2017

GENERATING AMHARIC PRESENT TENSE VERBS: A NETWORK MORPHOLOGY & DATR ACCOUNT

T. Michael W. Halcomb
University of Kentucky, michael.halcomb@uky.edu
Digital Object Identifier: https://doi.org/10.13023/ETD.2017.222

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

T. Michael W. Halcomb, Student

Dr. Andrew Hippisley, Major Professor

Dr. Rusty Barrett, Director of Graduate Studies

GENERATING AMHARIC PRESENT TENSE VERBS: A NETWORK MORPHOLOGY & DATR ACCOUNT

THESIS

A thesis submitted in partial fulfillment of the requirements for the degree of
Master of Arts in Linguistic Theory & Typology
in the College of Arts and Sciences
at the University of Kentucky

By

T. Michael W. Halcomb

Co-Directors: Dr. Gregory Stump, Professor of Linguistics and Dr. Andrew Hippisley, Professor of Linguistics

Lexington, Ky

2017

ABSTRACT OF THESIS

GENERATING AMHARIC PRESENT TENSE VERBS: A NETWORK MORPHOLOGY & DATR ACCOUNT

In this thesis I attempt to model, that is, computationally reproduce, the natural transmission (i.e. inflectional regularities) of twenty present tense Amharic verbs (i.e. triradicals beginning with consonants) as used by the language's speakers. I root my approach in the linguistic theory of network morphology (NM) and model it using the DATR evaluator. In Chapter 1, I provide an overview of Amharic and discuss the *fidel* as an abugida, the verb system's root-and-pattern morphology, and how radicals of each lexeme interacts with prefixes and suffixes. I offer an overview of NM in Chapter 2 and DATR in Chapter 3. In both chapters I draw attention to and help interpret key terms used among scholars doing work in both fields. In Chapter 4 I set forth my full theory, along with notation, for generating the paradigms of twenty present tense Amharic verbs that follow four different patterns. Chapter 5, the final chapter, contains a summary and offers several conclusions. I provide the DATR output in the Appendix. In writing, my main hope is that this project will make a contribution, however minimal or sizeable, that might advance the field of Amharic studies in particular and (computational) linguistics in general.

KEYWORDS: Amharic, Ethiopic, Network Morphology, DATR

<u>T. Michael W. Halcomb</u>

<u>March 20th, 2017</u>

GENERATING AMHARIC PRESENT TENSE VERBS:
A NETWORK MORPHOLOGY & DATR ACCOUNT

By

T. Michael W. Halcomb

Dr. Andrew Hippisley
Co-Director of Thesis

Dr. Gregory Stump
Co-Director of Thesis

Dr. Rusty Barrett
Director of Graduate Studies

5/23/17
Date

Dedicated to Emush and Aschalew,
whom I have continually thought about and prayed for throughout this process.

ACKNOWLEDGEMENTS

Although this thesis bears my name as the author, the end-result is only possible because of the efforts of numerous other individuals who have helped along the way. First and foremost, I must give thanks to God, who has sustained me through the duration of my coursework, research, and writing. Thanks are also due to my wife, Kristi, who has been a constant source of support and encouragement. I hope that my children, although they only seem to have a partial idea of what it is I have been doing, have witnessed my love for language as well as what it means to work hard. I am also grateful to my friend, a native Amharic speaker who offered some feedback for this project, Fasil Woldemariam, whom I hope to work with more in the future.

I wish, too, to thank the Department of Linguistics at the University of Kentucky. My time studying here has opened up a new world of interests and possibilities for me. Thanks to both Dr. Andrew Byrd and Dr. Jennifer Cramer, who early on, not only allowed me to bring my linguistics to my study of the Bible but encouraged it. I am particularly grateful to Dr. Mark Lauersdorf, who demonstrated time and again what it looks like to make an investment in students. Finally, I would like to thank both Dr. Andrew Hippisley, the one who introduced me to Network Morphology and DATR, and Dr. Greg Stump, who embodies the notion of a “gentleman and scholar,” for co-chairing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

Today's world is, in large part, computer-driven. Government offices and officials, businesses and business owners, educators and students, and many others rely on technology. This, too, is true of linguists. Indeed, many branches of the government, business sector, and academy have come to the realization that computational linguists can be incredibly valuable assets. As Clark et al. (2013: 1) note, "The field of computational linguistics (CL), together with its engineering domain of natural language processing (NLP), has exploded in recent years."

This is the case because many (although, certainly not all!) computational linguists often work for companies driven by interests in mining "big data." Increasingly, for example, computational linguists are gaining expertise in the fields of cognitive psychology, artificial intelligence, mathematics, formal logic, speech processing, and more. The ability to leverage inter- and / or cross-disciplinary skills and insights has taken on great significance. While there is more cross-fertilization today, this interdisciplinary mindset has been present since the 1950s, the early days of CL's predecessor—Machine / Mechanical Translation (MT).[1]

It is interesting, however, to juxtapose this with the comments of Nick Cercone: "The narrow approaches to machine translation of the early 1960s pale when compared to the considerable assortment of methodologies available to the modern computational linguist" (1983: v). Given the advances since then, a computational linguist in 2017 could likely make similar judgments of the state of the field in 1983; the same will probably be true thirty years from now. Nevertheless, just three years after Cercone's remarks, Ralph Grishman noted in 1986 that, "It [computational linguistics] has the potential for

[1] Also: Machine/Mechanical Learning (ML). For more on the history of MT (Melby, 1995: 13-42).

expressing an enormous range of ideas, and for conveying complex thoughts succinctly. Because it is so integral to our lives, however, we usually take its powers and influence for granted. The aim of computational linguistics is, in a sense, to capture this power" (1).

To cite Grishman again, "By understanding language processes in procedural terms, we can give computer systems the ability to generate and interpret natural language. This would make it possible for computers to perform linguistic tasks…and make it much easier for people to access computer-stored data" (1). If one fast-forwards a bit closer to the present, they will find this perspective still deeply embedded in much of the literature. One example is displayed in the 2104 work of Roland Hausser who notes that, "The goal of computational linguistics is to reproduce the natural transmission of information by modeling the speaker's production and hearer's interpretation on a suitable type of computer" (xix).

In this thesis, I essentially proceed with Hausser's definition in mind. To be more precise: the goal of this thesis is to computationally reproduce the natural transmission (i.e. inflectional regularities) of present tense Amharic verbs as used by the language's speakers. Framed by the linguistic theory of network morphology (NM) and expressed in the DATR representation language, the aim is to develop a minimally redundant description of the paradigms for twenty present tense verbs. This, in turn, might assist interpreters in their efforts to more efficiently and effectively engage, understand, and utilize the language. Thus, I believe this work has the potential to fit well within the realm of computer assisted language learning (CALL) by being of pedagogical use to teachers and of research use to learners. It might also provide a means of spell- or form-checking verbs among readers, writers, and translators. In this work, however, I do not address natural language processing (NLP) or MT applications.

I, of course, am not the first to bring Amharic into conversation with CL. Others, perhaps most notably, Michael Gasser (2010; 2011; 2012), have already undertaken an immense amount of work on this matter. Moreover, at Addis Ababa University, in Ethiopia, many students continue to produce quite a bit of CL research on Amharic (Bayou 2000; Bayu 2002; Gebreegziabher 2011; Alemuu 2013; Demelash 2013; and Alemu 2013). Yet, to my knowledge, work on the relationship between NM and Amharic remains to be undertaken. My hope is that this brief study will fill that gap just a bit and, if possible, make some sort of lasting contribution to the fields of CL and Amharic studies

Chapter 1: A Brief Overview of Amharic

Amharic, a sister language of Tigrinya, is the national language of Ethiopia (Tadross and Teklu 2015: 9). It descended from Ge'ez, which is now a strictly liturgical variety. Amharic belongs in the Afro-Asiatic language family and is characterized by most as a Semitic language. Through language contact, however, it has also acquired a number of Cushitic features (Leslau 1945: 59-82; and Little 1974: 267-73).[2] Impressively, Amharic boasts nearly 26 million global speakers today and, over the last several decades, has received quite a bit of interest from linguists.

In addition to Amharic's fascinating script, the alphabet—or *fidel*—is what linguists often refer to as an abugida. This stands in contrast to Février's (1995: 330) earlier label of "neosyllabary" as well as Householder's (1959: 379-83) notion of "pseudo-alphabet." In reaching an understanding of what an abugida is, a helpful place to begin is with Lyovin's et al. (2017: 43) note that, "In perhaps all syllabically organized phonemographies, consonants are treated as more basic entities than vowels are." In other scripts, however, "vowels are represented, but are graphically subordinated to any preceding consonant" (43). Each letter (or orthographic representation), then, typically consists of a consonant plus a specific vowel. Whereas the consonant always retains the same sound (but may morph or modify orthographically), the vowel sound changes (cf. Halcomb 2015). This type of writing system is what Daniels (1990: 731) refers to as an abugida.

Unlike English, for instance, where each individual letter stands on its own regardless of whether it is a consonant or vowel, in Amharic each consonant self-contains

[2] It is worth mentioning that, while Amharic is not considered on its own in Zaborski (1975: 1-183), it is used comparatively on numerous occasions and, as such, the work may prove beneficial for some.

the vowel (with the exception of the two vowels, አ (Alf) and ዐ (‘Ayn), that essentially function as consonantal placeholders). Thus, only a character is written when representing a consonant-vowel pair. For example, in English one would need two characters to form the word ‘he,’ namely the consonant h and the vowel e. In Amharic, if one wanted to write the orthographic equivalent of ‘he,’ they would simply write ሀ. Here, one character does the job whereas English would require two. Since there are seven vowels in the *fidel*, each representing its own “order,” the shape of the character essentially remains the same but takes on a minor change depending on which of the seven orders (or vowels) it is working in tandem with. The seven orders, according to the IPA, are represented by a or ɛ, u, i, ɐ, e, ɪ, o.[3] Thus, the letter representing h is going to slightly change according to each “order” (listed here in sequence) as follows: ሀ | ha, ሁ | hu, ሂ | hi, ሃ | he, ሄ | hɐ, ህ | hɪ, ሆ | ho.

It should be noted here that the “sixth order” forms (e.g. ህ, ል, ሕ) are able to, depending on their position in the word, either keep or lose the vowel both phonetically and orthographically. A good rule of thumb is that “sixth order” forms defining a syllable or word boundary drop the vowel. This, however, does not always happen and, so, one must do due diligence to discern whether or not this is occurring with individual words on a case-by-case basis.

