

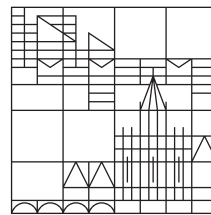
Syntactic Constraints on the Morphophonology of Reduplication: A Theoretical and Computational Perspective

Master Thesis
presented

by
Romi Aobao Hill

at the

Universität
Konstanz



Faculty of Linguistics
Speech and Language Processing

- | | |
|-----------------|------------------------------|
| 1. Evaluated by | Junior Professor Colin Davis |
| 2. Evaluated by | Dr. Tina Bögel |

Konstanz, 2023

Romi Aobao Hill

Syntactic Constraints on the
Morphophonology of Reduplication:
A Theoretical and Computational
Perspective

Acknowledgements

I would like to express my heartfelt appreciation for my first supervisor, Junior Professor Colin Davis. His insight, guidance, and numerous hours of discussion were pivotal in shaping the direction of this thesis. I am also deeply grateful for his encouragement and invaluable advice beyond my degree.

I extend my sincere thanks to my second supervisor, Dr. Tina Bögel, who not only supported me during my thesis but also provided me the opportunity to work as a research assistant throughout my academic journey here.

My friends, both here and back home, have consistently supported and motivated me during my master's degree. In particular, I want to express my gratitude to Iara and Naomi for our (perhaps overly extended) coffee breaks throughout the semester and their unwavering emotional support, both within and outside our academic lives. Special and sincere appreciation also goes to Lukas, with whom I shared many Feierabendbiers, and who generously offered to read a final draft of this thesis shortly before submission.

Finally, my parents, Robin and Jiami, have consistently stood by my side, offering unconditional support even when I'm far from home. Their lifelong support is something I will forever cherish.

Abbreviations and Glossing Conventions

1, 2, 3	1st, 2nd, 3rd person
1S	First personal agreement
APPL	applicative
ASP	aspect
AUG	augment
DEF	definitive
FV	final vowel
NC	noun class marker
NEG	negative
NOM	nominative
PL	plural
POL	polarity
PTC	participle
RECIP	reciprocal
RED	reduplicant
SBJ	subjunctive
SG	singular

Table of Contents

1. Introduction	1
1.1. Distributed Morphology	2
1.2. Reduplication	7
1.3. Bantu Language Family	9
1.4. Research Questions and Outline	10
2. Zulu	12
2.1. Reduplication of Bisyllabic Roots	12
2.2. Reduplication of Monosyllabic Roots	16
2.3. Reduplication of Vowel Initial Roots	21
2.4. Summary	28
3. Kerewe	30
3.1. A Puzzle: The Final Vowel on The Reduplicant	31
3.2. Mismatch between the Reduplicant and Base	33
3.3. Partial Reduplication	37
3.4. Summary	41
4. Theoretical Implications	42
5. Computational Implementation	45
5.1. Usage	45
5.2. Input Files	46
5.3. Output of the Program	48
5.4. Structure of Program	52
5.5. Read in Files	52
5.5.1. Phrase Structure Rules	52
5.5.2. VI rules	54
5.5.3. Phonological Rules	54
5.6. Generate Underlying Base Structure	54
5.7. Apply VI Rules to Underlying Structure	56
5.7.1. Mark Depth of Structure	56
5.7.2. Apply VI Rules to Structure	57

5.8. Reduplicate Base Structure	58
5.8.1. Apply RED Phonological Material	59
5.9. General Phonological Processes	60
5.10. Save Output	61
5.11. Evaluation	62
6. Conclusion and Further Questions	64
Appendix	69
A. List of Output Words	70
A.1. Zulu	70
A.2. Kerewe	72

CHAPTER 1

Introduction

This thesis seeks to analyze word internal patterns that emerge from the morphophonological process of reduplication. Specifically, I argue that these patterns can be predicted by positing a hierarchical syntactic structure within words, as has been hypothesized by theoretical frameworks such as Distributed Morphology (Halle & Marantz, 1993). Reduplication is the morphophonological process in which all or part of the word is repeated to express some sort of meaning. For example, in the Australian language Warlpiri, plurality in nominals is expressed by ‘repeating’ the full nominal stem, as shown in example 1.

- (1) Warlpiri Reduplication (Hayes, 2009, p. 105)
- a. *kuɽu*
child
“child”
 - b. *kuɽu-kuɽu*
RED-child
“children”

Even in this relatively simple example, we can see that reduplication is a process that interacts with both phonology and morphosemantics. The *form* of the repeated segment is sensitive to the phonology of the word, and the *function* of reduplication is used to express some grammatical feature. Reduplication, then, is a process that interacts with morphology, phonology, semantics, and - as I will argue - syntax.

As a process that sits at the interface of linguistic subsystems, an investigation into the behavior of reduplication can contribute significantly to our understanding of language. One overarching goal of theoretical linguistics is to uncover the set of general principles that govern the nature of language. These principles should be constrained enough to explain the commonalities between languages, and also be broad enough to capture the diversity of the world’s languages. One prominent hypothesis in the literature thus far has been that these principles are relatively uniform at the underlying syntactic structure, but the expression of its phonological form adheres to language-specific requirements. In this view, the diversity of the world’s languages is due to the “surface level” phonological expression of a relatively uniform underlying syntactic structure.

By probing into the interactions between - for example - morphology and syntax, we can directly test the ways in which the mapping of syntactic structure to its

phonological form is applied. In doing so, we may find that these underlying syntactic structures influence how surface level structures are expressed, and thereby account for facts which would otherwise be unexplained.

The findings presented in this thesis contribute to the growing body of literature on the morphosyntactic interface, and the ways in which syntactic structure may influence the morphophonological expression of words (e.g. Moskal, 2015; Newell, 2009; Nevins, 2012). Crucially, these findings suggest that positing syntactic structure “all the way down” (Harley & Noyer, 1999, p. 3) to the word level provides unique insight into these seemingly unexpected linguistic patterns.

1.1 Distributed Morphology

This section introduces the framework *Distributed Morphology* (Halle & Marantz, 1993), where I discuss the key concepts that will be necessary for the analyses presented in chapters 2 and 3. Distributed Morphology is a framework that aims to be a complete model of syntax and (morpho)-phonology, with a particular focus on the *interactions* between the syntactic part of the grammar, and the phonological part. To do so, the theory holds three central hypotheses (Halle & Marantz, 1994). These are:

1. “*Syntactic Hierarchical Structure All The Way Down*”

- The operations that build the syntax and morphology are *the same*. Consequently, it follows that there is evidence of constituent structure at the word level, in addition to the phrasal level.

2. “*Late Insertion*”

- Morphemes are purely abstract, syntactic heads, and do not possess phonological information inherently. Their phonological content only after the syntactic structure has been built.¹

3. “*Underspecification*”

- A phonological item can be assigned to a morpheme if that item is defined as corresponding to some or all (i.e. a subset) of the grammatical features that the morpheme possesses.

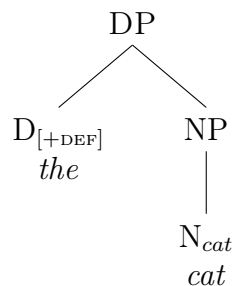
These hypotheses distinguish Distributed Morphology from other theories on the architecture of grammar. For example, in the *Word and Paradigm* approach (Stump, 2001), words are analyzed as holistic and unsegmentable, and do not contain a hierarchical structure. It is also distinct from other syntactic frameworks such as Lexical

¹Distributed Morphology uses the term “morpheme” slightly differently to other areas of linguistics. Traditionally, a morpheme refers to the phonological chunk, but in Distributed Morphology, it refers to the abstract terminal node.

Functional Grammar (LFG) (Bresnan, 2015), which assumes that different principles of composition apply to morphology and syntax. Consequently, the terminal nodes of syntactic structures in LFG are fully inflected word forms, and syntactic processes cannot manipulate internal morphological structure. Distributed Morphology, on the other hand, does not posit separate processes to form complex words and phrases: instead, they are the *same*.²

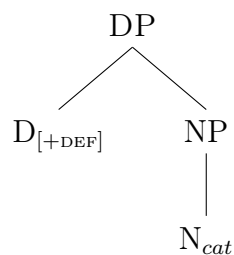
To exemplify how these hypotheses are used to model the structure of words, I will step through a simple example of the English phrase “the cat”. The final derivation for this phrase is shown in figure 2 (Embick, 2015, p. 10). There are a few steps we need to discuss before arriving at figure 2, but the example gives us a glimpse into the kind of structure we are working towards.

(2) *the cat*: Final Derivation



To derive this final structure, we first need a structure that specifies the syntax of the phrase “the cat”, consistent with the “*Syntactic Structure All The Way Down*” hypothesis. Additionally, this structure will have no phonology assigned to it just yet, consistent with the “*Late Insertion*” hypothesis. Assuming the determiner phrase analysis (Abney, 1987), the initial derivation of the underlying structure is shown in figure 3.

(3) *the cat*: Initial Derivation



Once the structure of the word has been built in the syntax, the structure is sent to the phonological part of the grammar, and phonological content can be

²Though see *Lexical-Realizational Functional Grammar* (Asudeh & Siddiqi, 2022) for an endeavor to integrate LFG and Distributed Morphology.

assigned. The assignment of phonological material is conducted via a process called *Vocabulary Insertion* (VI). In this process, objects referred to as *Vocabulary Items* are inserted into the syntactic structure. A *Vocabulary Item* is a relation between a phonological exponent and an array of semantic and syntactic features (McGinnis-Archibald, 2016, p. 391). This item is used in the VI process to assign phonological material to relevant positions in the structure. Some examples of Vocabulary Items are presented in example 4.

(4) Vocabulary Item Examples

- | | |
|---|--|
| a. $N_{cat} \longleftrightarrow \text{cat}$ | d. $D_{[-\text{DEF}]} \longleftrightarrow \text{a} / _ \text{C}$ |
| b. $N_{armadillo} \longleftrightarrow \text{armadillo}$ | e. $D_{[-\text{DEF}]} \longleftrightarrow \text{an} / _ \text{V}$ |
| c. $V_{play} \longleftrightarrow \text{play}$ | f. $D_{[+\text{DEF}]} \longleftrightarrow \text{the}$ |

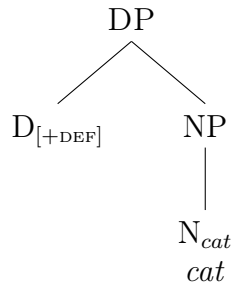
The left side of the mapping relation encodes the syntactic and semantic information of the morpheme, and the right side specifies the phonological expression of the morpheme.³ Vocabulary items specify phonological material for both lexical roots (examples 4a to 4c) and functional items (examples 4d to 4f). Both lexical and functional morphemes include a subscript that encodes relevant features, such as indefiniteness and plurality. For lexical roots, the subscript specifies its semantic interpretation. This subscript is merely a unique identifier: hypothetically, I could have chosen this identifier to be written in a language other than English, or even a set of integers (Harley, 2014). For clarity, I choose to specify this identifier as the root in English, though I emphasize that lexical morphemes do not contain any information on their phonology inherently. This information is specified entirely by the right side of the mapping relation.⁴ Finally, Vocabulary Items can be context specific, as shown in examples 4d and 4e.

Using these Vocabulary Items, the Vocabulary Insertion process can be applied to the structure presented in figure 3. It has been argued in the literature (Bobaljik, 2000) that this process is applied from the root outwards: that is, from the bottom up. This direction of the Vocabulary Insertion process will also be assumed for the upcoming analysis, and we will see that it is in fact *necessary* for the reduplicant to receive its phonological material from morphemes in a lower structural position. Based on the structure presented in figure 3, the VI process can use the Vocabulary Items posited in example 4a to the lowest part of the structure. This is shown in figure 5.

³To be more concrete, it may have been best to use IPA to represent the phonological strings of each morpheme. However, since the fine phonetic details of these morphemes are not relevant to the present discussion, I choose to use the orthographic representations.

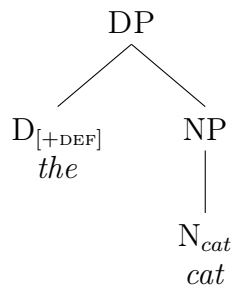
⁴Some authors, such as Embick (2000) and Embick and Halle (2005), draw on suppletion patterns to argue that some roots do contain phonological material inherently. However, as this discussion is not relevant to the upcoming analysis, I assume that the phonological material of morphemes are specified entirely on the right side of the mapping relation.

- (5) *the cat*: VI Applied to N_{cat}



The Vocabulary Item in example 4f can then be applied to the highest morpheme $D_{[+DEF]}$, resulting in the final derivation, repeated in figure 6.

- (6) *the cat*: Final Underlying Syntactic Structure

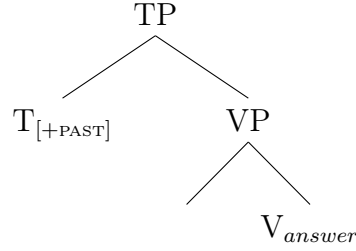


In this example, we have seen how terminal nodes (in this case, free morphemes) in a syntactic structure are assigned their phonological form within the Distributed Morphology framework. I now turn to a brief discussion on how complex words are derived within this framework, as the focus of this thesis is on complex words. In Distributed Morphology, complex words are assumed to be constructed from a combination of syntactic nodes (McGinnis-Archibald, 2016, p. 392). It follows, then, that both free and bound morphemes are terminal nodes of a syntactic structure, and the grammatical features associated with functional (i.e. non-lexical) morphemes are assigned by their position within the syntactic tree. As these positions are generally taken to be language universal, the surface level morphemes order within the complex word is determined by head movement (Julien, 2002; Vicente, 2007). For example, Julien (2002, p. 56) provides the following example for the Uralic language Northern Saami.

- (7) Northern Saami
 a. *mu-n i-n vástid-án*
 I-NOM NEG-1s answer-PAST.PTC
 ‘‘I answered briefly’’

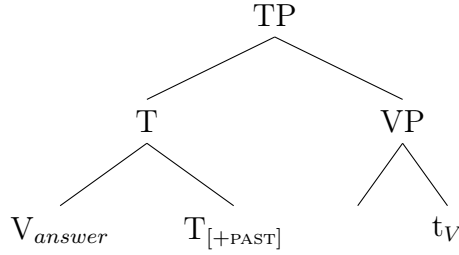
The complex head *vástid-án* contains two morphemes: the lexical root *vástid*; and the past tense suffix *án*. Assuming that verb phrases are complements of the tense phrase, the initial derivation of the TP is depicted in figure 8, prior to Vocabulary Insertion.

(8) Initial Derivation of *vástid-án*



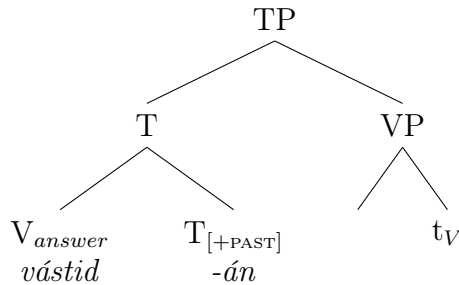
To derive the correct surface morpheme order, the lexical root undergoes head movement, as shown in figure 9.⁵

(9) Post Head Movement of *vástid-án*



Finally, morphemes are assigned their Vocabulary Items, and structure as shown in figure 10.

(10) Vocabulary Insertion the Verb in example 7



⁵See Julien (2002) for a comprehensive discussion on the motivations that trigger word-forming head movement operations.

As the focus of this thesis is on the morphophonology of reduplication, and whether this is constrained by internal syntactic structure, I am not concerned with the precise motivations for head movement in Zulu and Kerewe verbs. As we will see, the syntactic structure of complex words post-head movement will be sufficient to explain the facts of the morphophonology of reduplication. For the remainder of this thesis, then, tree diagrams of complex words are depicted after head movement has occurred.

Up to now, I have outlined the important aspects of Distributed Morphology that are required for the remainder of the thesis. We have seen two hypotheses that characterize Distributed Morphology: complex words are hierarchically and syntactically structured; and the syntactic part of a word is built before the phonology is applied. If these hypotheses can predict patterns from reduplication processes, then this provides compelling evidence that words exhibit internal syntactic structure. Although the analysis of this simple example has not contributed much insight into the morpho-phonological expression of words, these tools will allow us to gain insights into patterns of reduplication, as presented in chapters 2 and 3. Before analyzing these data, however, I will first provide the necessary background on *reduplication*.

1.2 Reduplication

Reduplication is the morpho-phonological process in which all or part of a word is repeated to convey some form of meaning. The morpheme that is reduplicated is referred to as the *reduplicant*, and the part of the word that is to be copied is known as the *base*. One simple example of this phenomenon is the Warlpiri example presented in example 1 above, repeated below in example 11. In this example, the whole word is repeated to express plurality, where the reduplicant is underlined and glossed as RED.

