Don't Gloss Over the Gloss: Using Grammatical Features to Improve Morphotactic Inference in the AGGREGATION System

Isaac Schifferer

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2026

Committee:
Emily M. Bender
Fei Xia

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Don't Gloss Over the Gloss: Using Grammatical Features to Improve Morphotactic Inference in the AGGREGATION System

Isaac Schifferer

Chair of the Supervisory Committee:
Emily M. Bender
Linguistics

In the field of linguistics, language data is often presented in the form of interlinear glossed text (IGT). Due to the large amount of work and language analysis required to produce IGT, as well as its widespread use and somewhat standardized format, IGT data is a rich source of structured linguistic information. The AGGREGATION project is an ongoing effort to employ this information for the automatic inference of implemented grammars. One of the perennial challenges for automatic grammar inference is producing grammars that not only make correct predictions about language data, but also whose structures are intelligible to linguists. In this regard, grammars produced by the AGGREGATION system often fall short, particularly when it comes to languages' morphotactic systems, which are modeled as a series of position classes. This thesis works towards remedying this issue by exploring the ways in which the grammatical feature information in IGT gloss lines can be better utilized for morphotactic inference. By reorienting the inference process towards creating position classes based on grammatical features, the quality of the resulting grammars was improved without impacting their performance. These changes were evaluated using datasets representing six diverse languages, each from a different language family and geographical location. Three of these datasets (comprising of data from Wambaya, Lezgi, and Manipuri) were consulted during the development process, while the other three (comprising of data from Wakhi, South Efate, and Hiaki) were held out and used in the final evaluation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF AlgorithmS

ABBREVIATIONS

ACE: Answer Constraint Engine

AGGREGATION: Automatic Generation of Grammars for Endangered Languages from Glosses and Typological Information

BASIL: Building Analyses from Syntactic Inference in Low-resource languages

HPSG: Head-Driven Phrase Structure Grammar

IGT: interlinear glossed text

INTENT: Interlinear Text Enrichment Toolkit

LC: lexical class

LKB: Linguistic Knowledge Building system

LRI: lexical rule instance

LRT: lexical rule type

MOM: Matrix-ODIN Morphology inference module

PC: position class

ACKNOWLEDGMENTS

First, I must thank my advisor, Emily M. Bender, for her support throughout the thesis process. I truly appreciate her mentorship and the thoroughness and precision with which she works, in advising and beyond. I would also like to thank Fei Xia for being my reader and for providing me with valuable feedback.

I also must thank my CLMS classmates and fellow treehouse dwellers for welcoming me to Seattle and for helping to keep me accountable over this last year.

Lastly, I would like to thank my family, my parents, sisters, grandparents, cousins, and all the rest, for their support of me and my academic aspirations, even as I moved across the country to pursue them.

After-lastly, and least importantly, thank you to crosswords and volleyball and just about every kind of of grid-based logic puzzle.

Chapter 1

INTRODUCTION

Implemented grammars are a useful tool with which linguists can conduct linguistic hypothesis testing (Bender, 2008). Linguists can rapidly and frequently test their implemented grammar against a set of data as they develop their analysis of a language, greatly reducing the amount of work required to evaluate the interactions of analyses in the grammar, therefore saving time and helping to limit unexpected behavior. Despite these benefits, the process of developing an implemented grammar can itself be very time-consuming, especially when working within precision grammar frameworks such as HPSG. The LinGO Grammar Matrix (Bender et al., 2002, 2010; Zamaraeva et al., 2022) is a system that was designed with the goal of speeding up this process by drawing from a pool of generalized analyses that have been developed for various linguistic phenomena. The AGGREGATION project (Bender et al., 2013, 2014) aims to further this goal by using the information encoded in IGT data to automatically infer Matrix grammars.

Given the difficulty of this task, no grammar produced by the AGGREGATION pipeline will be perfect, so the user-linguist will want to continue developing it after its initial generation. Because of this, it's important that the inferred grammars are easily interpretable and that (ideally) the structures in the grammar are close to what the linguist would define if they were building the grammar from scratch. One place where inferred grammars often fall short of this standard is in the model of a language's morphotactic system. As it stands, the morphotactic inference process relies primarily on the relative order of affixes on words and only sparsely takes advantage of the grammatical feature information expressed by affixes. In light of this, the research question asked in this thesis is, "How can the gram-

matical feature information in IGT gloss lines be utilized to enhance the automatic inference of morphotactics?"

Chapter 2 provides the relevant background for this thesis and situates it within the larger context of the AGGREGATION project, including information about how morphotactic systems are represented in Matrix grammars, how those systems are currently inferred in the AGGREGATION pipeline, and the concepts from the theoretical morphology literature that are instructive for designing a morphotactic inference process based on feature information. Chapter 3 describes the six datasets used to develop and test this new inference strategy and the methods employed to evaluate its effects. It also details the parsing and grammar complexity metrics used to quantify the results as well as the approach taken to manually analyze the resulting grammars. Chapter 4 discusses the changes made to the morphotactic inference process that leverage feature information with the goal of producing more realistic models of morphotactics. It also explains how these changes were informed by the morphotactic diversity seen in the world's languages and how this was balanced with the challenges presented by using data that was not specifically designed for this task. Chapter 5 presents and analyzes the results of the experimental changes and includes discussion of the most prominent patterns observed in the results. Lastly, Chapter 6 summarizes the findings of the experiments and describes potential avenues for future work.

Chapter 2
BACKGROUND

This chapter provides details about the grammar inference pipeline that this thesis operates within as well as the theoretical linguistic background that motivates my approach to morphotactic inference. Section 2.1 will describe the LinGO Grammar Matrix project with attention to the way it handles morphology. Section 2.2 will briefly introduce the IGT data format. Section 2.3 will lay out the AGGREGATION project and discuss the end-to-end pipeline that is used to perform grammar inference. Finally, Section 2.4 will provide the relevant linguistic background, focusing on morphotactics and inflectional morphology.

2.1 The LinGO Grammar Matrix

This section describes the grammar customization system that the grammar inference pipeline is built around, including details about the grammar specification that is produced by the inference process and used by the customization system to produce an implemented grammar. The LinGO Grammar Matrix (the Matrix; Bender et al., 2002, 2010; Zamaraeva et al., 2022) is a system that allows users to interactively develop machine-readable grammars in the Head-Driven Phrase Structure Grammar (HPSG; Pollard and Sag, 1994) framework. By populating an online questionnaire with grammatical information about their language of choice, users produce a grammar specification, also called a choices file, which is a plain-text encoding of the information they provided. From this choices file, the Grammar Matrix customization system generates a grammar that can be used parse and generate sentences (see Section 2.3.4 for more details of this process post-generation).

The Matrix is a useful tool for linguists because it provides a jump start on the development of grammars that can be used for linguistic hypothesis testing (Bender, 2008). This is

enabled by the wealth of generalized implementations of various aspects of grammar in the Matrix's core grammar and collection of modular libraries. The core grammar, which is hypothesized to be useful for developing a grammar for any language, defines a type hierarchy that is used in all Matrix grammars. It contains basic type definitions for everything from booleans to lexical and phrase structure rules, and it is the basis upon which the libraries and eventually the user-linguist build a grammar for a specific language. Each library focuses on some linguistic phenomenon (e.g. adnominal possession (Nielsen, 2018), valence-changing morphology (Curtis, 2018), evidentiality (Haeger, 2017)) and provides implemented analyses for many of the expressions of each phenomenon documented in the typological literature. If a user fills out the corresponding library page in the questionnaire, the analyses matching the user's answers will be incorporated into their grammar. For example, if the language at hand has clausal complements, the user can define any number of clausal complement types and for each one specify where object complement clauses are placed, whether propositions or questions are being embedded, whether they are marked with a complementizer, and so on (Zamaraeva et al., 2019b). From these selections, the backend code will generate the proper types (lexical rules, feature types and values, etc.) to add to the grammar.

2.1.1 Morphology in the Matrix

Most relevant to this thesis is the Matrix's handling of morphology, first implemented by O'Hara (2008) and then revised by Goodman (2013). The primary concern of the Matrix with respect to morphology is morphotactics, which has to do with the ordering and co-occurrence restrictions of morphemes. This is largely due to the fact that the Matrix assumes a morphophonological analyzer that "standardizes" the data by handling things like allomorphy and mapping any kind of non-concatenative morphology (e.g. infixes) to a strictly concatenative representation (Bender and Good, 2005; Bender et al., 2010). Matrix grammars define the morphotactics of a language in terms of a set of lexical types and lexical rule types.

Lexical entries, also called lexical classes (LCs), instantiate the lexical types and make

```
verb1_name=verb1
  verb1_feat1_name=case
  verb1_feat1_value=acc
  verb1_feat1_head=obj
verb1_valence=trans
  verb1_stem1_orth=aliyulu
  verb1_stem1_pred=_find_v_rel
  verb1_stem2_orth=didima
  verb1_stem2_pred=_tell_v_rel
```

Figure 2.1: An LC from a Wambaya grammar as analyzed from Nordlinger (1998)

up the grammar's lexicon. An LC is composed of one or more stems with corresponding predicate values. LCs can also have any number of supertypes and feature-value pairs defined on them. Figure 2.1 shows an example of a transitive verb LC. This LC has no supertypes, constrains the object of the verb to having accusative case, and contains two stems.

The lexical rules of a grammar are modeled as a set of position classes (PCs). Each PC represents a prefix or suffix affix slot and can be obligatory or not. A PC has a set of inputs and is composed a set of lexical rule types (LRTs). Each input is either an LC or another PC. An LRT contains one or more lexical rule instances (LRIs) and has some number of feature-value pairs associated with it. Each LRI represents the orthographic form of an affix expressing the feature values defined by the LRT. Figure 2.2 shows an example of an optional PC for verb suffixes. This PC has three inputs (two LCs and one PC) and two LRTs. The first LRT has one form and expresses no features and the second has one one form and expresses nonfuture tense.

To enforce co-occurrence restrictions among types, LCs, PCs, and LRTs can have 'requires' or 'forbids' constraints defined on them. A 'requires' constraint, as the name suggests,

```
verb-pc1_name=verb-pc1
verb-pc1_order=suffix
verb-pc1_inputs=verb1, verb-pc6, verb15
  verb-pc1_lrt1_name=verb-pc1_lrt1
    verb-pc1_lrt1_lri1_inflecting=yes
    verb-pc1_lrt1_lri1_orth=-j
  verb-pc1_lrt2_name=verb-pc1_lrt2
    verb-pc1_lrt2_feat1_name=tense
    verb-pc1_lrt2_feat1_value=nfut
    verb-pc1_lrt2_feat1_head=verb
    verb-pc1_lrt2_lri1_inflecting=yes
    verb-pc1_lrt2_lri1_orth=-lumi
```

Figure 2.2: A PC from a Wambaya grammar as analyzed from Nordlinger (1998)

requires one of the (stems or affixes of the) LCs, PCs, or LRTs listed in it to co-occur with (one of the stems or affixes of) the type it is defined on. Similarly, 'forbids' constraints disallow any of the (stems or affixes of the) listed types from co-occurring with (any of the stems or affixes of) the type it is defined on.

The overall morphotactic system of any given part of speech[1] in a grammar can be conceptualized as a directed, acyclic graph with the LCs and PCs as the nodes and the inputs defined in each PC forming edges from the input LCs/PCs to the PC. This is sometimes referred to as an "input graph". From this perspective, a word is formed by starting at some LC node and moving through the input graph by following an out-edge at each node, picking up a stem in the LC node and an affix at each subsequent PC node, stopping at any point that satisfies all of the constraints (i.e. obligatory PCs and co-occurrence restrictions). The orthographic form of the word is also constructed along this path, with each affix being appended to the appropriate edge of the word form based on its status as a prefix or a suffix. There are no initial restrictions on the relative ordering of prefixes and suffixes, so prefixes can be inputs to suffixes and vice versa, though grammars will of course end up with some defined order of affixation based on the inputs of each PC. The set of stem and affix combinations that can be created via a path through an input graph make up the set of valid inflected words in the grammar.

This section has reviewed the morphotactic system of the Grammar Matrix because the goal of this thesis is to improve the morphotactic inference module of the grammar inference pipeline, which handles the inference of the LCs and PCs. Just as crucial is an understanding of the data used for inference, which will be outlined in the next section.

2.2 IGT

The primary input to the grammar inference system is a corpus of interlinear glossed text (IGT) data for the language being modeled. IGT is a common way that linguists present

[1]The Matrix currently allows users to define the morphology of nouns, verbs, adjectives, copulas, and determiners

annotated language data and as such is a rich source of linguistic information. The format of IGT data is generally consistent across resources, following a standard such as the widely-adopted Leipzig Glossing Rules (Bickel et al., 2015), though there is some level of variation across IGT, especially in terms of how different grammatical features are glossed. An instance of IGT typically contains a sentence worth of data and consists of at least three lines, a language line, a gloss line, and a translation line. An example of this is given in (1):

(1) Wambaya (Nordlinger, 1998; pg. 73)

Ngajbi ng-a ilarra-wulu
see 1SG.A-PST eaglehawk-DU(ACC)

'I saw two eaglehawks.'

The language line contains the sentence in the language being studied where the morpheme boundaries in each word are marked, typically by hyphens. In (1), *ng-a* and *ilarra-wulu* have two morphemes each. There is sometimes an additional line before this one that presents the sentence in the standard orthography of the language. The gloss line is word-aligned to the language line and provides a gloss for each morpheme. Glosses can represent different types of things, such as (from the perspective of HPSG) feature values or predicate names. If a single morpheme expresses multiple features and/or predicates, each value is given in the morpheme gloss, normally separated by periods. Each feature value in a gloss is referred to as a *gram*. For example, in (1), the morpheme *ilarra* is glossed with the predicate 'eaglehawk' and the prefix *ng-* is glossed with three grams, with 1 representing first person, SG representing singular number, and A representing ergative case (Nordlinger, 1998; pg. 56). The translation line gives a translation of the sentence in a language of wider communication such as English.

The work presented in this thesis aims to improve the extent to which the information in IGT data is utilized, in this case by incorporating the feature information in the glosses more fully. This is also true in general for the grammar inference pipeline, which is described in the next section.

2.3 AGGREGATION

This section presents the grammar inference system that the development work of this thesis sits within. The AGGREGATION project (Automatic Generation of Grammars for Endangered Languages from Glosses and Typological Information; Bender et al., 2013, 2014) centers on approaches for automatically inferring Matrix grammars based on the analysis of information stored in IGT data, essentially by mirroring the process of filling out the Matrix's online questionnaire to produce a choices file that can then be used by the Matrix to generate a grammar. The most recent major iteration on this project was introduced by Howell (2020), who established an end-to-end pipeline for grammar inference, compiling much of the work done for the AGGREGATION project up to that point as well as expanding upon its capabilities. The high-level outline of the pipeline is given in Figure 2.3.

In the first section of the pipeline, IGT data is converted to another data format called Xigt (Goodman et al., 2015) and is enriched with additional information such as part-of-speech tags. This will be described in more detail in Section 2.3.1. The main inference step of the pipeline is made up of two modules, BASIL and MOM, that use the data to infer a grammar specification in the form of a choices file. These modules will be discussed in Sections 2.3.2 and 2.3.3, respectively. After the inference step, the pipeline takes advantage of several existing tools to generate the grammar, compile it, and then use it to parse sentences to evaluate its performance. These steps will be described in Section 2.3.4.

2.3.1 Xigt

The initial input to the AGGREGATION pipeline is IGT data. Before it is passed to the grammar inference modules, the data is converted to the Xigt format and enriched with the INTENT package (Georgi, 2016; Xia et al., 2016). Xigt is an XML-based format for IGT that was designed to be able to encode the various structural annotations added to the original IGT in the enrichment process (Goodman et al., 2015). An abbreviated example of a Xigt instance is given in Figure 2.4.

