だんだん Myazedi を置き換えつつ、一般に普及していった。

ある時、Ko Ngwe Tun 氏の会社が、Myazedi の海賊版フォント（= Zawgyi）を使用した会社や Zawgyi の開発者たちを訴えると表明した。これを受けて、Zawgyi の開発者たちはパクリだとみなされないように Zawgyi フォントに改変を加え、Myazedi との互換性をなくし、Unicode 標準からさらに離れることとなった。

その間 Unicode 方面はどうなっていたかというと、2005 年に Windows XP で Unicode ビルマ文字の表示が可能になってからは、Unicode フォントが作成された。だが完全なものではなく、扱うのに技術的な知識が必要だったこともあり、Unicode は普及しなかった。また、2008 年の Unicode 5.1 で

[*4] Myazedi という名は、1112 年に刻まれたとされるミャゼーディー碑文に由来するものかもしれない。これは現在知られている限りビルマ語最古の碑文であり、パーリ語、ピュー語、モン語、ビルマ語の 4 言語で碑文が刻まれているため、ミャンマーのロゼッタストーンとも呼ばれる。

後方互換性のない大きな変更がなされるまで、ミャンマー文字ブロックは大幅な見直しが行われていて、エンコードが安定していなかった[*5]。

2.3.4　Zawgyi の普及

その後の Zawgyi のリリースで、一般ユーザーとってインストールは簡単だが、アンインストールが困難な形態のものが登場した。例えば、Arial フォントを書き換えたり、アンインストール方法のない Internet Explorer プラグインとして提供されるなどの形態があったとのこと。

Zawgyi の普及により、ネット上のビルマ語のほとんどが Zawgyi によって書かれているものとなったため、新しくコンピュータを買った人は Zawgyi をインストールしたし、一般に売られるスマホも、最初から Zawgyi がインストールされた状態で提供された。

このように事実上の標準となった Zawgyi にロックインされてしまい、Unicode への移行は困難になっていた。

2019 年に国が正式に Unicode に切り替えるぞ！と言うまでは広く広く使われていた。それからはだんだん Unicode が普及しているらしい。

[*5] とはいえ、Unicode ミャンマー文字がほとんど使われてなかったから互換性のない変更を行うことが可能だったわけで、エンコードが安定していないのが普及しない原因だった訳ではなさそうではある。

第 3 章

Zawgyi の実装

Zawgyi は、Unicode のインド系文字で使われている複雑なレンダリングの仕組みを回避するものであるため、実装は単純である。

図 1 がコード表であるが、白色部分が Unicode 5.0 以前と共通の文字であり、灰色部分は Zawgyi で独自に割り当てられた文字である。

子音字・独立した母音字などは Unicode と共通している。Unicode 5.0 以前に存在していたビルマ文字については、同じ文字が当てられているように見える。そのほかに、下に重ねて書く子音字や、記号類のバリエーション、合字などが追加されている。

