HornMorpho 2.0 User's Guide

Michael Gasser
Indiana University, School of Informatics and Computing
`gasser@cs.indiana.edu`

18 June, 2010

1. Introduction

(For a very quick introduction to HornMorpho, see the Quick Reference that is included with the distribution as `horn2.0_quick.pdf`.)

HornMorpho is a Python program that analyzes Amharic, Oromo, and Tigrinya words into their constituent morphemes (meaningful parts) and generates words, given a root or stem and a representation of the word's grammatical structure. It is part of the L^3 project at Indiana University <`http://www.cs.indiana.edu/~gasser/Research/projects.html`>, which is dedicated to developing computational tools for under-resources languages.

Natural language applications, such as question-answering, speech recognition, information retrieval, and machine translation, rely on a lexicon of possible forms in the relevant language. Morphological analysis is important for morphologically complex languages like Amharic, Oromo, and Tigrinya because it is practically impossible to store all possible words in a lexicon, and many words have close to 0 probability of occurrence in any given corpus. This becomes obvious in the context of machine translation to a morphologically simple language such as English, where the correspondence between words in Amharic, Oromo, or Tigrinya and the other language will often be many-to-one. The Amharic word **ባይከፈትላቸውም**, for example (which incidentally does occur in an online corpus), could be translated as 'even if it isn't opened for them'. While a system for processing English could include all of the English words in the translation (*even*, *if*, *it*, *isn't*, *opened*, *for*, *them*) in its lexicon, an Amharic system that includes all words such as **ባይከፈትላቸውም** is clearly impractical. For translation *into* Amharic, Oromo, or Tigrinya and sophisticated question-answering, morphological generation is also desirable because it is probably impossible to store all of the words that the system will output. Finally, for Semitic languages such as Amharic and Tigrinya, morphological analysis can make explicit some of the phonological features of the languages that are not reflected in the orthography; these features may be important for text-to-speech applications. For example, the Tigrinya word **ዚፍለጥ** 'which (it) is known' is correctly pronounced with gemination (lengthening) of the third consonant: *zifIlleT.* The gemination in this case is grammatical, and a morphological analyzer can infer it based on its knowledge of Tigrinya verb roots and the particular templates that they occur with (see Section 5 for more on this aspect of Amharic and Tigrinya gramar).

HORNMORPHO requires Python 3.0 or 3.1; it will not run under Python 2.*.[1] In order to use the program for Amharic and Tigrinya, you must also have a Unicode Ge'ez (Ethiopic) font such as Ethiopia Jiret or GF Zemen[2] on your computer. In addition, if you want to use the program to analyze particular words in these languages, rather than (or in addition to) Amharic or Tigrinya text in a file, you will need a way to enter Ge'ez characters. HORNMORPHO has been tested under Windows, Linux, and MacOS. Some of the options are not possible with Windows because of issues related to the display of non-roman characters (see the discussion in Section 8.b), but the main functionality of the program applies across all three platforms. Since Oromo is written in the roman alphabet, you will not have to worry about these font- and keyboard-related issues if you only want to use the program for Oromo.

HORNMORPHO is a work in progress, and it is certain to have bugs, as well as the limitations described in the last section of this document and features that make it awkward to use. Questions, comments, corrections, and suggestions (to <`gasser@cs.indiana.edu`>) are very welcome. The main goal of the project is to make the program useful to the widest possible group of users.

Morphological processing relies on a lexicon of roots or stems, and none of this work would have been possible without the help of researchers who have made their dictionaries available to the research community or to me. For Amharic, I am indebted to Amsalu Aklilu for his excellent Amharic-English dictionary and to Daniel Yaqob for making this work available digitally. For Tigrinya, I thank Efrem Zacarias for his continuing work in putting together an online Tigrinya-English dictionary and for his permission to use his Tigrinya word list. For Oromo, I am grateful to Prof. Catherine Griefenow-Mewis and Tamene Bitima for making available digitized versions of two excellent Oromo-English dictionaries, those of Gragg (1982) and of Tamene himself (2000).

In the rest of this document, it is assumed that you have at least a basic knowledge of Amharic, Oromo, and/or Tigrinya and basic grammatical terminology. No familiarity with Python is assumed.

2. Installation

If you haven't already done it, uncompress the file that you downloaded. This will yield a directory (folder) called `HornMorpho-2.0`, which contains all of the files that you need to run HORNMORPHO.

The program is most useful if you install it on your computer; this places the program files in the place where other third-party Python programs reside on your system so that you can use them wherever you are running Python. If you are on a system with multiple users and separate adminis-

[1] To download Python to your computer, go to <`http://www.python.org/download/`>

[2] Downloadable at <`http://www.senamirmir.com/download/jiret.zip`> and <`ftp://ftp.ethiopic.org/pub/fonts/TrueType/gfzemenu.ttf`> respectively.

trative privileges, you will need these privileges to install the program, or you can ask a system administrator to do it for you.

To install HornMorpho, you will need to open a shell (often called “command prompt window” in Windows).[3] In the shell, go to the `HornMorpho-2.0` directory (folder), and enter the following if you are on a Unix or Unix-like system, making sure that you are running Python 3.0 or 3.1.

```
    python setup.py install
```

If you are using Windows, it will probably suffice to enter:

```
    setup.py install
```

To test whether the installation succeeded, start up the Python interpreter, again making sure that you are running at least Python 3.0 (see Section 8.a if you don’t know how to do this), and type

```
    import l3
```

If you don’t want to install the program, you can still use it. Just move the whole directory to a convenient place in your file system, and then make sure you run the Python interpreter from the `HornMorpho-2.0` directory, wherever that is. See Section 8.a for details.

The next section describes the conventions used in this document and in HornMorpho for representing Amharic and Tigrinya words using roman characters. Section 4 discusses the theory behind the program. If you just want to use the program and are not interested in the theoretical background, you can skip that section and go straight to Sections 5 and 6, which are concerned with aspects of Amharic, Oromo, and Tigrinya morphology and phonology that are implemented in the program. Section 7 deals with implementation details, in particular, the format of the data files; you can skip this section if you don’t intend to look at any of these files. Section 8 is the most important section for most users; it describes the functions available in the program. If you are mainly interested in finding out what the program can do, go straight to Section 8. Section 9 lists some of the limitations of the current version of the program.

3. Conventions

In the rest of the document, Amharic and Tigrinya words and morphemes will sometimes be written in Ge’ez characters and sometimes romanized. The romanization follows the conventions of the SERA system (Yitna and Yacob, 1996), with the following exceptions.

1. **Gemination** (consonant lengthening; **ጥብቀት**) is not normally indicated in the Ge’ez script. However, since it plays a significant role in the morphology of both languages and is important for speech applications, it is shown in the romanized forms of words, both in this document and within HornMorpho. The underscore character following the consonant indicates gemination: *teTeq_emeb_et* (**ተጠቀመበት**).

2. Ge’ez script does not distinguish between consonants that are not followed by a vowel and consonants that are followed by the high central vowel (phonetically [ɨ], sometimes

[3] How you open a shell depends on your operating system. In Unix or Unix-like systems, including MacOS, you run a terminal application (in MacOS this is found in Applications/Utilities). On computers running Windows, you need to run “Command Prompt” in Vista, “cmd.exe” in other versions.

also represented as [ə]) that the language uses to break up clusters of consonants, known in linguistics as an “epenthetic” vowel (Amharic **ሰርጎ-ገብ**, Baye, 2000 E.C.). Both are represented by the sixth order (**ሳድስ**) character in a series. Again because this distinction matters for speech applications, it is usually made here, with the character *I* used for this vowel: *zIfIS_Im* (**ዝፍጽም**).

3. SERA uses two-character sequences to represent three consonants: *`s* (**ሠ**, etc.), *`h* (**ኀ**, etc.), *`S* (**ፀ**, etc.). However, the backquote is also used for the consonant in the **ዐ** series in Tigrinya. To avoid confusion between sequences such as **ዕስ** and **ሥ**, the backquote is here replaced by the caret (^), for example, in *^ser_a* (**ሠራ**).

4. All bare occurrences of vowels in Amharic are preceded by either an apostrophe (representing the **አ** series) or a backquote (representing the **ዐ** series). This is a more direct representation of Ge’ez orthography and agrees with Tigrinya, where these two series always represent consonants plus vowels. For example, *‘IbEtu ‘al_e* (**እቤቱ አለ**).

5. Tigrinya has a vowel, phonetically [ɛ] or [æ], in addition to those represented by the seven Ge’ez orders, that is of somewhat uncertain status. It appears after the laryngeal consonants *’*, *`*, *h*, *H*, where we would otherwise expect the vowel *e*, and is sometimes written with the first order (**ግዕዝ**) character, sometimes with the fifth order (**ሓምስ**) character, for example, **በጽሐ** / **በጽሔ**, **መጺአን** / **መጺኤን**. In HORNMORPHO, this Tigrinya vowel is represented by @: *beSH@*, *meSi’@n*.

Other than the cases noted in 3 and 4, HORNMORPHO uses the SERA conventions for Amharic and Tigrinya consonants: **ሀ** *ha*, **ለ** *le*, **ሐ** *Ha*, **መ** *me*, **ሠ** *^se*, **ረ** *re*, **ሰ** *se*, **ሸ** *xe*, **ቀ** *qe*, **ቐ** *Qe,* **በ** *be*, **ተ** *te*, **ቸ** *ce*, **ኀ** *^he*, **ነ** *ne*, **ኘ** *Ne*, **አ** *'a*, **ከ** *ke*, **ኸ** *Ke*, **ወ** *we*, **ዐ** *`a*, **ዘ** *ze*, **ዠ** *Ze*, **የ** *ye*, **ደ** *de*, **ጀ** *je*, **ገ** *ge*, **ጠ** *Te*, **ጨ** *Ce*, **ጰ** *Pe*, **ጸ** *Se*, **ፀ** *^Se*, **ፈ** *fe*, **ፐ** *pe*. Labialized consonants are represented with a following *W*, for example, **ቈ** *qWe*, **ቧ** *bWa*. The vowels, in traditional Ge’ez order are *e*, *u*, *i*, *a*, *E*, (*I*), *o*.

A few other conventions apply to the representation of verb roots; these are discussed below.

4. Theory

a. FINITE STATE MORPHOLOGY

When analyzing words, HORNMORPHO takes as input an Oromo word in the usual roman orthography (qubee) or an Amharic or Tigrinya word in Ge’ez script. If Ge’ez, the word is first romanized, using the variant of the SERA romanization system described above. Next the program checks to determine whether the word is stored in a list of unanalyzable or pre-analyzed words. If not, it attempts to perform a morphological analysis of the word. It first does this using a “lexical” approach, based on a built-in lexicon of roots or stems and its knowledge of Amharic, Oromo, or Tigrinya morphology. If this fails, it tries to guess what the structure of the word is, using only its knowledge of morphology and the general structure of roots or stems. If the “guesser” analyzer fails, the program gives up.

Both the lexical and guesser analyzers operate within the general framework known as **finite state morphology**.[4] Morphology (and phonology) is traditionally viewed within linguistics as the relationship between a **surface** form and a corresponding **lexical** (or underlying) form. For example, the surface Amharic form **ከቤታችን** can be analyzed into the sequence of three morphemes **ከ**+**ቤት**+**አችን** or, more abstractly, into a lexical representation that makes the structure of the word more explicit:

(1) `[stem=ቤት, possessor=[person=1, number=plural], preposition=ke]`

In ordinary English, this says that the word **ከቤታችን** is derived from the noun stem **ቤት**, with a first person plural possessor ('our') and the preposition **ከ**. The knowledge that an Amharic noun stem can be preceded by a preposition and followed by a possessive suffix is part of Amharic **morphotactics**.

Finite state morphology makes use of **finite state transducers (FST)**, computational devices that convert (transduce) one string into another. For example, one stage in the analysis of the Tigrinya word **ሰተና** *setena* 'we drank' is the conversion of the second vowel *e* into the sequence *ey,* revealing the third root consonant *y* of the verb whose complete root is *sty*. Expressed in romanized form:

(2) *setena* → *seteyna* (analysis)

The arrow represents the direction of **analysis**, from a form that is closer to the surface to a form that is closer to the lexical representation. The reverse direction is that of **generation**, from a more abstract to a less abstract form (closer to the surface). Seen from the perspective of generation, (2) represents the merging of the *ey* sequence to the single vowel *e*. This is an example of a **phonological rule**: it combines two vowels to make the word conform to the constraints of Tigrinya phonology. It is also an **orthographic rule** because the resulting vowel also appears in the conventional spelling of the word. Rules of both types are known within the finite state morphology community as **alternation rules** because they reflect alternations in the form of morphemes when they come together.

Consider an Oromo example. The word *argita* 'you (sing.) see' is based on the verb root *arg-* 'see' and the suffix *-ta* 'you (sing., present tense)'. Because Oromo does not permit three consonants in succession, there is a phonological rule that inserts an *i* to break up the sequence *rgt* that would otherwise occur in this word. In the generation direction, the rule affects the word in this way:

(3) *arg+ta* → *argta* → *argita* (generation)

One complication that arises when we think of morphological analysis and generation in terms of a set of alternation rules is that the rules often have to apply in a particular order. For example, there is another Oromo rule that converts the consonant *t* to *d* when it follows *b*, *d*, or *g*. Thus we have *dhugda* 'you (sing.) drink':

(4) *dhug+ta* → *dhugta* → *dhugda* (generation)

Consider the order of these two Oromo rules in the generation direction, assuming that the input is *arg+ta*. If the *i* insertion rule precedes the *t→d* rule, then we will get the correct form *argita* because the *t→d* rule will fail to apply. But with the reverse order, we will get the wrong form *argida* because the *t* will be changed to *d* before the *i* is inserted. Thus the order of the rules matters.

[4] For a more in-depth introduction to finite state morphology, see Beesley & Karttunen (2003), and for another example of the application of finite state technology to Amharic morphology, see Saba & Girma (2006).

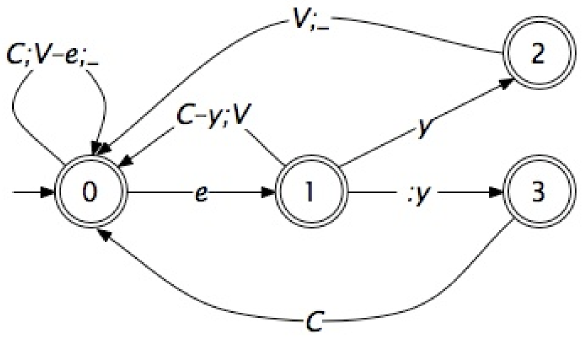

Figure 1. FST for converting surface *e* to lexical *ey* in Tigrinya.

(5a) 1. *i* insertion rule: *argta → argita*
2. *t→d* rule: *argita* (rule fails to apply)
(5b) 1. *t→d* rule: *argta → argda*
2. *i* insertion rule: *argda → argida*

An FST consists of a set of **states**, joined by **transitions**, with one state designated the **initial state**. Each transition has a condition on it, consisting of an input and an output character, either of which may be zero (the empty character). To transduce an input string into an output string, we start in a start state and then, as long as it is possible, move from one state to the next by replacing an input character with an output character. That is, as the FST "consumes" the characters in the input string, one by one, it adds characters to a resulting output string. If we reach the end of the input string in one of the designated **final states** of the FST, the transduction has succeeded and the resulting output string is returned.

For example, consider a simple FST that is responsible for the optional merging of the sequence *ey* into the single vowel *e* in a Tigrinya word like **ሰተና** *setena*, shown in Figure 1 above.[5] In the figure, states are represented by circles and transitions by arrows. Input and output character pairs are separated by colons and multiple conditions on the same transition are separated by semicolons. A single character on a transition represents the same input and output character, and the absence of a character on either side of a colon represents an empty character. The character *C* represents any consonant, and *C* alone represents the same input and output consonant. *C–y* represents any consonant but *y*. The initial state is 0, and the circles with double borders (in this case, all states) represent final states.

The FST in the figure applies to the analysis direction. Given the input string *seteye*, this FST outputs the same string because it goes eventually from state 0 to state 1 to state 2, then back to state 0, and none of the transitions changes the input character to a different output character. Given the input string *setena*, there are two possibilities. Either it outputs the same string, this time staying in states 0 and 1, or it outputs the string *seteyna*, passing through states 1 and 3 between the *t* and *n*, as shown in (6).

(6) `0 s:s 0 e:e 1 t:t 0 e:e 1 :y 3 n:n 0 a:a 0`

Note also that the FST fails on the input string *seteyna*; after state 2 is reached following the *ey*, there is no state that can be reached with an input *n*.

[5] The rule shown in the figure is an oversimplification in several ways of what happens in the language and what is implemented in HORNMORPHO.

Among the great achievements of the theorists of finite state morphology (Johnson, 1972; Kaplan & Kay, 1994; Karttunen et al., 1992; Koskenniemi, 1983) are the following results.

- All of the normal alternation rules known from natural languages, as well as morphotactics, can be captured by FSTs; that is, more complex devices requiring stacks, for example, are not necessary. Furthermore, since FSTs are invertible, an FST for analysis can be trivially converted into one for generation.
- A lexicon of roots or stems can be incorporated into a morphotactic FST.
- A series of FSTs, for example, implementing a sequence of ordered alternation rules and the morphotactics for a particular word, can be **composed** into a single FST which behaves exactly like the series of FSTs when applied to an input in the order that they are composed.

