Automatically Inferring Grammar Specifications for Valence-changing Verbal Morphology from Interlinear Glossed Text

Yi-Chien Lin

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

Master of Science

University of Washington

2023

Committee:
Emily M. Bender
Shane Steinert-Threlkeld

Program Authorized to Offer Degree:
Linguistics

University of Washington

Abstract

Automatically Inferring Grammar Specifications
for Valence-changing Verbal Morphology
from Interlinear Glossed Text

Yi-Chien Lin

Chair of the Supervisory Committee:
Professor Emily M. Bender
Department of Linguistics

This work builds upon the AGGREGATION project and extends it by adding an inference module targeting valence-changing verbal morphology. This module automatically collects information that answers the questionnaire of the valence-changing verbal morphology library in the LinGO Grammar Matrix. In this study, I show how the module identifies interlinear glossed text (IGT) items with valence-changing morphology and infers information about various types of valence-changing operations from a given IGT corpus. The module was tested on three development (Abui, Hiaki, and Wakhi) and three held-out (Hixkaryana, Old Javanese, and Yaoyos Quechua) languages, all of which are from different language families. The experiments show that the module is able to successfully infer information about multiple valence-changing operations. For all types of operations, the algorithm identifies the transitivity of the verbal stems; for valence-increasing operations, it collects additional information such as the position of the erstwhile subject/added argument on the complements list, the POS of the added argument, and the predicate. In this study, I also perform an extensive error analysis of the experiments, which reveals the limitations in both my module and valence-changing verbal morphology library.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF LISTINGS

ACKNOWLEDGMENTS

First and foremost, I am very grateful to my advisor Emily M. Bender. Thank you, Emily, for introducing me to the field of grammar engineering and assisting me to find the direction of this work. You have taught me how to conduct a research and become a better linguist. Most importantly, thank you for encouraging me when I felt frustrated, and telling me I can do it when I did not think I could.

I would also like to thank a number of people, from whom I received a lot of support throughout this thesis work. Shane Steinert-Threlkeld, thank you so much for being my second reader and providing your helpful feedback. Elizabeth Conrad, thank you for making the AGGREGATION setup smooth and easy, and for creating the evaluation tools which enabled me to gather the experimental results without getting lost. Allison Dods, thank you for sharing your experience in developing an inference module and answering my questions about developing one. Edi Xin, thank you for participating in the LURAP project and helping with me on treebanking. Tom Liu and Tara Wueger, thank you for organizing the data which made the development of this thesis work smooth. Kristen Howell, thank you for your work on BASIL, which this thesis builds on. Christian M. Curtis, thank you for developing the valence-changing verbal morphology library that this work interacts closely with. During the development of my thesis work, I also received a lot of support from the members of the Matrix+AGG developer group, EMB students, and the DELPH-IN community. Thank you. In addition, I also received a lot of help from the staff in the linguistics department. Brandon Graves, thank you for helping me with fixing the technical issues on patas. Joyce Parvi, thank you for patiently answering my tons of questions about the logistics of the program.

Lastly, I am so grateful to have my incredible family, friends, and partner around me. Thank you for the encouragement, food, and jokes that supported me through this process. Thank you.

DEDICATION

To dad and mom, 林景源 (Chǐng-Yuán Lín) and 張碧瑤 (Pì-Yáo Chāng)

Chapter 1

INTRODUCTION

The AGGREGATION project aims to automate the extraction of various grammatical properties of languages from their IGT corpora (Wax, 2014; Howell, 2020), and thereby generate machine-readable grammars through the LinGO Grammar Matrix (Bender et al., 2002, 2010). The machine-readable grammars can facilitate the performance of linguistic analyses by automating some parts of the analyses that are needed to be done manually. Prior to the point of the development of this thesis, the AGGREGATION system was able to provide inference outputs for different types of lexicons, and a variety of phenomena in morphology and syntax from an input IGT corpus (as developed by previous authors, including Zamaraeva et al. 2019; Howell 2020; Dods 2022). However, it was not able to extract information about valence-changing verbal morphology in languages despite this functionality being in the Grammar Matrix customization system (Curtis, 2018). The *valence* of a verb is the number of arguments it governs (Tesnière, 1959), and *valence-changing verbal morphology* refers to the morphemes that alter the valence of a verb (Haspelmath and Müller-Bardey, 2004). The goal of this thesis is to add the inference functionality for valence-changing verbal morphology to the AGGREGATION project.

I extend the inference system by adding a new inference module targeting valence-changing morphology.[1] In the context of the AGGREGATION project, the inference for this phenomenon must provide multiple pieces of information. For all valence-increasing and -decreasing operations, the pieces of information should include the type of the valence-changing operation and the transitivity of the verbal stems that the morphemes attach to;

[1]The module `infer_val_change.py` can be found in the AGGREGATION codebase: `https://git.ling.washington.edu/agg/aggregation`

for valence-increasing operations, additional pieces of information such as the position of specific arguments, the part of speech (POS), and the predicate are also required (Curtis, 2018).

The rest of this thesis is organized as follows. Chapter 2 reviews the relevant background on the AGGREGATION pipeline and how LinGO Grammar Matrix deals with valence-changing verbal morphology. Chapter 3 describes the implementation of the inference algorithm targeting the valence-changing verbal morphology. Chapter 4 shows the evaluation methodology I adopted to examine the performance of the module along with the information about the language data I used during the implementation and evaluation phases. Chapter 5 presents the results for the experiments and their error analyses. Finally, Chapter 6 provides a summary of this work and potential future directions for improving the system further.

Chapter 2
BACKGROUND

This thesis builds upon the AGGREGATION project and extends it by adding the inference functionality for valence-changing verbal morphology. Section 2.1 walks through the AGGREGATION pipeline and provides more information about its core phases. Section 2.2 shows how the LinGO Grammar Matrix deals with valence-changing verbal morphology and how the library responsible for this phenomenon relates to my module.

2.1 The AGGREGATION Project

The AGGREGATION project[1] aims to automatically extract grammatical properties of languages from their IGTs to generate machine-readable Head-driven Phrase Structure Grammars (HPSG; Pollard and Sag, 1994). Since this project contributes to the documentation of low-resource languages, the AGGREGATION system adopts a specific approach, *grammar inference*, to generate grammar automatically. Grammar inference is defined as "automatic grammar generation based on text annotated with partial grammatical information but not full parse trees or logical forms" (Howell, 2020, p. 6). Unlike high-resource languages, low-resource languages often do not have data such as treebanks or corpora containing sentences along with their logical forms. However, many low-resource languages are often documented in the form of IGT. Therefore, the IGT corpora are suitable inputs for the grammar inference approach (Howell, 2020).

Figure 2.1 shows an illustration of the AGGREGATION pipeline. As shown in Figure 2.1, the pipeline can be broadly divided into the following phases:

---

[1] `https://depts.washington.edu/uwcl/aggregation/`

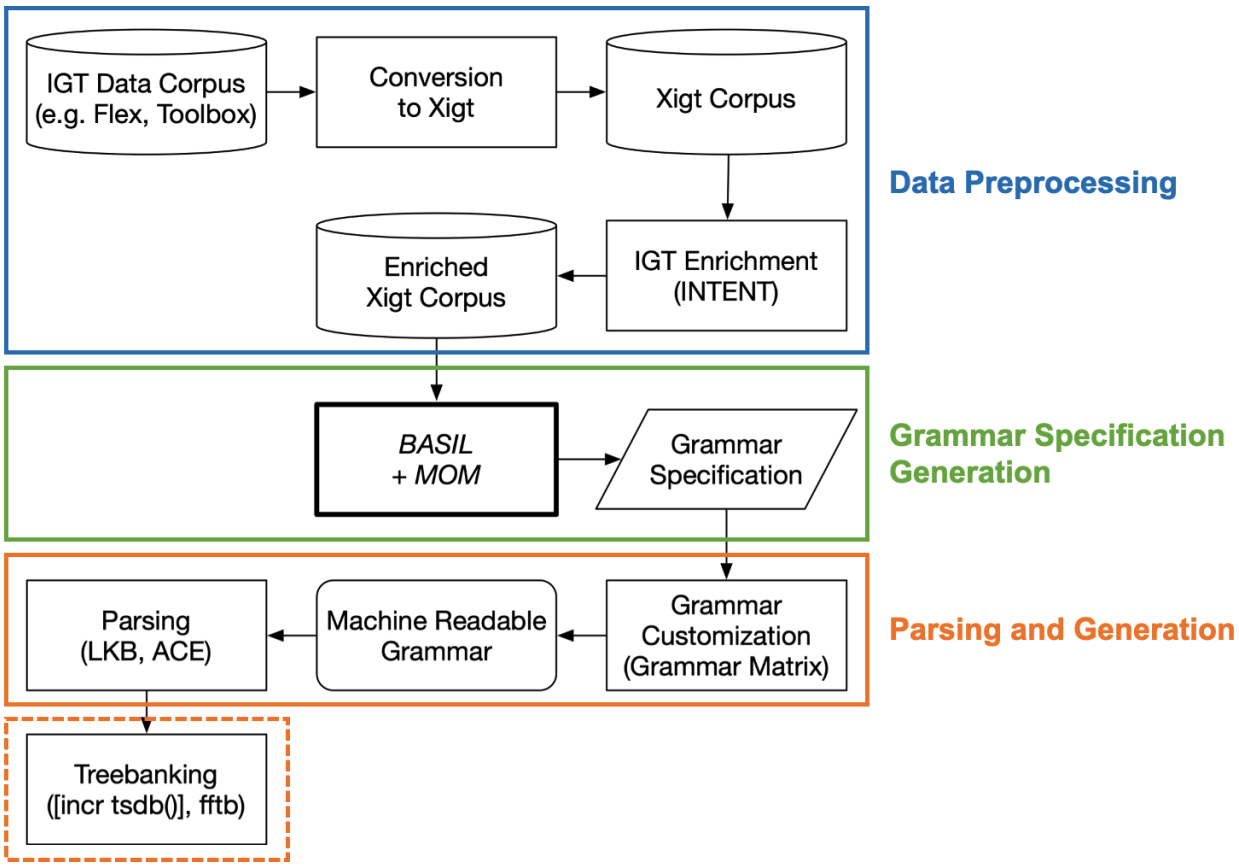

Figure 2.1: AGGREGATION pipeline, recreated from (Howell, 2020, p. 16)

1. **Data Preprocessing**: Accepting an IGT corpus of a given language and converting the corpus into data containing alignment, POS, and dependency information (Section 2.1.1).

2. **Grammar Specification Generation**: Extracting information about various linguistic phenomena for the input language, and creating a grammar specification file with the extracted information (Section 2.1.2).

3. **Parsing and Generation**: Generating a language-specific machine-readable grammar with the grammar specification file, and parsing sentences in the input language (Section 2.1.3).[2]

The algorithm I implemented is situated in the *grammar specification generation* phase, specifically in the step that infers information about linguistic phenomena in an input language. Detailed information about each phase is presented in the following subsections.

2.1.1 Data Preprocessing

This subsection describes the information contained in an IGT item and a series of data conversion an IGT corpus undergoes in order to be used in the *grammar specification generation* phase.

Interlinear Glossed Text

First, the AGGREGATION pipeline accepts an IGT corpus from a given language as its input. The IGT corpora used in the AGGREGATION project are usually provided in formats such as FLeX (Rogers, 2010) or Toolbox (SIL International, 2015). Following the guideline provided in the Leipzig Glossing Rules (Bickel et al., 2008), an IGT item usually contains

[2] *Treebanking* is an optional step in the *Parsing and Generation* phase. Section 2.1.3 provides more details on treebanking.

four parts – the *source language line*, the *morpheme-segmented line*, the *gloss line*, and the *translation*.

The source language line is a sentence in the documented language. This line also accompanied with a morpheme-segmented line. The morpheme-segmented line is followed by a gloss line, which shows the corresponding glosses of the morphemes. For the purpose of language documentation, an IGT also includes the translation of the source language text into a different language. In the context of the AGGREGATION project, the codebase requires translations into English. An example of IGT from Abui is shown in (1). As shown in (1), the morpheme *ong-* is a causative morpheme (indicated with CAUSE- in the gloss line) and *fak* is the root of the word *ongfak*.

(1) *na* *netoku* *ongfak*
na ne-toku ong-fak
1SG.AGT 1SG.AL-leg CAUSE-break
'I broke my leg.' (lit. I made my leg break.) [abz] (Kratochvíl, 2019)

Xigt

After being accepted as the input of the AGGREGATION pipeline, the input IGT corpus is then converted into the format that is applicable to the automated grammar specification generation process. In this data conversion process, the input IGT corpus is processed into a format based on XML, called Xigt (eXtensible Interlinear Glossed Text; Goodman et al., 2015). Listing 2.1 is the Xigt format of the IGT shown in (1). Different types of information in an IGT item are organized in *tiers* in Xigt. For example, the **m** tier contains all morphemes in the morpheme-segmented line and the source language words aligned to the morphemes (indicated with the `segmentation` attribute of each item in the **m** tier); the **g** tier contains the segmented glosses and the morphemes they are aligned to (indicated with the `alignment` attribute of each item in the **g** tier).

```
<tier id="w" type="words">
    <item id="w1">Na</item>
    <item id="w2">netoku</item>
    <item id="w3">ongfak.</item>
</tier>
<tier id="p" type="phrases">
    <item id="p1">na ne- toku ong- fak</item>
</tier>
<tier id="m" type="morphemes" segmentation="w">
    <item id="m1" segmentation="w1">na</item>
    <item id="m2" segmentation="w2">ne-</item>
    <item id="m3" segmentation="w2">toku</item>
    <item id="m4" segmentation="w3">ong-</item>
    <item id="m5" segmentation="w3">fak</item>
</tier>
<tier id="g" type="glosses" alignment="m">
    <item id="g1" alignment="m1">1sg.agt</item>
    <item id="g2" alignment="m2">1sg.al-</item>
    <item id="g3" alignment="m3">leg</item>
    <item id="g4" alignment="m4">cause-</item>
    <item id="g5" alignment="m5">break</item>
</tier>
<tier id="pos" type="pos" alignment="m">
    <item id="pos1" alignment="m1">pro</item>
    <item id="pos2" alignment="m2">n.pref-</item>
    <item id="pos3" alignment="m3">n</item>
    <item id="pos4" alignment="m4">lv.0-</item>
    <item id="pos5" alignment="m5">v.0</item>
</tier>
<tier id="t" type="translations" alignment="p">
    <item id="t1" alignment="p1">I broke my leg.</item>
</tier>
```

Listing 2.1: A Xigt snippet

Enriched Xigt

Finally, in the last step of this data preprocessing phase, the Xigt data converted from the previous step is then enriched. Specifically, the AGGREGATION pipeline uses INTENT (the INterlinear Text ENrichment Toolkit; Georgi, 2016) to provide additional information to the Xigt corpus. When this enrichment process is completed, information about word-alignment, POS tags, and dependency structures will be added to the input Xigt corpus.

For the first step of this enrichment process, by adopting a variety of heuristic and statistical approaches, INTENT aligns the source language words to the English translation words. After obtaining the word-alignment information, INTENT makes use of this alignment to project the available POS and dependency information in the English translation onto the source language since English is one of the most high-resource languages and is well-studied in tasks such POS and dependency tagging. The POS and dependency information in the English translation is obtained by using the POS tagger and parser in the `spaCy` library (Honnibal et al., 2020).

A snippet of enriched Xigt (enriched from the Xigt shown in Listing 2.1) is shown in Listing 2.2. Listing 2.2 does not show all information added by INTENT. Here, only information that is most relavent to this thesis is presented. The **w-ds** tier shows the dependency information about the source language line. As shown in the **w-ds** tier, **w3** (i.e., *ongfak*) is the root of the source language line and **w1** (i.e., *na*) is the subject of **w3**. INTENT adds useful information for the understanding of the source language structures. However, information about the source language dependency structures and the bilingual word alignment is sometime incomplete or absent. Therefore, since my algorithm heavily relies on the two pieces of information, the algorithm also performs additional word-alignment during the inference process. Detailed descriptions of the additional word alignment are provided in Section 3.2.2.

```
<tier id="tw" type="words" segmentation="t">
    <item id="tw1">I</item>
    <item id="tw2">broke</item>
```

```
    <item id="tw3">my</item>
    <item id="tw4">leg.</item>
</tier>
<tier id="w-ds" type="dependencies" data-creation-time="2020-01-29 08
   :38:07" data-method="project" data-provenance="INTENT2-2.0a6" dep="w"
   head="w">
    <item id="w-ds-dep1" dep="w1" head="w3">nsubj</item>
    <item id="w-ds-dep2" dep="w2" head="w3">nsubj</item>
    <item id="w-ds-dep3" dep="w3">root</item>
</tier>
<tier id="bilingual-alignments_a" type="bilingual-alignments" data-
   creation-time="2020-01-29 08:38:07" data-method="heuristic" data-
   provenance="INTENT2-2.0a6" source="tw" target="g">
    <item id="tw_g_aln_1" source="tw1" target="g1" />
    <item id="tw_g_aln_2" source="tw1" target="g2" />
    <item id="tw_g_aln_3" source="tw2" target="g5" />
</tier>
<tier id="bilingual-alignments_b" type="bilingual-alignments" data-
   creation-time="2020-01-29 08:38:07" data-provenance="INTENT2-2.0a6"
   source="tw" target="w">
    <item id="tw_w_aln_1" source="tw1" target="w2" />
    <item id="tw_w_aln_2" source="tw1" target="w1" />
    <item id="tw_w_aln_3" source="tw2" target="w3" />
</tier>
```

Listing 2.2: An enriched Xigt snippet

2.1.2 Grammar Specification Generation

Once the IGT corpus is converted into enriched Xigt corpus, the enriched Xigt corpus is then passed into the *grammar specification generation* phase. The grammar specification generation phase consists of two parts: *grammar inference* and *grammar specification generation*. The grammar inference part extracts grammatical properties from the input enriched Xigt

corpus, and the inference output generation part creates a file with grammar specifications. This file will then be passed to the next phase of the pipeline and the Grammar Matrix can use this file for the construction of a language-specific machine-readable grammar. Information about these two parts is presented in the following subsections.

Grammar Inference

In the grammar specification generation phase shown in Figure 2.1, the grammar inference part is represented as the bolded box. The grammar inference part involves two systems: MOM (Matrix-Odin Morphology system; Wax, 2014; Zamaraeva, 2016; Zamaraeva et al., 2017) and BASIL, a system for Building Analyses from Syntactic Inference in Low-resource languages (Howell, 2020). This thesis is situated in this grammar inference part and contributes specifically to extending BASIL.

MOM and BASIL infer different types of information. The two systems intertwine with each other to perform grammar inference. MOM deals with inferring verbal and nominal lexicons, and their morphotactic rules. On the other hand, BASIL extracts a variety of information about lexical items for other POS, syntactico-semantic features, morphotactics, and syntax. Since the development of the initial BASIL system (Howell 2020, which built on prior work in the AGGREGATION project such as Bender et al. 2014 and Zamaraeva et al. 2019), BASIL has been extended to reduce the ambiguity in inferred grammars (Conrad, 2021) and to perform automatic inference targeting adnominal possession (Dods, 2022). The extracted information from the two systems is output to a grammar specification file.