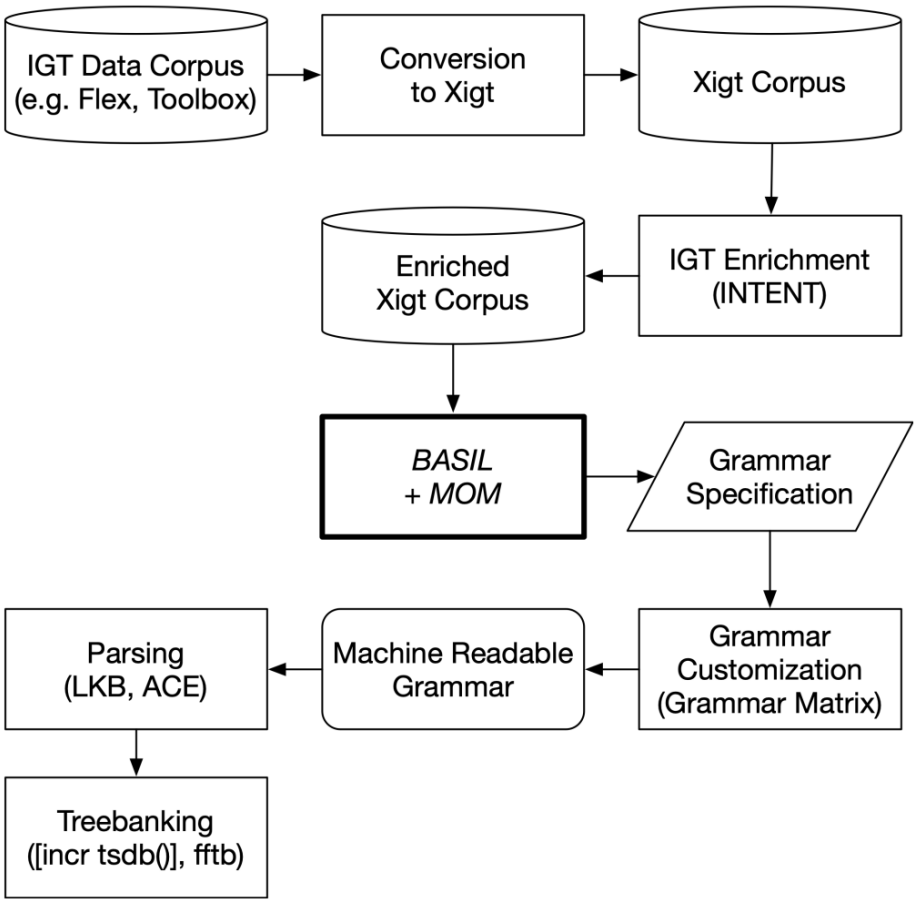

Figure 2.3: AGGREGATION pipeline (Howell, 2020, pg. 16)

```
<igt id="wmb-58">
    <tier id="p" type="phrases">
      <item id="p1">Jigama-yulu ny-a aliyulu</item>
    </tier>
    <tier id="m" type="morphemes" segmentation="w">
      <item id="m1" segmentation="w1">Jigama</item>
      <item id="m2" segmentation="w1">yulu</item>
      <item id="m3" segmentation="w2">ny</item>
      <item id="m4" segmentation="w2">a</item>
      <item id="m5" segmentation="w3">aliyulu</item>
    </tier>
    <tier id="g" type="glosses" alignment="m" segmentation="gw">
      <item id="g1" alignment="m1" segmentation="gw1">yam.III</item>
      <item id="g2" alignment="m2" segmentation="gw1">DU.ACC</item>
      <item id="g3" alignment="m3" segmentation="gw2">2.SG.A</item>
      <item id="g4" alignment="m4" segmentation="gw2">PST</item>
      <item id="g5" alignment="m5" segmentation="gw3">find</item>
    </tier>
    <tier id="gw-pos" type="pos" alignment="gw" data-creation-time="2019-04-12 09:54:33"
      data-method="project" data-provenance="INTENT2-2.0a6">
      <item id="gw-pos-3" alignment="gw3">VERB</item>
    </tier>
    <tier id="t" type="translations">
      <item id="t1">You found two yams.</item>
    </tier>
</igt>
```

Figure 2.4: Xigt instance, original IGT from Nordlinger (1998)

Like the multiple lines of an IGT instance, a Xigt instance has several *tiers*, a subset of which directly correspond to the basic IGT lines. The language line maps to the "phrases" tier (id="p"), but is also represented in the "words" tier (not shown) and the "morphemes" tier (id="m"). Similarly, there are several tiers that represent the information given in the gloss line. In Figure 2.4, one of the gloss tiers (id="g") gives the glosses for each morpheme. In each *item* of the morpheme glosses tier, the ID of the morpheme being glossed is given (e.g. alignment="m5") as well as the ID of the word gloss that the morpheme gloss comes from (e.g. segmentation="gw3"). The translation line is predictably mapped to the "translations" tier (id="t").

The INTENT package (INterlinear Text ENrichment Toolkit) adds several more tiers with inferred information to Xigt instances. It does this by parsing the translation line (which is assumed to be in English), aligning the translation line to the language line via the gloss line, and then projecting the syntactic structure from the parsed English sentence onto the language line. From this, tiers are added to each Xigt instance with part-of-speech tags and dependency parses for the original and English sentences. For instance, in Figure 2.4, the "pos" tier with id="gw-pos" gives partial part-of-speech tags for the Wambaya sentence.

2.3.2 BASIL

BASIL (Building Analyses from Syntactic Inference in Low-resource languages; Howell, 2020) encompasses the parts of the AGGREGATION pipeline that use Xigt data to infer a grammar specification. It includes much of the work on syntactic and morphological inference done up to the point of its establishment, some of it reimplemented or improved. For example, it incorporates modules used for inferring a language's word order, case system, and case frames. Howell also added functionality targeting several additional phenomena such as argument optionality, sentential negation, and coordination. MOM (see Section 2.3.3) plays an important role in BASIL, contributing the inference of the lexicon and morphotactics for nouns and verbs. Furthermore, Howell builds on previous work integrating the syntactic and morphological inference components (Zamaraeva et al., 2019a) by using the output of

MOM to infer the feature systems for several features, namely person, number, gender, tense, aspect, and mood, as well as to identify and define pronouns and auxiliaries.

2.3.3 MOM

A key component of BASIL is the MOM morphology inference module (Wax, 2014; Zamaraeva, 2016; Zamaraeva et al., 2017, 2019a), which infers the lexicon and morphotactics for nouns and verbs in the form of lexical classes (LCs) and position classes (PCs) (see Section 2.1.1). The initial algorithm for inferring PCs was implemented by Wax (2014), and though improvements to MOM have been made over time, the core of the algorithm that uses an input-based overlap score to merge morphological objects has remained largely unchanged. Eventually, it was extended to be used for inferring verb LCs by Zamaraeva et al. (2017), and then to infer the lexicon and morphotactics of nouns as well (Zamaraeva et al., 2019a). The inference processes occur separately for nouns and verbs, but the algorithm is the same for both parts of speech.

One piece of the morphotactic inference process is interpreting the feature information from the Xigt data expressing it in terms of MOM's internal representation of features and feature values. This is an important step because only the information that is successfully interpreted will make it through to the output grammar. It is especially important for this work, as it is also the primary information that my changes to MOM rely on. The first half of this procedure takes place before MOM is called, during the initial processing of the Xigt data. Each dataset has a configuration file called 'glosses.txt' that lists the unique morpheme-level glosses in the dataset that are only made up of feature grams (i.e. no stems), and this glosses file is used when the data is first read in to determine which affixes are made up of feature grams. If a gloss has a match on the glosses list, each gram is added to that affix's features list. Then, when the affixes are being initialized as LRTs at the start of MOM's inference process, every gram on an affix's features list is checked against MOM's list of known feature grams and if there is a match, that feature value is added to the affix's LRT. At this step, only a select number of features are considered. For verbs, MOM looks

for person, number, gender, tense, aspect, mood, and negation feature values, and for nouns, MOM only looks for case and number feature values.

If any feature values are found on a stem, they are recorded as zero-marked features. This is because MOM makes the assumption that stems cannot express features, and so any features that do show up on a stem are assumed to be realized via zero-inflection. When the feature set on an affix is being defined as described above, there is an additional check to see if that feature set matches one of the sets of zero-marked features. If there is a match and the values of those features are not the same, a non-inflecting LRT is added to the same PC as the affix for the zero-marked feature values. If no full set of zero-marked features matches, MOM will check for individual features that match (and have a different value) and add a non-inflecting LRT for each one.

The rest of this section will give an overview of MOM's inference algorithm. First, for each word in the data that has been tagged with the relevant part of speech, MOM initializes an LC for the stem and a PC for each affix, setting the input of each PC to be the LC or PC that comes directly before it in the order of affixation.[2] It is at this point that feature values are defined on the affix's LRT within the PC as described above. If a stem or affix is identical to a stem or affix that has already been seen, a new object will not be created and instead the input sequence will be routed through the existing object. Two stems are considered the same if they have the same orthography, predicate value, and (for verbs) have the same inferred case frame and transitivity. A stem's predicate value is derived from its gloss by stripping away anything that MOM recognizes as a feature gram.[3] Two affixes are considered the same if they have the same orthography, are both prefixes or both suffixes, and have all of the same feature values.[4]

---

[2]MOM assumes a fixed order of affixation, that being all prefixes before all suffixes, and starting with the affix closest to the stem and moving outwards for each type. So, the first affix is most commonly the prefix that attaches directly to the stem, but it could also be the closest suffix.

[3]With the exception of person, number, gender, and case grams on nouns.

[4]The fact that the feature values of affixes are compared is a (now) known bug. The intended behavior is that the glosses of affixes are compared. The difference here is that not every gram in a gloss is necessarily recognized by MOM as representing a feature value, so there is potential to erroneously combine affixes

After the LCs and PCs are initialized, MOM performs one round of PC merging followed by a configurable number of rounds of LC merging and PC merging (the default value is 1). In each round of merging, all of the LCs or PCs with an overlap score above a configurable threshold (the default is 0.2) are merged, with the pairs with the highest overlap scores getting merged first. After each individual merge, the overlap scores are updated for each pair. Once all of the possible merges happen for a given round, MOM switches to merging the other class type or finishes merging altogether, depending on how many rounds of merging are configured for.

For a pair of PCs, the overlap score is defined as the proportion of the PCs' collective inputs that are shared. For example, if *verb-pc1* has inputs {*verb1*, *verb2*, *verb-pc2*} and *verb-pc3* has inputs {*verb1*, *verb2*, *verb-pc4*}, they have four collective inputs {*verb1*, *verb2*, *verb-pc2*, *verb-pc4*} and two shared inputs {*verb1*, *verb2*}. So, their overlap is $\frac{2}{4} = 0.5$. Because LCs don't have inputs, their outputs, i.e. the set of PCs that have them as an input, are used to calculate their overlap. For both PCs and LCs, there are also a few basic compatibility checks that are run when calculating the overlap, and if any of them are not met, the overlap metric is automatically 0. For example, if two PCs are not both prefixes or not both suffixes, their overlap will be 0.

When two PCs are merged, all of the LRTs and inputs of one PC are added to the other. Then, the inputs of other PCs are updated accordingly, essentially swapping the name of the merged PC for the name of the PC it merged into, and the overlap scores are updated as necessary. The process is the same for LCs except the stems from one LC are added to the other instead of LRTs.

After all of the merging is complete, the result is a set of LCs and PCs that have (ideally) been generalized to the point that they can successfully parse words that did not appear in the training set but that are nonetheless valid words in the targeted language.

---

with different glosses that were not fully interpreted as a set of feature values.

2.3.4 Post-inference steps

After a grammar specification has been inferred, there are several pieces of software used to evaluate its performance. First, the Matrix generates a grammar from the specification which can then be compiled by the LKB (Linguistic Knowledge Building system; Copestake, 2002) or ACE (Answer Constraint Engine; Crysmann and Packard, 2012). After compilation, the LKB or ACE also completes the parsing step, but often an additional testsuite management tool like [incr tsdb()] (Oepen, 2001) or art[5] is used to call the LKB/ACE to process the entire set of evaluation sentences at once. For each sentence, these tools report whether the sentence was successfully parsed, and if so, how many parses were found. The tools can also be used for treebanking, where the user selects the correct parse out of the possible parses given.

2.3.5 Summary

This section gave an overview of the AGGREGATION system, including the MOM morphotactic inference module, which I attempt to improve in this thesis. In the next section, I lay out the concepts from theoretical morphology that I drew on to design my proposed improvements.

2.4 Morphology

This section provides the details of the concepts in theoretical morphology that motivate my changes to the morphotactic inference process. Section 2.4.1 will frame the use of position classes within the definition of canonical morphotactics and Section 2.4.2 will identify a few relevant typological parameters of inflectional morphology.

---

[5]http://sweaglesw.org/linguistics/libtsdb/art.html

2.4.1 Morphotactics

This thesis is primarily focused on the inference of canonical morphotactic systems. Bonami and Crysmann (2013) describe the canonical ideal of morphotactics as a situation where all morphemes are organized into position classes, where a position class is a collection of "morphs expressing different values for the same feature cluster in a single linear position, strictly ordered with respect to positions serving for the realisation of other features." (p. 28). In other words, in a perfectly canonical morphotactic system, the realization of all words (in a given part of speech) can be described as a fixed sequence of affixations where at each step the affix chosen expresses a unique set of feature values for a specific set of features that are not expressed in any other position. Additionally, no affix is used to express more than one set of feature values.

For example, Stump (1993, pg. 138) gives an analysis of Swahili verbs through a position class lens:

Negative - Subject Agreement - Tense - Relative - Object Agreement - root

In this framework, an affix expressing object agreement would be prefixed to the verb root first, followed by a relative prefix, a tense prefix, and so on (though there may not always be an affix for each slot). Any phenomena that deviate from this standard, then, fall under the umbrella of non-canonical morphotactics. One example of this again comes from Swahili, where the set of subject agreement affixes is largely the same as the set of object agreement affixes. For instance, the prefix *ni-* is used for first person singular agreement in both slots (Stump, 1993, pg. 143). Bonami and Crysmann (2013, pg. 28) call this "positional disambiguation" because the same morpheme expresses different (though related) properties depending on its position.

Given the very narrow definition of canonical morphotactics, it's easy to see how many languages can and do subvert this ideal. However, it provides a solid starting point for describing morphotactic systems generally and as such informs my approach to morphotactic inference. This is discussed further in Section 4.1.1.

2.4.2 Inflectional Morphology

Bickel and Nichols (2007) define inflection as the types of morphology "that are regularly responsive to the grammatical environment in which they are expressed" (pg. 169). For some types of inflection, the relevant grammatical environment is syntactic. This is the case for agreement, where for example a verb might be inflected differently based on the person, number, or gender of its argument(s). For others, the relevant grammatical environment is morphological. Bickel and Nichols give the example of Russian future tense, which is expressed periphrastically for imperfective verbs and synthetically for perfective verbs (2007, pg. 171). Here, the expression of future tense is conditioned on the aspect value in the morphological context.

There are several typological parameters of inflectional morphology that are useful for describing the morphotactics of a language, including fusion, flexivity, and exponence. These parameters are orthogonal to each other, so there are no theoretical restrictions on what parameter values can co-occur, even if some combinations are more represented than others in the world's languages. It's also worth mentioning that these parameters are meant describe *formatives* or *sets of formatives* as opposed to whole languages, due to the fact that most (if not all) inflectional systems have some variation with respect to each parameter, though it is generally possible to compare the degrees to which languages exhibit certain traits.[6]

Fusion deals with how tightly formatives are phonologically fused to their host. On one end of the fusion scale are *isolating* formatives, which are phonologically free words. In the middle are *concatenating* formatives, which are phonologically bound to words but are segmentable from them. Lastly, *nonlinear* formatives are realized as modifications to word stems and so are not segmentable from the rest of the word. As mentioned in Section 2.1.1, Matrix grammars assume that they are dealing with data where all inflection is represented as

[6]Bickel and Nichols (2007) separate the notions of 'formative' and 'affix', where a formative is any inflectional exponent, bound or free, and an affix is any bound unit. In this work, I am only concerned with bound formatives, and will simply refer to them as 'formatives' or 'affixes', with the slight difference being that a formative can have allomorphs while an affix has a specific surface form.

concatenative, regardless of the actual surface forms in the language. The AGGREGATION pipeline also operates under this assumption, though the data being used for inference does not necessarily adhere to this standard.

A formative is *flexive* if it displays lexical allomorphy, meaning it has several possible forms and its realization is determined by the lexical item it is attached to. On the other hand, *nonflexive* formatives have only a single form and any variation they do exhibit happens at the phonological level. An example of flexive morphology is the English plural suffix. For the largest class of nouns (with respect to the plural suffix), it is realized as '-s', as in *dogs*, but for a much smaller class, it is realized as '-en,' as in *children*. In this case, the classes that determine the plural suffix are irrelevant to any other part of English, but this is not always true cross-lingually. Among concatenative expressions of morphology, flexivity and nonflexivity are both common (Bickel and Nichols, 2007; pg. 186-187). The existence of flexivity is something that I try to account for in my implementation of the PC overlap function, and this will be detailed in Section 4.1.2.

Exponence has two possible values, *separative* and *cumulative*, describing whether a formative expresses one or multiple grammatical features. Nonflexive concatenative morphology more commonly co-occurs with separative exponence, and flexivity more commonly co-occurs with cumulative exponence (Bickel and Nichols, 2007; pg. 189). There are several groupings of grammatical features that have been documented as being realized cumulatively across languages, though the following is not a comprehensive list. It is common for different combinations of person, number, and gender to be expressed cumulatively (Kibort and Corbett, 2008). Within that group, person and number are most commonly combined (Bickel and Nichols, 2007, pg. 228; Plank, 1999, pg. 292). Relatedly, number and case are another pair of features that have been observed to be cumulated, especially in Indo-European languages (Bickel and Nichols, 2007, pg. 188; Bickel and Nichols, 2013; Plank, 1999, pg. 302). Tense, aspect, and mood, or subsets thereof, are regularly realized together, in part because it is often difficult to make a clear distinction between the three semantically (Dahl and Velupillai, 2013; Kibort, 2008). For a more in-depth look at the typology of specific features and the

realizational splits within them, see Plank, 1999. In this thesis, I attempt to recognize instances of cumulative exponence by looking for occurrences of these selected feature groups, and this will be discussed further in Section 4.2.

2.5 Summary

This chapter provided the background necessary for exploring a feature-based approach to morphotactic inference. It described how a language's morphotactics are represented in Matrix grammars and explained how MOM uses IGT data to infer this portion of the grammar in its current state. Additionally, it detailed the relevant concepts and typological patterns from the inflectional morphology literature that informed my implementation. The next chapter will explain the data and evaluation methods employed in the development and testing processes.

Chapter 3
EVALUATION METHODOLOGY

This chapter will describe the data and evaluation methods I used to develop and then test my implementation of a feature-based morphotactics inference process. Over the course of the work done for this thesis, I maintained a data-driven workflow, wherein a portion of the data was used to inform the decisions I made throughout the development process, and the remainder was held out until the changes were finalized in order to measure the broad impact of the changes. This will be discussed in Section 3.1. Section 3.2 will elaborate on the method of evaluation and the metrics used to quantify the results as well as a qualitative methodology for exploring the resulting grammars.