次から具体的な実装がどうなっていくかを見ていく。

3.1　実装の方針

3.1.1　文字の並べ替えをせず、左から右に書く

Unicode では、できるだけ発音順に文字をデータとして収録する方針をとっており、そのため表示する際に文字を並べ替える必要がある。

Zawgyi では、並び変えの必要をなくし、左から順番に並べていくだけである。

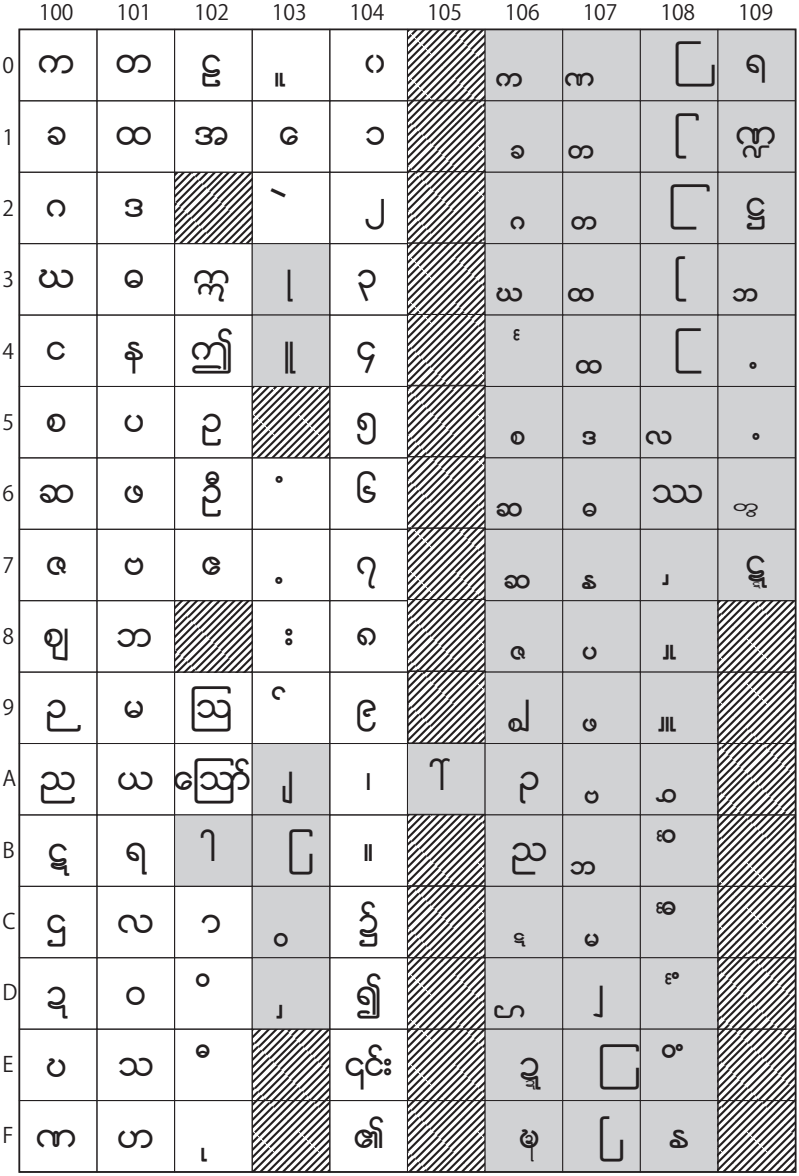


図 1. Zawgyi のコード表

図 2 に ၸျော့ cŝ「秀麗な」という語の Zawgyi と Unicode での表現の比較を示す。Zawgyi では見た通りに文字が並べられ、Unicode では発音する順に並べられているのがわかると思う。

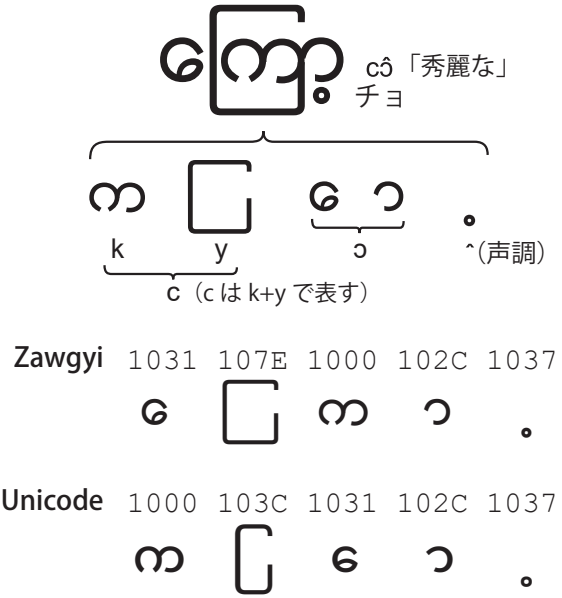


図 2. ビルマ語単語例の Zawgyi と Unicode の比較

3.1.2 文字の形のバリエーションに対して、別々のコードポイントを割り付ける

Unicode では、一つの文字が前後の文字によって様々に形を変える場合、その文字のバリエーション一つ一つに別々のコードポイントをあてることはせず、一つのコードポイントの文字を、プログラムやフォントの機能を使って形を替えることで対応するのを基本としている。

Zawgyi では、文字の形のバリエーションや合字に対して、すべて別々の

コードポイントを割り付けている。これにより、前後の文字に応じて文字の形を変更する必要がなくなり、プログラムやフォントは変形への対応が不要になる。

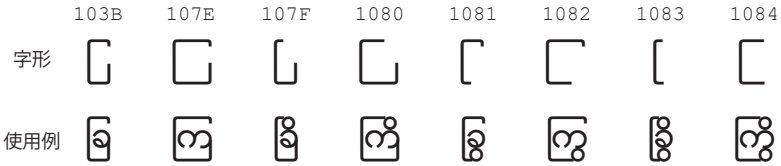


図 3. Zawgyi のもつ 8 種類の介子音記号 *ra*

顕著な例としては、介子音記号の *ra* は、子音字をぐるっと囲むような記号であるが、

1. 幅が狭い文字に付く場合／広い文字に付く場合
2. 上に記号が来ない／来る場合
3. 下に記号が来ない／来る場合

の場合分けにより、8 通りの形が収録されている（図 3）。

3.2 特徴と欠点

結果として、Zawgyi は場当たり的なものであり、次のような特徴をもつ。

表示できればいい

- 文字コードとしては定義が不十分であり、扱いづらい。
- まともな文字コードや実装が普及するまでの「つなぎ」のつもりで作られたため。

検索や照合が非常に大変

- 同じ文字の形状のバリエーション全てに別コードを当てている。
- 上下につく記号など、送り幅[*6]を持たない文字・記号は、どの順番にするかが決まっていない。
  - Unicode では、順番が変わっても表示が同じになる合成記号類は、正規化（normalization）により一定の順番になるように仕様が定められている。

ミャンマー文字を使う他言語との衝突

ミャンマーのビルマ語以外の言語向けの文字のために確保されていた当時の未定義部分（現在はすでに諸言語向け文字が収録されている）に勝手に記号類を収録しているため、そのような言語を扱う場合に困る。

[*6] advance width. 次の文字を表示する位置がこれだけ進行方向前方にズレる、という幅。

第 4 章

Unicode ミャンマー文字の実装

さて、Unicode の複雑さを回避するために Zawgyi が出てきたことを考慮すると、なぜ Zawgyi が普及するに至ったかを考えるのには、Unicode による符号化がどんなであるかを見ていく必要があるだろう。

4.1　Unicode 登場まで

Unicode は、それまでに存在していたテキストエンコーディングを置き換えることを意図して作られた。そして、レガシーな文字コードとの相互運用性を重視していることから、Unicode 以前から存在する既存の文字コード標準や、広く通用したデファクトスタンダードな文字コードは尊重される。

このため、ビルマ文字の文字コードが存在していれば、それに近い形で Unicode にビルマ文字が入ったはずである。

しかし実際には、Unicode 以前にビルマ文字の標準的な文字コードが作られたことはなさそうで、1999 年に Unicode 3.0 で実装されたのが初めてのビルマ文字の符号化標準であった。従って、デーヴァナーガリーなどの他のイ

ンド系文字の方式を参考に、新規に標準が作成されたと思われる[*7]。

4.2 論理順

インド系文字の符号化にかかわってくる Unicode の原則として、論理順がある[6]。論理順の原則というのは、かんたんに言うと、文字を、発音する通りの順番でデータとして収録するという原則である[*8]。