(11) Full Reduplication (Hayes, 2009, p. 105)

- | | |
|------------------------------------|---|
| a. <i>kuɽu</i>
child
“child” | b. <i><u>kuɽu</u>-kuɽu</i>
RED -child
“children” |
|------------------------------------|---|

When the whole word is copied, such as in example 11, this process is known as *full* reduplication. Full reduplication may not necessarily copy the entire word, but may instead target the root, or the stem. For example, in the Bantu language Kinande, nouns are reduplicated to convey the meaning “a real X” (Mutaka & Hyman, 1990, p. 76). Nouns usually contain two prefixes: a class marker (NC); and an accompanying “augment” prefix (AUG) that is determined by a combination of

syntactic and semantic features. When nouns are reduplicated in this language, the root is copied completely, and the prefixes are omitted from the reduplicant.⁶

(12) Root reduplication in Kinande

- | | |
|---|---|
| a. <i>o-ku-gulu</i>
AUG-NC-leg
“leg” | c. <i>o-ku-gulu-gulu</i>
AUG-NC- RED -leg
“a real leg” |
| b. <i>o-mu-góngò</i>
AUG-NC-back
“back” | d. <i>o-mu-góngo-góngò</i>
AUG-NC- RED -back
“a real back” |

Although the reduplicant does not copy the whole word, the root is copied in its entirety. In other languages, reduplication may not necessarily copy the entire word, root, or stem, and only a subpart of the target. This type of reduplication is known as *partial* reduplication, and the reduplicant is often analyzed as a prosodic template (Marantz, 1982). For example, in the Oceanic language Mokilese, the reduplicant targets the first syllable of the base. Additionally, there is a prosodic requirement that the reduplicant must satisfy. The reduplicant must be a heavy syllable, in the sense that it must contain either a coda or a long vowel. In the event that the first syllable of the base is not heavy, the reduplicant also copies segments from the second syllable until the heavy syllabic requirement is satisfied. This is shown in example 13 below (Urbanczyk, 2017), where the syllabic weight of the first syllable of the base in example 13a is heavy and is repeated exactly in its reduplicated form in example 13e. In the remaining examples, however, the weight of the first syllable in the non-reduplicated verb is light and the reduplicant in their reduplicated forms is not an exact copy of that syllable. Instead, the reduplicant pulls in the first consonant of the second syllable, to ensure that the reduplicant’s first syllable is heavy.

(13) Partial Reduplication in Mokilese

- | | |
|--------------------------------------|---|
| a. <i>soo.rək</i>
tear
‘tear’ | d. <i>ni.kid</i>
save
‘save’ |
| b. <i>pɔ.dok</i>
plant
‘plant’ | e. <i>soo-soo.rək</i>
RED -tear
‘tear’ |
| c. <i>ka.sɔ</i>
throw
‘throw’ | f. <i>pɔd-pɔ.dok</i>
RED -plant
‘planting’ |

⁶Tones are also copied onto the reduplicant, though they are not always an exact copy. A further discussion on the behavior of tones in Kinande reduplication is provided in Mutaka and Hyman (1990).

g. *kas-ka.sɔ*
RED-throw
 ‘throwing’

h. *nik-ni.kid*
RED-save
 ‘saving’

Reduplication can have a range of functions, many of which frequently re-occur in the world’s languages, such as meanings of contempt, smallness, lack of control, plurality, intensity, affection, continuity, and completion (Regier, 1998, p. 887; summarized in Lǐ and Ponsford, 2018, p. 58). Often, these functions express some form of “duplicative” meaning, suggesting that the semantics of reduplication are often iconic. Reduplication is often used to express plurality in nouns (e.g. the Australian language Warlpiri; Nash, 1980), repetition in verbs (e.g. the Bantu language Lusaamia; Marlo, 2002), and frequentatives in verbs (e.g. the Oceanic language Nadrogā; Geraghty, 2001) (summarized in Inkelas and Downing, 2015, p. 503).

Reduplication, then, is a process that is phonologically conditioned, and expresses some grammatical feature which is typically iconic. As such, reduplication is a process that sits at the interface between multiple linguistic subsystems. The form of the reduplicant is sensitive to phonology, the reduplicant copies the phonetic form of the base. Furthermore, in partial reduplication, the shape of the reduplicant may have to satisfy some prosodic requirement. This is seen in the Mokilese example, where the reduplicant must be a heavy syllable, and is therefore sensitive to syllabic weight. Reduplication is also used to express some *grammatical* feature, such as plurality in Warlpiri, and progressive aspect in Mokilese. The function of reduplication is often iconic, in that it is frequently used to express some type of duplicative meaning. These observations suggest that reduplication is a process that is sensitive to phonology and semantics, and is used to express some grammatical meaning.

1.3 Bantu Language Family

In this thesis, I focus on verbal reduplication of two Bantu languages *Zulu* and *Kerewe*. Bantu languages are spoken in the south of Nigeria, the Central African Republic, the Democratic Republic of Congo, Uganda, Kenya, and southern Somalia (Nurse & Philippson, 2003, p. 1). Bantu languages are agglutinating, with complex verbal morphology. The verb in Bantu languages encodes a rich amount of information, including negation, tense, aspect, subject (person/noun class), object (person/noun class), derivational extensions, mood, and more (Nurse & Philippson, 2003, p. 8). The verbal structure in most Bantu languages (including Zulu and Kerewe) is summarized in example 14 (Meeussen, 1967; Nurse and Philippson, 2003, p. 91).

(14) Verbal structure of most Bantu languages⁷

Initial - Subject - Negative - Tense/Aspect - Object - root - Extension(s) -
Final - Suffix

Inflectional morphology, such as tense and aspect, is expressed in the *tense/aspect* and *final* slots. The *extension(s)* slot is reserved for a small set of valency-changing categories, such as causative, applicative, reciprocal, and passive. This slot will be important for the upcoming analysis of reduplication in Kerewe presented in subsection 3.3, where we will see that the amount of material that can be copied from this slot onto the reduplicant appears to be sensitive to an underlying syntactic word structure. The analyses presented in chapters 2 and 3 also involve the expression of the *subjunctive* mood, which is held in the *suffix* slot. This slot also holds the “default” final vowel *-a*, which is associated with all tenses and aspects (Nurse & Philippson, 2003, p. 92).

Bantu languages exhibit reduplication in nouns, adjectives, pronouns, some demonstratives, and verbs (Nurse & Philippson, 2003, p. 88). Only verbs, however, reduplicate productively, in that verbal reduplication patterns can be extended to novel words. For nouns and adjectives, the other open classes, reduplication is lexicalized and therefore does not apply to all nouns and adjectives.

1.4 Research Questions and Outline

In this chapter, I have provided the necessary background for the upcoming analysis. I have outlined the main concepts of the framework Distributed Morphology, a theoretical linguistic overview of reduplication, and an introduction to Bantu languages.

I have shown that reduplication is a complex phenomenon, which is sensitive to a range of linguistic subsystems. The complexity of these patterns motivates many interesting questions concerning the mapping from internal syntactic structure to phonological form: how and when, for example, does the reduplicant receive its phonological form? With this in mind, this thesis aims to address the following research questions:

1. What patterns in reduplication processes may be influenced by syntactic structure?
2. Can these patterns be accounted for in a unified and systematic manner, across different languages?
3. What implications do these findings have for linguistic theory?

⁷Nurse and Philippson (2003, p. 91) includes a \neq symbol between the Object and root morphemes. It is unclear what this symbol means, and as far as I can tell, it does not influence the upcoming analysis. As such, I keep them as separate morphemes here.

In the following two chapters, I discuss the verbal reduplication process of the Bantu languages Zulu and Kereve. Chapter 4 summarizes these findings, and discusses them with respect to the theoretical linguistic literature. Chapter 5 builds on these findings, and discusses the accompanying Python code which automatically implements the analysis presented in chapters 2 and 3, and produces images of constituent trees for these reduplicated verb forms. I conclude in chapter 6, with some directions for future research.

CHAPTER 2

Zulu

In this section, I present an analysis of reduplication in the Bantu language Zulu, by expanding on the discussion presented in Cook (2018). In Zulu, verbs are reduplicated to convey a meaning of “doing a bad job of (the action)”. Reduplication in Zulu is partial, and the reduplicant morpheme is always bisyllabic (CVCV) (Hyman et al., 2008). Tones are also excluded from the reduplicant. These facts are most clearly observed when the root itself is bisyllabic, as shown in example 15.

(15) Reduplication of a Bisyllabic Root (Cook, 2018, p. 48)

a. *u-sébenz-a*
2SG-work-FV
‘you work’

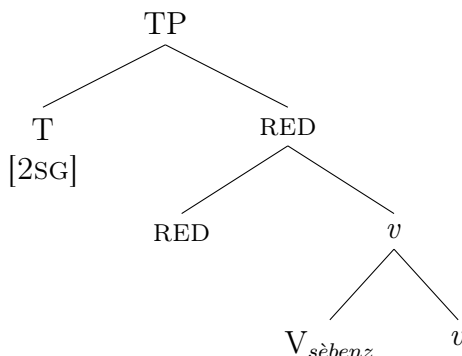
b. *u-sebe-sébenz-a*
2SG-**RED**-work-FV
‘you did a bad job of working’

Verbal reduplication exhibits different patterns, however, when the root is monosyllabic (CVC), and another distinct pattern when the root is vowel initial (VC). It is in these patterns that I find that reduplicated verbs exhibit word internal structure. Before presenting the analysis underpinning this observation, I first provide a step-by-step derivation of example 15b in section 2.1. The purpose of this section is to demonstrate how the tools in the Distributed Morphology framework can be utilized for reduplication. Though this particular example does not necessarily suggest an underlying structure, the analysis presented here lays the foundations for an understanding of the more complex Zulu data discussed in sections 2.2 and 2.3, and on the Kerewe data in chapter 3. It is here, I argue, that the morphophonology of reduplication is sensitive to syntactic word-internal constraints.

2.1 Reduplication of Bisyllabic Roots

As discussed in section 1.1, the upcoming analyses will assume two hypotheses: that there is “*syntactic hierarchical structure all the way down*”; and phonological material of this structure is “*inserted late*”. As such, we require a list of Vocabulary Items - which will specify the phonological content to be included in the structure - and an initial underlying structure. I suggest that example 15b contains an underlying structure as shown in figure 16, prior to any Vocabulary Insertion. Recall that all trees are depicted after head movement.

- (16) *Step 1: Pre-Vocabulary Insertion*
u-sebe-sèbenz-a “you do a bad job of working.”



The crucial information that this structure provides is the *position* of the heads, including both the linear order, and the hierarchical position of the phrase within the structure. Since our focus is on the interactions between (morpho)phonology and syntax, a discussion on the precise labels of the phrases and heads is beyond the scope of the thesis. Following Cook (2018), I assume that the subject prefix *u-* holds a position in the T slot. Additionally, I assume that the Final Vowel *-a* is a phonological realization of the verbalizing head *v*, per Cook (2013).

I have labeled the reduplicant as a RED head, and have placed it between the TP and *v*. It is important to acknowledge that reduplication in Zulu appears to convey some sort of *aspectual* meaning, in that doing a “bad job” of something could imply that the action was repeatedly attempted, but failed continuously. This is also in keeping with cross-linguistic studies on reduplication, which find that the function of reduplication is often used to express some form of aspectual meaning, as discussed in section 1.2. As such, it may be possible to suggest that the reduplicant is held within an *asp* head. However, a thorough semantic analysis would be required to confirm that this is the case for Zulu. I therefore leave the label of the reduplicant to be a RED.

Once the syntactic structure has been built, as shown in example 16, the morphemes can receive their phonological content. To do so, we can posit the following Vocabulary Items for each morpheme in example 17.

- (17) a. $T_{[2SG]} \longleftrightarrow u$
 b. $V_{work} \longleftrightarrow sèbenz$
 c. $v \longleftrightarrow a$

The Vocabulary Item for the RED head cannot be stated as a simple phonological string. As the reduplicant is bisyllabic copy of the phonological string of the morpheme to its right, the Vocabulary Item contains a bisyllabic template, as shown in example 18.

(18) Vocabulary Item for RED Morpheme (First Version)

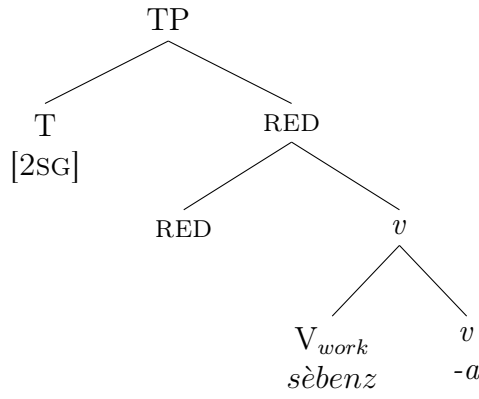
$\text{RED} \longleftrightarrow \sigma\sigma$

where $\sigma\sigma$ copies the first two syllables to the right of the RED morpheme.

Now that we have all the Vocabulary Items for every morpheme in the structure, we can begin the Vocabulary Insertion process. Assuming that VI is applied from the bottom up (following Bobaljik, 2000), we can assign the Vocabulary Items 17b and 17c to the morphemes in the lowest parts of the structure: the VP. This is shown in Figure 19.

(19) *Step 2: VI applied to VP and vP*

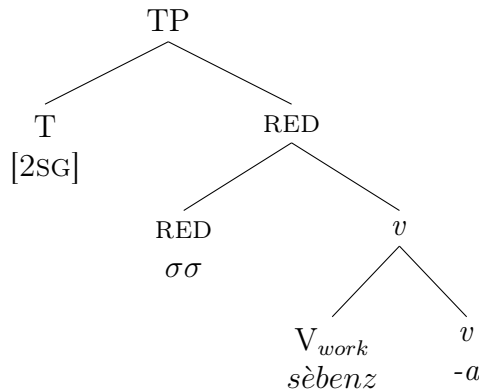
u-sebe-sébenz-a “you do a bad job of working.”



Moving up the structure, the next morpheme to receive its phonological content is the RED morpheme. Following the Vocabulary Item presented in example 18, the RED is assigned its bisyllabic template, as shown in figure 20.

(20) *Step 3: VI applied to REDP*

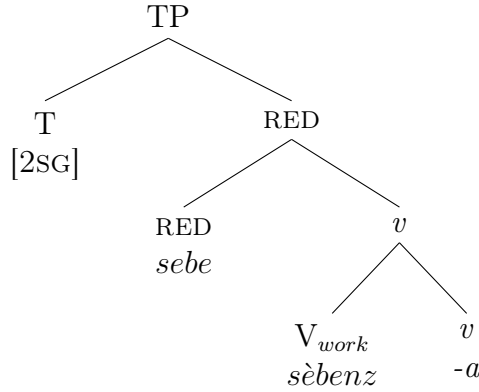
u-sebe-sébenz-a “you do a bad job of working.”



The Vocabulary Item in example 18 specifies that the morpheme receives its phonological content by copying two syllables to the right of it. Since the root has

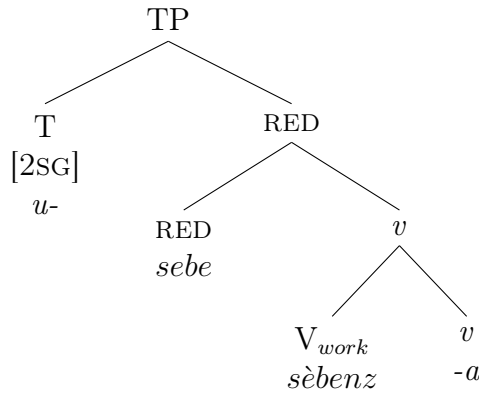
had Vocabulary Insertion applied to it, the reduplicant can copy the first syllables of the root V_{work} . This means the bisyllabic template $\sigma\sigma$ is filled with *sebe*, as shown in figure 21.

- (21) *Step 4: RED Bisyllabic Template Filled*
u-sebe-sèbenz-a “you do a bad job of working.”



Finally, the highest morpheme in the structure, the subject prefix, can be assigned its phonology, and the final structure is shown in figure 22.

- (22) *Step 5: VI applied to TP; Final Derivation.*
u-sebe-sèbenz-a “you do a bad job of working.”



In this section, we have seen how the reduplicant morpheme receives its phonological material by assuming a hierarchical structure within the word. Specifically, the Vocabulary Item of the reduplicant morpheme contains a bisyllabic template, which copies material from the morpheme to its right: the root. When the root itself is bisyllabic, as we have seen above, then this process is relatively straightforward. This, however, raises the question of what the reduplicant can copy if the root is monosyllabic. How can the grammar fill the bisyllabic template when there is not enough phonological material to copy? In the following section, I discuss how monosyllabic roots provide evidence of an internal syntactic structure, where the reduplicant contains phonological material that would otherwise be unexplained.

2.2 Reduplication of Monosyllabic Roots

Monosyllabic roots, with a consonantal onset, follow a similar pattern to that presented in section 2.1. When the Final Vowel is the default *a*, the reduplicant targets both the root and the FV morpheme, as shown in example 23.

(23) Monosyllabic CVC root with default *a* FV.

- | | |
|--|--|
| <p>a. <i>u-fúnd-a</i>
 2SG-study-FV
 ‘you study’</p> | <p>b. <i>u-funda-fúnd-a</i>
 2SG-RED-study-FV
 ‘you do a bad job of studying’</p> |
|--|--|

Given this data, we may expect that reduplication is a purely phonological process, as the reduplicant appears to blindly copy any phonological string to the right of it, until the bisyllabic requirement is satisfied. However, additional data shows that this is not always the case. Specifically, when the final vowel is not the default *a* phoneme, Zulu reduplication exhibits unusual behavior. One context in which the final vowel is pronounced as a different phoneme is in the *subjunctive* mood. In this case, the final vowel is expressed as *e*, as shown in example 24 below.

(24) Monosyllabic CVC root with subjunctive *e* FV.

- a. *ni-fúnd-e*
 2PL-study-FV.SBJ
 ‘you (PL) study (subjunctive)’

When this verb form is reduplicated, the reduplicant does not contain the final vowel *e*. Instead, the final vowel of the reduplicant is still the default final vowel *a*, as shown in example 25.

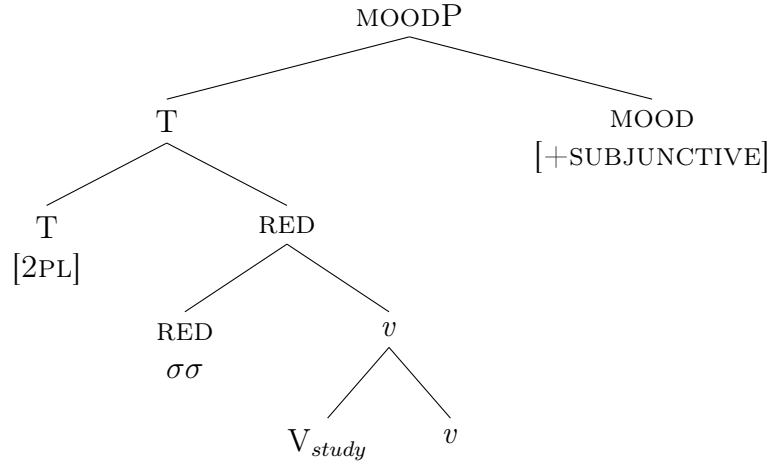
(25) Reduplicated verb with monosyllabic CVC root and subjunctive *e* FV.