3.1 Data

The data used in this thesis is comprised of six datasets, each containing IGT data from a different language. The datasets were selected from the 36 in the AGGREGATION project's data repository. Each dataset was produced by a linguist or linguists who were generous enough to allow their work to be used in the context of this project. It is crucial to acknowledge that this work (and the AGGREGATION project as a whole) would not be possible without the large amount of manual work done by linguists and language consultants to produce the data.

The amount of data in the datasets varies greatly, ranging from less than 200 to over 10,000 IGT instances, which typically contain a single sentence each. The datasets, as a result of each one representing a different language and being produced for different purposes, also exhibit a range of values for various dataset characteristics such as the average IGT length in words or in morphemes. When picking datasets from the repository, I looked at three

Language [iso]	Number of IGT	Number of Words	Morphemes per Word	Type Stems Ratio	Allomorph Ratio
Lezgi [lez]	1,531	18,724	2.17	3.05	1.48
Manipuri [mni]	1,785	15,887	1.51	1.80	1.39
Wambaya [wmb]	797	2,933	1.54	1.86	1.35
Wakhi [wbl]	770	3,219	1.79	1.86	1.17
South Efate [erk]	2,240	24,774	1.4	2.55	1.38
Hiaki [yaq]	10,971	77,894	1.19	1.44	3.69

Table 3.1: Dataset characteristics of development and test datasets (Liu, 2025)

morphologically-relevant characteristics and tried to select the datasets that best represented the ranges of their values. Specifically, I looked at the average number of morphemes per word, the *Type Stems Ratio*, and the *Allomorph Ratio* of each dataset, as calculated by Liu (2025). The average number of morphemes per word is calculated by dividing the total number of morphemes in the dataset by the total number of words.[1] The *Type Stems Ratio* is calculated as the number of unique word forms divided by the number of unique stems. This equates to the number of different ways a stem in the data is inflected on average. The *Allomorph Ratio* is calculated as the number of unique morpheme forms divided by the number of unique morpheme glosses. This equates to the number ways each gloss in the data is orthographically realized on average. All of these values, as well as the total number of IGT and words in each dataset, can be seen in Table 3.1. I also considered the language family and location of each language (as reported by Glottolog (Hammarström et al., 2025)) to ensure a reasonable level of overall typological and areal diversity.

After selecting the six datasets, I partitioned them into development and test groups, maintaining dataset diversity within each half as best as I could. As is common practice,

[1]Liu (2025) does not explicitly report this metric, but does report the *Average IGT Length in Words* and the *Average IGT Length in Morphemes*. Dividing the latter by the former is equivalent to dividing the total number of morphemes by the total number of words, yielding the desired metric.

the development datasets were utilized throughout the development process while the test datasets were held out, meaning that they were only used for the final evaluation. Because there is no language overlap between the development and test sets, the testing phase serves to evaluate the cross-lingual efficacy of the changes made. Ultimately, the languages represented in the development set are Manipuri[2] [mni] (Chelliah, 2019), a Sino-Tibetan language of northeast India, Lezgi [lez] (Clifton et al., 2024), a Nakh-Daghestanian language of the North Caucasus, and Wambaya [wmb] (Nordlinger, 1998), a Mirndi language of northern Australia. The languages represented in the test set are Wakhi [wbl] (Obrtelová, 2019), an Indo-European language of northeast Afghanistan, South Efate [erk] (Thieberger, 2006), an Austronesian language of Vanuatu, and Hiaki [yaq] (Harley, 2019), an Uzo-Aztecan language of northwest Mexico.

3.2 Evaluation

To measure the performance of the system with and without the changes made to MOM, the AGGREGATION pipeline is run end-to-end and several outcomes relevant to morphotactic inference are analyzed. First, given a certain train/evaluation split of a dataset (more details in Section 3.2.1), the training split is passed to BASIL, which uses it to infer a choices file. From there, the Matrix generates a grammar from the choices file and ACE compiles the grammar. Lastly, ACE attempts to parse each sentence in the evaluation split using the compiled grammar. From the outputs, namely the inferred choices file and the parsing results, various metrics are computed to estimate their quality.

The rest of this section will elaborate on some of the details of this process. Section 3.2.1 will discuss the use of cross-validation. Section 3.2.2 will describe the metrics related to the parsing performance of resulting grammars, and Section 3.2.3 will describe the metrics related to grammar complexity. Section 3.2.4 will discuss the steps taken to analyze the input graphs of grammars.

---

[2]also referred to as Meitei or Meithei

3.2.1 Cross-validation

Following several of the previous AGGREGATION papers (e.g. Conrad, 2021 and Lin, 2023), I employ 10-fold cross-validation in my evaluation. This involves splitting a dataset into 10 equally-sized folds and running the system 10 times, each time using a different fold as the evaluation set and the rest of the folds as the training set, and averaging the values of the resulting evaluation metrics. Also like past papers, I use the folds created by Howell (2020) so that the results are more comparable across studies. This approach effectively allows for a system to be evaluated 10 times with the same amount of data, increasing the stability of the results and the likelihood that differences in performance (when compared to the baseline) reflect the quality of the changes made to the system rather than the idiosyncrasies of some evaluation set.

3.2.2 Grammar Performance Metrics

To quantify the parsing performance of the inferred grammars, I look at a subset of the metrics examined by Howell 2020). These metrics, especially parse coverage and the average number of parses per parsed sentence, are the most common way that inferred Matrix grammars (and Matrix grammars in general) are evaluated (e.g. see Conrad, 2021; Dods, 2022; Lin, 2023). By directly measuring how well a grammar is able to parse sentences, we are able to get a broad idea of how well it functions as a model of a language.

The first metric is *lexical coverage*, which is defined as the proportion of sentences for which a grammar is able to produce an analysis for each word. This is the performance metric most directly relevant to this study, as a grammar's lexicon and morphotactics entirely determine whether a word is analyzed. The second metric is *parse coverage*, defined as the proportion of sentences for which a grammar yields at least one complete analysis. This is not as relevant of a metric for this work, especially given that lexical coverage is a necessary condition of a sentence being parsed. However, it still possible that a change to the morphotactic inference process could affect overall coverage without changing the amount

of lexical coverage. For example, a change that impacts whether agreement features are realized on a certain affix or what agreement feature values can be expressed on that affix may not prevent a word containing that affix from being recognized, but it could prevent an analysis at the sentence level due to incompatible agreement markings. For these first two metrics, higher values are generally more desirable because the data only includes examples of grammatical sentences (though there are some cases where a sentence is parsed with only incorrect analyses).

The changes presented in this thesis are generally focused on the quality of inferred morphotactic systems, i.e. how realistic their structure is as opposed to their raw ability to produce analyses of words, and because of this, the coverage metrics are not necessarily expected to change in a meaningful way and therefore may not provide much insight into the effects of said changes. Additionally, because lexical coverage (and therefore sentence coverage) depends on parsing all of the words in a sentence, increasing the noun or verb lexical coverage would not guarantee an increase in the main metrics, especially because there are a number of word types that are not yet covered by any part of the grammar inference process. Furthermore, increasing the number of affixes with features (Section 4.3), i.e. the number of constraints, has the potential to decrease the amount of words and therefore sentences parsed, but this would be good loss, i.e. a reduction in undesirable analyses. Despite all of this, it is still important to track and report these metrics as points of comparison to previous studies and to make sure that no changes drastically harm grammars' parsing performance.

The third and fourth metrics have to do with ambiguity. *Ambiguity* is the average number of parses per sentence for the sentences with at least one parse. *Maximum Results* is the number of parses for the single sentence in the evaluation set with the most parses. For these metrics, a lower number is more desirable than a higher one.

3.2.3 Grammar Complexity Metrics

The second category of metrics I am concerned with has to do with grammar complexity. Given that no grammar, much less an automatically inferred one, is comprehensive, a linguist

using the AGGREGATION pipeline will want to continue developing their grammar after its initial generation. Because of this, it is desirable for the system to produce grammars that not only perform well in terms of the traditional parsing metrics, but that are also concise and well-organized. The closer an inferred grammar looks to one that a linguist would create, the less time linguists will have to spend understanding the grammar and fixing confusing or incorrect analyses before they are able to expand upon it in novel directions. To assess grammars in this way, I take advantage of several metrics defined by Liu (2025).

The *number of LCs* is simply the number of noun or verb lexical classes in the lexicon, and the *number of PCs* is the number of noun or verb position classes in the morphology section of the grammar. For both of these metrics, the numbers are reported separately for nouns and verbs. In its current state, MOM greatly under-combines LCs and PCs. For example, the inferred Chintang grammars in Zamaraeva's (2016) case study had at the fewest 48 PCs, while the hypothesized number of PCs from the literature was only 13. However, merging LCs and PCs for the sake of having fewer of them is no better for modeling a language well. So, while I predict that my proposed changes will result in fewer PCs and LCs, it is also important to meaningfully evaluate their quality, and this will be discussed in the next section.

There are a couple of factors that affect these metrics but that are not directly relevant to morphotactic inference in MOM. The first factor is the inference of pronouns. This occurs outside of the MOM process, but because pronouns are nouns, the *number of noun LCs* metric includes the number of pronouns. However, pronouns use the same PCs as regular nouns, so they do not affect the number of noun PCs. The inference of auxiliaries also happens outside of MOM, however unlike pronouns, the number of auxiliary LCs is not included in the *number of verb LCs*.[3] Also unlike pronouns, the number of auxiliary PCs does impact the *number of verb PCs* because auxiliaries have their own set of PCs separate

[3]This is because the metrics are calculated by looking at the names of LCs in a choices file. Pronouns follow the naming convention of other nouns, e.g. 'nounX', and so are included in the count, while auxiliaries do not, e.g. 'auxX' vs 'verbX'.

from the regular verb PCs. Because the inference of pronouns and auxiliaries happens outside of MOM, my code changes do not affect those processes and the contributions of pronouns and auxiliaries to the number of noun LCs and verb PCs stay constant.

3.2.4 Input Graph Analysis

The last component of grammar evaluation that I employ is an analysis of the noun and verb input graphs (as described in Section 2.1.1). The purpose of this analysis is to qualitatively compare grammars (though I still use some quantitative measures) to evaluate the ways and the extent to which each grammar posits a realistic model of morphotactics. With this evaluation, I don't make direct comparisons to the established knowledge about any given language, but I do make comparisons to what is attested in the training data or what is generally reasonable for any language to indicate which input graphs are likely to be more accurate representations of their languages. When comparing grammars produced by two different versions of the code, I always look at ones that were inferred from the same set of training data in order to be able to make direct comparisons.[4]

To facilitate this analysis, I use the MOM visualization tool[5] (Lepp et al., 2019), which generates interactive graph visualizations of a grammar's noun and verb input graphs. A screenshot of the tool is given in Figure 3.1. In the visualization, the purple boxes with circles represent LCs and the green boxes with triangles represent suffix PCs (prefix PCs are blue with a triangle pointing the opposite way, but this particular grammar does not have any noun prefixes). The size of each box indicates the number of stems/affixes in the LC/PC. The left side of the screen has an expandable tab for each LC that lists each stem and its corresponding predicate, as well as one for each PC that lists each affix and its corresponding feature values.

---

[4]The grammar inference process is not rigorously deterministic, but it is to a large enough degree that grammars inferred from and evaluated on the same train/evaluation split of the data are able to be more or less directly compared.

[5]http://uakari2.ling.washington.edu/aggregation/mom-gui/

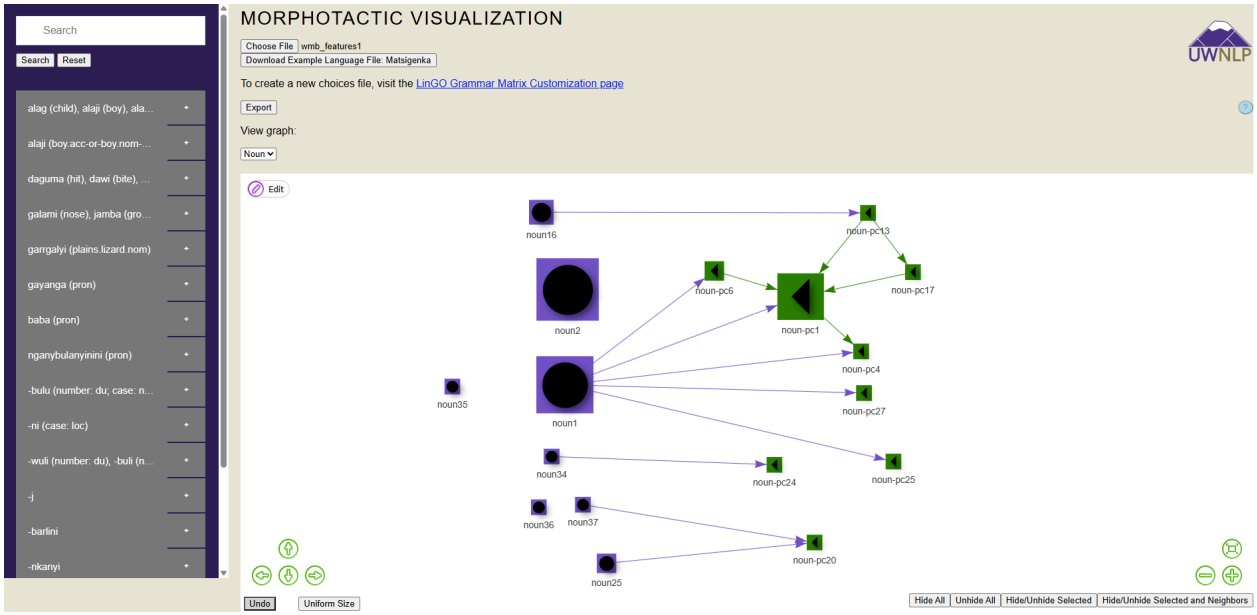

Figure 3.1: Screenshot from the MOM visualization tool

When comparing the input graphs of two grammars, I make note of any significant differences between the two. Due to the fact that my changes to MOM are based around feature information, the main difference between two grammars usually has to do with how the affixes with features are organized into PCs. By this I mean that in the non-baseline grammars, affixes with some set of features are much more often in a PC with other affixes with the same set of features and they are less often in the same PC as affixes with non-overlapping sets of features.

There is also frequently a difference in the longest path through the input graph and how many PCs with features are on that path. The sequence of PCs that makes up the longest path serves as an upper bound on how many affixes a word can have, which is also often an upper bound on how many features a word can inflect for, though there are sometimes shorter paths with more feature-having affixes. However, the longest path is not necessarily the true maximum number of affixes or features on a word according to the grammar because the feature values on the affixes must all agree. Despite this, paths with many more PCs

than there are affixes on any given word in the data are still a problem from the broader perspective of inferring high-quality grammars that are interpretable by linguists, and for this reason they can be a helpful point of comparison between two grammars.

Additionally, when making comparisons between input graphs and the word forms that exist in the training data, I look at the data that the input graphs were actually inferred from, i.e. the training data after it is initially processed by MOM as described in Section 2.3.3. Because there are various reasons why there might be differences between the original data and the MOM's internal representation of it, e.g. MOM only preserves a specific set of features, using the processed version of the data allows me to better isolate the consequences of the morphotactic inference algorithm for my analysis.

3.3 Summary

This chapter detailed the data used in this thesis for developing and testing the changes to MOM as well as the method of evaluation. It also discussed the the metrics used to quantify the results and the approach taken to analyze the input graphs of inferred grammars. The next chapter will describe the motivations for and implementation of said changes.

Chapter 4

METHODOLOGY

This chapter will discuss my changes to MOM and the motivations behind them as it relates to the research question of this thesis, which asks how feature information can be used to improve morphotactic inference. Section 4.1 will outline the implementation of a new version of the overlap function that drives the merging of position classes (PCs) in MOM. Section 4.2 will describe an additional modification to the overlap function to make considerations for specific features and their relationship to one another. These two sections represent my primary hypothesis with respect to the research question, which is that morphotactic inference can be improved by using grammatical feature information to (1) model canonical morphotactics (as described in Section 2.4.1) and (2) anticipate instances of common typological patterns. Section 4.3 will discuss adding the ability to manually map gloss grams to features that MOM will recognize.

Throughout the chapter, I will use a small set of abstract words to illustrate the intended effect of each component of the implementation. Consider the following data in (2):

(2) a. stemA-abc
nounA-NOM

b. stemA-xyz
nounA-NOM.PL

c. stemA-def
nounA-ACC

d. stemB-abc
nounB-NOM

e. stemC-ghi
nounC-NOM

Given this dataset, MOM would initialize three LCs (called *noun1*, *noun2*, and *noun3* for the *stemA*, *stemB*, and *stemC*, respectively) and four PCs. The PC for *-abc* (called *noun-pc1*) would have two inputs (LCs *noun1* and *noun2*) and the PCs for *-xyz*, *-def*, and *-ghi* (called *noun-pc2*, *noun-pc3*, and *noun-pc4*, respectively) would have one input each (LCs *noun1*, *noun1*, and *noun3*, respectively).

4.1 Feature-based Overlap Function

This section introduces my implementation of the PC overlap function. The current process of merging PCs (and LCs) in MOM is dictated by an overlap function that uses the number of shared inputs (or outputs) between pairs of PCs (or LCs) to determine which pairs should be merged and in what order. Accordingly, the most direct way to reorient MOM towards creating canonical position classes is to modify the overlap function to favor shared features over shared inputs.

