Finite-state Parsing of Cayuga Morphology

by

© Dougal Graham

A thesis submitted to the
School of Graduate Studies
in partial fulfillment of the
requirements for the degree of
Master of Arts

Department of Linguistics
Memorial University of Newfoundland

June 2007

St. John's Newfoundland and Labrador

Abstract

This paper presents a detailed description of the design and implementation of a computerized morphological segmentation tool for Cayuga nouns. Speakers of polysynthetic First Nations languages are presented with an array of difficulties when it comes to word segmentation and dictionary access. This program demonstrates that finite-state techniques are applicable to these morphologically complex languages and are worth further study and development in order to create useful tools for speakers and learners of thesis languages.

Acknowledgments

I would like to thank my supervisor, Dr. Carrie Dyck, for the constant support and insight she provided me during the course of this project and the writing of this document and for travel funding. As well, I thank Dr. Todd Wareham for providing me with an introduction to the world of finite-state computing.

My parents have been very supportive throughout my work, and my academic career, for which I am very thankful.

The kind donation of on-campus working space by Dr. Yvan Rose was also very much appreciated and saved me a lot of walking time.

I would like to thank the department of linguistics at Memorial for helping me to stay well organised, despite my worst efforts, and for the funding I received.

Lastly, I'd like to thank Krista Gammon for encouraging me to keep on working and persevering with me.

Abbreviations

Terminology

FS Finite-State

FSM Finite-State Machine

FSA Finite-State Automaton

FST Finite-State Transducer

TTS Text-to-speech

NLP Natural Language Processing

kB Kilobytes

Glosses

UR Underlying representation

SR Surface representation

A Agent: arbitrary term used to denote the type of prefix that represents the subject of an active transitive verb

P Patient: arbitrary term used to denote the type of prefix that represents the direct object of an active transitive verb

1-3 1st to 3rd persons

i Inclusive: the 2nd person is included in the speech act

e Exclusive: the 2nd person is excluded from the speech act

s Singular: one participant

d Dual: two participants, including the speaker and another

p Plural: three or more participants

dp Non-singular: two or more participants

m Masculine

f(i) Feminine(-Indefinite): refers to females, a mixed group of people or 'someone'

zn Zoic-Neuter: refers to animals or inanimate objects

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Recently, finite-state methodology has emerged as the primary framework for natural language processing (NLP) and computational models of languages (Beesley and Karttunen, 2003:pg. XV). In the past several years, much work has been done to create finite-state implementations of lexicons, spelling correction systems (Vilares *et al.*, 2004), morphological parsers (Reichel and Weilhammer, 2004), speech synthesis programs and so forth for a wide variety of languages (Alegria *et al.*, 2002; Beesley and Karttunen, 2003; Beesley and Karttunen, 2000). The initial demand for the production of these sorts of systems tends to fall on the most widely spoken languages, and those with the longest linguistic traditions, such as English, Finnish, German, French and other European languages. Not much, if any, work, however has yet been done with respect to the specific needs of First Nations languages in particular or, more generally, polysynthetic languages.

Such work would, however, be worthwhile. It would allow for a thorough and systematic evaluation of the posited morpho-phonological rules for the language studied. It would extend our understanding of finite-state machines (FSM) since polysyntehtic languages tend to be significantly more morphologically complex than the European languages traditionally studied. Furthermore, such a project could lead to further useful applications in the future such as an auto-segmenting dictionary or text to speech (TTS) system for the language.

In that light, I have created an FSM that recognizes[1], generates[2] and segments all and only the valid noun forms of Cayuga (Iroquoian) with four primary goals in mind.

The first goal is to demonstrate that useful segmentation tools could be quickly generated for morphologically complex languages in which speakers encounter difficulties because of segmentation issues. Secondly this project will provide a thorough testing of the morpho-phonological rules currently posited for Cayuga nominal forms. Thirdly the project will determine the potential computational costs and usability issues of designing finite-state transducers (FST) with different kinds of output. Finally, this project will provide a preliminary investigation into the applicability of the finite-state framework to polysynthetic languages.