The upshot of these three facts is that it is possible to build a single FST that handles all aspects of the morphology for a particular class of words in a language, for example, an FST that analyzes all Amharic nouns.

A **deterministic** FST is one for which there is only one possible next state for a given unconsumed input string. For a given input string, a deterministic FST yields either nothing, if it fails on the string, or a single output string. In general, words in natural languages may be ambiguous; that is, a given word may have more than one legitimate analysis. Consider the Amharic orthographic word **ብትሰማ**. This has at least four possible analyses, corresponding to the English translations 'if you hear', 'if you are heard', 'if she hears', and 'if she is heard'. For most applications, we would like a morphological analyzer to return all of the possible analyses. Therefore the FSTs that HORNMORPHO uses are **non-deterministic**; for a given input string and FST, there may be multiple successful paths through the FST, each corresponding to a different grammatical analysis of the string.

b. HANDLING NON-CONCATENATIVE MORPHOLOGY

Semitic languages like Amharic and Tigrinya, in particular their verbs, present a problem for finite state morphology, a problem that is well-known in the literature. Finite state morphology, without the introduction of special-purpose mechanisms, becomes very inefficient for **non-concatenative morphology**, that is, any case in which words don't consist simply of sequences of morphemes or morphemes don't consist simply of sequences of characters.[6] Amharic and Tigrinya morphology is non-concatenative in at least two ways.

First, there are discontinuous morphemes (so-called circumfixes), which are separated by other morphemes, for example, the morpheme that indicates negation in non-subordinate Tigrinya verbs: **አይፈለጠን** ***ay***-*feleTe*-***n***. A related problem is that the morphotactics of the end of the verb depends on what occurs at the beginning. For example, the Tigrinya relative prefix **ዝ** preceding the negative prefix normally precludes the negative suffix **ን**: **ዝይፈለጠ** ***zI-ay***-*feleTe*.

Second, as in other Semitic languages, Amharic and Tigrinya verb stems have a so-called **root-template** structure. Consider these Amharic verbs, all based on the verb root meaning 'repeat':

(7) **ይደግማል** *yIdegmal*

[6] For a more complete discussion of non-concatenative finite state morphology, see Cohen-Sygal & Wintner (2006).

ይደገማል *yId_eg_emal*
ያስደግማል *yasdeg_Imal*
ይደጋገማል *yIdegag_Imal*

Each of these verbs has the prefix and suffix that indicate a verb in the imperfective tense/aspect (corresponding roughly to English present tense) with a third person singular masculine subject (‘he’ or ‘it’). What is left with the prefix and suffix removed is the verb **stem** (Amharic **አምድ**, Tigrinya **ዓምዲ**). Each of these stems is composed of two parts, the **root** (Amharic **ስር**, Tigrinya **ሱር ግሲ**), consisting of the consonant sequence *dgm*, and the **template**, consisting of a pattern of vowels and in some cases a prefix or the gemination of one or more consonants (the gemination is of course not indicated in Amharic orthography). For example, the template in the first case consists of the vowel *e* between the first and second root consonants and no vowel between the second and third consonants, and the template in the second case consists of the vowel *e* in both positions and the gemination of the first and second consonants. We can represent the templates using C_1, etc. to represent the stem consonants.

(8)	*dgm* + $C_1eC_2C_3$	→	*degm*
	dgm + $C_1C_1eC_2C_2eC_3$	→	*d_eg_em*

Root-template morphology is non-concatenative because the two morphemes that make up the stem, that is, the root and the template, are interleaved with one another rather than being concatenated; as with the negative circumfix, the root and template are discontinuous.

Non-concatenative morphology presents a problem because finite state morphology has no memory other than the states themselves. For example, consider how an FST would handle a negative Tigrinya verb like **ኣይፈለጠን** *’ayfeleTen*. Once the initial *a* and *y* have been consumed by the FST, it needs to “know” that the verb is negative so that it will expect the *n* at the end. But how will it remember this while the stem of the verb (**ፈለጠ**) is being processed? The only option would seem to be an increase in the number of states, with separate negative and affirmative states for each of the positions between the negative prefix and suffix. But there are a number of other dependencies between prefixes and suffixes, and the required increase in the number of states would be sizable.

To avoid the massive duplication that this approach would require, finite state morphology researchers have introduced several techniques to provide limited memory or divide the work of recognizing different morphemes among separate parallel FSTs (see Cohen-Sygal and Wintner, 2006 for an extended discussion). HornMorpho makes use of the proposal of Amtrup (2003), which adds the possibility of grammatical constraints on the FST transitions in addition to the input-output character conditions. The application of this proposal to the analysis and generation of Tigrinya verbs is described at length in Gasser (2009). Its application to Amharic and Tigrinya is summarized in what follows.

One way of representing grammatical structure is through so-called **feature structures (FSs)** (Carpenter, 1992), consisting of sets of feature-value pairs. An Amharic example is provided by (1) above, repeated here for convenience, with abbreviated feature and value names:

(9) `[stem=ቤት, poss=[pers=1, num=plur], prep=ke]`

The features in this FS include `stem`, `poss`, and `prep`; their values follows the equal signs. Note that the values of features can be simple strings, as in the case of `stem` and `prep` or FSs in their own right, as in the case of `poss`. The fundamental operation on FSs is **unification**, essentially a

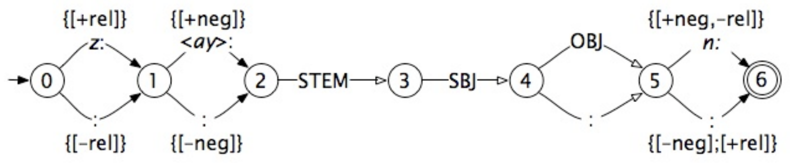

Figure 2. FST handling relativization and negation for perfective Tigrinya verbs.

pattern-matching process. Informally, unification takes two FSs, and if they are compatible, it returns the combination of the two. The details of how unification works are complex, but a few examples should make the general notion clear. If we attempt to unify the FS in (9) with a second FS, `[conjunction=m]`, the result is this more complex FS:

(10) `[stem=ቤት, poss=[pers=1, num=plur], prep=ke, conj=m]`

In fact this represents the structure of the grammatical Amharic word **ከቤታችንም**. If we attempt to unify the FS in (9) with the FS `[prep=be]`, unification fails because the feature `prep` has two different values in the two FSs. Informally this corresponds to the constraint that a word can only have one preposition. Unification also fails if we attempt to unify the FS in (9) with the FS `[poss=[pers=1, num=sing]]` because the `num` feature in the FS that is the value of the `poss` feature has different values (`plur` and `sing`) in the two FSs. Informally, a noun cannot have both a first person plural possessor ('our') and a first person singular possessor ('my').

The insight of Amtrup's (2003) work was to recognize that sets of FSs could be added to the transitions in an FST, adding additional constraints to the possible paths through the FST, at the same time maintaining the fundamental properties of FSTs that make them desirable, in particular the possibility of concatenating, inverting, and composing FSTs. An FST augmented with FS constraints works as follows:

- Processing begins with not only an input string but also an initial FS set, representing any grammatical constraints that are given, and at any given point during processing, a current FS set is maintained along with the remaining unconsumed string.
- When there is an attempt to traverse a transition, as with ordinary FSTs, the input character on the transition must be present at the beginning of the unconsumed input string. In addition, the current FS set must unify with the FS set on the transition (if there is one). The result of unifying two FS sets is the set of all possible unifications of pairs from the two sets. If the set unification fails (no two pairs of FSs unify), it is impossible to traverse the transition. If the set unification succeeds, the resulting FS set (the set of all of the successful unifications of pairs from the two sets) becomes the new current FS set.
- For each successful path through the FST, the resulting output string *and* the resulting FS set are both returned. That is, each analysis consists of a string and set of possible grammatical representations.

FSTs augmented with FS constraints solve the problem of non-concatenative morphology because the FS set that accumulates along each possible path through the FST represents a sort of memory for what previous transitions have been traversed. Consider again the case of the Tigrinya negative circumfix. Figure 2 shows a simplified representation of an FST for handling negation in the perfective tense/aspect. The input character sequence <*ay*> is an abbreviation for a transition with a single character and one intervening state. The transitions with uppercase labels, `STEM`, `SBJ`, and `OBJ`, are abbreviations for the multiple states and transitions that are responsible for the stem, subject suffix, and optional object suffix of perfective verbs. Transitions labeled ":" consume no input character and add no output character. Six of the transitions have FS set constraints. The relevant features for

state	input string	output string	FS set
0	*deq_ese*		`{}`
1	*deq_ese*		`{[–rel]}`
2	*deq_ese*		`{[–rel,–neg]}`
3	*e*	*dq_s*	`{[-rel,–neg]}`
4		*dq_s*	`{[–rel,–neg,sbj=[...]]}`
5		*dq_s*	`{[-rel,-neg,sbj=[...]]}`
6		*dq_s*	`{[–rel,–neg,sbj=[...]]}`

Table 1. Analysis of the string **ደቀሰ** *deq_ese*.

state	input string	output string	FS set
0	*aydeq_esen*		`{}`
1	*aydeq_esen*		`{[-rel]}`
2	*deq_esen*		`{[-rel,+neg]}`
3	*en*	*dq_s*	`{[-rel,+neg]}`
4	*n*	*dq_s*	`{[-rel,+neg,sbj=[...]]}`
5	*n*	*dq_s*	`{[-rel,+neg,sbj=[...]]}`
6		*dq_s*	`{[-rel,+neg,sbj=[...]]}`

Table 2. Analysis of the string **አይደቀሰን** *aydeq_esen*.

state	input string	output string	FS set
0	*zaydeq_ese*		`{}`
1	*aydeq_ese*		`{[+rel]}`
2	*deq_ese*		`{[+rel,+neg]}`
3	*e*	*dq_s*	`{[+rel,+neg]}`
4		*dq_s*	`{[+rel,+neg,sbj=[...]]}`
5		*dq_s*	`{[+rel,+neg,sbj=[...]]}`
6		*dq_s*	`{[+rel,+neg,sbj=[...]]}`

Table 3. Analysis of the string **ዘይደቀሰ** *zeydeq_ese* (*zaydeq_ese*).

this example are `rel`, which has the value `True` for relativized verbs such as **ዝደቀሰ** and **ዘይደቀሰ** and the feature `False` for non-relativized verbs such as **ደቀሰ** and **አይደቀሰን**, and `neg`, which has the value `True` for negative verbs such as **አይደቀሰን** and **ዘይደቀሰ** and the value `False` for affirmative verbs such as **ደቀሰ** and **ዝደቀሰ**. (The values `True` and `False` are abbreviated with + and – signs before the relevant features.) A semicolon separates different FSs within an FS set; the only example is on the lower transition from state 5 to state 6. The interpretation of this FS set is that this transition

state	input string	output string	FS set
0	*zaydeq_esen*		`{}`
1	*aydeq_esen*		`{[+rel]}`
2	*deq_esen*		`{[+rel,+neg]}`
3	*en*	*dq_s*	`{[+rel,+neg]}`
4	*n*	*dq_s*	`{[+rel,+neg,sbj=[...]]}`
5	*n*	*dq_s*	`{[+rel,+neg,sbj=[...]]}`
6	*n*	*dq_s*	`{[+rel,+neg,sbj=[...]]}`

Table 4. Failed analysis of the string **ዘይደቀሰን** *zeydeq_esen* (*zaydeq_esen*).

is compatible with either `-neg` or `+rel` (or both). The FST does not show how the prefix combination *z+ay* becomes *zey*; this results from a rule discussed in Section 6.a.ii.

Now consider how this FST succeeds or fails on different input strings. In each case we assume that there are no initial grammatical constraints; that is, the initial FS set contains the single "top" element which unifies trivially with any FS. Presented with the string **ደቀሰ** *deq_ese*, Table 1 shows what happens. It is assumed that the `STEM` portion of the FST outputs the three-character root of the verb (the gemination character "_" is also part of the root) and that the `SBJ` portion of the FST accumulates the grammatical properties of the subject of the verb on the FS set.

Tables 2, 3 and 4 show the processing of the strings **አይደቀሰን** *aydeq_esen*, **ዘይደቀሰ** *zeydeq_ese* (*zaydeq_ese*), and the ungrammatical string **ዘይደቀሰን** *zeydeq_esen*.[7] The analysis of *zeydeq_esen* fails because the final state in the FST is reached while the unconsumed input string is not empty (it still consists of the character *n*).

For other examples of the use of FS sets as grammatical constraints on transitions, see the discussion of Amharic morphology below and the examples of particular FSTs among the program data files.

Oromo morphology is not characterized by the root-pattern structure of Semitic, and because its only affixes are suffixes, the problem of long-distance dependencies does not occur.[8] However, it is still convenient to implement Oromo morphology using the approach that is outlined in this section.

5. Grammar

Any computational morphology system is based on a theory (explicit or implicit) of the grammar of the language. HornMorpho relies on a long history of research on Amharic, Oromo, and Tigrinya grammar, while introducing a few new notions. I have also been helped greatly by the output of Biniam Gebremichael's web crawler for the Amharic and Tigrinya, available at <http://www.cs.ru.nl/~biniam/geez/crawl.php>, which provides counts for 227,984

[7] In fact **ዘይደቀሰን** *zeydeq_esen* is grammatical, but the *n* suffix at the end is the one that translates as 'and' rather than the negative suffix. This distinction is not handled by the simplified FST in Figure 2.

[8] Oromo does make use of another sort of non-concatenative morphology, reduplication (as do Amharic and Tigrinya in fact), for example, in the verb *caccabsuu*, from *cabsuu*. The feature-structure approach discussed here does not solve the problem of reduplication, which is notorious in finite state morphology. In any case, this aspect of Oromo is not not handled by the current version of HornMorpho.

Tigrinya and 397,352 Amharic words found on the Internet. These data are extremely useful as a means of assessing current usage, especially for Tigrinya.

This section is not meant to be anything like a thorough overview of Amharic, Oromo, and Tigrinya morphology; for more in-depth coverage, see Baye (2000 E.C.), Bender & Hailu (1978), or Leslau (1995) for Amharic; Griefenow-Mewis & Tamene (1994), Griefenow-Mewis (2001), or Abarraa (2003) for Oromo; Leslau (1941) or Amanuel (1998) for Tigrinya.

The grammar of the Cushitic language Oromo is very different from that of the Semitic languages Amharic and Tiginya, but all three languages share aspects of the complex morphology that they apparently inherited from their common Afro-Asiatic ancestor.

An Amharic, Oromo, or Tigrinya word consists of a **lexical** part, or stem (Amharic **አምድ**, Tigrinya **ዓምዲ**), and one or more **grammatical** parts. This is easy to see with a noun, for example, the Amharic noun **አገሮቻቸውን** 'their countries (as direct object of a verb)'. The lexical part is the stem **አገር**; this conveys most of the important content in the noun. Since the stem can't be broken into smaller meaningful units, it is a **morpheme** (Amharic **ምዕላድ**, Oromo *latii*), a primitive unit of meaning. This is followed by three grammatical suffixes, each of which provides information that is more abstract and less crucial to the understanding of the word than the information provided by the stem: -*oc_*, -*ac_ew*, and -*n*. Each of these suffixes can be seen as providing a value for a particular grammatical feature (or dimension along which Amharic nouns can vary):

(11)	-*oc_*:	`[number=plural]`
	-*ac_ew*:	`[possessor=[person=3,number=plural]]`
	-*n*:	`[case=accusative]`

Since each of these suffixes can't be broken down further, it can be considered a morpheme. Exactly what these grammatical morphemes contribute to the meaning of the whole word is complex and will not be discussed further in this document.[9] The output of a morphological analyzer such as HORNMORPHO is not really a *semantic* representation of the input word; it is a representation of the grammatical structure of the word. In any case, the grammatical structure would be needed by any system that performs a semantic analysis.

HORNMORPHO2.0 analyzes Amharic nouns and verbs and Oromo and Tigrinya verbs. We begin with a discussion of verbs, which are significantly more complex than nouns but which are dealt with more thoroughly by the program. Because its morphology is very different from that of the two Semitic languages, Oromo will be treated separately.

a. AMHARIC AND TIGRINYA VERBS

i. Stems: templates

An Amharic or Tigrinya verb consists of zero or more prefixes, zero or more suffixes, and a stem. The prefixes and suffixes are all grammatical morphemes; they are discussed below. The stem is the lexical part of the verb and also the source of most of its complexity.

[9] For more on morphology, including the distinction between lexical and grammatical morphemes, see Gasser (2006).

As discussed in Section 4.b., an Amharic or Tigrinya verb stem can be broken down into two components, a verb root (Amharic **ስር**) and a grammatical template, each of which can be viewed as a morpheme in its own right. A given root can be combined with as many as 40 templates for Amharic and 24 templates for Tigrinya, though it is rare for a root to occur with all. Different templates can be described as varying along three grammatical dimensions:

- **tense/aspect/mood**
  Tense/aspect/mood (TAM) is signalled mainly by the prefixes and suffixes that indicate the subject of the verb but is also reflected in the stem template. In both languages, TAM has four possible values, traditionally referred to as **perfective** (or perfect), **imperfective** (or imperfect), **jussive/imperative**, and **gerundive** (or gerund). Here are examples from both languages of each with everything else held constant; in the romanization, the stem appears in boldface.