Along with these orthographical principles of Amharic, another oft-discussed feature of this Semitic-based language, especially with regard to the verb system, is its root-and-pattern system of morphology (RPM) (Schluter 2008: 287-301). Amberber (2008: 83) describes RPM as being “characterised by a root that consists of consonantal radicals and a pattern that comprises consonantal positions and vowels. In general, the

[3] For either a broad or narrow (i.e. non-IPA) English transliteration see Halcomb (2015).

roots encode lexical meaning, whereas the patterns encode grammatical meaning.” I offer here an example of the root for the word ‘begin,’ whose Amharic radicals are ጀመረ | d͡ʒ-m-r-.[4] It is important to note that, in the immediately preceding parentheses, the - represents a missing vowel which, in this case, is simply an ɛ. Thus, ጀመረ results in the transliteration d͡ʒɛmɛrɛ. On the one hand, the consonants, representing the root (or lexeme), encode the lexical meaning ‘begin.’ On the other hand, the vowels, representing the pattern, encode grammatical meaning, that is, they convey things like tense, aspect, mood, and person (the -ɛ-ɛ-ɛ or -1-1-1 or –v-v-v pattern here represents a PRF IND 3MS form resulting in the specific meaning ‘he began’).

I should point out here that, in Amharic, gemination is a topic that has received much attention. It is not within the scope of this project to address it in great length, but it is worthy of a brief comment. As Fabri et al. (2014: 6) note, “most words contain at least one geminated consonant, and spoken Amharic lacking gemination sounds quite unnatural.” They continue, “there are relatively few minimal pairs because of redundancy” and syntax “must be relied on to disambiguate these words” (6-7). In his *Reference Grammar of Amharic* (1995: 12-13), Leslau gives fifteen examples (e.g. ገና | gana ‘still’ - ገና | ganna ‘Christmas’ and ዋና | wana ‘swimming’ - ዋና | wanna ‘chief’). In his *Amharic Textbook* (1968: 5), Leslau also recycles a few of those examples and offers a handful of additional ones. As the work of Anberbir and Takara (2009: 47) demonstrates, when it comes to a computational approach of Amharic, “The lack of orthography of Amharic to show geminates is the main problem.” Indeed, they developed their own gemination mark (‘) to attempt to account for this. Rather than insert foreign

4 Since each - represents an e here, that is, the vowel of the “first order,” one could replace the - with a 1. Such a practice is not uncommon in scholarly Amharic literature. Thus, instead of d͡ʒ-mm-r-, one could write d͡ʒ1mm1r1. Or, one could simply remove the - or 1 and write d͡ʒmmr, which is also common.

marks like this one into the orthography, which might be confusing to readers since it has not received widespread acceptance, I have chosen to leave the Amharic as it stands. Even so, I have also opted to include gemination in the transliterations. In Chapter 5, I have included a brief discussion of gemination within my theory as it pertains to the verb patterns considered in this project.

The final item to consider in this section is the notion of affixes, specifically prefix-suffix pairs. Because I am focusing on present tense verbs in Amharic, both prefixes and suffixes require attention. Specifically, in the simple present, Amharic prefixes pair with suffixes denote to grammatical gender and number. With regard to gender, in Amharic there is no "neuter" grammatical gender and masculine is the default. Moreover and interestingly, in the first person singular there is no gender distinction (i.e. grammatical gender is "common," which may have to do with indexicality (Yasatuda 2010) or indicate the decrease in importance of grammatical gender in Amharic (Kramer 2014). I should note, too, that in formal descriptions of Amharic, as with other Semitic languages such as Hebrew, it is standard to treat the PRF 3MS as the lexical form. Because my interest is focused more on present tense verbs, I have chosen not to use that as my own starting point.

Continuing the line of thought just above, the relevant affixed affixes with their particular grammatical encodings (along with person) are: እ...አለሁ | ɪ...alɛhu (1CS);[5] ት...ለህ | tɪ...alɛ (2MS); ት...ሻል | tɪ...alɛʃ (2FS); ይ...አል | jɪ...al (3MS); ት...ለች | tɪ...lɛt͡ʃ (3FS); እን...ለን | ɪnnɪ...lɛn (1CP); ት...ላችሁ | tɪ...lat͡ʃhu (2CP); and ይ...ሉ | jɪ...lu (3CP). Essentially, one attaches these various suffixes to the end of the lexeme to denote the grammatical

[5] It is important to note that the አ here, which is a consonantal placeholder, is often assimilated into the preceding consonant-vowel character (via sandhi), thereby forming a "fourth order" form. This, in fact, happens repeatedly throughout the paradigm.

meaning they want to encode. Thus, if one takes the PRF 3MS form ጀመረ | d͡ʒɛmmɛrɛ ('he began') and wishes to say instead 'I begin,' they do so by changing the pattern and adding the appropriate suffix, namely, አለሁ | alɛhu. The resultant form is እጀምራለሁ | ɪd͡ʒɛmmɪralɛhu.[6] I have included the above data, along with pertinent additional information, in a table below for ease of viewing.

Paradigm of መጀመር | mɛd͡ʒɛmɛr (inf.) 'to begin' (Table 1)					
Person, Gender, Number	**Prefix**	**Root & Pattern** 1-66-4	**Suffix**		**Final Form**
1, Comm, Sg 'I begin'	እ ɪ	ጀምሯ + d͡ʒɛmmɪra +	ለሁ lɛhu =		እጀምሯለሁ ɪd͡ʒɛmmɪralɛhu
2, Masc, Sg 'You begin'	ት tɪ		ለህ lɛh =		ትጀምሯለህ tɪd͡ʒɛmmɪralɛ
2, Fem, Sg 'You begin'	ት tɪ		ለሽ lɛʃ =		ትጀምሯለሽ tɪd͡ʒɛmmɪralɛʃ
3, Masc, Sg 'He/it begins'	ይ jɪ		አል al =		ይጀምሯአል jɪd͡ʒɛmmɪral
3, Fem, Sg 'She begins'[7]	ት tɪ		ለች lɛt͡ʃ =		ትጀምሯለች tɪd͡ʒɛmmɪralɛt͡ʃ
1, Comm, Pl 'We begin'	እን ɪnnɪ	ጀምሯ + d͡ʒɛmmɪra +	ለን lɛn =		እንጀምሯለን ɪnnɪd͡ʒɛmmɪralɛn
2, Comm, Pl 'You begin'	ት tɪ		ችሁ t͡ʃhu =		ትጀምሯላችሁ tɪd͡ʒɛmmɪralat͡ʃhu
3, Comm, Pl 'They begin'	ይ jɪ		ሉ lu =		ይጀምሯሉ jɪd͡ʒɛmmɪralu

The above overview, although succinct, should contain enough information in order to move forward with an NM analysis of Amharic present tense verbs. Before doing that, however, there is one last detail that I should mention. In Amharic verbs can, for all intents and purposes, be broadly grouped according to the number of their lexical radicals. The norm is to consider five categories: uniradicals, biradicals, triradicals, and quadriradicals, along with multi-radicals (any lexeme consisting of five or more radicals).

6 This pattern for this is -d͡ʒ-mm-r-l-h, that is, 6**d͡ʒ**1**mm**6**r**4**l**1**h**2, where the numbers represent the "order" of the vowel. This could be represented in general by simply replacing the numbers with "v" for vowel and "C" for consonant (e.g. vCvCCvCvCvC). I use this general representation later in this work.

7 In Amharic there is no "neuter" grammatical gender and masculine is the default.

For present expository purposes, I have chosen to limit my analysis to triradicals beginning with consonants only. Now that I have presented these details of the Amharic present tense verb system, I turn my attention to NM.

Chapter 2: A Brief Overview of Network Morphology

According to Brown and Hippisley (2012: 6), "Network Morphology is a paradigm-based framework: morphological generalizations are gathered at the level of the paradigm." They note that the "fundamental object of enquiry in morphology" in a paradigm-based approach "is the lexeme rather than the morpheme" (6). Thus, in NM, the notion of the paradigm is central. In addition, and as the name implies, Hippisley asserts that in NM, "Morphological knowledge is represented as a network, and this allows for an elegant account of inflectional classes and various other dissociations between syntax and morphology, such as syncretism and deponency" (2016: 482). As Corbett and Fraser (1993: 116) note, NM rests on the assumption that "Lexical information is organized as a network whose basic elements are nodes and facts, and whose structure consists of relationships between basic elements."

This coincides with Stump's comment that, in NM, a "network of nodes can be represented as a hierarchy in which dominated nodes inherit from dominating nodes" (2001: 261). That is, a node can inherit facts from another node and, in doing so inherit specific features that result in generalizations within the paradigm (261). Thus, as Parker Brody has aptly stated, "the paradigm of an inflectional system is generated by associating the cells of the paradigm with the morphosyntactic properties they encode. As each word passes through the model, it draws on the assumptions of the nodes above it, as well as overrides that stipulate irregularities in the system" (2014: 8). This is what produces the "elegant account" of inflectional classes that Hippisley refers to and corresponds with Stewart's assertion that, in NM, because lexical classes and subclasses are defined in this way, “this allows generalizations to refer to individual nodes or hierarchically related nodes” (2008: 178).

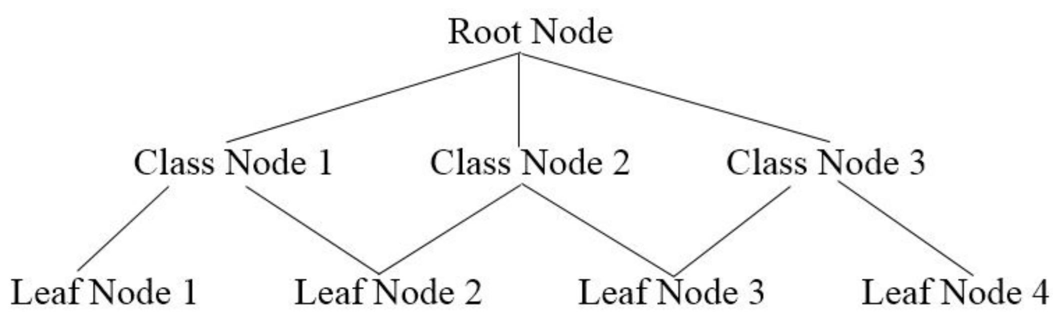
NM essentially employs the language of object-oriented programming to lay bare shared morphological features and make connections between shared lexemes and affixes. As such, NM employs a basic inheritance hierarchy of nodes. In NM, the top-most nodes are dominant. Moreover, there is a principle of “the longest path wins” (Hippisley 2010: 36). Stump (2011: 10) points out that this is essentially Panini’s Principle (or: the Elsewhere Condition), that is, the idea that each cell or block in a paradigm has rules that become ranked. Hippisley (2016: 489-90) echoes this saying, “This is the elsewhere statement in lexical phonology, or Paninian default inference, and is used to resolve competition among rules. In other words, Network Morphology subscribes to the Panini Determinism Hypothesis” (cf. Brown and Hippisley 2012: 22). Thus, in an environment where multiple rules could apply, whichever rule has the greatest degree of specificity wins, thereby preventing the others from being applied.