Inference Output Generation

Once the grammar inference part is completed, the inference output is then recorded in a grammar specification file (also referred to as a "choices file"), which is the final output of this grammar specification generation phase. The pieces of information stored in the grammar specification file are called "choices", which are the answers to the web-based questionnaire

in the LinGO Grammar Matrix customization system.[3]

There are different sections asking questions targeting different types of grammatical properties in the questionnaire. For example, in the *Argument Optionality* section (Saleem, 2010), the questionnaire asks for information such as whether subject dropping occurs with any verbs or only with certain verbs, and the possibility of object dropping. An example format of the grammar specification is shown in Listing 2.3. Listing 2.3 describes a language which allows subject-dropping to occur with any verb (i.e., `subj-drop=subj-drop-all`) and allows object-dropping (i.e., `obj-drop=obj-drop-all`).

```
section=arg-opt
subj-drop=subj-drop-all
subj-mark-drop=subj-mark-drop-opt
subj-mark-no-drop=subj-mark-no-drop-opt
obj-drop=obj-drop-all
obj-mark-drop=obj-mark-drop-opt
obj-mark-no-drop=obj-mark-no-drop-opt
```

Listing 2.3: A sample excerpt of a grammar specification for the argument optionality section

2.1.3 Parsing and Generation

Once the inferred information is collected, the information is then used by the LinGO Grammar Matrix (Bender et al., 2002, 2010) for the construction of a language-specific machine-readable HPSG grammar (Pollard and Sag, 1994). This grammar can be used to parse and generate strings in the language in question. Following subsections provide more details about the generation of grammars, and the software for parsing and generation in the AGGREGATION pipeline.

[3]Although the generation of the grammar specification file is fully-automated in the AGGREGATION pipeline, linguists can choose to enter the answers to the questionnaire for the LinGO Grammar Matrix customization system manually to generate a choices file.

LinGO Grammar Matrix

The file with the inference outputs serves as the input to the LinGO Grammar Matrix, which generates a language-specific machine-readable grammar. The LinGO Grammar Matrix includes different libraries targeting different linguistic phenomena. Specifically, each library is responsible for modeling a specific linguistic phenomenon. The libraries are designed to take in specifications for their phenomena and interact with each other to output analyses specific to the input language. For example, for the *Person* library (Drellishak, 2009), which deals with the grammatical distinction between the participants in a discourse (Siewierska, 2004), if a language does not distinguish between participants in a discourse, the generated grammar would pose no grammatical constraints on different discourse participants; on the other hand, if a language has first, second, and third person, the generated grammar would include grammatical constraints that distinguish between different person values.

Parsing Software

The generated grammar can then be used to automatically provide analyses for sentences. Sentences can be parsed by two pieces of software developed for grammars in the formalism used by the Grammar Matrix: the LKB (Linguistic Knowledge Building; Copestake, 2002) and ACE (Answer Constraint Engine; Crysmann and Packard, 2012). The analyses provided by the two pieces of software include the *readings* for an input sentence and their corresponding *semantic representations*. Given an input grammar, the readings are represented as the possible parse trees of the sentence. To represent the semantics for each reading, the pieces use the Minimal Recursion Semantics (MRS; Copestake et al., 2005) framework.

The LKB and ACE only accept one input sentence at a time. Due to the needs of researchers to perform large-scale linguistic analyses for the AGGREGATION project, other test suite managing software pieces are also used in this linguistic analysis process. Since large-scale linguistic analyses are required for the evaluation in this thesis, I also use two

software pieces: [incr tsdb()] (Oepen, 2001) and ART[4], which can interface with the LKB or ACE to parse a collection of sentences.

Treebanking Software

After parsing the test sentences, if researchers are interested in validating the analyses output by the parsing software, they may choose to proceed to perform treebanking on the parsed results. Treebanking refers to the process of determining the "correct" structural and semantic representation(s) of a sentence. This process can be performed by using the Full Forest Treebanker (FFTB; Packard, 2015).[5] The FFTB shows the differences between all possible parse trees in forms of lexical entries and rules. Therefore, when treebanking a sentence with multiple possible parses, users can eliminate invalid parse trees efficiently by rejecting inappropriate lexical entries/lexical and phrase structure rules. Additionally, in order to obtain the final parse tree(s) of a sentence, users also need to determine whether the semantic representation of a parse tree is appropriate. For this thesis, I performed treebanking as a part of the evaluation of the grammar inference part in the AGGREGATION pipeline. Detailed explanations are presented in Chapter 4.

2.1.4 Summary

This section walked through core phases of the AGGREGATION pipeline. I first described how the input IGT corpus is processed into enriched Xigt, the format used in the grammar specification generation phase. Then, I explained how grammar inference is performed and how the grammar specifications for the input language is stored. Lastly, I presented the steps within the parsing and generation phase, including the generation of a language-specific machine-readable HPSG grammar and ways of analyzing test sentences with the generated grammar. This thesis builds upon this pipeline, with specific focus on the grammar infer-

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

[5] https://github.com/delph-in/docs/wiki/FftbTop#Obtaining_it

ence part within the grammar specification generation phase. More information about the inference I add to the pipeline is covered in the next section (Section 2.2).

2.2 Valence-changing Verbal Morphology in the LinGO Grammar Matrix

My inference module aims to collect information about valence-changing verbal morphology. Specifically, in the context of the AGGREGATION project, this goal is to automatically fill out the LinGO Grammar Matrix questionnaire for the `valence-changing verbal morphology` section (Curtis, 2018). Therefore, my approach is designed specifically to extract information from a given language to answer the necessary questions in the valence-changing verbal morphology section.

Valence-changing verbal morphology refers to morphemes that increase, decrease, or reorder the obligatory argument(s) of a verb (Haspelmath and Müller-Bardey, 2004). For example, in English, the morpheme *-ed* passivizes *surprise*, as in *The cat surprises the girl*, into *surprised*, as in *The girl is surprised (by the cat)*. The passive morpheme *-ed* reorders the arguments of *surprise* (i.e., *the girl* and *the cat*). The current library for the valence-changing verbal morphology is implemented by Curtis (2018). The implementation of the library adopted the framework from Haspelmath and Müller-Bardey 2004. Based on the possible valence-changing operations mentioned in Haspelmath and Müller-Bardey's work, Curtis categorized the operations in terms of how the valence is altered (i.e., valence-increasing or -decreasing), and which arguments (i.e., subjects and objects) are affected. The types of operation implemented in this library include: *subject-adding (e.g., causative)*, *subject-demoting (e.g., passive)*, *subject-removing (e.g., anticausative)*, *object-removing (e.g., deobjective)*, and *object-adding (e.g., applicative)* operations. (2) shows an example of subject-adding operation (i.e., causative) in Vengo [bav] (Grassfields Bantu). As shown in (2b), the causative morpheme *-s* enables the addition of a new subject *m* 'I' of the verb *nìi* 'enter'. Since this library is designed specifically for valence-changing verbal morphology, structures such as periphrastic constructions are beyond the scope of both the library and this thesis.

(2) a. *nw nìi táa nìì*
nw nìi táa nìì
he enter in house

'He entered the house.' [bav] (Schaub, 1982, in Haspelmath and Müller-Bardey, 2004, p. 11)

b. *m nìis nw táa nìì*
m nìi-s nw táa nìì
I enter-CAUS him in house

'I made him enter the house.' [bav] (Schaub, 1982, in Haspelmath and Müller-Bardey, 2004, p. 11)

Figure 2.2 shows a snapshot of the valence-changing verbal morphology portion of the questionnaire for the LinGO Grammar Matrix customization system. The library locates in the `morphology` section of a grammar specification. Users can define a valence-changing operation after creating a lexical rule type within a position class. A *position class* is defined as "a collection of lexical rules, mutually exclusive in word-formation, that appear at a particular position in a word." (Goodman, 2013, p. 16); a position class consists of *lexical rule types*, which may define features and are associated with lexical rule instances giving the specific spelling of the affixes. The information in a valence-changing operation is conceptualized as a feature of the lexical rule type. To create a grammar specification for valence-changing verbal morphology, users need to provide the following information:

- **Valence-changing operation type**: *subject-adding (e.g., causative)*, *subject-demoting (e.g., passive)*, *subject-removing (e.g., anticausative)*, *object-removing (e.g., deobjective)*, or *object-adding (e.g., applicative)* operations

- **Transitivity of the verbal stems**: the transitivity of the verbal stems that the valence-changing morpheme attaches to.

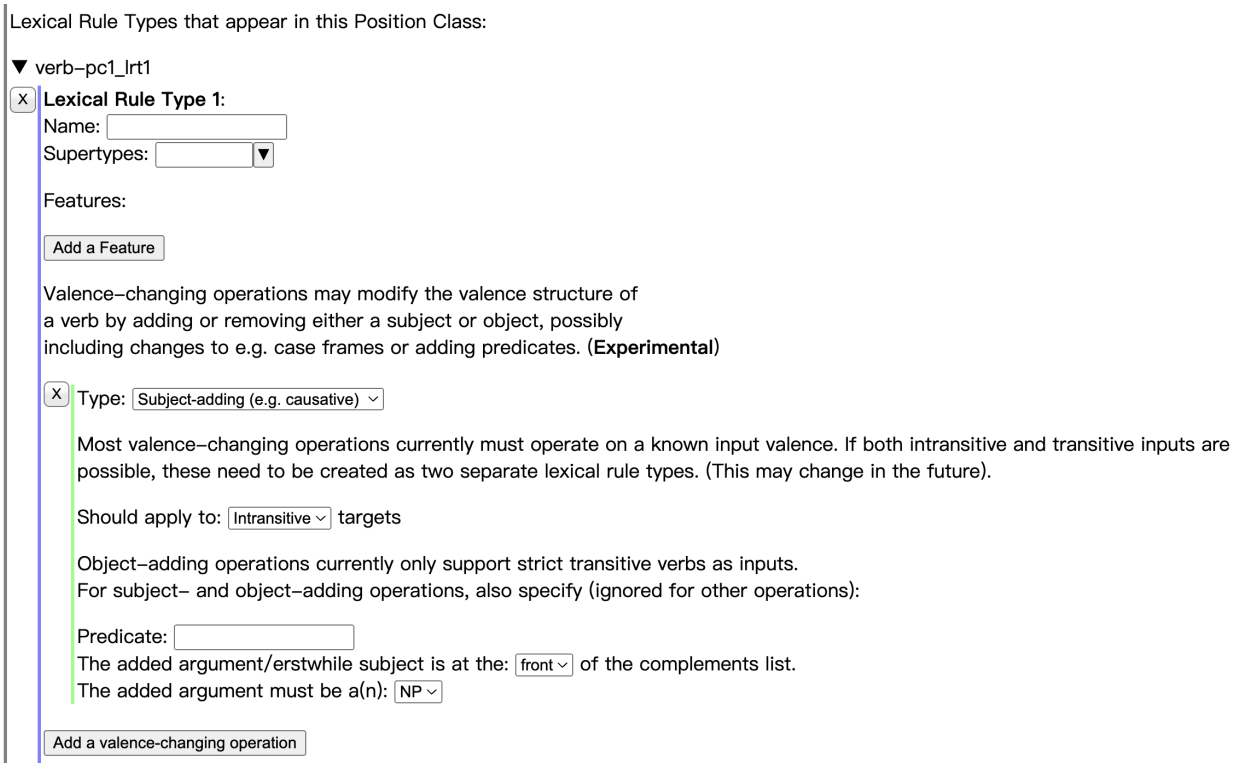

Figure 2.2: A snapshot of the valence-changing verbal morphology library in the LinGO Grammar Matrix customization system

- **Predicate information** (valence-increasing operations only): the semantic relation added by the valence-changing verbal morpheme.

- **Position of the *erstwhile subject* or *added argument* on the complements list** (valence-increasing operations only): the position of the *erstwhile subject* on the complements list for *subject-adding operations* or that of the *added argument* on the complements list for *object-adding operations*.

- **POS of the added argument** (valence-increasing operations only): *noun phrase* or *adpositional phrase*.

For all types of operation, users need to provide information about the *operation type* and the *transitivity of the verbal stems*. Note that for object-adding operations, only strict transitive verbs are supported in the current library (Curtis, 2018). For valence-increasing operations (i.e., subject- and object-adding operations), users need to provide additional information about: *predicate*, *position of the erstwhile subject/added argument on the complements list*, and the *POS of the added argument*. Here, a *complements list* (COMPS list) refers to a list containing the complements of a verb. For example, consider the English sentence *She cooked my sister a meal*. In this sentence, the complements of the verb *cooked* are *my sister* and *a meal*. The COMPS list of *cooked* is ⟨ **NP (my sister), NP (a meal)** ⟩, with the indirect object *my sister* at the front of the list and the object *a meal* at the end of it. The order of the complements in the COMPS list depends on the obliqueness of the complements to the head verb: the complement with the less obliqueness precedes the one with more obliqueness on the COMPS list.

An example for the valence-changing operation specification for (1) is shown in Listing 2.4. As shown in Listing 2.4, based on the example item in (1), Abui has a subject-adding operation (i.e., causative) that applies to intransitive verbal stems, and the position of the erstwhile subject *ne-toku* is at the front (represented as `pre` as the value for `argpos` in the

grammar specification) of the COMPS list.[6] For the added argument (i.e., *na* in (1)), it is a noun and therefore the value for `argtype` is `np`.

```
verb-pc36_lrt2_name=verb-pc36_lrt2
    verb-pc36_lrt2_valchg1_operation=subj-add
    verb-pc36_lrt2_valchg1_inputs=intrans
    verb-pc36_lrt2_valchg1_predname=causative
    verb-pc36_lrt2_valchg1_argpos=pre
    verb-pc36_lrt2_valchg1_argtype=np
    verb-pc36_lrt2_lri1_inflecting=yes
    verb-pc36_lrt2_lri1_orth=ong-
```

Listing 2.4: A sample grammar specification for valence-changing operation for (1)

The approach implemented in this thesis is designed specifically to extract the aforementioned information from an input IGT corpus, and only consider cases that are accounted in the valence-changing verbal morphology library.

2.3 Summary

This section describes where my thesis work is situated in the AGGREGATION project, and provided an overview of how the LinGO Grammar Matrix deals with the valence-changing verbal morphology. In the next chapter (Chapter 3), I explain the structure of my algorithm for inferring information about the valence-changing verbal morphology from a given language.

[6]Since the verbal stem *fak* is an intransitive verb, there is only one element on the COMPS list of the causative derived from it.

Chapter 3
IMPLEMENTATION

In this chapter, I present the implementation of the valence-changing inference module. Examples in this chapter are drawn from the development languages, principally Abui [abz]. For full details of the development and test corpora, see Section 4.5. An illustration of the overview of the algorithm is shown in Figure 3.1.[1] As shown in Figure 3.1, this algorithm contains three main stages: it first **identifies** the items with valence-changing morphology in the input enriched Xigt corpus (Section 3.1). Then, it **collects information** from those items (Section 3.2). Once the information is collected, it **outputs the grammar specifications** for valence-changing operations (Section 3.3). Information regarding each stage is detailed in the following sections.

3.1 Inference Stage 1: Target Items Identification

The first stage of the valence-changing morphology inference algorithm is to identify the items of interest in input enriched Xigt corpus. The operation types and specific operations considered in this thesis are shown in the first and second columns in Table 3.1. To identify items with the operations of interest, the algorithm looks at the gloss (**g**) tier in the enriched Xigt corpus. The **g** tier stores the information available in the gloss line of an IGT. A gloss line of an IGT consists of two elements: the *grams* and the *glosses*. A gram shows the grammatical information borne by the corresponding morpheme/word (e.g., case, gender, valence-changing operation, etc.); a gloss is the unrestricted translation of the corresponding

[1]The algorithm classifies items into two categories: items with valence-increasing verbal morphemes and items with non valence-increasing ones. The reason why the algorithm classifies items into the two categories is because the valence-changing verbal morphology library requires users to provide additional information specifically for valence-increasing operations. Details are provided in Section 3.2.

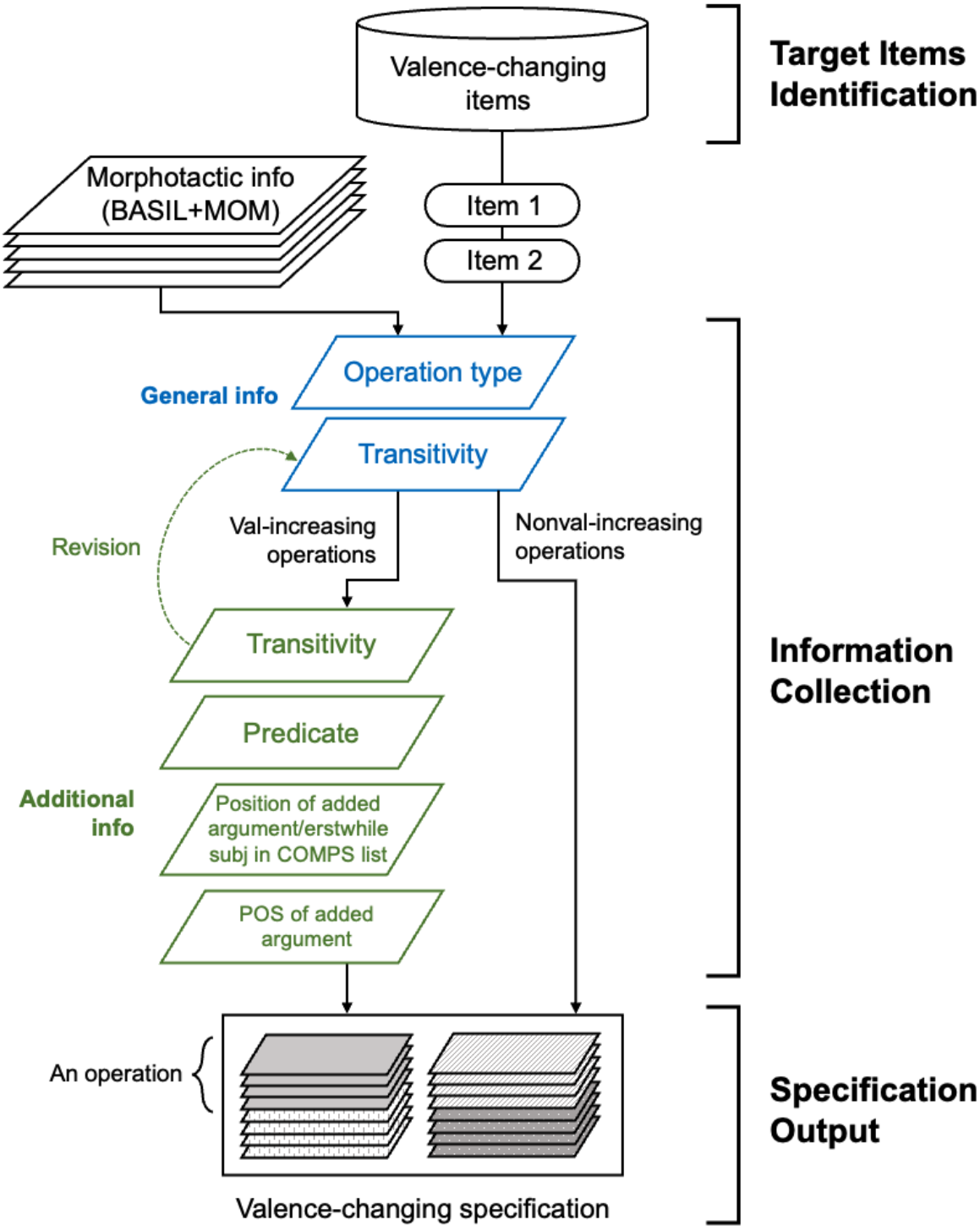

Figure 3.1: An overview of the valence-changing verbal morphology inference module

morpheme/word (Lewis, 2003). Precedent in the AGGREGATION project has made use of grams to extract grammatical information from an IGT (e.g., Bender et al. 2013; Howell 2020; Dods 2022). In my module, I also make use of a list of grams to identify items containing the valence-changing verbal morphology of interest. I generated this list of grams by taking the ODIN database (Lewis and Xia, 2010) and the GOLD ontology (Farrar and Lewis, 2007). The operation types (see Section 2.2) and their corresponding grams considered in the module is shown in the third column of Table 3.1. Once the algorithm identifies the grams that matches the ones of interest (i.e., finding any of the grams listed in Table 3.1 in the **g** tier), it cross-references the affixes with those inferred by MOM to make sure the grams are actually affixes attaching to verbal stems.