This document has been designed so as to satisfy the needs of the disciplines of both linguistics and computer science. With that in mind, it should be clear why this document is somewhat different from a standard linguistics thesis, and from a computer science thesis. The first half of the document (chapters 2-4) are primarily linguistically oriented. Chapter 5 sits nicely on the fence and forms a bridge into the latter half of the document (chapters 6-7) which are generally more computationally oriented.

The document is organized as follows: chapter 2 discusses prior work in finite-state natural language processing; chapter 3 contains an overview of what finite-state machines are and how they work; chapter 4 is a discussion of Cayuga nominal morphology and morpho-phonology; the 5th chapter describes some linguistic and computational problems arising from the data discussed in chapter 4. Chapter 6 lays out the requirements for the creation of the program and describes the tools and data structures used to create it. Chapter 7

[1]A finite-state machine that recognizes all and only the valid words of a language reports an error on any word that does not exist in the language and never reports an error for a word that does exist in the language.

[2]A finite-state machine can also be set to output all words that it recognizes (footnote 1).

contains a detailed explanation of the program and its components as well as an analysis of how well the program was able to meet the requirements laid out in chapter 6. Finally, chapter 8 contains an overview of lessons learned and potential applications for future work.

Chapter 2

Prior Work

The applications of FSMs have been an important domain for research in the fields of mathematics, computer science and more recently, linguistics. Numerous applications make use of FSMs to work with a wide variety of languages, from Spanish (Tzoukerman and Liberman, 1990; Tinsley, Accessed: 2007 02 13), to Finnish (Koskenniemi, 1997) to Arabic and Malay (Beesley and Karttunen, 2000).

Finite-state technology is widely used in the field of NLP (Beesley and Karttunen, 2003: and references; Roche and Schabes, 1997: and references; Mohri, 1997: and references; Karttunen, 2001: and references), but the focus has been on European languages(Tzoukerman and Liberman, 1990; Kiraz and Möbius, 1998), with some attention paid to agglutinating[1] languages(Koskenniemi and Church, 1988; Oflazer, 1994), but no attention to highly polysynthetic[2] languages. I will briefly describe here a few implementations of finite-state technology that deal with a variety of languages in order to illustrate the versatility of finite-state methodology.

[1]Agglutinative: "A language in which words typically contain a linear sequence of morphs" (Crystal, 2003:17)

[2]Polysynthetic: "A language "characterized by morphologically complex long word forms" (Crystal, 2003:359)

2.1 On-Line Spanish Morphological Analyser/Generator

Finite-state methods are ideal for the decomposition and analysis of concatenative[3] morphology. John Tinsley developed a morphological analyser and generator for Spanish using *XFST* (Tinsley, Accessed: 2007 02 13). The machine takes user input in the form of sentences, parses them into single-word tokens using a program called *tokenize* and analyses them individually using another program named *lookup*, then returns the result to the user.

The analyser currently achieves approximately 85% coverage on unrestricted text. The machine parses each word-form from a lexical form into a surface form as follows:

(1) Lexical: hablar+Verb+PresInd+1P+Sg
Surface: hablo

Tinsley's model is a good demonstration of how finite-state machines can be applied to practical applications. The entered word is associated with a set of semantic tags describing the class of word (noun, verb, etc.) and also its semantic constituents (singular, plural, present, past etc.) all of this information is then output to the user in an easy to read manner.