The hierarchy’s top is the “root node” and at its bottom sit the “leaf nodes.” In NM the “class nodes” inherit properties from the root node (i.e. the syntactical node).[8] A node inherits properties from a node that dominates it and the inherited properties are said to be defaults. These defaults, however, are subject to overrides—contrasting properties specified in a class node. In addition, if there are properties not present in the root node, due to variation, for example, a class can have its own properties. In NM, the leaf nodes inherit from the class nodes. The leaf node contains entries that include lexical, semantic,

[8] Because different forms in a paradigm convey different meanings or functions—what proponents of NM often refer to as “features”—they are relevant to syntax. For more on syntax and NM see the Surrey Morphology Group’s website, particularly the page on morphosyntactic features: http://www.surrey.ac.uk/LIS/SMG/morphosyntacticfeatures.html#morphosyntacticfeatures. Last accessed 4/6/17.

phonological, and morphological information. But for those nodes to manifest, they must draw features from one or more class nodes. This "drawing" effect is known as inheritance—inheriting properties (or: facts) from higher nodes. The properties inherited via paths are represented in Figure 1 below by lines.

Figure 1, Nodes in Network Morphology



The above diagram puts on display the hierarchical and network-based structure of NM. I will discuss these matters in relation to DATR in the next section, but it will prove beneficial at this point to clarify a bit of relevant terminology. To do this, I will draw on the work of Corbett and Fraser (1993: 116-17), as well as Hippisley (2016: 482-83).

In NM a node is "a named location at which one or more facts may be stored" (Corbett and Fraser 1993: 117). More precisely, these are "*inheritable* facts" (Hippisley 2016: 483). Facts themselves consist of attribute:value pairs. It is worth noting, however, that the literature on NM often uses the language of path:value pairs, too, to mean the same thing. Moreover, at least in terms of coding NM, angle brackets <> represent paths (i.e. the means by which an attribute is expressed) and, specifically, path delimiters. This path's value may be atomic, another path, or even a mixture of the two. Here I will simply use the language of attribute:value.

According to Corbett and Fraser, “A value may be stated directly or referenced indirectly by means of another attribute having that value” (Corbett and Fraser 1993: 118). Attributes (or: paths) “may be atomic” or “consist of a list of atoms” and these “increase in specificity from left to right” (118). Similarly, “Values may be atomic or list-structured, where a list consists of a sequence of atoms” (118). I consider the “atom” to be a single or individual property of an attribute or value. While an atom can appear in list (or: sequence) form, each atom should be taken on its own merit (e.g. see below: love ing where both ‘love’ and ‘ing’ are atoms) (Evans and Gazdar 1996: 169). In order to help visualize these rather abstract concepts, below I have provided an example from Evans and Gazdar based on the lexeme ‘love,’ in the form of a table (169). Note that, in this table, syn represents “syntactic,” “cat” represents “category,” “mor” represents “morphological,” and the double equal sign == directs the values assigned to the attribute.

Path/Value Pairs for **Love** (Table 2)		
attribute	**path**	**value**
<syn cat>	==	verb
<syn type>	==	main
<syn form>	==	present participle
<mor form>	==	love ing

The expected output here would, of course, be ‘loving’ (not loveing). Nevertheless, the point of the table is to simply give a more concrete image of what NM starts to look like when employed. I will have occasion below to demonstrate how the attribute-path-value strings work and factor into the overall NM framework. For now, however, I shall move on to a discussion of DATR—NM’s formalism (i.e. formally explicit language that is computationally interpretable).

Chapter 3: A Brief Overview of DATR

As already noted, DATR is essentially a lexical representation language that can express default inheritance. Thus far, I have not been able to pinpoint any literature in which the letters in DATR are discussed as an acronym. It seems to be the case however, that DATR is based on PATR or perhaps, its descendant, PATR-ii. The former, developed in the mid-1980s, was an acronym for **PA**rsing and **TR**anslation (Bussmann, 2006, 870). According to Sikkel (2012: 168), PATR has since "fallen into oblivion" and for that reason the letters in its descendant, PATR-ii, "no longer form an acronym." For this reason, DATR is likely not a descendant acronym but merely a name. On the interfacing of DATR with PATR, see Kilbury (1991: 137-42).

DATR shares many characteristics with Object Oriented Programming (OOP). As Seidl et al. note, "object orientation" models were introduced in the 1960s using the SIMULA programming language. This language was built "on a paradigm that was as natural to humans as possible to describe the world" (2014: 6). As such, the "object-oriented approach corresponds to the way we look at the real world" taking seriously the fact that a) "objects are elements in a system whose data and operations are described," and b) objects "interact and communicate with one another," thus playing a key role in object-oriented approaches (6). It is no coincidence, then, that some of the terminology is adopted and used by advocates of NM and DATR. Several terms of significance, including a few noted already and a few not yet noted, are worthy of attention at this point. Building on the work of Seidl, with specific regard to terminology and concepts, I provide these terms and their corresponding definitions in list form below. In addition, I offer both a running example of code and its output.

- **Class**: A class defines an attribute or set of attributes as well as a value or set of values for a set of objects. To draw on Seidl's analogy, for instance, "people have a name, an address, and a social security number." As such, the atoms (or: instances) of these objects create a group or class (6). Unlike OOP, however, in DATR there are no methods (i.e. actions upon an object within a class).

OPP Example 1: Atoms, Attributes, Classes, and Objects (Table 3)	
Person: <> == yes <has name> == yes <has address> == yes <has social> == yes <has wings> == no.	% Here "Person" is an object while name, address, % social, and wings are atoms of that object which, % collectively, denote a class. Each atom has an % attribute of yes or no but there is also an affirm-; % ation of all unspecified properties.

- **Object**: The end-result of compiling a class's atoms (or: instances) is an object. For example, a name, address, and a social security number are atoms that, collectively, denote the object "person" (see above).
- **Encapsulation**: This is the act of protecting the internal state of an object against unauthorized access or grouping (7). Stated differently, it is like putting an object (and hence its atoms) inside a capsule. Importantly, only members of the same class or subclass have authorized access to that object. Thus, encapsulation prevents objects of different classes from being grouped together. Thus, if a class "Car" were to exist, the class "Person" and its atoms could be prevented from gaining access to the class it (see below). Likewise, "Car" and its atoms could be prevented from gaining access to the class "Person."
- **Path**: Also known as a "Message," the path allows and is the means by which objects communicate with one another. Borrowing from Seidl et al., a path "to an object represents a request to execute an operation. The object itself determines

whether and how to execute this operation. The operation is only executed if the sender is authorized to call the operation" (7).

- **Inheritance**: This is "a mechanism for deriving new classes from existing classes" (7-8). For instance, a subclass might derive from an existing (super) class and, as such, inherit all of its attributes or it may "define new attributes and/or operations, overwrite the implementation of inherited operations, [or] add its own code to inherited operations" (8). This allows the "reuse of program or model parts (thus avoiding redundancy and errors)…use as a modeling aid through a natural categorization of occurring elements, and support for incremental development" (8). Important, too, for DATR, are the concepts of direct and indirect inheritance. The former, per Keller, simply refers to a value specification expressed directly (i.e. it does not draw/inherit from elsewhere) and the latter denotes an occasion where "the value is obtained by local inheritance" (1996: 646). (Note: The % symbol functions to section off comments from code.)

OPP Example 2: Encapsulation, Inheritance, and Paths (Table 4)	
Person: <> == yes <has name> == yes <has address> == yes <has social> == yes <has wings> == no.	% Here "Person" is an object while name, address, % social, and wings are atoms of that object which, % collectively, denote a class. Each atom has an % attribute of yes or no but there is also an affirm- % ation of all unspecified properties.
Female: <> == Person.	% Here the class "Female" has an empty attribute % the value is set to "Person" and, thus, the path % leads it to inherit the defaults from the class % "Person."
Car: <has brakes> == yes <has windows> == yes.	% Here the class "Car" has two attributes with set % affirmative values. It does not inherit from Person % because this theory doesn't model a connection % between a car's brakes or windows and attributes % a person may have. Encapsulation is present.

- **Override**: Also known as “Overload,” in OOP this “enables an operation to be defined differently for different types of parameters” (Seidl, 2014: 7). This is a significant aspect of DATR. Indeed, when Evans and Gazdar (1996: 167) describe DATR as “a rather spartan nonmonotonic language for defining inheritance networks with path/value equations,” this seems to be part of what they’re referring to. The notion of “nonmonotonic” here appears to be borrowed from the field of logic and, more specifically, nonmonotonic reasoning (NR). According to Antoniou and Williams (1997: 5), NR “provides formal methods that enable an intelligent system to operate adequately when faced with incomplete and changing information.” Because NM and DATR are concerned with matters such as regularity and semi-regularity as well as lexical-paradigmatic predictability, and given that languages are living entities that change, a nonmonotonic approach is necessary.

OPP Example 3: Override (Table 5)	
Person: <> == yes <has name> == yes <has address> == yes <has social> == yes <has wings> == no.	% Here “Person” is an object while name, address, % social, and wings are atoms of that object which, % collectively, denote a class. Each atom has an % attribute of yes or no but there is also an affirm- % ation of all unspecified properties.
Female: <> == Person.	% Here the class “Female” has an empty attribute % the value is set to “Person” and, thus, the path % leads it to inherit the defaults from the class % “Person.”
Car: <has brakes> == yes <has windows> == yes.	% Here the class “Car” has two attributes with set % affirmative values. It does not inherit from Person % because this theory doesn’t model a connection % between a car’s brakes or windows and attributes % a person may have. Encapsulation is present.
Jane:	% Here the class “Jane” inherits from the class

<> == Female <has social> == no <is mean> == no.	% “Female,” which inherits from the class “Person” % but also has an override where <has social> % is not the default “yes” but, rather, overrides it % and becomes a “no.”
# hide Person Female.	% This just hides what does not need to be seen.
# show <has name> <has address> <has social> <has wings> <is mean> <has brakes> <has windows>.	% This shows what is necessary.