例えば、インド系文字では子音字の上下左右に母音記号がつき得るのだが、子音 → 母音の順に発音されるのだから、母音記号が子音字の左に書かれる場合でも、データの上では子音字 → 母音記号の順に収録し、表示する際に母音記号が子音字の左側に来るように並べ替えるというものである。

一例として、ဖြဲ့ pyèlɛ̀「うまくいく」という単語を見てみると、Unicode では図 4 のように前から p-y-è の順番のデータとなるが、見た目の順番では左から è-y-p の順に並んでいて、論理順と視覚順で順番が異なっている。

これにより、データの上ではきれいで、ソートなどもやりやすくなるのだが、表示するときには文字の入れ替えをしなければならない。この文字の入れ替えに関しては、テキストレンダリングエンジン……つまりテキストを表示するプログラムが責任を持つ。そのため、対応していないプログラムを使った場合、正しく表示されないということになる（図 4の「間違った表示」参照）。

ブラウザなどでの対応はずいぶん良くなったものの、これに対応していないソフトウェアや、設定で有効になっていなかったりして、正しく表示され

[*7] インドの諸文字やスリランカのシンハラ文字、タイのタイ文字などはそれぞれの国によって文字コード標準が作成されていて、それをもとに Unicode へと収録されたようだ。ミャンマー文字は、仕組みが似ていて当時標準が存在していなかったクメール文字と合わせて、新規に Unicode 標準策定が行われたようである。

[*8] 原則というのだから例外があり、例えばタイ文字なんかは論理順でなく表示順で左から右に並べるものになっている。これは、Unicode 策定時に TIS-620 というタイ語の文字コード標準が存在していたため、それがそのままに近い形で Unicode に入ったということのようだ。

図 4. pyèlɛ̀「うまくいく」の Unicode 表現

ないという事態は多々発生している。

話はややずれるが、ビルマ文字を手で書くときは基本的に左から右、内側から外側に書くので、上の pyèlɛ̀ の例では、è-p-y の順番で書くことになる。これは論理順と異なっているが、コンピュータでの入力の際には、手で書くような順番で入力すると、インプットメソッドが Unicode としていい感じのコード列に変換してくれたりするようだ。

4.3　グリフでなく文字

また、Unicode の原則として、グリフでなく文字を収録するというものがある[6]。ここでいう文字（character）とは、様々な形のバリエーションがあっても同じ字だと認識されるようなグループに関して、抽象化した文字を想定したものである。一方、グリフは文字の具体的な図像表現であり、文字が

目で見える形になったものがグリフである。同じ字でもフォントによってデザインが違うが、それらはグリフの違いであり、文字の違いではない。

これはつまり、一つの文字は原則として 1 つのコードポイントに割り付け、形のバリエーションは独立して収録しないというものだ。そのため、ある文字が文字の位置や前後の文字などに依存して形が変わる場合、変形後の形は Unicode に独立して含まれないのが原則であり、表示時にそのような別形への変換を行う必要がある。

これを実現するには、別形へと置換する情報をフォントに含めて、テキストレンダリングエンジンからその情報を参照して置き換えを行わなければならない。要するに、フォントとテキストを描画するプログラムの両方が対応していないと、正しく表示することができないということだ。

インド系諸文字でこれが関わってくるのが、ヴィラーマモデルである。

4.3.1 ヴィラーマモデル

インド系文字は音節文字であり、子音字は基本的には母音とセットになった状態で書かれる。子音字単体だと母音 a がついて発音される。ヴィラーマ（virama）は、子音字につけられて、母音がないことを示す記号である[*9]。ビルマ文字ではないが、図 5 に例を挙げる。ヴィラーマはサンスクリット語に

図 5. デーヴァナーガリー文字のヴィラーマの例

[*9] ヴィラーマは無母音記号だと説明したものの、どちらかというと母音をなくす概念の名前かもしれない。サンスクリット語に関する知識がないので詳しくはわからないが…。英語版 Wikipedia の Virama[7] など参照。

おける呼称であり、各言語によって呼ばれ方は異なるが、Unicode では基本的にすべてヴィラーマと呼んでいる。

ビルマ文字におけるヴィラーマはやや特殊なため、ここではインド系文字の一つであるデーヴァナーガリーを例として説明する。

デーヴァナーガリーには、ヴィラーマ記号の他に、母音をなくす表現として、半子音字と呼ばれる仕組みがある。図 6 に例を挙げる。これは子音字の一部を欠いたものであり、次の子音と連続して、子音連結として発音される場合に用いられる。

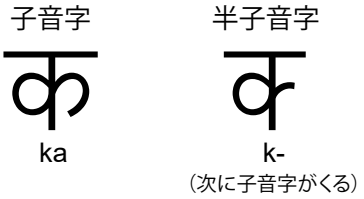


図 6. デーヴァナーガリーの半子音字の例

Unicode では、この半子音字に独立したコードポイントを当てるのではなく、子音字 + ヴィラーマの組み合わせで表現する。このように、子音字のバリエーションなどをヴィラーマを組み合わせることで表現するのがヴィラーマモデルである。

すべての子音字について対応する半子音字が存在するわけではなく、子音の組み合わせによっては縦に重なったり特別な合字で表されるものもあるが、図 7 のように同様にヴィラーマを利用して表現される[*10]。

4.3.2　ビルマ文字の場合

さて、ビルマ文字の場合を見てみよう。

[*10] サンスクリット語、ヒンディー語など、言語によって子音の組み合わせに対する合字の形や使い方が異なるため、フォントは言語毎に別の設定を行う必要があるようだ[8]。