Operation Type	**Operations**	**Grams** (lowercased)
Subject-adding	causative	**caus**, **cause**
Subject-demoting	passive	**pass**, **passive**
Subject-removing	anticausative	**antic**
	resultative	**res**
Object-adding	applicative	**appl**, **apl**
	benefactive	**ben**, **benf**
Object-removing	deobjective	**deobj**
	absolutive antipassive	**abs**

Table 3.1: List of operations and grams considered in this thesis

During the process of identifying items of interest, items containing certain sentential structures are excluded and not taken into consideration in my inference module. In my module, for simplicity, imperative or coordinate sentences are not taken into consideration even if the item contains valence-increasing morphology since it is more complicated to identify the complements of the verbs in these constructions. These sentences are filtered out

based on their English translations. If the English translation of an IGT lacks a subject or contains conjunctions, it is filtered out from the IGTs being considered. In addition, the current algorithm does not handle sentences containing multiple valence-changing morphology. Therefore, such sentences are also filtered out during this identification process.

3.2 Inference Stage 2: Information Collection

After collecting the items of interest, as shown in Figure 3.1, the algorithm proceeds to the information collection stage. In this section, the **general information** is referred to as the information required for all types of operation when creating a valence-changing operation in a grammar specification. The general information includes the valence-changing operation type, and the transitivity of the verbal stems, as described in Section 2.2. For valence-increasing operations (i.e., subject- and object-adding operations), the valence-changing morphology library also requires additional information such as the predicate information, the position of the added argument/erstwhile subject on the COMPS list, and the POS of the added argument in the grammar specification. These pieces of information are referred to as **additional information** in this section.[2] The descriptions of extracting both types of information are presented respectively in the following subsections.

3.2.1 General Information

The general information includes the operation type and the transitivity of the verbal stems. The information about the operation type is identified by using the grams listed in Table 3.1. If any of the listed grams is found in the **g** tier, a new valence-changing operation will be created for the corresponding morpheme in the morpheme-segmented line. For example, (3) is an IGT from Abui and Listing 3.1 shows the **m** and **g** tiers of this IGT. In the **g** tier of (3), a gram for causative operation (i.e., CAUSE-) can be found, showing that (3) is an item with subject-adding operation. Once the algorithm finds a valence-changing gram in

[2]During the evaluation stage, I realized that I implemented the passive construction incompletely. Details are discussed in Section 5.2.2.

the IGT, a valence-changing operation is initialized for the corresponding morpheme, *ong-*. As described in Section 2.2, since a valence-changing operation should be added to existing lexical rule types, when creating grammar specifications in the final stage of this algorithm (Section 3.3), *ong-* serves as an anchor to its locate lexical rule type and this operation will then be added to that lexical rule type.

(3) *na* *Fanmalei* *haieng* *ongakuti*
na Fanmalei ha-ieng ong-akut-i
1SG.AGT name 3.INAL-eye CAUSE-close.eyes.PFV-PFV
'I made Fanmalei to close his eyes.' [abz] (Kratochvíl, 2019)

```
<tier id="m" type="morphemes" segmentation="w">
    <item id="m1" segmentation="w1">na</item>
    <item id="m2" segmentation="w2">Fanmalei</item>
    <item id="m3" segmentation="w3">ha-</item>
    <item id="m4" segmentation="w3">ieng</item>
    <item id="m5" segmentation="w4">ong-</item>
    <item id="m6" segmentation="w4">akut</item>
    <item id="m7" segmentation="w4">-i</item>
</tier>
<tier id="g" type="glosses" alignment="m" segmentation="gw">
    <item id="g1" alignment="m1" segmentation="gw1">1sg.agt</item>
    <item id="g2" alignment="m2" segmentation="gw2">name</item>
    <item id="g3" alignment="m3" segmentation="gw3">3.inal-</item>
    <item id="g4" alignment="m4" segmentation="gw3">eye</item>
    <item id="g5" alignment="m5" segmentation="gw4">cause-</item>
    <item id="g6" alignment="m6" segmentation="gw4">close.eyes.Pfv</item>
    <item id="g7" alignment="m7" segmentation="gw4">-Pfv</item>
</tier>
```

Listing 3.1: The **m** and **g** tiers of the IGT shown in (3)

Additionally, to collect the transitivity information about the verbal stem, the algorithm employs the valence information inferred by BASIL. Since this valence-changing inference module is run after BASIL's inference, the results for that previous processing are available to this module. If, according to BASIL, a verbal stem verb can be either transitive or intransitive, separate operations will be created respectively for both transitive and intransitive verbs. For instance, if the causative morpheme *ong-* in (3) can be used with both transitive and intransitive verbs, two subject-adding operations will be created, with one operation applied to transitive verbs and another applied to intransitive verbs.

3.2.2 Additional Information

After collecting the general information from all types of operations, items with valence-increasing morphemes are passed on to this additional process. Recall that the design of the valence-changing verbal morphology library requires users to provide general information described in Section 3.2.1 for all types of operations. For valence-increasing operations, the library requires users to provide additional information. An illustration of this additional process is shown in Figure 3.2. As shown in Figure 3.2, items with valence-increasing morphemes along with the transitivity of the verbal stems are passed to this additional process.

This process aims to collect the required three pieces of information for valence-increasing operations: the position of added argument/erstwhile subject on the COMPS list, the POS of the added argument, and the predicate information. Additional transitivity inference for the verbal stem is performed in this process since this information is essential for extracting the information about the position of the added argument/erstwhile subject on the COMPS list. Therefore, the valence information (i.e., the transitivity of the verbal stems) input from general information collection process might be revised during this process.[3]

[3]The reason why, in this algorithm, the valence information may be revised for verbal stems in sentences with valence-increasing operations and not other operations is because based on my observation of the development languages, the English translations of these operations are relatively fixed. Therefore, compared to other valence-changing operations, it is less complicated to use the English translations and the characteristics of the valence-increasing operations (see Section 3.2.2 for more details) to infer the valence information about the verbal stem.

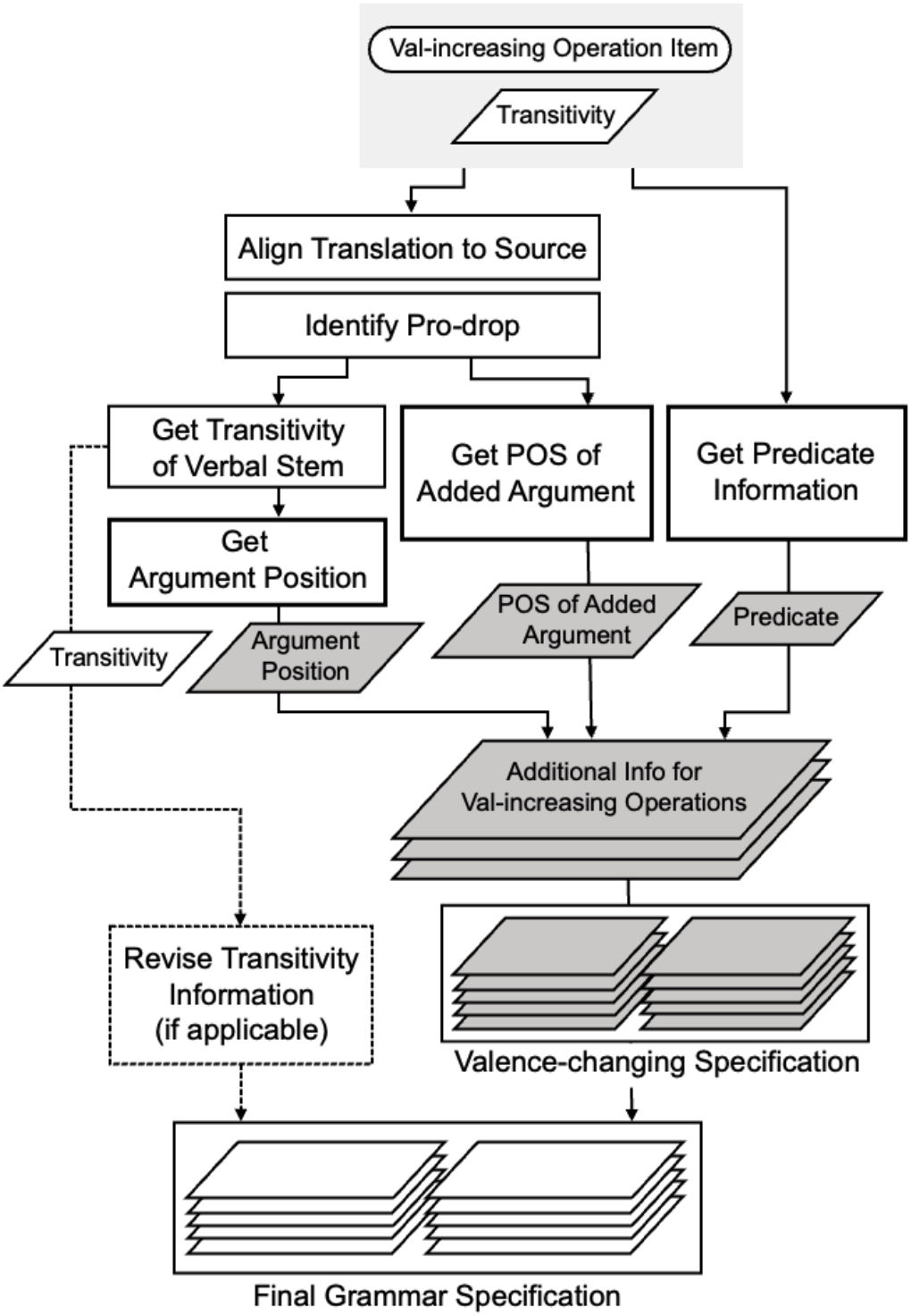

Figure 3.2: An illustration of the procedure for extracting additional information

Prior to collecting the information about the position of the added argument/erstwhile subject and the POS of the added argument, three intermediate steps, *aligning translation words to source language segments*, *identifying pro-drop*, and *getting transitivity information about the source language verbal stems*, need to be performed (indicated with the rectangular boxes with thin and solid outline in Figure 3.2). These three intermediate steps and the three main parts of collecting the additional information are detailed respectively in the following.

Aligning Translation to Source Language

In order to extract information about the position of the added argument/erstwhile subject on the COMPS list and the POS of the added argument, it is necessary for the algorithm to collect the dependency information about the source language first. Although as mentioned in Section 2.1.1, INTENT attempts to add a tier storing the bilingual alignment, this information may sometime be incomplete or absent. Therefore, this algorithm also needs to perform an extra step of bilingual alignment. Specifically, the algorithm achieves this goal by aligning the translation words to the segmented glosses. Since the alignment between the segmented source language line and the gloss line is always available, once the translation words are aligned to the glosses, the bilingual alignment can also be obtained. An illustration of the alignment process is provided in Figure 3.3.

As indicated on the right of Figure 3.3, the algorithm first starts by selecting the translation words that are to be aligned. Then, these words are aligned to the glosses. When selecting the translation words to be aligned, the algorithm searches for words with specific dependency relationship, `root`, `nsubj`, `dobj`, `dative`, `pobj`, and `ccomp`, since words with these dependencies are more likely to be the verbs or arguments of interest in the algorithm. The dependency information about the translation words is provided by the parser in `spaCy` library (Honnibal et al., 2020), and is available in the enriched Xigt. The target translation words can be categorized into three groups: pronouns, proper nouns, and others (i.e., other nouns and verbs). Different alignment methods are applied to each group.

To align the translation pronouns to the segmented gloss, the algorithm uses the gram-

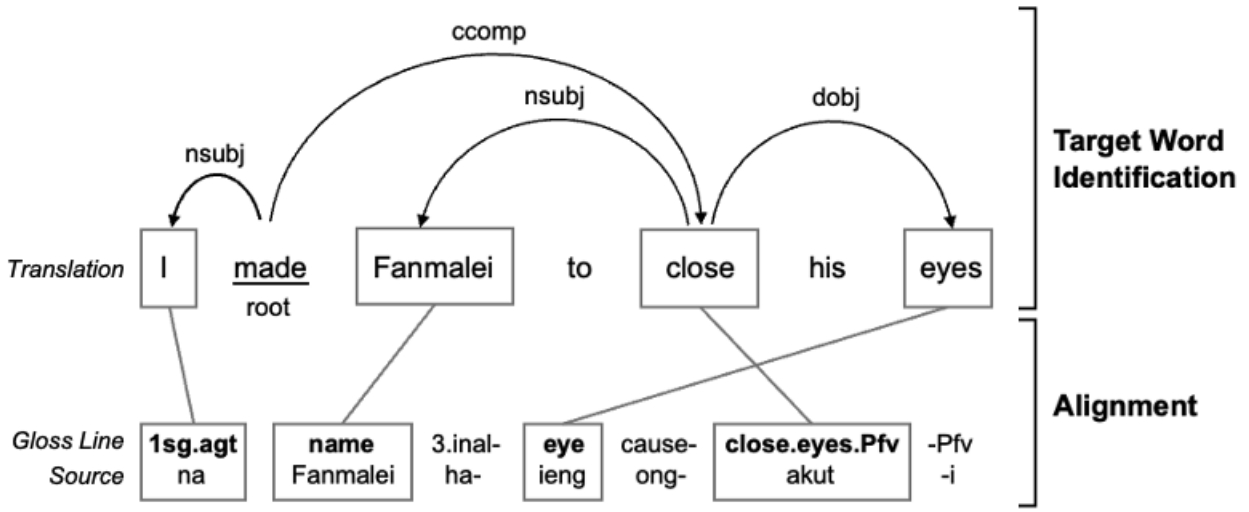

Figure 3.3: An illustration of the translation-to-source alignment of (3)

matical information available in the grams (e.g., person, number, or gender information) to align the two. For example, if a gram (i.e., a segment showing the grammatical information about the corresponding morpheme) contains an "1", indicating it is a first person pronoun, it would be aligned to a first person pronoun available in the translation line (e.g., *me*). If there are multiple pronouns with the same grammatical information in the input item, those pronouns would be aligned to the translation pronouns according to the order of occurrence of the translation pronouns: the pronoun occurring first in the gloss line would be aligned to the first pronoun occurring in the translation line, and so on.[4] To align proper nouns, if the POS tag of a translation word is *PROPN* (i.e., proper noun) and a gloss is *name*, the two would be aligned. Lastly, when aligning other translation words, the algorithm uses the GloVe word vectors (Pennington et al., 2014) to calculate the similarity between the gloss and the translation words. Specifically, the algorithm segments a gloss by words if there are multiple words in the gloss. For each gloss, the algorithm calculates the similarity between all translation words of interest and each word in that gloss. The translation word that has

[4]This is an arbitrary choice to simplify the alignment process.

the highest similarity with any of the words in the gloss is aligned to that gloss. Note that when aligning the words that belong to the *other* category, multiple translation words are allowed to be aligned to the same gloss since it possible that a gloss may be translated into separate words in the translation line.

After aligning the translation words to the gloss line, the alignment between the translation words and the segmented source language morpheme is immediately available, since the alignment between the segmented source language and gloss lines is always direct and one to one in the data. This alignment information is then used not only in one other intermediate step, getting the transitivity of the verbal stems in the input item, but also in the two main parts, getting the relative argument position on the COMPS list and getting the POS of the added argument.

Identifying Pro-drop

To extract information about the position of the added argument/erstwhile subject on the COMPS list and the POS of the added argument, it is necessary to identify whether the input item has pro-drop. The idea of identifying pro-drop is to compare the number of pronouns in the source language and those in the translation line. When identifying the pronouns in the source language, if an element in the **g** tier contains person, gender, or number information, and it corresponds to a stand alone morpheme, that corresponding morpheme is identified as a pronoun in the source language. After collecting the source language pronouns, the algorithm compares the number of the source language pronouns to that of the translation pronouns. If the number of source language pronouns is less that that of the translation pronouns, the input item is then identified as an item with pronoun-dropping. This information is then passed to the two main parts, getting the relative position of the elements in the COMPS list and the POS of the added argument, for further inference.[5]

[5]This method assumes that pro-drop in the source language always corresponds to a pronoun in the translation; however, it is possible that the amount of pronouns does not equal in the source language line and those in the translation line.

Getting Transitivity Information about the Verbal Stem

To collect information about the relative position of the elements on the COMPS list, it is crucial to know the transitivity of the verbal stems in the source language. If a verb is intransitive, having either *front* or *end* as the position of the added argument/erstwhile subject does not matter since the COMPS list contains only one element. The relative position of the elements on the COMPS list only matters when the COMPS list contains multiple elements. Although BASIL already performs transitivity inference for all verbal lexical items before this module is executed, this step is still necessary because of two reasons: (1) the aim of this algorithm is to collect the transitivity information about the verbal stems in items with valence-increasing morphemes instead of all items across the corpus, and (2) BASIL's inference output for valence information might be biased by the presence of the valence-changing morphemes (e.g., it is possible that BASIL infers an intransitive verb as transitive when that intransitive co-occurs with an object-adding morpheme).

When the algorithm moves on to this step, an item along with the transitivity of the verbal stem inferred by BASIL are input to this process. The input transitivity information could be "transitive", "intransitive", or "both" (Since the transitivity information from BASIL is inferred from the entire corpus, a verb can be inferred to be both transitive and intransitive). This input transitivity information might be revised due to the additional transitivity inference in this step. If the transitivity information inferred in this intermediate step is not available in BASIL's inference output, this information would be recorded and modified in the final stage (i.e., the specification output stage) of the algorithm.

The additional transitivity inference adopts a heuristic method by making use of the structure of the translation line. For valence-increasing operations, this thesis only focuses on specific types of operation, which have relatively fixed structures in the translation line of the items. Therefore, it is possible to infer the transitivity information from the translation line. For example, (4) shows translation structures which are most commonly seen for subject-adding constructions. The translations for subject-adding construction (i.e., causative con-

struction) can be roughly divided into two categories: the ones with *make/cause* in the translation (e.g., (4a) and (4b)) and the ones without the two words (e.g., (4c)). Assume that for items of which the translation lines are listed in (4), the verbal stems in the source language line are aligned respectively to *eat*, *dance*, and *broke*. For cases with *make/cause* in the translation, if the translation verbs aligned to the source language verbal stems have an object, the source language verbal stems are identified as transitive (i.e., (4a)); on the other hand, if the translation verbs aligned to the source language verbal stems do not have an object, as in (4b), the verbal stems are identified as intransitive. For cases like (4c), in which the verbal stem is aligned to the root of the translation, the source language verbal stem is identified as intransitive.

(4) a. I made my sister eat the apple/I caused my sister to eat the apple.

b. I made my sister dance/I caused my sister to dance.

c. He broke his leg. (lit. He made his leg break.)

Example translations of object-adding operation (applicative in this case) is shown in (5). The translation structures shown in (5) are the most common structures for applicative constructions in the development languages. If the translation verb aligned to the source verbal stem has one prepositional object or direct object (e.g., (5a)), the transitivity of the source verbal stem is identified as intransitive. On the other hand, if the translation verb has two objects (i.e., prepositional objects or indirect objects), as shown in (5b) and (5c), the source language verbal stem is identified as transitive.