- **Hierarchy**: A hierarchy, particularly with regard to classes (i.e. class hierarchy) “consists of classes with similar properties” and, as such, generates an inheritance tree. The hierarchy of classes are built upon and situated within nodes, with the root node being the top-most and default node. The example code provided here (see above or below) is structured hierarchically. As Keller (1996: 647) asserts, “A DATR hierarchy is defined by means of path-value specifications. Inheritance of values permits appropriate generalizations to be captured and redundancy in the description of data to be avoided.”
- **Multiple Inheritance**: Similar to what is described in OOP as “polymorphism,” this is essentially “the ability to adopt different forms” (Seidl, 2014: 8) with regard to referencing objects from different classes. I have not included an example within a table, but it is easy enough to understand how Jane could inherit properties from multiple classes, namely, “Person” and “Car.”
- **Redundancy**: In particular, this refers to what OOP programmers refer to as “spatial redundancy.” A portion of code is said to exhibit (spatial) redundancy if it “frequently makes the same decision because it is reached by the same code path”

(Odersky, 2004: 188). In short: redundancy is the unnecessary repetition of code. In an effort to keep my code clean, I have not included an example of redundancy here.

Each of these principles is important for understanding, navigating, and working within the DATR environment. That, then, brings me to a brief discussion about the platform from which I choose to run DATR. To be sure, DATR is compatible with and often used in computing environments like Prolog and Python. I, however, use Raphael Finkel's online evaluator.[9] Here, one simply pastes in their theory (i.e. code) and presses the submit button. In Finkel's environment one can paste in multiple portions of code or a single theory. Likewise, researchers have the option of telling the evaluator to output the results in either list format or paradigm format. In the context immediately below, I have provided examples of both. In the following chapter, however, I will use the latter. Bearing these items in mind, I now turn to the next chapter, which provides my full theory along with notation.

<table>
<tr><th colspan="2">Output of Theory in DATR (Table 6)</th></tr>
<tr><th>Fancy Formatting / Paradigms</th><th>Listed Results</th></tr>
<tr><td>
Car
| brakes | windows
has | yes | yes

Jane
| name | address | social | wings
has | yes | yes | no | no
| mean
is | no
| brakes | windows
has | no | no
</td><td>
Car <has,brakes> yes
Car <has,windows> yes
Jane <has,name> yes
Jane <has,address> yes
Jane <has,social> no
Jane <has,wings> no
Jane <is,mean> no
Jane <has,brakes> no
Jane <has,windows> no
</td></tr>
</table>

[9] Located at http://www.cs.uky.edu/~raphael/linguistics/DATR.cgi. Last accessed 3/16/17.

Chapter 4: A DATR Account of Amharic Simple Present Verbs

As I noted in Chapter 1, in this study I have chosen to limit my analysis to simple present verbs of a triradical nature that begin only with consonants. In order to make my theory slightly more interesting than it might be otherwise, a detail that will also enable me to demonstrate more of the power and versatility of DATR, I am going to also include several irregular verbs. In addition to the challenge of transliterating Amharic orthographical symbols into Roman characters, there are three particular matters stemming from Amharic morphology that my theory can account for, namely, phonological change (e.g. deletion/addition), germination, and root-and-pattern templates.

Concerning the root-and-pattern issue, the verbs I have chosen, which can be found in any standard Amharic dictionary, follow four different patterns. The first set has a stem that follows a CvCCvCv stem pattern, the second a CvCvCv pattern, the third a Cv pattern, and the fourth a CvCvCvCv pattern. DATR easily handles all four of these, even a few irregulars (see below). With regard to phonological change, when a sixth order I in a stem follows d, n, r, z, or l (all alveolars) deletion occurs. Similarly, when a fourth order form follows m, b, l, r, g, q, t or t͡ʃ, it changes to a third order form and a is added (i.e. addition occurs). This actually affects the root-and-pattern template, causing it to change. Again, DATR is easily able to account for these changes. In addition, and as I mentioned earlier in this work, germination is present in Amharic but not orthographically. In my theory I am able to account for gemination, which is not represented orthographically in Amharic, by employing transliteration to show where it does occur. In my theory I do this by providing a transliteration in IPA.

In this chapter, then, I provide the theory I have developed and also offer notation. The theory is, of course, not exhaustive when it comes to the Amharic verb system. It does not, for example, take into consideration other tenses, radicals, or even triradical forms that begin with vowels. If one wishes to use this code, it is likely that in attempting to copy it from this file and pasting it into Finkel's DATR evaluator will not work. This is the case because the word processors will probably introduce interference. Theories should be saved as simple text files with UTF8 encoding.

```
% % % % % % % % % % % % % % % % % % % % % % %
%Created by T. Michael W. Halcomb                    %
%Version 1: 5/1/17                                   %
%Title: Amharic Verbs                                %
%Purpose: Illustrates inheritance, overrides, alternative values%
%    of Amharic simple present verbs in DATR          %
%Email: Michael.Halcomb@AsburySeminary.edu           %
%Version: 2                                          %
% % % % % % % % % % % % % % % % % % % % % % %
```

```
#vars $Cons1: a b t͡ʃ t͡ʃ' d φ g ɲ h ı d͡ʒ k l m n q r s ʃ t t' w j.
#vars $Cons2: a b t͡ʃ t͡ʃ' d φ g ɲ h ı d͡ʒ k l m n q r s ʃ t t' w j.
#vars $Cons3: a b t͡ʃ t͡ʃ' d φ g ɲ h ı d͡ʒ k l m n q r s ʃ t t' w j.
#vars $Cons4: a b t͡ʃ t͡ʃ' d φ g ɲ h ı d͡ʒ k l m n q r s ʃ t t' w j.

%variable declarations; these variables are the consonants used in the sample verbs and
% utilized %by the “Stem” node

Mor_Verb:
  <syn cat> == verb
  <roman> == Prefix:<> “<stem>” Tense_suffix:<> Agr_suffix:<>
  <amharic> == TRANSLIT:<“<roman>”>.

%Mor_Verb is the root node of the inheritance network
%verb denotes the syntactical category
%<roman> sets the path value to specified items (e.g. prefix followed by stem, etc.) and
%each of these will be in Roman (IPA) characters; the code, then, is based on the Roman
%transliteration scheme
%Prefix:<> calls down to the Prefix node; “<stem>”, Tense_suffix:<>, and Agr_suffix:<>
%do the same;
%<amharic> calls down to the TRANSLIT node where the Amharic letters are matched
%with their Roman counterparts, resulting in the Amharic being transliterated in a letter-
%-for-letter fashion

Triradicals:
  <> == Mor_Verb
  <stem> == Stem:<”<root>”>
  <vowel1> == ε
  <vowel2> == ı
  <vowel3> == a
  <vowel3 2 fem sg simp_pres> == i a.

%This begins the first of the class nodes
%The Triradicals node, as well as those that follow it (e.g. Triradicals_Alt,
%Triradicals_Irregular, Triradicals_Irregular2, and Triradicals_Irregular3) all inherit from
%the root node, which calls down to the Stem node thereby selecting the appropriate
```

```
%transliteration pattern; specifics concerning vowel patterns are provided here, allowing
%for different patterns or overrides; note, for example, that each Triradical node has a
%different vowel pattern and different vowel changes (or a lack thereof)

Triradicals_Alt:
  <> == Mor_Verb
  <stem> == Stem2:<”<root>”>
  <vowel3> == a
  <vowel3 2 fem sg simp_pres> == i a.

Triradicals_Irregular:
  <> == Mor_Verb
  <stem> == Stem3:<”<root>”>
  <roman> == “<stem>” Agr_suffix:<>
  <amharic> == TRANSLIT:<”<roman>”>.

Triradicals_Irregular2:
  <> == Triradicals_Irregular
  <roman> == “<stem>” Tense_suffix:<> Agr_suffix:<>
  <root> == a
  <vowel1> ==.

Triradicals_Irregular3:
  <> == Triradicals
  <stem> == Stem4:<”<root>”>
  <vowel1> == ε
  <vowel2> == ε
  <vowel3> == ɪ
  <vowel4> == a
  <vowel4 2 fem sg simp_pres> == i a.

Prefix:
  <1> == ɪ <ɪ>
  <2> == t ɪ
  <3> == j ɪ
  <3 fem> == <2>
  <ɪ comm pl> == n n ɪ
  <> ==.

%The Prefix node creates a means of avoiding redundancy in the code due in large part to
%its economical use of the letter ɪ, which is used in several different environments within
%the prefix slot
```

```
Stem:
  <$Cons1 $Cons2 $Cons3> == $Cons1 “<vowel1>” $Cons2 $Cons2 “<vowel2>”
$Cons3 “<vowel3>”.

Stem2:
  <$Cons1 $Cons2 $Cons3> == $Cons1 “<vowel1>” $Cons2 “<vowel2>” $Cons3
“<vowel3>”.

Stem3:
  <$Cons1> == $Cons1 “<vowel1>”.

Stem4:
  <$Cons1 $Cons2 $Cons3 $Cons4> == $Cons1 “<vowel1>” $Cons2 “<vowel2>”
$Cons3 “<vowel3>” $Cons4 “<vowel4>”.

%The Stem node reveals four distinct verb patterns; this root-and-pattern template forms
%the basis for the code’s transliterator

Tense_suffix:
  <> == l l ɛ
  <3 masc sg simp_pres> == l
  <2 comm pl simp_pres> == l l a
  <3 comm pl simp_pres> == l l u.

%The Tense_suffix node simply indicates the tense suffixes

Agr_suffix:
  <1 comm sg simp_pres> == h u
  <2 masc sg simp_pres> == h
  <2 fem sg simp_pres> == ʃ
  <3 fem sg simp_pres> == t͡ʃ
  <1 comm pl simp_pres> == n
  <2 comm pl simp_pres> == t͡ʃ h u
  <> ==.

%The Agr_suffix node simply indicates the person/number/gender agreement suffixes

TRANSLIT:
  <a> == አ <>
  <b a> == ባ <>
  <b i a> == ቢ አ <>
  <b ɪ> == ብ <>
  <b b ɪ> == ብ <>
  <t͡ʃ> == ች <>
  <t͡ʃt͡ʃ> == ች <>
  <t͡ʃt͡ʃ a> == ቻ <>
```