2.2 Templatic Morphology and Reduplication

Initially, it is unclear whether FS technology can handle complex morphosyntactic phenomena such as reduplication and templatic morphology. Recursive processes in natural languages, such as reduplication, are context sensitive and therefore cannot be generally represented as finite-state (Chomsky, 1956). However, these non-finite-state aspects of lan-

[3]Concatenative: "Characterised by the joining together of a linear sequence of morphemes" (Crystal, 2003:93)

guage can be implemented in the finite-state for any specific case of a bounded length. (Frank and Satta, 1998)

The finite-state framework can not only deal with concatenative morphology but also complex non-concatenative morphology such as reduplication as found in Malay or templatic morphology as found in Arabic. *Compile-replace* is a formalism developed by Beesley and Karttunen (2000:375-420) to handle phenomena such as templatic morphology(2) and reduplication(3), and is a great example of the versatility and applicability of finite-state methods to a variety of morphologically complex languages.

In (2) we see an example of templatic morphology in Arabic in which there is a consonantal root template which can take a variety of vowels to convey aspect, voice, etc.

(2) a. k _ t _ b -Triliteral Root
b. C a C a C -Template
c. katab -Surface Representation

The *compile-replace* formalism makes it possible to create general templates (2-a, 2-b) from specific words (2-c). The consonantal and vowel templates can be separated by one FST and then processed by a second FST in order to determine the meaning of the triliteral root and the meaning of the CaCaC templatic morpheme.

Without *compile-replace*, reduplication can be difficult to formalise in a traditional finite-state framework. There are, essentially, two types of reduplication: fixed-length reduplication in which a constant number of phones or syllables are reduplicated or full-length reduplication in which an entire morpheme or group of morphemes form is copied.

Fixed-length reduplication is easily formalised by specifying the criteria describing what must be reduplicated. Full-length reduplication, however, is of variable length and relies

on information that lies in the lexicon (i.e., a stem, or some other morpheme group) of potentially unbounded length. This means reduplication cannot easily be specified using only a normal rule-based formalism, since it does not have access to lexical information.

In (3) we see an example of the type of full-stem reduplication that *compile-replace* can handle.

(3) a. buku
book
book

b. buku -buku
book -Reduplication (Plural)
books

The programmer can specify "buku" as a stem that may be reduplicated and the *compile-replace* algorithm will take such information into account, producing the form in example (3-b). *Compile-replace* can also handle less complex types of reduplication such as reduplicated prefixes and affixes. One need only specify the sub-lexicon containing the forms that can undergo reduplication and the rule for processing them and then XFST is able to copy the form according to the specifications.[4]

This work demonstrates the finite-state framework's ability to deal with both concatenative and non-concatenative morphology. The following discussion explains other applications of the finite-state framework in NLP such as syllabification and spelling correction.

[4]XFST and the notion of "sub-lexicon" are detailed in chapter 6.3

2.3 Syllabification

Text to speech (TTS) systems often face the problem of creating a natural intonation contour and stress pattern in the words they produce. Such patterns are generally based at least partly upon the syllable structure of the words being spoken. It is therefore often very desirable to have an algorithm for the syllabification of word-forms.

With these applications in mind, Kiraz and Möbius (1998) created a finite-state application in order to syllabify German and English words. For example, the English nouns below in (4) and (5)[5] take different syllabifications, affecting the pronunciation:

(4) a. Nightrate

b. [naɪt˺-ɹet]

(5) a. Nitrate

b. [naɪ-t̥ɹet]

As can be seen in (4)-(5) the different syllabifications affect both the voicing of the [ɹ] and the release of the [t].

This, particular example, however, could not at first be accounted for using Kiraz *et al.*'s syllabification system. Because the two words are underlyingly identical (UR: /naɪtɹet/), the machine has no way of determining that there is a difference between the two. In this case, Kiraz *et al.* were able to encode references to morpheme boundaries as a way of dealing with this specific problem. By specifically defining a morpheme boundary, the machine was able to syllabify taking that reference point into account, thereby generating

[5]From (Kiraz and Möbius, 1998)

proper syllabifications for most compound nouns. However, Kiraz *et al.* mention (without examples) that in some cases even the specific designation of morpheme boundaries was not sufficient for the machine to determine which syllabification was correct, and so in some cases the most common syllabification was not always correct.