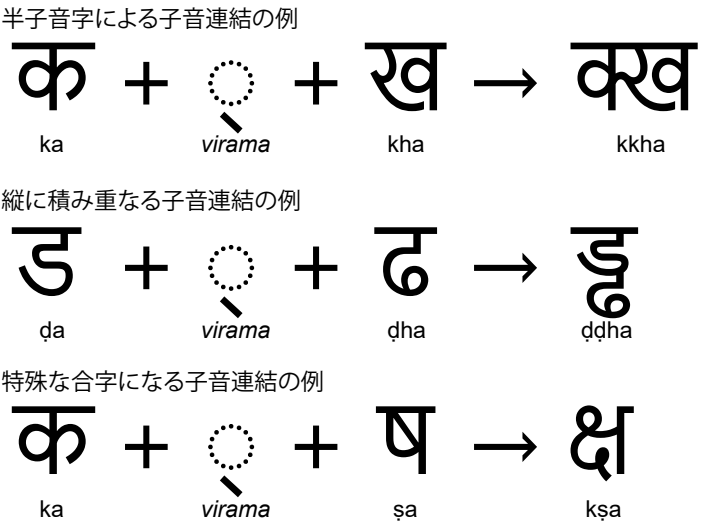


図 7. ヴィラーマモデルによるデーヴァナーガリーの子音連結の表現例

図 8. ビルマ文字の *asat*

ビルマ文字でヴィラーマに相当する記号は *asat*（အသတ် /ʔaṭaʔ/ アタッ）である（図 8）。しかし、これは母音・末子音・声調などを表すために他の文字と組み合わせて書かれ、母音がないことを示すヴィラーマとは少し性質が異なった使い方がされている。単語の綴りの中に含まれていて、後ろに文字が続くからといって書かれないということはない。

現在の Unicode では、*asat* には独立したコードポイントが与えられているが、Unicode 5.0 以前ではそうではなかった。これについては次の章でも説明する。

また、ビルマ文字では縦に子音字を重ねることがある。これは主にパーリ語などのインド系言語からの借用語にのみ現れ、借用元の言語で子音連続となっている部分を上下に重ねて書く。図 9 に例を挙げる。

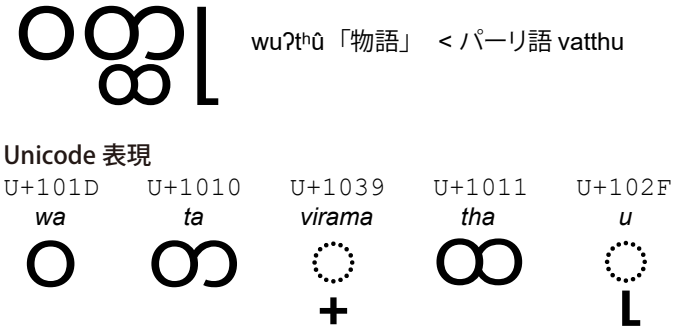


図 9. 子音字を縦に積み重ねる語の例

Unicode では、この重ね文字は、子音字等と子音字の間にヴィラーマを挟むことで表現される。この重ね文字については、文字の形を変えて積み重なるようにするための情報をフォントに含め、表示を行うプログラムからその情報に基づいて描画する必要がある。

Unicode でのミャンマー文字の扱い方については、Unicode が公開するFAQ[9] があったり、UTN #11 *Representing Myanmar in Unicode*[10] に詳細があるので、詳しくはそちらを参照されたい。

最後に、စင်္ကြံ zínjàn「歩道」と သင်္ဘော tɪ́nbɔ́「船」という例を図 10 に挙げてこの章を終わる。データは論理順に並んでいるが、表示順からは想像できないような順番になっているのがわかると思う。

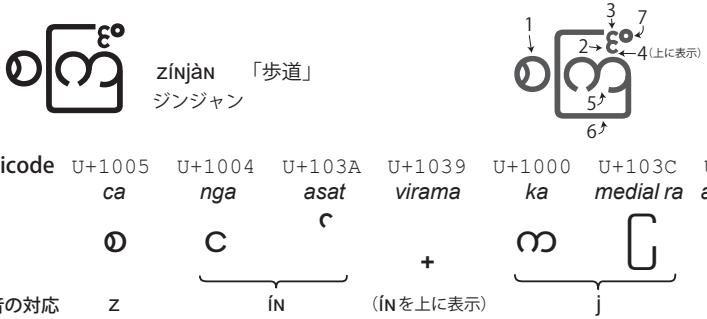


図 10. zínjàn「歩道」と ţínbó「船」の Unicode 表現

第 5 章

Unicode 5.0.0 と 5.1.0 の比較

ビルマ文字は Unicode 5.1.0 で大きな変更があってからは、互換性が壊れるような変更は行われず、安定していると思われる。この章では、バージョン 5.0.0 と 5.1.0 との間でのビルマ文字に関する差異を見ることで、Unicode のビルマ文字エンコーディングがどう変化したかを概観する。なお、ここではビルマ語以外のためにミャンマー文字ブロックに追加された文字については取り上げない。

基本的な変化としては、以前は特定のシーケンスをフォントで置換することで対応していた変化形の一部に対して独立したコードポイントが当てられ、よりプログラムやフォントの実装がやりやすくなっているように思われる。

5.1　*asat* への独立したコードポイントの割り当て

5.1.0 で *asat*（အသတ် /ʔəθaʔ/ アタッ）に独立したコードポイントが割り当てられた（`U+103A`）。それまでは *virama* `U+1039` + `ZWNJ` `U+200C` で

表していた[*11]。ZWNJ とは Zero Width Non-Joiner の略で、前後の文字に応じて形が変わる文字について、文字形状の変化を制御するために使われる。