Up until this point, MOM's overlap function for PCs and LCs has been essentially the same, save for a few basic compatibility checks for things like transitivity alignment between LCs and cycle prevention among PCs. However, because MOM assumes that stem morphemes do not have any features (see Section 2.3.3), my alternate overlap function that incorporates feature information is only implemented for PCs and the overlap metric for LCs remains based solely on the number of shared outputs. The baseline overlap function is shown in Algorithm 1 and I define my feature-based overlap function for PCs in Algorithm 2.[1]

Algorithm 1 Input-based overlap function

```
1: function OVERLAP(pc1, pc2)
2:     input_overlap ← |pc1.inputs ∩ pc2.inputs|/|pc1.inputs ∪ pc2.inputs|
3:     return input_overlap
```

[1]For brevity, these definitions do not include the compatibility checks that occur at the start of the algorithm.

Algorithm 2 Feature-based overlap function

```
1: function OVERLAP(pc1, pc2)
2:     input_overlap ← |pc1.inputs ∩ pc2.inputs|/|pc1.inputs ∪ pc2.inputs|
3:     if |pc1.feats| = 0 ∨ |pc2.feats| = 0 then          ▷ input fallback
4:         return input_overlap
5:     feat_overlap ← |pc1.feats ∩ pc2.feats|/|pc1.feats ∪ pc2.feats|
6:     if input_overlap > 0 then                          ▷ input overlap prerequisite
7:         return feat_overlap
8:     else
9:         return 0
```

4.1.1 Basic feature overlap

In line with the discussion of canonical morphotactics (Section 2.4.1), the feature-based overlap function works under the assumption that (with some exceptions) the desired end result is a set of PCs where each PC contains affixes that express values of the same feature or set of features. To facilitate this, the core component of the overlap metric is feature overlap, which is calculated in the same way that input overlap is, as the simple proportion of shared features. This value is calculated in line 5 in the algorithm, where $pc.feats$ is the set of all features (not feature values) expressed by LRTs in a PC. For example, consider the affixes *-abc* and *-xyz* from (2). For *noun-pc1*, the PC containing *-abc*, $|$*noun-pc1*$.feats|$ is 1 because its affixes (in this case only one affix) express case. Similarly, $|$*noun-pc2*$.feats|$ is 2 because *-xyz* expresses case and number. Therefore, because *noun-pc1* and *noun-pc2* share case but not number, $feat\_overlap = \frac{1}{1+2-1} = 0.5$.

4.1.2 Input overlap prerequisite

The main departure from the ideal of a canonical position class that I attempt to account for is the existence of flexivity, or lexical allomorphy, when there is a group of affixes that

express the same feature value(s) but have different orthographic forms, and the specific form that any given word takes is lexically conditioned. As discussed in Section 2.4.2, this non-canonical phenomenon is commonly observed in the world's languages, so it would be beneficial to be able to accurately predict when instances of it exist in a language from its data. Even though lexical allomorphs may fill the same affix slot in the sense that they inflect the same feature in the same relative word position, it would be an error to place them in the same position class (for this implementation of position-class morphology) because it would cause the resulting grammar to produce analyses for invalid words. For example, looking at the data in (2), suppose the allomorphy of the nominative case suffix is lexically conditioned such that the stem *stemA* selects *-abc* as in (2a) and the stem *stemC* selects *-ghi* as in (2e). If feature overlap was the sole factor in the overlap metric, *-abc* and *-ghi* would end up in the same position class because they have complete feature overlap, which in turn would allow for the erroneous analyses of **stemA-ghi* and **stemC-abc*. To counteract this, I retain the input overlap component from the original overlap metric and use it as a prerequisite for having an overall overlap score greater than 0 (line 6). Ideally, this will help to maintain separation between parts of the inflection graph that exists due to lexical allomorphy in real scenarios as well. However, this is not a perfect solution. Sticking with the example, suppose that there is only one suffix for accusative case, i.e. *stemC-def* is also a word. In that case, the PC containing *-def* would be initialized with both *noun1* and *noun3* as inputs, enabling it to merge with the PC for *-abc* and the PC for *-ghi* and again opening an avenue for the disallowed **stemA-ghi* and **stemC-abc*.

4.1.3 Input fallback

The final thing that I attempt to handle in the overlap function is affixes that do not express any features. There are several reasons why an affix might not correspond to any feature at this point in the process. It could be that it is glossed in the IGT with a gloss that MOM does not recognize or with features that MOM does not consider. It could also be that it is inconsistently glossed or not glossed at all in the IGT. In reality, very few collections of

IGT data are thoroughly glossed, and as such, these types of situations are expected for the data input to the AGGREGATION pipeline. So, in order to not exclude a large number of affixes from the merging process, I fall back to using input overlap when either PC expresses no features, because otherwise the overlap would always be 0 (lines 3-4). This decision is not strictly linguistically motivated but it is instead more of a practical consideration given this known source of noise/information loss in the system. That being said, I think there is still potential for this adjustment to result in meaningful PC merges by "sweeping up" unglossed affixes that in actuality express the same feature(s) as some of the glossed affixes. For example, in the Manipuri data, there are multiple instances of the locative suffix *-tə*, only some of which have a locative gloss (Chelliah, 1997, pg. 243; Chelliah, 2019). In this case, since the affixes are the same morpheme, there should be, with a large enough dataset, instances of the glossed affix in the same environment(s) as the unglossed affix, and therefore there should be some shared inputs between the PCs of the two versions of the affix, so the input overlap fallback would enable their PCs to be merged. This distribution-based intuition should also apply more generally, such as to cases where an unglossed affix with no glossed counterpart is merged with a PC whose affixes express the same feature as it, or where a feature is not glossed for at all but the affixes that express its values exist in the data and are grouped together.

4.1.4 Unlimited merging rounds

Aside from the overlap function itself, I made one change to a relevant hyperparameter during the development process, the number of merging rounds. When MOM is run with its default configuration, it completes one round of LC merging followed by PC merging after the initial pass at PC merging, as described in Section 2.3.3. For my experiments, I configured MOM to run for "unlimited" rounds, which with the development datasets equated to 2-3 rounds of LC and PC merging. In each round of merging, merging happens until there is no pair of objects with an overlap above the configured threshold, and the only way for more merging to happen in a subsequent round is if some amount of merging happens for the other type

of object (LCs for PCs and vice versa) that changes the inputs/outputs enough to result in the overlap score for some pair being above the threshold. So, once there is one round where either no LC merging or no PC merging happens, no future merges will happen and the process can move on. Like the addition of the input overlap fallback, this change is directly an effort to reduce the final number of LCs and PCs, and the overlap function as implemented should be particular enough to prevent catastrophic over-merging because for PCs with features, there must be at least one shared input and one shared feature, and for pairs where one or the other PC has no features, at least one shared input is required for an overlap greater than 0.

4.2 Feature-specific Considerations

From the initial implementation of the feature-based overlap function, I made an additional change to attempt to recognize instances of cumulation based on groups of features that are known to be commonly realized together. The revised implementation is given in Algorithm 3. This version of the overlap function adds a subroutine BOOST_OVERLAP (lines 1-6) that is called several times in the main overlap function (lines 11-15). Along with the feature sets from the two PCs, it takes an additional argument that is also a set of features. If both PC feature sets share at least one feature with the extra feature set and one of the PC feature sets shares at least two, then all of the features from the extra feature set will be added to the PC feature sets.[2] This will inflate the feature overlap for the pair of PCs, effectively prioritizing their merging.

Like flexivity, cumulation is a well-established aspect of inflectional morphology, so being able to predict its occurrence based on the data has the potential to improve the inferred morphotactics. Additionally, certain sets of features are known to be commonly expressed together on nouns or verbs, as explained in Section 2.4.2, and these are the feature sets

[2]In the actual implementation, the set modifications are done on copies of the PC feature sets so that the overlap score is boosted but the features that are reported to be represented do not change.

Algorithm 3 Feature-based overlap function with feature-specific considerations

```
1: function BOOST_OVERLAP(feats1, feats2, feat_set)
2:     if |feats1 ∩ feat_set| ≥ 1 ∧ |feats2 ∩ feat_set| ≥ 1 then
3:         if |feats1 ∩ feat_set| + |feats2 ∩ feat_set| ≥ 3 then
4:             feats1 ← feats1 ∪ feat_set
5:             feats2 ← feats2 ∪ feat_set
6:     return feats1, feats2
7: function OVERLAP(pc1, pc2)
8:     input_overlap ← |pc1.inputs ∩ pc2.inputs|/|pc1.inputs ∪ pc2.inputs|
9:     if |pc1.feats| = 0 ∨ |pc2.feats| = 0 then                ▷ input fallback
10:        return input_overlap
11:    if pc1.pos_tag = 'verb' then                             ▷ boosted feature groupings
12:        pc1.feats, pc2.feats ← BOOST_OVERLAP(pc1.feats, pc2.feats, {'P', 'N', 'G'})
13:        pc1.feats, pc2.feats ← BOOST_OVERLAP(pc1.feats, pc2.feats, {'T', 'A', 'M'})
14:    else
15:        pc1.feats, pc2.feats ← BOOST_OVERLAP(pc1.feats, pc2.feats, {'N', 'case'})
16:    feat_overlap ← |pc1.feats ∩ pc2.feats|/|pc1.feats ∪ pc2.feats|
17:    if input_overlap > 0 then                                ▷ input overlap prerequisite
18:        return feat_overlap
19:    else
20:        return 0
```

passed as the extra argument of BOOST_OVERLAP.[3]

As mentioned, this 'feature boosting' change only requires that one of the two PCs have more than one feature from a feature group. This is beneficial, for instance, in cases where some feature value is not explicitly marked and correspondingly not glossed. Going back to the example in (2), suppose that any noun whose suffix does not express number is implicitly singular or is underspecified for number (and as a result, no number information is given in the gloss line). In that scenario, affixes with number and case probably belong in the same PC as affixes with only case, and this adjustment to the overlap function would prioritize that. For example, instead of the overlap between *noun-pc1* (*-abc*, nominative case) and *noun-pc2* (*-xyz*, nominative case and plural number) being $\frac{1}{2} = 0.5$, it would be $\frac{2}{2} = 1.0$. For this small amount of data, the desired merges would still happen without certain overlap scores being boosted, but this would not always be the case in a more realistic scenario. Even if having one shared feature is enough to have an overlap score above the threshold for merging, having a lower score means that there is a greater possibility of other merges happening first and preventing the more desired merge from taking place.[4]

4.3 Mapping Gloss Grams to Features

Within the MOM process, there are regularly various challenges that arise along the path from a gloss existing in the data to being realized as a feature or set of features in MOM's internal representation. As a result, MOM's ability to do feature-based morphotactic inference is potentially limited, depending on what misalignments exist for each particular dataset. One predictable source of error is that the gloss grams used in the data often don't perfectly align with the set of grams expected by MOM, even with MOM checking several possible grams for most feature values. For example, the Wambaya dataset (Nordlinger, 1998) uses

[3]In the algorithm definition, 'P', 'N', and 'G' stand for 'person', 'number', and 'gender', respectively, and 'T', 'A', and 'M' stand for 'tense', 'aspect', and 'mood', respectively.

[4]When a pair of PCs has a non-zero overlap score at one point and 0 overlap later, it is because merging the PCs at that point would create a cycle in the input graph. See Section 5.3.5 for a more in-depth discussion.

```
GPL    pl
IMP    imp
INTEND intt
PERMIT perm
PRO    prosp
PROHB  proh
START  incep
```

Figure 4.1: Manual feature mapping for the Manipuri dataset

the gram 'NF' to indicate nonfuture tense. However, MOM recognizes it as marking nonfinite form, which in this case results in a featureless affix in the grammar because MOM does not specify form on verbs. Based on my development data, it is more common for a gloss gram to not be recognized at all than for it to be mismapped, but in most cases the end result is the same, an affix that appears to express no features. Additionally, it is sometimes the case that a gloss is missing from the dataset's glosses list (see Section 2.3.2), meaning that its grams will be ignored by MOM.

To help counteract this source of noise, I implemented a simple system for directly mapping gloss grams onto MOM's internal features. Before running the AGGREGATION pipeline, users can create a configuration file and on each line define a link between a specific gloss gram and MOM's corresponding feature value name. In the main configuration file, the field 'gram map' can be set to the path of the file with the feature mapping. An example configuration for the Manipuri dataset is given in Figure 4.1.

Each line of the configuration is whitespace-separated with the gloss from the data on the left and MOM's internal feature value on the right. For the Manipuri dataset, 'GPL' is mapped to plural number with 'pl', 'IMP' is mapped to imperative mood with 'imp' (it

was initially mapped to imperfective aspect), and so on. MOM's feature values are found in FeatDict.py in the xigtifiedtoolbox library. The file defines a dictionary for each feature where the keys are possible gloss grams and the values are the grams that MOM uses. For example, in the tense dictionary the keys 'past' and 'pst' both have the value 'pst'. There are also dictionaries for combined gloss grams, i.e. glosses not broken up by periods. For example, the dictionary for person and number maps '1sg' to the tuple ('1st', 'sg'). To map a gloss gram to one of these tuples of values in the gram map file, users can simply list each value in the tuple, e.g. for the gloss gram '1sing', the line would be "1sing 1st sg."

For my datasets, I used the following procedure to create the gram map for each dataset. First, using a script that I wrote to aid in the process, I constructed a list of the unique morpheme glosses and a list of the unique gloss grams in a dataset. For the second list, I made the simplifying assumption that grams are stylized with all uppercase letters, and this seemed to be true for all of my datasets. Next, using the list of morpheme glosses, I added any that were missing from the dataset's glosses file to it. Then for any of the gloss grams from the second list that did not match any of MOM's feature values or that would be incorrectly interpreted by MOM, I added a line to the dataset's gram map to pair it with the intended feature value. When the intended value of a gloss gram was not obvious from its form or usage in the data, I consulted a grammar for the language. In each case where this was necessary, I directly used the grammar's list of abbreviations because either the data came from that grammar originally or the data and the grammar were published by the same author.[5]

4.4 Summary

This chapter described the experimental changes made to MOM that were motivated by the idea of incorporating feature information into the morphotactic inference process. I laid out a new implementation of MOM's PC overlap function that attempts to use feature

[5]The Wambaya, Wakhi, and South Efate data came from grammars (Nordlinger, 1998; Obrtelová, 2019; Thieberger, 2006), and the publisher of the Manipuri data also published the grammar (Chelliah, 1997).

information to model canonical morphotactics as well as flexivity and cumulative exponence, while also making practical considerations based on the data that the grammar inference pipeline typically works with. Additionally, I introduced a way for users to directly map feature glosses to the corresponding feature values internal to MOM. The next chapter will describe the results of the experiments run with these changes for the development and test datasets.

Chapter 5

RESULTS

This chapter presents the results of adding the experimental changes to the MOM module. The effects of these changes were quantified by running the end-to-end AGGREGATION pipeline and evaluating the inferred grammars produced by it with the performance- and complexity-based metrics described in Section 3.2. Also as described in Section 3.2, I employed cross-validation to get 10 grammars for each dataset, and the values shown here are the averages of the metrics over the 10 runs.[1]

In each set of results, the performance and complexity metrics are reported for several stages of the development process.[2] Each stage represents the culmination of the changes up to and including the named change. For example, the version of the code that produced the results for the "Input Overlap Prerequisite" stage included the initial feature-based overlap score and the requirement that input overlap be greater than 0 in order to have an overall overlap score above 0. Additionally, the results include an iteration of the code with a few bug fixes but no experimental changes. I discovered these bugs during the development process and fixed them before any non-baseline results were recorded.[3] This set of results represents the true baseline for this study, and the initial baseline is only given for the sake of consistency with previous studies. As such, it will not be considered in the analysis of the results.

I report the results at each stage because, as was shown in Chapter 4, each change was

[1]The code to replicate these results is available here: https://git.ling.washington.edu/agg/repro/ijs-2026

[2]For the test datasets, i.e. the held-out data, these stages were simulated after development was completed.

[3]The first bug had to do with updating PC overlap values and was preventing any additional merging from happening after the first pass at LC merging. The second had to do with zero-marked features on word stems not being properly recognized.

independently motivated, whether it be by some aspect of morphological typology or by the type of data used for grammar inference. Because of this, any potential difference in behavior between stages can roughly be analyzed in terms of how successfully the most recent change accounts for the pattern or problem it attempts to handle. The particular sequence of changes mirrors my development process but otherwise does not have a specific motivation, apart from the fact that some changes are dependent on earlier changes, e.g. several changes are dependent on having the initial feature-based overlap score.

The results for the development and test datasets are presented in Sections 5.1 and 5.2, respectively. For each dataset, I make observations about the metrics as well as do some analysis of the resulting input graphs to identify if and how the grammars are likely to be more usable for the user-linguist. For the input graph analysis portions, I look at the training data from the first train/evaluation split and the baseline and final grammars inferred from it and make note of any significant differences between the two input graphs. Section 5.3 analyzes the overall effects on the grammar performance and complexity metrics as well as discusses some of the common patterns seen in the results in more detail.

5.1 Results for Development Datasets

5.1.1 Wambaya

The grammar performance metrics for Wambaya are given in Table 5.1. Over the different development stages, the performance metrics remained stable, and apart from slight deviations in the "Feature-based Overlap" and "Input Overlap Prerequisite" runs, coverage, ambiguity, and maximum results were constant. Lexical coverage was also mostly constant, with the final grammars showing a slight bump over the baseline, though the difference is relatively small.