• perfective:	**ደረሰ** ***der_es***-*e*	**ገደፈ** ***gedef***-*e*
• imperfective:	**ይደርሳል** *yI*-***ders***-*al*	**ይገድፍ** *yI*-***ged_If***
• jussive:	**ይድረስ** *yI*-***dres***	**ይግደፍ** *yI*-***gdef***
imperative:	**ድረስ** ***dres***	**ግደፍ** ***gIdef***
• gerundive:	**ደርሶ** ***ders***-*o*	**ገዲፉ** ***gedif***-*u*

- **voice**
  Voice has three possible values in Tigrinya, four in Amharic. It is signaled by the stem prefixes *te-* and *a-* for both languages and *as-* for Amharic, as well as particular patterns of vowels between the root consonants and gemination of the root consonants. I will refer to the plain form that has no prefix as **simplex** voice. Both languages have a further form that is conventionally called **passive/reflexive** (for simplicity, usually referred to here as "passive"). Amharic has two further forms, which I will call **transitive** and **causative**. In Tigrinya (as in Ge'ez) these correspond to a single form that I will call **transitive/causative** (for simplicity, usually shortened to "transitive"). These terms are only suggestive of the semantics, which in many cases depends on the particular root (for example, Amharic **ተቀበለ**, Tigrinya **ተቐበለ** is passive/reflexive in form but neither passive nor reflexive semantically). In fact, most Amharic roots fail to appear in all four forms, and many Tigrinya roots fail to appear in all three forms. Here are examples in both languages of each option in perfective and imperfective.

• simplex:	**ደረሰ** ***der_es***-*e*	**ይደርሳል** *yI*-***ders***-*al*
	ገደፈ ***gedef***-*e*	**ይገድፍ** *yI*-***ged_If***
• passive:	**ተደረሰ** ***teder_es***-*e*	**ይደረሳል** *yI*-***d_er_es***-*al*
	ተገድፈ ***tegedf***-*e*	**ይግደፍ** *yI*-***gId_ef***
• transitive:	**አደረሰ** ***ader_es***-*e*	**ያደርሳል** *y*-***aders***-*al*
transitive/causative:	**አግደፈ** ***agdef***-*e*	**የግድፍ** *y*-***agId_If*** (*yegId_If*)
• causative:	**አስደረሰ** ***asder_es***-*e*	**ያስደርሳል** *y*-***asder_Is***-*al*

- **stem-internal aspect**
  In Amharic and Tigrinya, like other Semitic languages spoken in the Horn of Africa, there are two ways to modify stems through the introduction of the vowel *a*. In combination with different values for the voice feature, these two forms convey a wide range of subtle meanings, depending to a

large extent on the particular verb root. I will refer to these two options by names which are based on their frequent meanings but by no means do justice to all of the possible interpretations. **Reciprocal** is formed by the insertion of *a* between the third and second consonants from the end of the root; this can only occur in combination with passive and transitive voice. **Iterative** is formed by the reduplication (copying) of the second consonant from the end of the root and the insertion of *a* between the two copies of this consonant; this occurs in combination with all four voice options in Amharic (though only rarely with the causative) and all three voice options in Tigrinya. Other details of the templates depend on the values of the TAM and voice features. The plain form, with no *a* inserted in the stem, is again referred to as **simplex**. Here are examples of the three stem-internal aspect options, in combination with perfective TAM and both simplex and passive voice.

- Simplex aspect:
  +simplex voice: **ደረሰ** ***der_es***-*e* +passive voice: **ተደረሰ** ***teder_es***-*e*
  ገደፈ ***gedef***-*e* **ተገድፈ** ***tegedf***-*e*
- Reciprocal aspect:
  +simplex voice: (not possible) +passive voice: **ተዳረሰ** ***tedar_es***-*e*
  ተጋደፈ ***tegadef***-*e*
- Iterative aspect:
  +simplex voice: **ደራረሰ** ***derar_es***-*e* +passive voice: **ተደራረሰ** ***tederar_es***-*e*
  ገዳደፈ ***gedadef***-*e* **ተገዳደፈ** ***tegedadef***-*e*

ii. Stems: roots

The Amharic and Tigrinya verb stem is complicated not only because of the large number of combinations of TAM, voice, and stem-internal aspect; a whole set of additional complications results from the different categories that roots fall into. Like TAM, voice, and stem-internal aspect, the category of a root can affect the template vowels and consonant gemination.

HORNMORPHO relies on a new classification of root types, based on seminal earlier work by Bender & Hailu (1978) and also designed to reflect the commonalities between the Semitic languages, in particular for those cases which have only two overt root consonants in Amharic but three in Tigrinya and Tigre. Each root consists of a sequence of consonants and in addition, possibly a single *a*, a single character ("_") indicating gemination, and a single character ("|") following the first consonant indicating that it is always "fused" to the next consonant. In the rest of this document, roots are written surrounded by angle brackets ("<>"). The root categories are described below. Each root example is followed by its **citation form**, that is, the third person singular masculine perfective form in simplex voice if this is possible for the root, otherwise passive or transitive voice.

- **CCC(C)(C)**
  Most roots consist only of consonants. There are three main subcategories, differing in the number of consonants.
  - **CCC**: simple three-consonant roots. Examples: <*sbr*> **ሰበረ**, <*mrT*> **መረጠ**; <*mrS*> **መረጸ**.
  - **CCCC**: simple four-consonant roots. Examples: <*klkl*> **ከለከለ**, <*dngT*> **ደነገጠ**; <*dngS*> **ደነገጸ**.

    - **CCCCC**: simple five-consonant roots (very rare). Examples: *<wxngr>* **ወሸነገረ**; *<nblbl>* **ነበልበለ**.

- **CC_C**
  Three-consonant roots for which the second consonant is geminated in most templates. In Amharic this category is indistinguishable from CCC in the perfective, distinguished in pronunciation but not writing in the imperfective and gerundive, and distinguished in both writing and pronunciation in the jussive/imperative: **ይስበር** (CCC), **ይጀምር** (CC_C). In Tigrinya it is distinguishable in pronunciation but not writing from CCC in the perfective and gerundive, distinguished in both writing and pronunciation in the imperfective and jussive/imperative: **ይሰ፟ብር**, **ይስበር** (CCC); **ይጅ፟ምር**, **ይጀምር** (CC_C). Examples: *<Ty_q>* **ጠየቀ**, *<Cr_s>* **ጨረሰ**; *<Ty_q>* **ጠየቐ**, *<ws_k>* **ወሰ፟ኸ**.

- **C*a*CC**, **CC*a*CC**, **C|CC*a*CC**
  Roots in which the vowel *a* always appears before the second-to-last consonant. Examples: *<gabz>* **ጋበዘ**, *<mark>* **ማረከ**, *<glamT>* **ገላመጠ**; *<bakn>* **ባ፟ኸነ**, *<mark>* **ማረ፟ኸ**, *<brabr>* **ተበራበረ**; *<bakn>* **ባ፟ኸነ**, *<mark>* **ማረ፟ኸ**.

- **C|C*a*CC**, **C|CCCC**, **C|CC*a*CC**
  Roots in which the first consonant is never followed by a vowel.[10] These roots appear only in passive and transitive voice (never in simplex or causative). Examples: *<n|saff>* **ተንሳፈፈ**, *<s|gbgb>* **ተስገበገበ**, *<n|xratt>* **ተንሸራተተ**; *<H|xkWxkW>* **ኣሕሸ፟ኹሸ፟ኹ**, *<n|qsaqs>* **ተንቀሳቐሰ**.

Particular root consonants result in templates that deviate from what is expected. This is especially true for Amharic.

- When any of the consonants in an Amharic root is *'* or *`*, and when the second root consonant is ungeminated *w* or *y*, the consonant is realized as a vowel; hence these roots are usually described as biconsonantal. Examples: *<'sr>* **አሰረ**, *<s'm>* **ሰማ**, *<bl'>* **በላ**, *<'s_b>* **አሰበ**, *<TT_'>* **ጠጣ**, *<qwm>* **ቆመ**, *<mwt>* **ሞተ**, *<xyT>* **ሸጠ**, *<hyd>* **ሄደ**, *<'nTs>* **አነጠሰ**, *<fnd'>* **ፈነዳ**, *<b'b'>* **ባባ**.

- When any of the consonants in a Tigrinya root is a laryngeal (*'*, *`*, *h*, or *H*), the vowel before or after this consonant may differ from what is usual: *<'tw>* **ኣተወ**, *<SHf>* **ጸሓፈ**, *<bl`>* **በልዐ**, *<s`s``>* **ሳዕስዐ**.

- A special subcategory of Amharic CCC and CCCC roots consists of roots that originally ended in *y* and behave to some extent like CC' or CCC' roots. The symbol "*" is used to represent the final root consonant. Examples are *<mx*>* **መሸ**, *<fj*>* **ፈጀ**, *<slc*>* **ሰለቸ**, and *<zgy*>* **ዘገየ**. The exceptional roots *<qr*>* **ቀረ** and *<sT*>* **ሰጠ**, though they apparently never contained a *y*, are included in this subcategory because they behave like the other members.

- When the second or third consonant in a Tigrinya root is ungeminated *w* or *y*, the consonant is realized as a vowel in some templates. Examples: *<xyT>* **ሸጠ**, **ይሸይጥ**; *<ftw>* **ፈተወ**, **ፈቶ፟ኻ**, **ይፈቱ**; *<sty>* **ሰተየ**, **ሰተ፟ኻ**, **ይሰቲ**.

[10] Many of the C|CCCC roots are derived from CCC roots through the reduplication of the last two consonants, for example, Amharic *rgfgf* **እርገፈገፈ** from *rgf* **ረገፈ**. However, since this process is not particularly productive, and the meanings of the resulting C|CCCC roots are not always predictable, these are treated as independent roots in their own right in HornMorpho.

- In any of the four categories, root consonants may be **labialized** (Amharic የከናፍር; Baye, 2000 E.C.), indicated in the romanization with a following *W*. In some templates, this may cause the following vowel to be *o* or *u*, sometimes optionally. In HornMorpho these changes are handled by an alternation rule. Examples: *<kWr'>* ኰራ (CCC), *<dWldWm>* ዶለዶመ (CCCC), *<bWaCr>* ቧጨረ (C*a*CC), *<xmWaTT>* አሽሟጠጠ (C|C*a*CC); *<HqWf>* ሓቈፈ / ሓቖፈ (CCC), *<mWg_s>* ተሞገሰ (CC_C), *<mrkWs>* ተመርኰሰ / ተመርኮሰ (CCCC).

While most roots occur with all three values of stem-internal aspect, this is not the case for voice. A sizable subset of roots fail to occur in the simplex voice (in addition to the roots beginning with "C|", which, as noted above, appear only in passive and transitive voice). Examples: *<drg>* ተደረገ, *<qb_l>* ተቀበለ, *<mlkt>* ተመለከተ, *<qm_T>* ተቀመጠ, *<ds_t>* ተደሰተ; *<zrb>* ተዘርበ, ተዛረበ; *<rd_'>* ተረድአ; *<qb_l>* ተቐበለ; *<qm_T>* ተቐመጠ; *<Sb_y>* ተጸበየ.

iii. Affixes

The verb in both languages has four slots for prefixes before the stem and four slots for suffixes after the stem. The positions of the affixes are as follows. Parentheses indicate optionality, and a vertical bar indicates different options within a given slot. Each of the affixes is described briefly below.

```
(12)  Amharic:   (prep|conj)(rel)(neg) sbj
                 STEM
                 sbj (obj|def)(neg|aux|acc)(conj)
      Tigrinya:  (prep|conj)(rel)(neg) sbj
                 STEM
                 sbj (obj) (neg) (conj)
```

An Amharic or Tigrinya verb must agree with its subject. Subject agreement (`sbj`) is expressed by suffixes in the perfective, gerundive, and imperative and by prefixes and suffixes (in the second person singular feminine and second and third person plural) in the imperfective and jussive. There are eight possible combinations of person, number, and gender in Amharic and ten possible combinations in Tigrinya. Some forms are ambiguous; for example, Amharic ይመጣሉ *yImeTal_u* can have a second or third person singular polite subject, as well as a third person plural subject; Tigrinya ትኸዱ *tIKedu* can have a second person masculine singular polite subject, as well as a second person masculine plural subject.

Following the subject agreement suffix, there may be an object suffix (`obj`). Here there are nine possible forms for Amharic because the second person polite form (*-wo(t)*) is distinguished from the the third person plural (*-ac_ew*). Both languages also have the possibility of an indirect (or applicative or prepositional) object suffix. In Tigrinya, these are marked by the prefix *-l-* before the object suffix, with a range of meanings: 'to, for, for the benefit of, on, at, from, to, with, by, to the detriment of'. In Amharic, indirect objects are marked by one of two prefixes before the object suffix: *-l_-* 'to, for, for the benefit of', *-b_-* 'on, at, from, to, with, by, to the detriment of'. Thus there are a total of 20 possible object suffixes in Tigrinya, 27 in Amharic. Examples of each type: አቀፈቻቸው *'aq_efec_-**ac_ew*** 'she hugged **them**', አቀፈችላቸው *'aq_efec_-**Il_ac_ew*** 'she hugged (somebody) **for them**', አቀፈችባቸው *'aq_efec_-**Ib_ac_ew*** 'she hugged (somebody) **to their detriment**'; ሓቝፋቶም *HaQWifat_**om*** 'she hugged **them**', ሓቝፋትሎም *HaQWifatI**l_om*** 'she hugged **for/against them**'.

Negation (`neg`) in both languages is indicated by a prefix and, for non-subordinate verbs (see below), a suffix as well: **አልተረጋገጠም** ***'al**-teregag_eTe-**m***, **ኣይተረጋገጸን** ***'ay**-teregageSe-**n*** 'it was **not** verified'. The gerundive is never negated.

In both languages, verbs in the perfective and imperfective may be **relativized** (`rel`) with the prefix *ye-* (perfective) or *yem_-/Im_-* (imperfective) for Amharic, *z-* for Tigrinya. A relativized verb fills the verb position in a relative clause (Amharic **አዛማጅ አረፍተ ነገር**, Baye, 2000) and, as such, functions more like an adjective than a verb: **የደረቀ እንጨት** ***ye**-der_eqe 'InCet*, **ዝደረቐ ዕንጨይቲ** ***zI**-dereQe `InCeyti* 'wood that dried; dry wood'. A relativized verb without a modified noun behaves like a noun clause: **የደረቀው** 'what dried; the dry thing'. Note that the third person singular masculine object suffix in Amharic (but not Tigrinya) doubles as a marker of **definiteness** (`def`) for relativized verbs: **የደረቀው እንጨት** *yeder_eqe-**w** 'InCet* '**the** dry wood'.

Relativized verbs are **subordinate**; that is, they cannot act as the main verbs of sentences. Verbs can also be made subordinate through the addition of a prefix conjunctions (`conj`) such as Amharic *bI-* and *IndI-* and Tigrinya *nIKI-* and *'Inte-*. A verb with one of these prefixes functions as the verb in an adverbial clause: **ሥጋ ብበላ** *^sIga **bI**-bela* '**if** I eat meat', **በጊዜ እንዲደርስ** *begizE **'Ind**-iders* '**so that** he arrives on time'; **እንተረኺብካያ** ***'Inte**reKibkay_a* '**if** you (masc. sing.) meet her', **ንኽንኸውን** ***nIK-**In_IKew_In* '**so that** we become'. Except for the negative following **እንተ** *'Inte-*, prefix conjunctions never co-occur with the relativizing prefix.

A noun that is modified by a relative clause may be the object of a preposition, and in Amharic and Tigrinya this preposition (`prep`) is prefixed to the relativized verb rather than the noun itself, as would be the case with an unmodified noun (see the section on nouns below). With a prepositional prefix, the Amharic relative prefix drops out in the perfective and is shortened to *-m_-* in the imperfective. This does not happen in Tigrinya, and the preposition is often written as a separate word. Examples: **ለወደቀው ተማሪ** ***le**-wed_eqew temari* '**for** the student who failed', **በሚገኝበት ከተማ** ***be**-m_ig_eN_Ib_et ketema* '**in** the town where it is found'; **ንዝሓለፉ** 4 **ዓመታት** ***nI**-zIHalefu 4 `ametat* '**for** the four years that passed'.

A noun that is modified by a relative clause may also be the direct object of a verb, and in Amharic it then requires the **accusative** (`acc`) suffix under some circumstances. As with prepositions, this suffix is added to the relativized verb rather than the noun: **እጅ የሰጡትን ወታደሮች ማረኳቸው** *'Ij_ yeseT_ut-**In** wet_ad_eroc_ mar_ekWac_ew* 'they captured the soldiers who surrendered'.

Amharic verbs in the imperfective and gerundive may take a form of the auxiliary (`aux`) verb **አለ** *'al_e* as a suffix. The auxiliary is required for affirmative, non-subordinate imperfective verbs in most contexts: **ይወድቃሉ** *yIwedq-**al_u***. With the gerundive, the auxiliary creates a non-subordinate form similar to English present perfect: **ወድቀዋል** *wedqew-**al*** 'they have fallen'. The corresponding auxiliaries in Tigrinya are normally written as separate words, and they are not handled in HORNMORPHO.