- a. *ni-funda-fúnd-e*
 2PL-**RED**-study-FV.SBJ
 ‘you (pl) do a bad job of studying (subjunctive)’

This pattern suggests that Zulu reduplication cannot be explained by simply copying the phonological string: otherwise, the final vowel *e* would be copied in the reduplicant. This pattern can be adequately explained if we posit an underlying syntactic structure, where the reduplicant copies phonological material of morphemes that it c-commands.

The underlying word structure, which has been constructed in the syntactic part of the grammar, is presented in figure 26. This structure is similar to the structure for *u-sebe-sébenz-a* presented in figure 16, with the addition of a MOODP. The head of the MOODP specifies the subjunctive mood, and its phrasal projection dominates the T complex head. The placement of the MOODP above the T head is a reasonable position to take, given that the MOODP has similarities to a CP.

- (26) *Step 1: Prior to Vocabulary Insertion*
ni-funda-funde “you (pl) do a bad job of studying (subjunctive)”

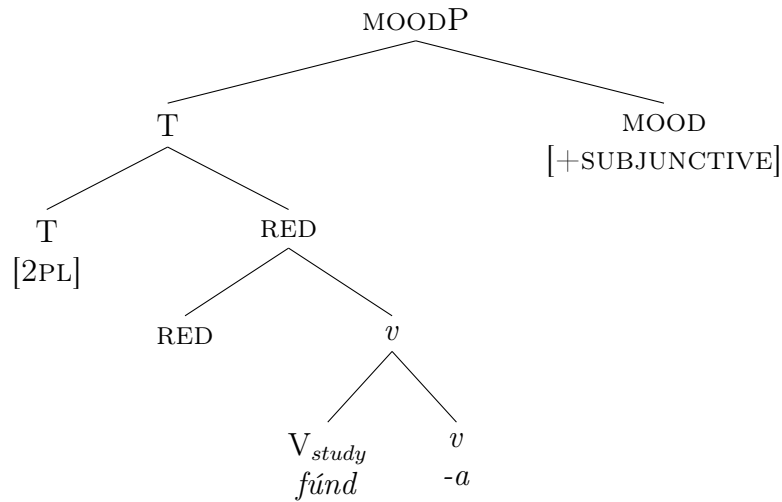


To assign phonology to the structure, we can posit the following Vocabulary Items for each head in the structure. The Vocabulary Items for the RED morpheme and default final vowel *v* are identical to those presented in examples 18 and 17c above, and repeated below for clarity.

- (27) a. $T_{[2PL]} \longleftrightarrow ni$
 b. $V_{study} \longleftrightarrow fúnd$
 c. $MOOD_{[+SUBJUNCTIVE]} \longleftrightarrow e$
 d. $v \longleftrightarrow a$
 e. $RED \longleftrightarrow \sigma\sigma$

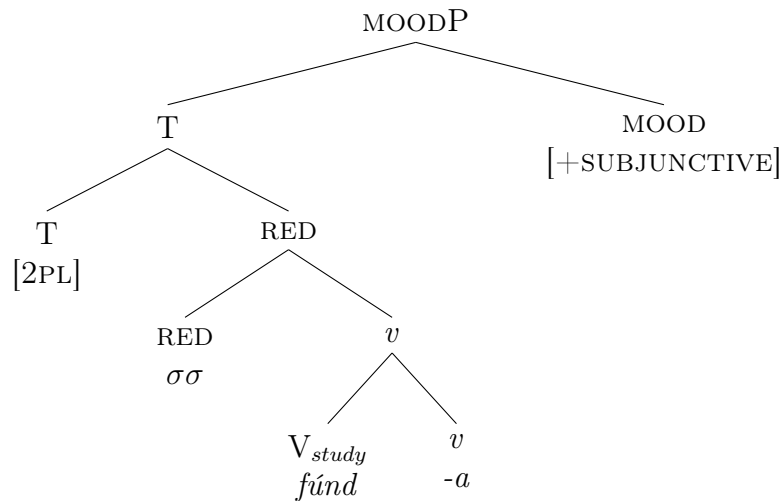
With these Vocabulary Items, we can assign the phonological content to each morpheme in the structure. Starting from the lowest point in the structure, the V and *v*, we can assign the phonological strings to those morphemes.

- (28) *Step 2: VI applied to vP*
ni-funda-funde “you (pl) do a bad job of studying (subjunctive)”



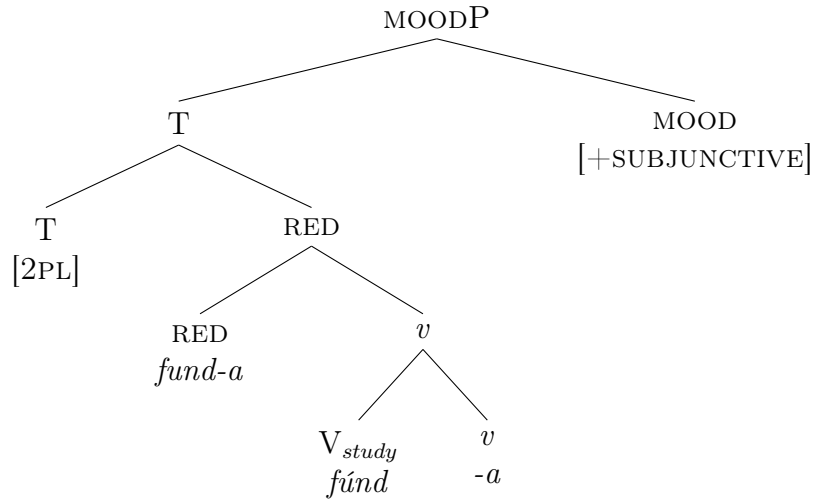
After the morphemes in the *v* are assigned their phonological material, the next morpheme, RED, is assigned its bisyllabic template.

- (29) *Step 3: VI Applied to REDP*
ni-funda-funde “you (pl) do a bad job of studying (subjunctive)”



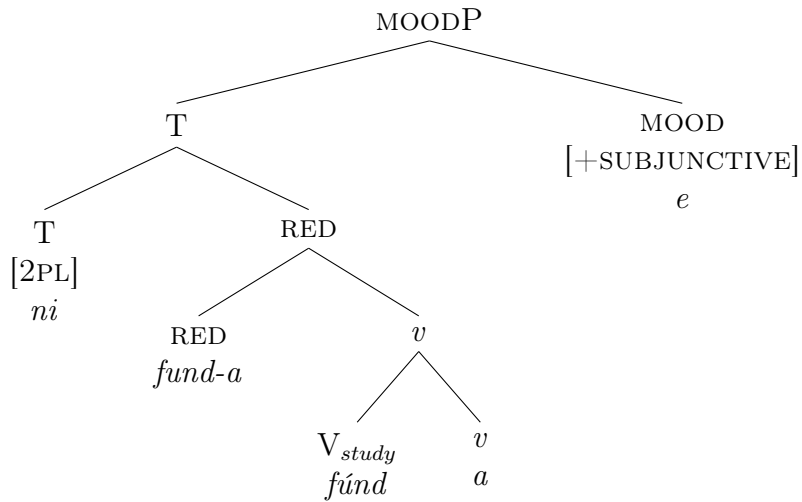
To fill the bisyllabic template, the RED morpheme copies the two syllables to the right of it. The root V_{study} and the default final vowel *v* have already been assigned their content, the reduplicant can copy this phonological material.

- (30) *Step 4: Bisyllabic Template Filled*
ni-funda-funde “you (pl) do a bad job of studying (subjunctive)”



Then, VI applies to the remaining morphemes, as shown in figure 31.

- (31) *Step 5: Final Vocabulary Items Applied*
ni-funda-funde “you (pl) do a bad job of studying (subjunctive)”

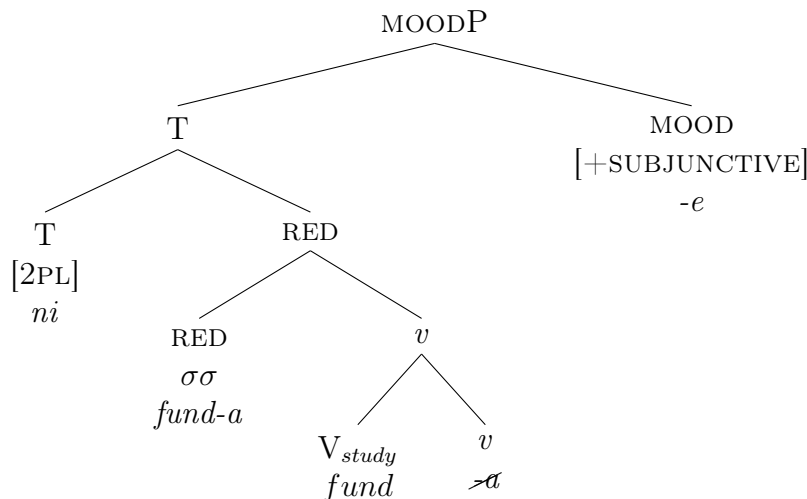


Once the structure has been assigned all its phonological content, additional phonological processes are applied to adhere to the overall phonological system of Zulu. Cook (2018, p. 50) states that the MOOD node is a phonological rule that converts the *v* node realized as [a] into [e]. However, I suggest that the VI rule for the MOOD node is $\text{MOOD}_{[+SUBJUNCTIVE]} \longleftrightarrow e$, and that the verbalizing suffix *a* is deleted by general phonological requirements. In Zulu, VV sequences are generally disallowed. Typically, /ae/ sequences are resolved by deleting the first vowel

(Posthumus, 2022, p. 17). It is, therefore, reasonable to suggest that the phonological material of the verbalizing suffix *a* is deleted once the MOOD node has been assigned phonological content. Doing so allows us to both delete the superfluous *a* in the *v* head, and maintain the default *a* vowel in the reduplicant.

The “phonological” rule that Cook (2018) proposes is a *readjustment* rule, which replaces all or part of a stem given some morphological context. Readjustment rules, however, have been received some criticism within the literature (Haugen & Siddiqi, 2016). Readjustment rules are typically idiosyncratic, and are an unrestricted tool that can overgeneralize word forms. This is in direct opposition to a core premise of generative grammar models: without constraints on what a *possible* language can be, we lose insight into which principles are universal, and which are language specific. By replacing the readjustment rule presented by Cook (2018) with a “typical” Vocabulary Item, the present analysis makes explicit the processes which are defined by syntax/morphology, and general phonological processes. As such, the final derivation of the Zulu verb is depicted in figure 32.

- (32) *Step 6: Vowel Hiatus Resolution*
ni-funda-funde “you (pl) do a bad job of studying (subjunctive)”



The occurrence of the final vowel /a/ on the reduplicant in Zulu has been a topic of discussion in the literature. In some analyses (e.g. Downing, 1997), the reduplicant is argued to be a verb stem, without any inflectional marking. In an Optimality Theory analysis, for example, there is a constraint that penalizes inflectional material on the reduplicant (Hyman et al., 2008). However, both of these analyses seem to be somewhat arbitrary, and do not capture the fact that the /a/ epenthesis vowel is identical to the verbalizing suffix. As such, the analysis presented here not only explains the descriptive facts of Zulu reduplication, but also provides a morphosyntactic explanation for the vowel mismatch in the subjunctive mood.

In this section, we have seen that the reduplicant morpheme copies phonological material which it c-commands. The phonological content of the base can then be

modified by general requirements of the phonological system. This analysis explains why there is an apparent mismatch between the base and reduplicant. The analyses up to now have involved the following hypotheses:

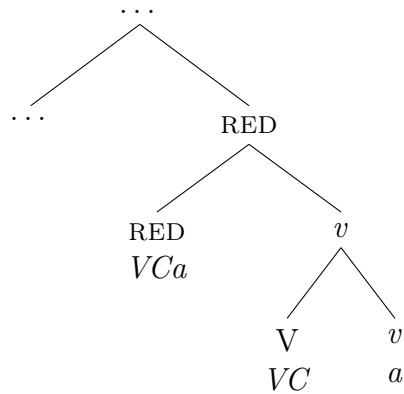
- There is a syntactic structure within the Zulu reduplicated verb.
- Phonological content of these morphemes are assigned in a systematic way.
- Additional phonological processes are applied once all the morphemes have been assigned their phonological content.

In doing so, we can coherently explain multiple aspects of the reduplication process. We can explain how the reduplicant is assigned its grammatical *function*, where reduplicant holds a position in the hierarchical structure. This position, situated between the T and V (post head movement), expresses some type of aspectual meaning - “doing a bad job” of the action. The analysis also explains how the *form* of the reduplicant is assigned its phonological material, where the reduplicant copies only material that has already been previously assigned. Additionally, this copying process is conducted before other, more general, phonological processes are applied. An important result from this analysis is the explanation for the vowel mismatch between the base and the reduplicant in the subjunctive mood.

2.3 Reduplication of Vowel Initial Roots

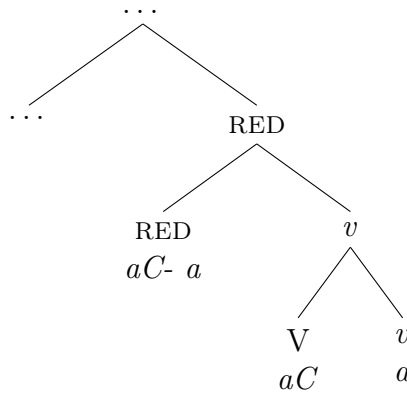
In the final section of this chapter, I discuss a class of verbal roots which, when reduplicated, create an illegal phonological environment. As mentioned above, the phonology of Zulu has a general dispreference for VV sequences. Additionally, we have seen that the reduplicant is required to be bisyllabic. As such, reduplicated verbs which contain a vowel initial and monosyllabic root is expected to produce an environment that is phonologically illegal in Zulu. That is, a vowel sequence is expected between the reduplicant and the base. This structure is schematized in figure 33.

(33) *Schematic Reduplicated Verb with VC Root*

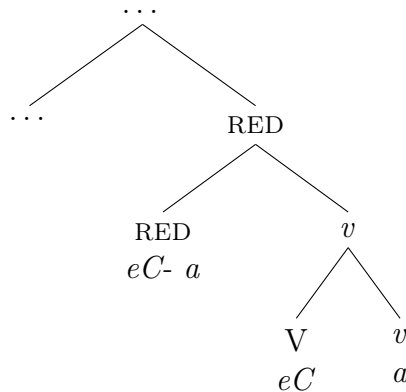


We can see that, when the verbal root has a VC prosodic structure, the phonological form of the reduplicant and base is *VCaVCa*. The only examples that Cook (2018) provides of monosyllabic verbal roots that are vowel initial begin with either an /e/ (*enz* “make”, *eb* “steal”) or an /a/ (*akh* “build”, *ang* “hug”). Since we expect that the default *v* morpheme is expressed as /a/, we would predict that the only VV sequences between the reduplicant and the base are /ae/ and /aa/. This is shown in the figures below.

(34) *Schematic Reduplicated Verb with /aC/ Root*



(35) *Schematic Reduplicated Verb with /eC/ Root*



Posthumus (2022, p. 16) states that VV sequences, where the first vowel is low, are resolved by either deletion, or coalescence. For /ae/ sequences specifically, these are resolved by deleting the first vowel /a/. If this were to apply to the reduplicated verb, then the final vowel /a/ in the reduplicant would be deleted. In this case, the reduplicant has the prosodic structure VC, and the bisyllabic requirement is unsatisfied.

A similar issue presents itself when attempting to resolve the /aa/ sequence. Generally, /aa/ sequences are realized as the short vowel /a/ (Posthumus, 2022, p. 22). This implies either the first /a/ vowel is deleted, or the second. Within the reduplicated verb, this means that either the final vowel in the reduplicant is deleted, or the initial vowel in the root is deleted. If the /a/ in the reduplicant is deleted, then the bisyllabic requirement of the reduplicant is unsatisfied. Alternatively, if the vowel in the root is deleted, then the surface form of the root would contain no vowel (other than the verbalizing suffix).⁸

There are two potential reasons that would prevent this from occurring. Firstly, from an Optimality Theoretic perspective, there may be an MAX_{Root} faithfulness constraint that requires every input of the root to appear in the output.⁹ Secondly, from a morphosyntactic perspective, there has been discussion within the literature that some phrases, such as the VP, “block off” additional phonological processes occurring within their complements. One example of this argumentation is Newell and Piggott (2014), who argue that VV sequences are allowed in Objibwe at the boundary of a *vP*, but not within it.¹⁰ For the present Zulu data, it would be reasonable to suggest that Zulu phonology does not allow the modification of the VP, and this is the motivation as to why additional processes are required for Zulu redu-

⁸In Zulu, it is possible to have verbal roots which do not contain a vowel. This means that vowel-less verbal roots is not disallowed by the general phonological system.

⁹See Trommer (2001) for an endeavor to integrate Distributed Morphology and Optimality Theory, as a means to investigate the interface between phonology and syntax.

¹⁰In Distributed Morphology terminology, VV sequences are permitted within a *phase*.

plication with vowel initial roots.¹¹ From both a phonological and morphosyntactic perspective, there are compelling reasons to avoid the modification of roots.

Regardless of which analysis we take, all these predictions result in unlikely conclusions. Specifically, they lead to environments that the general phonology of Zulu disallows. Given these facts, it should come as no surprise that reduplicated verbs with a monosyllabic root indeed behave differently to what was described above. There are two possible reduplicated verb forms for monosyllabic vowel initial roots, shown in example 36 below.