The complexity metrics for Wambaya are given in Table 5.2. The "Feature-based Overlap" run is the only one whose grammars greatly deviate from the baseline, especially in terms of the noun metrics, dropping down from 24.1 to 7.3 and from 20.3 to 8.5 for the

Development Phase	Lexical Coverage	Coverage	Ambiguity	Max Results
Initial Baseline	7.82 %	1.83 %	26.33	356
Bug Fixes (Baseline)	8.19 %	1.83 %	26.33	356
Feature-based Overlap	8.31 %	1.59 %	30.08	356
Input Overlap Prerequisite	8.19 %	1.59 %	30.08	356
Input Overlap Fallback	8.44 %	1.83 %	26.33	356
Unlimited Merging Rounds	8.44 %	1.83 %	26.33	356
Boosted Feature Groupings	8.44 %	1.83 %	26.33	356
Manual Gram Mapping	8.44 %	1.83 %	26.33	356

Table 5.1: Grammar Performance Metrics for Wambaya

Development Phase	N LCs	N PCs	V LCs	V PCs
Initial Baseline	24.1	29.8	19.3	26.0
Bug Fixes (Baseline)	24.1	20.3	19.3	24.1
Feature-based Overlap	7.3	8.5	17.5	28.2
Input Overlap Prerequisite	21.6	24.9	19.4	30.1
Input Overlap Fallback	21.5	21.3	19.3	24.1
Unlimited Merging Rounds	21.1	21.3	19.2	24.1
Boosted Feature Groupings	21.1	21.3	19.2	24.1
Manual Gram Mapping	23.8	23.1	19.3	24.1

Table 5.2: Grammar Complexity Metrics for Wambaya

number of lexical classes (LCs) and position classes (PCs), respectively. It is also the only run where the number of verb LCs is non-trivially lower than the baseline at 17.5, and one of two runs where the number of verb PCs increases, up to 28.2. This pattern of the first post-baseline run having very different results from the rest of the runs persists for rest of the datasets, and whether the numbers are much lower or much higher than the rest basically comes down to the amount of affixes with or without features. This will be discussed in detail in Section 5.3.3.

In the subsequent runs, the metrics return to being closer to the baseline, settling around 21 for the number of noun LCs and PCs, 19 for the number of verb LCs, and 24 for the number of verb PCs, with the exception of the final run whose grammars on average had 23.8 noun LCs and 23.1 noun PCs, as well as 19.3 verb LCs and 24.1 verb PCs. So, the only real difference between the baseline and the final run was that the final run yielded about 3 more noun PCs per grammar.

In addition to having similar amounts of LCs and PCs, the noun and verb input graphs in the grammars at each stage maintain the same general flow as the baseline, so the changes presented here did not fundamentally change the hypothesized morphotactics. However, there are still differences that are reflective of the changes made.

In the baseline grammar, there is one instance of an affix expressing only number in a PC that otherwise had suffixes expressing only case or no features. This is not the case in the final grammar or any of the grammars preceding it, which is a predictable outcome from the fact that merging was based on feature overlap, and this is the desired outcome in the pursuit of modeling canonical morphotactics. Additionally, the longest path in the baseline grammar's noun input graph has four PCs with case affixes, despite the fact that nouns in the training data (after being processed by MOM, see Section 3.2.4) had at most two case suffixes and no more than three affixes overall.

The baseline grammar's verb input graph has a path with three PCs including one tense PC and the verb input graph of the final grammar also has a path with three PCs but two of them have tense affixes. The training data does have an instance of a verb with three

affixes, but no verb is inflected for tense more than once, so the baseline grammar's input graph is slightly more realistic in this way, though as discussed in Section 3.2.4, grammars will not parse a word with conflicting feature values.

The final run adds manual gram mapping for features that did not get recognized or recognized correctly by MOM. For instance, the gram 'NF' got mapped to nonfuture tense instead of nonfinite form. As a result, a few more verb affixes gained features. This did not end up affecting the merging process, but it is still a positive change because more information is carried through to the final grammar. The gram 'COMIT' got mapped to comitative case and ended up on a noun affix. It appears that this did have some effect on the noun morphotactics based on the number of LCs/PCs going up slightly and the final input graph being slightly different. This arguably had a negative effect on the specific grammar because fewer merges took place and the final input graph was slightly less generalized as a result. However, I still think the overall impact is positive because the merging decisions were more informed. Lastly, three of the four gender grams, 'I', 'II', and 'IV', mapped onto gender values recognized by MOM (masculine, feminine, and neuter)[4] but because gender is only inflected for on noun stems in the data, the features did not show up in any PCs or affect the merging process.

Overall, there was relatively little change in the performance or complexity metrics for Wambaya, but the changes to the input graphs seem to be positive because the final grammar was more realistically constrained than the baseline in terms of the maximum amount of inflection and had more features with affixes due to the manual gram mapping.

5.1.2 Lezgi

The grammar performance metrics for Lezgi are given in Table 5.3. Lexical coverage and sentence coverage remained consistent across the runs, each one having a slight decrease in

[4]Nordlinger (1998) uses noun class and gender terms interchangeably. 'III' indicates vegetable gender which does not have a corresponding value in MOM. In this case, it would have been better to have been able to directly assign these grams as gender values without having to map them to one of the predefined values. See Section 5.3.7 for more discussion on this point.

Development Phase	Lexical Coverage	Coverage	Ambiguity	Max Results
Initial Baseline	11.19 %	9.02 %	2,970.58	23,416
Bug Fixes (Baseline)	11.61 %	9.36 %	3,315.30	28,110
Feature-based Overlap	11.19 %	9.02 %	2,105.69	18,745
Input Overlap Prerequisite	11.03 %	9.02 %	2,503.95	21,775
Input Overlap Fallback	11.11 %	9.19 %	2,429.56	21,983
Unlimited Merging Rounds	11.11 %	9.19 %	2,007.71	18,679
Boosted Feature Groupings	11.11 %	9.19 %	1,993.44	17,746
Manual Gram Mapping	11.11 %	9.11 %	2,087.67	23,968

Table 5.3: Grammar Performance Metrics for Lezgi

both metrics compared to the baseline, though the differences very small. Ambiguity was somewhat reduced, dropping from 3,315.3 average parses per parsed sentence to 2,087.67 by the final run. The maximum number of parses for a single sentence also decreased, from 28,110 parses to 23,968.

The complexity metrics for Lezgi are given in Table 5.4. The number of noun and verb LCs fluctuated as the runs progressed, but always stayed below the baseline counts. After the final run, grammars had 53.5 noun LCs and 29.1 verb LCs on average, which is 7.7 and 12 LCs fewer than the baseline, respectively. The main reason the final number of noun LCs is so high for the Lezgi grammars compared to most others (e.g. 23.8 for Wambaya or 17.7 for Manipuri) is that each one has 40 pronouns, which are not inferred via MOM but are included in the number of noun LCs, as discussed in Section 3.2.3.

For the first two runs post-baseline, the number of PCs almost doubled for nouns and verbs. After these runs, the number of noun and verb PCs returned to about the same as the baseline, and in the final run, there was 1 fewer noun PC per grammar and 1.9 fewer verb PCs per grammar.

Similar to the baseline Wambaya grammar, the baseline Lezgi grammar has several PCs

Development Phase	N LCs	N PCs	V LCs	V PCs
Initial Baseline	61.2	47.5	41.1	48.8
Bug Fixes (Baseline)	61.2	27.2	41.1	27.0
Feature-based Overlap	53.9	43.7	34.6	52.4
Input Overlap Prerequisite	58.4	51.0	37.7	53.7
Input Overlap Fallback	51.7	27.5	37.4	26.8
Unlimited Merging Rounds	50.4	27.1	29.1	25.1
Boosted Feature Groupings	53.5	26.2	29.1	25.1
Manual Gram Mapping	53.5	26.2	29.1	25.1

Table 5.4: Grammar Complexity Metrics for Lezgi

with affixes whose features have no overlap. For nouns, there are seven PCs with some affixes that express only case and others that only express number. For verbs, there are four PCs with different combinations of a variety of features, including tense, aspect, mood, and negation. In the grammar produced by the final run, features are largely separated into different PCs, though for the nouns there is a large PC with a mix of case-only and number-only affixes as well as one affix with case and number, and for verbs there is a PC with affixes expressing tense, negation, and both at once. Despite these few instances of PCs with mixed features, the final grammar is still much more organized than the baseline grammar in this regard.

Though they differ in how the features are sorted, the resulting input graphs for the baseline and the final grammars are large, very interconnected, and messy. For example, there are many paths in the input graphs that pass through multiple PCs whose affixes express the same feature. Without doing an exhaustive search, I found a path in the baseline grammar comprised of seven PCs, with five of them having case affixes, and in the final grammar, I found a four-long path where each had case affixes. However, in the training data, the largest noun has five affixes and it is only marked for number and case once each, though there are a few examples of nouns with three or four affixes that are inflected for number

once and case three times. Looking to the verbs, there is at least one path in both input graphs through seven PCs, each of which has affixes with some feature value, whereas in the data, the verb with the most affixes has five and the verb with most features has three.

In the final run, mapping grams to feature values that MOM recognizes resulted in a few more affixes having features, but it did not noticeably alter the inferred grammars or the evaluation metrics. However, it is still a positive thing overall because more information from the data was retained in the grammar. The grams 'IMPV' (imperative mood) and 'PROHIB' (prohibitive mood) were not in the dataset's list of glosses, so they were added in order to not be filtered out. Additionally, 'PROHIB' was mapped to 'proh' to be recognized by MOM.

All together, the final grammars for Lezgi seem to be of higher quality than the baseline. On average, they have significantly fewer LCs and similar numbers of PCs, though they also dropped a small amount. Looking at specific grammars, the baseline and final grammars are not easily comprehensible due to their size and they both have the ability to parse nouns and verbs with way too many affixes and features based on the training data, but in terms of the types of affixes co-occurring in PCs, the final grammar appears to be somewhat better organized.

5.1.3 Manipuri

The grammar performance metrics for Manipuri are given in Table 5.5. The initial two runs after the baseline caused the lexical coverage and sentence coverage to drop by about a percentage point each, but they recovered in the subsequent runs and ended up at essentially the same values as the baseline. On the other hand, the amount of ambiguity was noticeably reduced between the baseline and the final run. The average amount of parses per parsed sentence dropped from 3,340.79 to 1,697.07, and the maximum number of parses dropped from 32,592 to 23,711.

The complexity metrics for Manipuri are given in Table 5.6. Following the baseline, all of the complexity metrics increased for the "Feature-based Overlap" and "Input Overlap

Development Phase	Lexical Coverage	Coverage	Ambiguity	Max Results
Initial Baseline	6.17 %	5.36 %	1,490.09	32,010
Bug Fixes (Baseline)	6.99 %	6.17 %	3,340.79	32,592
Feature-based Overlap	5.88 %	5.18 %	777.88	22,756
Input Overlap Prerequisite	5.88 %	5.18 %	850.61	24,079
Input Overlap Fallback	7.28 %	6.46 %	1,875.06	23,938
Unlimited Merging Rounds	7.28 %	6.46 %	1,655.36	23,711
Boosted Feature Groupings	7.28 %	6.46 %	1,655.38	23,711
Manual Gram Mapping	7.05 %	6.23 %	1,697.07	23,711

Table 5.5: Grammar Performance Metrics for Manipuri

Development Phase	N LCs	N PCs	V LCs	V PCs
Initial Baseline	20.8	31.9	33.2	49.2
Bug Fixes (Baseline)	20.8	23.7	33.2	32.9
Feature-based Overlap	24.1	44.4	34.9	66.4
Input Overlap Prerequisite	25.2	46.1	35.3	67.2
Input Overlap Fallback	21.3	24.1	31.3	33.4
Unlimited Merging Rounds	17.7	23.0	25.3	32.0
Boosted Feature Groupings	17.7	23.0	25.3	32.0
Manual Gram Mapping	17.7	23.0	25.5	34.1

Table 5.6: Grammar Complexity Metrics for Manipuri

Prerequisite" runs before returning to baseline levels with the "Input Overlap Fallback" run. In line with the other datasets, the number of PCs was affected most dramatically, roughly doubling with the introduction of the feature-based overlap function. The most significant lasting change to the amount of LCs and PCs came from the "Unlimited Merging Rounds" run, which saw a decrease in each of the complexity metrics, though the impact on the number of LCs is more notable. The final grammars had 3.1 fewer noun LCs, 7.7 fewer verb LCs, 0.7 fewer noun PCs, and 1.2 more verb PCs on average compared to the baseline.

Looking at the noun input graphs for the baseline and final grammars, their makeup is somewhat similar. They both have a decent number of PCs with no feature-having affixes (14 for the baseline, 13 for the final grammar) compared to the number of PCs with features (both have six). In the baseline grammar, all of the affixes with a feature express case, which is the same of the final grammar except for one number suffix. The longest path through both input graphs contains six PCs, four of which have at least one affix that expresses some feature. For the final grammar, this includes the PC with the number suffix. In comparison, the nouns in the training data have at most four affixes and at most express two feature values. Based on this information, neither grammar's noun input graph seems to be constructed more intuitively, nor is either one more effective at restricting the maximum amount of affixation.

Like the noun input graphs, the verb input graphs are also similar to each other in that they are both large and seemingly very disorganized, making them hard to visually interpret. The baseline verb input graph has 26 featureless PCs in addition to the eight with feature-having affixes. There are no affixes with multiple features, but there are a few different combinations of features within PCs from different affixes, including aspect and mood, aspect and negation, and mood and negation. In the graph there is at least one path that passes through nine PCs, six of which have at least one affix that expresses some feature value. The input graph in the final grammar also has 26 PCs with no affixes expressing features. Among the PCs with affixes that do express features, there are none with affixes that express different features from each other, but as a consequence the feature-

having affixes are spread out among 12 PCs compared the baseline's eight. Also like the baseline grammar, there is a path through final grammar's verb input graph through nine PCs and seven of them have at least one affix that expresses some feature value. In contrast to both of these grammars, the verbs in the training data have at most five affixes and are inflected for a maximum of three feature values.

The gram map for Manipuri added several pairings for features that MOM recognizes, including one number gram, two aspect grams, and four mood grams. For each of these mappings, the result in the final grammar was an additional affix with that feature value, but because of the large and disorganized nature of the input graphs, none of these additions had a noticeable effect on the merging process. Regardless, the increase in the amount of information preserved from the training data is a positive outcome.

On the whole, the changes made to morphotactic inference appear to be positive for the Manipuri dataset. Despite the fact that the final input graphs were not much more organized or interpretable than the baseline input graphs, if at all, the final grammars produced less ambiguity and had fewer LCs on average. Additionally, the manual mapping of gloss grams to MOM feature values increased the amount of affixes that express features.

5.2 Results for Test Datasets

5.2.1 Wakhi

The performance metrics for Wakhi are given in Table 5.7. There was some fluctuation in the amount of lexical coverage and coverage over the course of the development stages, but the values never strayed far from the baseline. In the final run, the average grammar had 25.18% lexical coverage, which is .59 percentage points lower than the baseline, and 14.79% sentence coverage, which is 1.32 percentage points higher than the baseline. The amount of ambiguity essentially stayed constant between the baseline and the final run.

The complexity metrics for Wakhi are given in Table 5.8. The results follow a similar pattern to what was seen for the development datasets, where the number of LCs and PCs

Development Phase	Lexical Coverage	Coverage	Ambiguity	Max Results
Initial Baseline	25.48 %	13.18 %	18.44	586
Bug Fixes (Baseline)	25.77 %	13.47 %	18.21	586
Feature-based Overlap	25.33 %	14.64 %	17.24	586
Input Overlap Prerequisite	25.04 %	14.64 %	17.62	586
Input Overlap Fallback	26.06 %	15.81 %	17.53	586
Unlimited Merging Rounds	26.06 %	15.81 %	17.39	586
Boosted Feature Groupings	26.50 %	15.81 %	17.39	586
Manual Gram Mapping	25.18 %	14.79 %	18.68	586

Table 5.7: Grammar Performance Metrics for Wakhi

Development Phase	N LCs	N PCs	V LCs	V PCs
Initial Baseline	12.0	8.4	8.6	13.8
Bug Fixes (Baseline)	12.0	6.9	8.6	9.1
Feature-based Overlap	14.6	11.7	7.4	17.6
Input Overlap Prerequisite	15.0	12.7	8.5	17.6
Input Overlap Fallback	12.1	7.0	8.0	10.3
Unlimited Merging Rounds	11.9	7.0	6.9	10.3
Boosted Feature Groupings	11.9	6.9	6.9	10.3
Manual Gram Mapping	11.9	6.9	6.6	9.7

Table 5.8: Grammar Complexity Metrics for Wakhi

jumps up for the first two runs after the baseline and then returns to values similar to the baseline for the "Input Overlap Fallback" run. Also like other results, the number of PCs about doubles for these initial runs. In the final run, the output grammars had on average 11.9 LCs and 6.9 PCs for nouns, showing negligible change from the baseline. For verbs, the grammars had an average of 6.9 LCs and 9.7 PCs, which is 1.7 fewer LCs and .6 more PCs than the baseline.

The grammars produced by the baseline and the final run both have relatively small input graphs for nouns, but neither is clearly superior. They both have one main connected portion of the graph composed of two LCs and four PCs, and three of the PCs have at least one affix that expresses case or case and number. In each graph, there is a path that visits each of these PCs, and in the training data, there is one noun that has four affixes. However, there is only one noun in the training data with multiple feature-having affixes and it has one case affix and one case and number affix.

In contrast to the noun input graphs, the final verb input graph is much more organized than that of the baseline run. The baseline grammar has several PCs with unintuitive combinations of affixes with respect to the features they express. For example, one PC has an affix that expresses negation, another that expresses tense, and a third that expresses person and number. On the other hand, the verb PCs in the final grammar are generally organized by feature set, the only exception being one that has a mix of tense, person/number, and person/number/tense affixes. Additionally, the final verb input graph is more realistic with respect to its longest path. Similar to the nouns, there is one instance of a verb with four affixes the first split's training data, and both the baseline and final input graphs have a path through four PCs. Unlike the nouns, there are several verbs in the data that are inflected for multiple different features, though the only feature that is inflected for multiple times is tense, which is inflected for twice on several verbs. Due to the disorganization of the PCs in the baseline grammar, its verb input graph has paths comprised of four PCs that each have at least one tense affix, or that each have at least one person and number affix. On the other hand, the final grammar's verb input graph has paths that go through at most two tense

PCs or two person and number PCs. This behavior is clearly more desirable, even if having multiple person and number values is not ideal given what is attested in the data.