(5) a. I am going to Japan.

b. The librarian gives the student a book.

c. The librarian gives a book to the student.

Getting Argument Position

After completing the three intermediate steps, the algorithm proceeds to this main part to get the argument's position on the COMPS list. Specifically, for subject-adding operations, this part gets the position of the erstwhile subject on the COMPS list; for object-adding operations, this part gets the position of the added argument on the COMPS list. Two values, `pre` (i.e., the front of the COMPS list) and `post` (i.e., the end of the COMPS list), are possible values for this specific piece of information. Note that the current library for the valence-changing verbal morphology is only able to deal with intransitive and transitive verbs. Therefore, after increasing the valence of a verb, the maximum number of arguments on the COMPS list is two.

The process begins by checking whether the input item contains pro-drop, which is inferred in an intermediate step, as shown in Figure 3.2 (see Section 3.2.2 for more details). If the pro-drop is found in the input item, the default position, `pre`, is set as the argument position for items with intransitive verbs and `post` is set as the default position for items with transitive verbs.[6] For items without pro-drop, the algorithm then checks the transitivity of the source language verbal stem. If the verbal stem is intransitive, the argument position is set to `pre`. On the other hand, if the verbal stem is transitive, the algorithm moves on to use the bilingual alignment for determining the argument position.

When determining the argument position for the transitive verbal stems in the source language, the algorithm checks the alignment of different dependencies for different types of operations. Specifically, for subject-adding operations, the algorithm collects the source language arguments that are aligned to the subject and direct object in the translation, and returns the position of the source language argument aligned to the translation subject. Between the two source language arguments, the one closer to the source language verbal

[6]For intransitive verbal stems, since there is only one argument on the COMPS list, the position of the erstwhile subject/added argument is always `pre`; for transitive verbal stems, the default position `post` is determined based on my observation of the development languages. Both subject-adding and object-adding operations in the development languages have the argument of interest at the end of the COMPS list for most cases when the verbal stems are transitive.

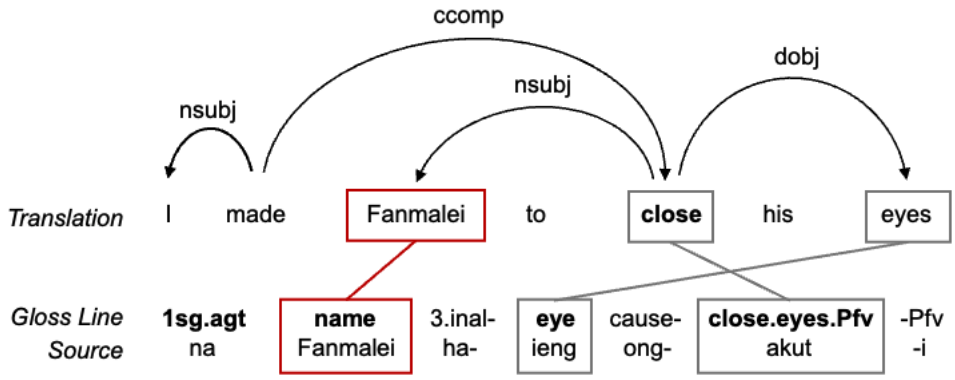

Figure 3.4: The source-to-translation alignment for subject-adding example (3)

stem is at the front of the COMPS list and another one is at the end of the list. For example, Figure 3.4 shows the alignment result for causative construction in (3). In this example, the source language verbal stem is aligned to *close* (indicated in boldface) in the translation line. In this case, the source language arguments on the COMPS list are the ones aligned to the translation subject of *close* (i.e., *Fanmalei*), and its direct object (i.e., *eyes*). After determining the arguments on the COMPS list for the subject-adding operation, the algorithm checks and returns the position of the source language erstwhile subject on the COMPS list, which is the source language argument aligned to *Fanmalei*. For this example, the COMPS list of the source language verbal stem is ⟨ **NP (ieng), NP (Fanmalei)** ⟩, with the less-oblique complement *ieng* 'eye' at the front of the list and the more-oblique complement *Fanmalei* at the end of the list. Therefore, the position of the erstwhile subject on the COMPS list for (3) is `post` (i.e., `argpos=post`).

On the other hand, for object-adding operations, the algorithm searches for the source language arguments which are aligned to the indirect/prepositional and direct objects of the root in the translation. The added argument in the source language is the element aligned to the indirect or prepositional object in the translation. An example IGT with applicative from Hiaki (Harley, 2019) is shown in (6). The bilingual alignment is shown in Figure 3.5. As shown in Figure 3.5, the source language verbal stem is aligned to *caught* (indicated in

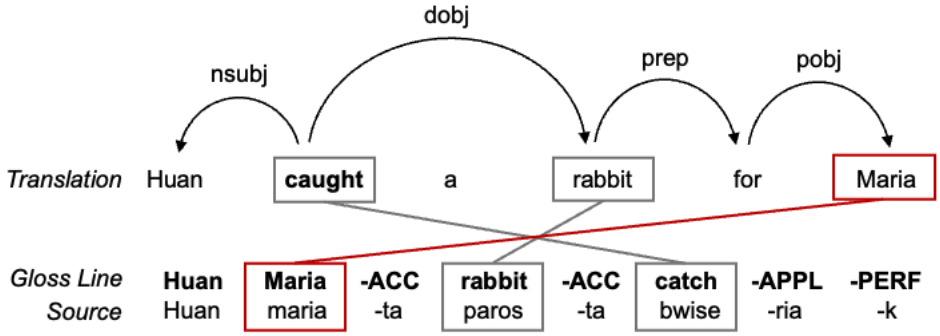

Figure 3.5: The translation of an object-adding example item

boldface) in the translation. The arguments on the COMPS list in the source language are the arguments aligned to *Maria* and *rabbit*, which are the prepositional and direct objects of *caught* in the translation line. In this example, the COMPS list of the source verbal stem *bwise* 'catch' is ⟨ **NP (paros), NP (maria)** ⟩ since *bwise* 'rabbit' is closer (i.e., less-oblique) to *bwise* and *maria* is further from it (i.e., more-oblique). The algorithm looks for the position of added argument *Maria* on the COMPS list of the source language verbal stem. Therefore, for this example, the position of the source language argument is `post` (i.e., `argpos=post`).

(6) *Huan mariata parosta bwiseriak*
Huan maria-ta paros-ta bwise-ria-k
Huan Maria-ACC rabbit-ACC catch-APPL-PERF

'Huan caught a rabbit for Maria.' [yaq] (Harley, 2019)

Getting POS of Added Argument

The second main part of the process of collecting additional information is to get the POS of the added arguments. Similar to the process of getting the position of the designated arguments on the COMPS list, this part also makes use of the structures of the translation

to get the added argument in the source language and then get their POSs in the source language. The current valence-changing verbal morphology library allows two possible POS values, `np` (i.e., noun phrase) and `pp` (i.e., adpositional phrase). For subject-adding operations, the added arguments in the source languages are the ones aligned to the subject of the translation root. For example, for items with the translations in (4), which shows the most common translation structures for causative constructions, the added arguments are be the source arguments aligned to *I* in (4a), *I* in (4b), and *he* in (4c). On the other hand, for object-adding operations, as described in the previous main step (i.e., Getting Argument Position), the added arguments are the source language arguments aligned to the indirect or prepositional objects of the root in the translation.

After identifying the added argument in the source language, the algorithm uses the adposition information inferred by BASIL. The algorithm collects the adpositions inferred by BASIL and checks the elements nearby the added argument. If an adpostion can be found nearby the added argument, the POS of the added argument would be `pp`; if no adposition can be fonud nearby, the POS of the added argument would be set to `np`.[7]

Getting Predicate Information

The algorithm gets the predicate value based on the valence-increasing operation type. A designated predicate value is set for each valence-increasing operation considered in this module. Specifically, the predicate for causative (subject-adding), applicative (object-adding), and benefactive (object-adding) constructions are respectively `causative`, `applicative`, and `benefactive`. This predicate symbol is associated with the semantic relation added by the valence-changing operation, which in turn allows the arguments involved in the valence-increasing operations to have their roles in the semantic representation.

[7]Here, "nearby" refers to the position immediately next to the added argument.

3.2.3 Final Valence-changing Operation Specification Determination

In the previous subsections, I first presented an overview of the information collection process for valence-changing operations. I then described the methods used for collecting information required all types of operations (i.e., the *general information*). In addition, I also detailed the algorithm I implemented to collect the *additional information* required only for valence-increasing operations.

Once the information is collected, the algorithm organizes the collected information. Specifically, the organization focuses on two aspects: merging valence-changing operations and organizing the revised valence information. When merging the operations, to avoid creating ambiguity, for each operation type of each valence-increasing morpheme, the algorithm keeps a unique operation for each type of input (i.e., intransitive or transitive).[8] For example, the algorithm disallows the grammar specification shown in Listing 3.2, where the subject-adding operation of the valence-changing morpheme *ong-* allows the erstwhile subjects to occur at both the front (`argpos=pre`) and the end (`argpos=post`) of COMPS list of transitive verbal stems.[9] Therefore, when multiple argument positions are possible for an operation applying to transitive verbs, the algorithm selects the final argument position with majority voting. If, for a specific morpheme, there are more items with the argument of interest occurring at the front of the COMPS list, `pre` is selected as the argument position; if there are more items with the argument occurring at the end of the COMPS list, `post` is selected as the final value of the argument position.[10] If there is a tie between the two

[8]Each operation collected from the corpus is stored with the co-occurring morpheme. A morpheme may have both operations that take transitive and intransitive inputs. Before this checking process, it is also possible for the operation taking transitive inputs to have both `pre` and `post` as the value of argument position.

[9]The algorithm only checks the valence-increasing operations that take transitive inputs since the argument position of those taking intransitive inputs will always have the arguments at the front (i.e., `pre`) of the COMPS list.

[10]Note that as mentioned in 3.2.1, the valence-changing morpheme serves as an anchor to locate the lexical rule type that the valence-changing operation belong to. Therefore, this checking process is performed for all valence-increasing operations within the same lexical rule type.

positions, `post` is selected as the default position.[11]

```
verb-pc36_inputs=verb8, verb209
    verb-pc36_lrt1_name=verb-pc36_lrt1
        verb-pc36_lrt1_valchg1_operation=subj-add
        verb-pc36_lrt1_valchg1_inputs=trans
        verb-pc36_lrt1_valchg1_predname=causative
        verb-pc36_lrt1_valchg1_argpos=post
        verb-pc36_lrt1_valchg1_argtype=np
        verb-pc36_lrt1_valchg2_operation=subj-add
        verb-pc36_lrt1_valchg2_inputs=trans
        verb-pc36_lrt1_valchg2_predname=causative
        verb-pc36_lrt1_valchg2_argpos=pre
        verb-pc36_lrt1_valchg2_argtype=np
        verb-pc36_lrt1_lri1_inflecting=yes
        verb-pc36_lrt1_lri1_orth=ong-
```

Listing 3.2: A sample of operation disallowed by the algorithm

In addition, the algorithm also organizes the valence information that needs revision by gathering the verbal stems with the same valence that needs to be revised. The verbal stems are stored along with the valence-changing morphemes they co-occur with. Similar to the morphemes' role to the collected valence-changing operations, these morphemes also serve as anchors for the module to locate the position classes and the lexical rule types in which the revised information needs to be added. Section 3.3 provides more details on how the valence information is added to the final grammar specification. Once the information is merged and organized, the organized information is passed to the next stage of the module to output the valence-changing specifications.

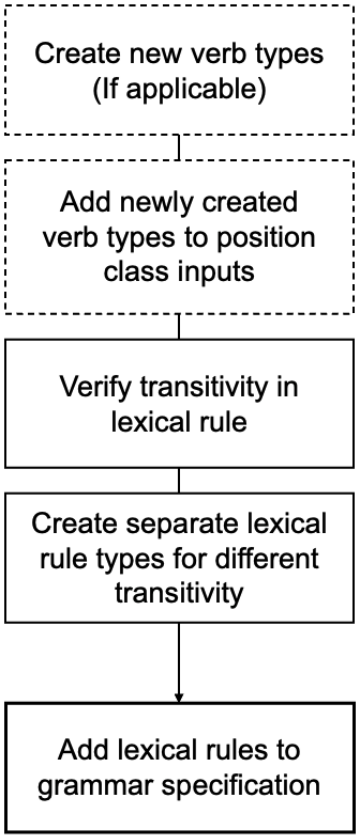

Figure 3.6: The procedure of grammar specification output stage in Figure 3.1

3.3 Inference Stage 3: Specification Output

As shown in Figure 3.1, the final stage of the valence-changing morphology inference module is to add the collected information to the grammar specification file. Figure 3.6 provides an overview of the steps in this final stage. The first step in this stage, if applicable, is to create new verb types for the verbal stems whose transitivity is modified during the information collection stage, as described in Section 3.2.2.[12] In this step, based on the valence revision information organized in the previous step, all verbal stems with the same transitivity are put together in the same newly created verb types. For example, if both transitive and intransitive verbal stems need to be added to the existing grammar specification, a new verb type will be created for all new transitive verb lexical entries and another new verb type will be created for all new intransitive verb lexical entries. The existing verb types from BASIL's inference output are maintained as is.[13] If new verb types are created, as shown in Figure 3.6, the next step is to make the valence-changing operations applicable to the new verb types. To enable the applicablility of the operations to the new verb types, the algorithm first locates the position classes containing the morphemes used with the new lexical entries in the new verb types; then, the algorithm adds the verb types to its inputs.[14]

The next step in this stage is to verify the transitivity of the verbal stems derived in the previous stage. If no new verb types are created, the grammar specification output stage starts from this step. In this step, the algorithm checks whether the transitivity information about the verbal stems collected in the previous stage is consistent with that of the inputs

---

[11]This is an arbitrary choices based on my observation of the development languages. For most transitive verbs in the development languages, the verbs have the erstwhile subject/added argument at the end of their COMPS list in valence-increasing constructions.

[12]If, after the information collection stage, no transitivity information is modified, the first two steps indicated in Figure 3.6 will be skipped.

[13]The module does not modify the existing verb types because those verb types might be associated to the specification in other libraries and I want to avoid any potential conflicts.

[14]As described in Section 3.2.3, the revised verbal stems are stored with the valence-changing morphemes they co-occur with so the module is able to use this information to locate the position classes with the morphemes.

to the position classes containing the valence-changing operation. If a valence-changing operation is inferred to be applicable to intransitive verbs but intransitive verbs are not possible inputs to the position class that should contain this operation, this operation will not added to the grammar specification since this might cause conflict when generating the final grammar.[15]

In addition, when adding the information collected from the previous stage, the algorithm also checks whether the valence-changing operations are applicable to verbs with different transitivity. Until this step, some valence-changing morphemes may be stored with the same type of operation that has both transitive and intransitive inputs. For example, the valence-increasing *ong-* in (3) might have subject-adding operations for both transitive and intransitive verbs. If the algorithm adds the two operations to the lexical rule type based on the the location of the morpheme *ong-*, the two subject-adding operations would be added to the same lexical rule type. However, due to the valence-changing morphology library's current way of assembling the rule components (see Section 5.2.2 and 5.2.3 in Curtis (2018) for more details on this), if the same type of operation is applicable to both transitive and intransitive verbs, the two operations need to be created in separate lexical rule types. In other words, the current valence-changing morphology library does not handle the situation where a lexical rule type contains operations applicable to both transitive and intransitive verbs.

Once the information collected in the previous stage is fully checked, new operations are added to the existing grammar specification file. This file is then passed on to the next phase of the AGGREGATION pipeline for the generation of a machine-readable grammar. Listing 3.3 shows a grammar specification snippet for (1) and (3) (the same IGTs are listed here again for clearer illustration). (7) is an example of an intransitive verb *fak* 'break' attaching to a causative morpheme *ong-* while (8) shows an example of a transitive verb *akut* 'close

[15]I tried to minimize the changes to the existing inference output. Therefore, in this case, I chose to drop the operation instead of moving the verb with inconsistent valence information to a different verb type with consistent valence information.

eye' attaching to the morpheme. Since *fak* is inferred by BASIL as transitive, an additional intransitive lexical entry needs to be created for it. This new lexical entry for *fak* is added to the grammar specification by creating a new intransitive verb type (i.e., `verb209`) and adding a new entry to the verb type. Listing 3.3 also shows the case in which the valence-changing morpheme *ong-* can be attached to both transitive (i.e., *akut*) and intransitive (i.e., *fak*) verbs. In this case, two separate lexical rule types (i.e., `verb-pc36_lrt1` and `verb-pc36_lrt2`) are created for operations taking verbs with different transitivity type.

(7) *na* *netoku* *ongfak*
na ne-toku ong-fak
1SG.AGT 1SG.AL-leg CAUSE-break

'I broke my leg.' (lit. I made my leg break.) [abz] (Kratochvíl, 2019)

(8) *na* *Fanmalei* *haieng* *ongakuti*
na Fanmalei ha-ieng ong-akut-i
1SG.AGT name 3.INAL-eye CAUSE-close.eyes.PFV-PFV

'I made Fanmalei to close his eyes.' [abz] (Kratochvíl, 2019)

```
verb209_name=verb209
  verb209_valence=intrans
    verb209_stem1_orth=fak
    verb209_stem1_pred=_break_v_rel

verb-pc36_inputs=verb8, verb209
  verb-pc36_lrt1_name=verb-pc36_lrt1
    verb-pc36_lrt1_valchg1_operation=subj-add
    verb-pc36_lrt1_valchg1_inputs=trans
    verb-pc36_lrt1_valchg1_predname=causative
    verb-pc36_lrt1_valchg1_argpos=post
```

```
    verb-pc36_lrt1_valchg1_argtype=np
    verb-pc36_lrt1_lri1_inflecting=yes
    verb-pc36_lrt1_lri1_orth=ong-
  verb-pc36_lrt2_name=verb-pc36_lrt2
    verb-pc36_lrt2_valchg1_operation=subj-add
    verb-pc36_lrt2_valchg1_inputs=intrans
    verb-pc36_lrt2_valchg1_predname=causative
    verb-pc36_lrt2_valchg1_argpos=pre
    verb-pc36_lrt2_valchg1_argtype=np
    verb-pc36_lrt2_lri1_inflecting=yes
    verb-pc36_lrt2_lri1_orth=ong-
```

Listing 3.3: A sample of operation disallowed by the algorithm

3.4 Summary

This section presented and detailed the algorithm for inferring information about valence-changing morphology, including identifying target items, collecting information required for different types of operations, and outputting the inferred information to the grammar specification file. In the next section, I describe the evaluation methodology I used to examine the performance of my inference module.

Chapter 4
EVALUATION METHODOLOGY

In this chapter, first, I show the language corpora I used in this thesis. Then, I present the metrics used for the evaluation of my algorithm. My module was evaluated on a set of selected languages. Each language was evaluated from three aspects: *parsing* (Section 4.2), *treebanking* (Section 4.3), and *grammar specification output* (Section 4.4). In addition to the description of the evaluation metrics, this section also presents the dataset statistics of each language corpus (Section 4.5).

4.1 Language Data

Precedents in the AGGREGATION project have developed the inference system in a data-driven fashion (e.g., Howell 2020; Conrad 2021; Dods 2022). That is, throughout the implementation, developers regularly test their changes to the system on a selected set of languages. This data-driven approach was also adopted for the implementation of my algorithm. I developed the algorithm on three selected *development languages*, and also used these languages to evaluate the performance of my algorithm. To examine the whether the algorithm is generalizable enough across languages, I also tested the algorithm on three *held-out languages*.

