Modulo Class Ordering of Fidel

by Daniel Yacob

Yet another way to encode the Fidel matrix that is a unrestrained by tradition, operating systems, or economics. But is bound rigidly by, and exploits fully, a simple mathematic property woven into the Fidel lattice. We have all seen it, sometimes thought about it, but before computer interest came along it was not worth the effort to fully unravel this hidden level of order within Fidel.

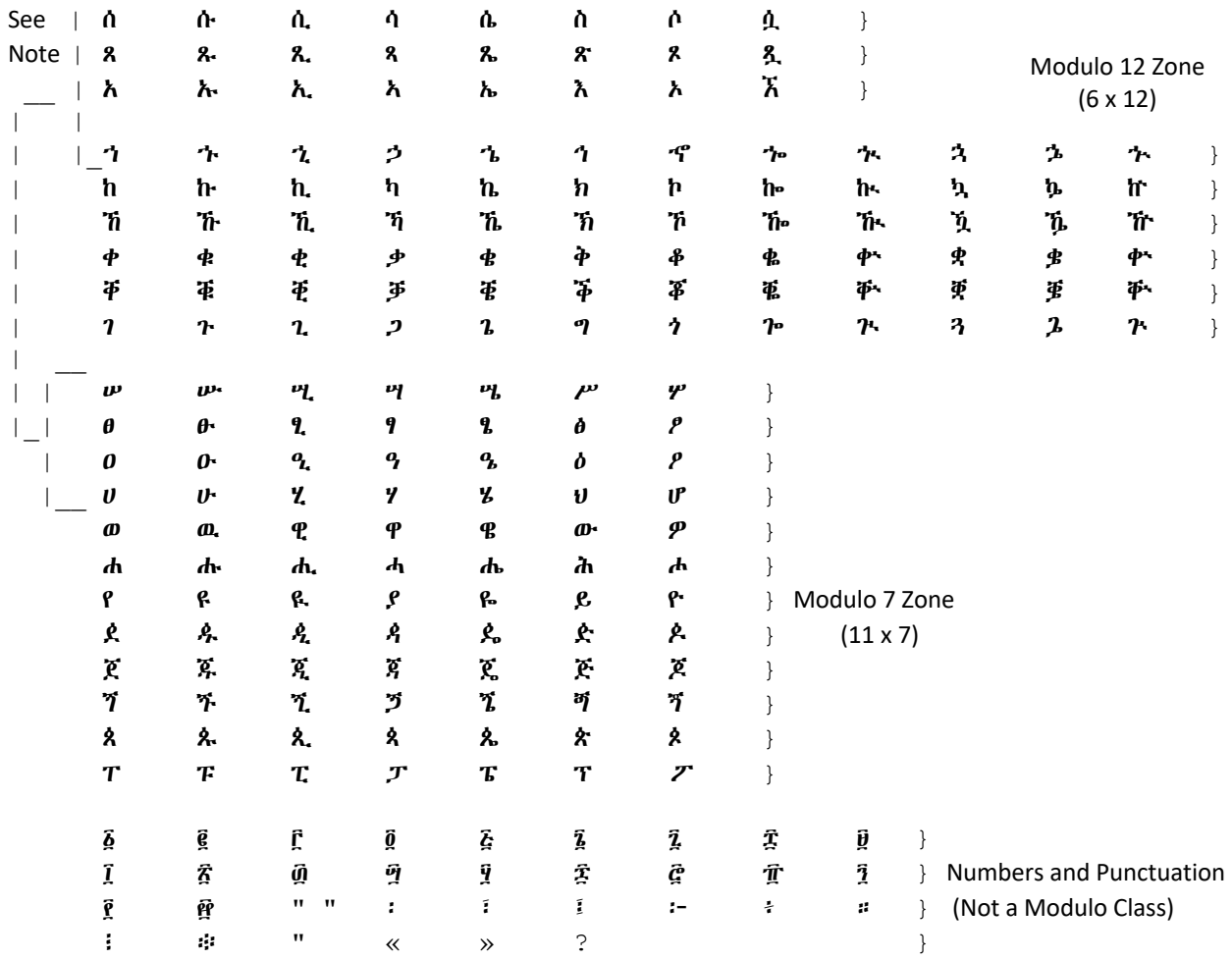
At the beginning the Fidel writing system (WS) was well ordered, the first kernel had 26 letters of 7 forms each. Before long however user requirements changed as users changed and the customer base expanded. Script hackers of the time patched auxiliary letters onto the end of the matrix. This was not too bad, the 26x7 matrix remained intact 8th forms and greater were at the end. Later as add-ons for Amharic reached 7 new series and extended orders grew as well, these modules were finally integrated into the kernel of new release for the WS. The same cycle has been repeated for extension made for Tigrigna (ቐ), Oromiffa (ዸ), and Bilin, Agew, and Gurage languages (ኻ, ጐ, etc.).

So, what was once a simple and regular system for a single language became the necessary patchwork quilt of systems that would serve all languages that would use Fidel. Since from speech we know the sounds in the language already, it is for most characters not a big deal to remember who has 7, 8, or 12 forms. When the problem is encountered in the composition of a new computer application, the programmer soon finds what he thought was natural is now a major issue to resolve. The problem boils down to *"if a computer can determine a letter is "ሪ" -how will it know it know if there is a ሯ?"*. Recognizing the letter as "**ሪ**" in the first place, or that "**ሪ**" is the 3rd form of "**ር**" is no simple matter either.