There were only two relevant feature grams that were not initially recognized by MOM, 'SBJNC' for subjunctive mood and 'PLPF' for distant past.[5] With the addition of the gram map, these feature values showed up on one verb affix each. It appears that the verb input graph from the "Boosted Feature Groupings" run, which included all of the changes except for the gram map, is slightly cleaner than the final grammar's. This because it only has one PC with affixes that inflect for person and number (compared to two) and that PC also does not have any affixes that only inflect for tense, whereas the equivalent PC in the final grammar does. However, one of the affixes with no features in that PC is the one that expresses distant past in the final grammar, so the two grammars are not really different in this way. Based on this, incorporation of gram map seemed to minimally degrade the quality of the verb input graph, but the inclusion of more feature information is arguably worth the tradeoff.

Overall, the final grammars for Wakhi show very little improvement over the baseline grammars in terms of the performance and complexity metrics and in the individual grammars I looked at, the baseline and final noun input graphs were fairly similar in layout and quality. However, the verb input graph of the final grammar is much improved over the baseline grammar's in terms of the organization of affixes in PCs based on their features and the constraints on the range of possible word forms.

5.2.2 South Efate

The evaluation on the South Efate dataset was carried out with a slightly altered version of the code due to the format of the dataset's glosses file (see Section 2.3.3), which lists individual grams instead of whole morpheme glosses. As a consequence, MOM basically ignored any multi-gram morpheme glosses and because the dataset mostly has multi-gram

[5]Technically 'pluperfect', but this is not an option in MOM and Obretelová (2019; pg. 53) equates pluperfect with distant past.

Development Phase	Lexical Coverage	Coverage	Ambiguity	Max Results
Initial Baseline	14.71 %	7.01 %	1,840.20	13,205
Bug Fixes (Baseline)	14.87 %	7.01 %	5,628.68	20,013
Feature-based Overlap	14.38 %	7.01 %	4,820.51	21,224
Input Overlap Prerequisite	14.38 %	7.01 %	4,799.45	21,225
Input Overlap Fallback	14.87 %	7.01 %	5,628.98	20,186
Unlimited Merging Rounds	15.30 %	7.11 %	4,790.44	19,538
Boosted Feature Groupings	15.30 %	7.11 %	4,791.85	19,663
Manual Gram Mapping	15.33 %	6.95 %	6,428.85	23,417

Table 5.9: Grammar Performance Metrics for South Efate

morpheme glosses, the initial inferred grammars had very few affixes with features in the PCs. In order to have results that more accurately reflect the changes made to MOM, I made a temporary modification to the code to work with the South Efate dataset's configuration. The resulting behavior is close to but not exactly the same as the standard behavior, so the original evaluation results will be provided in Appendix A for consistency.

The performance metrics for South Efate are given in Table 5.9. Neither coverage metric changed much between any given pair of development stages, and the final values are fairly similar to the baseline. The overall amount of ambiguity increased from the baseline to the final stage, though the actual increase occurred between the second-to-last and last runs, where the overall amount of ambiguity increased from 4,791.85 to 6,428.85 and the maximum number of parses increased from 19,663 to 23,417. This increase will be analyzed more thoroughly in Section 5.3.7.

The complexity metrics for South Efate are given in Table 5.10. Following the baseline run, the number of noun PCs, verb LCs, and verb PCs ballooned before returning to baseline levels with the "Input Overlap Fallback" run. With the "Unlimited Merging Rounds" run, each metric decreased by some amount, most notably the number of noun LCs. The final

Development Phase	N LCs	N PCs	V LCs	V PCs
Initial Baseline	34.5	29.4	13.8	27.7
Bug Fixes (Baseline)	34.5	11.9	13.8	11.1
Feature-based Overlap	35.4	40.3	21.6	51.6
Input Overlap Prerequisite	35.4	40.3	21.4	51.7
Input Overlap Fallback	34.5	11.9	13.8	11.1
Unlimited Merging Rounds	19.8	9.0	10.7	9.5
Boosted Feature Groupings	19.8	9.0	10.7	9.5
Manual Gram Mapping	20.7	9.0	11.5	9.7

Table 5.10: Grammar Complexity Metrics for South Efate

grammars had 13.8 fewer noun LCs, 2.9 fewer noun PCs, 2.3 fewer verb LCs, and 1.4 fewer verb PCs on average compared to the baseline.

Looking at the noun input graphs of the baseline and final grammars, they both only have two PCs with any features, and in each one, there is only one affix that expresses case. The baseline graph has three more PCs than the final grammar's graph, but for the PCs they do share, the affix makeups and input relations are essentially identical. The longest path through the baseline noun input graph contains five PCs, including one of the PCs with a case affix. Despite having fewer PCs, the final noun input graph has a longest sequence of six PCs, also including one of the PCs with a case affix. In contrast, the nouns in the training data have at most four affixes and are inflected for case. So, while the baseline grammar is slightly more restricted in the maximum number of affixes, the input graph of the final grammar is decently more compact due to having fewer PCs.

The verb input graphs of the baseline and final grammars are similar in terms of and how the are PCs connected to each other, though the final graph has one fewer PC and the graphs differ somewhat in the affixes that make up each PC. So, the longest path through each input graph is the same and contains six PCs, but the possible feature combinations are different. In the baseline graph, there are two suffix PCs with affixes that express either

person or person and number, and both of these PCs are in the longest path. The final grammar also has two PCs with features. One is a suffix PC that contains all of the affixes in the baseline grammar that have features. The other is a prefix PC whose affixes (because of the gram map) express different combinations of person, number, aspect, and mood. Some have person and aspect or person and mood and others have person, number, and aspect or person, number, and mood. Additionally, one affix has person and number and one zero-marked affix has mood only. So, the longest path also contains two person and number PCs, but one of them also has aspect and mood affixes. The verbs in the training data have up to four affixes and in the most extreme case have five feature values, two person, two number, and one mood value. In this regard, both grammars are similar in their potential to analyze maximally inflected words, though the final grammar is ultimately better because it has the aspect and mood affixes.

As is clear from the final grammar's verb input graph, manually mapping feature grams from the data to MOM's feature values resulted in a number of verb prefixes gaining features. In total, 22 affixes had features in the final grammar of the 54 that did not before. The new feature values included real and irrealis mood, perfective aspect, first person, and dual number. The mapping also led to additional affixes having second and third person values. Aside from the features being added, this did change the input graph slightly because it shifted which prefixes ended up in each of the two prefix PCs. No new features were added to noun affixes from this.

On the whole, the final grammars had fewer LCs and PCs on average compared to the baseline and based on the inspection of the individual grammars, they also have more affixes with features, though there was an increase in the amount of ambiguity from the baseline to the final run.

5.2.3 Hiaki

The performance metrics for Hiaki are given in Table 5.11. As was the case with most of the other datasets, coverage and lexical coverage stayed stable across the stages of development.

Development Phase	Lexical Coverage	Coverage	Ambiguity	Max Results
Initial Baseline	16.11 %	8.99 %	37.32	564
Bug Fixes (Baseline)	16.47 %	9.71 %	131.47	11,904
Feature-based Overlap	16.33 %	9.22 %	52.31	1,556
Input Overlap Prerequisite	16.29 %	9.17 %	52.97	1,556
Input Overlap Fallback	16.47 %	9.75 %	65.11	992
Unlimited Merging Rounds	16.55 %	9.80 %	44.26	1,560
Boosted Feature Groupings	16.51 %	9.80 %	30.07	1,125
Manual Gram Mapping	16.55 %	9.35 %	30.33	1,125

Table 5.11: Grammar Performance Metrics for Hiaki

The metrics did vary slightly from run to run, but none of the fluctuations appear to be meaningful. In contrast to this, the amount of ambiguity was significantly reduced between the baseline and final runs, both in terms of the average number of parses per parsed sentence and the maximum number of parses for a single sentence. The largest change to these metrics came with the first run post-baseline which reduced the average parses from 131.47 to 52.31 and the maximum parses from 11,904 to 1,556. Over the rest of the runs, the amount of ambiguity fluctuated some, ending at 30.33 average parses, which is down from the baseline by a factor of about four. In the final run, the maximum parses was 1,125, which is about a tenth of the number of parses.

The complexity metrics for Hiaki are given in Table 5.12. In the first three post-baseline runs, the number of noun LCs was roughly maintained and the number of verb PCs worked down from 34.0 to 30.0. In the first two runs, the number of PCs approximately doubled for nouns and verbs, but then returned to baseline levels with the "Input Overlap Fallback" run. In the "Unlimited Merging Rounds" run, the numbers of LCs and verb PCs were reduced, most significantly for verb LCs which decreased to 19.7 from 30.0. These metrics held for the remaining two runs. The final grammars had 3.9 fewer noun LCs, 0.2 fewer noun PCs,

Development Phase	N LCs	N PCs	V LCs	V PCs
Initial Baseline	23.7	16.8	34.0	39.8
Bug Fixes (Baseline)	23.7	11.7	34.0	24.7
Feature-based Overlap	24.0	18.0	31.8	45.8
Input Overlap Prerequisite	24.1	18.1	32.3	46.1
Input Overlap Fallback	23.1	11.6	30.0	23.7
Unlimited Merging Rounds	19.8	11.5	19.7	20.6
Boosted Feature Groupings	19.8	11.5	19.3	19.9
Manual Gram Mapping	19.8	11.5	19.5	20.0

Table 5.12: Grammar Complexity Metrics for Hiaki

14.5 fewer verb LCs, and 4.7 fewer verb PCs on average compared to the baseline.

The noun input graphs of the baseline and final grammars are somewhat similar. They both have two PCs with a single case affix and a PC with affixes that express case, number, or both. The baseline grammar has an additional PC with affixes that express number or number and case. The input graphs for both grammars have longest paths that include all of their respective PCs with features. Neither of these maximal affix combinations are particularly reasonable, as the nouns in the data have at most one case and one number value, but the final grammar is ultimately more realistic because it is more restrictive, especially because there is no path with multiple number PCs.

The input graphs for verbs are similar between the baseline and final grammars as well because they both have a large number of PCs with a lot of inputs/outputs that make them hard to interpret. The baseline grammar has two PCs that inflect aspect, two that inflect mood, and three whose affixes express different combinations of tense, aspect and mood. As a result of the many input/output relations in the graph, it has a longest path of seven PCs, including all five that have features. The final grammar has three PCs that inflect aspect, one that inflects mood, one that inflects number, and one with affixes that express tense, aspect, or mood, as well as one affix that expresses tense and aspect. It also has a longest

path of seven PCs, but up to six of them have affixes with features. Unsurprisingly, the training data does not have any examples of this amount of inflection. Among the verbs, words have at most four affixes and at most two feature values, so neither input graph is nearly restrictive enough. At first it appears that the baseline is slightly better because its longest path has one less feature PC, but the number feature in the final grammar only showed up because of the manual gram mapping step, and the affix that gained the feature is also in the baseline grammar's longest path.

The changes to the final grammar from the gram map were minimal. The dataset had no grams that could not be recognized by MOM (for the features that it looks for), but two grams were missing in the dataset's list of detectable glosses, 'SG' for singular number and 'LOC' for locative case. As a result of adding these grams, one noun affix gained locative case and moved from an otherwise featureless PC to the PC with most of the affixes expressing case and/or number. Additionally, one verb affix, also previously in a featureless PC, gained singular number but it did not affect any merges.

For grammar inference from the Hiaki dataset, the changes to the code brought about a severe drop in ambiguity and some decent reductions for the complexity metrics, particularly for the number of verb LCs. On top of that, the final grammar has PCs whose affixes are more organized by feature for nouns and verbs and it seems to handle the restriction of over-inflected words better for nouns.

5.3 Analysis

In this section, I will discuss the effects of the changes to MOM on the resulting grammars in more detail, focusing on patterns that appeared across multiple datasets. Overall, the results and patterns seen in the evaluation of the development datasets carried over to the test datasets. This kind of generalization is expected because because the changes made to the system were directly motivated by the typological literature and/or known characteristics of the data used for grammar inference broadly, and they generally did not consider any specific patterns observed in the development datasets, for better or for worse.

Dataset	Lexical Coverage	Coverage	Ambiguity	Max Results
Wambaya	8.19 %	1.83 %	26.33	356
Wambaya	8.44 %	1.83 %	26.33	356
Lezgi	11.61 %	9.36 %	3,315.30	28,110
Lezgi	11.11 %	9.11 %	2,087.67	23,968
Manipuri	6.99 %	6.17 %	3,340.79	32,592
Manipuri	7.05 %	6.23 %	1,697.07	23,711
Wakhi	25.77 %	13.47 %	18.21	586
Wakhi	25.18 %	14.79 %	18.68	586
South Efate	14.87 %	7.01 %	5,628.68	20,013
South Efate	14.89 %	6.93 %	5,928.53	21,786
Hiaki	16.47 %	9.71 %	131.47	11,904
Hiaki	16.55 %	9.35 %	30.33	1,125

Table 5.13: Baseline and Final Grammar Performance Metrics by Dataset

5.3.1 Performance Metrics

The overall changes in the performance metrics are presented in Table 5.13. For each pair of results, the first line has the baseline metrics and the second line has the final metrics.

As noted in the sections for each dataset, there was not much change in the amount of coverage between the baseline and final values. The largest changes for occurred with the Lezgi dataset, whose grammars lost about half of a percent of lexical coverage on average, and the Wakhi dataset, whose grammars also lost around a half percent of lexical coverage but gained a little over a percent of sentence coverage on average. This is not surprising to see given the nature of the changes made in this thesis, as was discussed in Section 3.2.2.

For the other two performance metrics, ambiguity and the maximum number of results, there were improvements in some cases, particularly for datasets with larger amounts of ambiguity initially. The grammars inferred from the Lezgi, Manipuri, and Hiaki datasets all

had a reduction in ambiguity, while the others maintained similar levels from the baseline to the final run. For all three datasets where the ambiguity was reduced, the largest drop came with the first change to the baseline code, where the overlap function for PCs was changed to be based on feature overlap, and that level of ambiguity was maintained through the rest of the runs. The exception to this is the Manipuri dataset, where the average number of parses was initially reduced by a factor of about four and then later was doubled, so the final results produced on average about half the number of parses as the baseline. The initial reductions in ambiguity can at least partially be explained by the sharp decrease in the amount of PC merging seen in the initial runs for most of the datasets. One source of ambiguity with respect to morphotactics is having multiple affixes with the same orthographic form but that express different feature sets. For example, in Manipuri the suffix -*lə* is used to mark both perfect aspect and prospective aspect (Chelliah, 1997). When there are multiple affixes with the same orthographic form in an input graph that is very connected, there will generally be more ways to parse words that have an affix with that form. On the flip side, when the graph is less condensed (i.e. there are fewer affixes and fewer inputs/outputs per PC and more PCs overall), there will generally be fewer cases where the same word can be parsed in multiple different ways by using different affixes with the same form.[6] However, the fact that the amount of ambiguity remained below baseline levels when the number of PCs eventually went back down to baseline levels (and in some cases decreased even further) suggests that the orientation towards feature-based PCs can improve the quality of inferred grammars in this way.

5.3.2 Complexity Metrics

The overall changes in the complexity metrics are presented in Table 5.14. For each pair of results, the first line has the baseline metrics and the second line has the final metrics.

[6] In this particular case for Manipuri, the amount of ambiguity caused by -*lə* might have actually gone up initially because both morphemes inflect aspect and so are likely to have ended up in the same PC, but if they did not inflect the same feature, then the amount of ambiguity would more likely decrease.

Dataset	N LCs	N PCs	V LCs	V PCs
Wambaya	24.1	20.3	19.3	24.1
Wambaya	23.8	23.1	19.3	24.1
Lezgi	61.2	27.2	41.1	27.0
Lezgi	53.5	26.2	29.1	25.1
Manipuri	20.8	23.7	33.2	32.9
Manipuri	17.7	23.0	25.5	34.1
Wakhi	12.0	6.9	8.6	9.1
Wakhi	11.9	6.9	6.6	9.7
South Efate	34.5	11.9	13.8	11.1
South Efate	20.7	9.0	11.8	9.8
Hiaki	23.7	11.7	34.0	24.7
Hiaki	19.8	11.5	19.5	20.0

Table 5.14: Baseline and Final Grammar Complexity Metrics by Dataset

In most cases, the average number of LCs and PCs stayed the same or decreased by a small amount. The most significant decreases came in the number of LCs, namely for Lezgi nouns and verbs, Manipuri verbs, and Hiaki verbs. Looking at the runs from the different development stages, these drops all came from the "Unlimited Merging Rounds" run, and this is unsurprising because this change affected LC and PC merging equally, whereas the rest of the changes (aside from the manual gram mapping) occurred within the PC overlap function. Even though the merges that happen between PCs do have an effect on what LCs are merged, it makes sense that the more general-purpose adjustment had the greatest effect. There were also a few instances where the number of PCs had a net increase, though the differences are relatively small. This happened for the Wambaya nouns, Manipuri verbs, and Wakhi verbs.

5.3.3 Using the basic feature overlap function

In the baseline system, PC overlap is calculated as the proportion of inputs that are shared between two PCs. When developing the feature-based overlap method, my initial attempt (corresponding to the "Feature-based Overlap" evaluation run) mirrored that, calculating overlap as the proportion of features that are shared between two PCs. However, this was not adequate for a few reasons.