(36) VC roots and reduplicated forms

- a. Non-reduplicated form (where the subject prefix /si/ is realized as /s/ when it occurs before a vowel)

si-akh-a → *s-akh-a*

1PL-build-FV

“we build”
- b. VC Root with glide *y* epenthesis.

s-akha-y+akh-a

1PL-**RED**-y+build-FV

“we do a bad job of building”
- c. VC Root with Prefix Material Copied

sakha+s-akh-a

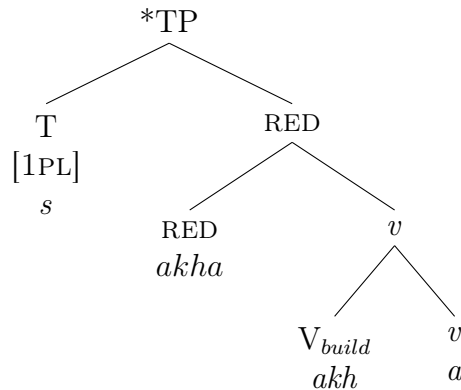
RED+1PL-build-FV

“we do a bad job of building”

To illustrate the issue of reduplication with monosyllabic and vowel initial roots, figure 37 illustrates how the reduplication of *s-akh-a* would be constructed under typical reduplication processes. The reduplicant copies its phonological material from the material that RED c-commands: the V head *akh* and verbalizing suffix *a*.

¹¹Note that modification of the *v*P must be permitted in Zulu, as the *v* head /a/ can be deleted, as we have seen above.

- (37) **s-akha-akha*: ‘we do a bad job of building’, epenthesis *y* included



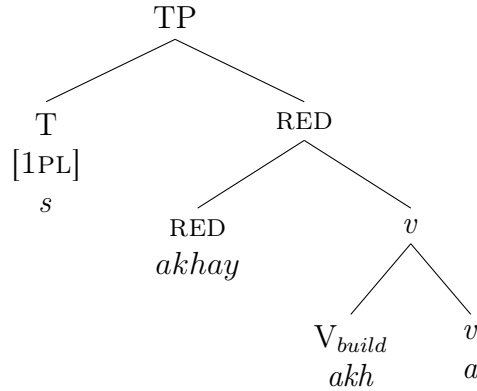
Examples 36b and 36c demonstrate how these illegal sequences, created by typical reduplication processes, are resolved. In example 36b, the RED morpheme targets the root and final vowel - the *v* head - as we have seen in sections 2.1 and 2.2. However, unlike before, the glide *y* is inserted between the reduplicant and base. Here I also depart from Cook (2018, p. 57), who includes the *y* glide as a part of the base (*s-akha+y-akh-a*). I instead include the glide as a part of the reduplicant, to make clear that the verb head does not undergo additional phonological processes, as discussed above. An extra context specific Vocabulary Item can be posited for this situation, as shown in example 38.

- (38) Additional RED Vocabulary Item for Vowel Initial Roots
 RED \longleftrightarrow $\sigma\sigma$ y / $_\text{V}$

To formalize this analysis, I posit the following Vocabulary Items in example 39 for each morpheme, where the Vocabulary Item for the default *a* is identical to the one provided in example 17c. I have also chosen to represent the Vocabulary Item for the subject marker as a context specific rule. The structure of example 36b is provided in figure 40 below. As the derivation of this structure is identical to the previous examples, I present the final derivation of *s-akha-y-akh-a* with all nodes already assigned their Vocabulary Items.

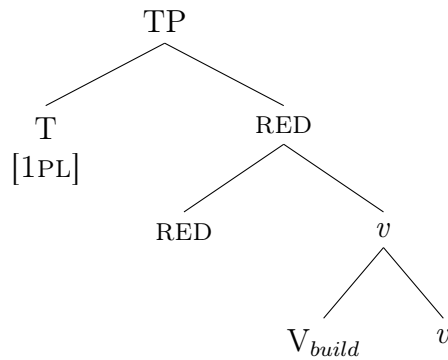
- (39) a. $T_{[1PL]} \longleftrightarrow s / _ V$
 b. $V_{build} \longleftrightarrow akh$

- (40) *s-akhay-akh-a*: “we do a bad job of building”, epenthesis *y* included



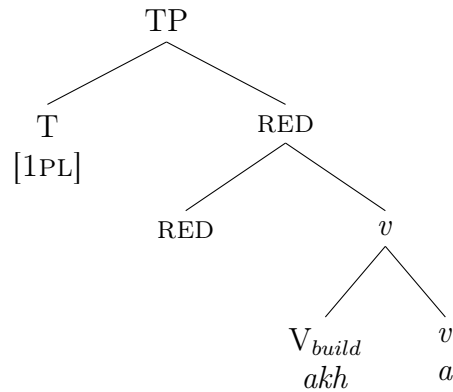
The verb with a monosyllabic and vowel initial root also can be reduplicated as shown in example 36c. Here, the RED morpheme includes the subject prefix, suggesting that the RED morpheme is placed in a higher structural position than what we have seen up to now. The idea that morphemes move to a different position in order to satisfy language-specific phonological requirements has been discussed in the previous literature (e.g. Caha, 2010; Baunaz and Lander, 2018); Wiland, 2019, and may be the motivation for this variant form. To illustrate this idea, I will now step through the analysis of example 36c, based on the Vocabulary Items in examples 38 and 39. To begin, the underlying structure of the reduplicated form is generated. As the structure does not have access to any phonological material yet, the underlying structure is the “default” form, where the reduplicant is situated between the TP and *v*.

- (41) *sakha-s-akh-a*: “we do a bad job of building”
Step 1: Underlying Structure



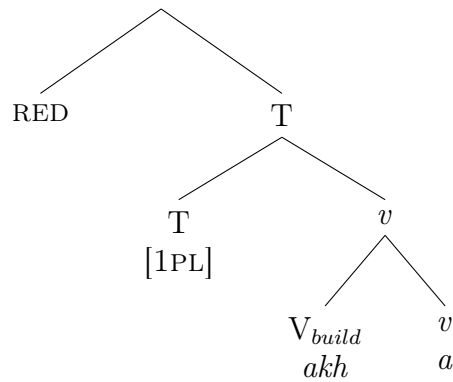
With this underlying base structure, the Vocabulary Insertion process can begin. The morphemes at the lowest point in the structure, *V_{akh}* and *a*, can be assigned their Vocabulary Items.

- (42) *sakha-s-akh-a*: “we do a bad job of building”
 Step 2: V_{akh} and a assigned their Vocabulary Items



At this stage, the Vocabulary Item for the RED attempts assignment to its terminal node. However, this would create an illegal phonological environment, and typical phonological processes cannot resolve it. The RED morpheme therefore moves to a position above the T.¹²

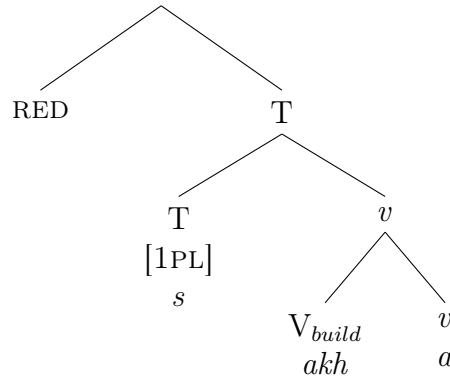
- (43) *sakha-s-akh-a*: “we do a bad job of building”
 Step 3: RED moves to a different position



With this new underlying structure, the Vocabulary Insertion process continues. The $T_{[1PL]}$ prefix is assigned its Vocabulary Item, as shown in figure 44.

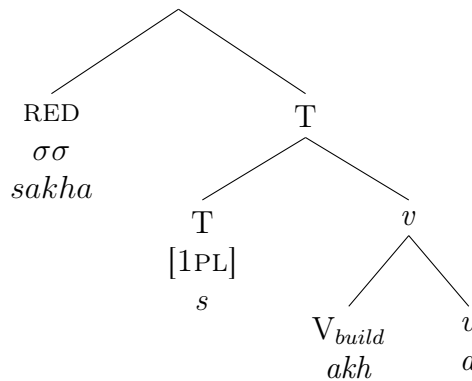
¹²I have left the root of this structure unlabelled, as it is not clear whether this should still be a TP, or REDP, without further investigation into word formation head movement.

- (44) *sakha-s-akh-a*: “we do a bad job of building”
 Step 4: T is assigned its Vocabulary Item



Finally, the reduplicant receives its phonological form. To save space, I have collapsed two steps into one, in that the bisyllabic template has already been filled with its phonological material.

- (45) *sakha-s-akh-a*: “we do a bad job of building”
 Step 5: RED is assigned its Vocabulary Item



This example has shown that the reduplicant morpheme can be placed in a different position to its “default” position under certain conditions. Specifically, if the default position between the T and *v* produces an illegal phonological sequence, then the RED morpheme moves to a higher structural position to resolve this issue.

2.4 Summary

In this section, I have explored how the phonological form of Zulu reduplicated verbs can be understood by positing a word-internal syntactic structure. I have suggested that the reduplicant morpheme copies phonological material that has been

previously assigned from morphemes that are in a lower position of the structure to the reduplicant. I have argued that this analysis can explain two aspects of Zulu reduplication: in cases where there is a mismatch between the reduplicant and the base; and where there are various reduplication forms in vowel initial roots. Based on the discussion presented in this chapter, I will now present a similar analysis from data on another Bantu language, Kerewe.

CHAPTER 3

Kerewe

Kerewe (also known as Kikerewe and Kerebe, among others) is spoken in the Ukerewe Islands of Lake Victoria, Tanzania (Odden, 1996, p. 111).¹³ Like Zulu, Kerewe is a Bantu language which exhibits reduplication. In this language, verbal reduplication is used to express that the action was completed “here and there”. Reduplicated verbs in Kerewe have two variants, which express the same semantics and apply to all verbs regardless of their prosodic structure: full copy and asymmetrical reduplication. In the full copy variant, the reduplicant copies all phonological material that follows it, as shown in example 46.¹⁴

- (46) Hyman (2009, p. 185)
- a. Non-reduplicated form
ku-lim-il-an-a
 INF-cultivate-APPL-RECIP-FV
 “to cultivate for each other”
 - b. Full copy
ku-lim-il-an-a-lim-il-an-a
 INF-**RED**-cultivate-APPL-RECIP-FV
 “to cultivate for each other here and there”

In the second variant, asymmetrical reduplication, the reduplicant copies the root, and optionally other derivational suffixes. This is shown in example 47, where suffixes that are omitted from the reduplicant are crossed out.

- (47) Hyman (2009, p. 185)
- a. Non-reduplicated form
ku-lim-il-an-a
 INF-cultivate-APPL-RECIP-FV
 “to cultivate for each other”
 - b. Truncated reciprocal *-an*
ku-lim-il-~~an~~-a-lim-il-an-a
 INF-**RED**-cultivate-APPL-RECIP-FV

¹³The data reported here has been taken from Hyman (2009) and Odden (1996).

¹⁴In both reduplication variants, tones are excluded on the reduplicant.

- c. Truncated reciprocal *-an* and applicative *-il*

ku-lim-il-an-a-lim-il-an-a

INF-**RED**-cultivate-APPL-RECIP-FV

Additionally, the final vowel on the reduplicant is always the default final vowel *-a*, even when the base does not contain the default final vowel. Even in the subjunctive, where the final vowel is expressed as *-e*, the final vowel of the reduplicant is *-a* and not *-e*, as shown in example 48.

- (48) Odden (1996, p. 136)

- a. Non-reduplicated form

noo-habúúl-é

3PL-advise-SUBJN

“we should advise”

- b. Asymmetrical reduplication

noo-habuul-a-habúúl-é

3PL-**RED**-advise-SUBJN

“we should advise here and there”

In this chapter, I argue that the patterns in the asymmetrical variant can provide evidence of hierarchical structure within words. I do so by investigating environments in which additional phonological processes apply, where these processes are applied only after the reduplicant receives its phonological material. Additionally, I argue that the forms presented in example 47 can also be explained by placing the REDP node at different places within the hierarchical structure. Before beginning this discussion, however, I outline an issue with the placement of the final vowel *-a* on the reduplicant.

3.1 A Puzzle: The Final Vowel on The Reduplicant

This reduplication pattern in example 48 is reminiscent of the Zulu data presented in section 2.2. There, I argued that Zulu verbs contain a verbalizing suffix *-a*, which is deleted when the following suffix is vowel initial, such as in the subjunctive mood *-e*. The reduplication process, however, applies *before* the verbalizing suffix *v* is deleted, which allows the reduplicant to contain a final vowel *-a*. Though it is tempting to analyze the Kerewe data in example 54 in a similar manner, the general phonological requirements of Kerewe do not point to the same analysis.

In Kerewe, VV sequences are typically resolved by coalescence, and compensatory lengthening of the surviving vowel (Odden, 1995, p. 91). For example, the palatal glide *y* optionally deletes when it occurs in stem initial position. The deletion of the *y* in this position may create an environment with a VV sequence. If a VV sequence occurs, and the first vowel is /a/, then the second vowels merge into one non-high vowel. If the second vowel is /i/, then the resulting vowel is /e/. This is shown

in example 49, where the glide is retained in example 49a, and is deleted (and the vowels are subsequently merged) in example 49b.

(49) VV Hiatus Resolution in “they steal”

- a. ba-yíbá
- b. béébá

Another example of asymmetrical reduplication is in the perfective, as shown in example 50.

(50) Asymmetrical Reduplication with Perfective Suffix

- a. Perfective Form
a-báník-ílé
 3SG-roast-PERF
 “he roasted”
- b. Reduplicated Perfective Form
a-baník-a-báník-ílé
 3SG-**RED**-roast-PERF
 “he roasted here and there”

Based on the general phonological requirements outlined in Odden (1995), we would expect the surface form of *a-baník-a-baník-a-ílé* - where there is an underlying *-a* after the verbal root - to be realized as **a-baník-a-baník-élé*. The surface of the perfective suffix, however, is unaltered. It is therefore not so simple to suggest that there is a final vowel *-a* which is deleted in the presence of a vowel initial suffix.

It is important to note, however, that this observation does not necessarily disprove the idea that the Kerewe verbs contain a vowel final suffix *a* like Zulu. The examples of VV resolution provided in Odden (1995), where the first vowel is /a/, always involves a prefix that contains a consonantal onset. It would be unsurprising if phonological processes perform differently when the /a/ vowel in question contains no onset. For example, the onsetless subject prefixes *o* and *e* undergo unexpected phonological processes, when compared to subject prefixes that contain an onset. This may suggest that onsetless morphemes (such as the default vowel *-a*) are subject to unexpected phonological processes, within the general requirements of Kerewe phonology. As such, the placement of the final vowel in Kerewe is a nontrivial issue that requires additional research.

Though the behavior of the final vowel on the reduplicant remains a mystery for now, its precise workings do not distract from the insights obtained from the upcoming analysis. It is sufficient for our purposes to posit that the Vocabulary Item for the Kerewe RED morpheme is defined as shown in example 51. Though somewhat unsatisfying, the rule accurately describes the facts, and can be used to illustrate other aspects of the reduplication process that are clear to understand.

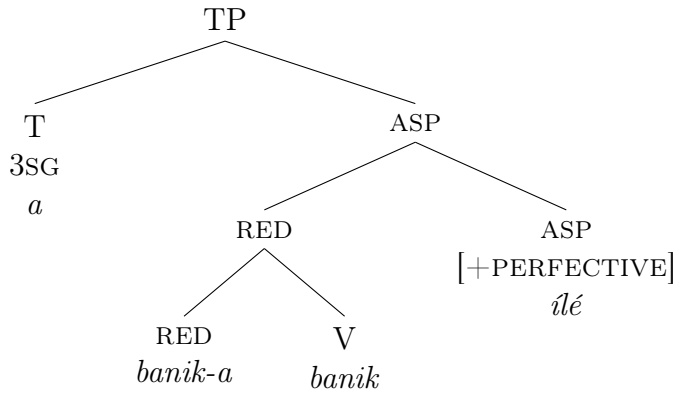
(51) Vocabulary Item for Kerewe Reduplicant

RED $\longleftrightarrow \gamma a$

where γ represents all phonological material that the reduplicant c-commands.

With this in mind, we can posit the underlying structure for the asymmetrical variant of example 54 in figure 52. As discussed in section 2.1, the focus of this analysis is not on its semantics, so I am not concerned with the labels within the tree. For simplicity, I roughly follow the same labelling conventions as presented in the Zulu data. The main difference is that I do not posit *v* layer, as there is no reason to suggest a verbalizing *v* suffix.

- (52) *Final derivation*
a-banik-a-banik-ilé “he roasted here and there”



Now that I have acknowledged the issue of the reduplicant’s final vowel, I present data in which there is a mismatch between the base and reduplicant.

3.2 Mismatch between the Reduplicant and Base

In this section, I provide examples where the reduplicant and the base exhibit different surface forms due to additional phonological processes. To analyze the data, I argue that positing a hierarchical structure can explain why there is a mismatch between these forms. In Kerewe, the perfective tense is expressed with the suffix *-ilé* in the “default vowel” slot (Odden, 1996, p. 139).

- (53) Perfective *-ilé* Suffix

a. Infinitive Form	b. Perfective Form
<i>ku-báník-a</i>	<i>a-báník-ilé</i>
INF-roast-FV	3SG-roast-PERF
“to roast”	“he roasted”

As discussed above, in asymmetrical reduplication, the right edge of the reduplicant is always *-a*, even when the base contains the perfective suffix *-ilé*. This example is repeated below.

(54) Asymmetrical Reduplication with Perfective Suffix

- | | |
|---|--|
| <p>a. Perfective Form
 <i>a-báník-ílé</i>
 3SG-roast-PERF
 “he roasted”</p> | <p>b. Reduplicated Perfective Form
 <i>a-baník-a-báník-ílé</i>
 3SG-RED-roast-PERF
 “he roasted here and there”</p> |
|---|--|

The perfective suffix also triggers a morphophonemic rule. Oral coronals that precede the suffix undergo anticipatory spirantization: $t \rightarrow s$; $d, l \rightarrow z$.