The Kiraz *et al* algorithm also used a probabilistic 'weighting' method to determine which syllabification was *most likely* in a given case. This allowed the machine to run more quickly (as per the demands of a TTS system), but had the drawback of occasionally producing incorrect output if the most common syllabification was not always correct. Despite these algorithmic drawbacks, Kiraz *et al.*'s system is a good example of the use of the finite-state framework for the development of a complex high speed NLP application. A further example of such an application follows below.

2.4 Spelling Correction

As a back end to future NLP work in Basque, Alegria *et al.* (2002) created a finite-state spelling correction program and analyser/generator composed of three separate modules. This modular approach lends itself well to the finite-state framework, as discussed in more detail in §3.6.

The first module checks the spelling of a word against the standard Basque spelling. If unable to find a correct spelling in the first module, the machine then employs the second module to see if the word conforms to certain dialectal variances or common competence errors. Finally, should both those options fail, the program accesses a third module that attempts to guess what form the user was attempting to produce such that it might be corrected. The advantage of such an approach is that one can implement all three dictionary types in a single machine. The second and third modules are especially useful for Basque,

since there are numerous regional dialectal variances.

Another spell-checking/error-correction system has been developed for Galician (Vilares *et al.*, 2004), and a system has been developed for Turkish as well (Oflazer, 1994). This wide variety of languages for which such systems have been developed demonstrates that finite-state techniques are widely cross-linguistically applicable. Additionally such systems have a variety of uses, for example in checking potential mis-spellings in an online dictionary.

2.5 Finite-state Applications to Polysynthetic Languages

One motivation for this project was to attempt to determine if the finite-state framework would be computationally and practically useful for designing tools for morphologically complex languages.

Most languages for which finite-state approaches have been used are less morphologically complex than First Nations languages (§2). There are several aspects of First Nations languages that could cause problems for the finite-state framework and it is necessary to determine if the framework is computationally adequate for the needs of these languages.

Most Indo-European languages do not have extensive obligatory prefixation, and furthermore only generally allow for a small number of a small set of prefixes to attach to a word, with very few changes occurring; the same is true of their suffixes. Cayuga words, however, require a high number of obligatory prefixes and many allomorphic rules.

For example the finite-state analyzer of Spanish applies only 4 replace rules to deal with changes to the form of noun stems and affixes (Tinsley, Accessed: 2007 02 13). Cayuga, on the other hand, will minimally require 10. The greater number of morpho-phonological rules and the need to encode long-distance dependancies are the two primary potential sources of

computational complexity facing the design of a finite-state application for a First Nations language.

As discussed in §2.2, other very morphologically complex languages, such as Arabic and Malay, have benefited from applications implemented in the finite-state framework (Beesley and Karttunen, 2003). However, these languages display a different kind of complexity of morphology from First Nations languages. While reduplication and templatic morphology are very complex, they are essentially different from the large degree of prefixation, morphological variation and long-distance dependancy that occur in First Nations languages. The following chapters demonstrate that despite these differences and challenges, it is generally plausible to implement polysynthetic morphology in the FS framework.

Chapter 3

Finite-state machines

Before describing the structure of my program, it is necessary to define finite-state machines and some of their formal properties.

Abstractly, finite state machines (FSM) are a model of behavior, which is described as consisting of states, transitions and actions.[1] More concretely, they can be viewed as flexible computer programs that can implement a wide variety of NLP tasks (§2). FSMs can be broken down into two main sub-categories, the finite-state automata (FSA), and the finite-state transducers (FSTs). In this chapter I will briefly discuss the properties of both FSAs and FSTs.

3.1 FSA as a Model of Behaviour

An FSA is the simplest type of FSM, and can be used to model a behaviour. An FSA takes input from a user, but the only feedback it gives is whether or not the input is valid. It can also generate a list of all valid input sequences.