The three development languages include Abui (abz; Trans-New Guinea; Kratochvíl 2019), Hiaki (yaq; Uto-Aztecan; Harley 2019), and Wakhi (wbl; Indo-European; Kaufman et al. 2020).[1] These three languages were chosen from the corpora available to the AGGREGATION project. Specifically, I followed two criteria when selecting the languages: (1) the

[1]For all development and test languages mentioned in this section, I show their ISO 639-3 codes, their language families, and the corpora used in this thesis.

selected language corpora should contain IGT items with valence-changing morphology, and (2) these languages should belong to different language families. I randomly drew languages until there are three languages that meet the criteria.

In addition to the development languages, I also followed the aforementioned criteria to choose five held-out languages from the corpora available to the AGGREGATION project. Among the five test languages, three of them were selected as the initial held-out languages. However, after running the current version of system on the first three held-out languages, my module generated no additional inference for two held-out languages. Therefore, to better analyze the performance of my system, I selected two additional held-out languages. The newly selected languages do not belong to the families of the languages which my module is able to provide additional inference. The initially selected held-out languages selected for this project are: Hixkaryana (hix; Cariban; Meira 2020), Chintang (ctn; Sino-Tibetan; Bickel et al. 2013), and South Efate (erk; Austronesian; Thieberger 2006), and my module is not able to produce additional inference for the latter two languages.[2] The newly selected held-out languages are Old Javanese (jav; Austronesian; Arci 2023) and Yaoyos Quechua (qux; Quechuan; Shimelman 2012). Unlike the development languages, the held-out languages were not viewed during the implementation phase; they were viewed only after I froze the development of my module.

The performance of my module was evaluated on both development and held-out languages. Statistics regarding the training dataset size of each language and the number of instances with valence-changing morphology is presented in Section 4.5.

4.2 Parsing

As described in Chapter 2, this thesis contributes to the grammar specification generation phase of the AGGREGATION pipeline. The pipeline thereby creates machine-readable grammars with the grammar specifications for the analysis of sentences in the languages in

[2]Therefore, the additional held-out languages cannot be in the language families: Trans-New Guinea, Uto-Aztecan, Indo-European, and Cariban.

question. Therefore, a direct way of examining the performance of my module is to look at the parsed results for the generated grammar. Specifically, for this evaluation aspect, I look at the overall *coverage* and *ambiguity* of the parsed results. Descriptions of the process of using the two metrics and details of the two metrics are presented in the following subsections.

4.2.1 Cross Validation

When calculating the two metrics, I follow the tradition of the AGGREGATION project to use *cross-validation* (Bender et al., 2014; Zamaraeva et al., 2019; Howell, 2020; Conrad, 2021; Dods, 2022). The idea of cross-validation is to divide the corpus of each language into ten folds. For each language, the system is then evaluated for ten rounds. For each round of the experiments, nine portions of the corpus are used to perform grammar inference (referred to as a *training fold*); the remaining one portion is used to evaluate the system performance (referred to as a *test fold*). Each of the ten folds would be selected as the test fold in one of the experiments.

4.2.2 Overall Coverage and Ambiguity

In the context of the AGGREGATION project, the *coverage* refers to the number of items that can be analyzed by the generated grammar. In Chapter 5, this metric is represented as the percentage of the amount of parsed sentences over the size of the test fold. On the other hand, *ambiguity* refers to the average number of analyses among sentences with any analysis.

The sentences to be evaluated can be parsed by the generated grammar with the parsing software: the LKB (Copestake, 2002) or ACE (Crysmann and Packard, 2012). After collecting the parsed results across the ten rounds of experiment, the two metrics are then examined using the evaluation package developed by Elizabeth Conrad.[3]

[3]This package is developed on Conrad's work in the AGGREGATION project (Conrad, 2021): `https://git.ling.washington.edu/agg/aggregation-evaluation`

4.3 Treebanking

Although the overall coverage and ambiguity provide a general idea of the system performance, the two metrics are unable to show whether the parses include correct analyses. Therefore, to validate the analyses provided by the generated grammar, treebanking is also performed. I use the FFTB (Packard, 2015) to treebank sentences that can be analyzed by the grammar. More specifically, I use the FFTB to determine whether any of the analyses of a sentence include appropriate semantic representations.

Since it would be prohibitively time-consuming to treebank all sentences across the test folds in the context of this project, only a subset of sentences is treebanked. This thesis follow the evaluation metric used in Dods' work for adnominal possession (Dods, 2022), the most recent inference work in the AGGREGATION project. In her work, for each language she evaluated, she makes a possession-specific profile from the first fold such that the profile contains only IGT items with adnominal possession. I also follow her method for validating the coverage by creating a separate test profile. Specifically, to examine the effectiveness of the algorithm in producing inference and that of the generated grammar in providing correct analyses for items with valence-changing morphology, I arbitrarily selected a test fold from each language such that both training and test folds contain the items of interest.[4]

After deciding the test fold to evaluate for this metric, I selected the IGT items containing valence-changing morphology from this fold. Similar to what Dods does in her evaluation metric for validating coverage, I also treebank all items in this profile with at least one analysis. Due to the focus of this thesis, I only validate the analysis of the valence-changing morphology. Therefore, I accept analyses that are correct for the valence-changing morphology parts but have other independent errors. The validated coverage is represented as the number of items that can be correctly analyzed for the valence-changing morphology part

[4]Some language corpora contain very few items with valence-changing morphology (e.g., Wakhi [wbl], as shown later in Table 4.1). Therefore, there might be cases in which the test folds do not contain items with valence-changing morphology. For all languages except for Old Javanese, the fifth folds were selected for treebanking. The first fold is selected for Old Javanese since its fifth fold contains only one sentences, and there are more sentences of interest in the first fold.

out of the total number of items in the test profile.

Following Dods' (2022) evaluation procedure for this metric, if the number of items with correct analyses is fewer than ten, I proceed to evaluate other items in this test profile. For each language, this evaluation process is completed when (1) the items being evaluated are ten in total, or (2) the entire profile is evaluated.[5] To complete the evaluation process in this case, I first randomly select items without analyses from the test profile. Then, for the evaluation of these items, I use the LKB (Copestake, 2002) to analyze the cause of the parse failure. In particular, I examine whether the parse failure is caused by my inference output.

4.4 Grammar Specification Output

In addition to evaluating the effectiveness of my inference module from the parsing and treebanking aspects, I also evaluate the grammar specification outputs by following Dods' (2022) methodology for this metric: I examine how consistent the grammar specifications generated by my module is with the expected grammar specifications, which are constructed based on the descriptive grammar resources. For the purpose of the thesis, the expected grammar specifications focus specifically on the valence-changing operations. Since the generation of the parsing and treebanking results involve multiple components in the AGGREGATION pipeline, the inference system and the Grammar Matrix, it can sometimes be difficult to tease the effect of the two components apart by only examining the parsing and treebanking results. Therefore, to evaluate the performance of my inference module itself, it is also necessary to evaluate the inferred grammar specifications from this evaluation aspect. For this metric, I choose to evaluate the grammar specification corresponding to the fold I evaluate for the treebanking aspect.

When constructing the expected grammar specifications, I only create grammar specifications for valence-changing operations available in the corpora.[6] For example, if, according

[5]Here, a sentence *being evaluated* refers to the situation where that setnence is either treebanked with FFTB or examined with the LKB.

[6]I identify the presence of the operations with the grams listed in Table 3.1 and do not consider other

the a descriptive grammar, a language has both causative and applicative constructions but the former is present in the language corpus and the latter is not, only grammar specifications for causative operations will be constructed. To construct the expected grammar specifications, I consult the descriptive grammar resource for each language. Additionally, I also manually inspect the IGT items with valence-changing morphology since some information required for creating a valence-changing operation might not be covered in the descriptive grammars. For the held-out languages, I create the grammar specifications before viewing the system generated grammar specifications.

4.5 Dataset Statistics

This section provides the information about the training dataset size of the development and held-out languages, which is shown in Table 4.1.[7] For each language, the table presents a summary of the average size of one training fold and the average occurrence of valence-changing morphology within one training fold. The valence-changing morphemes taken into consideration are listed in Table 3.1. These numbers do not imply the actual amount of sentences with valence-changing morphology since a sentence can contain multiple valence-changing morphemes.

4.6 Summary

This chapter laid out my evaluation methodology, including quantitative and qualitative evaluation plans. I also provided some basic statistics showing the dataset size of each language. In Chapter 5, I report the results for the experiments on each language along with the analyses for the results.

---

valence-changing operations represented by the grams not listed in the table.

[7]The reasons why I was unable to calculate the actual instances containing valence-changing morphology are provided in Chapter 5.

Language		**Sentences per training fold**		**Instances of valchg morphology**	
		mean	std.	mean	std.
Dev	Abui [abz]	1441.2	0.6	26.1	1.6
	Hiaki [yaq]	2519.5	1.5	260.9	5.2
	Wakhi [wbl]	701.7	2.1	5.5	1.0
Held-out	Hixkaryana [hix]	5174.1	0.3	43.2	2.1
	Old Javanese [jav]	275.4	0.5	28.8	1.5
	Yaoyos Quechua [qux]	4183.2	0.4	299.7	7.6
	Chintang [ctn]	9701.1	0.3	N/A	N/A
	South Efate [erk]	1687.5	1.5	N/A	N/A

Table 4.1: Dataset statistics for development and test languages

Chapter 5

RESULTS

In this chapter, I present the evaluation of the performance of my inference module. First, I show the statistics and analyses of the overall results (Section 5.1). Then, I present the results along with a more in-depth analysis of each development and held-out language (Section 5.2 and 5.3).[1] The experiments on two of the held-out languages (Old Javanese and Yaoyos Quechua) revealed an error in my algorithm and a limitation in the valence-changing verbal morphology library. The error and limitation caused a compiling failure in ACE when constructing grammars. For the purpose of this thesis, I manually fixed this error in the grammars and reran the experiments for the two languages. Details about the issue and how I fixed it are documented in Section 5.4. In Section 5.5, I also explain the reason why my module was not able to infer information about valence-changing verbal morphology for two of the initially selected held-out languages (Chintang and South Efate), as mentioned in Section 4.1.

5.1 Overall Results

For each language, the inference module was evaluated from the following aspects (see Chapter 4 for more details):

1. **Parsing**: This aspect includes two metrics: (1) *coverage*, represented as the percentage of the amount of parsed sentences over the size of the test fold, and (2) *ambiguity*, represented as the average number of analyses among the sentences with any analyses.

2. **Treebanking**: This aspect includes one metric, the *validated parse coverage*. This

[1] See the footnote in Section 4.3 for the details on the test folds selected for analyses.

metric focuses on the items within the test profile I created for this evaluation aspect; it examines the amount of items receiving appropriate semantics within this profile.

3. **Grammar Specification Output**: This metric shows whether the algorithm infers expected valence-changing operations from each language corpus.

In this section, I report the results for the first two evaluation aspects, parsing and treebanking. The evaluation of the grammar specification output is presented in the results and error analysis sections for each language. Table 5.1 shows the average test fold sizes and the results for the first two aspects for the development languages; the statistics for the held-out languages are shown in Table 5.2. The results in both tables are the averages over ten test folds. For the coverage and ambiguity, the two tables also show the relative performance differences between the current and baseline system.[2] Detailed analyses of the validated parse coverage for each language are provided in the following subsections.

Language	**Sentences per test fold**		**Coverage**			**Ambiguity**	
	Mean	Std.	Absolute	Change	Validated	Absolute	Change
Abui [abz]	156.8	0.6	33.23%	-0.12%	0/4	301.59	0.02%
Hiaki [yaq]	223.5	1.5	8.81%	0.04%	0/17	44.28	0.19%
Wakhi [wbl]	68.3	2.1	13.18%	0.15%	1/3	18.12	0.01%

Table 5.1: Results for development languages

For the development languages, the coverage remains roughly the same, with slight improvements in Hiaki and Wakhi, and a slight decrease in Abui. For Abui, the grammar produced with the current inference system was unable provide analyses for two sentences

[2]The baseline system refers to the system version before including my module.

Language	Sentences per test fold		Coverage			Ambiguity	
	Mean	Std.	Absolute	Change	Validated	Absolute	Change
Hixkaryana [hix]	574.9	0.3	18.46%	-0.27%	0/3	2873.66	0.04%
Old Javanese [jav]	30.6	0.5	1.31%	0%	0/3	1.25	0%
Yaoyos Quechua [qux]	464.8	0.4	10.52%	0.17%	0/36	4290.79	0.47%

Table 5.2: Results for held-out languages

that had parses in the baseline system: the two sentences contain valence-changing morphology. The drop in the number of sentences with analyses was caused by the lack of lexical coverage. More specifically, the verbal stems in the two sentences did not occur with valence-changing morphemes in the training data so the algorithm was unable to either revise the transitivity information or add new valence-changing operations that are applicable to verbs with specific transitivity.[3] However, the drop in coverage does not include sentences with good analyses.[4] The detailed analysis of Abui is provided in Section 5.2.1.

Similarly, for the held-out languages, the coverage remains roughly the same, with a slight improvement in Yaoyos Quechua, a slight decrease in Hixkaryana, and no change in Old Javanese. For Hixkaryana, the cause of the loss in coverage was different from the cause for Abui. The loss was caused by the transitivity revision process in my algorithm. As described in Section 3.3, my module adds the transitivity information that needs revision to the existing grammar specification instead of removing the information from it. This significantly increases the ambiguity of sentences containing the revised verbal stems such

[3]Note that the algorithm only deals with items with valence-changing morphology. Therefore, it is unable to revise the transitivity of a verbal stem that does not occur with a valence-changing morpheme.

[4]The two sentences include phenomena that the baseline AGGREGATION inference system (i.e., the system without my module) is unable to infer. Therefore, the grammar is not able to provide appropriate analyses for the two sentences in the baseline system.

that the parsing software was unable to analyze these sentences due to the lack of resources. However, similar to the case for Abui, the drop in coverage does not include sentences with good analyses.

In general, the ambiguity of the grammars for each language increases after including my inference module. An increase in ambiguity is expected as my module adds new transitivity information to the existing inference output. The algorithm adds this information by creating new verb types containing new lexical entries for the verbal stems of which the transitivity information needs revision. Increasing the amount of lexical entries leads to an increase in ambiguity. For future work, this could be improved by not adding new verb types and lexical entries to the existing grammar specifications. Instead, the algorithm could check the verbal stems which need revision in the entire corpus, and their co-occurrence with the valence-changing morphology. For example, if a verbal stem is sometimes used with an object-adding morpheme and is inferred as transitive in those uses but otherwise is intransitive, the algorithm should only keep the intransitive lexical entry.

5.2 Evaluation Results for Development Languages

In this section, for each development language, I present the results and an analysis, with specific focus on the grammar specifications and the treebanking results. First, I show how valence-changing morphology is realized (i.e., the pieces of information that should be included in the valence-changing operations) in each language by referring to the descriptive grammar resources and the IGT items in the corpus. Then, I show and analyze the generated grammar specifications relevant to my module and the treebanking results.[5]

[5]The grammar specifications result I show are generated with the training folds corresponding to the test fold I treebank. That is, if the fifth test fold is selected to construct a valence-changing test profile, I also show the grammar specification generated with the fifth training fold.

5.2.1 Abui [abz]

Abui belongs to the Trans-New Guinea family (Hammarström et al., 2022). Three types of valence-changing operations can be found in the corpus: subject-adding, subject-removing, and object-adding operations (Kratochvíl, 2019).

In Abui, there is one causative morpheme *ong-*. According to the descriptive grammar of Abui, this morpheme can be used with verbs which "refer to events that do not include acting and volitional human participants such as 'die'" (Kratochvíl, 2007, p. 375). However, the information about the verbal stems' transitivity, position of the erstwhile subject, and POS of the added argument is not available in the descriptive grammar. Therefore, I made the judgement for the information based on the items available in the corpus. Based on the items containing this causative morpheme, this morpheme can be used with both intransitive (as in (9)) and transitive (as in (10)) verbal stems.[6]

(9)

na	*netoku*	*ongfak*
na	ne-toku	ong-fak
1SG.AGT	1SG.AL-leg	CAUSE-break

'I broke my leg.' (lit. I made my leg break.) [abz] (Kratochvíl, 2019)

(10)

na	*Fanmalei*	*haieng*	*ongakuti*
na	Fanmalei	ha-ieng	ong-akut-i
1SG.AGT	name	3.INAL-eye	CAUSE-close.eyes.PFV-PFV

'I made Fanmalei to close his eyes.' [abz] (Kratochvíl, 2019)

[6](10) is the only instance with transitive verbal stem occurring with *ong-* in the corpus. The judgement that it can be used with the transitive verb is based on my interpretation. However, given my limited understanding of Abui, there might be other possible interpretations. For example, *Fanmalei ha-ieng* could be a phrase containing possessive and the literal meaning of the instance would be "I made Fanmalei's eyes close". If the sentence is interpreted this way, the verb *akut* would be considered an intransitive verb.

For resultative (i.e., subject-removing), Abui has a morpheme *-ta*, as shown in (11). The descriptive grammar does not mention the transitivity of the verbs which can be used with this morpheme. However, according to my observation of the instances containing this morpheme, it can be used with transitive verbal stems; whether it can be used with intransitive verbal stems is unclear to me.

(11) *2 hen ama tukoi hen ama ba talúk kaang*
2 hen ama tukoi hen ama ba ta-lúk kaan
2 3.COP person be.strong 3.COP person REL RES-fight can

'He is a strong man.' [abz] (Kratochvíl, 2019)

For applicative (i.e., object-adding), there are two morphemes: *ming-* and *lang-*. Based on my observation of *ming-*, it occurs with intransitive verbal stems in the corpus, as shown in (12); however, with limited understanding of Abui and the data available in the corpus, it is unclear to me how *lang-* is used.[7]

(12) *na Kalangfat mingyaa*
na Kalangfat ming-yaa
1SG.AGT place APPL-go

'I am going to Kalabahi.' [abz] (Kratochvíl, 2019)

Based on the information collected from the descriptive grammar and the corpus, the algorithm is expected to output the operations shown in Table 5.3.