(55) Spirantization rule $l \rightarrow z$

- | | |
|--|---|
| <p>a. Infinitive Form
 <i>ku-ful-a</i>
 INF-clean-FV
 “to clean”</p> | <p>b. Perfective Form
 <i>a-fuz-ílé</i>
 3SG-clean-PERF
 “he cleaned”</p> |
|--|---|

(56) Spirantization rule $t \rightarrow s$

- | | |
|---|---|
| <p>a. Infinitive Form
 <i>ku-buut-a</i>
 INF-choke-FV
 “to choke”</p> | <p>b. Perfective Form
 <i>a-buus-ílé</i>
 3SG-choke-PERF
 “he choked”</p> |
|---|---|

(57) Spirantization rule $d \rightarrow z$

- | | |
|--|---|
| <p>a. Infinitive Form
 <i>ku-geend-a</i>
 INF-go-FV
 “to go”</p> | <p>b. Perfective Form
 <i>a-geenz-ílé</i>
 3SG-go-PERF
 “he went”</p> |
|--|---|

When these verbs are reduplicated, however, the spirantization rule targets only the base. The surface form of the reduplicant is identical to the underlying form of the root, with the addition of the default vowel *-a*.

(58) Rule $l \rightarrow z$ Does Not Target the Reduplicant

- | | |
|---|---|
| <p>a. Perfective Form
 <i>a-fuz-ílé</i>
 3SG-clean-PERF
 “he cleaned”</p> | <p>b. Reduplicated Perfective Form
 <i>a-ful-a-fuz-ílé</i>
 3SG-RED-clean-PERF
 “he cleaned here and there”</p> |
|---|---|

(59) Rule $t \rightarrow s$ Does Not Target the Reduplicant

a. Perfective Form

a-buus-îlé

3SG-choke-PERF

“he choked”

b. Perfective Form

a-buut-a-buus-îlé

3SG-RED-choke-PERF

“he choked here and there”

(60) Rule $d \rightarrow z$ Does Not Target the Reduplicant

a. Perfective Form

a-geenz-îlé

3SG-go-PERF

“he went”

b. Perfective Form

a-geend-a-geenz-îlé

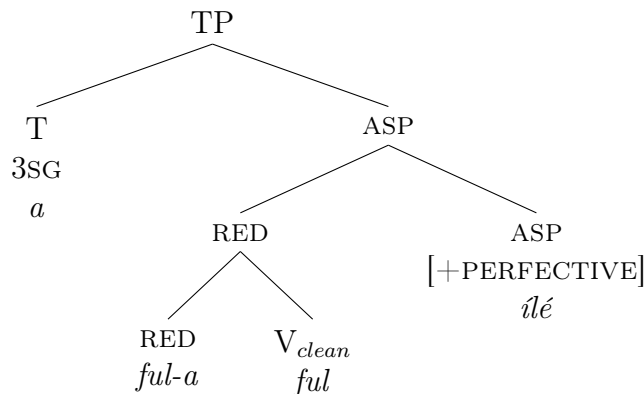
3SG-RED-go-PERF

“he went here and there”

These data can be accurately understood by positing a hierarchical structure, in which the reduplicant copies phonological material before more general phonological processes are applied. An underlying form for the non-reduplicated variant, after all Vocabulary Items have been assigned, in example 61 is shown below. Recall that the Vocabulary Item of the reduplicant states it copies all phonological material that the morpheme c-commands, with the default final vowel *-a*.

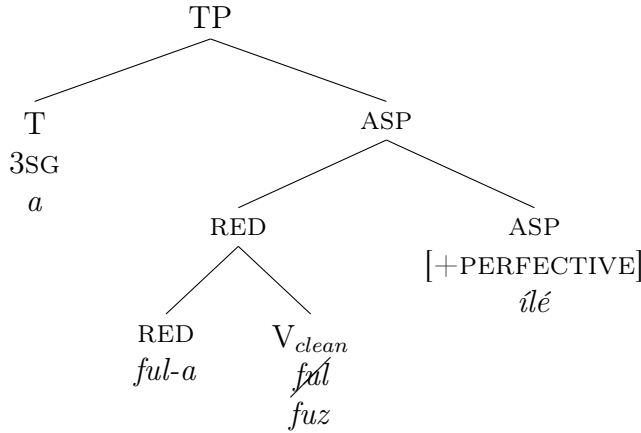
(61) *Underlying Form*

a-ful-a-fuz-îlé “he cleaned here and there”



Once all morphemes have been assigned their Vocabulary Items, general phonological processes apply. In this case, the spirantization rule is applied to the root *ful*, resulting in the final derivation as shown in figure 62.

- (62) *Final Derivation*
a-ful-a-fuz-îlé “he cleaned here and there”



This analysis predicts the following data, which includes an additional phonological rule. In Kerewe, /nl/ sequences are disallowed, and the /l/ phoneme is realized as /d/ (Odden, 1996, p. 130). An environment with this sequence can be created by using the 1SG subject marker *n* with a stem initial /l/. This is shown in example 63, where the stem initial /l/ is realized as /d/. Additionally, the perfective morphophonemic rule is triggered, where the stem final /l/ is realized as /z/.

- (63) /l/ assimilates to /d/, following /n/ (plus spirantization rule $d \rightarrow z$)

- a. Infinitive Form

ku-lol-a

INF-see-FV

“to see”

- b. Perfective Form

n-doz-îlé

1SG-see-PERF

“I saw”

In its reduplicated form, we can see an even greater mismatch between the reduplicant and the base, in which the subject marker /n/ affects only the initial segment on the reduplicant, and not the base.

- (64) Reduplicated form of *n-doz-îlé* “I saw”

- a. Perfective Form

n-doz-îlé

1SG-see-PERF

“I saw”

- b. Reduplicated Perfective Form

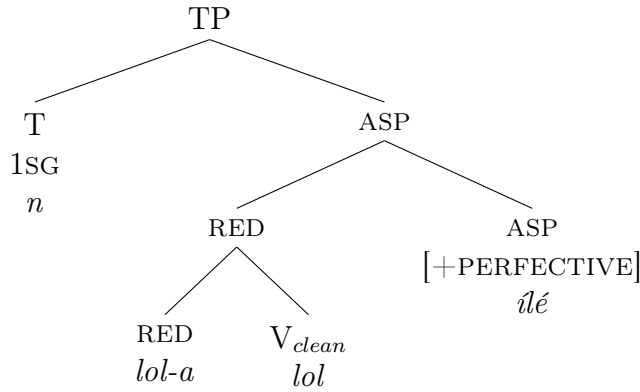
n-dol-a-loz-îlé

1SG-**RED**-see-PERF

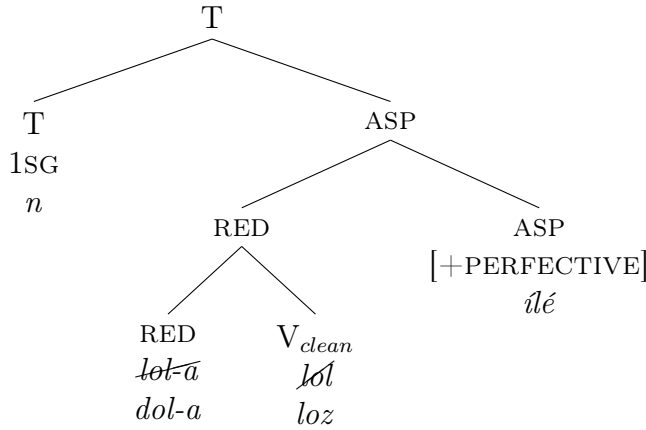
“I saw here and there”

The derivation of example 64b is shown in figures 65 and 66 below.

- (65) *Underlying Form*
n-dol-a-loz-ilé “he cleaned here and there”



- (66) *Final Derivation*
n-dol-a-loz-ilé “he cleaned here and there”



By positing a hierarchical structure in this way, we can observe that the reduplicant receives its phonological form from morphemes that it c-commands: the root. Then, additional phonological processes apply, which generates a mismatch between the reduplicant and base.

In the following section, I look at a different set of data, which does not involve general phonological processes. Instead, I argue that different reduplication variants can be generated by placing the REDP in different positions within the hierarchy. This again suggests that the patterns of reduplication in Kerewe can be understood with a hierarchical syntactic structure.

3.3 Partial Reduplication

Asymmetrical reduplication in Kerewe can be partial. As in full reduplication, the reduplicant targets the verbal stem, that is, the root and any derivational suffixes.

In partial reduplication, however, the reduplicant may truncate (i.e. omit) some derivational suffixes. This was shown in example 47, and is repeated below. The reduplicant may omit the reciprocal suffix - as shown in example 68 - or, the reduplicant omits both the applicative and reciprocal suffixes - as shown in example 69. Finally, it is ungrammatical for the reduplicant to omit the applicative suffix while also including the reciprocal suffix. Note, again, that the reduplicant in the asymmetrical reduplication variant ends with the final vowel *-a*.

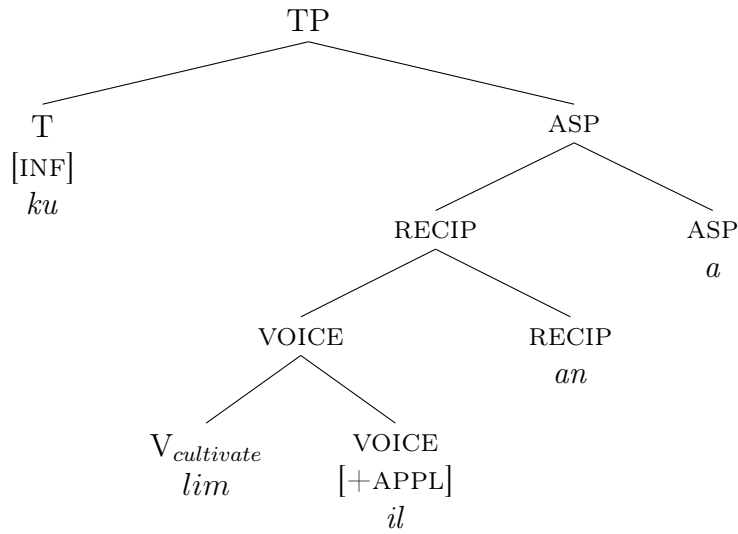
- (67) Full reduplication *-an* Hyman (2009, p. 185)
 a. *ku-lim-il-an-a-lim-il-an-a*
 INF-**RED**-cultivate-APPL-RECIP-FV
- (68) Truncated reciprocal *-an* Hyman (2009, p. 185)
 a. *ku-lim-il-~~an~~-a-lim-il-an-a*
 INF-**RED**-cultivate-APPL-RECIP-FV
- (69) Truncated reciprocal *-an* and applicative *-il* Hyman (2009, p. 185)
 a. *ku-lim-il-~~an~~-a-lim-il-an-a*
 INF-**RED**-cultivate-APPL-RECIP-FV
- (70) Truncated applicative *-il* only is ungrammatical
 a. **ku-lim-~~il~~-an-a-lim-il-an-a*
 INF-**RED**-cultivate-APPL-RECIP-FV

A list of Vocabulary Items for these words is presented in example 71. Downing (1999, p. 10) states that the *ku* prefix is a prefix that marks for the infinitive, and I therefore also assume that the *ku* prefix is a T head, with the feature [INFINITIVE]. I also assume that the default final vowel *-a* indicates “default” aspect, and is therefore the head of an aspect head. The Vocabulary Item for the reduplicant has already been presented in example 51.

- (71) a. $T_{[INF]} \longleftrightarrow ku$
 b. $V_{cultivate} \longleftrightarrow lim$
 c. $VOICE_{[+APPL]} \longleftrightarrow il$
 d. $RECIP \longleftrightarrow an$
 e. $ASP \longleftrightarrow a$

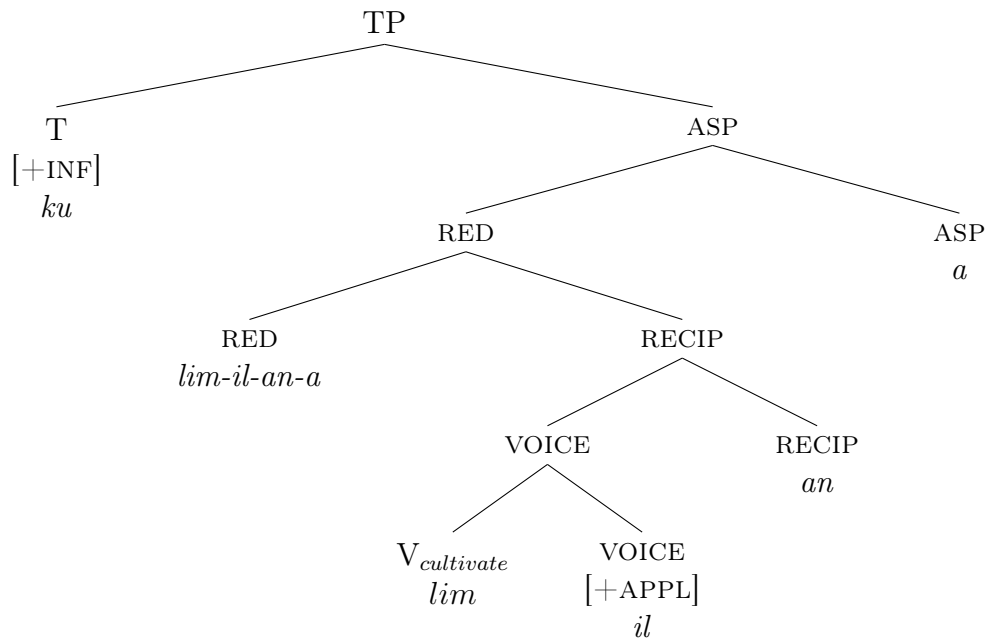
The underlying structure of the non-reduplicated form is shown in figure 72. I will not go through a step-by-step analysis of the VI process, as it is identical to that was shown above.

(72) *kulimilana* “to cultivate”



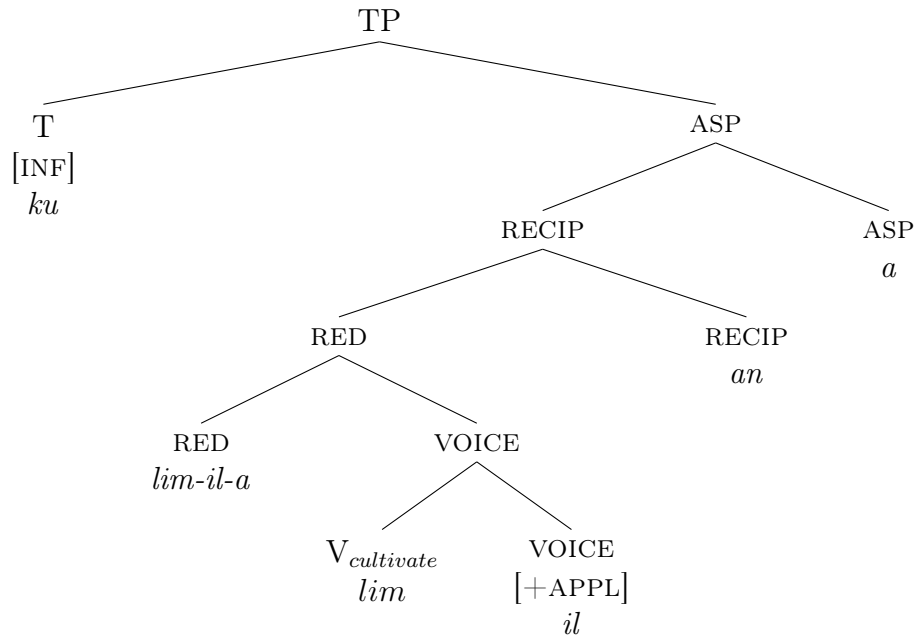
To generate the reduplication variants, the RED can be placed above the RECIP, the VOICE, or the V, to generate the three reduplicated variant forms. For example, the reduplicated form of the verb that copies both derivational suffixes is shown in figure 73, where the REDP dominates the whole RECIP. Again, I do not present each step one-by-one, as the overall approach is identical to the Zulu examples.

(73) *ku-lim-il-an-a-lim-il-an-a*: full reduplication



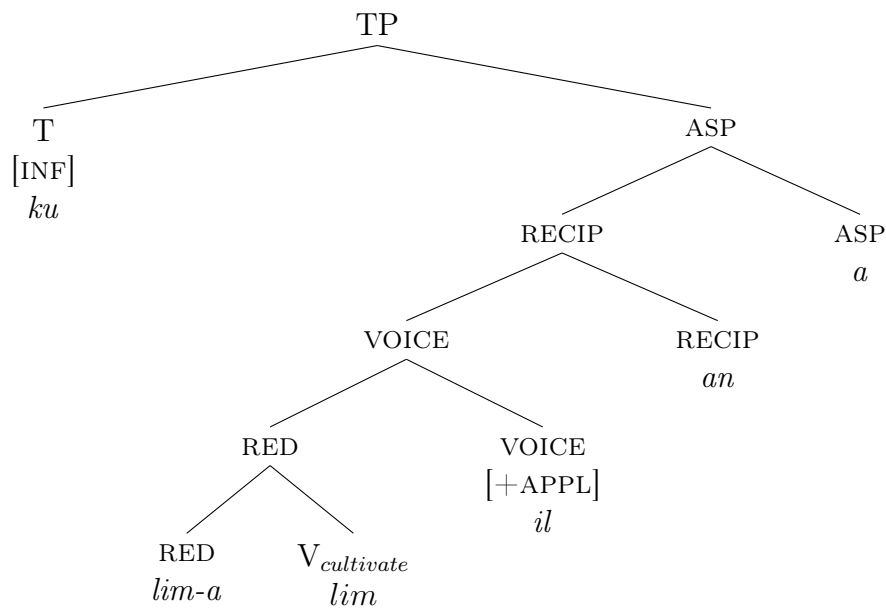
The reduplicated form which omits the reciprocal form is shown in figure 74. The RED dominates the VOICE, such that the reduplicant cannot copy material from the RECIP head.