Most immediately obvious was the fact that the number of PCs for nouns and/or verbs was very large for many of the datasets, up to doubling the baseline numbers. This happened because most of the affixes in the data expressed no features, and without an alternative to merging based on feature overlap (which came with the "Input Overlap Fallback" run), any affix with no features remained in its own PC. For example, in the first train/evaluation split of the Manipuri dataset, only 7 of 46 noun affixes and 12 of 71 verb affixes express some feature and as a result, the number of noun PCs increased from 23 to 44 and the number of verb PCs increased from 34 to 66 when the overlap function switched to using feature overlap. So, even if the basic feature overlap function was enough to precisely model the morphotactics of most languages, the reality is that there is still a large gap between the feature information encoded in the IGT data used for grammar inference and the information that is preserved through the automatic extraction process.

On the flip side, most of the noun affixes in the Wambaya dataset express either case or case and number simultaneously. As a result, basically all of the affixes with features ended up in one large PC in the grammars produced using the basic feature overlap function and the few remaining PCs had affixes expressing only number or no features at all. In the evaluation run, the resulting grammars had on average only 7.3 noun LCs and 8.5 noun PCs compared to 24.1 LCs and 20.3 PCs in the baseline. Initially, this seems like an ideal outcome because a singular case PC was formed and the number of noun PCs and LCs looks a lot closer to the number a linguist-created grammar might have. However, it is very possible that there are more things that need to be accounted for that this single PC would not gracefully handle.

For example, if the inflection of case exhibited lexically-conditioned allomorphy, i.e. certain affixes can only co-occur with certain noun stems, then a more nuanced implementation, such as having multiple parallel PCs, would be required to properly constrain the range of valid forms. Indeed, some cases suffixes in Wambaya do exhibit allomorphy that is conditioned on things like number and lexical item (Nordlinger, 1998; pg. 81).

In sum, while the basic feature overlap function is the most straightforward way to infer a canonical morphotactic system because the PCs of the resulting grammar will have affixes that are sorted by feature, the reality of the data that the AGGREGATION system works with and the fact that no language has perfectly canonical morphotactics require that additional considerations be made.

5.3.4 Non-canonical PCs

One aspect of the inferred grammars that definitely improved was the affix makeup of PCs, i.e. the degree to which the features expressed on affixes make sense together. As discussed in the individual results sections, the baseline grammars often had affixes in the same PC that likely didn't belong together given their features. For example, the baseline Wakhi grammar had a PC with a negation affix, a tense affix, and a person and number affix. While most of the final grammars did not end up with PCs like this (the final Wakhi grammar did not have that combination), there are still a few types of PCs that fall outside of the strict bounds of canonical morphotactics.

The first type is simply PCs whose affixes don't have any features. There are a few reasons why an affix might not have any features. The first is that there are some features that MOM does not put on affixes, even if they are successfully recognized from the data. For verbs, affixes can only have person, number, gender, tense, aspect, mood, and negation features, and for nouns, affixes can only have number and case features. Any other feature grams are ignored. For some grams, such as ones marking passive voice, it is because the Matrix does not represent that concept as a feature. The second main reason, or really category of reasons, is that there was some issue identifying a feature value with an affix/its

gloss during the initial processing of the input data (see Section 2.3.3). When no affixes in a PC have features, it is harder for a linguist using the grammar to know what function that PC might serve in the language's morphological system, or if its combination of affixes makes any sense at all. This type of PC is certainly nothing new given the amount of affixes that end up with no features, but it is more relevant with an inference process based around features because featureless affixes no longer have the same status as those with features, so there is more to gain from closing the gap between the feature information in the IGT data and successfully extracted from it.

The next type is PCs with affixes that have partially overlapping or non-overlapping feature sets. These PCs were less common in the final grammars, but there are still ways for them to be formed in the merging process with the new implementation of the overlap function. For instance, there is a verb PC in the final Hiaki grammar with affixes that express, tense, aspect, or mood, as well as one affix with tense and aspect. Obviously, the tense affix was still able to end up in the same PC as the affix with tense and aspect because they have some feature overlap. Similarly, the tense affix and the aspect affix were able to end up together because they each have partial feature overlap with the tense and aspect affix and the overlap function looks at the set of features in the PC as a whole, so it doesn't matter that two individual affixes have no overlap. Finally, the mood affixes were merged into the PC via the stipulation I added to the overlap function to encourage merging between features that are known to sometimes be inflected together, in this case tense, aspect, and mood. In the strict sense of canonical morphotactics, these affixes do not belong in the same PC because their feature sets are not exactly equivalent, but there are some reasons why it still might make sense to have them in the same PC anyway, as discussed in Chapter 4. So, while it would take a more in-depth case study of specific languages to determine whether these "non-canonical" PCs align with reality, the fact that it's clear from the code how these PCs were formed already makes them more interpretable.

5.3.5 Issues with Featureless Affixes

While the final grammars often showed improvements over the baseline grammars in terms of how organized their input graphs were or how easy they were to interpret, this was not always the case. For Lezgi nouns and verbs, Manipuri nouns and verbs, and Hiaki verbs, the final input graphs, like their baseline counterparts, were very large and difficult to interpret. Most of their PCs had many inputs and outputs, and while they were often more organized by feature, there were still many long paths in the input graphs passing through multiple PCs with the same features. There were also some instances where the input graphs were relatively small and simple and did not change much between the baseline and final grammars, as was the case with the noun input graphs inferred from the Wakhi dataset.

As an example of a large input graph, the final Lezgi noun input graph is shown in Figure 5.1 and the same graph is shown in Figure 5.2 with a highlighted path of 4 consecutive PCs that are each mostly comprised of affixes that express case. The same graph is shown a third time in Figure 5.3, where I manually sorted the PCs by what features are on their affixes using the interactive interface. The top left cluster is number PCs, the top middle PC has case and number affixes, the top right cluster is case PCs, and the bottom cluster is PCs with no features. Here, there still doesn't seem to be any obvious organization that would point to what a typical noun might look like in Lezgi.

In both of these situations, where a large or small input graph did not change much from the baseline, the input graphs have several PCs with affixes expressing the same feature that were not merged, e.g. the 13 case PCs in the Lezgi noun input graph shown in Figure 5.3. Despite these PCs sharing features and inputs, they are prevented from being merged, and this is because merging any pair of them would result in a cycle in the input graph, which is disallowed in Matrix grammars. For instance, there are many cases where two PCs have the same features but one PC is an input of the other, so they cannot be merged because a PC cannot have itself as an input. This would be the expected outcome if the training data contained words with multiple affixes expressing the same features, but that is often not the

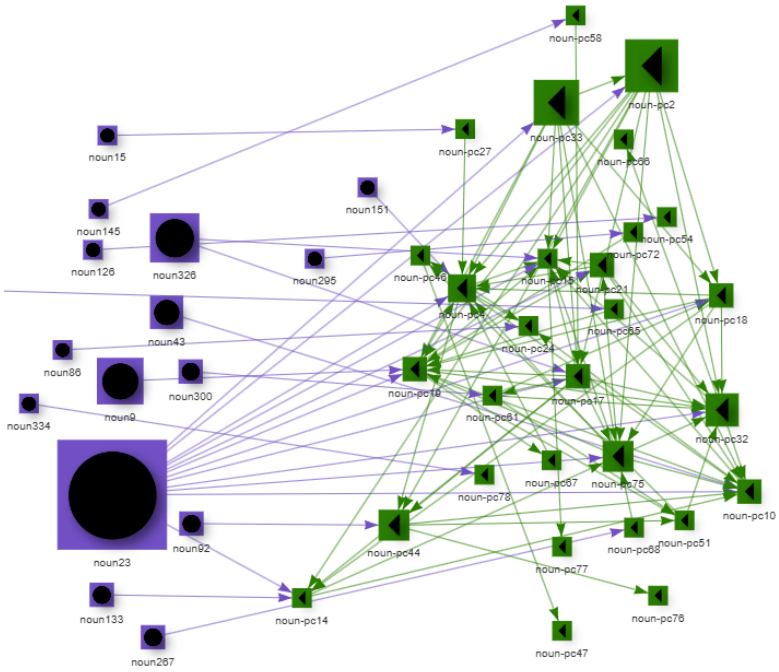

Figure 5.1: Final Lezgi noun input graph

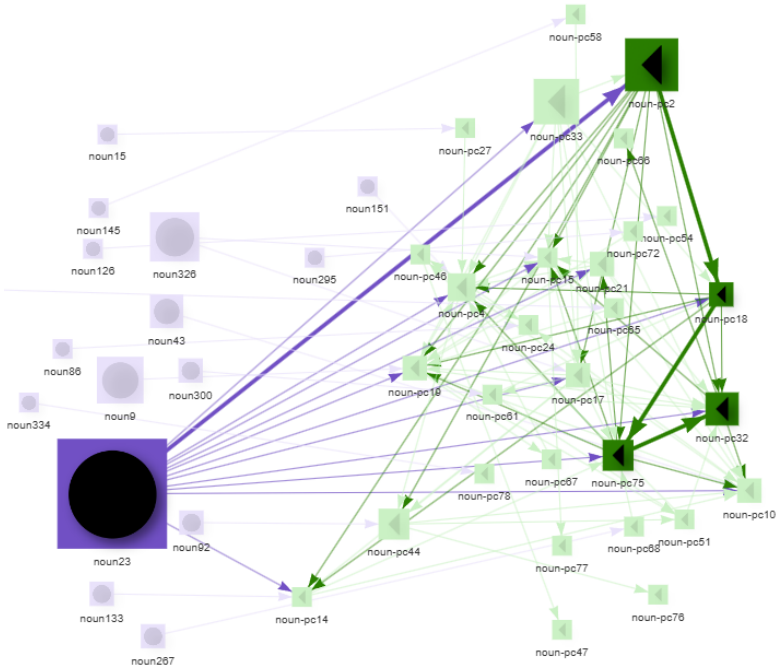

Figure 5.2: Final Lezgi noun input graph with a path through 4 case PCs highlighted

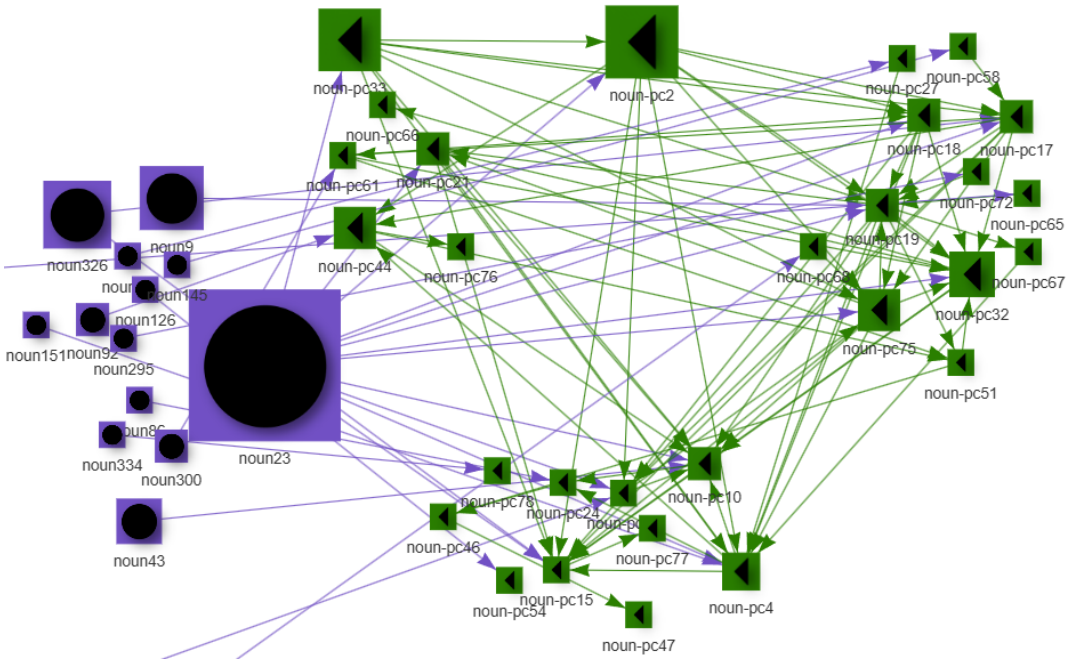

Figure 5.3: Final Lezgi noun input graph with PCs sorted by feature

case.

In reality, this is happening because of merges between PCs whose affixes have features and PCs whose affixes do not. One example of this took place during the inference process for Wakhi nouns. Initially, the PC created for the affix *-m*, which expresses proximate case, had one LC input and was an input of a PC whose affix has no features, and the PC created for the affix *-a*, which has no features, had the same LC input and was an input of three different PCs, of which two had affixes with some case value. During the first round of merging, the *-m* and *-a* PCs had an overlap of 1.0 and were merged because *-a* had no features and the PCs had complete input overlap, resulting in a sequence of two PCs with case affixes (from *-m* and the output affixes of *-a*), despite the fact that there was only a single noun in the dataset with multiple case affixes. This also meant that *-m* could not end up in the same PC as the other case affixes. Once all of the merging was completed, the longest path through the input graph included all three of its PCs with case affixes.

Starting with the "Input Overlap Fallback" run, the overlap function reverts to using input overlap whenever one or both of the PCs in question have no features, as was seen in the example above. The intent with this was to increase the amount of merging by creating a way for affixes with no features to be merged with other affixes, and some version of this is necessary to prevent all of the affixes with no features from ending up in their own PC. However, it seems that this approach is too greedy, allowing some merges to happen earlier in the process but preventing certain higher quality merges, i.e. merges between PCs with shared features, later on.

To counteract this, it might make sense to have the overlap function change as the merging rounds go on, for instance only enabling the input overlap fallback after the first round. This might cause fewer merges to happen initially, but it could help to prioritize feature-based merges. Another approach to this could be simply devaluing input-based overlap compared to feature-based overlap, say by dividing any input overlap value by two.

5.3.6 Long Paths Through Input Graphs

One thing I took note of for each of the baseline and final inferred grammars was the length of the longest path through the noun and verb input graphs and how many PCs it passed through whose affixes had features. In many cases, these maximal constructions were much larger than anything seen in the training data, so it is clear that there is still work to be done in this regard, for the sake of more accurately modeling languages but also for inferring grammars that are simpler and more interpretable for a linguist using the grammar.

In reality, not all of the obviously incorrect forms that seem to be validated by the input graphs would be parsed by the grammars because conflicting feature values are not allowed. However, because most PCs end up having at least one affix with no features, this fact doesn't really cut down on the maximum number of affixes a word can have. Even if the grammars did somehow prevent the rest of these unreasonable affix combinations via other mechanisms, the large and messy input graphs would still be far less than ideal from the perspective of wanting to infer grammars that are a good jumping-off point for further

development because they are often completely uninterpretable.

5.3.7 Issues with Manual Gram Mapping

Overall, the addition of a way to manually map grams from the data to MOM's internal feature values had a positive impact because it helped to reduce the amount of information lost in the data extraction process which in turn allowed the decisions made in the morphotactic inference process to be more informed. However, there are a couple of small issues that arose with this feature.

First, there are a few feature values that are not possible to map to. This is because I set up the configuration file to only take the string that maps to the internal feature value, but those values are not all unique. When attempting to match that string to a feature value, the features are always checked in the same order, so if there are multiple matches, it always takes the first one. For example, I attempted to map a gram to intentive mood for the Manipuri dataset, but because the string 'intt' is used by MOM for both intentive mood and intentive aspect, the gram got incorrectly mapped to intentive aspect. This could be remedied by changing the gram map configuration to specify feature names instead of feature values and using the gloss grams from the dataset as the feature value. It's reasonable to assume that the gloss gram used for a given feature value will be consistent within a dataset, so there shouldn't be a need for further standardization in that respect. In fact, this may be more desirable for a linguist using the system who also created the IGT because they will be used to using a certain set of grams. However, standardization may still be necessary for certain features whose possible values are less open-ended, e.g. person and number, because other grammar inference modules may utilize those features and expect a certain set of values.

Secondly, the addition of the gram map caused an increase in ambiguity in some cases, namely for the Lezgi and South Efate datasets. This difference is the result of grams being variably glossed, whether that be because there is legitimate allomorphy or because of inconsistent data/data processing. If none of the glosses of a variably glossed affix are interpreted as containing relevant feature values, all of the affixes are treated as representing the same

morpheme and there will only be one affix with that form in the grammar. Then when the gram map is introduced, if some or all of the glosses are newly recognized as having features, there will be an affix for each unique feature combination in the grammar. This creates the potential for more ambiguity in the grammar because when a word with that affix is parsed, there may be more ways to parse it (depending on how the affixes are organized into PCs). In the South Efate dataset, there were five affix forms that ended up being recognized as multiple morphemes after the gram map was added. One was the verb prefix *i-*, which is glossed in three different ways. (3) shows an example word from the data for each gloss:

(3) South Efate (Thieberger, 2006)

a. i-tae
   3S.RS-know

b. i-mai
   i-come

c. i-srakor-o
   3S.3SRS-hide-TS

As described in Section 4.3, the process of getting feature grams to be recognized by MOM requires adding the morpheme-level gloss to the dataset's glosses list and/or mapping the individual grams to MOM's internal feature values with the gram map. For the prefix in (3a), I added 3S.RS to the dataset's glosses list and mapped RS to realis mood, resulting in this affix being correctly interpreted as marking 3rd person (subject agreement) and realis mood. The morpheme gloss 'i-' for the prefix in (3b) appears to be a glossing error in the data, so this affix remained featureless. Finally for (3c), I added 3S.3SRS to the dataset's glosses list and mapped 3SRS to 3rd person and singular number. In reality, this gram also marks realis mood, but because MOM does not check for grams that express this feature combination, only two of the feature values were preserved. It seems likely here that the prefix in each case represents the same morpheme (specifically the 3rd person singular realis subject agreement proclitic, as analyzed by Thieberger (2006; pg. 109)), so any increase in ambiguity caused by the variable glossing would be spurious.