Results for Abui

The inferred valence-changing operations for the fifth fold in shown in Listing 5.1. The three morphemes listed in Table 5.3 were identified by my algorithm, and their corresponding

[7]In the corpus, *lang-* either occurs in structures that are relatively complicated (e.g., questions, sentences with conjunctions, etc.) or sentences containing only one noun, which made me unable to identify the added arguments and their relevant information.

morpheme	operation	inputs	argpos	argtype	predname
ong-	subj-add	intrans	pre	np	causative
	subj-add	trans	post	np	causative
ta-	subj-rem	trans	N/A	N/A	N/A
ming-	obj-add	intrans	pre	np	applicative

Table 5.3: Expected valence-changing inference for Abui

operations were added to the lexical rule types they belong to. For the causative morpheme *ong-*, the algorithm successfully identified the operation for intransitive inputs along with other required information. However, the subject-adding operation with transitive inputs was not inferred. The grammar specification also includes the correct inference for the subject-removing morpheme, *ta-*. For object-adding operation, while the algorithm correctly inferred the operation with intransitive inputs along with other required information, it inferred an unexpected object-adding operation that applies to transitive verbal stems.[8] The errors are discussed in the following error analysis section.

```
verb-pc37_name=verb-pc37
  verb-pc37_order=prefix
  verb-pc37_inputs=verb138, verb211
    verb-pc37_lrt1_name=verb-pc37_lrt1
      verb-pc37_lrt1_valchg1_operation=subj-add
      verb-pc37_lrt1_valchg1_inputs=intrans
      verb-pc37_lrt1_valchg1_predname=causative
      verb-pc37_lrt1_valchg1_argpos=pre
      verb-pc37_lrt1_valchg1_argtype=np
      verb-pc37_lrt1_lri1_inflecting=yes
```

[8]The operations for *lang-* are not expected to be added to the inference output based on the available items in the Abui corpus, as described in Section 5.2.1. The reason why *lang-* had valence-changing operations was because it belonged to the same lexical rule type containing *ming-*. That is, the two morphemes were put into the same lexical rule type by MOM. Therefore, the operations were added because of *ming-*.

```
      verb-pc37_lrt1_lri1_orth=ong-

verb-pc21_name=verb-pc21
  verb-pc21_order=prefix
  verb-pc21_inputs=verb25, verb58
    verb-pc21_lrt1_name=verb-pc21_lrt1
      verb-pc21_lrt1_valchg1_operation=subj-rem
      verb-pc21_lrt1_valchg1_inputs=trans
      verb-pc21_lrt1_lri1_inflecting=yes
      verb-pc21_lrt1_lri1_orth=ta-

verb-pc26_name=verb-pc26
  verb-pc26_order=prefix
  verb-pc26_inputs=verb-pc10, verb138
    verb-pc26_lrt1_name=verb-pc26_lrt1
      verb-pc26_lrt1_valchg1_operation=obj-add
      verb-pc26_lrt1_valchg1_inputs=trans
      verb-pc26_lrt1_valchg1_predname=applicative
      verb-pc26_lrt1_valchg1_argpos=pre
      verb-pc26_lrt1_valchg1_argtype=np
      verb-pc26_lrt1_lri1_inflecting=yes
      verb-pc26_lrt1_lri1_orth=lang-
      verb-pc26_lrt1_lri2_inflecting=yes
      verb-pc26_lrt1_lri2_orth=ming-
    verb-pc26_lrt2_name=verb-pc26_lrt2
      verb-pc26_lrt2_valchg1_operation=obj-add
      verb-pc26_lrt2_valchg1_inputs=intrans
      verb-pc26_lrt2_valchg1_predname=applicative
      verb-pc26_lrt2_valchg1_argpos=pre
      verb-pc26_lrt2_valchg1_argtype=np
      verb-pc26_lrt2_lri1_inflecting=yes
      verb-pc26_lrt2_lri1_orth=lang-
      verb-pc26_lrt2_lri2_inflecting=yes
```

```
44      verb-pc26_lrt2_lri2_orth=ming-
```

Listing 5.1: An excerpt of the inference output for Abui from the fifth fold

The test profile created for evaluation (i.e., the treebanking aspect) consisted of four sentences. Among the four sentences, one of them had analyses; however, it did not have any appropriate semantics. In addition, I also investigated into the reasons preventing the other three sentences from having parses. The error analysis of these sentences is provided in the following.

Error Analysis of Abui

The algorithm added most expected valence-changing operations with expected input transitivity and required information to the correct corresponding lexical rule types. The reason why the subject-adding operation for transitive input was not inferred was because the only item with causative morpheme being used with a transitive verb was in the test data and not in the training data. Therefore, the algorithm did not have enough data to infer this operation. For object-adding operations, the algorithm incorrectly inferred an operation that takes transitive inputs. This was caused by the incorrect transitivity of the verbal stem used with the object-adding morpheme *ming-* in the training data. When identifying the transitivity of a verbal stem in an object-adding operation, the algorithm highly relies on the number of objects (i.e., direct and indirect objects) of the aligned translation verb. For cases like (13), in which the aligned translation verb (i.e., *pooed*) is transitive and the source language verbal stem (i.e., *eit* 'defecate') is intransitive, the algorithm incorrectly identifies the verbal stem as transitive since there are two objects (direct and prepositional objects) in the translation.

(13) *moku fila nuku di kirengkilai mingeiti*
moku fila nuku di kirengkilai ming-eit-i
kid be.young INDEF 3.AGT flatworm APPL-defecate.PFV-PFV
'The child pooed a flatworm in his stool.' [abz] (Kratochvíl, 2019)

When analyzing the sentences within the test profile, I discovered two issues relevant to the algorithm:

1. The lexical entry with correct transitivity was not in the inputs of the position class containing the valence-changing operation.[9]

2. The lack of lexical coverage and items with valence-changing morphology

First, although the algorithm adds the newly created verb types to the inputs of the position class (see Section 3.3 for more details) to ensure the newly added verb types are applicable to the valence-changing operations, the algorithm does not check whether the verbal entries with the transitivity applicable to the operation are in the inputs of the position class. That is, the algorithm only checks whether the lexical entries with the expected transitivity information is in the existing inference output but does not check whether they are in the inputs of the position class. As such, it is possible to have the lexical entries with correct transitivity in the existing inference output but not have them in the inputs of the position class of interest. Second, the sentences that did not have parses contained either verbal stems or operations that were unseen in the items with valence-changing morphology in the training fold. Therefore, the transitivity information about the verbal stems in the test sentences could not be revised and the unseen operations could not be added by the module.

In sum, for future work, adding a more sophisticated mechanism for identifying source language transitivity would enable the valence-changing operations to be more accurately

[9]A valence-changing operation should be created within a lexical rule type, which is a part of a position class.

represented. In addition, checking whether the inputs of the position classes contain the correct lexical entries (i.e., the lexical entries with the matching transitivity information about the verbal stems in sentences containing valence-changing verbal morphemes) would ensure the applicability of the operations to the lexical entries.

5.2.2 Hiaki [yaq]

Hiaki belongs to the Uto-Aztecan language family (Hammarström et al., 2022). In the Hiaki corpus (Harley, 2019), three types of operation were found: subject-adding, object-adding, and valence-reordering operations. A summary of the expected valence-changing operations in Hiaki is shown in Table 5.4.

In Hiaki, there are two causative morphemes, *-tua* and *-tevo*. The two morphemes differ in the presence of the causee (Tubino Blanco, 2011). An explicit causee argument is required to be accompanied with *-tua*, as shown in (14). On the other hand, an explicit causee is disallowed to co-occur with *-tevo*, as shown in (15). Both morphemes can be used with intransitive and transitive verbs. However, the current valence-changing verbal morphology library is only able to model the structure of *-tua* since the causees (i.e., the erstwhile subjects) are absent from the sentences with *-tevo*. Therefore, the algorithm is not expected to correctly infer operations for *-tevo*. For *-tua*, as mentioned previously, it can be used with transitive and intransitive verbal stems. When *-tua* is used with transitive verbal stems, the erstwhile subjects is the more oblique argument (i.e., `argpos=post`), and the POS of the added arguments is noun phrases (i.e., `argtype=np`); when it is used with intransitive verbal stems, the erstwhile subjects is less oblique (i.e., `argpos=pre`), and the POS of the added arguments is also noun phrases (i.e., `argtype=np`).[10]

[10]As mentioned in Section 3.2.2, the position of erstwhile subjects/added arguments is set to `pre` for intransitive verbs. This choice is in fact arbitrary, since `post` would lead to the same COMPS values.

(14) *Nee Artta ne suatua*
Nee Art-ta ne sua-tua
1S Art-A 1S take.care-CAUSE
'I'm making Art take care of me.' [yaq] (Tubino Blanco, 2011, p. 170)

(15) *Inepo ino suatevo*
Inepo ino sua-tevo
1S 1S(REFL) take.care-CAUSE(I)
'I'm having myself taken care of.' (i.e., I'm having somebody guard me) [yaq] (Tubino Blanco, 2011, p. 170)

For object-adding operations, Hiaki has an applicative morpheme *-ria*. It can be used with both transitive and intransitive verbal stems (Harley et al., 2009). When the verbal stems are transitive, the added arguments occur at the end of the COMPS list (i.e., `argpos=post`) as noun phrases (i.e., `argtype=np`); when the verbal stems are intransitive, the added arguments are at the front of the COMPS list (i.e., `argpos=pre`).[11]

Lastly, for valence-reordering operations, Hiaki has two passive morphemes: *-na* and *-wa*. Based on my observation of the items in the corpus, both morphemes can be used with transitive and intransitive verbal stems. However, the current valence-changing verbal morphology library is only capable of handling passive constructions for transitive verbs. Therefore, only valence-reordering operations with transitive inputs are listed in Table 5.4. In my module, an issue was found in the implementation for valence-reordering operations which prevented the module from producing correct inference for passive, as listed in Table 5.4.[12] More details on this issue are presented in the following error analysis section.

[11]This, again, is an arbitrary choice I set for intransitive verbal stems since the position of arguments of interest does not matter for intransitive verbal stems.

[12]Here, since there is only one element on the COMPS list of the transitive verb with passive morpheme, specifying the position of the affected argument as `argpos=pre` or `argpos=post` yields the same results.

morpheme	**operation**	**inputs**	**argpos**	**argtype**	**predname**
-tua	subj-add	trans	post	np	causative
	subj-add	intrans	pre	np	causative
-ria	obj-add	trans	post	np	applicative
	obj-add	intrans	pre	np	applicative
-na	*subj-dem + obj-prom*	*trans*	*pre*	*np*	*N/A*
-ta	*subj-dem + obj-prom*	*trans*	*pre*	*np*	*N/A*

Table 5.4: Expected valence-changing inference for Hiaki

Results for Hiaki

The inferred valence-changing operations for the fifth fold is shown in Listing 5.2. For valence-increasing operations (i.e., subject- and object-adding operations), the algorithm correctly inferred all required information. For passive constructions, due to an issue in implementation, the results shown here are incorrect. Details of the cause of these errors are discussed in the error analysis section.

```
verb-pc5_name=verb-pc5
  verb-pc5_order=suffix
  verb-pc5_inputs=verb3, verb11, verb-pc43, verb60, verb192
    verb-pc5_lrt1_name=verb-pc5_lrt1
      verb-pc5_lrt1_valchg1_operation=subj-add
      verb-pc5_lrt1_valchg1_inputs=trans
      verb-pc5_lrt1_valchg1_predname=causative
      verb-pc5_lrt1_valchg1_argpos=post
      verb-pc5_lrt1_valchg1_argtype=np
      verb-pc5_lrt1_lri1_inflecting=yes
      verb-pc5_lrt1_lri1_orth=-tua
    verb-pc5_lrt2_name=verb-pc5_lrt2
      verb-pc5_lrt2_valchg1_operation=subj-add
```

```
      verb-pc5_lrt2_valchg1_inputs=intrans
      verb-pc5_lrt2_valchg1_predname=causative
      verb-pc5_lrt2_valchg1_argpos=pre
      verb-pc5_lrt2_valchg1_argtype=np
      verb-pc5_lrt2_lri1_inflecting=yes
      verb-pc5_lrt2_lri1_orth=-tua

verb-pc18_name=verb-pc18
  verb-pc18_order=suffix
  verb-pc18_inputs=verb192, verb-pc36, verb-pc5, verb-pc42, verb-pc43,
   verb124
    verb-pc18_lrt1_name=verb-pc18_lrt1
      verb-pc18_lrt1_valchg1_operation=obj-add
      verb-pc18_lrt1_valchg1_inputs=trans
      verb-pc18_lrt1_valchg1_predname=applicative
      verb-pc18_lrt1_valchg1_argpos=post
      verb-pc18_lrt1_valchg1_argtype=np
      verb-pc18_lrt1_lri1_inflecting=yes
      verb-pc18_lrt1_lri1_orth=-ria
    verb-pc18_lrt2_name=verb-pc18_lrt2
      verb-pc18_lrt2_valchg1_operation=obj-add
      verb-pc18_lrt2_valchg1_inputs=intrans
      verb-pc18_lrt2_valchg1_predname=applicative
      verb-pc18_lrt2_valchg1_argpos=pre
      verb-pc18_lrt2_valchg1_argtype=np
      verb-pc18_lrt2_lri1_inflecting=yes
      verb-pc18_lrt2_lri1_orth=-ria

verb-pc40_name=verb-pc40
  verb-pc40_order=suffix
  verb-pc40_inputs=verb-pc21
    verb-pc40_lrt1_name=verb-pc40_lrt1
      verb-pc40_lrt1_valchg1_operation=subj-dem
```

```
      verb-pc40_lrt1_valchg1_inputs=trans
      verb-pc40_lrt1_lri1_inflecting=yes
      verb-pc40_lrt1_lri1_orth=-wa
    verb-pc40_lrt2_name=verb-pc40_lrt2
      verb-pc40_lrt2_valchg1_operation=subj-dem
      verb-pc40_lrt2_valchg1_inputs=intrans
      verb-pc40_lrt2_lri1_inflecting=yes
      verb-pc40_lrt2_lri1_orth=-wa

verb-pc9_name=verb-pc9
  verb-pc9_order=suffix
  verb-pc9_inputs=verb192, verb68, verb37, verb-pc5, verb43
    verb-pc9_lrt1_name=verb-pc9_lrt1
      verb-pc9_lrt1_valchg1_operation=subj-dem
      verb-pc9_lrt1_valchg1_inputs=trans
      verb-pc9_lrt1_lri1_inflecting=yes
      verb-pc9_lrt1_lri1_orth=-na
    verb-pc9_lrt2_name=verb-pc9_lrt2
      verb-pc9_lrt2_valchg1_operation=subj-dem
      verb-pc9_lrt2_valchg1_inputs=intrans
      verb-pc9_lrt2_lri1_inflecting=yes
      verb-pc9_lrt2_lri1_orth=-na
```

Listing 5.2: An excerpt of the inference output for Hiaki from the fifth fold

The test profile created specifically for evaluation (i.e., the treebanking aspect) consisted of 17 sentences. The generated grammar was able to provide syntactic and semantic analyses for one sentence. However, the grammar was not able to provide any appropriate semantic representations for that sentence. I also reviewed nine sentences out of the remaining 16 sentences which could not be analyzed by the grammar. The analysis of these sentences is discussed in the following.

Error Analysis of Hiaki

For passive constructions, I discovered an issue in the implementation, causing the inference outputs for passive constructions to be incorrect across languages. The design of the current valence-changing morphology library models passive constructions with two types of operation: subject-demoting and object-promoting operations (Curtis, 2018). That is, when modeling one passive construction in the library, users need to create these two types of operations. In addition, users also need to specify the position of the argument on the COMPS list that is affected in the subject-demoting and object-promoting operations. This design for passive constructions implies that (1) although it is possible for intransitive verbs to be passivized across languages, only transitive verbs can be handled in the library since this design requires non-empty COMPS list, and (2) the amount of arguments should remain the same after the reordering (i.e., the subject cannot be removed after a verb is passivized). No passive constructions for intransitive verbs are allowed in the current library. Therefore, the algorithm handles passive constructions incorrectly in several ways: (1) the algorithm allows intransitive inputs for passive constructions, (2) the algorithm does not use object-promoting operations in conjunction with subject-demoting operations (for transitive inputs), and (3) the algorithm does not specify the position of the affected argument on the COMPS list in subject-demoting operations (for transitive inputs).[13]

The one sentence that had parses did not have appropriate semantics due to the incorrect implementation of passive constructions. Among the nine additional sentences I reviewed, six sentences had issues related to incorrect implementation of passive construction. For future work, the algorithm could correctly model passive constructions by using subject-demoting operations in conjunction with object-promoting operations for transitive verbs.

[13]Adding (3) alone to the algorithm is still incorrect. To correctly model passive construction in the valence-changing morphology library, users need to add two operations (i.e., subject-demoting and object-promoting operations), and also specify the position of the affected argument in both operations.

5.2.3 Wakhi [wbl]

Wakhi belongs to the Indo-European family (Hammarström et al., 2022). The Wakhi corpus contains one verbal valence-changing morpheme, *-ɨv*, a causative morpheme (Kaufman et al., 2020). Since I was unable to locate any grammatical description concerning about the causative construction in Wakhi, I constructed the expected valence-changing operations based on my observation of the items in the corpus. In the corpus, *-ɨv* is only shown to be used with intransitive verbal stems, as in (16). For intransitive verbal stems, erstwhile subjects are at the front of the COMPS list (i.e., `argpos=pre`)[14] and the added arguments are noun phrases (i.e., `argtype=np`).

(16) *ja* *ʂelʐinʂ* *dʒojɨvd*
j-a ʂelʐin=ʂ dʒoj-ɨv-t
DEM-MED woman=PROG read-CAUS-3SG.NPST

'the woman teaches.' (lit. The woman makes someone read.) [wbl] (Kaufman et al., 2020)

Results for Wakhi

The inferred valence-changing operations for the fifth fold is shown in Listing 5.3. The valence-changing operation for the causative morpheme *-ɨv* was correctly added to its lexical rule type. In addition, all pieces of information required for the operation were correctly inferred.

```
verb-pc9_name=verb-pc9
  verb-pc9_order=suffix
  verb-pc9_inputs=verb2, verb6, verb-pc16
    verb-pc9_lrt2_name=verb-pc9_lrt2
      verb-pc9_lrt2_valchg1_operation=subj-add
```

[14]For intransitive verbs, this is an arbitrary choice since the positions of the arguments of interest do not matter for intransitive verbs. Details are provided in Section 3.2.2.

```
6       verb-pc9_lrt2_valchg1_inputs=intrans
7       verb-pc9_lrt2_valchg1_predname=causative
8       verb-pc9_lrt2_valchg1_argpos=pre
9       verb-pc9_lrt2_valchg1_argtype=np
10      verb-pc9_lrt2_lri1_inflecting=yes
11      verb-pc9_lrt2_lri1_orth=-iv
```

Listing 5.3: An excerpt of the inference output for Wakhi from the fifth fold

The test profile created specifically for evaluation (i.e., the treebanking aspect) included three sentences. Among the three sentences, one sentence had parses. After performing treebanking, I found the one with parses had appropriate semantics. I also investigated into the remaining two sentences that did not receive analyses. The analysis is provided in the following error analysis section.

Error Analysis of Wahki

One sentence I investigated failed to parse because of the lack of lexical coverage. The parsing failure of this sentence was caused by the lack of lexical coverage irrelevant to the inference for valence-changing morphology. In this sentence, the lexical rule containing the valence-changing morpheme, *-iv*, was successfully applied. Another sentence failed to parse because the lexical entry was not in the input of the position class. This is the same issue analyzed in Section 5.2.1. The current algorithm adds the newly created verb types to the inputs of a position class (see Section 3.3 for more details) to ensure the newly added verb types are applicable to its valence-changing operation; however, the algorithm does not check whether the verbal entries with the transitivity applicable to the operation are in the inputs of that position class. It is possible that MOM does not add the verb type containing the lexical entry of interest to the inputs of the position class which contains the valence-changing morpheme being used with the verb entry.[15] For future work, similar to what was proposed

[15]Note that the algorithm adds information to the existing lexical rule types, which need to be defined within position classes. The two types of information are output by MOM.

in Section 5.2.1, the algorithm could check whether the lexical entries are in the inputs of the position classes of interest to ensure the applicability of the lexical rule types containing valence-changing operations.

In this section, I presented the results and analyses of the three development languages. Some issues could have been addressed if a complete error analysis, including the analyses of the inference output, generated grammar, and parsed results, were done before running experiments on the held-out languages. However, due to the time constraint, I was only able to perform partial error analysis (i.e., analysis of the inference output) before freezing the algorithm and running experiments on the held-out languages. In the next section, I show the results and analyses of the held-out languages.