5.1.1 例 1: ကော် /kɔ̀/

ကော် /kɔ̀/の Unicode 表現の変化が図 11 である。

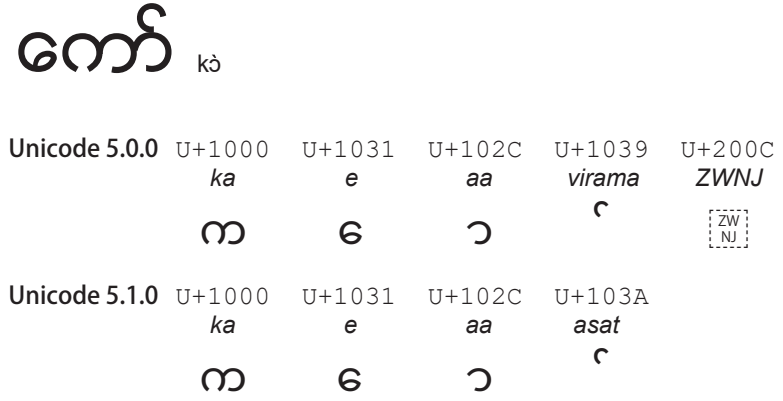


図 11. ကော် の Unicode 表現の変化

virama + ZWNJ で明示的にヴィラーマに相当する文字を表示するのはヴィラーマモデルでは典型的な挙動であり、デーヴァナーガリーなどではそうなっている。ただし、ミャンマー文字の *asat* は次に子音字が後続しようが常に表示されるものであり、母音・末子音・声調などを表すために非常に頻繁に用いられる。そのため、これが 1 つのコードポイントだけで表現できるようになったことで、かなりの効率化が図れたと思われる。

[*11] *The Unicode Standard, Version 5.0*[11] の p.380 参照。

5.1.2　例 2: ၎် （「イギリス人」 ဒဂၤလိပ် /ʔíŋgălei̯ʔ/ の一部など）

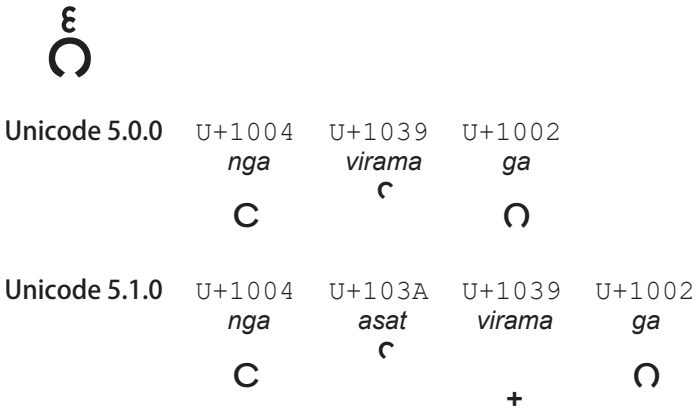


図 12. ၎် の Unicode 表現の変化

図 12 が ၎် の Unicode 表現の変化である。*asat* に独立したコードポイントが割り当てられた結果、この例に関してはコードの数が増えている。် が上に乗るのを *kinzi* というらしい[*12]。

5.2　介子音記号への独立したコードポイントの割り当て

介子音記号は、「*virama* + 子音字」の組合せから、個別のコードポイントが割り当てられるようになった（図 13）。

[*12] *kinzi* は最初入力方法がわからなかった。独立したキーを当てるとかしないと初見では入力無理なのでは…。

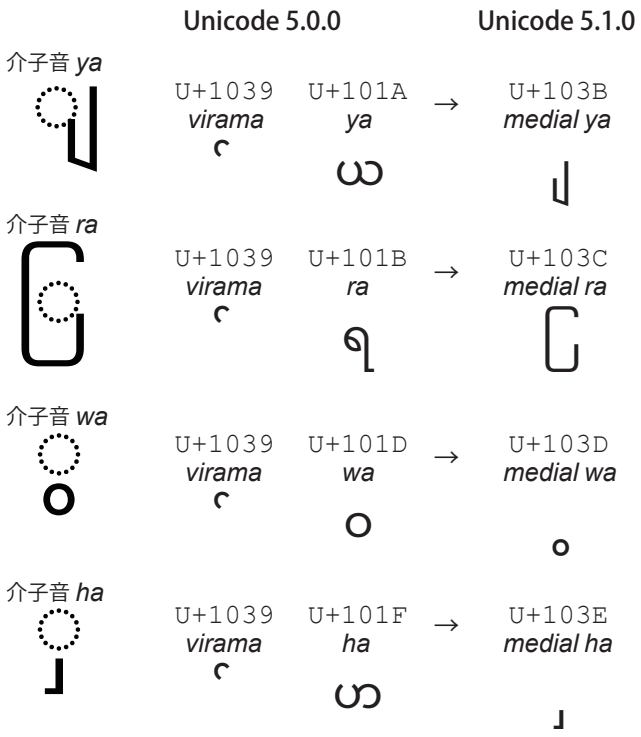


図 13. 介子音記号の Unicode 表現の変化

これにより、形状の置換を減らせるのでフォントの実装がシンプルになるとともに、入力する側としてもより直観的なデータ表現になったと思われる。

5.3 縦長の *aa* への独立したコードポイントの割り当て

ビルマ語では、縦長の *aa* (◌ါ) は、普通の丸い *aa* (◌ာ `U+102C`) が来ると紛らわしくなる場合に用いられる（例: ပါ /pà/ ↔ ပာ /h/を表す子音字）。*aa*

（長母音 a; ◌ာ U+102C ）の縦長版は直前の文字に応じてフォントで形を変えることになっていたが、独立したコードポイント（◌ါ U+102B ）が割り当てられた（図 14）。