(74) *ku-lim-il-a-lim-il-an-a*: truncated reciprocal suffix



Alternatively, the RED can dominate V, in which case the reduplicant copies material only from the root.

(75) *ku-lim-a-lim-il-an-a*: truncated reciprocal and applicative suffix



By allowing the RED morpheme to attach to various positions in the structure, we can accurately model the various forms of the reduplicated verb. Furthermore, this analysis predicts the *ungrammaticality* of example 70, repeated below.

- (76) Truncated applicative *-il* only is ungrammatical

**ku-lim-~~il~~-an-a-lim-il-an-a*

INF-RED-cultivate-APPL-RECIP-FV

It is not possible to place the RED morpheme into the non-reduplicated structure, shown in figure 72, such that its head c-commands only the root and the reciprocal suffix *-an*. The analysis we have seen thus far, then, offers an explanation as to why example 70 is ungrammatical: that the phonological material of the reduplicant is not c-commanded by its morpheme. This fact underscores the hypothesis that the morphophonology of words is influenced by their internal syntactic structure. Therefore, the analysis not only accurately models the morphophonology of Kerewe reduplicated verbs, but also offers an explanation into why certain word forms are ungrammatical.

3.4 Summary

In this chapter, I have presented data from two areas of the verbal reduplication process in Kerewe. Firstly, I showed that general phonological processes are applied to the reduplicated verb *after* the reduplicant morpheme has copied its phonological material. I then showed that the variation in partial reduplication patterns can be explained by placing the RED head in various positions in the hierarchy. Together, both sets of data can be accurately modeled and explained by positing a hierarchical structure within the reduplicated verb. Though the analysis of the final vowel *-a* on the reduplicant still requires additional work, I argued that this does not detract from the main analysis presented here.

CHAPTER 4

Theoretical Implications

In this thesis, I have sought to investigate interactions within the morphosyntactic interface, and the ways in which syntactic structure may influence phonological form. To do so, I focussed on reduplication patterns of verbs in two Bantu languages, Zulu and Kerewe, and argued that they can be understood by positing a hierarchical syntactic structure within words. The results presented in chapters 2 and 3 therefore address research question 1, which asks whether there are patterns in reduplication processes that may be influenced by an internal syntactic structure. The present chapter aims to provide a unified and systematic explanation for these results, in order to address research question 2. I then contextualize these findings within the literature, to discuss their implications for linguistic theory, addressing research question 3.

We have seen that in both Zulu and Kerewe, the reduplicant morpheme copies phonological material from morphemes that it c-commands. These analyses not only model the observed data accurately, but also validate the assumption within Distributed Morphology that phonological material is assigned from the bottom up, as has been argued by Bobaljik (2000). Since the reduplicant is placed in a higher syntactic position than the base, the phonological material of the base must already be assigned in order for the reduplicant morpheme to copy it. The tools provided by Distributed Morphology can therefore accurately explain patterns in reduplication, offering evidence that syntactic constraints can influence the morphophonology of reduplication.

I have also presented data where there is a mismatch between the reduplicant and the base, suggesting that reduplication in these languages is not blindly copying material that follows the reduplicant. These patterns can be understood by proposing that general phonological processes are applied only *after* morphemes have been assigned their Vocabulary Items. For Zulu, I suggested that the verbalizing suffix *-a* is adjacent to the verbal root, and is deleted in the presence of the vowel expressing the subjunctive mood *-e*. Importantly, this deletion occurs only after the reduplicant has copied its phonological material. The same logic is applied to the Kerewe data, with the morphophonemic spirantization rule triggered by the perfective suffix *ílé*. In reduplicated verbs, this rule is applied only *after* the reduplicant morpheme has copied its phonological content from the root. Consequently, spirantization applies only to the base, and not to the reduplicant. Due to the timing of these processes - deletion in Zulu, and spirantization in Kerewe - there is a mismatch between the

reduplicant and the base. This corroborates the idea that general phonological processes are applied only after the Vocabulary Insertion process has been conducted, in both Zulu and Kerewe.

Finally, Zulu and Kerewe can exhibit various reduplicant forms. I proposed that variants in reduplicated verbs can be explained by placing the RED in different positions of the structure. Zulu reduplication typically targets only the root and not the subject prefix. The exception to this is when the root is monosyllabic and vowel initial. I argued that this creates an environment with a VV sequence, and that the typical phonological processes to resolve this sequence result in illegal phonological environments. To deal with this issue, the RED morpheme is placed above the subject prefix and copies phonological content from this morpheme. Illegal phonological environments are therefore avoided, and the surface word form adheres to the general requirements of Zulu phonology. Partial reduplication in Kerewe can also produce a number of variant forms. I suggested that these forms can be analyzed by placing the RED in different positions in the structure. This analysis also predicts the ungrammaticality of reduplicated words, in which the applicative morpheme is truncated and not the reciprocal morpheme.

These findings are summarized in table 1, where the final row generalizes the patterns found in Zulu and Kerewe. By hypothesizing that reduplicated words have internal syntactic structure, we have a unified account to explain the *form* of the reduplicant in both languages investigated here.

	Material Copied in Reduplicant	Phonological Processes	Reduplication Variants
Zulu	Bisyllabic template	Verbalizing suffix /a/ deleted after subjunctive /e/	Monosyllabic VC roots have two reduplicant variants
Kerewe	Verbal stem, where derivational suffixes can be optionally omitted	Perfective /ile/ triggering spirantization rule	Partial reduplication can truncate derivational suffixes
Generalization	Reduplicant copies material that the RED morphemes c-commands	Phonological processes are applied after Vocabulary Insertion	Position of RED morpheme can be variable

Table 1. Summary of language specific and generalized reduplication patterns

The findings presented in this thesis suggest that the morphophonology of reduplication can be understood by assuming an internal syntactic structure. Following this assumption challenges the validity of the Lexicalist Hypothesis (typically attributed to Chomsky, 1970): that the system that assembles complex words is separate from the system that assembles syntactic phrases. This hypothesis has been influential within linguistic theory, and underpins several linguistic theoreti-

cal frameworks (e.g. Head Driven Phrase Structure Grammar, Müller et al., 2021; Lexical Functional Grammar, Bresnan, 2015; discussed in Bruening, 2018).

By assuming only one formation process to generate phrases and words, as has been presented in this thesis, we have a more streamlined model to explain the diversity of the world's languages. A simpler model, which contains fewer formation processes, is a desirable feature in order to unravel the fundamental nature of language. By positing fewer processes in a model, and making explicit which processes are language universal and language-specific, we can explore the extent to which linguistic diversity may or may not be constrained by universal linguistic principles. On the other hand, separating processes for word formation and phrase formation may obfuscate similarities between languages, thereby losing insight into the underlying nature of language. Although I do not claim that the findings presented in this thesis are definitive proof of a word internal structure, I have shown that assuming a word internal structure allows for a comprehensive analysis of the data. This, then, offers an explanation for otherwise puzzling observations in verbal reduplication in Zulu and Kerewe.

CHAPTER 5

Computational Implementation

In chapters 2 and 3, I provided a theoretical analysis of verbal reduplication within the Distributed Morphology framework. In this chapter, I outline an accompanying Python implementation of this analysis, `parse_redup.py`. The program can be accessed on Github.¹⁵ The Python program generates step-by-step diagrams of the Vocabulary Insertion process, in which the reduplicant morpheme copies only phonological material that has been previously assigned. This program aims to formalize my findings from the aforementioned discussion. The program generates an underlying, hierarchical structure for words, and utilizes this structure to produce correct surface word forms. The accuracy of the code verifies the theoretical analyses presented here, by generating the correct surface form of reduplicated words.

The core of the program is designed to be extensible, and the language universal properties are hard coded into the program. Language-specific requirements, such as reduplication scope and additional phonological processes, are specified in the input files. These processes are dealt with accordingly in the relevant helper functions. I have provided input files based on the analysis for Zulu and Kerewe.

For the purposes of this program, the output diagrams do not show head movement, and therefore include their phrasal projections. All hierarchical positions are identical.

5.1 Usage

To run the program, two arguments are required. The input folder, which contains the user input files, is specified with `-i`. The output folder, where the output .svg files are stored, is specified with `-o`. An example of its usage is shown in Listing 5.1. The help documentation can be called with the `-h` flag.

```
python3 parse_redup.py -i zulu_input -o zulu_output
```

Listing 5.1 Example usage of `parse_redup.py`.

¹⁵https://github.com/romiHill/reduplication_in_dm, accessed September 12th 2023.

5.2 Input Files

The program requires five .txt files, which are located in the input folder specified with `-i`. These files specify language specific processes, such as vocabulary insertion rules, and reduplication specific patterns.

1. `vi_rules.txt`

Vocabulary Insertion rules are specified in this file, which are coded as a three column file separated by commas. The first column contains the head, the second column contains any features of the head, and the final column contains the phonological string. All terminal nodes must be included in this file (even if the phonological string is empty). All nodes must be grouped together (i.e. all “T” morphemes in one section). A small sample of the text file is outlined in Listing 5.2. Note that there is no Vocabulary Item for the reduplicant, as this is generated automatically from the program.

```
T,2sg,u
V,,akh
V,,sebenz
v,,a
MOOD,subjunctive,e
```

Listing 5.2 Example VI rules for Zulu

2. `psr.txt`

This file contains a list of phrase structure rules, which is used to generate the underlying structure of words. The first row specifies the root node, and all remaining rows contain the mother and its daughters. Phrase structure rules must be unary or binary branching, and phrases cannot contain new line characters. As mentioned above, the output diagrams are depicted with their phrasal level projections. This is because mother nodes must contain unique names, to distinguish between terminal nodes and non-terminal nodes. For simplicity, I have chosen to label non-terminal nodes with “P”, but any labelling convention is possible.

```
MOODP
MOODP,TP,MOOD
TP,T,vP
vP,VP,v
VP,V
```

Listing 5.3 Phrase structure rules for Zulu

3. red.txt

This text file is a three column .csv file that contains all nodes that the redP can dominate. The first column is the phrase that the redP can dominate. The second column specifies the phonological environment that the redP needs, in order to dominate the phrase. Currently, the code deals only for environments when the following morpheme is vowel initial. This condition is specified with **VOWEL**. If the reduplicant morpheme can attach to any phonological environment, then this column is empty. The third column specifies any epenthesis phonemes that appear after the reduplicant. A sample of this text file is provided in Listing 5.4, which states that the redP dominates the vP in any environment, and dominates the TP only when the following morpheme is vowel initial. Additionally, there is an epenthesis glide *y* when there is a vowel-vowel sequence.

```
vP , ,  
TP , VOWEL , y
```

Listing 5.4 Reduplication conditions for Zulu.

4. scope.txt

`scope.txt` is a single line file that specifies the prosodic template of the reduplicant. That is, the amount of phonological material to be copied. Currently, the code works for ‘bisyllabic’ templates (CVCV). If the reduplicant copies all material that it c-commands, then this file is empty.

```
bisyllabic
```

Listing 5.5 Scope of the reduplicant in Zulu.

5. phono_rules.txt

Language specific phonological rules are saved in a three column .csv file, separated by commas. The first column states the string of phonemes to be changed, the second column states the start of the string that is changed, and the third column states the end of the string is not changed.

```
ae , , e  
ua , w , a  
ia , , a  
ie , , e  
ue , w , e
```

Listing 5.6 General phonological processes in Zulu.

The program on provided on Github includes three input folders. `zulu_input/` generates all non-reduplicated and reduplicated words for the roots *akh*, *fund*, and

sebenz, in the subjunctive and indicative moods, and subject prefixes for 1SG, 2SG, 1PL, and 2PL.

I decided to separate two datasets for Kerewe, one that illustrates the mismatch between the base and the reduplicant, and the other that demonstrates the behavior of partial reduplication. I did so as it is not clear whether all verbal roots can combine with the derivational suffixes presented in section 3.3, based on the data provided in Odden (1996) and Hyman (2009). `kerewe_mismatch/` generates word forms for the roots *ful* and *lol*, in the perfective aspect, with 3SG and 1SG subject prefixes. `kerewe_partial/` generates all partial reduplication variants for the root *lim* with the reciprocal, and applicative derivational suffixes.

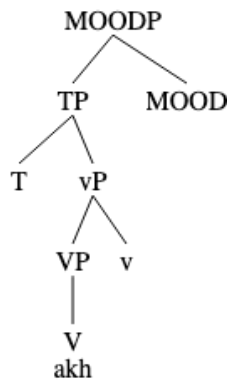
5.3 Output of the Program

The program generates a series of SVG (Scalable Vector Graphics) files, which contain diagrams of syntax trees at every step within the analysis. The program also produces a single column .txt file of all words that were produced by the program.¹⁶ This output file separates non-reduplicated and reduplicated words with the string `---reduplicated words ---`. The output figures for the word *wakhe* ‘I build (subjunctive)’ and its reduplication variants *wakhayakhe* and *wakhawakhe* are shown below in figures 77, 78, and 79. There are a total of 383 output files for Zulu and Kerewe, which are available to access on Github.¹⁷ Finally, a list of all final words is provided in Appendix A.

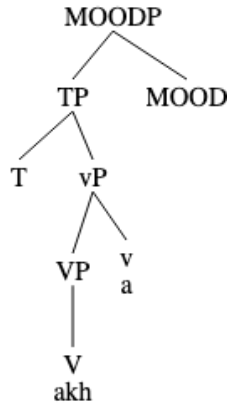
¹⁶In order to include these graphics into Latex, I converted the .svg files to .png. I did so by writing a bash script that utilizes the `svgconvert` library.

¹⁷https://github.com/romiHill/reduplication_in_dm/tree/master/output_kerewe_mismatch for Kerewe data where there is a mismatch between the reduplicant and the base
https://github.com/romiHill/reduplication_in_dm/tree/master/output_kerewe_partial for Kerewe data for partial reduplication
https://github.com/romiHill/reduplication_in_dm/tree/master/output_zulu for Zulu data.
 Accessed September 12th 2023.

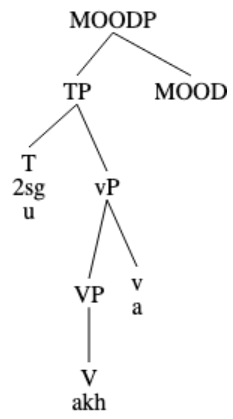
(77) Step by step output for *wakhe*



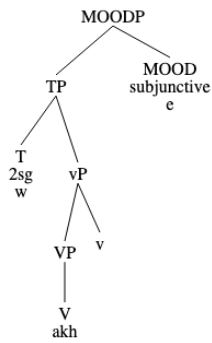
(a) Step 1



(b) Step 2

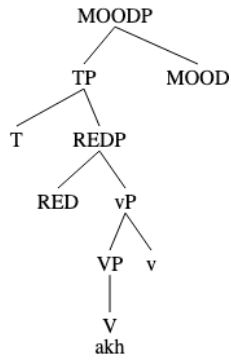


(c) Step 3

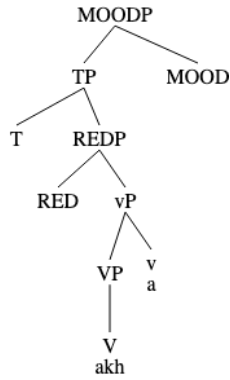


(d) Step 4

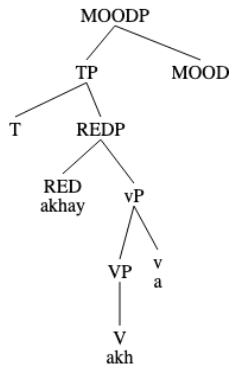
(78) Step by step output for *wakhayakhe*



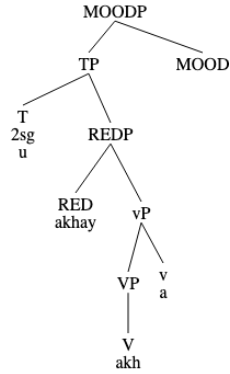
(a) Step 1



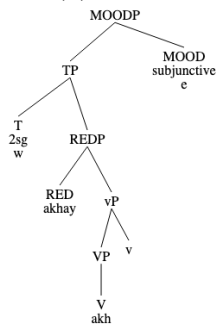
(b) Step 2



(c) Step 3

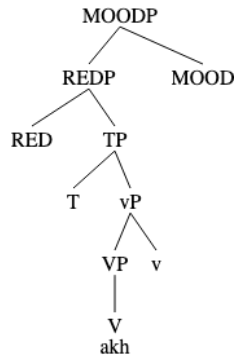


(d) Step 4

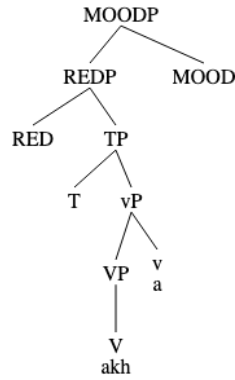


(e) Step 5

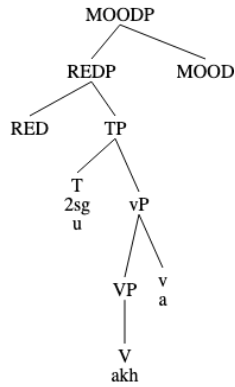
(79) Step by step output for *wakhawakhe*



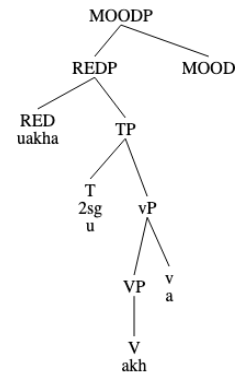
(a) Step 1



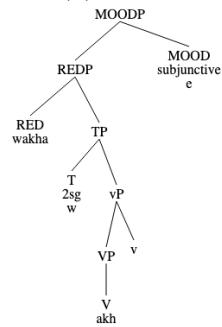
(b) Step 2



(c) Step 3



(d) Step 4



(e) Step 5

5.4 Structure of Program

The purpose of the program is to demonstrate that the reduplicant receives its phonological material from morphemes that are situated in a lower position of the hierarchical structure. This is shown by the fact those morphemes have been assigned phonological material, which is available to copy. The program aims to conduct this analysis automatically.