Finally, verbs in both languages may take one of a set of conjunctive or adverbial suffixes (`conj`): **ትውጣና** *tIwTa-**n_a*** 'let her go out **and**...', **የተደበቀውም** *yetedeb_eqew-**m*** '**also** what was hidden'; **ዘምጽኦምን** *zemSI'om-**In*** 'who brought them **and**...', **ተረዲኡኪዶ** *tered_i'uk_i-**do*** '**do** you (fem. sing.) understand**?**'

b. Oromo verbs

i. *Suffixes*

An Oromo verb consists of a stem and one or more suffixes. Some writers also treat the negative morpheme *hin* as a prefix, for example, writing *hindubbadhu* in place of the more usual *hin dubbadhu* 'I don't speak'. We'll first look at the different suffixes that are possible and then look at the stem.

Oromo verbs divide into those that must agree with their subject, **finite verbs**, and those that don't, **non-finite verbs**. The morphotactics of finite verbs is as follows. As above, parentheses indicate optionality.[11] The details are not completely clear to me; for example, I am not sure whether the continuative can co-occur with a conjunctive suffix.

(13) Oromo finite verbs: `STEM sbj tam (contin) (case) (conj) (1s_sbj)`

Finite verbs take suffixes that indicate **subject** agreement, specifically one of the seven basic person/number/gender categories of the language: first person singular and plural, second person singular and plural, third person singular masculine and feminine, and third person plural. A finite verb suffix also must indicate one of six **tense/aspect/mood** (TAM) categories. I will refer to these categories with the English terms used by Griefenow-Mewis (2000): present, past, perfect, imperative, subordinate, and contemporary. Here are examples of each with second person singular subject agreement and the verb stem *deem-* 'go':

(14)	**present**	*deem-t-a*	'you go'
	past	*deem-t-e*	'you went'
	perfect	*deem-t-eerta*	'you have gone'
	imperative	*deem-i*	'go!'
	subordinate	(*akka*) *deem-t-u*	'(that) you go'
	contemporary	(*otuu*) *deem-t-uu*	'(while) you are/were going'

A discussion of the functions of the different TAM categories is beyond the scope of this document; in fact, future work on Oromo grammar could result in a different sort of breakdown as well as a better understanding of the semantics of the categories. I only note here that the negative of the present tense uses the subordinate form: *hin deem**tu*** 'you don't go'.

Note that the perfect is actually a shortened version of the longer form consisting of the "past" form of the verb followed by the present of the auxiliary verb *jira*: *deemteerta = deemtee jirta.* The language has a number of other compound tenses, but these will not concern us here because they are never contracted like the perfect. Note also that some suffixes combine the subject agreement with the TAM. This is true for the imperative.

Some suffixes are ambiguous. Except for the present, the examples above could also refer to the third person singular feminine in addition to the second person singular: *deemte* 'you (sing.), she went'. The second person and third person plural suffixes are ambiguous with respect to TAM. For example, the word *deem**ani*** could be present, past, or subordinate. For the past negative, there is a single form for all person/number/gender categories *hin deem**ne*** 'I/you/he/she/we/they didn't go'.

[11] The details are not completely clear to me. For example, I do not know whether the case and conjunctive suffixes can co-occur.

Under certain circumstances, the final vowel of the verb suffix(es) is lengthened, especially for yes/no questions and when there is a sequence of verbs representing consecutive events: *deemt**ee**?* 'did you/she go?', *deebi'an**ii** dhufan* 'they returned and came (= they came back)'. I will refer to these uses as "continuative" (`contin` in (13)).

Following the subject agreement and TAM suffix(es), several optional suffixes are also possible with finite verbs. Case is normally marked on nouns in Oromo, but with certain verbs, the case suffixes *-f* ("dative") and *-n* ("instrumental") may appear on the verbs instead. In these cases the suffixes refer to arguments of the verb that are left unstated, corresponding to pronouns in English: *kennee**f*** 'he gave (him/her/them)', *jedhee**n*** 'he said (to him/her/them)'. Note that these suffixes, like some others in Oromo, lengthen a preceding short vowel.

Oromo has several suffixes functioning as conjunctions that can be attached to verbs: *-f*, *-s*, *-ti*, *-tti*, *-llee*, and *-yyuu.* Some of these are normally combined with other conjunctions consisting of separate words appearing before the verb: ***waan** hin arganne**ef*** 'because it (he, etc.) was not found'.

Finally, non-subordinate finite verbs with first person singular subjects are treated specially in Oromo; -(*a*)*n* is suffixed to the word *preceding* the verb. Because verbs can appear in succession, this suffix (`1s_sbj`) can appear on a verb that is followed by another verb. For example, *deemee**n** ture* 'I had gone'.

Non-finite Oromo verbs include those referred to as **participle** and **gerund** by Griefenow-Mewis, for example, *deem**aa*** 'going' and *deem**naan*** 'having gone'.

The Oromo **infinitive** is a noun formed from a verb, for example, *deem**uu*** 'to go'. As a noun, an infinitive can take any of the noun suffixes of the language, for example, *deemuu**dhaaf*** 'for going'. Since HornMorpho does not currently handle nouns generally, noun morphology is not discussed further in this document.

ii. Stems

The stem of an Oromo verb can also consist of multiple morphemes because the language has many ways to **derive** verbs from other words, including other verbs. For example, the stem of the verb *barsiisa* 'he teaches', *barsiis-*, is derived from the verb **root** *bar-* 'learn'.

Oromo has four ways of deriving verb stems from verb roots with derivational suffixes.

- **Passive**: *-am*, for example, *beek**am**e* 'he/it was known'.
- **Causative**: *-s*, *-sis*, *-siis*, for example, *beek**siis**e* 'he announced (caused to know)'.
- **Autobenefactive**: *-adh*, for example, *beek**at**a* 'he realized (knew for himself)'.
- **Intensive**: reduplication of first consonant and vowel of root, for example, ***waw**waame* 'he called intensely'. The intensive is not handled in HornMorpho.

Combinations of these morphemes are also possible, for example, *arg**amsiis**e* 'he/it brought about (caused to be seen)' (passive + causative).

Oromo also has ways of deriving verbs from nouns and adjectives, for example, *dhugoomuu* 'to become true' from *dhugaa* 'true', but these are not handled in HornMorpho.

c. Amharic nouns

An Amharic noun consists of a stem and zero or more prefixes and suffixes. Since Amharic nouns and adjectives are similar morphologically, they are not distinguished in HornMorpho, and many of the words referred to as "nouns" in what follows are more properly thought of as adjectives.

i. Stems

Noun stems are of two types. Some are unanalyzable in that they are not derived from a simpler stem or root. Examples: **ቤት** *bEt*, **ምሳ** *mIsa*, **ጨረቃ** *Cereqa*.

Other stems are derived from verb, adjective, or noun roots. Like other Semitic languages, Amharic has many ways of deriving nouns from verbs. As with verb stems, each of these **deverbal noun** stems is formed through a combination of a verb root and a grammatical template. HornMorpho handles four categories of deverbal nouns, those that are possible with all verb roots.

- **infinitive**
  The infinitive is recognizable by the stem prefix *m(e)-*. As with full-fledged verbs, an infinitive stem has values for voice and stem-internal aspect. The infinitive is the only deverbal noun that has a negative form. Examples: **መቍረጥ** *meqWreT*, root: <*qWrT*>, voice: simplex, aspect: simplex; **አለማሳደብ** *'alemas_adeb*, root: <*sdb*>, negative, voice: transitive, aspect: reciprocal.

- **instrumental**
  The instrumental is based on the infinitive, differing only in the suffix and its possible effect on the last root consonant. The name "instrumental" comes from a common meaning of this form, the instrument that is used in the action associated with the verb, but it has other functions as well. Like the infinitive, it varies in terms of voice and stem-internal aspect. Examples: **መክፈቻ** *mekfeca*, root: <*kft*>, voice: simplex, aspect: simplex; **መታጠቢያ** *met_aTebiya*, root: <*'Tb*>, voice: passive, aspect: simplex.

- **agentive**
  The agent stem may refer to the "doer" of the action of the verb, but it has other functions as well, including adjective-like functions. Like the infinitive, agentive forms vary in terms of voice and stem-internal aspect. Examples: **ገዥ** *geZ*, root: <*gz'*>, voice: simplex, aspect: simplex; **አወዳዳሪ** *'aw_edadari*, root <*wdr*>, voice: transitive, aspect: iterative.

- **manner**
  The manner stem refers to the manner or way in which the action of the verb is performed. Its template is based on the transitive iterative template, but it is invariant across all voice and aspect values. Examples: **አነጋገር** *'an_egager*, root: <*ngr*>, **አስተዳደር** *'astedader*, root: <*'dr*>.

Amharic noun stems are also derived from adjectives, for example, **ነጻነት** *neSa-**nnet***, and nouns, for example, **አብዮታዊ** *'abIyot-**awi***. Of these possibilities, only the *-awi* suffix is handled in HornMorpho; the program does not analyze stems such as **ነጻነት** further.

ii. Affixes

The Amharic noun has two prefix slots and four suffix slots. None of the affixes is obligatory. The possibilities are as follows. Each is described briefly below.

(15) `(prep | gen) (distrib) STEM (plur) (poss | def) (acc) (conj)`

The plural (`plur`) noun suffix is normally *-oc_*. Other cases are treated as irregular plurals; only some of these are known to HornMorpho, for example, ***መንግሥታት*** *mengI^st-**at***.

There are nine possible possessive (`poss`) suffixes for the different combinations of person, number, and gender (including the second person polite suffix *-wo(t)*). The third person singular suffixes are ambiguous since they double as definite (`def`) articles, for example, **ክፍሉ** *kIfl-**u*** '**his** room; **the** room'.

If the noun is definite and the direct object of a verb, it must normally take the accusative (`acc`) suffix *-n*.

When a noun is the object of a preposition and not modified by an adjective or relative clause, the preposition (`prep`) is prefixed on the noun. The **genitive** (`gen`) marker *ye-* appears in the same position.

The distributive (`distrib`) prefix *Iy_e-* translates roughly as 'each'.

Finally, more or less the same set of conjunctive or adverbial suffixes (`conj`) is possible with nouns as with verbs.

Some examples: **ለየክልሉ** *le-y_e-kIl_Il-u*, stem: *kIl_Il*, `[prep=`*`le`*`, +distrib, +def]`; **የርምጃቸውንና** *ye-rmIj_a-c_ew-n-In_a*, stem: *'IrmIj_a*, `[+gen, +acc, poss= [pers=3, num=plur], conj=`*`na`*`]`; **ከወንድሞቻችሁም** *ke-wendIm_-oc_-ac_Ihu-m*, stem: *wendIm_*, `[prep=`*`ke`*`, +plur, [poss= [pers=2, num=plur], conj=`*`m`*`]`.

6. Alternation rules

As discussed in the last section, Amharic and Tigrinya are quite similar at the level of verb morphotactics, as well as with respect to the options that are available for each of the positions in the verb. But there is more to words than morphotactics, which is meant to apply as generally as possible. In specific cases, when particular phones appear in particular positions within words or particular combinations of phones come together, the surface wordform that results may be different from what the morphotactics would dictate. Alternation rules, implemented as finite state transducers, specify these changes. The alternation rules for Amharic and Tigrinya are very different, so they are dealt with separately in what follows.

a. Allomorphy

i. Amharic

Allomorphy refers to variation in the form of a morpheme, depending on its context. HornMorpho includes three alternation rules for Amharic allomorphy, listed below along with the data files in which they are implemented and some examples.

1. Third person singular object suffix
   t following *o* and *u*, *w* following other vowels, *ew* following consonants
   Implemented in `3sm.fst`, where the suffix is represented on the lexical end by "3"
   ወደዱት *wed_edu-**t***, **ወደደው** *wed_ede-**w***, **ይወደዋል** *yIwed_-**ew**-al*

2. Consonant in the first person singular and second person masculine perfective subject suffixes
   h following vowel, *h* or *k* following consonant
   Implemented in `kh.fst`, where the consonant is represented on the lexical end by “7”
   ረሳሁ *res_a-**hu***, **አልኩ** *’al-**ku***, **አልሁ** *’al-**hu***

3. Third person singular possessive suffix, noun masculine definite suffix
   u following consonants, *w* following vowels
   Implemented in `u2w.fst`
   ልጁ *lIj-**u***, **አልጋው** *’alga-**w***.

ii. Tigrinya

HORNMORPHO includes two alternation rules for Tigrinya allomorphy.

1. Relativizing prefix
   z-, optionally *I*- (geminating the next consonant) before a prefix beginning with *t* or *n*
   Implemented in `rel.fst`, where the prefix is represented as “R”
   Examples: **ዝሰበረ** ***zI**-sebere*, **ዝተሰብረ** ***zI**-tesebre,* **እተሰብረ** ***’It**-tesebre*

2. *a/e* in transitive and negative prefixes
   In the transitive stem prefix *a*- and the negative prefix *ay*-, the initial vowel is realized as *e* unless it starts the word (that is, after the consonant *’*).
   Implemented in `A.fst`, where the vowel is represented as “A”
   Examples: **አይረአዮን** *’**a**yre’ayon*, **ዘይረአዮ** *z**e**yre’ayo*, **አቐመጦም** *’**a**Qem_eTom*, **ዘቐመጦም** *z**e**Qem_eTom*

iii. Oromo

HornMorpho includes three alternation rules for Oromo allomorphy.

1. Singular imperative suffix
   -*u* following autobenefactive stems (almost all stems ending in -*dh*), -*i* in all other cases
   Implemented in `dh+.fst`, where the suffix is represented as *I*
   Examples: *dubbadhu* (from *dubbadh-*), *bani* (from *ban-*)

2. Infinitive suffix
   -*ch* replaces the final -*dh* of autobenefactive verbs before -*uu*; -*uu* only suffixed to all other stems
   Implemented in `dh+.fst`, where the suffix is represented as *Cuu*
   Examples: *dubbachuu* (from *dubbadh-*), *banuu* (from *ban-*)

3. Passive suffix, negative imperative suffix, third person singular masculine subject agreement
   -t replaces the final -dh of autobenefactive verbs at the beginning of each of these suffixes; no such change following other stem-final consonants
   Implemented in dh+fst, where each of the suffixes begins with T
   Examples: *dubbatama* (from *dubbadh-*), *banama* (from *ban-*); *hin dubbatin*, *hin banin*; *dubbata*, *bana*

b. PHONOLOGY AND ORTHOGRAPHY

When morphemes come in contact with one another in Amharic, Oromo, or Tigrinya words, there are a number of general phonological processes and a few orthographic rules that apply, all of which can be captured in alternation rules.

i. Amharic

The following is a list of the Amharic alternation rules that have been implemented in HORNMORPHO, along with the names of the files specifying the FSTs for the rules and one or more examples. Each of the rules is described as it applies in the generation direction, that is, from more abstract (lexical) to more concrete (surface). In the rule notation, "*C*" means any consonant (note that in the romanization of words it represents the consonant of **ጨ**); "*V*" means any vowel; "#" represents the end of a word; "_" indicates gemination of the preceding consonant; "|" represents alternatives; other abbreviations are given following the rule. Unless otherwise indicated, the rules are obligatory.

1. *CC* → *C_*
   Implemented in `gem.fst`
   *yIbe**rr**al* → *yIbe**r_**al* (**ይበራል**)

2. *gk* | *qk* → *k_*, *Tt* → *t_*; optional
   Implemented `CC_ass.fst`
   *fel_e**gk**ut* → *fel_e**k_**ut* (**ፈለኩት**), *se**Tt**oal* → *se**t_**oal (→ setW_al)* (**ሰቷል**)

3. *sS* → *S_*; *S*: *^s*, *S*, *^S*, *x*, *z, Z*; only applies within verb stems (*as*- causative prefix)
   Implemented in `stem_SS.fst`
   *a**sz**ereg_a* → *a**z_**ereg_a* (**አዘረጋ**)

4. *CWe* → *Co*, *CW(I)* → *Cu*; optional for *g*, *k*, *q*; only applies within verb stems
   Implemented in `stem_lab.fst`
   ***xWe**l_eke* → ***xo**l_eke* (**ሾለከ**)

5. *Ko* → *KWe*, *Ku* → *KW*; *K*: *g*, *k*, *q*; optional; only applies within verb stems
   Implemented in `stem_lab.fst`
   ***qo**me* → ***qWe**me* (**ቈመ**)

6. *Cwa* | *Coa* | *Cua* → *CWa*; optional
   Implemented in `w2W.fst`, `ou2W.fst`
   *^ser**gwa*** → *^ser**gWa*** (**ሥርጓ**), *sebsI**boal*** → *sebsI**bWal*** (**ሰብስቧል**)

7. *VE* → *VyE*, *Vo* → *Vwo*, *ia* → *iya* | *Iya*, *ua* → *uwa*, *oa* → *owa*; mostly optional
   Implemented in `n_C_epen.fst`, `y_epen.fst`
   *gel**aE*** → *gel**ayE*** (**ገላዬ**), *gel**ao**c_* → *gel**awo**c_* (**ገላዎች**), *tIfel_Igi**a**l_ex* → *tIfel_Igi**ya**l_ex* (**ትፈልጊያለሽ**)

8. *aa* | *(e)ea* | *ae(a)* → *a*, *eu* | *au* → *u*, *ao* → *o*, *ai* → *i*, *ee* → *e*, *VI* → *V*; obligatory with exception of *ao* and *ea* in some circumstances
   Implemented in `VV.fst`, `n_VV.fst`, `ao.fst`, `ea.fst`
   *gIb**au*** → *gIb**u*** (**ግቡ**), *qer_**ee**c_* → *qer_**e**c_* (**ቀረች**)

9. *IyC → iC*, *IyV → iyV*
   Implemented in `y2i.fst`
   *s**IyS**If → s**iS**If* (**ሲጽፍ**)

10. *JiV | J(_)EV → J(_)V*, *Ji# → J#*; *J*: *c*, *C*, *j*, *N*, *x*, *y*, *Z*; the second rule is optional
    Implemented in `vn_pal.fst`
    *teqem_**IC_Ea**l_ehu → teqem_**IC_a**l_ehu* (**ተቀምጫለሁ**)

Another phonological process characteristic of Amharic, **palatalization** (**ላንቃዊነት** in Baye, 2000 E.C), occurs only in certain morphological environments. Palatalization makes the following changes in consonants: *t→c, d→j*, *s|^s→x*, *z→Z*, *T|S|^S→C*, *n→N*, *l→y*. It is implemented in the files `pal_iE.fst` (where "8" represents the palatalization context on the lexical side) and `vn_pal.fst`. It occurs before the vowels *i* and *E* in the following contexts.