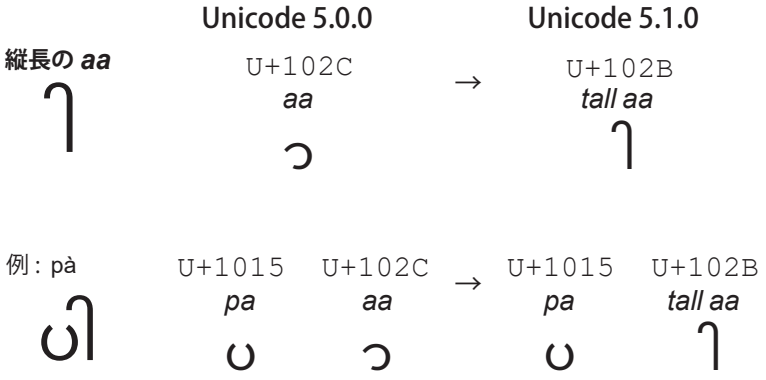
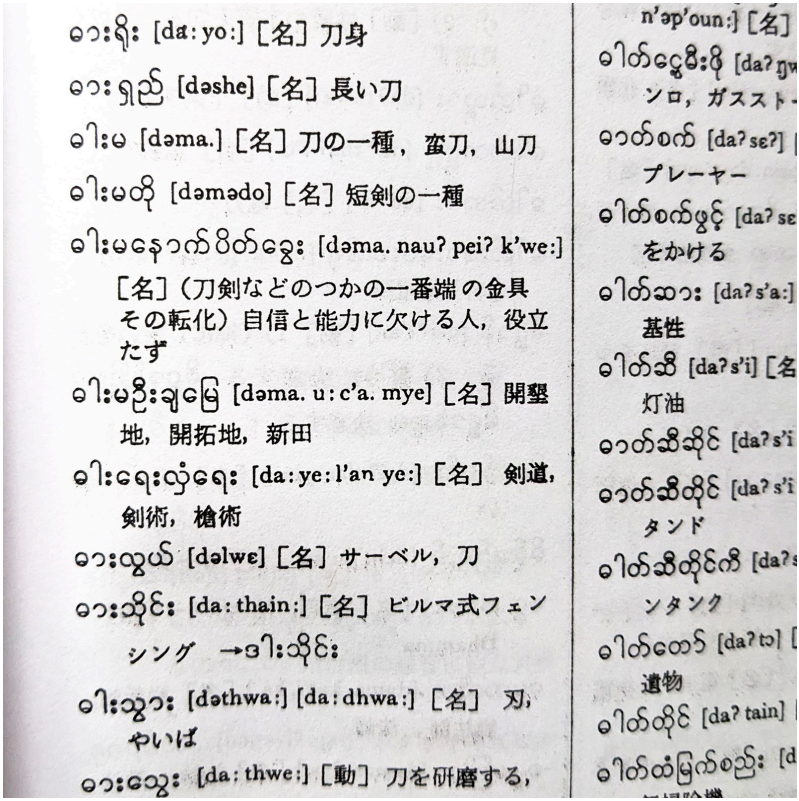


図 14. 縦長の *aa* の Unicode 表現の変化

この変更については、スゴー・カレン語（S’gaw Karen）の正書法においては *aa* は縦長の形しか用いられないため、それへの対応として分離されたようだ[12]。

とはいえ、ビルマ語で使い分けがどのようにされているのかは自明ではないかもしれない。ミャンマーで出版されたあるビルマ語辞書の ဓ *dha* の項目を見たところ、次に *aa* が来て ဓာ となるものと、*tall aa* が来て ဓါ となるものとが並んでいた（図 15）。

単に好みで形を選んでいるのか、何らかの区別を行っているのかは、私は知識がないのでわからないが…。この辞書を文字の形をそのまま電子化するには、古い方式では難しかったかもしれない。



ဓား:ရိုး: [da:yo:] [名] 刀身

ဓား:ရှည် [dəshe] [名] 長い刀

ဓါး:မ [dəma.] [名] 刀の一種，蛮刀，山刀

ဓါး:မတို [dəmədo] [名] 短剣の一種

ဓါး:မနော်က်ပိတ်ခွေး: [dəma. nau? pei? k'we:] [名]（刀剣などのつかの一番端の金具 その転化）自信と能力に欠ける人，役立たず

ဓါး:မဦး:ချင်မြေ [dəma. u:c'a. mye] [名] 開墾地，開拓地，新田

ဓါး:ရေး:လှံရေး: [da:ye:l'an ye:] [名] 剣道，剣術，槍術

ဓား:ထွယ် [dəlwɛ] [名] サーベル，刀

ဓား:သိုင်း: [da:thain:] [名] ビルマ式フェンシング →ဒါး:သိုင်း:

ဓါး:သွား: [dəthwa:] [da:dhwa:] [名] 刃，やいば

ဓား:သွေး: [da:thwe:] [動] 刀を研磨する，

n'əp'oun:] [名]

ဓါတ်ငွေဖိုး:ဝို့ [da?ŋw ンロ，ガスト

ဓာတ်စက် [da?sɛ?] プレーヤー

ဓါတ်စက်ဖွင့် [da?se をかける

ဓါတ်ဆား: [da?s'a:] 基性

ဓါတ်ဆီ [da?s'i] [名 灯油

ဓာတ်ဆီဆိုင် [da?s'i

ဓာတ်ဆီထိုင် [da?s'i タンド

ဓါတ်ဆီထိုင်ကီ [da?s ンタンク

ဓါတ်တော် [da?tɔ] 遺物

ဓါတ်တိုင် [da?tain]