Diagrammatically, an FSM is much like a flow chart: one starts at a given point and moves following the appropriate arrows in a diagram. Each circle is called a 'state' (1-a) (generally denoted with the letter 'q' and a subscript number). A circle marked with an arrow represents the initial, or starting, state; and a circle with double lines is a potential

[1] A full review of basic information on FSMs, can be found in Beesley and Karttunen (2003) and Nederhof (1996)

final state. Each arrow joining two states is called a **transition** (1-b). Transitions have conditions (1-c) attached to them that determine which state to move to next.

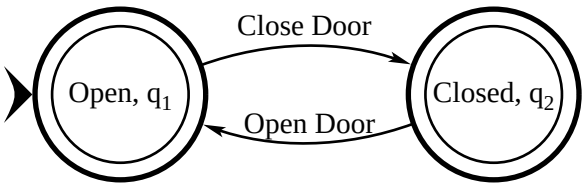
(1) a. States: 

b. Transitions: →

c. Conditions/Actions: "open door", "close door", etc.

An action can be performed either upon entering a state, leaving a state or during a transition. (Wikipedia, 2006; Beesley and Karttunen, 2003; Sproat, 1992). In the following example I will use the concept of a door, and the user's input is the action of either opening or closing the door. The machine shown in (2) can generate a list of all valid sequences of opening and closing the door.

(2) An FSA representing the use of a door (Wikipedia, Accessed: 2006 03 24)[2]



From the 'open' state the door can be 'closed' by performing the 'close door' action and following the topmost arrow to the 'closed' state. Or, if there are no more actions, then the machine stops since 'open' is a valid final state. If the machine moves to the 'closed' state, it can either stop if there are no further actions, or it can perform the 'open door' action to move once more into the 'open' state.

However, if one were to try to tell the machine to close the door from the closed state,

[2]This example from Wikipedia was used because of its simplicity and clarity. More complex examples can be found in (Beesley and Karttunen, 2003), (Sproat, 1992) and elsewhere in this document.

there would be an error because the 'close' action flows into the 'closed' state, not out from it. The machine can stop running when it reaches either final state, or it can continue indefinitely so long as the actions being performed are a series of alternating openings and closings.

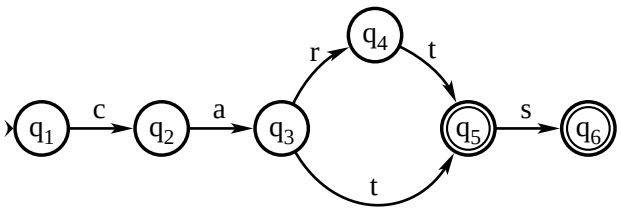
3.2 FSA as Linguistic Model

FSAs can also model linguistic behaviour, by treating transitions as symbols, with all possible paths through an FSA defining a 'language'. A formal language means merely the collection of strings[3] that the FSA will recognize and generate. Suppose we have a language; "Splort"; with only four words, as defined below:

(3) Language 'Splort':
   a. Cat
   b. Cats
   c. Cart
   d. Carts

This language can be expressed by the following FSA:

(4)



The automaton diagrammed in (4) can both recognize and generate all the strings in the language Splort. If the FSM were to generate all words it would generate each possible word

[3]A string is a linear sequence of symbols (words, phones, morphs, graphs, or even features can all be viewed as symbols) that can be recognized by the machine, where each symbol is an indivisible unit. Symbols are defined within an FSM, such that one machine make take words to be indivisble symbols and another letters. In the context of my program letters can be accepted as the basic symbols along with a few diacritics to mark morpheme and word boundaries.

in sequence. For example, starting in state q_1, the only possible option for the first output symbol is 'c'. Once the machine has output 'c' it moves to state q_2.

From q_2 the machine will have no choice but to output 'a' and move to q_3. At q_3, however, the machine will have to choose between moving to q_4 and outputting 'r', or q_5 and outputting 't'. If the machine moved to q_5 it could stop there and go back to q_1 to output another word, or it could continue on to q_6 and output 's'. If the machine moved to q_4, on the other hand, it would have to continue on at least until q_5 before stopping. Once it was done outputting one form, it would then move on to another until all possible paths through the machine had been completed.