1. Imperfective and imperative second person singular feminine subject suffix *i*
   Example*: ber**t-i** → ber**ci*** (**በርቺ**)

2. Gerundive first person singular subject suffix *E* (in this case the preceding consonant is always geminated)
   Example: *kef**Il_-E** → kef**Iy_E*** (**ከፍዬ**)

3. Agentive suffix *i* (the *i* is usually dropped by rule 10 above if it ends the word)
   Example: *Cek_a**n-i** → Cek_a**Ni** (→ Cek_aN)* (**ጩካኝ**)

4. Instrumental suffix *ia* (the *i* is later dropped by rule 10 above)
   Example: *meCer_e**s-ia** → meCer_e**xia** (→ meCer_exa)* (**መጨረሻ**)

ii. Tigrinya

The following is a list of most of the Tigrinya phonological and orthographic rules that have been implemented in HornMorpho, along with the names of the files specifying the FSTs for the rules and one or more examples. This aspect of Tigrinya grammar is notoriously complex, and a number of details, in particular concerning the interaction of the different rules, have been omitted. Each of the rules is described as it applies in the generation direction, that is, from more abstract (lexical) to more concrete (surface). In the rule notation, "*C*" means any consonant; "*V*" means any vowel; "#" indicates the end of a word; "*$*" indicates a stem boundary; "_" indicates gemination of the preceding consonant; "|" represents alternatives. Unless otherwise indicated, the rules are obligatory. The first three rules apply only within verb stems; rule 4 applies at the boundary between stems and suffixes; rules 5, 6, and 7 apply within affixes; the other rules apply anywhere.

1. Within verb stems; *CWe → Co*, *CW(I) → Cu*; optional for *g*, *k*, *q*
   Implemented in `stem_lab.fst`
   ***mWe**leQe → **mo**leQe* (**ሞለቐ**)

2. Within verb stems; *Xo → XWe*, *Xu → XW*; *X*: *g*, *k*, *q*; optional
   Implemented in `stem_lab.fst`
   ***qo**me → **qWe**me* (**ቈመ**)

3. Within verb stems: *ew(e)C → oC*, *ey(e)C → eC*, *ewiC → oyC*, *eyiC → eyC*, *CweC → CuC*, *CyeC → CiC*
   Implemented in `stem_wy.fst`

*me**we**te* → ***mo**te* (**ሞተ**), *yI**Key**du* → *yI**Ke**du* (**ይኼዱ**), *k**ewi**nu* → *k**oy**nu* (**ኮይኑ**), *x**eyi**Tu* → *x**ey**Tu* (**ሸይጡ**), ***qwe**m* → ***qu**m* (**ቁም**), ***kye**du* → ***ki**du* (**ኪዱ**)

4. At stem-suffix boundaries: *ew$C* → *oC*, *ey$C* → *eC*, *ey$e* → *e*, *Iw${C|#}* → *u{C|#}*, *Iy${C|#}* → *i{C|#}*, *iw$V* | *iy$V* → *yV*, *iw$C* | *iy$C* → *iC*; vary in obligatoriness
   Implemented in `Vwy.fst`
   *fet**ewk**a* → *fet**ok**a* (→ *fetoKa*) (**ፈቶኻ**), *del**eyn**a* → *del**en**a* (**ደለና**), *teSeb_**eye**n_i* → *te-SeB_**e**n_i* (**ተጸበኒ**), *yIfet_**Iw*** → *yIfet_**u*** (**ይፈቱ**), *`aS**iwo**m* → *`aS**yo**m* (**ዓጽዮም**), *qen_**iyk**In* → *qen_**ik**In* (→ *qen_iKIn*) (**ቀኒኽን**)

5. Within affixes: *Iy(I)* → *i* | *I*, *Iye* → *e*, *uw* → *Iw*; the second rule optional
   Implemented in `CyC.fst`, `u2Iw.fst`
   *z**IyI**geb_Ir* → *z**i**geb_Ir, z**I**geb_Ir* (**ዚገብር**, **ዝገብር**), *’ayre’ay**uw**on* → *’ayre’ay**Iw**on* (**ኣይረኣይዎን**)

6. Within affixes: *ea* → *a*, *eo* → *o*, *ee* → *e*
   Implemented in `VV.fst`
   *re’ay**eo*** → *re’ay**o*** (**ረኣዮ**), *feleT**ee**n* → *feleT**e**n* (**ፈለጠን**)

7. Within affixes: *ey(yI)* → *E*, *Iye* → *E*; optional
   Implemented in `ey2E.fst`
   zeytensI’u → *zEtensI’u* (**ዜተንስኡ**), *zeyyISIg_Imek_a* → *zESIg_Imek_a* (**ዜጽግመካ**)

8. *CC* → *C_*
   Implemented in `gem.fst`
   *yIse**dd**u* → *yIse**d_***u* (**ይሰዱ**)

9. *Vk* → *VK*, *Vq* → *VQ* (unless *k* or *q* is geminated)
   Implemented in `KQ.fst`
   bekeye → *beKeye* (**በኸየ**), *yIwIs_Ik* → *yIwIs_IK* (**ይውስኽ**), *ziqIm_eT* → *ziQIm_eT* (**ዚቕመጥ**)

There is some variation in the vowels that occur before laryngeal consonants (*’*, *`*, *h*, or *H*) within verb stems. Some of this is handled in HORNMORPHO, but I am unclear on exactly what the rules should be. Examples: **ኣይከኣለን** *’ayk**e**’alen*, **ኣይካኣለን** *’ayk**a**’alen*, **ኣይክኣለን** *’ayk**I**’alen*.

iii. Oromo

The phonological rules of Oromo include the following implemented in HORNMORPHO. In the rule notation, “*C*” means any consonant; “*V*” means any vowel; “v” means a short vowel; “vv” means a long vowel; “+” means the boundary at the beginning or end of the root or stem; “|” represents alternatives.

The first four rules apply to verbs whose stems end in *’*, *w*, or *y*. Such verbs appear to be indistinguishable from one another when the stem is followed by a vowel; in these cases, the stem-final consonant may be realized as *’*, *w*, *y*, or *h* (Rule 1).[12] These three classes of stems are distinguished, however, when the stem is immediately followed by a consonant (Rules 2, 3, 4). Rules 5 and 6 represent two different ways the language has of handling clusters of three consonants.

[12] A more careful investigation may ultimately reveal more constraints.

1. *'+V | w+V | y+V → 'V | hV | yV | wV*
Implemented in `GS+.fst`
deebi'+a → deebiha | *deebiya* | *deebiwa* (as well as *deebi'a*); *beelaw+a → beelaha* | *beelaya* | *beela'a* (as well as *beelawa*); *dhagay+a → dhagaha* | *dhagawa* | *dhaga'a* (as well as *dhagaya*)

2. *+Cv'+C → CvvC*; *vvC(C)v'+C → vvC(C)vC*; *vC(C)v'+C → vC(C)vvC*
Implemented in `GS+.fst`
bu'+ta → buuta; *deebi'+ta → deebita; qufa'+ta → qufaata*

3. *Vy+t* | *Vy+s → eess*; *Vy+n → eeny*
Implemented in `wy+.fst`
dhagay+ta → dhageessa; *dhagay+sis+a → dhageessisa*;

4. *vw+C → ofC*; *vvw+C → oofC*
Implemented in `wy+.fst`
beelaw+ta → beelofta; *lakkaaw+ta → lakkoofta*

5. CC+C → CCiC
Implemented in `CCiC.fst`
arg+ta → argita; *kolf+ta → kolfita*

6. *r|lC*$_1$*+C*$_2$ *→ C*$_1$*ar|lC*$_2$
Implemented in `metath.fst`
arg+ta → agarta; *kolf+ta → kofalta*

7. *b+t → bd*; *d+t → dd*; *g+t → gd*
Implemented in `t2d.fst`
qab+ta → qabda; *eeg+ta → eegda*

8. *l+n → ll*; *r+n → rr*
Implemented in `lr_n.fst`
gal+na → galla; *bar+na → barra*

9. *t|d+n → nn*
Implemented in `T2n.fst`
bit+na → binna; *fid+na → finna*

10. *q+t → qx*
Implemented in `t2x.fst`
dhaq+ta → dhaqxa

11. *s+C → fC*
Implemented in `s2f.fst`
baas+ta → baafta

A further process in Oromo that is not really a phonological or orthographic rule is implemented as an alternation rule. There are a number of environments in which Oromo short vowels are lengthened, for example, at the end of yes/no questions: *beektuu?* 'do you (pl.) know?'. The lengthening is implemented at the lexical level as the special character "L", and the combination of a vowel with this character is realized as a long vowel in the FST `VL.fst`.

7. Implementation

a. FSTs and feature structure

Some of the code implementing finite state automata and feature structures is based on modules in the Natural Language Toolkit, an open source natural language processing project, accessible at `http://www.nltk.org/`.

b. Cascade of composed FSTs

HornMorpho includes separate lexical and guesser analyzer FSTs for Amharic and Tigrinya verbs and for Amharic nouns, stored in the files `Am/v.fst`, `Am/v0.fst`, `Ti/v.fst`, `Ti/v0.fst`, `Am/n.fst`, and `Am/n0.fst`. For Oromo verbs there are two lexical analyzer FSTs, one that outputs the verb stem, the other that outputs a segmentation of the input word, stored in `Om/v.fst` and `Om/v+.fst`. There are also separate lexical and (for Amharic and Tigrinya) guesser generator FSTs, stored in `Am/vG.fst`, `Am/v0G.fst`, `Ti/vG.fst`, `Ti/v0G.fst`, `Am/nG.fst`, `Am/n0G.fst`, and `Om/vG.fst`. Though an analyzer FST can be trivially inverted to yield a generator FST, in the program, the generator FSTs are created separately because they do not need to handle the phonological or orthographic variation that is required by the analyzer FST. For example, the lexical analyzer for Amharic verbs yields a single analysis for the alternate spellings **ሰርቆአል** and **ሰርቋል**, but the generator that starts with this analysis produces only **ሰርቋል**. For Amharic, there is an additional generator FST, `Am/vPG.fst`, that outputs a phonological, rather, than an orthographic representation of the word.

There are also lexical analyzers for the **copula** (**ነው**, **ናችሁ**; **እዩ**, **ኢና**, etc.) in Amharic and Tigrinya, treated separately because of its irregularities and stored in `Am/cop.fst` and `Ti/cop.fst`. For Amharic, the negative of the copula (**አይደለም**, **አይደለንም**, etc.) is also included in this FST; for Tigrinya, the negative of the copula (**ኣይኮነን**, etc.) is only recognized as the regular negative past perfective of the verb <*kwn*> **ኮነ** and not included in the copula FST. The past of the copula (**ነበረ**, **ነበር**, etc.) for both languages is treated only as the regular perfective of the verb <*nbr*>. The generators for the copula are created by inverting the analyzers; they are not stored separately in files.

Each of these 18 FSTs results from the composition of a **cascade** of simpler FSTs, each of which is responsible for an alternation rule, for the morphotactics of a word or stem belonging to a particular class, or for a list of lexical roots or stems. The overall structure of this cascade for the Amharic lexical verb FST is shown in Figure 3 and described in what follows. The architecture for Tigrinya verbs is similar. Oromo is simpler because there is no need for a separate cascade of FSTs to handle the verb stem. Each rectangle in the figure represents an FST; the symbol ".o." represents the composition operation (from lower to higher rectangle).

The files for the FSTs that are composed to make the complete FST are listed in in the cascade file `Am/v.cas`. By far the most important of these is the last one, the FST that covers the morphotactics of the verb, stored in the file `Am/vb_mtax.fst` and represented by the large yellow box in the figure. Within this there is a sequence of prefix and a sequence of suffix positions. These are concatenated onto the FST representing the verb stem (the darker yellow box). This is the output of the composition of an embedded cascade of FSTs, listed in the cascade file `Am/vb_stem.cas`. Two of the constituent FSTs (in red) are phonological, representing alternation rules that apply only to the verb stem. Three (in blue) are morphotactic, representing the structure of the stem, in particular how

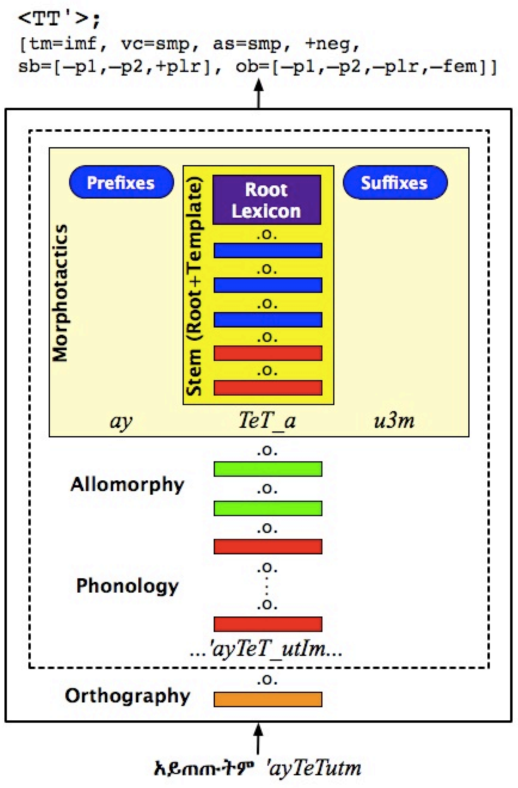

Figure 3. Cascade of FSTs in Amharic lexical verb analyzer

the root and template fit together, at three levels of abstraction. One (in purple) is the FST that encompasses all of the verb roots known to the system, stored in the lexical FST file `Am/vb_root.lex`. The FSTs making up the stem are composed in order from bottom (closer to the surface) to top (the lexical end). The verb stem FST can analyze a bare verb stem. For example,

(16) *teTeyay_q* → *Ty_q*; `[tm=ger,vc=ps,as=it]`

That is, the stem *teTeyay_q* has the root <*Ty_q*>, gerundive TAM, passive voice, and iterative stem-internal aspect.

The remaining FSTs implement alternation rules that apply beyond the stem, two for the first two allomorphic rules described in Section 6.a. (in green in the figure), eleven for the phonological rules described in Section 6.b. and a few other miscellaneous rules (in red), and one to convert between the phonological representation of the input that includes gemination and the vowel *I* and a purely orthographic representation that omits these features (in orange).

The figure illustrates the analysis of the word **አይጠጡትም** 'they don't drink it'. This is first romanized to *'ayTeTutm*. The orthographic-to-phonological FST converts this to all possible pronunciations of the input string, including the correct one, shown in the figure above the dashed border that surrounds the phonological part of the system. This form and the other surviving strings are processed by the intervening phonological FSTs. One of these replaces *u* with *au*; another replaces *t* with *3*, the character representing the third person singular masculine object suffix. Among the strings that make it to the morphotactic FST is *ayTeT_au3m*, which is analyzable as the pair of prefixes *a+y*, the stem *TeT_a*, and the three suffixes *u+3+m*. The final output analysis is shown at the top of the figure. The root of the word is <*TT_'*> 'drink', its TAM is imperfective, its polarity is negative, its voice and stem-internal aspect are simplex, its subject is third person plural (‘they’), and its object is third person singular masculine (‘him/it’).

The cascade of FSTs that are composed to form the Amharic lexical noun FST is similar, except that it includes a separate lexicon of simple noun stems such as **ቤት** (in the file `Am/n_stem.lex`) alongside the embedded cascade that is responsible for deverbal noun stems such as **ማድረግ** *madreg*.