5.3 Evaluation Results for Held-out Languages

Similar to Section 5.2, this section shows the results and evaluation of the held-out languages. Different from development languages, I only worked with the held-out languages after I froze my system. For each held-out language, I first describe how valence-changing verbal morphology is realized based on the descriptive grammar resources and the IGT items in the corpus. Then, I evaluate the inference output of the algorithm with the expected grammar specifications, and analyze the randomly selected sentences in the test profile I created.

5.3.1 Hixkaryana [hix]

Hixkaryana belongs to the Cariban language family (Hammarström et al., 2022). The Hixkaryana corpus contains one verbal valence-changing morpheme, *-ho*, a causative morpheme (Meira, 2020). According to Derbyshire (1979), the causative morpheme can be used with both transitive and intransitive verbal stems, as in (17) and (18). As shown in (17), for transitive verbal stems, the erstwhile subject, *tɨnyo* 'husband', is at the end of the COMPS list (i.e., `argpos=post`). On the other hand, the erstwhile subject (i.e., *bɨyekomo* 'boy') of

the intransitive verbal stem is at the front of the COMPS list (i.e., `argpos=pre`).[16] Since the POS of the added argument is not available in the descriptive grammar, I constructed the expected grammar specification based on my observation of the IGT items. The POS of the added arguments (i.e., *wosɨ* 'woman') in causative constructions is noun phrase (i.e., `argtype=np`). A summary of the expected valence-changing operations is shown in Table 5.5.

(17)

	bɨyekomo	*yotahahono*	*wosɨ*	*tɨnyo*	*wya*
	bɨyekomo	y-otaha-ho-no	wosɨ	tɨnyo	wya
	boy	3.S.3.O.OVERT-hit-CAUS-IMM.PST	woman	3.REFL-husband.of	by

'The woman caused her husband to hit the boy.' [hix] (adapted from Derbyshire 1979, p. 89)

(18)

	bɨyekomo	*horymamnohye*	*wosɨ*
	bɨyekomo	horymamɨ-noh-ye	wosɨ
	boy	grow-CAUS-DIST.PAST.COMPL	woman

'The woman raised the boy.' (lit. The woman causes the boy to grow.) [hix] (adapted from Derbyshire 1979, p. 89)

morpheme	**operation**	**inputs**	**argpos**	**argtype**	**predname**
-ho	subj-add	trans	post	np	causative
	subj-add	intrans	pre	np	causative

Table 5.5: Expected valence-changing inference for Hixkaryana

[16] As mentioned previously in Section 3.2.2, this is an arbitrary choice I set for intransitive verbs.

Results for Hixkaryana

The inferred valence-changing operations for the fifth fold is shown in Listing 5.4. The causative morpheme *-ho* (line 23 and 45) along with its corresponding valence-changing operations for transitive (line 5-9) and intransitive (line 27-31) were consistent with the expected grammar specifications shown in Table 5.5. However, the lexical rule types containing *-ho* also included other morphemes that do not function as valence-changing morphology. The detailed analysis of this is discussed in the error analysis section.

```
1  verb-pc4_name=verb-pc4
2    verb-pc4_order=suffix
3    verb-pc4_inputs=verb-pc25, verb-pc28, verb-pc33, verb-pc34, verb100,
       verb-pc41, verb-pc48, verb-pc55, verb121
4      verb-pc4_lrt1_name=verb-pc4_lrt1
5        verb-pc4_lrt1_valchg1_operation=subj-add
6        verb-pc4_lrt1_valchg1_inputs=trans
7        verb-pc4_lrt1_valchg1_predname=causative
8        verb-pc4_lrt1_valchg1_argpos=post
9        verb-pc4_lrt1_valchg1_argtype=np
10       verb-pc4_lrt1_lri1_inflecting=yes
11       verb-pc4_lrt1_lri1_orth=-txahke
12       verb-pc4_lrt1_lri2_inflecting=yes
13       verb-pc4_lrt1_lri2_orth=-0
14       verb-pc4_lrt1_lri3_inflecting=yes
15       verb-pc4_lrt1_lri3_orth=-nihi
16       verb-pc4_lrt1_lri4_inflecting=yes
17       verb-pc4_lrt1_lri4_orth=-tano
18       verb-pc4_lrt1_lri5_inflecting=yes
19       verb-pc4_lrt1_lri5_orth=-nohka
20       verb-pc4_lrt1_lri6_inflecting=yes
21       verb-pc4_lrt1_lri6_orth=-yehka
22       verb-pc4_lrt1_lri7_inflecting=yes
23       verb-pc4_lrt1_lri7_orth=-ho
```

```
      verb-pc4_lrt1_lri8_inflecting=yes
      verb-pc4_lrt1_lri8_orth=-tihka
    verb-pc4_lrt2_name=verb-pc4_lrt2
      verb-pc4_lrt2_valchg1_operation=subj-add
      verb-pc4_lrt2_valchg1_inputs=intrans
      verb-pc4_lrt2_valchg1_predname=causative
      verb-pc4_lrt2_valchg1_argpos=pre
      verb-pc4_lrt2_valchg1_argtype=np
      verb-pc4_lrt2_lri1_inflecting=yes
      verb-pc4_lrt2_lri1_orth=-txahke
      verb-pc4_lrt2_lri2_inflecting=yes
      verb-pc4_lrt2_lri2_orth=-0
      verb-pc4_lrt2_lri3_inflecting=yes
      verb-pc4_lrt2_lri3_orth=-nihi
      verb-pc4_lrt2_lri4_inflecting=yes
      verb-pc4_lrt2_lri4_orth=-tano
      verb-pc4_lrt2_lri5_inflecting=yes
      verb-pc4_lrt2_lri5_orth=-nohka
      verb-pc4_lrt2_lri6_inflecting=yes
      verb-pc4_lrt2_lri6_orth=-yehka
      verb-pc4_lrt2_lri7_inflecting=yes
      verb-pc4_lrt2_lri7_orth=-ho
      verb-pc4_lrt2_lri8_inflecting=yes
      verb-pc4_lrt2_lri8_orth=-tihka
```

Listing 5.4: An excerpt of the inference output for Hixkaryana from the fifth fold

The test profile created specifically for evaluation (i.e., the treebanking aspect) included three sentences. Among the three sentences, none can be parsed by the generated grammar. Therefore, I did not perform treebanking for Hixkaryana. Instead, I looked into the three sentences, with specific focus on the verbs occurring with the causative morpheme *-ho*. Upon review, I found that three sentences failed to parse due to reasons irrelevant to my algorithm. More specifically, I found that the verbs of interest in one of the sentences had analyses with

appropriate valence information and semantics. The grammar was unable to parse the verbs (i.e., the verbs used with valence-changing morphemes) in other two sentences due to a reason irrelevant to my changes: the relationship between some position classes is not captured by the inference system so the lexical rule types in those position classes do not coordinate well with each other.

Error Analysis of Hixkaryana

Upon examining the grammar specification output by my algorithm and relevant IGT items in the corpus, I discovered several issues in the current algorithm:

1. The algorithm has limitations in inferring specifications given items with pro-drop.

2. Valence-changing operations are added to lexical rule types containing other morphemes that are not valence-changing morphology.

The first issue is anticipated. Although the inference output appeared to be correct, the algorithm is in fact, unable to determine the position of the arguments of interest on the COMPS list for a lot of items.[17] In the Hixkaryana corpus, many items have pro-drop. Currently, as described in Section 3.2.2, the algorithm's way of handling items with pro-drop is to assign default position values to `argpos` in the operation collected from those items (i.e., assigning `argpos=pre` for operations with intransitive inputs and `argpos=post` for operations with transitive inputs). For operations that take transitive inputs, this default setting may bias the algorithm toward selecting operations with `post` as the value of `argpos` since the algorithm uses majority voting when deciding the final operations to output (see Section 3.2.3 for more details). That is, the more items with pro-drop are available in the corpus, the more likely that the final valence-increasing operation will have `post` as the value

[17]This only matters when the verbal stems are transitive since for intransitive verbal stems, the choice for the position of the erstwhile subjects/added arguments is arbitrary since it amounts to the same results (i.e., for intransitive verbs, `argpos=pre` and `argpos=post` yields the same results).

of `argpos`. Recall that the current algorithm only identifies whether an item contains pro-drop but does not check whether the dropped arguments are the ones on the COMPS lists of the verbal stems. For future work, this bias could be reduced by checking whether the arguments on the COMPS list are dropped in items with pro-drop. If the dropped arguments are not the ones on the COMPS list, the algorithm could proceed to find out the positions of the arguments of interest; if any of the dropped arguments are on the COMPS list, the algorithm could discard the instance unless there is not a sufficient amount of instances to infer information about valence-changing verbal morphology for the input language. In the latter case, the algorithm could assign a default value for the argument position of interest.

The second issue resulted from the way the algorithm adds the operations. As shown in Listing 5.4, in addition to the actual causative morpheme *-ho*, there were also other morphemes in the same lexical rule types. This misrepresented those morphemes as valence-changing morphology and caused the grammar to apply lexical rules containing valence-changing operations to situations they should not apply. The current algorithm produces such results because instead of creating new lexical rule types for valence-changing operations, it adds valence-changing operations to existing lexical rule types. Future work could create new lexical rule types which contain only the relevant valence-changing morphology within the existing position classes. By doing so, the grammar could more accurately model the valence-changing constructions of interest and prevent the valence-changing operations from being applied to other irrelevant morphemes.

5.3.2 Old Javanese [jav]

Old Javanese belongs to the Austronesian language family (Hammarström et al., 2022). The Old Javanese corpus contains three types of valence-changing morphology: subject-adding, object-adding, and valence-reordering (i.e., passive) (Arci, 2023). The expected valence-changing operations are listed in Table 5.6.

The descriptive grammar resource I found is based on a modern descendant of Javanese spoken in Malang. According to this resource (Hemmings, 2013), in Javanese, *-i* and *-aké*

can function as both causative and applicative morphemes. When functioning as causative morphemes, both morphemes can be used with transitive and intransitive verbal stems. (19) is an example of *-i* being used with a transitive verb *mangan* 'eat', and (20) is an example of *-aké* being used with the same transitive verb. When being used as causative morphemes with transitive verbs, *-i* and *-aké* function respectively as direct and indirect causative markers. That is, when transitive verbs are used with *-i*, the causee (i.e., erstwhile subject) is less oblique than the object; when transitive verbs are used with *-aké*, the causee is more oblique than the object (Hemmings, 2013). This is shown in the differences in terms of the erstwhile subjects' positions on the COMPS list. The erstwhile subject in (19) appears at the front of the COMPS list (i.e., `argpos=pre`) while the erstwhile subject in (20) appears at the end of the COMPS list (i.e., `argpos=post`). The POS of the added arguments (i.e., *aku* 'I') in both examples is noun phrase (i.e., `argtype=np`).

(19) *aku* *mangani* *kucing* *iwak*
aku mangan-i kucing iwak
1SG(A) AV.eat-CAUS cat(O) fish

'I fed the cat fish.' [jav] (Hemmings, 2013, p. 169)

(20) *aku* *manganaké* *iwak* *menyang* *kucing*
aku mangan-aké iwak menyang kucing
1SG AV.eat-CAUS fish towards cat

'I fed the fish to the cat.' [jav] (Hemmings, 2013, p. 170)

For object-adding operations, *-i* can be used with intransitive verbal stems, as shown in (21); *-aké* can be used with transitive verbal stems, as shown in (22). Whether *-i* can be used with transitive verbal stems and whether *-aké* can be used with intransitive verbal stems are unclear to me based on the descriptive grammar I was able to locate and the items available in the corpus.

(21) *pelem* *nyeblòki* *gentèng* *ómahku*
pelem nyeblòk-i gentèng ómah-ku
mango(A) AV.fall-APPL roof(O) house-1SG.POSS
'A mango fell on the roof of my house.' [jav] (Hemmings, 2013, p. 168)

(22) *aku* *masakaké* *Karolina* *jajan*
aku masak-aké Karolina jajan
1SG(A) AV.cook-APPL Karolina(O) cake
'I baked Karolina a cake.' [jav] (Hemmings, 2013, p. 168)

For valence-reordering operations, Old Javanese has four passive morphemes, *-in-*, *ka-*, *pa-*, and *-ĕn*. I was unable to find the descriptive grammar about the transitivity of the verbal stems that can be used with these morphemes. Based on my observation of the corpus, *-in-*, *ka-*, and *pa-* can be used with both intransitive and transitive verbs. In the corpus, *-ĕn* only occurs once with a transitive verb; whether it can be used with intransitive verbs is unclear based on the resource I was able to locate. However, since the library can only model passive constructions for transitive verbs, only the operations with transitive inputs are listed in Table 5.6.[18]

Results for Old Javanese

The grammar specifications output by the algorithm were the specifications for passive constructions. No inference for subject and object-adding operations were output by the algorithm. Since, as described previously in Section 5.2.2, passive constructions are incorrectly modeled by the algorithm, the specifications are not presented here. The detailed analysis

[18]The operations for passive constructions listed in Table 5.6 are the expected grammar specifications. Grammar specifications for passive are unable to be correctly inferred by the algorithm due to the issue discussed in Section 5.2.2.

morpheme	**operation**	**inputs**	**argpos**	**argtype**	**predname**
-i	subj-add	trans	pre	np	causative
	subj-add	intrans	pre	np	causative
	obj-add	intrans	pre	np	applicative
-aké	subj-add	trans	post	np	causative
	subj-add	intrans	pre	np	causative
	obj-add	trans	pre	np	applicative
-in-, ka-, pa-, and -ĕn	*subj-dem+obj-prom*	*trans*	*pre*	*N/A*	*N/A*

Table 5.6: Expected valence-changing inference for Javanese

of the absence of valence-increasing operations in the inference output is provided in the following error analysis section.

Error Analysis of Old Javanese

The main cause that prevented the algorithm from producing inference output for valence-increasing operation was the absence of a required tier in the enriched Xigt corpus. For items containing valence-increasing morphemes, the algorithm refers to the source language POS tier, the **m-pos** tier, for information about source language objects and pronouns. The algorithm skips the item if it is an item containing a valence-increasing morpheme but does not contain this tier. Since the all items with valence-increasing morphology in the Old Javanese corpus are missing the **m-pos** tier, the algorithm was not able to infer valence-increasing operations with those items. Future work could add a default strategy to handle items that lack required tiers.

5.3.3 Yaoyos Quechua [qux]

Yaoyos Quechua belongs to the Quechuan language family (Shimelman, 2017). The Yaoyos Quechua corpus contains two types of valence-changing morphology: subject-adding (i.e., causative) and valence-reordering (i.e., passive) (Shimelman, 2012). The expected valence-changing operations are shown in Table 5.7.[19]

Based on my observation of the items in the corpus, Yaoyos Quechua has a causative morpheme *-chi*; it can be used with both transitive and intransitive verbs. For intransitive verbal stems, the erstwhile subject is at the front of the COMPS list (i.e., `argpos=pre`) and the POS of the added argument is noun phrase (i.e., `argtype=np`). For transitive verbal stems, the only sentence I was able to locate that does not involve pro-drop for the arguments on the COMPS list is shown in (23). The position of the erstwhile subject *pashña* 'girl' is ambiguous in this case. That is, the two arguments on the COMPS list, *pashña* 'girl' and *wasi* 'house' are equally close to the verbal stem *tr'aya* 'arrive'. Therefore, based on the resource I was able to gather, the position of the erstwhile subject could be either at the front or the end of the COMPS list.

(23)

Kalamina	*wasiman*	*tr'ayarachin*	*pashñata*
Kalamina	wasi-man	tr'aya-ra-chi-n	pashña-ta
tin.roof	house-ALL	arrive-URGT-CAUS-3	girl-ACC

'He had the girl come to a house with a tin roof.' [qux] (Shimelman, 2012)

For passive constructions, Yaoyos Quechua has a passive morpheme *-raya*. Based on my observation of the items in the corpus, it can be used with transitive verbs.

[19]For Yaoyos Quechua, again, the algorithm is unable to produce correct grammar specifications for passive constructions due to the issue discussed in Section 5.2.2.

morpheme	**operation**	**inputs**	**argpos**	**argtype**	**predname**
-chi	subj-add	trans	post/pre	np	causative
	subj-add	intrans	pre	np	causative
-raya	*subj-dem+obj-prom*	*trans*	*pre*	*N/A*	*N/A*

Table 5.7: Expected valence-changing inference for Yaoyos Quechua

Restuls for Yaoyos Quechua

The inferred valence-changing operations for the fifth fold is shown in Listing 5.5. Here, since passive constructions were implemented incorrectly in the algorithm, the inference outputs for passive are not shown. As shown in the excerpt, the subject-adding operations for *-chi* met the expected specifications. That is, for the operations with intransitive inputs, the erstwhile subjects are at the front of the COMPS list and the added arguments are noun phrases (line 13-17 and line 37-41). On the other hand, for the operations with transitive inputs, the erstwhile subjects are at the end of the COMPS list and the added arguments are noun phrases (line 5-9 and line 27-30).

```
1  verb-pc16_name=verb-pc16
2    verb-pc16_order=suffix
3    verb-pc16_inputs=verb-pc9, verb140, verb407, verb27, verb-pc29, verb-
       pc65, verb-pc71, verb-pc70, verb85
4      verb-pc16_lrt1_name=verb-pc16_lrt1
5        verb-pc16_lrt1_valchg1_operation=subj-add
6        verb-pc16_lrt1_valchg1_inputs=trans
7        verb-pc16_lrt1_valchg1_predname=causative
8        verb-pc16_lrt1_valchg1_argpos=post
9        verb-pc16_lrt1_valchg1_argtype=np
10       verb-pc16_lrt1_lri1_inflecting=yes
11       verb-pc16_lrt1_lri1_orth=-chi
12     verb-pc16_lrt2_name=verb-pc16_lrt2
```

```
      verb-pc16_lrt2_valchg1_operation=subj-add
      verb-pc16_lrt2_valchg1_inputs=intrans
      verb-pc16_lrt2_valchg1_predname=causative
      verb-pc16_lrt2_valchg1_argpos=pre
      verb-pc16_lrt2_valchg1_argtype=np
      verb-pc16_lrt2_lri1_inflecting=yes
      verb-pc16_lrt2_lri1_orth=-chi

verb-pc80_name=verb-pc80
  verb-pc80_order=suffix
  verb-pc80_inputs=verb161, verb-pc7, verb407
    verb-pc80_lrt4_name=verb-pc80_lrt4
      verb-pc80_lrt4_valchg1_operation=subj-add
      verb-pc80_lrt4_valchg1_inputs=trans
      verb-pc80_lrt4_valchg1_predname=causative
      verb-pc80_lrt4_valchg1_argpos=post
      verb-pc80_lrt4_valchg1_argtype=np
      verb-pc80_lrt4_lri1_inflecting=yes
      verb-pc80_lrt4_lri1_orth=-chi
      verb-pc80_lrt4_lri2_inflecting=yes
      verb-pc80_lrt4_lri2_orth=-yku
      verb-pc80_lrt4_lri3_inflecting=yes
      verb-pc80_lrt4_lri3_orth=-manchik
    verb-pc80_lrt7_name=verb-pc80_lrt7
      verb-pc80_lrt7_valchg1_operation=subj-add
      verb-pc80_lrt7_valchg1_inputs=intrans
      verb-pc80_lrt7_valchg1_predname=causative
      verb-pc80_lrt7_valchg1_argpos=pre
      verb-pc80_lrt7_valchg1_argtype=np
      verb-pc80_lrt7_lri1_inflecting=yes
      verb-pc80_lrt7_lri1_orth=-chi
      verb-pc80_lrt7_lri2_inflecting=yes
      verb-pc80_lrt7_lri2_orth=-yku
```

```
46      verb-pc80_lrt7_lri3_inflecting=yes
47      verb-pc80_lrt7_lri3_orth=-manchik
```

Listing 5.5: An excerpt of the inference output for Yaoyos Quechua from the fifth fold

The test profile created specifically for evaluation (i.e., the treebanking aspect) included 36 sentences. None of the 36 sentences had analyses. Therefore, treebanking was not performed for Yaoyos Quechua. Instead, I looked into ten randomly selected sentences from this test profile with specific focus on the verbs containing the valence-changing morphemes. Four of the sentences had correct valence information and semantics for the verbs. Three of the sentences included passive morphemes, which are incorrectly implemented in the algorithm. Among the remaining three sentences, one had the parsing failure relevant to the algorithm. Details are discussed in the following error analysis section.