<t͡ʃt͡ʃ ɪ> == ች <>
<t͡ʃ a> == ጫ <>
<t͡ʃ i> == ጪ <>
<d a> == ዳ <>
<d ɛ> == ደ <>
<φ ɛ> == ፊ <>
<g a> == ጋ <>
<g ɛ> == ገ <>
<g g ɛ> == ገ <>
<g i a> == ጊ አ <>
<ɲ ɲ> == ኝ <>
<h> == ህ <>
<h u> == ሁ <>
<ɪ> == እ <>
<d͡ʒ a> == ጃ <>
<d͡ʒ ɛ> == ጀ <>
<d͡ʒ i a> == ጂ አ <>
<k ɪ> == ክ <>
<k k ɛ> == ከ <>
<k k ɪ> == ክ <>
<l> == ል <>
<l a> == ላ <>
<l i a> == ሊ አ <>
<l ɪ> == ል <>
<l l a> == ላ <>
<l l ɪ> == ል <>
<l l ɛ> == ለ <>
<l l u> == ሉ <>
<m a> == ማ <>
<m ɛ> == መ <>
<m i a> == ሚ አ <>
<m ɪ> == ም <>
<m m ɛ> == መ <>
<m m i a> == ሚ አ <>
<m m ɪ> == ም <>
<n> == ን <>
<n a> == ና <>
<n ɛ> == ነ <>
<n n ɛ> == ነ <>
<n n ɪ> == ን <>
<q a> == ቃ <>
<q ɛ> == ቀ <>
<q ɪ> == ቅ <>
<q i a> == ቂ አ <>
<q q ɛ> == ቀ <>
<q q ɪ> == ቅ <>

```
  <r> == ር <>
  <r a> == ራ <>
  <r i a> == ር አ <>
  <s a> == ሳ <>
  <s ɛ> == ሰ <>
  <ʃ> == ሽ <>
  <ʃ a> == ሻ <>
  <ʃ ɛ> == ሸ <>
  <ʃ i a> == ሺ አ <>
  <t a> == ታ <>
  <t ɛ> == ተ <>
  <t ɪ> == ት <>
  <t' ɛ> == ጠ <>
  <w> == ው <>
  <w ɛ> == ወ <>
  <w w ɛ> == ወ <>
  <j ɪ> == ይ <>
  <j j ɪ> == ይ <>
  <> ==.

%The TRANSLIT node is where the Roman characters corresponding to Amharic letters
%are transliterated

Arrive:
  <> == Triradicals_Alt
  <gloss> == arrive
  <root> == d r s
  <root 2 fem sg simp_pres> == d r ʃ
  <vowel1> == ɛ
  <vowel2> == .

%Arrive is the first of the leaf nodes; it inherits from the Triradicals_Alt node; it is built
%on the root "d r s," which is transliterated into Amharic; note that in the 2 fem sg the s
%becomes ʃ; note the override in terms of vowel patterning here

Ask:
  <> == Triradicals
  <gloss> == ask
  <root> == t' j q.

%Ask inherits from the Triradicals class and has no irregularities or overrides

Be:
  <> == Triradicals_Irregular
  <gloss> == be
  <root> == n
```

```
 <vowel1> == ɛ
 <roman 1 comm sg simp_pres> == <stem> ɲ ɲ
 <roman 3 masc sg simp_pres> == <stem> w
 <roman 3 comm pl simp_pres> == <stem> t͡ʃt͡ʃ a w.

%Note the irregularities in Be (irregular in most of the world's languages) and thus the
%overrides (roman 1 comm, 3 masc, and 3 comm) used to account for this

Beget:
 <> == Triradicals_Alt
 <gloss> == beget
 <root> == w l d
 <root 2 fem sg simp_pres> == w l d͡ʒ
 <vowel1> == ɛ
 <vowel2> == .

Begin:
 <> == Triradicals
 <gloss> == begin
 <root> == d͡ʒ m r.

Break:
 <> == Triradicals
 <gloss> == break
 <root> == s b r.

Build:
 <> == Triradicals
 <gloss> == build
 <root> == g n b
 <vowel2> == ɛ.

Carry:
 <> == Triradicals
 <gloss> == carry
 <root> == ʃ k m
 <vowel2> == ɛ.

Carve:
 <> == Triradicals
 <gloss> == carve
 <root> == w q r.

Exist:
 <> == Triradicals_Irregular2
 <gloss> == exist.
```

```
Have:
  <> == Exist
  <gloss> == have
  <roman 1 comm sg simp_pres> == <stem> Tense_suffix:<> ɲ ɲ
  <roman 3 masc sg simp_pres> == <stem> Tense_suffix:<> w
  <roman 3 comm pl simp_pres> == <stem> Tense_suffix:<> t͡ʃt͡ʃ a w.

Plant:
  <> == Triradicals
  <gloss> == plant
  <root> == t k l.

Play:
  <> == Triradicals_Alt
  <gloss> == play
  <root> == t͡ʃʼ w t
  <root 2 fem sg simp_pres> == t͡ʃʼ w t͡ʃʼ
  <vowel1> == a
  <vowel2> == ɛ
  <vowel3> == a.

Search:
  <> == Triradicals
  <gloss> == search
  <root> == φ l g.

Sit:
  <> == Triradicals
  <gloss> == sit
  <root> == q m t
  <root 2 fem sg simp_pres> == q m t͡ʃʼ
  <vowel2> == ɛ.

Speak:
  <> == Triradicals
  <gloss> == speak
  <root> == n g r
  <vowel1> == a
  <vowel2> == ɛ.

Throw:
  <> == Triradicals_Irregular3
  <gloss> == throw
  <root> == w r w r.
```

```
Use:
  <> == Triradicals
  <gloss> == use
  <root> == t q m
  <vowel2> == ɛ.

Wake:
  <> == Triradicals_Irregular3
  <gloss> == wake
  <root> == q s q s.

Walk:
  <> == Triradicals
  <gloss> == walk
  <root> == r m d
  <root 2 fem sg simp_pres> == r m d͡ʒ
  <vowel1> == a
  <vowel2> == ɛ.

%Begin show and hide of elements to be shown and hidden

#show:
  <roman 1 comm sg simp_pres>
  <roman 2 masc sg simp_pres>
  <roman 2 fem sg simp_pres>
  <roman 3 masc sg simp_pres>
  <roman 3 fem sg simp_pres>
  <roman 1 comm pl simp_pres>
  <roman 2 comm pl simp_pres>
  <roman 3 comm pl simp_pres>
  <amharic 1 comm sg simp_pres>
  <amharic 2 masc sg simp_pres>
  <amharic 2 fem sg simp_pres>
  <amharic 3 masc sg simp_pres>
  <amharic 3 fem sg simp_pres>
  <amharic 1 comm pl simp_pres>
  <amharic 2 comm pl simp_pres>
  <amharic 3 comm pl simp_pres>.

#hide:
Triradicals Triradicals_Alt Triradicals_Irregular Triradicals_Irregular2 Stem Mor_Verb
TRANSLIT Prefix Tense_suffix Agr_suffix.
```

Chapter 5: Summary & Conclusions

In this thesis I have attempted to computationally reproduce the natural transmission of twenty present tense Amharic verbs (i.e. triradicals beginning with consonants) as modeled by the language's speakers. I have rooted my approach in the linguistic theory of network morphology (NM) and modeled it using the DATR parsing engine. One of my hopes is that this work might assist those wanting to work with Amharic in their efforts to more efficiently and effectively engage, understand, and utilize the language for a variety of purposes such as learning Amharic, learning about Amharic, or perhaps cross-linguistic analysis. As noted earlier, I believe this work has the potential to fit well within the realm of computer assisted language learning (CALL) by being of pedagogical use to teachers and of research use to learners. Likewise, it might also provide a means of spell- or form-checking verbs among readers, writers, and translators.

In the first chapter, I provided an overview of Amharic. Specifically, I discussed the *fidel* as an abugida, the verb system's root-and-pattern morphology, and how radicals of each lexeme interacts with prefixes and suffixes. Following that, in Chapter 2, I discussed NM. I drew attention to the fact that NM is concerned with lexemes at the paradigm level. I also noted drew attention to the fact that NM, theoretically speaking, shares many similarities with OOP. In addition, I provided clarity with regard to a handful of key terms used in NM literature.

An overview of DATR was the focus of Chapter 3. As with the previous chapter, here I showed connections with OOP and aimed to shed light on a number of additional key terms and concepts. To reiterate, each of the principles addressed there is important for understanding, navigating, and working within the DATR environment. Finally, in Chapter 4, I set forth my full theory for parsing twenty present tense Amharic verbs. As the code makes clear, I divided those twenty verbs into four sets based on their shared morphological features. The first contained a CvCCvCv stem pattern, the second a CvCvCv pattern, the third a Cv pattern, and the fourth a CvCvCvCv pattern. For the DATR output, please see the Appendix.

In writing, my main hope is that this project will make a contribution, however minimal or sizeable, to the field of Amharic studies in particular and (computational) linguistics in general. In terms of scalability, I have been working extensively on writing theory for the entire verb system but because it is not yet close to being finished, it will not be able to bear fruit until sometime in the future. I do believe, however, that I am off to a good start and this project, in and of itself, hopefully stands as a testament to that. Considering other binyanim, including those beginning with vowels, would be part of such a project. Perhaps a fuller discussion of other morphosyntactic feature sets (e.g. gender, number, person, case, definiteness, respect, etc.) would be useful, too. Amharic, for instance, has more and less formal verb forms, which grammatical gender factors heavily into and, as such, some focus on that might bear fruit.

As I have already stated, I view this project as a work in progress. While quite a bit has been written on Amharic, little has been done in terms of utilizing DATR. I hope to do more in that regard particularly with regard to the verb. Thus, I will increase the scope of coverage as it pertains to the verb system. Perhaps I will venture into nouns as well. As I continue to learn more about the intricacies of DATR, for example, I will be able to tighten up the code and make it more sophisticated at least in part by weeding out any looming unnecessary redundancies. As I was finishing up this work, for instance, Dr. Greg Stump demonstrated a couple of ways I might begin to do that. Unfortunately, I simply did not have time to try to put that into action here. From a theoretically standpoint, I am interested in looking deeper into PATR and PATR-ii as well as exploring in a more in-depth manner the possible relationship between DATR and OOP.