The cascades for the Amharic noun and verb guesser FSTs are similar, except that they lack the stem and root lexicons that are part of the full lexical FSTs. There are also some restrictions on the root categories that the guesser FSTs recognize; this prevents massive ambiguity that would other occur. For Amharic they only accept verb roots in the canonical categories (CCC and CCCC, where none of the consonants are *’*, *w*, or *y*). Thus the Amharic verb guesser analyzer will analyze the word **ትዝርማላችሁ**, based on the imaginary root <*Zrm*>, but not the word **ትዝራላችሁ**, based on the imaginary root <*Zr’*>.

The lexical analyzers can deal with irregular words as well as those that obey the rules. One sort of irregularity is in the formation of verb stems. For example, the Amharic verb root <*drg*> has the alternate transitive imperfective stem *arg* (as in **ያርጋል**), alongside the regular stem *aderg* (as in **ያደርጋል**), and the Tigrinya verb root <*whb*> ‘give’ loses the *w* in most of its forms (**ሃበ**, **ይህብ**, **ይሃብ**, **ሂቡ**). Irregular stems are stored in separate files, `Am/irr_stem.lex` and `Ti/irr_stem.lex`, and the analyzer tries these stems along a separate path in the FST in parallel with the path implementing the regular stems.

For Amharic, constraints on the voice that occurs with particular roots are included within the root file, `Am/vb_root.lex`, in the form of FS sets. For example, along with the root <*qm_T*> is the feature set `{[vc=ps];[vc=cs];[vc=tr,as=it]}`, representing the constraint that this root can only occur in the passive and causative voice and the iterative transitive voice.

c. DATA FILES

Data used by HORNMORPHO appear in three file types, distinguished by their extensions: `.fst`, files describing FSTs explicitly in terms of states and transitions; `.cas`, files specifying cascades of FSTs to be composed; `.lex`, files containing lists of roots, stems, or complete words, to be converted to FSTs. Any of these files may contain lines with comments; these begin with “#”.

The only files needed to run the program are the FST files containing the fully composed analyzers and generators for verbs, nouns, and copulas. The other files are included in the distribution for those who would like to use the grammatical and lexical information that is contained in them for other purposes.

i. FST files

An FST (`.fst`) file implements one or more alternation rules or the morphotactics for a word class in the form of an FST. Each uncommented line in an FST line takes one of the following forms:

- `-> STATE`
  Makes `STATE` the initial state of the FST.
- `STATE ->`
  Makes `STATE` a final state of the FST.

- `STATE1 -> STATE2  CONDITIONS  FSSet`
  Creates one or more transitions from `STATE1` to `STATE2`. `CONDITIONS` has one of the following forms.

  - `[IO1;IO2;...]`
    Each of `IO*` represents the condition on a transition from `STATE1` to `STATE2`. It has one of the following forms:

    - `CHAR1:CHAR2`
      `CHAR1` and `CHAR2` are either characters from the alphabet of romanized Amharic or Tigrinya characters, a character set abbreviations (defined in the cascade file for this FST file), or nothing (the empty character).
      Examples:

`[i:y]`	input character *i* corresponds to output character *y*
`[X:]`	any input consonant (represented by the character set abbreviation "X") corresponds to nothing in the output
`[:]`	no input character is consumed and no output generated on this transition

    - `CHAR`
      `CHAR` is either a character from the alphabet or a character set abbreviation. This represents identical input and output characters.
      Examples:

`[i]`	input and output characters are both *i*
`[X]`	input and output characters are the same consonant

  - `<CHARS:> ; <CHARS> ; <CHARS1:CHARS2> ; <CHARS1(CHARS2)CHARS3:CHARS4>`
    `CHARS`, `CHARS1`, etc. are sequences of characters from the alphabet. This represents a sequence of states, each with a transition into it with one of the characters as the input character. With nothing following the colon, there is no output character. With no colon, the output sequence is the same as the input. Characters following the colon specify output characters. Parentheses on the input side specify optional transitions.
    Example:
    `STATE1 -> STATE2  <et:>`
    equivalent to
    ```
    STATE1 -> STATE1.2  [e:]
    STATE1.2 -> STATE2  [t:]
    ```

  `FSSet`, which is optional, represents any FS constraints which apply to the transition. Each FS has the form `[FV1, FV2, ...]`. Each `FV*` represents a single feature-value pair. It has one of these forms

  - `+FEATURE, -FEATURE`
    These are equivalent to `FEATURE=True` and `FEATURE=False` respectively.

  - `FEATURE=VALUE`
    Values is either a string or a full FS.

  If there are multiple FSs in the set, they are separated by semicolons. The interpretation of multiple FSs is that at least one of the FSs must unify with the current accumulated FS at the point an attempt is made to traverse the transition.

- `STATE1 -> STATE2  >>FST<<`
  `FST` is the name of an FST file (without the `.fst` extension). The FST described in this file is to be inserted between `STATE1` and `STATE2`.

- `STATE1 -> STATE2  +LEX+`
  `LEX` is the name of a lexical file (without the `.lex` extension). The FST represented by the list of roots, stems, or words in the file is inserted between `STATE1` and `STATE2`.

HORNMORPHO currently has no facility for compiling FSTs from abstract descriptions of linguistic rules or from regular relations, as is possible in most other FST packages, such as OpenFst (<http://www.openfst.org/>) and the Xerox finite state tools (<http://www.stanford.edu/~laurik/fsmbook/home.html>). In later versions of the program, this capability may be added.

ii. Cascade files

A cascade file (`.cas`) gives a list of FST to be composed to create a single FST. Each of the FSTs is represented by the name of an FST or lexical file. Each of these appears on a separate line without its extension and surrounded by “>...<” for FST and “+...+” for lexical files.

Before the list of files, a cascade file may also contain a list of character set abbreviations. Each abbreviation is a character or sequence of characters (for example, `X`, `V`, `!X`, `JJ`). The specification for each abbreviation appears on a separate line. It begins with the abbreviation, is followed by “=” and the list of characters in the set, separated by commas, and surrounded by “`{}`”, for example,

(17) `JJ = {d, l, n, s, t, T, z}`

iii. Lexical files

A lexical file (`.lex`) contains a list of lexical forms (roots, stems, or complete words) which are to be converted to an FST that accepts the forms as sequences of input characters. Each uncommented line has one of the following forms.

- `FORM`
  `FORM` is a string representing a root, stem, or word. It is incorporated into the FST as a sequence of states joined by transitions with identical input and output characters.

- `FORM    ROOT    FSSet`
  `FORM` is a string representing a root, stem, or word. `ROOT` is the root that is to be associated with `FORM`. In this case, the `FORM` is incorporated into the FST as a sequence of states joined by transition with different input and output characters; that is, `FORM` as input results in `ROOT` as output. `FSSet` specifies a set of FS constraints that are associated with `FORM`. `FSSet` is optional. If `FSSet` appears, and no separate root is to be specified, the second position is filled by the empty string ''. An example, from the file `Am/irr_stem.lex`:

  (18) `tegenaNt   gN*   [tm=ger,vc=ps,as=it]`

  This indicates that the form *tegenaNt* is the passive, iterative, gerundive stem of the Amharic verb with the root <*gN**> **እገኝ**. (This is irregular because we would normally expect *tegeNaNt*.)

  Roots which are regular but which fail to occur with certain voice features are also notated with FS constraints in the lexicon file where they appear. An example, from the file containing regular Amharic verb roots, `Am/vb_root.lex`:

(19) `ds_t   ''   [vc=ps];[vc=cs];[v=man,pos=n]`

This indicates that the Amharic root <*ds_t*> **ተደሰተ** occurs only in the passive and causative voices or the manner form for nouns.

8. Use

a. Starting the program

To use HornMorpho, you must be running the Python interpreter. There are several ways to do this; you can run Python as a self-standing program or from within an integrated development environment, such as IDLE, the one that comes with the Python distribution. If you want to use the capability of the program to process Ge'ez characters within the Python interpreter (only possible with Linux and MacOS), you should not use IDLE because it currently fails to handle Unicode characters properly. Before you go any further, be sure your computer has version 3.0 or 3.1 of Python. If not, you can download version 3.1 from the Python site: <http://www.python.org/download/>. (The program will not run under Python 2.*. If for some reason you can't or don't want to run Python 3, use version 1.01 of HornMorpho, which runs under Python 2.5 and 2.6.)

To start up Python on computers running Unix (including Macintosh computers) or Linux, you type `python` in a shell. On computers running Windows, you start the interpreter either by selecting Run in the Start menu and then typing `python`, by double-clicking on the Python application icon, or by typing `python` in a command prompt window. (If the third option fails, you will need to change your `path` variable so that it knows where to find Python. How to do this is beyond the scope of this document.)

If you (or a system administrator) have installed HornMorpho on your computer, you can run Python from anyplace in the file system. If not, you need to run it in the `HornMorpho-2.0` directory. You can either change to this directory in the shell before you start up Python, or you can change directories within Python after you start it up. To change directories within Python, first import the `os` module, then use the `os.getcwd()` command to see where you are, and the `os.chdir()` command to move around. For example,

```
>>> import os
>>> os.chdir('../../Projects/HornMorpho2.0')
>>> os.getcwd()
'/Users/gasser/Projects/HornMorpho2.0'
```

On a Windows machine, you would use "`\\`" instead of "`/`" for the directory separator.

If HornMorpho is installed on your computer or you are in the `HornMorpho-2.0` directory, you should be able to import all the files you need.

If the `import` statement fails, you will probably see this message: `ImportError: No module named l3`. This happens because the path that the interpreter searches doesn't include the current directory. To fix this, do the following:

```
>>> import sys; sys.path.append('')
```

Now the `import` statement should work.

HORNMORPHO has five functions. Of these the most useful for most users will be `anal_file`, which analyzes the Amharic, Oromo, or Tigrinya words contained in one file, writing the analysis to another file. If you only want to use this function, you can skip to section d. The other functions analyze or generate individual words; by default these expect words to be written in Ge'ez characters for Amharic and Tigrinya. That is, for these languages, the Python interpreter must be able to handle Unicode, both as input and output. As far as I can tell, this is not yet possible under Windows, but it is straightforward under Linux and MacOS. The next section explains what you need to know to make the Python interpreter deal with Ge'ez under these operating systems.

b. GE'EZ CHARACTERS IN THE PYTHON INTERPRETER

When you start Python in a shell, the characters that Python can recognize and print out depend on the character encoding of the terminal program you are running. You need to make sure that the encoding is set to UTF-8 in order for Ge'ez characters to show up properly in the Python interpreter.

In Linux, UTF-8 is usually the default, in which case you won't have to set anything. If you do have to change the setting, what you do will depend on the terminal program you're running. In RedHat Linux, for example, you would be running a program called Terminal, which has a menu called "Terminal" with a menu item called "Set character encoding".

To set the character encoding in MacOS, start up the Terminal program; then go to the Preferences menu item (in the Terminal menu). You'll see different configuration of Terminal windows on the left; select the configuration that you'd like to set the character encoding for. Click on the Advanced tab. At the bottom of the window, under "International", you'll see a a list titled "Character encoding". Select "Unicode (UTF-8)". Then close the Preferences window. Now when you open a new Terminal window (shell) with the configuration that you changed, it should be able to display Ge'ez characters.

You can test your shell without running Python by typing something like the following ("`%`" is the prompt).

```
% echo 'ፈተና'
ፈተና
```

Assuming you have a Unicode Ge'ez font on your computer, you should see what appears above.

c. ANALYZING A WORD

To analyze a single word, enter the following at the Python interpreter prompt:

```
l3.anal_word(language, word)
```

where *`language`* is a Python string representing the language you want to analyze, either `'am'`, `'om'`, or `'ti'`, and *`word`* is a word, in Ge'ez characters for Amharic and Tigrinya, and enclosed in quotation marks (either single or double). If you are using Windows or running Python in a shell that is not set up to handle Unicode characters, you can type an Amharic or Tigrinya word in romanized form instead, setting the `roman` option to `True`.

```
l3.anal_word(language, word, roman=True)
```

In the romanization you should not include gemination or the vowel *I*.

If you have not called `anal_word` or one of the other functions to be described below for `language`, the program will first have to load the data it needs to perform the analysis. Because the data files are very large, it could take perhaps as long as a minute for this to complete.

After the data are loaded, the program checks to see whether the word is in its lexicon of unanalyzed words; in these cases, it just prints out the word.[13]

(20)
```
>>> l3.anal_word('ti', 'ናብ')
Word: ናብ
```

The romanized alternative would be as follows.

(21)
```
>>> l3.anal_word('ti', 'nab', roman=True)
Word: nab
```

Some of these "unanalyzed" words could be analyzed into constituent morphemes, for example, the Amharic adverb **ይልቁንም**, but because they function as idiomatic semantic wholes, such an analysis would not be particularly useful.

If the word is not in the unanalyzed lexicon, the program checks its lexicon of pre-analyzed words, words which either have irregular morphology or are very common. If it is not there, the program attempts to analyze it using one of its FSTs, first with lexical FSTs for verbs and nouns that contain lists of noun stems and verb roots, then (for Amharic and Tigrinya) with "guesser" FSTs that attempt to guess an unknown stem or root based on the prefixes and suffixes and the shape of the possible stem.

If the program succeeds in analyzing a word, it will print out the root or stem, the citation form, and the grammatical analysis. Here are examples of the analysis of relatively complex verbs in all three languages.

(22)
```
>>> l3.anal_word('am', 'የማያስፈልጋትስ')
Word: የማያስፈልጋትስ
POS: verb, root: <fl_g>, citation: አስፈለገ
 subject: 3, sing, masc
 object: 3, sing, fem
 grammar: imperfective, causative, relative, negative
 conjunctive suffix: s
>>> l3.anal_word('ti', 'ብዘጋጥመና')
Word: ብዘጋጥመና
POS: verb, root: <gTm>, citation: ኣጋጠመ
 subject: 3, sing, masc
 object: 1, plur
 grammar: imperfective, reciprocal, transitive, relative
 preposition: bI
>>> l3.anal_word('om', 'afeeramaniiru')
Word: afeeramaniiru
POS: verb, root: <afeer>, citation: afeeramuu
 subject: 3, plur
 derivation:  passive
 TAM: perfect
```

[13] For Oromo, there is currently no list of unanalyzed words.

The first line in the analysis just repeats the word. The second line gives the part of speech, the root of the word, and the citation form. For Amharic and Tiginya verb root categories and their representation in HORNMORPHO, see Section 5.a.ii above. The citation form used for Amharic and Tigrinya incorporates the voice and stem-internal aspect of the input word (see Section 5.a.i for a discussion of these grammatical features). Thus what appears here is **አስፈለገ** for the Amharic example and **ኣጋጠመ** for the Tigrinya example rather than the more basic forms **ፈለገ** and **ገጠመ**, which have the value `simplex` for both voice and stem-internal aspect. For Oromo the citation form is the infinitive, incorporating any derivational features of the input word, here the fact that it is passive. On the following lines is the grammatical analysis of the word, beginning with the features of the verb's subject and, for Amharic and Tigrinya, also its object. See Section 5 for details on the various grammatical features.

Here is an example of the analysis of an Amharic noun that is not derived from a verb (more precisely, a noun that the program does not *know* is derived from a verb).

```
(23) >>> l3.anal_word('am', 'ከየኃላፊዎቻቸው')
     Word: ከየኃላፊዎቻቸው
     POS: noun, stem: ኃላፊ
      possessor: 3, plur, masc
      grammar: plural, definite, distrib(Iyye-)
      preposition: ke
```

Since there is no known verb root for the word, only the stem is given in the second line: **ኃላፊ**. The third line gives features of the possessor; it is third person plural. The fourth line gives miscellaneous grammatical properties—the noun is plural, definite and distributive (that is, it has the prefix **እየ** *'Iyye-*)—and the fifth line gives any prepositional, conjunctive, or adverbial prefixes or suffixes, in this case the preposition **ከ** *ke-*.

Here is an example of the analysis of an Amharic noun that is derived from a verb.

```
(24) >>> l3.anal_word('am', 'ባለማጠናቀቃችን')
     Word: ባለማጠናቀቃችን
     POS: infinitive, root: <Tnqq>, citation: አጠናቀቀ
      possessor: 1, plur
      grammar: reciprocal, transitive, negative
      preposition: be
```

The part of speech that is given in the second line of the analysis of a deverbal noun is either infinitive, agentive noun, instrumental noun, or manner noun; the root given is the verb root; and the citation form is that of the verb, incorporating the voice and stem-internal aspect of the input word (in the example, transitive voice and reciprocal aspect).