The automaton can also check if a given word exists in the language by comparing the first symbol (letter) in a candidate word against all possible transitions from the start state q_1. If there is a transition whose condition matches the first letter from the given word, the automaton would move on to the state joined by that transition (q_2) and for a transition whose condition matches the next symbol in the word. If, however, there are no transitions matching the next letter, or if the automaton runs out of letters while in a non-final state, then the automaton can tell us that the given word does not exist in the language. If for example, we wished to test that the word 'can' is in the language "Splort" using (4), the machine would arrive at state q_3, but it would then report an error, since there is no transition with the condition 'n' attached to it leading out of q_3. Similarly, the word 'car' would also fail, because q_4 is a non-final state, so 'car' is not a word that exists in "Splort".

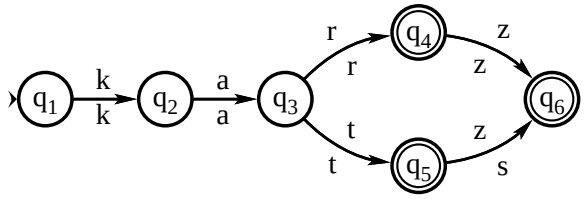
An FSA can both recognize and generate a language that consists of any number of strings. Finite-state Transducers, described below, are even more powerful: they are able to represent the relationship between two languages.

3.3 Finite-state Transducers

A more complex type of FSM is the finite-state transducer (FST). An FST not only generates and recognizes valid strings from a language, but also generates output. It maps the relation between two strings (or languages) by allowing one to convert one string (or language) into another (Beesley and Karttunen, 2003; Nederhof, 1996; Sproat, 1992). This model can implement transformational rules that convert an underlying form into a surface form and vice-versa.

Example (5) diagrams an FST which converts between underlying and surface forms, modeling allomorphy.

(5) An FST transducing UR to SR for cat(s) and car(s):



In the FST formalism, all characters that appear above a transition line are part of one language, often referred to as the "input language", while those below the line are their counterparts in the so-called "output language". The FST in (5) translates between the underlying representation and the pronunciation by applying the rules for plural suffix allophones (ie: devoicing [z] following a voiceless obstruent). If the above machine were to be fed the string [kætz] it would output [kæts]. Conversely it would also be able to output [kætz] as the UR form if it were given [kæts] as the SR. This is due to the property of bidirectionality, described below.

3.4 Bidirectionality

All FSTs, are actually bidirectional. This means that they can be run just easily "backwards" as "forwards". This distinction does not mean running from a final state to a start state, but instead it means generating an 'input' from an 'output'. The labels 'input' and 'ouput' are really only useful for distinguishing which specific language one is working with in the context of a given FST.

Bidirectionality is useful because the machines created are able to fulfill the roles of (a) generator/recognizer of 'input' forms, (b) generator/recognizor of 'output' forms, (c) translator between 'input' and 'output' forms and, (d) generator/recognizer of input-output form pairs. In the case of (a), the machine just ignores the "upper" transition labels, acting like an FSM. For (b) it does the same but in reverse, ignoring the "lower" transition labels. To account for (c) the machine follows the "upper" or "lower" path and outputs the symbols on the other side of the transition. For (d) the machine outputs both symbols on each

transition as it follows the path.

One could take a collection of underlying morphemes and use them to generate the surface forms, and with the same FST then take the surface forms and break them down into their constituent parts again. All FSTs described in §2 are bidirectional. For example, the German/English syllabification system described in 2.3 could de-syllabify words as well as syllabify them.

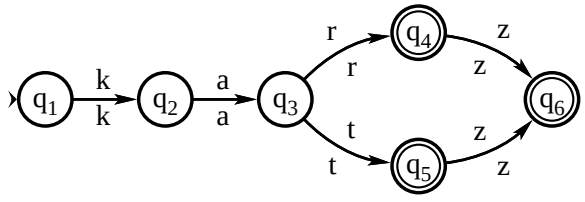
The bidirectional properties of FS machines greatly increases their usefulness and applicability to NLP. The specific applications of bidirectionality will be discussed in greater detail in chapter 6.