ဓါတ်တံမြက်စည်း: [d

図 15. 辞書における ဓါ / ဓာ の例
ဦးကျော်နိုင် 編『ビルマ語辞典 မြန်မာ-ဂျပန် အဘိဓာန်』(နဒီမဂ််လာစာပေ 2004) p.221

5.4 *aforemention*ed の扱いの変更

ဌ U+104E *aforementioned* の扱いが変更された。以前は U+104E 単独で ဌင်း: lăgáuɴ という語全体を表していたが、5.1.0 では lăgáuɴ の最初の部分 ဌ だけになった。

この字は ဂၤ lăgáuN という語にしか使用されない[*13]ものの、lăgáuN の別表記で、 ဂ် （ ် が ဂ の上につく）という形に対応するためにこうなったらしい[*14]。

これにより、lăgáuN のエンコーディングが図 16 のように変更になった。

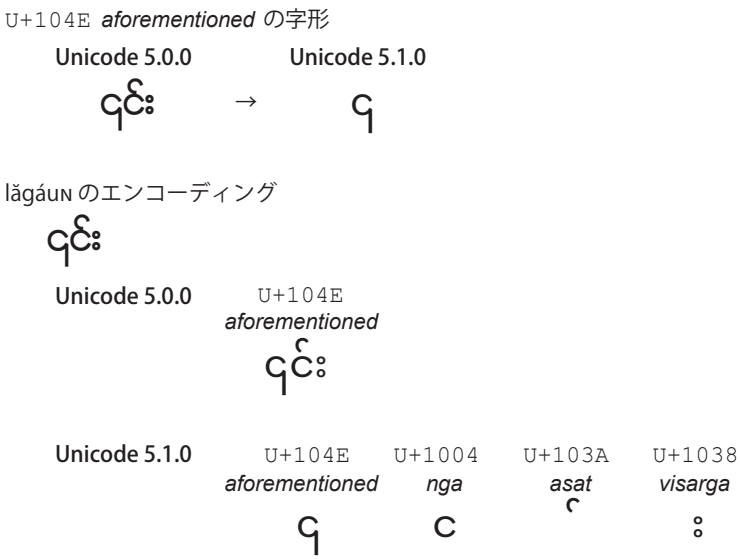


図 16. U+104E *aforementioned* の変更

[*13] ဎ U+104E のこの形は元々はビルマ数字の 4 “ ”に由来するが、これは数字の 4 の発音と lăgáuN の最初の発音が当時同じだったために、最初の部分を数字の 4 で代用した略記がなされ、広まったとのこと。発想は英語スラングの 2nite (= tonight), 4get (= forget) とかに近いようだ。

[*14] UTN #11 *Representing Myanmar in Unicode*[10] の p.11 参照。

5.5 *great sa* への独立したコードポイントの割り当て

ꩍ U+101E *sa* が 2 つ重なる場合は、代わりに *great sa* ꩍꩍ という特別な形が使われるのだが、これに独立したコードポイントが割り当てられた（図 17）。これに並行して、古いシーケンスは ꩍ *sa* が規則通り縦に重なる形（ꩍ꩓）に変更になっている。

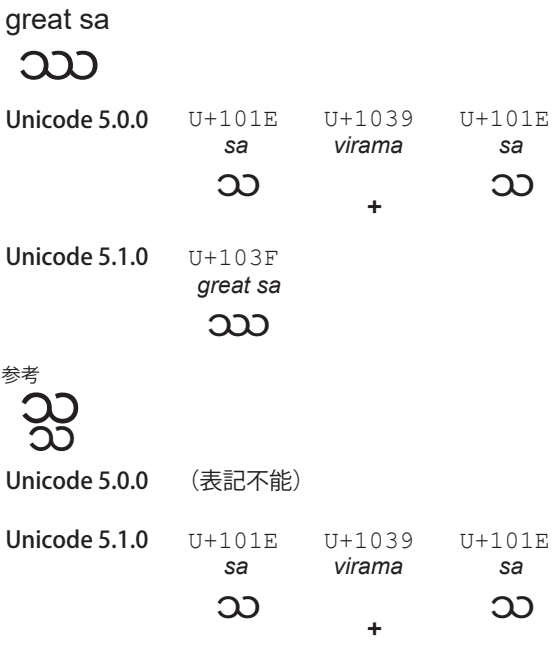


図 17. *great sa* の Unicode 表現の変更

第 6 章

Zawgyi のこれから

Zawgyi から Unicode に切り替えたとしても、過去の Zawgyi のテキストはそのままの形で残る。そのため、過去の資産を生かすためには、これからも Zawgyi を無視することはできない。しかし幸いなことに、文字コードの自動判定・変換ツールが存在している。

6.1 Zawgyi/Unicode 判定・変換プログラム

Facebook では、Zawgyi/Unicode を自動判定し、ユーザーの環境に応じて変換しているとのことだ[13]。同様の対応をしているサービスは他にもあるだろう。

Zawgyi/Unicode の判定、および変換をするプログラムやライブラリは複数ある。一例をあげると、Google が公開している次のものがある。

Myanmar Tools (Zawgyi detection & conversion)

`https://github.com/google/myanmar-tools/`

このプログラムでは機械学習を用いて判定を行っている。理由としては、人の手でルールを記述することで Zawgyi/Unicode 判定を行う場合では、本来の Unicode でシャン語やモン語などが書かれている場合を誤判定してし

まうことがあるためだとしている。

つまり、ビルマ語だけを判定するのでよければ、人の手でルールを記述すれば十分な精度が得られる場合はあるようである。母音の並び順、Unicode のビルマ文字以外の符号位置の利用などを見るだけでも、十分な長さがあればそれなりの精度で判定はできそうだ。とはいえ判定ルーチンを自分で書くことはまずないと思う。