If `anal_word` relies on one of its guesser analyzers, the second line of the analysis is preceded by a question mark. In the following Tigrinya example, the verb guesser analyzer is called for because the verb root <*mnTl*> is not in the lexical analyzer's lexicon.

```
(25) >>> l3.anal_word('ti', 'እናመንጠለት')
     Word: እናመንጠለት
     ?POS: verb, root: <mnTl>, citation: መንጠለ
      subject: 3, sing, fem
      grammar: perfective
      conjunctive prefix: Inna
```

If `anal_word` fails to analyze the word, then a question mark precedes the single line that is printed out.

```
(26) >>> l3.anal_word('ti', 'ፔፕሲ')
     ?Word: ፔፕሲ
```

If multiple analyses are possible, they are all printed out by `anal_word`.

```
(27) >>> l3.anal_word('am', 'ነገራችሁ')
     Word: ነገራችሁ
     POS: noun, stem: ነገር
      possessor: 2, plur
     POS: verb, root: <ngr>, citation: ነገረ
      subject: 3, sing, masc
      object: 2, plur
      grammar: perfective
     POS: verb, root: <ngr>, citation: ነገረ
      subject: 2, plur
      grammar: perfective
     >>> l3.anal_word('om', 'dubbanne')
     Word: dubbanne
     POS: verb, root: <dubbadh>, citation: dubbachuu
      TAM: past, negative
     POS: verb, root: <dubbadh>, citation: dubbachuu
      subject: 1, plur
      TAM: past
```

Sometimes what appears to be ambiguity results from the presence of a complex stem in the lexicon that can also be analyzed into a root and derivational affixes. This is especially true for Oromo verbs and Amharic nouns.

```
(28) >>> l3.anal_word('om', 'argadhe')
     Word: argadhe
     POS: verb, root: <arg>, citation: argachuu
      subject: 1, sing
      derivation:  autobenefactive
      TAM: past
     POS: verb, root: <argadh>, citation: argachuu
      subject: 1, sing
      TAM: past
```

The program returns two analyses in this case because its lexicon contains the complex stem *ar-gadh-* 'find', which is the autobenefactive form of the root *arg-* 'see', as an unanalyzed stem, but it also knows how the autobenefactive form of *arg-* is realized. The two analyses represent two separate paths through the FST.

Amharic and Tigrinya have two verbs which behave like no others, the copula (**ነው**; **እዩ**, etc.) and the verb of existence (**አለ**, **የለም**; **ኣሎ**, **የለን**, etc.). HORNMORPHO outputs <*ne*> for Amharic and <*'y*>

for Tigrinya as the root of the copula and <*al_e*> as the root of the verb of existence. Here are two examples. The Oromo copula forms (*dha*, *ti*, *miti*) are not currently handled by the program.

(29)
```
>>> l3.anal_word('am', 'አይደለችም')
Word: አይደለችም
POS: copula, root: <ne>
 subj: 3, sing, fem
 negative
>>> l3.anal_word('am', 'የሌለባችሁ')
Word: የሌለባችሁ
POS: verb, root: <al_e>, citation: አለ
 subject: 3, sing, masc
 object: 2, plur, prep:-b-
 grammar: present, relative, definite, negative
>>> l3.anal_word('ti', 'ድዮም')
Word: ድዮም
POS: copula, root: <'y>
 subject: 3, plur, masc
 grammar: yes/no
>>> l3.anal_word('ti', 'ዘየብለይ')
Word: ዘየብለይ
POS: verb, root: <al_e>, citation: ኣሎ
 subject: 3, sing, masc
 object: 1, sing
 grammar: present, relative, negative
```

If you would like to suppress the root/stem, citation form, and/or grammatical analysis that are printed out by `anal_word`, you can do this by specifying `False` for the options `root`, `citation`, and/or `gram`.

(30)
```
>>> l3.anal_word('am', 'ቢያስጨንቁአቸው', root=False, gram=False)
Word: ቢያስጨንቁአቸው
POS: verb, citation: አስጨነቀ
>>> l3.anal_word('om', 'teenya', root=False)
Word: teenya
POS: verb, citation: taa'uu
 subject: 1, plur
 TAM: present
>>> l3.anal_word('ti', 'ዝፈልሰፉ', gram=False)
Word: ዝፈልሰፉ
?POS: verb, root: <flsf>, citation: ፈልሰፈ
```

Instead of printing out a description of the grammatical properties of the word, you can have `an-al_word` return the raw output of the transducers by setting the optional `raw` parameter to `True`. The output is a list of Python tuples, each consisting of a (string) root or stem and a feature structure description. (The format of the feature structure descriptions is beyond the scope of this document, but see Section 8.e for some more information.)

(31)
```
>>> l3.anal_word('am', 'ለዘመዶቻችንም', raw=True)
[('zemed', [-acc, cnj='m', der=[-ass], -dis, +plr, pos='n',
poss=[+expl, +p1, -p2, +plr], pp='le', rl=[-acc, +p], v=None])]
```

For Oromo only, you can have the program print out a segmentation of the input word rather than its root by specifying `True` for the `segment` option of `anal_word`.

```
(32) >>> l3.anal_word('om', 'afeeramaniiru', segment=True)
     Word: afeeramaniiru
     POS: verb, segmentation: {afeer-am}-an-r-u
      subject: 3, plur
      derivation: passive
      TAM: perfect
     >>> l3.anal_word('om', 'dubbanne', segment=True)
     Word: dubbanne
     POS: verb, segmentation: {dubbadh}-ne
      TAM: past, negative
     POS: verb, segmentation: {dubbadh}-n-e
      subject: 1, plur
      TAM: past
```

In the segmentation, the stem is surrounded by braces, and other morphemes are separated by hyphens. If you want only the segmentation and no grammatical analysis, you can set the `gram` option to `True`.

For Amharic only, you can convert an input word in Ge'ez characters to a phonetic representation of the word with the function `phon_word`.

`l3.`**`phon_word`**`('am', `*`word`*`)`

The phonetic representations printed out by `phon_word` conform to the romanization conventions described in indicate which consonants are geminated and the presence of the sixth order vowel, symbolized by "*I*".

```
(33) >>> l3.phon_word('am', "ይመታሉ")
     yImetal_u yIm_et_al_u
     >>> l3.phon_word('am', "አነጋገራቸው")
     an_egag_erac_ew an_egagerac_ew
```

By default, phon_word just prints out the possible phonetic representations. If you want the corresponding grammatical analyses, set the `gram` option to `True`.

```
(34) >>> l3.phon_word('am', "ይመታሉ", gram=True)
     -- yImetal_u
     POS: verb, root: <mt'>
      subject: 3, plur
      grammar: imperfective, aux:alle
     -- yIm_et_al_u
     POS: verb, root: <mt'>
      subject: 3, plur
      grammar: imperfective, aux:alle, passive
```

d. ANALYZING A FILE

To analyze all the words in a file and write the analyses to another file, enter the following at the Python interpreter prompt.

```
l3.anal_file(language, input_file, output_file)
```

where *`language`* is again either `'am'`, `'om'`, or `'ti'`, *`input_file`* is a file name or a path to a file containing Amharic, Oromo, or Tigrinya text, and *`output_file`* is a file name or a path to an output file. Amharic and Tigrinya must appear in Unicode Ge’ez. All arguments be surrounded by quotation marks. The file at *`input_file`* must exist; otherwise the program reports an error. The file at *`output_file`* need not exist, but if *`output_file`* is a path to a file in a directory, the directory must exist, and the user must have permission to write to a file there.

The function *`anal_file`* segments the input file into words and attempts to analyze each of these, using *`anal_word`*. It writes the result of the analyses to the output file. Punctuation marks, other than ‘-’, which can be word-internal in Amharic, Oromo, and Tigrinya, are separated and treated as unanalyzed words. No attempt is made to analyze Amharic or Tigrinya words that contain non-Ge’ez characters or Ge’ez numerals.

Assume that following sentence (the first sentence from the new preface to Abe Gubegna’s novel **አልወለድም**) is contained in the file `am.txt`, which is located in a directory called `Data` in the `l3` directory under the directory where Python is running (this file is included with the distribution in that location).

ይህ መጽሐፍ የዛሬ 01 **ዓመት ገደማ በደንቡ ምርመራ አልፎ ታትሞ በወጣ ጊዜ ታላቅ ችግር ፈጥሮብኝ ነበር ።**

Then the following will analyze this sentence, writing the analysis to a file in the same directory called `alweledm_out.txt`. (You would use ‘`\\`’ instead of ‘`/`’ in the paths in Windows.)

```
(35a) >>> l3.anal_file('am', 'l3/Data/am.txt', 'l3/Data/am_out.txt')
      Analyzing words in l3/Data/am.txt
      Writing analysis to l3/Data/am_out.txt
```

The contents of `am_out.txt` are then as follows.

```
(35b) Word: ይህ
      POS: noun, stem: ይህ

      Word: መጽሐፍ
      POS: noun, stem: መጽሐፍ

      Word: የዛሬ
      POS: noun, stem: ዛሬ
       grammar: genitive
      POS: noun, stem: ዛር
       possessor: 1, sing
       grammar: genitive

      Word: 01

      Word: ዓመት
      POS: noun, stem: ዓመት

      Word: ገደማ
      POS: noun, stem: ገደማ
```

```
Word: በደንቡ
POS: noun, stem: ደንብ
 possessor: 3, sing, masc
 preposition: be
POS: noun, stem: ደንብ
 grammar: definite
 preposition: be

Word: ምርመራ
POS: noun, stem: ምርመራ

Word: አልፎ
POS: verb, root: <'lf>, citation: አለፈ
 subject: 3, sing, masc
 grammar: gerundive
POS: verb, root: <l'f>, citation: አላፈ
 subject: 3, sing, masc
 grammar: gerundive, transitive

Word: ታትሞ
POS: verb, root: <'t_m>, citation: ታተመ
 subject: 3, sing, masc
 grammar: gerundive, passive

Word: በወጣ
POS: verb, root: <wT'>, citation: ወጣ
 subject: 3, sing, masc
 grammar: perfective, relative
 preposition: be

Word: ጊዜ
POS: noun, stem: ጊዜ

Word: ታላቅ
POS: noun, stem: ታላቅ
POS: verb, root: <lqq>, citation: አላቀቀ
 subject: 3, sing, fem
 grammar: jussive/imperative, reciprocal, transitive

Word: ችግር
POS: noun, stem: ችግር

Word: ፈጥሮብኝ
POS: verb, root: <fTr>, citation: ፈጠረ
 subject: 3, sing, masc
 object: 1, sing, prep:-b-
 grammar: gerundive

Word: ነበር
```

```
        Word: ።
```

To produce representations of the Amharic words in a file, use the function `phon_file`, which works like `phon_word`, described in Section 8.c above.

```
l3.phon_file('am', input_file, output_file)
```

Here is an example, using the same file as in the previous example.

```
(36a) >>> l3.phon_file('am', 'l3/Data/am.txt', 'l3/Data/am_phon.txt')
      Analyzing words in l3/Data/Am/am.txt
      Writing analysis to l3/Data/Am/am_phon.txt
```

This will produce the following in the file `am_phon.txt`:

```
(36b) ይህ yIh
      መጽሐፍ meShaf
      የዛሬ yezarE
      01
      ዓመት amet
      ገደማ gedema
      በደንቡ bedenbu
      ምርመራ mIrmera
      አልፎ alfo
      ታትሞ tat_Imo
      በወጣ beweT_a
      ጊዜ gizE
      ታላቅ tal_aq_ tal_aq
      ችግር cIg_Ir
      ፈጥሮብኝ feTrob_IN_
      ነበር neb_er
      ።
```

To include grammatical analyses for each pronunciation, set the `gram` option to `True`.

e. GENERATING A WORD

You can also use HORNMORPHO to generate words, given their root or stem and grammatical features. In order to do this, you will need to be familiar with the way verb roots and grammatical features are represented in the program.

In its simplest form, the generation function looks like this:

```
l3.gen(language, root/stem)
```

where *`language`* is again either `'am'`, `'om'`, or `'ti'` and *`root/stem`* is a noun stem or verb root, romanized for Amharic and Tigrinya. In this form, the function generates the wordform that conforms to a default set of features associated with the part-of-speech for the word. For verbs, the default grammatical structure is as follows.

- subject: third person masculine singular; no object
- tense/aspect/mood: perfective for Am/Ti; past for Om

- voice (Am/Ti): simplex; stem-internal aspect (Am/Ti): simplex
- affirmative, non-relative
- no prepositional, conjunctive, or adverbial affixes

Here are examples of verbs with the default structure.

(37)
```
>>> l3.gen('am', "mWl'")
ሞላ
>>> l3.gen('om', 'sirb')
sirbe
>>> l3.gen('ti', "gWyy")
ጎየየ
```

That is, **ሞላ** *mol_a* is the third person masculine singular perfective form of the verb with the root <*mWl'*>. Note that the root is surrounded by quotation marks, which in the first case must be double because the root contains the single quotation character. For more on the representation of Amharic and Tigrinya verb roots in HORNMORPHO, see Section 5.

If you are using Windows or running Python in a shell that is not set up for Unicode, you can have this function produce romanized output for Amharic and Tigrinya by setting the `roman` option to `True`.

(38)
```
>>> l3.gen('am', "mWl'", roman=True)
mola
```

For nouns (Amharic only), the default grammatical structure is as follows.

- no possessor
- singular number, indefinite
- no prepositional prefix, genitive prefix, or conjunctive/adverbial suffix
- not derived from a verb

Thus if you give the generation function a noun stem and no grammatical specification, the function just returns the stem unchanged.

(39)
```
>>> l3.gen('am', "meng^st")
መንግሥት
```

To specify grammatical structure other than the default, you put this after the root or stem (with a comma in between), using abbreviated versions of the feature structure representation described in Sections 4.b. and 7.c.i. above. Each feature-value pair that you include represents a change in the default set of features. Feature structures are enclosed in brackets and surrounded by quotation marks. A simple feature-value pair is represented by the feature and the value separated by an equals sign. For example, if you want the passive form of an Amharic or Tigrinya verb, with no other features in the default changed, you would set the `voice` feature (`vc`) to `passive` (`ps`).

(40)
```
>>> l3.gen('am', "mWl'", '[vc=ps]')
ተሞላ
>>> l3.gen('ti', 'HSb', '[vc=ps]')
ተሐጽበ
```

Here is an Oromo example, in which the only change is the tense.

(41)
```
>>> l3.gen('om', 'sirb', '[tm=prs]')
sirba
```

Values of `True` and `False` can be represented using `+` or `-` signs before the feature name. For example, to make an Amharic noun definite (`def`) and plural (`plr`), you would do the following.

(42)
```
>>> l3.gen('am', "meng^st", '[+plr,+def]')
መንግሥታቱ
```

To specify the subject or object of a verb, you will need a nested feature structure because the value of the subject and object features is itself a feature structure. For example, to make the subject (`sb`) of a verb second person (`p2`) singular feminine (`fem`) and the object (`ob`) third person plural (`plr`), you would do the following.

(43)
```
>>> l3.gen('am', "mWl'", '[sb=[+p2,+fem],ob=[+plr]]')
ሞላሻቸው
>>> l3.gen('ti', 'HSb', '[sb=[+p2,+fem],ob=[+plr]]')
ሐጸብክዮም
```

Here's an Oromo example with a third person singular feminine subject.

(44)
```
>>> l3.gen('om', 'sirb', '[sb=[+fem],tm=prs]')
sirbiti
```

Amharic and Tigrinya object suffixes may also be indirect. To generate an indirect object in Tigrinya, you use the object feature `+prp`. For Amharic, you have to specify one of the features `+b` or `+l`, depending on the indirect object prefix.

(45)
```
>>> l3.gen('am', "mWl'", '[sb=[+p2,+fem],ob=[+plr,+l]]')
ሞላኸላቸው
>>> l3.gen('ti', 'HSb', '[sb=[+p2,+fem],ob=[+plr,+prp]]')
ሐጸብክሎም
```

To generate an Amharic noun that is derived from a verb, use `gen` with a verb root, and, in addition to the part-of-speech (`pos=n`), specify which category of deverbal noun you want as the value of the `v` feature: infinitive (`inf`), agentive noun (`agt`), instrumental noun (`ins`), or manner noun (`man`). For all but the manner noun, you may also specify values of the voice (`vc`) and stem-internal aspect (`as`) features.

(46)
```
>>> l3.gen('am', "sdb", '[pos=n,v=agt,vc=cs,as=rc]')
አሳዳቢ
```

The generation function does not normally use the guesser FSTs; if you'd like to force the use of the guesser FSTs (for Amharic and Tigrinya only), use the option `guess=True`.

(47)
```
>>> l3.gen('am', 'kongo', '[pp=be]')
This word can't be generated!
>>> l3.gen('am', 'kongo', '[pp=be]', guess=True)
በኮንጎ
```

Amharic noun stems must be romanized and must include gemination, even though this is not indicated in the standard orthography. The vowel *I* should not be included.

(48)
```
>>> l3.gen('am', 'wddr', '[+gen, poss=[+p1,+plr]]')
This word can't be generated!
>>> l3.gen('am', 'wdd_r', '[+gen, poss=[+p1,+plr]]')
የውድድራችን
```

To see the full list of features and possible values for a language and part-of-speech, use the function `get_features`.

`l3.`**`get_features`**`(`*`language`*`, part_of_speech)`

This function returns a Python dictionary whose keys are grammatical features and whose values are lists of possible values for the features in the case of simple features and dictionaries of features and possible values in the case of complex features such as subject (`sb`).