The hierarchical structure of words are coded by nested lists, where each list can contain one or two elements. The element can either be a string, which represents a morpheme, or a list, which represents a phrase. In order to traverse and manipulate the arbitrarily embedded nested lists, the program uses a variety of recursive helper functions. An outline of the program is displayed in figure 80.

5.5 Read in Files

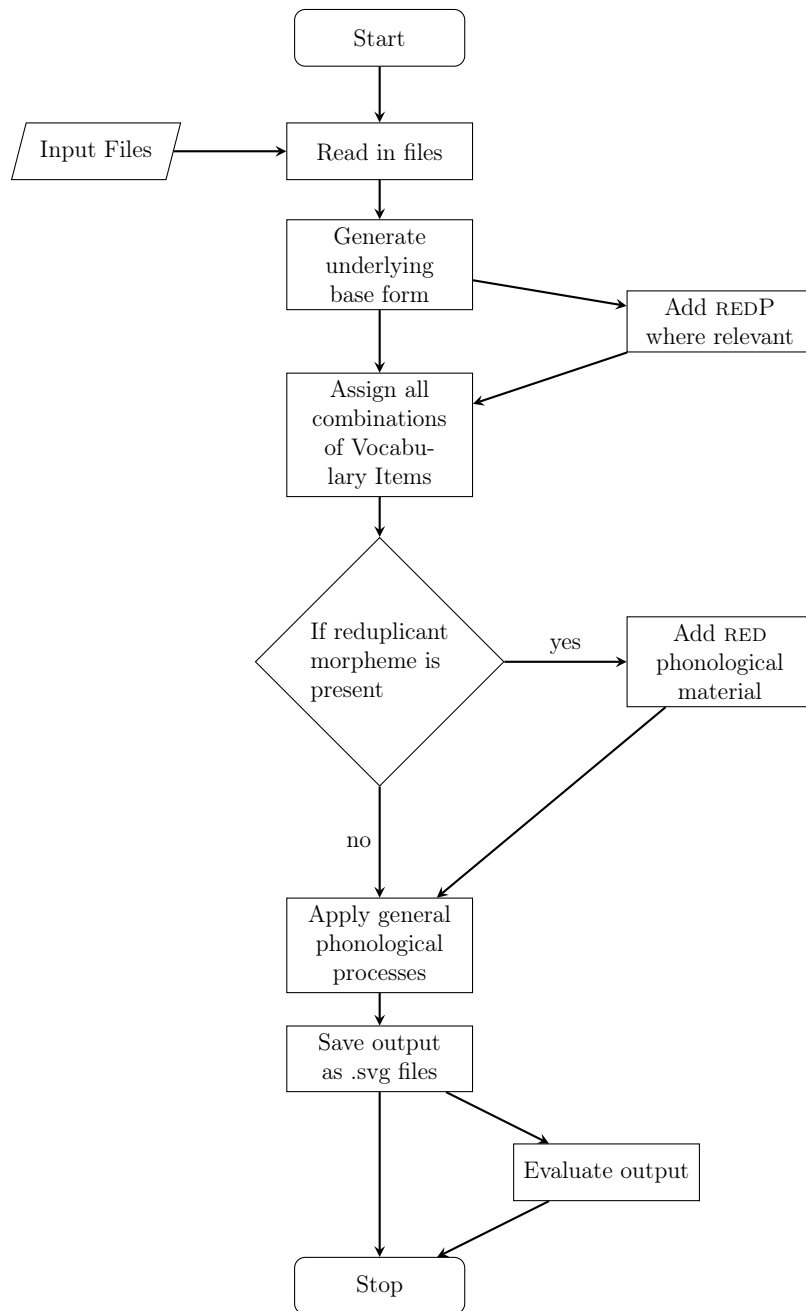
5.5.1 Phrase Structure Rules

Phrase structure rules are stored as a dictionary, where the key represents the mother node and the value represents the daughter nodes. The value is a two element or one element list. If the element in the list is a string, then that element represents a terminal node. If the element is a list, then that element represents another phrase. Additionally, that element must also be a key in the dictionary, which contains a terminal node.

```
{ 'MOODP': [['TP'], 'MOOD'], 'TP': ['T', ['vP']], 'vP': [['VP'], 'v'], 'VP': ['V'] }
```

Listing 5.7 Phrase structure rules as stored in parse_redup.py

(80) Outline of Python Implementation



5.5.2 VI rules

VI rules are stored as a list of dictionaries, where each dictionary represents a possible combination of VI rules.

```
{{'T': ['2sg', 'u'], 'V': ['', 'akh'], 'v': ['', 'a'], 'MOOD': ['subjunctive', 'e']},
 {'T': ['2sg', 'u'], 'V': ['', 'sebenz'], 'v': ['', 'a'], 'MOOD': ['subjunctive', 'e']}}
```

Listing 5.8 VI rules as stored in `parse_redup.py`

5.5.3 Phonological Rules

Phonological rules are saved as dictionaries, where the key represents illegal environments and the value represents the changed sequence of phonemes. The value is a two element list, where the first element contains the first half of the new string, and the second element contains the second half of the new string.

```
{'ae': ['', 'e'], 'ua': ['w', 'a'], 'ia': ['', 'a']}
```

Listing 5.9 Phonological rules as stored in `parse_redup.py`

5.6 Generate Underlying Base Structure

The code first generates the underlying base structure, before any phonological or morpho-semantic feature assignment, using the information provided in `psr.txt`. As mentioned in subsection 5.5.1, phrase structure rules are saved as a dictionary, where daughters that are phrases are stored as a list. In order to generate the base structure, the code progressively unpacks lists until there are no more embedded lists with a single element. For example, using the `psr.txt` outlined above, the rules would be expanded as such:

```
['MOODP', ['TP'], 'MOOD']
['MOODP', ['TP', 'T', ['vP']], 'MOOD']
['MOODP', ['TP', 'T', ['vP', ['VP'], 'v']], 'MOOD']
['MOODP', ['TP', 'T', ['vP', ['VP', 'V'], 'v']], 'MOOD']
```

Listing 5.10 Progressively expanding PSR.

The code achieves this by locating a list with a single element, and inserting its daughters after it. Since this list is arbitrarily embedded within the outer list, I use a helper function that uses depth first search to identify this list.


```

def find_embedded_index(input_list, elem):
    """
    Find the index of an element in an embedded list using
    depth first search
    """
    for i in range(len(input_list)):
        if isinstance(input_list[i], list):
            # if current element is a list, call the function
            # again
            result = find_embedded_index(input_list[i], elem)
            # return the result (index of elem within its
            # index) and i (the index of current list)
            if result:
                return [i] + result
            # return the index when the element is found
        elif input_list[i] == elem:
            return [i]
    # if the element doesn't exist return false
    print(f'{elem} doesn\'t exist in {input_list}, please fix
    ')
    sys.exit(1)

```

Listing 5.11 Find the index of an element in an embedded list

Once the index of the unexpanded phrase is found, I use a helper function `add_branches_at_rule()` to add the daughters at the relevant position. If the daughters are added at the end of the list, then the final index for slicing the outer list must be empty.

```

def add_branches_at_rule(inner, embedded_index,
    branches, rule):
    """
    Unpack rule at given index
    Rule is binary
    """
    # end of list requires different slicing
    if embedded_index[SECOND_LAST_INDEX] == len(inner) -
        1:
        inner[embedded_index[SECOND_LAST_INDEX]:] = [[
            rule, branches[INITIAL_INDEX], branches[
                SECOND_INDEX]]]
    else:
        inner[embedded_index[SECOND_LAST_INDEX]:
            LAST_INDEX] = [[rule, branches[INITIAL_INDEX],
                branches[SECOND_INDEX]]]

```

```
return inner
```

Listing 5.12 Add branching rule to base structure

In the end, the example underlying base form will be stored as a list, as shown in below.

```
[ 'MOODP', [ 'TP', 'T', [ 'vP', [ 'VP', 'V'], 'v' ] ], 'MOOD' ]
```

Listing 5.13 Final underlying base form

5.7 Apply VI Rules to Underlying Structure

Once the underlying syntactic structure has been generated, the VI process can be applied to the structure. To ensure that the VI process starts from the deepest list to the most surface level list, the code first marks each level of the structure with the symbol *, where the number of *s is equal to the height of the structure. Based on this, the code then assigns Vocabulary Items from the fewest number of *s to the most.

5.7.1 Mark Depth of Structure

The code first finds the deepest level of the structure, using the recursive helper function `find_deepest_depth()`. Then, the recursive helper function `mark_depth_for_vi()` marks all terminal nodes with the inverse depth that the node is currently positioned in. The base case of the recursive function is when the type of the element is a string. In this situation, if the element is a terminal node (that is, a head in a Vocabulary Item), or is RED, then the function returns that string appended with the number of *s equal to the inverse of the current depth. The recursive case is when the element is a list, and recursively calls on itself for every element in the list.

```
def mark_depth_for_vi(data, vi_rules, depth):
    """
    Insert same of *s at each depth of the tree
    To mark where Vocabulary Items are inserted
    In preparation for VI from bottom up
    """
    # Check if the element is a list
    if isinstance(data, list):
        result = []
        for item in data:
            result.append(mark_depth_for_vi(item,
                                             vi_rules, depth - 1))
```

```

        return result

    elif isinstance(data, str):
        if data in vi_rules or data == "RED":
            return data + '*' * (depth)
        else:
            return data
    else:
        return data

```

An example of the output is presented below.

```

['MOODP', ['TP', 'T****', ['REDP', 'RED***', ['vP',
    ['VP', 'V*'], 'v**']]], 'MOOD*****']

```

Listing 5.14 Marked syntactic data

5.7.2 Apply VI Rules to Structure

VI rules are applied to the structure from the bottom up. Since the structure has already been marked for depth, the helper function `replace_stars_with_vi()` inserts Vocabulary Items, starting at the deepest level. This helper function is also recursive. There are two base cases in this function. The first base case is when the element is not a list, the function returns the element unmodified. The second base case is when the element is string, and this string contains the same number of * characters as the current depth we are applying VI rules at. The recursive case is when the element is a list. In this case, for each item in the list, the function `replace_stars_with_vi()` is called again.

```

def replace_stars_with_vi(data, current_depth,
    vi_rules):
    """
    Replace * characters with vocabulary items, to apply
    VI from bottom up
    """
    if isinstance(data, list):
        return [replace_stars_with_vi(item, current_depth,
            vi_rules) for item in data]
    elif isinstance(data, str):
        if data.count('*') == current_depth:
            rule = vi_rules[data.strip("*")]
            out_rule = "\n".join(rule)
            if out_rule[INITIAL_INDEX] != '\n':
                out_rule = '\n' + out_rule
            return data.strip('*') + out_rule

```

```
return data
```

Listing 5.15 Replace stars with phonological material

This function is called at every level of the base structure, and the output list is saved at every level. To exemplify this, all lists generated by the helper function at every stage, using the VI rules in Listing 5.16, is presented in Listing 5.17.

```
{'T': ['2sg', 'u'], 'V': ['', 'akh'], 'v': ['', 'a'],
  'MOOD': ['subjunctive', 'e']}
```

Listing 5.16 Example VI Rules

```
['MOODP', ['TP', 'T***', ['vP', ['VP', 'V\nakh'], 'v
**']], 'MOOD****']
['MOODP', ['TP', 'T***', ['vP', ['VP', 'V\nakh'], 'v\
na']], 'MOOD****']
['MOODP', ['TP', 'T\n2sg\nu', ['vP', ['VP', 'V\nakh
'], 'v\na']], 'MOOD****']
['MOODP', ['TP', 'T\n2sg\nu', ['vP', ['VP', 'V\nakh
'], 'v\na']], 'MOOD\nsubjunctive\ne']
```

Listing 5.17 Step-by-step VI Process

5.8 Reduplicate Base Structure

After the base structures have been generated, the corresponding reduplication forms are processed. The code does so by inserting REDP and RED nodes at relevant positions in the base underlying structure. These positions are specified in the input file `red.txt`. Since some reduplicant forms are possible only in some phonological environments, the insertion of the REDP is done so only when VI rules have been applied to the base structure. For example, given the VI rules specified in Listing 5.2, the combinations of VI rules is stated below.

```
{'T': ['2sg', 'u'], 'V': ['', 'akh'], 'v': ['', 'a'],
  'MOOD': ['subjunctive', 'e']}
```

```
{'T': ['2sg', 'u'], 'V': ['', 'sebenz'], 'v': ['', 'a
'], 'MOOD': ['subjunctive', 'e']}
```

For each VI combination, the code then takes the phonological string of the root 'V'. For now, the code only checks whether the root is vowel initial, as this is required by the reduplication patterns of Zulu. Additional work may be required to extend the functionality to other languages.

```
for vi_rules in all_vi_rules:
    all_redup_vi_lst = []
```

```

# generate reduplicant forms
root = vi_rules['V'][LAST_INDEX]
all_syntactic_redup = reduplicate_base_structure(
    syntactic_base, dominated_nodes, root)

```

Listing 5.18 Generate underlying reduplicant form, for every VI rule combination.

The helper function `reduplicate_base_structure()` adds the nodes REDP and RED to the relevant positions of the list.

5.8.1 Apply RED Phonological Material

Since the phonological material of the RED morpheme is not explicitly stated in `vi_rules.txt`, and instead relies on the phonological material of previously assigned morphemes, the VI process for the RED morpheme requires a process. The helper function `add_vi_rule_redup()` deals with this process. This function is called during the VI process outlined in subsection 5.7.2, and when the morpheme is identified as RED. The function identifies any string marked by a new line as phonological material, as this is how the VI process is conducted above.

```

phonological_content = ''
for item in data:
    # new line character means phonological output has
    # been added to the structure
    if '\n' in item:
        item = item.split('\n')

        phonological_content += item[
            PHONOLOGICAL_MATERIAL_INDEX]

```

Listing 5.19 Assign phonological material to RED (without prosodic template).

This approach works for the Kerewe example, where the reduplicant copies all phonological material that is c-commands. However, for a language like Zulu, the reduplicant copies only part of the previously assigned morphemes. To deal with this, the reduplicant copies phonemes until the template is filled.

```

if scope == 'bisyllabic':
    vowel_count = 0

    for char in item[PHONOLOGICAL_MATERIAL_INDEX]:
        if not template_filled:
            phonological_content += char
            if char in VOWEL_LST:
                vowel_count += 1
            if vowel_count >= 2:

```

```

        template_filled = True
    else:
        break

```

Finally, epenthesis vowels may be added to the reduplicant morpheme. The code checks whether the root is vowel initial, and adds the epenthesis vowel only when it is in the correct environment. If the epenthesis vowel is required in any position (such as the Kerewe example), the code deals with that situation, too.

```

if epenthesis:
    if environment == "VOWEL":
        if vowel_initial and node == 'V':
            phonological_content += epenthesis
    else:
        phonological_content += epenthesis

```

For future work, it would be worthwhile to extend the bisyllabic template, and conditions in which epenthesis vowels are required. For now, however, the code works sufficiently well for the data described above.

5.9 General Phonological Processes

In the previous section, I outlined how the underlying phonological form of each morpheme is assigned to the base structure. Once that has been completed, the code applies more general phonological processes to derive the final surface form. As discussed in sections 2 and 3, there is evidence that these general phonological processes are applied *after* the VI process is complete: the surface form of the reduplicant in these languages is a copy of the phonological form of the base *before* these more general processes are applied. As such, in the computational implementation, general phonological processes are applied only after the VI process is complete.

The helper functions `apply_phonological_processes()` and `find_and_replace_element()` apply phonological processes to the final output produced by `replace_stars_with_vi()`. As outlined in section 5.5.3, the phonological rules text file is read in as a dictionary, with the illegal phonological environment as keys and the changed phonemic string as values. The helper function uses this information, and checks whether the (flat) phonological word contain any illegal environments. If there are, the code locates those environments in the hierarchical structure, and replaces those phonological strings as specified by the phonological process.

Illegal phonological sequences may occur within a morpheme, or at a morpheme boundary. Words in this script are stored hierarchically, meaning that morphemes are stored in separate lists. As such, the process to manipulate strings changes depending on whether the strings occur at morpheme boundaries. The function `find_and_replace_element()` contains the boolean argument `is_separate_node`,

to specify whether the sequence occurs at a morpheme boundary (TRUE) or not (FALSE).

If the sequence occurs at a morpheme boundary, the script replaces the last index of the first morpheme, and the first element of the second morpheme.

```
if is_separate_node:
    # if we are replacing the first node, replace the
    # last character
    if is_initial_node:
        phon_string[LAST_INDEX] = replace_phon_string
    # if we are replacing the second node, replace the
    # first character
else:
    phon_string[INITIAL_INDEX: len(
        replace_phon_string)] = replace_phon_string
```

Listing 5.20 Resolve illegal phonological sequence at morpheme boundary

If the sequence occurs within a morpheme, then the normal indices can be used to replace the string at the relevant position.

```
# replace the first character
if is_initial_node:
    phon_string[index_to_replace.initial] =
        replace_phon_string
# replace the second character
else:
    phon_string[index_to_replace.final] =
        replace_phon_string
```

Listing 5.21 Resolve illegal phonological sequence within same morpheme

This process is applied continuously until there are no more illegal phonological environments. The code does so using while loop, and calling Python's built in `any()` function.