As I plan to continue working on this, I hope others will be encouraged and enter the fray. This could obviously be extended to other parts of speech in Amharic, especially its rich nominal system. Perhaps someone will rise to the occasion and purse such a task. At the end of the day, if I have piqued the reader's attention in either Amharic or DATR or even both, then, as minute as that may seem, I believe will have accomplished much.

Appendix

Beginning on the next page is the output of my DATR theory. These paradigms were generated using Raphael Finkel’s online DATR evaluator and, as such, are in the standard format (layout, color scheme, etc.) it produces. I have arranged the output tables in alphabetical order with the Roman forms preceding the Amharic. The abbreviations in the table, which, for computational purposes I distinguish from the abbreviations in the body of this work and on the Abbreviations page near the end of this document, are defined as follows: simp_pres = simple present; sg = singular; pl = plural; comm = common; 1, 2, and 3 = first, second, and third person respectively; masc = masculine; and fem = feminine.

Arrive

roman	comm							
1	comm	simp_pres	sg	ɪdɛrsallɛhu				
2	masc	simp_pres	sg	tɪdɛrsallɛh	fem	simp_pres	sg	tɪdɛrʃiallɛʃ
3	masc	simp_pres	sg	jɪdɛrsal	fem	simp_pres	sg	tɪdɛrsallɛt͡ʃ
1	comm	simp_pres	pl	mmɪdɛrsallɛn				
2	comm	simp_pres	pl	tɪdɛrsallat͡ʃhu				
3	comm	simp_pres	pl	jɪdɛrsallu				

amharic	comm							
1	comm	simp_pres	sg	እደርሳለሁ				
2	masc	simp_pres	sg	ትደርሳለህ	fem	simp_pres	sg	ትደርሺአለሽ
3	masc	simp_pres	sg	ይደርሳል	fem	simp_pres	sg	ትደርሳለች
1	comm	simp_pres	pl	እንደርሳለን				
2	comm	simp_pres	pl	ትደርሳላችሁ				
3	comm	simp_pres	pl	ይደርሳሉ				

Ask

roman	comm	
1	comm simp_pres sg ɪt'ɛjjɪqallɛhu	
2	masc simp_pres sg tɪt'ɛjjɪqallɛh	fem simp_pres sg tɪt'ɛjjɪqiallɛʃ
3	masc simp_pres sg jɪt'ɛjjɪqal	fem simp_pres sg tɪt'ɛjjɪqallɛt͡ʃ
1	comm simp_pres pl mnɪt'ɛjjɪqallɛn	
2	comm simp_pres pl tɪt'ɛjjɪqallat͡ʃhu	
3	comm simp_pres pl jɪt'ɛjjɪqallu	

amharic	comm	
1	comm simp_pres sg እጠይቃለሁ	
2	masc simp_pres sg ትጠይቃለህ	fem simp_pres sg ትጠይቂአለሽ
3	masc simp_pres sg ይጠይቃል	fem simp_pres sg ትጠይቃለች
1	comm simp_pres pl እንጠይቃለን	
2	comm simp_pres pl ትጠይቃላችሁ	
3	comm simp_pres pl ይጠይቃሉ	

Be

roman	comm	
1	comm / simp_pres / sg: nɛɲɲ	
2	masc / simp_pres / sg: nɛh	fem / simp_pres / sg: nɛʃ
3	masc / simp_pres / sg: nɛw	fem / simp_pres / sg: nɛt͡ʃ
1	comm / simp_pres / pl: nɛn	
2	comm / simp_pres / pl: nɛt͡ʃhu	
3	comm / simp_pres / pl: nɛt͡ʃt͡ʃaw	

amharic	comm	
1	comm / simp_pres / sg: ነኝ	
2	masc / simp_pres / sg: ነህ	fem / simp_pres / sg: ነሽ
3	masc / simp_pres / sg: ነው	fem / simp_pres / sg: ነች
1	comm / simp_pres / pl: ነን	
2	comm / simp_pres / pl: ነችሁ	
3	comm / simp_pres / pl: ነቸው	

Beget

roman	comm			
1	comm	simp_pres		
	sg	ɪwɛldallɛhu		
2	masc	simp_pres	fem	simp_pres
	sg	tɪwɛldallɛh	sg	tɪwɛld͡ʒiallɛʃ
3	masc	simp_pres	fem	simp_pres
	sg	jɪwɛldal	sg	tɪwɛldallɛt͡ʃ
1	comm	simp_pres		
	pl	mnɪwɛldallɛn		
2	comm	simp_pres		
	pl	tɪwɛldallat͡ʃhu		
3	comm	simp_pres		
	pl	jɪwɛldallu		

amharic	comm			
1	comm	simp_pres		
	sg	እወልዳለሁ		
2	masc	simp_pres	fem	simp_pres
	sg	ትወልዳለህ	sg	ትወልጂአለሽ
3	masc	simp_pres	fem	simp_pres
	sg	ይወልዳል	sg	ትወልዳለች
1	comm	simp_pres		
	pl	እንወልዳለን		
2	comm	simp_pres		
	pl	ትወልዳላችሁ		
3	comm	simp_pres		
	pl	ይወልዳሉ		

Begin

roman	comm				
1	comm	simp_pres			
	sg	ɪd͡ʒɛmmɪrallɛhu			
2	masc	simp_pres		fem	simp_pres
	sg	tɪd͡ʒɛmmɪrallɛh		sg	tɪd͡ʒɛmmɪriallɛʃ
3	masc	simp_pres		fem	simp_pres
	sg	jɪd͡ʒɛmmɪral		sg	tɪd͡ʒɛmmɪrallɛt͡ʃ
1	comm	simp_pres			
	pl	mɪnɪd͡ʒɛmmɪrallɛn			
2	comm	simp_pres			
	pl	tɪd͡ʒɛmmɪrallat͡ʃhu			
3	comm	simp_pres			
	pl	jɪd͡ʒɛmmɪrallu			

amharic	comm				
1	comm	simp_pres			
	sg	እጀምራለሁ			
2	masc	simp_pres		fem	simp_pres
	sg	ትጀምራለህ		sg	ትጀምርአለሽ
3	masc	simp_pres		fem	simp_pres
	sg	ይጀምራል		sg	ትጀምራለች
1	comm	simp_pres			
	pl	እንጀምራለን			
2	comm	simp_pres			
	pl	ትጀምራላችሁ			
3	comm	simp_pres			
	pl	ይጀምራሉ			

Break

roman	comm					
1	comm	simp_pres				
	sg	ɪsɛbbɪrallɛhu				
2	masc	simp_pres		fem	simp_pres	
	sg	tɪsɛbbɪrallɛh		sg	tɪsɛbbɪriallɛʃ	
3	masc	simp_pres		fem	simp_pres	
	sg	jɪsɛbbɪral		sg	tɪsɛbbɪrallɛt͡ʃ	
1	comm	simp_pres				
	pl	mnɪsɛbbɪrallɛn				
2	comm	simp_pres				
	pl	tɪsɛbbɪrallat͡ʃhu				
3	comm	simp_pres				
	pl	jɪsɛbbɪrallu				

amharic	comm					
1	comm	simp_pres				
	sg	እሰብራለሁ				
2	masc	simp_pres		fem	simp_pres	
	sg	ትሰብራለህ		sg	ትሰብርአለሽ	
3	masc	simp_pres		fem	simp_pres	
	sg	ይሰብራል		sg	ትሰብራለች	
1	comm	simp_pres				
	pl	እንሰብራለን				
2	comm	simp_pres				
	pl	ትሰብራላችሁ				
3	comm	simp_pres				
	pl	ይሰብራሉ				

Build

roman	comm					
1	comm	simp_pres	sg	ɪgɛnnɛballɛhu		
2	masc	simp_pres	sg	tɪgɛnnɛballɛh	fem, simp_pres, sg	tɪgɛnnɛbiallɛʃ
3	masc	simp_pres	sg	jɪgɛnnɛbal	fem, simp_pres, sg	tɪgɛnnɛballɛt͡ʃ
1	comm	simp_pres	pl	mnɪgɛnnɛballɛn		
2	comm	simp_pres	pl	tɪgɛnnɛballat͡ʃhu		
3	comm	simp_pres	pl	jɪgɛnnɛballu		

amharic	comm					
1	comm	simp_pres	sg	እገነባለሁ		
2	masc	simp_pres	sg	ትገነባለህ	fem, simp_pres, sg	ትገነቢአለሽ
3	masc	simp_pres	sg	ይገነባል	fem, simp_pres, sg	ትገነባለች
1	comm	simp_pres	pl	እንገነባለን		
2	comm	simp_pres	pl	ትገነባላችሁ		
3	comm	simp_pres	pl	ይገነባሉ		

Carry

roman	comm							
1	comm	simp_pres	sg	ɪʃɛkkɛmallɛhu				
2	masc	simp_pres	sg	tɪʃɛkkɛmallɛh	fem	simp_pres	sg	tɪʃɛkkɛmiallɛʃ
3	masc	simp_pres	sg	jɪʃɛkkɛmal	fem	simp_pres	sg	tɪʃɛkkɛmallɛt͡ʃ
1	comm	simp_pres	pl	mnɪʃɛkkɛmallɛn				
2	comm	simp_pres	pl	tɪʃɛkkɛmallat͡ʃhu				
3	comm	simp_pres	pl	jɪʃɛkkɛmallu				

amharic	comm							
1	comm	simp_pres	sg	እሸከማለሁ				
2	masc	simp_pres	sg	ትሸከማለህ	fem	simp_pres	sg	ትሸከሚአለሽ
3	masc	simp_pres	sg	ይሸከማል	fem	simp_pres	sg	ትሸከማለች
1	comm	simp_pres	pl	እንሸከማለን				
2	comm	simp_pres	pl	ትሸከማላችሁ				
3	comm	simp_pres	pl	ይሸከማሉ				