(49)
```
l3.get_features('ti', 'v')
{'vc': ['tr', 'smp', 'ps'], 'yn': [True, False], 'pos': ['v'],
'as': ['smp', 'rc', 'it'], 'sub': [True, False], 'pp': ['sIle',
'kem', 'nI', 'ab', 'Inte', 'nab', 'kab', 'bI'], 'd': [True,
False], 'neg': [True, False], 'ob': {'p2': [True, False], 'p1':
[True, False], 'plr': [True, False], 'xpl': [True, False],
'fem': [True, False], 'prp': [True, False]}, 'tm': ['ger',
'imf', 'j_i', 'prf', 'prs'], 'rel': [True, False], 'cj2': ['n',
's', 'ke', 'do', 'Immo'], 'sb': {'p2': [True, False], 'p3':
[True, False], 'fem': [True, False], 'p1': [True, False], 'plr':
[True, False]}, 'cj1': ['InkI', 'kI', 'Inte', 'mIs', 'nI', 'mI',
'nIKI', 'Inna']}
```

For reference, here is a more or less complete list of abbreviations for features and values that you may need:

- subject: `sb`; object: `ob`; possessor (nouns): `poss`
- 1st person: `p1`; 2nd person: `p2`; feminine: `fem`; plural: `plr`, `pl` (Oromo); formal: `frm`; prepositional object: `l`, `b` (Amharic), `prp` (Tigrinya)
- tense/aspect/mood: `tm`; perfective/perfect: `prf`, imperfective: `imf`, jussive/imperative: `j_i`, gerundive/gerund: `ger`, past: `pst`, present: `prs`, contemporary: contemp, imperative: `imv`; participle: `prt`
- voice: `vc`; simplex: `smp`, passive-reflexive: `ps`, transitive: `tr`, causative (Amharic): `cs`
- derivation (Oromo): `der`; passive: `ps`, causative: `cs`, autobenefactive: `autoben`
- stem-internal aspect: `as`; simplex: `smp`, reciprocal: `rc`, iterative: `it`
- noun case (Oromo): `case`; base: `bs`, subject: `sb`, dative: `dat`, instrumental: `ins`, locative: `loc`, ablative: `abl`
- genitive (nouns): `gen`; definite (Amharic): `def`; relative: `rel`; negative: `neg`; accusative (Amharic): `acc`; distributive (Amharic, *Iyye-*): `dis`
- auxiliary (Amharic): `ax`; alle: `al`
- prefix conjunction: `cj1`; suffix conjunction (verbs): `cj2`; suffix conjunction (nouns): `cnj`
- preposition: `pp`
- miscellaneous (Oromo): first person subject: `1s_sb`; infinitive: `inf`, agent: `agt`, feminine: `fem`, continuative (interrogative, non-final past or imperative): `cont`

More examples:

(50)
```
>>> l3.gen('am', "ngr", '[pp=ke,tm=imf,vc=ps,as=it,cj2=m]')
ከሚነጋገርም
```

```
>>> l3.gen('am', 'ne', '[+neg, sb=[+p1,+plr]]')
አይደለንም
>>> l3.gen('om', 'sob', '[der=[+autoben],sb=[+p2],+neg,tm=prs]')
sobattu
>>> l3.gen('om', 'barbaad', '[+inf,cnj=f]')
barbaaduuf
>>> l3.gen('ti', 'n|qTqT', '[vc=ps,tm=imf,sb=[+p1,+plr]]')
ንንቅጥቀጥ
>>> l3.gen('ti','gdf','[tm=j_i,+neg,sb=[+p2],ob=[+plr],vc=ps,as=rc]')
ኣይትጋደፉም
```

9. Limitations

a. General usability

In HornMorpho speed is sacrificed in favor of coverage. Because all possible analyses are returned, the search is non-deterministic; after an analysis has been found, the program backtracks and looks for another, until there are no more options. This can take considerable time. Generation is even slower, especially for relatively long verbs; this is due to the placement of feature structure constraints at the end of roots, a problem that Amtrup (2003) discusses.

A related problem that is less easy to fix concerns the program's inability to rank different analyses by their likelihood. Consider the Amharic word **ታላቅ**. The usual interpretation of this word is as an adjective (called "noun" in HornMorpho). But there is a much less likely possibility, that the word represents the third person singular feminine jussive form of the verb <*lqq*> in transitive voice and reciprocal aspect, an interpretation that might be translated 'let her make someone let go of something'. HornMorpho returns both analyses, with no indication of preference. The problem is worse with the Amharic noun guesser analyzer because of the different ways in which noun suffixes can be segmented when the stem is unknown. The guesser analyzer is called in the following Amharic example because **ዲፕሎማት** is not in the program's lexicon of noun stems.

```
(51) >>> l3.anal_word('am', 'ዲፕሎማቶቻችን')
     Word: ዲፕሎማቶቻችን
     ?POS: noun, stem: ዲፕሎማቶች
      possessor: 1, plur
     ?POS: noun, stem: ዲፕሎማት
      possessor: 1, plur
      grammar: plural
     ?POS: noun, stem: ዲፕሎማቶቻች
      grammar: accusative
     ?POS: noun, stem: ዲፕሎማታ
      possessor: 1, plur
      grammar: plural
```

The program would perhaps be usable to a wider audience if more of the interface, including the grammatical terminology, were in Amharic, Oromo, or Tigrinya rather than English. The author will need the help of native speakers to make this improvement in future versions of HornMorpho.

b. Morphological issues

A significant weakness of the Oromo and Tigrinya components of HornMorpho is that it only handles verbs. Tigrinya nouns are more complex than Amharic nouns because of their plural formation, and there is apparently no digitized dictionary that lists plural forms along with the corresponding singular forms. Oromo nouns will be added in the next version of the program.

For Amharic and Tigrinya common words are stored in a dictionary of forms that the program returns unanalyzed. In the current version, no part-of-speech or other information is provided for these words.

An important category of Amharic and Tigrinya words (in fact one of the important defining features of the Ethio-Semitic language family) that HornMorpho does not currently analyze are those that Baye (2000 E.C.) calls **የአደራረግ አዕማድ** in Amharic. These are derived from verbs and behave semantically like verbs (or adverbs) but are not conjugated themselves; instead they combine with the verbs **አለ** or **አደረገ** in Amharic, **በለ** or **ኣበለ** in Tigrinya. For example, **<u>ስብር</u> አለ**, **<u>እቱ</u> በለ**.

In general, HornMorpho handles Amharic nouns and adjectives less completely than verbs. Specifically, the program does not yet know about the following.

a. Noun plurals formed with the prefix *In_e*, for example, **እነከበደ**.

b. Adjective plurals formed through reduplication, for example, **ትንንሽ**.

c. Noun and adjective derivation from other other nouns or adjectives through the suffixes *-n_et*, *-m_a*, *-N_a,* for example, **ነጻነት**, **ውኃማ**, **ደንበኛ**.

d. Noun and adjective derivation from verbs beyond the four categories discussed above, for example, **ክብደት** (from <*kbd*>), **ሞጥሟጣ** (from <*mWTmWT*>), **ፍጹም** (from <*fS_m*>).

c. Lexical issues

The Amharic root and stem lexicons in the program were derived mainly from the Amharic-English dictionary of Amsalu (1979 E.C.), which is available under the Creative Commons Attribution-Noncommercial 3.0 United States License at <`http://nlp.amharic.org/resources/lexical/word-lists/dictionaries/amsalu-aklilu-amharic-english-dictionary/`>. The verb root lexicon contains 1,842 entries, certainly an incomplete list. The noun stem lexicon, containing 6,473 entries, is even more deficient; in particular it is missing all personal names and almost all place names.

The Oromo verb stem lexicon consists of 4,113 stems, extracted from the dictionaries of Gragg (1982) and Tamene (2000). Most of these stems are derived from simpler roots, and some are orthographic or phonological variants of one another. Clearly the list is far from complete.

The Tigrinya verb root lexicon in the program is limited to verb roots that could be extracted from Efrem's (2009) online Tigrinya-English dictionary (in the version as of February 2008) and a few others that have been added by hand; there are currently 598 entries. Thus the guesser analyzer must be relied on much of the time.

Intransitive and transitive verb roots are not distinguished in the lexicon, so the program analyzes a number of words that are not actually possible, such as Amharic ***ሞተኝ**, Oromo *du'ame*, and Tigrinya *ጥይቱኒ. There is also nothing that prevents Amharic or Tigrinya verbs in passive/reflexive voice from taking direct objects, so an impossible word like Amharic *ተመታኋት or Tigrinya *ተሃሪመያ is also analyzed as though it were grammatical. (Note that some formally passive Amharic and Tigrinya verbs *can* take object suffixes: ተሰማኝ, ተቀበልኩት; ተሰሚዑኒ, ተቐቢለዮ; etc.) Within the framework used to implement HORNMORPHO, it is straightforward to include constraints like these in the lexicon; the root is simply annotated with a feature structure constraint that limits the possible values for the `ob` feature. A goal in future work is to have this happen automatically in response to what possibilities occur in Amharic and Tigrinya corpora. Similarly it is possible to annotate roots for whether they can occur in particular derivational forms. The Oromo root *du'*- 'die', for example, should have the feature `[der=[-ps]]` to indicate that it cannot occur in the passive. Again, it should be possible to extract these features from corpora.

d. ORTHOGRAPHIC AND PHONOLOGICAL ISSUES

There is some variation in the spelling of Amharic words, and the program handles some of the options, for example, ሰጧችሁ ~ ሰጡአችሁ. Other variation is not covered. For example, only the first of these alternative spellings is analyzed by the program: ሰረቅሽው ~ ሰረቅሺው.

Amharic orthography includes two ways to represent the phoneme *s*, two ways to represent the phoneme *S*, three ways to represent the phoneme *h*, and two ways to represent vowels with no preceding consonant or a glottal stop. Although lexicographers have attempted to standardize the spellings of words with these sounds, and there is some agreement for common words, there is still considerable variation in the selections that writers make to spell these sounds. Ideally a morphological analyzer would handle all of this variation, but HORNMORPHO currently does not. Although it does include alternate spellings for a few roots and stems, for the most part, a word must conform to the single spelling the program is familiar with in order to be analyzed.

(52)
```
>>> l3.anal_word('am', 'ውሀ')
Word: ውሀ
POS: noun, stem: ውሀ
>>> l3.anal_word('am', 'ውኃ')
?Word: ውኃ
```

Within Tigrinya writing, there appear to be two different competing tendencies regarding the use of the characters that have the same pronunciation (note that in Tigrinya the አ and ዐ series are *not* pronounced the same). One is to attempt to standardize the spellings of words with these sounds based on the way they, or related words, were written in Ge'ez itself. Another is to forego the use of the less common characters altogether, that is, to avoid the ኀ, ሠ, and ፀ sets. HORNMORPHO follows this second trend and thus fails to handle any words containing characters in these sets.

(53)
```
>>> anal_word('ti', 'ተሰርሐ')
Word: ተሰርሐ
POS: verb, root: <srH>, citation: ተሰርሐ
 subject: 3, sing, masc
 grammar: perfective, passive
>>> anal_word('ti', 'ተሠርሐ')
?Word: ተሠርሐ
```

Future versions of the program will have an option allowing users to choose to handle characters in the ኀ, ሠ, and ፀ sets for Tigrinya.

Of course HORNMORPHO also fails to handle genuine misspellings, including some that are quite common, such as the substitution of ዉ for ው.

Within Oromo writing, there is a good deal of variation in the writing of short and long vowels and consonants. Some of this variation may be due to dialect differences, as suggested by Griefenow-Mewis (2001). Other examples are clearly misspellings. Except for a few cases where the variation is incorporated in the morphotactics (for example, the program accepts both *dubbatin* and *dubbatiin* for the negative imperative), HORNMORPHO currently has no way of handling these differences. A possible future application of the program is an Oromo spell-checker.

10. Changes

VERSION 2.0

Amharic

- Introduced the `phon_word` and `phon_file` functions for converting Ge'ez to phonetic representations that spell out gemination and sixth-order vowels.
- Fixed errors with the sequence *he* ኸ, as in መታኸው *met_ahew*.

Oromo

- Oromo verb analysis and generation added. Analysis includes the `segment` option, possible so far only with Oromo.

VERSION 1.1

Converted all modules to Python 3.1.

Amharic

- Added explicit role (`rl`) feature to raw analysis, for example,
- (41)
  ```
  >>> l3.anal_word('am', 'ቤቴን', raw=True)[0][1]['rl']
  [+acc, -gen, -p]
  ```
  This value of the `rl` feature indicates that the word has no prepositional prefix, is accusative, and is not genitive.

Tigrinya

- Added alternate form with *-te-* for iterative transitive imperfective and jussive: የተፈላልጥ *yetefelalIT* (alongside the more common የፈላልጥ *yefelalIT*).
- Fixed several errors with roots ending in *y*.
- Eliminated the possibility of CaC_C roots.

VERSION 1.01

Amharic

- Fixed some errors in object pronoun suffixes:
  - when there is no subject suffix, *I* as well as *e* is permitted before -*wo*(*t*).
  - -*ew* (rather than -*w*) follows 1st person plural subject suffix -*n*.
- Fixed an error that prevented a sequence of two labialized consonants, which can occur when the second consonant in a CCC root is labialized: **ተኰሷል** *tekW_IsWal,* **ያጨናጕሏቸዋል** *yaC_enag-W_IlWac_ewal.*

Tigrinya

- Fixed errors that prevented the following from being analyzed or generated:
  - iterative forms of CCC verbs with *w* or *y* in the second position: **ተቐዋወመ** *teQewaweme*, **ሽያየጠ** *xeyayeTe.*
  - transitive forms of CCC verbs with a laryngeal consonant in the second position: **ኣስዓመ** *'as`ame*.
  - imperfective forms of CCCC verbs with labialized consonant in first position: **ይኹስኵስ** *yIK-WIskWIs.*
  - relativized forms of imperfective verbs with second person or first person plural subjects before geminated stem-initial consonants: **ዝንዛረብ** *zIn_Iz_areb*, **እትዛነዩ** *'It_Iz_aneyu*.
- Changed the entries for the irregular verbs with roots *whb* and *tHz*, preventing the incorrect forms ***ወሃበ** *wehabe* and ***ተሓዘ** *teHaze*.
- Fixed the entries for the irregular verb with root *bhl* so that its transitive forms are generated and analyzed: **ኣበለ** *'ab_ele*, **የብል**, *yeb_Il*, etc.

References

ባየ ይማም። 2000 ዓ.ም.። የአማርኛ ሰዋስው፣ የተሻሻለ ሁለተኛ እትም። አዲስ አበባ፣ እሌኒ ማተሚያ ቤት።

አምሳሉ አክሊሉ። 1979 **ዓ.ም.። አማርኛ - እንግሊዝኛ መዝገበ ቃላት** Amharic-English Dictionary. **አዲስ አበባ፣ ኩራዝ አሳታሚ ድርጅት።**

Abarraa Nafaa (Ed.) (2003). *Caasluga Afaan Oromoo*. Finfinnee: Komishinii ‘Aadaafi Turizimii Oromiyaa.

Amanuel Sahle (1998). **ሰዋስው ትግርኛ ብሰፊሑ**. *A comprehensive Tigrinya grammar.* Lawrenceville, New Jersey: Red Sea Press.

Amtrup, J. (2003). Morphology in machine translation systems: efficient integration of finite state transducers and feature structure descriptions. *Machine Translation*, *18*, 213-235.

Beesley, K.R., and Karttunen, L. (2003). *Finite state morphology*. Stanford, California: CSLI Publications.

Bender, M.L. and Hailu Fulass. (1978). *Amharic verb morphology*. East Lansing, Michigan: African Studies Center, Michigan State University.

Carpenter, B. (1992). *The logic of typed feature structures.* Cambridge: Cambridge University Press.

Cohen-Sygal, Y., and Wintner, S. (2006). Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, *32*, 49-82.

Efrem Zacarias (2009). Memhir.org Dictionaries (English-Tigrinya, Hebrew-Tigrinya dictionaries). <http://www.memhr.org/dic/>

Gasser, M. (2006). *How language works*. <http://www.indiana.edu/~hlw/>.

Gasser, M. (2009). Semitic morphological analysis and generation using finite state transducers with feature structures. *Conference of the European Chapter of the Association for Computational Linguistics*, *12*.

Gragg, G. (1982). *Oromo dictionary*. East Lansing, Michigan, USA: Michigan State University Press.

Griefenow-Mewis, C. (2001). *A grammatical sketch of written Oromo*. Köln: Rüdiger Köppe Verlag.

Griefenow-Mewis, C. and Tamene Bitima (1994). *Lehrbuch des Oromo*. Köln: Rüdiger Köppe Verlag.

Johnson, C.D. (1972). *Formal aspects of phonological description.* The Hague: Mouton.

Kaplan, R.M., and Kay, M. (1994). Regular models of phonological rule systems. *Computational Linguistics*, *20*, 331-378.

Karttunen, L., Kaplan, R.M., and Zaenen, A. (1992). Two-level morphology with composition. *Proceedings of the International Conference on Computational Linguistics*, *14*, 141-148.

Koskenniemi, K. (1983). *Two-level morphology: A general computational model for word-form recognition and production*. Publication 11, University of Helsinki, Department of General Linguistics, Helsinki.

Leslau, W. (1941). *Documents tigrigna: grammaire et textes*. Paris: Libraire C. Klincksieck.

Leslau, W. (1995). *Reference grammar of Amharic*. Wiesbaden: Harrassowitz.

Saba Amsalu and Girma A. Demeke. (2006). Non-concatenative finite-state morphotactics of Amharic simple verbs. *ELRC Working Papers*.

Tamene Bitima. (2000). *A dictionary of Oromo technical terms*. Köln: Rüdiger Köppe Verlag.

Yitna Firdyiwek and Yaqob, D. (1997). The System for Ethiopic Representation in ASCII. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.3191>.