5.10 Save Output

The output produced by the script is saved using the library `svgling`. The library convert tuples into linguistic-style constituent trees, and are saved in .svg format. These files can be opened in a standard internet browser.

The nested lists are converted into tuples, and are then saved in the output folder specified in the command line. The output filename includes the step iteration and word iteration. The final derivation is saved with “_FINAL” appended to the filename. Possible reduplication variants are also included in the filename, where the variants of the same base word will contain the same word iteration.

```

def save_svg_file(structure, filename, output_folder)
:
"""
Save svg file
"""
output_filename = f'{output_folder}/{filename}.svg'
structure = list_to_tuple(structure)
structure = svgling.draw_tree(structure)
structure.get_svg().saveas(output_filename)

```

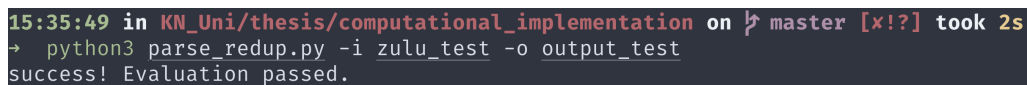
Listing 5.22 Save svg file

The word iteration in the filename matches the number in the output .txt file, which the user can use to look up a particular word.

5.11 Evaluation

The output generated by the program, for the Zulu and Kerewe data, is saved on Github. The script can also check whether all expected words are produced with the input files. The optional evaluation file is named `eval.txt`, and is a single column text file that contains all expected words. If all words produced by the script are included in the evaluation file, the script prints **success! Evaluation passed** to the standard output, as shown in figure 81.

(81) Screenshot of an evaluation that has passed



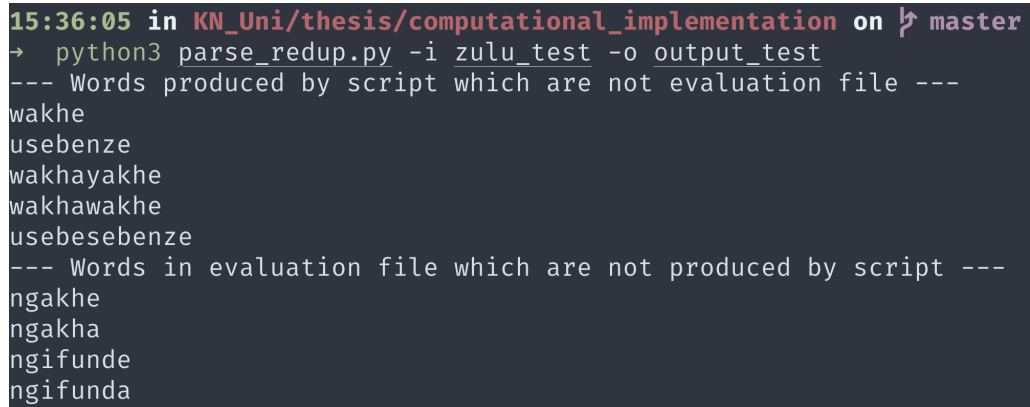
```

15:35:49 in KN_Uni/thesis/computational_implementation on master [x!?] took 2s
→ python3 parse_redup.py -i zulu_test -o output_test
success! Evaluation passed.

```

If there are some words that the script produced that are not included in `eval.txt`, then these words are printed to the standard output. Similarly, if there are words that the script did not produce, which are included in `eval.txt`, then these are also printed to standard output. An example is shown in figure 82.

(82) Screenshot of an evaluation that has failed



```
15:36:05 in KN_Uni/thesis/computational_implementation on master
→ python3 parse_redup.py -i zulu_test -o output_test
--- Words produced by script which are not evaluation file ---
wakhe
usebenze
wakhayakhe
wakhawakhe
usebesebenze
--- Words in evaluation file which are not produced by script ---
ngakhe
ngakha
ngifunde
ngifunda
```

All datasets provided on Github pass the evaluation, and a list of all words are provided in Appendix A.

CHAPTER 6

Conclusion and Further Questions

In this thesis, I have analyzed verbal reduplication patterns in two Bantu languages, Zulu and Kerewe. As discussed in chapter 1, an investigation into the morphosyntactic interface provide insight into how syntactic structure can influence the expression of phonological form. By focussing on reduplication specifically, I investigated whether the phonological form of the reduplicant can be influenced by an internal syntactic structure within words. At least in the data presented here, I have found that proposing a word internal syntactic structure can explain patterns that would be otherwise unexplained, thereby suggesting that words are sensitive to an internal structure. Specifically, a mismatch between the base and the reduplicant, and the variant forms of reduplicated forms, can be understood by internal syntactic constraints. These findings were formalized in a computational implementation in Python, which automatically generates diagrams of each step of the analysis. As the program is designed to be extensible to other reduplication patterns in other languages, it is hoped that it can be used to gain insight into reduplication patterns in other languages.

There are some areas which are open for future work. Firstly, it was found that both Zulu and Kerewe exhibit variant reduplication forms. In Zulu, I argued that this was due to language-specific phonological requirements, where typical phonological processes to resolve illegal sequences were insufficient to adhere to Zulu phonology. For Kerewe, the motivation for variant forms in partial reduplication was less clear. Though the analysis explains *how* the reduplicant receives its phonological material, it does not explain *why* this is the case. One avenue to investigate this issue is to focus on the triggers of head movement that construct these complex words. As discussed in chapter 1.1 complex words in Distributed Morphology are constructed via head movement. The findings in this thesis suggest that syntactic constraints may influence phonological form, thereby providing evidence that words may be sensitive to an internal structure. Further research into how these words are constructed through head movement (Julien, 2002) may also provide an explanation as to why partial reduplication in Kerewe has variant forms. An investigation into this area may also shed light on the mystery of the final vowel on the reduplicant, as discussed in section 3.1. Finally, the computational implementation of this analysis has focussed on the general principles of reduplication: that the reduplicant receives its phonological material from morphemes that are placed in a lower structural position than the position of the reduplicant. The program includes language

specific phonological processes for the data presented here, but for future work it would be worthwhile to include more processes for a range of languages.

Though there are still some open issues, this thesis has emphasized the importance of investigating the interface between syntax and phonology. By focussing on reduplication - a process that sits at the heart of phonology, morphology, and syntax - we can directly test the ways in which these subsystems relate to each other. In doing so, we can gain a deeper understanding of each subsystem, and the ways in which the underlying structure of words may be uniform across languages.

Bibliography

- Abney, S. P. (1987). *The English Noun Phrase in its Sentential Aspect* (Doctoral dissertation). Massachusetts Institute of Technology.
- Asudeh, A., & Siddiqi, D. (2022). Realizational Morphosemantics in LRFG. In M. Butt, J. Y. Findlay, & I. Toivonen (Eds.), *The Proceedings of the LFG'22 Conference*. CSLI Publications.
- Baunaz, L., & Lander, E. (2018, June 21). *Nanosyntax* (Vol. 1). Oxford University Press. <https://doi.org/10.1093/oso/9780190876746.003.0001>
- Bobaljik, J. D. (2000). The Ins and Outs of Contextual Allomorphy.
- Bresnan, J. (2015). *Lexical Functional Syntax* (Second edition). Wiley-Blackwell.
- Bruening, B. (2018). The lexicalist hypothesis: Both wrong and superfluous. *Language*, 94(1), 1–42. <https://doi.org/10.1353/lan.2018.0000>
- Caha, P. (2010). The parameters of case marking and spell out driven movement. *Linguistic Variation Yearbook*, 10, 32–77. <https://doi.org/10.1075/livy.10.02cah>
- Chomsky, N. (1970). Remarks on nominalization. In R. A. Jacobs & P. S. Rosenbaum (Eds.), *Readings in english transformational grammar* (pp. 184–221). Ginn.
- Cook, T. (2013). Explaining the Final Vowel Mismatch in Zulu Reduplication. *University of Pennsylvania Working Papers in Linguistics*, 19(1). <https://repository.upenn.edu/pwpl/vol19/iss1/6>
- Cook, T. (2018). The Inclusion of Prefixal Material in Zulu Reduplication. *Studies in African Linguistics*, 43–69. <https://doi.org/10.32473/sal.v47i1.107653>
- Downing, L. (1997). Correspondence Effects in Siswati Reduplication. *Studies in the Linguistic Sciences*, 25.
- Downing, L. (1999). Morphological constraints on Bantu reduplication. *Linguistic Analysis*, 29.
- Embick, D. (2000). Features, Syntax, and Categories in the Latin Perfect. *Linguistic Inquiry*, 31(2), 185–230. <https://doi.org/10.1162/002438900554343>
- Embick, D. (2015). *The Morpheme: A Theoretical Introduction* (Vol. 31). De Gruyter Mouton. <https://go.exlibris.link/9zt0QwpB>
- Embick, D., & Halle, M. (2005). On the Status of *Stems* in Morphological Theory. In T. Geerts, I. van Ginneken, & H. Jacobs (Eds.), *Current Issues in Linguistic Theory* (pp. 37–62). John Benjamins Publishing Company. <https://doi.org/10.1075/cilt.270.03emb>

- Geraghty, P. (2001, December 21). Nadrogā. In J. Lynch, M. Ross, & T. Crowley (Eds.), *The Oceanic Languages*. Routledge Handbooks Online. <https://doi.org/10.4324/9780203820384.ch46>
- Halle, M., & Marantz, A. (1993). Distributed Morphology and the Pieces of Inflection. In K. Hale & S. J. Keyser (Eds.), *The View From Building 20* (pp. 111–176). MIT Press.
- Halle, M., & Marantz, A. (1994). Some key features of Distributed Morphology. *MIT working papers in linguistics*, 21, 275–288
21(275), 88.
- Harley, H. (2014). On the identity of roots. *Theoretical Linguistics*, 40(3-4). <https://doi.org/10.1515/tl-2014-0010>
- Harley, H., & Noyer, R. (1999). Distributed Morphology. In L. Cheng & R. Sybesma (Eds.), *Glott International* (pp. 463–496). De Gruyter Mouton. <https://doi.org/10.1515/9783110890952.463>
- Haugen, J., & Siddiqi, D. (2016, June 15). Towards a restricted realization theory: Multimorphemic monolistemicity, portmanteaux, and postlinearization spanning. In D. Siddiqi & H. Harley (Eds.), *Morphological Metatheory* (pp. 343–387). John Benjamins Publishing Company. <https://doi.org/10.1075/la.229>
- Hayes, B. (2009). *Introductory Phonology*. Wiley-Blackwell
OCLC: 212893710.
- Hyman, L. M. (2009). The Natural History of Verb-Stem Reduplication in Bantu. *Morphology*, 19(2), 177–206. <https://doi.org/10.1007/s11525-009-9140-y>
- Hyman, L. M., Inkelas, S., & Sibanda, G. (2008). Morphosyntactic Correspondence in Bantu Reduplication. In K. Hanson & S. Inkelas (Eds.), *The Nature of the Word*. The MIT Press. <https://doi.org/10.7551/mitpress/7894.003.0017>
- Inkelas, S., & Downing, L. J. (2015). What is Reduplication? Typology and Analysis Part 1/2: The Typology of Reduplication: What is Reduplication? Typology. *Language and Linguistics Compass*, 9(12), 502–515. <https://doi.org/10.1111/lnc3.12166>
- Julien, M. (2002). *Syntactic heads and word formation*. Oxford University Press.
- Lǐ, Y., & Ponsford, D. (2018). Predicative reduplication: Functions, their relationships and iconicities. *Linguistic Typology*, 22(1), 51–117. <https://doi.org/10.1515/lingty-2018-0003>
- Marantz, A. (1982). Re Reduplication. *Linguistic Inquiry*, 13(3), 435–482. Retrieved June 7, 2023, from <http://www.jstor.org/stable/4178287>
- Marlo, M., R. (2002). *Reduplication in Lusaamia* (Working Paper).
- McGinnis-Archibald, M. (2016). Distributed Morphology. In A. Hippisley & G. Stump (Eds.), *The Cambridge Handbook of Morphology* (pp. 390–423). Cambridge University Press. <https://doi.org/10.1017/9781139814720.015>
- Meeussen, A. E. (1967). Bantu grammatical reconstructions. *Africana Linguistica*, 3(1), 79–121. <https://doi.org/10.3406/affin.1967.873>
- Moskal, B. (2015, August 24). *Domains on the Border: Between Morphology and Phonology*. MIT. <https://opencommons.uconn.edu/dissertations/892>

- Müller, S., Abeillé, A., Borsley, R. D., & Koenig, J.-P. (Eds.). (2021). *Head driven phrase structure grammar. The handbook*. Language Science Press.
- Mutaka, N., & Hyman, L. M. (1990). Syllables and morpheme integrity in Kikande reduplication. *Phonology*, 7(1), 73–119. <https://doi.org/10.1017/S0952675700001123>
- Nash, D. (1980). *Topics in Warlpiri grammar*. Massachusetts Institute of Technology.
- Nevins, A. (2012, September 27). Haplological Dissimilation at Distinct Stages of Exponence. In J. Trommer (Ed.), *The Morphology and Phonology of Exponence* (pp. 84–116). Oxford University Press. <https://doi.org/10.1093/acprof:oso/9780199573721.003.0003>
- Newell, H. (2009). *Aspects of the Morphology and Phonology of Phases*. McGill University. Canada. Retrieved April 18, 2023, from <https://escholarship.mcgill.ca/concern/theses/1c18dh59v>
- Newell, H., & Piggott, G. (2014). Interactions at the Syntax–Phonology interface: Evidence from Ojibwe. *Lingua*, 150, 332–362. <https://doi.org/10.1016/j.lingua.2014.07.020>
- Nurse, D., & Philippson, G. (Eds.). (2003). *The Bantu languages*. Routledge.
- Odden, D. (1995). The status of onsetless syllables in Kikerewe. *Working Papers in Linguistics*, 47, 89–110.
- Odden, D. (1996). *Patterns of Reduplication in Kikerewe* (Working Paper). Ohio State University. Department of Linguistics. Retrieved June 7, 2023, from <https://kb.osu.edu/handle/1811/81491>
- Posthumus, L. (2022). A Systemized Explanation for Vowel Phoneme Change in the Inadmissible Phonological Structure /VV/ in Zulu. *STUDIES IN AFRICAN LANGUAGES AND CULTURES*, (56). <https://doi.org/10.32690/56.1>
- Regier, T. (1998). Reduplication and the Arbitrariness of the Sign. In M. A. Gernsbacher & S. J. Derry (Eds.), *Proceedings of the 20th Annual Conference of the Cognitive Science Society*. Lawrence Erlbaum.
- Stump, G. T. (2001, February 22). *Inflectional Morphology: A Theory of Paradigm Structure* (1st ed.). Cambridge University Press. <https://doi.org/10.1017/CBO9780511486333>
- Trommer, J. (2001). *Distributed Optimality* (Doctoral dissertation). Universität Potsdam.
- Urbanczyk, S. (2017, March 29). Phonological and Morphological Aspects of Reduplication. *Oxford Research Encyclopedia of Linguistics*. Oxford University Press. <https://doi.org/10.1093/acrefore/9780199384655.013.80>
- Vicente, L. (2007). *The syntax of heads and phrases: A study of verb (phrase) fronting*. LOT.
- Wiland, B. (2019). *The spell-out algorithm and lexicalization patterns: Slavic verbs and complementizers*. Language Science Press
OCLC: 1117896428.

Appendices

CHAPTER A

List of Output Words

A.1 Zulu

0. ngakhe
1. ngakha
2. ngifunde
3. ngifunda
4. ngisebenze
5. ngisebenza
6. ngipheke
7. ngipheka
8. ngenze
9. ngenza
10. wakhe
11. wakha
12. ufunde
13. ufunda
14. usebenze
15. usebenza
16. upheke
17. upheka
18. wenze
19. wenza
20. nakhe
21. nakha
22. nifunde
23. nifunda
24. nisebenze
25. nisebenza
26. nipheke
27. nipheka
28. nenze
29. nenza
30. sakhe

31. sakha
32. sifunde
33. sifunda
34. sisebenze
35. sisebenza
36. sipheke
37. sipheka
38. senze
39. senza
- reduplicated words ---
40. ngakhayakhe
41. ngakhangakhe
42. ngakhayakha
43. ngakhangakha
44. ngifundafunde
45. ngifundafunda
46. ngisebesebenze
47. ngisebesebenza
48. ngiphekapheke
49. ngiphekapheka
50. ngenzayenze
51. ngenzangenze
52. ngenzayenza
53. ngenzangenza
54. wakhayakhe
55. wakhawakhe
56. wakhayakha
57. wakhawakha
58. ufundafunde
59. ufundafunda
60. usebesebenze
61. usebesebenza
62. uphekapheke
63. uphekapheka
64. wenzayenze
65. wenzawenze
66. wenzayenza
67. wenzawenza
68. nakhayakhe
69. nakhanakhe
70. nakhayakha
71. nakhanakha
72. nifundafunde

73. nifundafunda
74. nisebesebenze
75. nisebesebenza
76. niphekapheke
77. niphekapheka
78. nenzayenze
79. nenzanenze
80. nenzayenza
81. nenzanenza
82. sakhayakhe
83. sakhasakhe
84. sakhayakha
85. sakhasakha
86. sifundafunde
87. sifundafunda
88. sisebesebenze
89. sisebesebenza
90. siphekapheke
91. siphekapheka
92. senzayenze
93. senzassenze
94. senzayenza
95. senzassenza

Listing A.1 Output of Zulu words

A.2 Kerewe

0. afuzile
1. alozile
2. nfuzile
3. ndozile
- reduplicated words ---
4. afulafuzile
5. alolalozile
6. nfulafuzile
7. ndolalozile

Listing A.2 Output of Kerewe Words, Mismatch Between Base and Reduplicant

0. kulimilana
- reduplicated words ---
1. kulimilanalimilana

2. kulimilalimilana
3. kulimalimilana

Listing A.3 Output of Kerewe Words, Parital Reduplication