These issues become important when you want to transcribe or transliterate a Fidel document into another system or syllable-edit of a Fidel character. Below is a matrix that simplifies the issue of character form recognition.

Fidel-95NT WS/2

ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ	ሏ	}
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ	ሟ	}
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ	ሯ	}
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ	ሿ	}
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ	ቧ	}
ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ	ቯ	}
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ	ቷ	}
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ	ቿ	}
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ	ኗ	} Modulo 8 Zone
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ	ኟ	} (19 x 8)
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ	ዟ	}
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ	ዧ	}
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ	ዷ	}
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ	ጧ	}
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ	ጯ	}
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ	ፏ	}



It should be fairly obvious what has happened. All letters with like number of forms are grouped together. The number of orders a letter has is its “modulo class”. For numbers and punctuation there is not as big a payoff later to putting them into as organized a group. It is convenient that numbers are together as it is for punctuation (also that all of the tens are equidistant from the ones).

Exploiting the Modulo Class Regions

Computer work with letters in the form of numbers, actually the work with everything in the form of numbers. Unlike us, computers don't have real eyes ears and other senses to know the difference between two things. So, everything gets converted to numbers which the computer can compare easily. So, since we are talking about “letters” we need a way to talk about letters as numbers.

Let’s use [ ] to mean “the address of” whatever is inside of [and]. For instance, [ሀ] would be 0 (computer types like to start with zero) and [me] would mean 8 using the table above. Also, let’s use ‘x’ to be any letter in the Fidel matrix. So, [x] is 0 if x = ሀ, and [x] = 8 if we wanted x = መ. For discussion it is convenient just to use ‘x’ without worrying about its true identity.

Given the Fidel table above we can make it easy to tell a computer how to figure out how many syllables the letter ‘x’ will have. We can use “greater than” and “less than” rules to do this:

```
IF   [ሀ]  <=  [x]  <  [ሀ]  THEN  'x' has  8 forms
IF   [ሀ]  <=  [x]  <  [ሠ]  THEN  'x' has 12 forms
IF   [ሠ]  <=  [x]  <  [፩]  THEN  'x' has  7 forms
```

Other things we can find out about ‘x’ from where it resides in the table (what we have been calling its “address”) are:

```
IF   [ሰ]   <=  [x] < [ጓ]   THEN  'x' has a homophonic cousin a fixed
                                     distance away.
IF   [፩]   <=  [x]         THEN  'x' is not a letter
IF   [፩]   <=  [x] < [" "] THEN  'x' is a number
IF   [" "] <=  [x]         THEN  'x' is a punctuation
```

So, when is it important to know how many forms a letter has? One example is for an editing feature to change only the order of a letter and not the whole letter. Let’s say the cursor is over ሬ and you want to change it to ሪ. This can be done by typing only ‘`i`’ or maybe Control-‘`i`’ without complicated math. A sample calculation:

> Let ‘x’ be any 5th order letter that has 7 forms and we want to adjust to the 3rd form.
>
> ```
> new_order = 'i' (ግልስ) = 2          (remember we start at zero!)
>
> IF  [ሠ] <= [x] < [፩]  THEN  forms = 7
>
> order = ( [x] - [ሠ] ) % forms = 4
>
> 'x' = 'x' + (new_order - order)
>     = 'x' + (2 - 4)
> ```

Here, the `%` symbol is the modulus operator which is available in some form in any programming or macro language (sometimes given the name “mod”). These few lines are enough to adjust any 5th order letter to the 2nd -and the beauty is that we didn't even have to know what letter it was!

If we did want to know what 'x' was a way to find out would be:

```
IF  [ሠ] <= [x] < [፩]  THEN  forms = 7

order = ( [x] - [ሠ] ) % forms = 4

iam = [x] - order
```

This guarantees that ‘`iam`’ will be a first (or ግዕዝ) form letter. The ‘`iam`’ variable will have to be checked in a table to see that it might be “ዩ” for instance. The advantage is that our lookup table has only 33 letters or so instead of more than 300. This saves memory and searching time. We could keep in this table only the Roman letter ‘y’ [ዩ] . Since we know ‘order’ a second table can tell us: order = 1 = ‘e’ , order = 2 = ‘u’ , etc. Then transcription is made simple by printing Table1[iam] and Table2[order] .

Technical

The modulo class ordering of Fidel was created out of necessity for a SERA transcriber for GohaTibeb a DOS WordPerfect add on package for Fidel. The WP macro language had limited logic capabilities and so Fidel had to be ordered logically. Another consideration for WP was that address space was minimal and so no “gaps” could be afforded within the matrix. Mule uses the same ordering system (a modification of the above) and it is found in Admas X-11 fonts.

An obvious flaw in the above ordering is that it makes for an unnatural sorting order. Sorters usually rely on character addresses. The modulo ordering is then *NOT* recommended as an address encoding system for Fidel. The merit of the ordering is purely its algorithmic facilitations and as such should be used only for *internal* modeling of Fidel by applications.