This issue could potentially be helped (such as in the case of glossing errors like (3b)) by making additional assumptions about affixes with no features, such as that they have the same features as an affix with the same orthography that does have features. This change would also impact similar feature-having/featureless affix pairs that existed before the inclusion of the gram map. To better handle cases like (3c) where a single gram expresses an unusual feature combination, the flexibility of the gram map configuration could be increased to allow the mapping of a gram to any number of feature values, though this would likely also require changes to how MOM fundamentally interprets grams.

5.4 Summary

Overall, the adjustments made to the morphotactic inference process were successful in maintaining grammar performance and reducing complexity in terms of the number of LCs and PCs for the inferred grammars. Additionally, while the ability to manually map gloss grams to feature values did not greatly affect the main metrics, it contributed to overall grammar quality by way of increasing the amount of information encoded in grammars.

Due to the large amount of affixes in each dataset that were not interpreted by MOM as having features, the resulting input graphs of grammars were often fairly similar to those of the baseline grammars, though the affixes that did express feature values were much more frequently organized by feature, resulting in more interpretable input graphs. For the grammars inferred from the Lezgi and Manipuri datasets, this led to a reduction in ambiguity for sentences that were successfully parsed.

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

This thesis presented a set of changes to the MOM morphological inference module aimed at improving the quality of automatically inferred grammars by reorienting the position class (PC) merging process around the features expressed on noun and verb affixes. In terms of grammar performance, these changes had relatively little effect on parsing coverage, but there was a reduction of ambiguity for three of the six datasets that the system was evaluated on. The inferred grammars also saw on average a decrease in the number of lexical classes (LCs) and PCs for nouns and verbs, and this contributed to the goal of more interpretable grammars. Lastly, analysis of grammars' noun and verb input graphs revealed that the final grammars were more legible than their baseline counterparts in many instances, though there were some cases where the differences between the grammars were minimal in this regard.

6.2 Future Work

With respect to incorporating feature information into the morphotactic and larger grammar inference process, there is still much room for improvement. In many of the situations where little or no improvement was seen, the root of the issue was the lack of feature information in the data. While some of this is lack of information is by design (i.e. MOM only looks for the values of certain features), there still exists a gap between what information is encoded in the gloss line of the IGT data and what is realized in MOM/the AGGREGATION pipeline's internal representation of the data. As discussed in Section 5.3.7, there are some ways in which the manual gram mapping functionality added in this thesis could be improved. First and foremost, mapping gloss grams to feature names as opposed to a specific value in a feature

would increase the flexibility of this piece of the configuration because it would allow the system to interpret additional gender, aspect, or mood values, for instance, without needing MOM's list of potential values to be updated. Apart from improving the manual gram map feature, it might also be worth exploring the effects of expanding the list of features that MOM looks for when adding feature values to affixes.

There are also some ways in which I think the implementation of the overlap function could be improved. As discussed in Section 5.3.5, I believe that merges between PCs where one or both have no features should be deprioritized, and this could be done by weighting the components of the overlap score or by having the overlap function change as the merging rounds go on, for example by only allowing merges between two PCs with features in the first round. Secondly, the calculation of the number of shared features does not adequately account for the difference between subject and object agreement. The number of shared features and total features is based on the set of feature names that each PC data structure maintains that contains all of the unique feature types expressed by affixes in that PC. However, this set does not make a distinction between person, number, or gender features that agree with the subject vs the object, so the morphotactics would likely be incorrectly inferred for languages whose verbs are inflected with subject and object agreement features.

Finally, I think it could be beneficial to expand the ways in which the grammars are evaluated, both quantitatively and qualitatively. On the quantitative side, the "longest path through the input graph" could be formally implemented as a metric of grammar quality, e.g. with a Python graph library. In my analysis of the results, I found the longest paths via the interactive input graph viewer, but for some of the larger and messier graphs, it was hard to know whether I had actually found the longest path. As discussed, this number does not necessarily coincide with the maximum number of affixes a word can have in the grammar, but I found this metric to be a good proxy for the somewhat abstract idea of how intuitive/interpretable an input graph is, at least in one aspect. In addition, it would be very useful to have word-level lexical coverage metrics, e.g. the percentage of nouns that have at least one parse, as the existing lexical coverage metric is only concerned with whether

every word in a sentence has a parse. With respect to qualitative evaluation, this thesis did not focus much on the actual structures of the languages being modeled or the analyses of individual examples by the inferred grammars. Future work could conduct a case study with a specific language and create a taxonomy of errors to gain insights about where there is still room for improvement.

In this thesis, my goal was to evaluate whether grammatical feature information could be used to infer morphotactic systems that are more intuitive to linguists and that more realistically model language. In pursuit of this, I implemented a new version of the PC overlap function in MOM that uses said feature information to orient the inference process towards creating canonical position classes while also making considerations for flexivity and common feature groupings, as well as certain practical realities like uninterpreted feature information. Though the amount of feature information available was often sparse, the experiment results provide solid evidence for the potential of this approach, and by continuing to close the gap between the feature information in the data and the internal representation of the feature information, I believe this potential can be more fully realized.

BIBLIOGRAPHY

Bender, E. M. (2008). Grammar Engineering for Linguistic Hypothesis Testing. In N. Gaylord, A. Palmer, & E. Ponvert (Eds.), *Proceedings of the Texas Linguistics Society X Conference: Computational Linguistics for Less-Studied Languages* (pp. 16–36). Stanford: CSLI Publications ONLINE.

Bender, E. M., Crowgey, J., Goodman, M. W., & Xia, F. (2014, June). Learning Grammar Specifications from IGT: A Case Study of Chintang. In J. Good, J. Hirschberg, & O. Rambow (Eds.), *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages* (pp. 43–53). Association for Computational Linguistics. https://doi.org/10.3115/v1/W14-2206

Bender, E. M., Drellishak, S., Fokkens, A., Poulson, L., & Saleem, S. (2010). Grammar Customization. *Research on Language and Computation*, *8*(1), 23–72. https://doi.org/10.1007/s11168-010-9070-1

Bender, E. M., Flickinger, D., & Oepen, S. (2002). The Grammar Matrix: An Open-Source Starter-Kit for the Rapid Development of Cross-linguistically Consistent Broad-Coverage Precision Grammars. *COLING-02: Grammar Engineering and Evaluation*. Retrieved July 2, 2025, from https://aclanthology.org/W02-1502/

Bender, E. M., & Good, J. (2005). Implementation for Discovery: A Bipartite Lexicon to Support Morphological and Syntactic Analysis. *Papers from the Regional Meeting of the Chicago Linguistic Society*, *41*(2), 1–16.

Bender, E. M., Goodman, M. W., Crowgey, J., & Xia, F. (2013, August). Towards Creating Precision Grammars from Interlinear Glossed Text: Inferring Large-Scale Typological Properties. In P. Lendvai & K. Zervanou (Eds.), *Proceedings of the 7th Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities* (pp. 74–

83). Association for Computational Linguistics. Retrieved April 17, 2025, from https://aclanthology.org/W13-2710/

Bickel, B., Comrie, B., & Haspelmath, M. (2015). The Leipzig Glossing Rules: Conventions for interlinear morpheme-by-morpheme glosses. Max Planck Institute for Evolutionary Anthropology and Department of Linguistics, University of Leipzig [https://www.eva.mpg.de/lingua/resources/glossing-rules.php].

Bickel, B., & Nichols, J. (2007). Inflectional Morphology. In T. Shopen (Ed.), *Language Typology and Syntactic Description: Volume 3: Grammatical Categories and the Lexicon* (2nd ed., pp. 169–240, Vol. 3). Cambridge University Press. https://doi.org/10.1017/CBO9780511618437.003

Bickel, B., & Nichols, J. (2013). Exponence of Selected Inflectional Formatives (v2020.4). In M. S. Dryer & M. Haspelmath (Eds.), *The World Atlas of Language Structures Online*. Zenodo. https://doi.org/10.5281/zenodo.13950591

Bonami, O., & Crysmann, B. (2013). Morphotactics in an information-based model of realisational morphology. *Proceedings of the International Conference on Head-Driven Phrase Structure Grammar*, 27–47. https://doi.org/10.21248/hpsg.2013.2

Chelliah, S. L. (1997). *A Grammar of Meithei* (1st ed., Vol. 17). Mouton de Gruyter.

Chelliah, S. L. (2019). Meithei texts. [Manipur Digital Resources in UNT Digital Library. University of North Texas Libraries. (Accessed August 2019)].

Clifton, J. M., Donet, C. M., & Smith, J. (2024). Lezgi Texts from Azerbaijan [Endangered Languages Archive. Handle: http://hdl.handle.net/2196/fbbf9e7c-2074-4ef1-88e1-6804b5b2ec92 [Accessed: January 2026]].

Conrad, E. (2021). Tracing and Reducing Lexical Ambiguity in Automatically Inferred Grammars. Retrieved April 23, 2025, from http://hdl.handle.net/1773/47085

Copestake, A. (2002). *Implementing Typed Feature Structure Grammars*. Stanford: CSLI.

Crysmann, B., & Packard, W. (2012). Towards Efficient HPSG Generation for German, a Non-Configurational Language. *Proceedings of COLING 2012*, 695–710. https://www.aclweb.org/anthology/C12-1043

Curtis, C. (2018). *A Parametric Implementation of Valence-changing Morphoplogy in the LinGO Grammar Matrix* [Master's thesis, University of Washington].

Dahl, Ö., & Velupillai, V. (2013). Tense and Aspect (v2020.4). In M. S. Dryer & M. Haspelmath (Eds.), *The World Atlas of Language Structures Online*. Zenodo. https://doi.org/10.5281/zenodo.13950591

Dods, A. (2022). Automatically Inferring Grammar Specifications for Adnominal Possession from Interlinear Glossed Text. Retrieved April 23, 2025, from http://hdl.handle.net/1773/49392

Georgi, R. (2016). From Aari to Zulu : Massively Multilingual Creation of Language Tools using Interlinear Glossed Text. http://hdl.handle.net/1773/37168

Goodman, M. W. (2013). Generation of Machine-Readable Morphological Rules from Human-Readable Input.

Goodman, M. W., Crowgey, J., Xia, F., & Bender, E. M. (2015). Xigt: Extensible interlinear glossed text for natural language processing. *Language Resources and Evaluation*, *49*(2), 455–485. https://doi.org/10.1007/s10579-014-9276-1

Haeger, M. (2017). *An Evidentiality Library for the LinGO Grammar Matrix* [Master's thesis, University of Washington].

Hammarström, H., Forkel, R., Haspelmath, M., & Bank, S. (2025). Glottolog 5.2. Max Planck Institute for Evolutionary Anthropology [Available online at http://glottolog.org, Accessed on 2026-02-20]. https://doi.org/https://doi.org/10.5281/zenodo.15525265

Harley, H. (2019). Hiaki text corpus. [University of Arizona. Unpublished FieldWorks (FLEx) project. (Accessed August 2019).].

Howell, K. (2020). Inferring Grammars from Interlinear Glossed Text: Extracting Typological and Lexical Properties for the Automatic Generation of HPSG Grammars. Retrieved April 16, 2025, from http://hdl.handle.net/1773/46080

Kibort, A. (2008). Grammatical Features Inventory: Tense. University of Surrey. http://dx.doi.org/10.15126/SMG.18/1.07

Kibort, A., & Corbett, G. G. (2008). Grammatical Features Inventory: Gender. University of Surrey. http://dx.doi.org/10.15126/SMG.18/1.01

Lepp, H., Zamaraeva, O., & Bender, E. M. (2019). Visualizing Inferred Morphotactic Systems. In W. Ammar, A. Louis, & N. Mostafazadeh (Eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)* (pp. 127–131). Association for Computational Linguistics. https://doi.org/10.18653/v1/N19-4022

Lin, Y.-C. (2023). Automatically Inferring Grammar Specifications for Valence-changing Verbal Morphology from Interlinear Glossed Text. Retrieved April 23, 2025, from http://hdl.handle.net/1773/50476

Liu, T. (2025). *The Interplay of Dataset Characteristics in Automated Grammar Generation: A Study with the AGGREGATION System* [Master's thesis, University of Washington]. Retrieved September 26, 2025, from https://hdl.handle.net/1773/53678

Nielsen, E. K. (2018). *Modeling Adnominal Possession in the LinGO Grammar Matrix* [Master's thesis, University of Washington].

Nordlinger, R. (1998). *A Grammar of Wambaya, Northern Australia*. Pacific Linguistics.

Obrtelová, J. (2019). *From Oral to Written: A Text-linguistic Study of Wakhi Narratives* [Doctoral dissertation]. Uppsala University.

Oepen, S. (2001). [incr tsdb()] — Competence and Performance Laboratory User Manual. Technical report, Saarbrücken, Germany.

O'Hara, K. (2008). A Morphotactic Infrastructure for a Grammar Customization System.

Plank, F. (1999). Split morphology: How agglutination and flexion mix. *3*(3), 279–340. https://doi.org/10.1515/lity.1999.3.3.279

Pollard, C., & Sag, I. A. (1994, August). *Head-Driven Phrase Structure Grammar*. University of Chicago Press. Retrieved July 2, 2025, from https://press.uchicago.edu/ucp/books/book/chicago/H/bo3618318.html

Stump, G. T. (1993). Position classes and morphological theory. In G. Booij & J. van Marle (Eds.), *Yearbook of Morphology 1992* (pp. 129–180). Springer Netherlands. https://doi.org/10.1007/978-94-017-3710-4_6

Thieberger, N. (2006). *A Grammar of South Efate*. University of Hawai'i Press. Retrieved January 13, 2026, from https://muse.jhu.edu/pub/5/monograph/book/8234

Wax, D. (2014). *Automated Grammar Engineering for Verbal Morphology* [Master's thesis, University of Washington].

Xia, F., Lewis, W. D., Goodman, M. W., Slayden, G., Georgi, R., Crowgey, J., & Bender, E. M. (2016). Enriching a massively multilingual database of interlinear glossed text. *Language Resources and Evaluation*, *50*(2), 321–349. https://doi.org/10.1007/s10579-015-9325-4

Zamaraeva, O. (2016, August). Inferring Morphotactics from Interlinear Glossed Text: Combining Clustering and Precision Grammars. In M. Elsner & S. Kuebler (Eds.), *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology* (pp. 141–150). Association for Computational Linguistics. https://doi.org/10.18653/v1/W16-2021

Zamaraeva, O., Curtis, C., Emerson, G., Fokkens, A., Goodman, M., Howell, K., Trimble, T. J., & Bender, E. M. (2022). 20 years of the Grammar Matrix: Cross-linguistic hypothesis testing of increasingly complex interactions. *Journal of Language Modelling*, *10*(1), 49–137. https://doi.org/10.15398/jlm.v10i1.292

Zamaraeva, O., Howell, K., & Bender, E. M. (2019a, February). Handling cross-cutting properties in automatic inference of lexical classes: A case study of Chintang. In A. Arppe, J. Good, M. Hulden, J. Lachler, A. Palmer, L. Schwartz, & M. Silfverberg (Eds.), *Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)* (pp. 28–38). Association for Computational Linguistics. Retrieved April 30, 2025, from https://aclanthology.org/W19-6005/

Zamaraeva, O., Howell, K., & Bender, E. M. (2019b). Modeling Clausal Complementation for a Grammar Engineering Resource. *Proceedings of the Society for Computation in Linguistics*, *2*, Article 6. https://scholarworks.umass.edu/scil/vol2/iss1/6/

Zamaraeva, O., Kratochvíl, F., Bender, E. M., Xia, F., & Howell, K. (2017, March). Computational Support for Finding Word Classes: A Case Study of Abui. In A. Arppe, J. Good, M. Hulden, J. Lachler, A. Palmer, & L. Schwartz (Eds.), *Proceedings of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages* (pp. 130–140). Association for Computational Linguistics. https://doi.org/10.18653/v1/W17-0118

Appendix A

South Efate	Lex. Coverage	Coverage	Ambiguity	Max Results
Initial Baseline	14.71 %	7.01 %	1,840.20	13,205
Bug Fixes (Baseline)	14.80 %	7.18 %	1,887.11	12,865
Feature-based Overlap	14.38 %	7.05 %	1,608.13	9,671
Input Overlap Prerequisite	14.38 %	7.05 %	1,608.22	9,670
Input Overlap Fallback	14.80 %	7.18 %	1,888.56	12,955
Unlimited Merging Rounds	14.78 %	6.98 %	1,267.19	10,601
Boosted Feature Groupings	14.78 %	6.98 %	1,267.60	10,583
Manual Gram Mapping	14.78 %	6.94 %	1,331.38	10,500

Table A.1: Original Grammar Performance Metrics for South Efate

South Efate	N LCs	N PCs	V LCs	V PCs
Initial Baseline	34.5	29.4	13.8	27.7
Bug Fixes (Baseline)	34.5	11.9	13.8	10.5
Feature-based Overlap	35.4	40.3	21.4	53.5
Input Overlap Prerequisite	35.4	40.3	21.4	53.5
Input Overlap Fallback	34.5	11.9	13.8	10.5
Unlimited Merging Rounds	19.8	9.0	10.5	9.3
Boosted Feature Groupings	19.8	9.0	10.5	9.3
Manual Gram Mapping	20.7	9.0	10.8	9.6

Table A.2: Original Grammar Complexity Metrics for South Efate