Carve

roman	comm			
1	comm sg	simp_pres: ɪwɛqqɪrallɛhu		
2	masc sg	simp_pres: tɪwɛqqɪrallɛh	fem sg	simp_pres: tɪwɛqqɪriallɛʃ
3	masc sg	simp_pres: jɪwɛqqɪral	fem sg	simp_pres: tɪwɛqqɪrallɛt͡ʃ
1	comm pl	simp_pres: mnɪwɛqqɪrallɛn		
2	comm pl	simp_pres: tɪwɛqqɪrallat͡ʃhu		
3	comm pl	simp_pres: jɪwɛqqɪrallu		

roman	comm			
1	comm sg	simp_pres: ɪʃɛkkɛmallɛhu		
2	masc sg	simp_pres: tɪʃɛkkɛmallɛh	fem sg	simp_pres: tɪʃɛkkɛmiallɛʃ
3	masc sg	simp_pres: jɪʃɛkkɛmal	fem sg	simp_pres: tɪʃɛkkɛmallɛt͡ʃ
1	comm pl	simp_pres: mnɪʃɛkkɛmallɛn		
2	comm pl	simp_pres: tɪʃɛkkɛmallat͡ʃhu		
3	comm pl	simp_pres: jɪʃɛkkɛmallu		

Exist

roman	comm							
1	comm	simp_pres	sg	allɛhu				
2	masc	simp_pres	sg	allɛh	fem	simp_pres	sg	allɛʃ
3	masc	simp_pres	sg	al	fem	simp_pres	sg	allɛt͡ʃ
1	comm	simp_pres	pl	allɛn				
2	comm	simp_pres	pl	allat͡ʃhu				
3	comm	simp_pres	pl	allu				

amharic	comm							
1	comm	simp_pres	sg	አለሁ				
2	masc	simp_pres	sg	አለህ	fem	simp_pres	sg	አለሽ
3	masc	simp_pres	sg	አለ	fem	simp_pres	sg	አለች
1	comm	simp_pres	pl	አለን				
2	comm	simp_pres	pl	አላችሁ				
3	comm	simp_pres	pl	አሉ				

Have

roman	comm			
1	comm sg	simp_pres allɛɲɲ		
2	masc sg	simp_pres allɛh	fem sg	simp_pres allɛʃ
3	masc sg	simp_pres allɛw	fem sg	simp_pres allɛt͡ʃ
1	comm pl	simp_pres allɛn		
2	comm pl	simp_pres allat͡ʃhu		
3	comm pl	simp_pres allɛt͡ʃt͡ʃaw		

amharic	comm			
1	comm sg	simp_pres አለኝ		
2	masc sg	simp_pres አለህ	fem sg	simp_pres አለሽ
3	masc sg	simp_pres አለው	fem sg	simp_pres አለች
1	comm pl	simp_pres አለን		
2	comm pl	simp_pres አላችሁ		
3	comm pl	simp_pres አለቻው		

Plant

roman	comm				
1	comm sg	simp_pres			
		ɪtɛkkɪlallɛhu			
2	masc sg	simp_pres		fem sg	simp_pres
		tɪtɛkkɪlallɛh			tɪtɛkkɪliallɛʃ
3	masc sg	simp_pres		fem sg	simp_pres
		jɪtɛkkɪlal			tɪtɛkkɪlallɛt͡ʃ
1	comm pl	simp_pres			
		mnɪtɛkkɪlallɛn			
2	comm pl	simp_pres			
		tɪtɛkkɪlallat͡ʃhu			
3	comm pl	simp_pres			
		jɪtɛkkɪlallu			

amharic	comm				
1	comm sg	simp_pres			
		እተከላለሁ			
2	masc sg	simp_pres		fem sg	simp_pres
		ትተከላለህ			ትተከሊአለሽ
3	masc sg	simp_pres		fem sg	simp_pres
		ይተከላል			ትተከላለች
1	comm pl	simp_pres			
		እንተከላለን			
2	comm pl	simp_pres			
		ትተከላላችሁ			
3	comm pl	simp_pres			
		ይተከላሉ			

Play

roman	comm					
1	comm	simp_pres				
	sg	ɪt͡ʃawɛtallɛhu				
2	masc	simp_pres		fem	simp_pres	
	sg	tɪt͡ʃawɛtallɛh		sg	tɪt͡ʃawɛt͡ʃallɛʃ	
3	masc	simp_pres		fem	simp_pres	
	sg	jɪt͡ʃawɛtal		sg	tɪt͡ʃawɛtallɛt͡ʃ	
1	comm	simp_pres				
	pl	mnɪt͡ʃawɛtallɛn				
2	comm	simp_pres				
	pl	tɪt͡ʃawɛtallat͡ʃhu				
3	comm	simp_pres				
	pl	jɪt͡ʃawɛtallu				

amharic	comm				
1	comm	simp_pres			
	sg	እጫወታለሁ			
2	masc	simp_pres		fem	simp_pres
	sg	ትጫወታለህ		sg	ትጫወጫለሽ
3	masc	simp_pres		fem	simp_pres
	sg	ይጫወታል		sg	ትጫወታለች
1	comm	simp_pres			
	pl	እንጫወታለን			
2	comm	simp_pres			
	pl	ትጫወታላችሁ			
3	comm	simp_pres			
	pl	ይጫወታሉ			

Search

roman	comm							
1	comm	simp_pres	sg	ɪφɛllɪgallɛhu				
2	masc	simp_pres	sg	tɪφɛllɪgallɛh	fem	simp_pres	sg	tɪφɛllɪgiallɛʃ
3	masc	simp_pres	sg	jɪφɛllɪgal	fem	simp_pres	sg	tɪφɛllɪgallɛt͡ʃ
1	comm	simp_pres	pl	mnɪφɛllɪgallɛn				
2	comm	simp_pres	pl	tɪφɛllɪgallat͡ʃhu				
3	comm	simp_pres	pl	jɪφɛllɪgallu				

amharic	comm							
1	comm	simp_pres	sg	እፈልጋለሁ				
2	masc	simp_pres	sg	ትፈልጋለህ	fem	simp_pres	sg	ትፈልጊአለሽ
3	masc	simp_pres	sg	ይፈልጋል	fem	simp_pres	sg	ትፈልጋለች
1	comm	simp_pres	pl	እንፈልጋለን				
2	comm	simp_pres	pl	ትፈልጋላችሁ				
3	comm	simp_pres	pl	ይፈልጋሉ				

Sit

roman	comm	
1	comm simp_pres sg ɪteqqɛmallɛhu	
2	masc simp_pres sg tɪteqqɛmallɛh	fem simp_pres sg tɪteqqɛmiallɛʃ
3	masc simp_pres sg jɪteqqɛmal	fem simp_pres sg tɪteqqɛmallɛt͡ʃ
1	comm simp_pres pl mnɪteqqɛmallɛn	
2	comm simp_pres pl tɪteqqɛmallat͡ʃhu	
3	comm simp_pres pl jɪteqqɛmallu	

amharic	comm	
1	comm simp_pres sg እቀመጣለሁ	
2	masc simp_pres sg ትቀመጣለህ	fem simp_pres sg ትቀመጪአለሽ
3	masc simp_pres sg ይቀመጣል	fem simp_pres sg ትቀመጣለች
1	comm simp_pres pl እንቀመጣለን	
2	comm simp_pres pl ትቀመጣላችሁ	
3	comm simp_pres pl ይቀመጣሉ	

Speak

roman	comm	
1	comm simp_pres sg ɪmaggɛrallɛhu	
2	masc simp_pres sg tɪmaggɛrallɛh	fem simp_pres sg tɪmaggɛriallɛʃ
3	masc simp_pres sg jɪmaggɛral	fem simp_pres sg tɪmaggɛrallɛt͡ʃ
1	comm simp_pres pl ɪnɪmaggɛrallɛn	
2	comm simp_pres pl tɪmaggɛrallat͡ʃhu	
3	comm simp_pres pl jɪmaggɛrallu	

amharic	comm	
1	comm simp_pres sg እናገራለሁ	
2	masc simp_pres sg ትናገራለህ	fem simp_pres sg ትናገርአለሽ
3	masc simp_pres sg ይናገራል	fem simp_pres sg ትናገራለች
1	comm simp_pres pl እንናገራለን	
2	comm simp_pres pl ትናገራላችሁ	
3	comm simp_pres pl ይናገራሉ	

Throw

roman	comm			
1	comm: simp_pres	sg: ɪwɛrɛwɪrallɛhu		
2	masc: simp_pres	sg: tɪwɛrɛwɪrallɛh	fem: simp_pres	sg: tɪwɛrɛwɪriallɛʃ
3	masc: simp_pres	sg: jɪwɛrɛwɪral	fem: simp_pres	sg: tɪwɛrɛwɪrallɛt͡ʃ
1	comm: simp_pres	pl: mnɪwɛrɛwɪrallɛn		
2	comm: simp_pres	pl: tɪwɛrɛwɪrallat͡ʃhu		
3	comm: simp_pres	pl: jɪwɛrɛwɪrallu		

amharic	comm			
1	comm: simp_pres	sg: እወር		
2	masc: simp_pres	sg: ትወር	fem: simp_pres	sg: ትወር
3	masc: simp_pres	sg: ይወር	fem: simp_pres	sg: ትወር
1	comm: simp_pres	pl: እንወር		
2	comm: simp_pres	pl: ትወር		
3	comm: simp_pres	pl: ይወር		

Use

roman	comm							
1	comm	simp_pres	sg	ɪtɛqqɛmallɛhu				
2	masc	simp_pres	sg	tɪtɛqqɛmallɛh	fem	simp_pres	sg	tɪtɛqqɛmiallɛʃ
3	masc	simp_pres	sg	jɪtɛqqɛmal	fem	simp_pres	sg	tɪtɛqqɛmallɛt͡ʃ
1	comm	simp_pres	pl	mnɪtɛqqɛmallɛn				
2	comm	simp_pres	pl	tɪtɛqqɛmallat͡ʃhu				
3	comm	simp_pres	pl	jɪtɛqqɛmallu				

amharic	comm							
1	comm	simp_pres	sg	እተቀማለሁ				
2	masc	simp_pres	sg	ትተቀማለህ	fem	simp_pres	sg	ትተቀሚአለሽ
3	masc	simp_pres	sg	ይተቀማል	fem	simp_pres	sg	ትተቀማለች
1	comm	simp_pres	pl	እንተቀማለን				
2	comm	simp_pres	pl	ትተቀማላችሁ				
3	comm	simp_pres	pl	ይተቀማሉ				