Error Analysis of Yaoyos Quechua

Upon examining the grammar specifications output by the algorithm and relevant IGT items in the corpus, in addition to the issues mentioned in Section 5.3.1, I also discovered an issue in the current algorithm: when determining the position of the argument of interest on the COMPS list, the algorithm should examine the obliqueness in terms of words instead of morphemes.

This issue is an erroneous design of the algorithm although it generated expected grammar specifications for the languages examined in the experiments. Currently, when determining the position of erstwhile subject/added argument on the COMPS list, the algorithm first identifies the arguments on the COMPS list by looking for the morphemes aligned to specific translation words.[20] Then, the algorithm compares the distance between the erstwhile subject/added argument and the verbal stem, and that between the remaining argument and the verbal stem. In the current algorithm, the distance is measured in terms of morphemes instead of words. For the example in (24), the verbal stem *tr'aya* 'arrive' has two arguments

[20]See section 3.2.2 for details.

on its COMPS list, *wasi* 'house' and *pashña* 'girl'. In terms of morphemes, *wasi* 'house' would be at the front of the COMPS list and *pashña* 'girl' would be the one at the end of the COMPS list. On the other hand, in terms of words, the two arguments should be equally close to the verb. In the context of the valence-changing morphology library, the algorithm should determine the position of the arguments of interest in the latter way. This issue was not discovered during the implementation phase because the development languages either have intransitive inputs for their valence-increasing operations or have both arguments on the same side of transitive verbal stems.[21] The issue became obvious only when the arguments occur on both sides of a verbal stem. For future work, this issue needs to be fixed in order to collect information that reflects the language more accurately.

(24)

Kalamina	*wasiman*	*tr'ayarachin*	*pashñata*
Kalamina	wasi-man	tr'aya-ra-chi-n	pashña-ta
tin.roof	house-ALL	arrive-URGT-CAUS-3	girl-ACC

'He had the girl come to a house with a tin roof.' [qux] (Shimelman, 2012)

In addition to the issues discovered from the grammar specification output, I also discovered an issue when I reviewed the test profile I created for evaluation. One sentence had incorrect transitivity information. The sentence had an intransitive verb but was identified as transitive. Upon checking the verb in the training fold, I found that the verb did occur in sentences containing valence-changing morphology but the algorithm was unable to revise the transitivity information. This was because the translation structure of the item containing that verb was overly complicated to be handled by the algorithm. As described in Section 3.2.2, the algorithm relies heavily on the structures of translation when inferring the transitivity information. The translations considered by the algorithm are limited to certain simple structures and a lot of items in the corpus have structures with more variety and complexity. Therefore, the current algorithm is unable to deal with items that have

[21] When both arguments are on the same side of a verbal stem, the order of the arguments on the COMPS list is the same based on the position of the morphemes or that of the words.

complicated translations. For example, the verb with the incorrect transitivity information in the test sentence is *saya* 'wait'. (25) is an example containing *saya* in the training data and the translation in (25) contains coordination (i.e., *and*). As described in Section 3.1, the algorithm filters out sentences containing complex translation structures such as imperative and coordination. For future work, the algorithm could rely less on the English translations when inferring the transitivity information. One possible way is to search for objects in the source language and only look at the translation when the algorithm needs additional information about the structure of the source language.

(25) *Motonta* *sayaykachishpa* *qarakurqa*
Motonta saya-yka-chi-shpa qara-ku-rqa-Ø
motorcycle-3-ACC wait-EXCEP-CAUS-SUBIS serve-REFL-PST-3

'He stopped his motorcycle and made the offering. I don't know.' [qux] (Shimelman, 2012)

5.4 Report on Grammar Compilation Error

When running experiments on Yaoyos Quechua and Javanese, the AGGREGATION system encountered a grammar compilation error after including my inference module. This grammar compilation error revealed two levels of error: an error in the valence-changing inference module and an error in the valence-changing verbal morphology library.

The error in the inference module was caused by the incorrect implementation of the valence-reordering operations (i.e., passive). As mentioned in Section 5.2.2, the current valence-changing morphology library models passives by using a subject-demoting operation in conjunction with an object-promoting operation. This design implies that only the valence-reordering of transitive verbs can be handled by the library. During the implementation, I only included the subject-demoting operation in the algorithm and did not restrict the transitivity of the verbal stems (see Section 5.2.2 for more details on this). This incorrect

implementation allows the creation of subject-demoting operations with intransitive inputs, which causes the grammar to produce a compilation error since subject-demoting operation requires at least an element on its COMPS list. I fixed this error by manually removing an additional element on the COMPS list of the daughter of the subject-demoting operation. The grammars for Yaoyos Quechua and Old Javanese were able to compile successfully after the manual fix.[22]

In addition, the incorrect implementation of passive construction also revealed an error in the validation part of the valence-changing verbal morphology library in the Grammar Matrix. Before generating grammars with the grammar specifications in the AGGREGATION pipeline, the grammar specifications need to pass a validation test. However, the incorrect grammar specifications were not detected by the validation process in the valence-changing verbal morphology library. This allowed the AGGREGATION to proceed to generate grammars with the incorrect grammar specifications.

5.5 Report on Experiment Failure for Initial Held-out Languages

As mentioned in Section 4.1, my module failed to produce new inferences for two of the initially selected held-out languages, Chintang [ctn] and South Efate [erk]. The causes which prevented my module from producing new inference were different for the two languages. When collecting the valence-changing verbal morphemes, the algorithm locates the morphemes by looking at the grams in the **g** tiers and get their corresponding morphemes in the **m** tiers. Then, it check the morphemes with the verbal affixes inferred by MOM to ensure that the morphemes are the verbal morphemes. The morphemes that do not match the verbal affixes inferred by MOM are not considered valence-changing verbal morphemes.[23]

---

[22]The reason why this error did not happen to Hiaki is because the inputs of the position class containing subject-demoting operations includes another position class. On the other hand, in Yaoyos Quechua and Old Javanese, the inputs contain only verb types. In the case illustrated by Hiaki, the Grammar Matrix customization system creates an intermediate "daughter" type for the lexical rule, which avoids the grammar compilation problem encountered in the other languages.

[23]Due to this reason, I was unable to calculate the actual instances containing valence-changing morphology for the two held-out languages.

For Chingtang, the failure of producing new inference was due to the mismatch of the affix form in the **m** tier in enriched Xigt corpus and that inferred by MOM. In the **m** tier of enriched Xigt for this language, the affixes do not keep the hyphens while the hyphens are kept with the verbal affixes inferred by MOM. This caused the morphemes in the **m** tier of the enriched Xigt for Chintang to be not considered "verbal affixes" since my algorithm requires the verbal affixes in the enriched Xigt to be in the same forms of those inferred by MOM. Therefore, no items in the Chintang corpus (Bickel et al., 2013) were identified as the ones containing valance-changing morphology.

South Efate was selected as a held-out language in error. My selection criterion was that the language should have valence-changing morphology, as indicated by grams characteristic of this morphology in the corpus. However, in this selection process, I did not verify that the grams were actually glossing verbal affixes. South Efate seemed to match the selection criterion because the South Efate corpus (Thieberger, 2006) contains the grams listed in Table 3.1 (e.g., 2S.BEN) although they are used with nominal stems and were filtered out during the verification process when the algorithm was selecting the items of interest.

These issues were not identified before selecting the two languages as held-out languages because for the purpose of held-out languages, I tried to look at the held-out corpora as little as possible. The only criterion I used when selecting the held-out languages is to check whether the gloss lines in the language data contain the grams of interest.

5.6 Summary

This chapter reports the results for the evaluation of both development and held-out languages. Overall, the inference module performed as expected for the development languages, except for passive constructions. Similarly, the algorithm was unable to produce correct grammar specifications for passive constructions for the held-out languages. For the held-out languages, the algorithm produced expected grammar specifications for subject-adding operations for both Hixkaryana and Yaoyos Quechua. However, the experiments for the held-out languages revealed the limitations of the algorithm. The current algorithm has very

strict requirements for the translation structures and the data (e.g., the tiers in the enriched Xigt) needed for inference. In addition to the results and evaluation, I show why the generated grammar failed to compile for two of the held-out languages and how I manually fixed the error. Lastly, I also explained why two of the initially selected held-out languages did not produce any additional inference output after adding my changes. The following chapter provides several directions for future work and concludes this work.

Chapter 6

CONCLUSION

This thesis presented an inference module that extracts information about the valence-changing verbal morphology automatically from an input IGT corpus. Specifically, I have described how the module collects required information for various types of valence-changing morphology. The module has been shown to be capable of adding correct valence-changing operations to the existing inference output for most development and held-out languages. In the following, I provide several directions of improvement that could be considered in the future work and some lessons I learnt from this project.

Upon evaluating the system performance, I recognized several aspects of the algorithm that could be improved in the future work. First, future work could consider adopting approaches that have less requirements on the data. For valence-increasing operations, the current algorithm has relatively strict requirements on not only the input enriched Xigt corpus but also the structures of the translations in the IGT items. Specifically, the algorithm requires the input enriched Xigt corpus to contain certain tiers which might not be available across corpora (e.g., the **m-pos** tier). The algorithm also has less flexibility in inferring information from IGT items of which the translations have complicated structures. For future work, loosening the requirements on data and translation structures could improve the generalizability of the algorithm. Second, future work could also consider improving the integration of the valence-changing operations with existing lexical rule types. The current algorithm adds valence-changing operations to existing lexical rule types that might contain other affixes irrelevant to valence-changing operations. This could cause the grammars to apply valence-changing operations to morphemes that they should not apply to. To prevent this, future work could check the lexical rule types containing the valence-changing affixes

and create separate lexical rule types if necessary.

In addition, I would also like to bring up some lessons I learnt in this study. First, the study again showed that it is important to examine the inference outputs alone, as highlighted in Dods's (2022) work on inference for adnominal possession: the system generated grammars could fail to provide analyses for sentences even when the information in the grammar specifications is correct. This suggests that the cause of the parsing failure might be irrelevant to the grammar specifications of interest. Second, the algorithm shows the strengths and limitations of using translations to deduce certain structural properties of the source language. The translations in the corpus the algorithm make it possible to identify the arguments of interest and their relation in the source languages; however, such approach can only be used when the structures of translation are relatively simple and predictable. In corpora across languages, translations often have a wide variety of structures. Therefore, it is important for future researchers to find a balance between when to rely on the translations and when not to incorporate them during the development of the algorithms.

BIBLIOGRAPHY

Andrea Arci. 2023. Bhuvanakośa, Chapter 3. Extract from the draft of an in-progress critical edition and annotated English translation of the Sanskrit-Old Javanese Bhuvanakośa (used with the author's permission).

Emily M. Bender, Dan Flickinger, and Stephan Oepen. 2002. The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In *COLING-02: Grammar Engineering and Evaluation*.

Emily M. Bender, Scott Drellishak, Antske Fokkens, Laurie Poulson, and Safiyyah Saleem. 2010. Grammar customization. *Research on Language and Computation*, 8(1):23–72.

Emily M. Bender, Michael Wayne Goodman, Joshua Crowgey, and Fei Xia. 2013. Towards creating precision grammars from interlinear glossed text: Inferring large-scale typological properties. In *Proceedings of the 7th workshop on language technology for cultural heritage, social sciences, and humanities*, pages 74–83.

Emily M. Bender, Joshua Crowgey, Michael Wayne Goodman, and Fei Xia. 2014. Learning grammar specifications from IGT: A case study of Chintang. In *Proceedings of the 2014 Workshop on the Use of Computational Methods in the Study of Endangered Languages*, pages 43–53.

Balthasar Bickel, Bernard Comrie, and Martin Haspelmath. 2008. The Leipzig glossing rules: Conventions for interlinear morpheme-by-morpheme glosses. Max Planck Institute for Evolutionary Anthropology and Department of Linguistics, University of Leipzig, 2008. `http://www.eva.mpg.de/lingua/resources/glossing-rules.php`.

Balthasar Bickel, Sabine Stoll Stoll, Martin Gaenszle, Novel Kishor Rai, Elena Lieven, Goma

Banjade, Toya Nath Bhatta, Netra Prasad Paudyal, Judith Pettigrew, Ichchha Purna Rai, Manoj Rai, Taras Zakharko, and Robert Schikowski. 2013. Audiovisual corpus of the chintang language, including a longitudinal corpus of language acquisition by six children, paradigm sets, grammar sketches, ethnographic descriptions, and photographs.

Elizabeth Conrad. 2021. Tracing and reducing lexical ambiguity in automatically inferred grammars. Master's thesis, University of Washington.

Ann Copestake. 2002. *Implementing typed feature structure grammars*. CSLI publications Stanford.

Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan A Sag. 2005. Minimal recursion semantics: An introduction. *Research on language and computation*, 3(2):281–332.

Berthold Crysmann and Woodley Packard. 2012. Towards efficient HPSG generation for german, a non-configurational language. In *Proceedings of COLING 2012*, pages 695–710.

Christian Michael Curtis. 2018. A parametric implementation of valence-changing morphology in the lingo grammar matrix. Master's thesis, University of Washington.

Desmond C. Derbyshire. 1979. *Hixkaryana syntax*. University of London.

Allison Dods. 2022. Automatically inferring grammar specifications for adnominal possession from interlinear glossed text. Master's thesis, University of Washington.

Scott Drellishak. 2009. *Widespread but not universal: Improving the typological coverage of the Grammar Matrix*. PhD thesis, University of Washington.

Scott Farrar and William D. Lewis. 2007. The GOLD community of practice: An infrastructure for linguistic data on the web. *Language Resources and Evaluation*, 41(1):45–60.

Ryan Georgi. 2016. *From Aari to Zulu: massively multilingual creation of language tools using interlinear glossed text*. PhD thesis, University of Washington.

Michael Wayne Goodman. 2013. Generation of machine-readable morphological rules from humanreadable input. *University of Washington Working Papers in Linguistics*, 30.

Michael Wayne Goodman, Joshua Crowgey, Fei Xia, and Emily M. Bender. 2015. Xigt: extensible interlinear glossed text for natural language processing. *Language Resources and Evaluation*, 49(2):455–485.

Harald Hammarström, Robert Forkel, Martin Haspelmath, and Sebastian Bank. 2022. Glottolog 4.7. Leipzig: Max Planck Institute for Evolutionary Anthropology. `https://doi.org/10.5281/zenodo.7398962` (Available online at `http://glottolog.org`, Accessed on 2023-03-07.).

Heidi Harley. 2019. Hiaki text corpus. University of Arizona. Unpublished FieldWorks (FLEx) project. (Accessed August 2019).

Heidi Harley, Mercedes Tubino Blanco, and Jason Haugen. 2009. Applicative constructions and suppletive verbs in Hiaki. *Rice working papers in linguistics*, 1:42–51.

Martin Haspelmath and Thomas Müller-Bardey. 2004. Valency change. *Morphology: A handbook on inflection and word formation*, 2:1130–1145.

Charlotte Hemmings. 2013. Causatives and applicatives: The case for Polysemy in Javanese.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spaCy: Industrial-strength natural language processing in python.

Kristen Howell. 2020. *Inferring grammars from interlinear glossed text: Extracting typological and lexical properties for the automatic generation of HPSG grammars*. PhD thesis, University of Washington.

Daniel Kaufman, Husniya Khujamyorova, and Ross Perlin. 2020. Wakhi texts. *Digital collection managed by KRATYLOS*. Uploaded from www.elalliance.org, Wakhi. In Finkel, R. and Kaufman, D., Kratylos: Unified Linguistic Corpora from Diverse Data Sources.

Uploaded April 28, 2020 and retrieved from https://www.cs.uky.edu/ raphael/ela/ on May 20 2020.

František Kratochvíl. 2007. *A grammar of Abui: A Papuan language of Alor*. Leiden University.

František Kratochvíl. 2019. Abui corpus. Electronic Database: Unpublished toolbox project (accessed March 2019).

William D. Lewis. 2003. Mining and migrating interlinear glossed text. In *Workshop on Digitizing and Annotating Texts and Field Recordings*.

William D. Lewis and Fei Xia. 2010. Developing ODIN: A multilingual repository of annotated language data for hundreds of the world's languages. *Literary and Linguistic Computing*, 25(3):303–319.

Sérgio Meira. 2020. Hixkaryana lexicon and texts. Unpublished Toolbox project. (Accessed March 2020).

Stephan Oepen. [incr tsdb()] – Competence and performance laboratory User manual. Technical report, Computational Linguistics, Saarland University, Saarbrücken, Germany, 2001.

Woodley Packard. 2015. Full forest treebanking. Master's thesis, University of Washington.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Carl Pollard and Ivan A Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press.

Chris Rogers. 2010. Fieldworks language explorer (FLEx) 3.0. *Language Documentation & Conservation*, 4.

Safiyyah Saleem. 2010. Argument optionality: A new library for the grammar matrix customization system. Master's thesis, University of Washington.

Willi Schaub. 1982. *Babungo*. London, etc.: Croom Helm.

Aviva Shimelman. 2012. Yauyos Quechua Collection of Aviva Shimelman, ailla.utexas.org. PID ailla:119766. (Accessed January, 2023).

Aviva Shimelman. 2017. *A grammar of Yauyos Quechua*. Language Science Press.

Anna Siewierska. 2004. *Person*. Cambridge: Cambridge University Press.

SIL International. 2015. Field linguist's toolbox. Lexicon and corpus management system with a parser and concordancer. `https://software.sil.org/toolbox/`.

Lucien Tesnière. 1959. *Éléments de syntaxe structurale*. Paris: Librairie C. Klincksieck.

Nick Thieberger. 2006. Dictionary and texts in South Efate. *Digital collection managed by PARADISEC [Open Access]*. (Accessed March 2019).

Mercedes Tubino Blanco. 2011. *Causatives in minimalism*. John Benjamins Publishing Company.

Jan F Ullrich and Ben Black Bear. 2018. *Lakota grammar handbook*. Lakota Language Consortium.

David Allen Wax. 2014. Automated grammar engineering for verbal morphology. Master's thesis, University of Washington.

Olga Zamaraeva. 2016. Inferring morphotactics from interlinear glossed text: combining clustering and precision grammars. In *Proceedings of the 14th SIGMORPHON Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 141–150.

Olga Zamaraeva, František Kratochvíl, Emily M. Bender, Fei Xia, and Kristen Howell. 2017. Computational support for finding word classes: A case study of Abui. In *Proceedings*

of the 2nd Workshop on the Use of Computational Methods in the Study of Endangered Languages, pages 130–140.

Olga Zamaraeva, Kristen Howell, and Emily M. Bender. 2019. Handling cross-cutting properties in automatic inference of lexical classes: A case study of Chintang. In *Proceedings of the 3rd Workshop on the Use of Computational Methods in the Study of Endangered Languages Volume 1 (Papers)*, pages 28–38.