また、変換に関しては変換ルールを書いていけばいいのだが、前に上げた zínjàn や t̰ínbɔ́ というような語など Zawgyi と Unicode 間の順番が大きく異なるのもあり、漏れなく変換ルールを書くのが難しい。

そのため、既存の実績のあるツールやライブラリを使うのがよいと思う。

6.2 Zawgyi のその他の利用法

Zawgyi は、ビルマ語に対応していないソフトウェアでビルマ文字を表示・印刷する場合に有用かもしれない。日本語組版ソフトなどではビルマ語に対応していないものも多いが、そのようなソフトでもビルマ文字をまともな見た目で表示できるからだ[*15]。

とはいえ Zawgyi は国際的な情報交換用としては不適なため、あまり誉められたものではないかもしれない。元データとしては Unicode で持っておき、ビルマ語非対応のソフトウェアでビルマ文字を扱う必要がある時は、Zawgyi-Unicode 変換器を使って Zawgyi に変換して表示するということは考えられる。

[*15] 本書の図を作る際に Adobe Illustrator CS6 を使ったが、このソフトはビルマ文字に対応していないため、ビルマ文字を表示するのに Zawgyi を利用した。

第 7 章

おわりに

Unicode に収録されたミャンマー文字をコンピュータで正しく表示することが困難だったために、その表示上の複雑さを回避する Zawgyi フォントが作られ、それがデファクトスタンダードとしてミャンマーで広まってしまった。そのため、グローバルスタンダードである Unicode への移行が困難になっていた。

この先 Unicode が普及していくとしても、過去の資産として Zawgyi で書かれたテキストデータが大量に存在するため、これを無視することができない。判別プログラムを使って判定し、必要に応じて変換を行うことが肝要だろう。

あとがき

この本は、私のブログ記事「Zawgyi と Unicode: ミャンマーの文字の電子化について - にせねこメモ」[*16]を本の形にしたものである。

表紙デザインは「ミャンマー文字」とビルマ語で書いたものと、裏表紙は Zawgyi テキストを Unicode フォントで表示させ文字化けさせたものである。

[*16] https://nixeneko.hatenablog.com/entry/2023/12/19/210000

参考文献

[1] Wikipedia：「ミャンマーの国名 — Wikipedia, the free encyclopedia」, https://ja.wikipedia.org/wiki/ミャンマーの国名（2024）[2024 年 5 月 18 日閲覧].

[2] 加藤 昌彦：「ニューエクスプレスプラスビルマ語」，白水社（2019）

[3] Wikipedia: “MLC Transcription System — Wikipedia, the free encyclopedia”, https://en.wikipedia.org/wiki/MLC_Transcription_System (2024) [2024 年 5 月 19 日閲覧].

[4] G. Hotchkiss: “Battle of the fonts | Frontier Myanmar”, https://www.frontiermyanmar.net/en/battle-of-the-fonts/ (2016) [2024 年 5 月 20 日閲覧].

[5] 後藤 修身：「ビルマ語(ミャンマー語)を windows で〜unicode 以前 | エヤワディ blog」，https://www.ayeyarwady.com/blog/archives/240（2009）[2024 年 5 月 18 日閲覧].

[6] Unicode Consortium: *The Unicode Standard Version 15.0 - Core Specification*, Chap.2.2 Unicode Design Principles, pp.14–24, https://www.unicode.org/versions/Unicode15.0.0/ch02.pdf (2022)

[7] Wikipedia: “Virama — Wikipedia, the free encyclopedia”, https://en.wikipedia.org/wiki/Virama (2024) [2024 年 5 月 18 日閲覧].

[8] 加藤 弘一：「『電脳社会の日本語』：ほら貝 — 結合音節文字の

編集」, http://www.horagai.com/www/moji/nihon/hosetu4.htm（2000）[2024 年 5 月 18 日閲覧].

[9] Unicode Consortium: “FAQ - Myanmar Scripts and Languages”, https://www.unicode.org/faq/myanmar.html (発行年不明) [2024 年 5 月 18 日閲覧].

[10] M. Hosken: *Unicode Technical Note 11 - Representing Myanmar in Unicode: Details and Examples; Version 4*, https://www.unicode.org/notes/tn11/UTN11_4.pdf (2012)

[11] Unicode Consortium: *The Unicode Standard, Version 5.0*, Chap.11.3 Myanmar, pp.379–381, https://www.unicode.org/versions/Unicode5.0.0/ch11.pdf (2006)

[12] Unicode Consortium: “Unicode 5.1.0”, https://www.unicode.org/versions/Unicode5.1.0/#Myanmar (2008)

[13] 後藤 修身：「ミャンマー語（ビルマ語）のフォントが zawgyi から unicode へ大改革 - enjoy yangon ヤンゴン, ミャンマーで暮らす旅する」, https://enjoy-yangon.com/ja/enyanblog/351-change-myanmar-font-zawgyi-to-unicode（2019）[2024 年 5 月 18 日閲覧].

奥付

書名	Zawgyi と Unicode: 普及しすぎたミャンマーのオレオレ文字コードと国際化
発行	ヒュアリニオス
著者	にせねこ (@nixeneko)
発行日	2024 年 5 月 26 日（技術書典 16）
電子版発行日	2024 年 5 月 25 日
連絡先	nixeneko.info@gmail.com
サポートページ	http://hyalinios.hatenadiary.com/entry/tbf16-zawgyi

© nixeneko 2024

本誌はクリエイティブ・コモンズ表示 4.0 国際ライセンスの下に提供されています。

https://creativecommons.org/licenses/by/4.0/deed.ja

မြန်မာ
အက္ခရာ