Wake

roman	comm							
1	comm	simp_pres	sg	ɪqɛsɛqɪsallɛhu				
2	masc	simp_pres	sg	tɪqɛsɛqɪsallɛh	fem	simp_pres	sg	tɪqɛsɛqɪsiallɛʃ
3	masc	simp_pres	sg	jɪqɛsɛqɪsal	fem	simp_pres	sg	tɪqɛsɛqɪsallɛt͡ʃ
1	comm	simp_pres	pl	mnɪqɛsɛqɪsallɛn				
2	comm	simp_pres	pl	tɪqɛsɛqɪsallat͡ʃhu				
3	comm	simp_pres	pl	jɪqɛsɛqɪsallu				

amharic	comm							
1	comm	simp_pres	sg	እቀሰቅሳለሁ				
2	masc	simp_pres	sg	ትቀሰቅሳለህ	fem	simp_pres	sg	ትቀሰቅ
3	masc	simp_pres	sg	ይቀሰቅሳል	fem	simp_pres	sg	ትቀሰቅሳለች
1	comm	simp_pres	pl	እንቀሰቅሳለን				
2	comm	simp_pres	pl	ትቀሰቅሳላችሁ				
3	comm	simp_pres	pl	ይቀሰቅሳሉ				

Walk

roman	comm					
1	comm	simp_pres				
	sg	ɪrammɛdallɛhu				
2	masc	simp_pres		fem	simp_pres	
	sg	tɪrammɛdallɛh		sg	tɪrammɛd͡ʒiallɛʃ	
3	masc	simp_pres		fem	simp_pres	
	sg	jɪrammɛdal		sg	tɪrammɛdallɛt͡ʃ	
1	comm	simp_pres				
	pl	mnɪrammɛdallɛn				
2	comm	simp_pres				
	pl	tɪrammɛdallat͡ʃhu				
3	comm	simp_pres				
	pl	jɪrammɛdallu				

amharic	comm			
1	comm	simp_pres		
	sg	እራመዳለሁ		
2	masc	simp_pres	fem	simp_pres
	sg	ትራመዳለህ	sg	ትራመጂአለሽ
3	masc	simp_pres	fem	simp_pres
	sg	ይራመዳል	sg	ትራመዳለች
1	comm	simp_pres		
	pl	እንራመዳለን		
2	comm	simp_pres		
	pl	ትራመዳላችሁ		
3	comm	simp_pres		
	pl	ይራመዳሉ		

Abbreviations

The abbreviations below, which are used throughout the body of this work, follow the Leipzig glossing format.

1	first person
2	second person
3	third person
C	common (not listed in Leipzig)
F	feminine
Inf	infinitve
M	masculine
Sg	singular
Simp_pres	simple present tense
PRF	perfect tense
Pl	Plural

Bibliography

Alemu, Besufikad. 2013. *A Named Entity Recognition for Amharic*. Master's thesis at Addis Ababa University. Addis Ababa, Ethiopia.

Amberber, Mengistu. 2008. *Semantic Primes in Amharic*. In C. Goddard (ed.), *Cross-Linguistic Semantics*, 83-119. Philadelphia, PA: John Benjamins Publishing.

Antoniou G., and Mary-Anne Williams. 1997. *Nonmonotonic Reasoning*. Cambridge, MA: MIT Press.

Bayou, Abiyot. 2000. *Developing Automatic Word Parser for Amharic Verbs and Their Derivation*. Master's thesis at Addis Ababa University. Addis Ababa, Ethiopia.

Bayu, Bayu. 2002. *Automatic Morphological Analyzer for Amharic: An Experiment Involving Unsupervised Learning and Autosegmental Analysis Approaches*. Master's thesis at Addis Ababa University. Addis Ababa, Ethiopia.

Brody, Parker. 2014. *Inferential-Realizational Morphology and Affix Ordering: Evidence from the Agreement Patterns of Basque Auxiliary Verbs*. Master's thesis at the University of Kentucky.

Brown, Dunstan and Andrew Hippisley. 2012. *Network Morphology*. Cambridge: Cambridge University Press.

Bussmann, Hadumod. 2006. In Gregory P. Trauth and Kerstin Kazzazi (eds.), *Routledge Dictionary of Language and Linguistics*. New York: Routledge.

Cercone, Nick. 1983. *Computational Linguistics*. Oxford: Pergamon Press.

Clark, Simon C., Fox, Chris, and Shalom Lappin. 2013. *The Handbook of Computational Linguistics and Natural Language Processing*. Malden, MA: Wiley-Blackwell.

Corbett, Greville G. and Norman M. Fraser. 1993. *Network Morphology: A DATR Account of Russian Nominal Inflection*. In *Journal of Linguistics* 29/1. 113-42.

Daniels, Peter T. 1990. *Fundamentals of Grammatology*. In J*ournal of the American Oriental Society* 110/4. 727-31.

Demelash, Biruk. 2013. Linguistically Motivated Amharic Information Retrieval. Master's thesis at Addis Ababa University. Addis Ababa, Ethiopia.

Evans R., and G. Gazdar. 1996. DATR: *A Language for Lexical Knowledge Representation*. In *Computational Linguistic*s 22/2. 167-216.

Fabri, R. 2014. *Linguistic Introduction: The Orthography, Morphology and Syntax of Semitic Languages*. In I. Zitouni (ed.), *Natural Language Processing of Semitic Languages.* Heidelberg: Springer.

Février, James G. 1995. *Histoire de l'ecriture*. Paris: Pavot.

Gasser, Michael and Mulugeta Wondwossen. 2012. *Learning Morphological Rules for Amharic Verbs Using Inductive Logic Programming*. Paper presented at *SALTMIL-AfLaT*.

Gasser, Michael. 2010. *A Dependency Grammar for Amharic*. Paper presented at the *Workshop on Language Resources and Human Language Technologies for Semitic Languages*. Bloomington, Indiana.

______. 2011. *HornMorpho: A System for Morphological Processing of Amharic, Oromo, and Tigrinya*. Paper presented at the *Conference on Human Language Technology for Development*. Alexandria, Egypt.

______. 2012. Toward *a Rule-Based System for English-Amharic Translation*. Paper presented at the *SALTMIL/AfLaT Conference*. Istanbul, Turkey.

Gebreegziabher, Nirayo H. 2011. *Modeling Improved Amharic Syllabification Algorithm*. Master's thesis at Addis Ababa University. Addis Ababa, Ethiopia.

Grishman, Ralph. 1986. *Computational Linguistics: An Introduction*. New York: Cambridge University Press, 1986.

Halcomb, T. Michael W. 2015. *Introducing Amharic: An Interactive Workbook*. Wilmore, KY: GlossaHouse, 2015.

Hausser, Roland. 2014. *Foundations of Computational Linguistics: Human-Computer Communication in Natural Language*. Berlin, Heidelberg: Springer.

Hippisley, Andrew. 2010. Lexical Analysis. In Nitin Indurkhya and Fredrick J. Damerau (eds.), *Handbook of Natural Language Processing*. Boca Raton, FL: Chapman & Hall/CRC. 31-58.

________. 2016. *Network Morphology*. In Andrew Hippisley and Gregory Stump (eds.), *The Cambridge Handbook of Morphology*. Cambridge: Cambridge University Press. 482-510.

Householder, Fred W. 1959. *Review of Chadwick's The Decipherment of Linear B*. In *Classical Journal* 54. 379-83.

Keller, Bill. 1996. *An Evaluation Semantics for DATR Theories*. In *COLING-96: Proceedings of the 16th Conference on Computational Linguistics*. Copenhagen: Copenhagen Center for Sprogteknologi. 646-51.

Kilbury, James, Petra Naerger and Ingrid Rinz. 1991. *DATR as a Lexical Component for PATR*. In *Proceedings of the 5th EACL Conference*. 137-42.

Kramer, Ruth. 2010. "Gender in Amharic: A Morphosyntactic Approach to Natural and Grammatical Gender." In *Language Sciences* 43. 102-15.

Leslau, Wolf. 1945. *The Influence of Cushitic on the Semitic Languages of Ethiopia: A Problem of Substratum*. *Word* 1/1. 59-82.

___________. 1968. *Amharic Textbook*.Wiesbaden: Otto Harrassowitz.

___________. 1995. *Reference Grammar of Amharic*.Wiesbaden: Otto Harrassowitz.

Little, Greta D. 1974. "Syntactic Evidence of Language Contact: Cushitic Influence in Amharic." In Roger W. Shuy and Charles-James N. Bailey (eds.), *Towards Tomorrow's Linguistics*. Washington D.C.: Georgetown University Press. 267-75.

Lyovin, Anatole, Brett Kessler, and William R. Leben. 2017. *An Introduction to the Languages of the World*. Oxford: Oxford University Press.

Melby, Alan K. and C. Terry Warner. 1995. *The Possibility of Language: A Discussion of the Nature of Language with Implications for Human and Machine Translation*. Philadelphia, PA: John Benjamins Publishing.

Odersky, Martin. 2004. *ECOOP 2004: Object-Oriented Programming: 18th European Conference Proceedings*. Berlin, Heidelberg: Springer.

Schluter, Kevin. 2008. Amharic Internal Reduplication and Foot Structure: A Word-Based Approach. *Kansas Working Papers in Linguistics* 31. 287-301.

Seidl, Martina, M. Scholz, C. Huemer, and G. Kappel. 2014. *UML @ Classroom: An Introduction to Object-Oriented Programming*. Berlin, Heidelberg: Springer.

Sikkel, Klaas. 2012. *Parsing Schemata: A Framework for Specification and Analysis of Parsing Algorithms*. Leiden: Springer.

Stewart, Tom. 2008. *A Consumer's Guide to Contemporary Morphological Theories*. In *Ohio State University Working Papers in Linguistics* 58. 138-230.

Stump, Gregory. 2001. *Inflectional morphology: A Theory of Paradigm Structure*. Cambridge: Cambridge University Press.

Sudo, Yasutada. “The Syntax and Semantics of Indexical Shifting in Modern Uyghur.” Unpublished Generals Paper (2010).

Tadross, Andrew and Abraham Teklu. 2015. *The Essential Guide to Amharic the National Language of Ethiopia*. Oakland, CA: Peace Corps, 2015.

Zaborski, Andrzej. 1975. *The Verb in Cushitic*. Krakow: Naklad Universitetu Jagiellenskiego.

Vita

Bachelor of Science; Biblical Studies and Youth & Family Ministries, Kentucky Christian University

Master of Divinity; Lexington Theological Seminary

Master of Arts in Biblical Studies; Asbury Theological Seminary

Doctor of Philosophy in Biblical Studies; Asbury Theological Seminary

Master of Arts in Linguistic Theory & Typology; University of Kentucky

T. Michael W. Halcomb