3.5 Composition

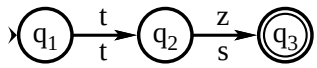
Up until now, I have discussed FSTs as individual discrete units. Here I will explain the composition operation that allows multiple small FSTs to be used as component parts of a single larger FST. Any two FSTs can be joined together into a single FST by composition (Kaplan and Kay, 1994). Composition allows us to initially declare two rules, for example, a→c and c→g and then morph them into a single rule: a→g. This only works when the output side of the first machine is identical to a valid input form of the second machine. Where there is such an occurrence the original output of the first machine is replaced with the output of the second machine.

Example (6) encodes a concatenation of symbols as a transducer where the UR and SR are identical. Example (7) encodes a rule that defines a morpho-phonological change. Example (8), discussed later, is the product of the composition of (6) and (7), a machine that encodes the relationship between differing URs and their SRs.

(6)

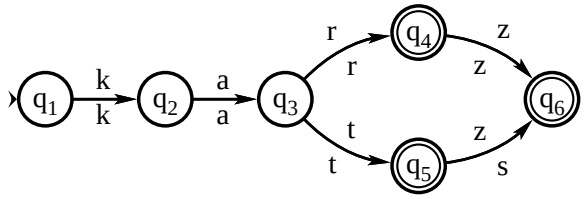


(7)



Composition works not only on single rules but also on entire FSTs; allowing us to join them together with rules. In order to compose the two transducers defined in (6) and (7) all one needs to do is look for an output sequence from (6) that is identical to the input sequence of (7) (tz). That output sequence ((6): $q_3 \rightarrow q_5 \rightarrow q_6$) is then replaced with the output from (7) ($q_1 \rightarrow q_2 \rightarrow q_3$: ts):

(8)



We can see now in (8) that the output for $q_3 \rightarrow q_6$ has changed to match the output from (7).

The process of composition means that we can separately formulate all the rules that we wish to implement, and then compose them together to create a final product. If we determine that a rule is mis-ordered or unnecessary, it can be moved or changed and then we can re-compose the rules. This mechanism emulates the sort of cascading or ordered rules commonly used to define morpho-phonological alternations.

Composition means that FSTs are almost infinitely expandable, as long as we stay within the bounds of disk space and processing power. Some examples of the power of FSTs were described above (chapter 2) including Alegria *et al.*'s (2002) spelling correction system, demonstrating the power of modularity in FSTs by being constructed of three separate modules composed together as one.

The drawback to composition is that can cause an FST to become very complex. The

un-minimized product of the composition of two FSTs has a number of states equal to the number of states of each original machine multiplied together. In some cases few of these extra state are redundant and cannot be removed, potentially allowing machine size to double with each composition.(Kaplan and Kay, 1994) If we are constantly replacing a rather simple group of states and transitions with a much more complex group, it is easy to imagine how complexity can be introduced rather quickly.

3.6 Modularity

A final property of FS machines is that of modularity. When creating a finite-state machine to recognize and generate valid words in a language, it is common practice to create a number of separate finite-state machines that each shoulder some small part of the burden of recognition and generation and join them together into a single final product using composition. This concept is known as "modularity" (Beesley and Karttunen, 2003:284). Broadly, it is possible to look at two main modules for most finite-state implementations of morphological parsers: the *lexicon module*, and the *rules module*.

The lexicon module is essentially a representation of the mental lexicon of underlying root forms and affixes, with as little redundancy included as possible. As few semantically equivalent allophones were included as possible. The idea is to not represent any regular phonological alternations, in order to reduce the storage of redundant forms in the lexicon. This module contains a description of which morphemes are valid in the language, and to which other morphemes they can join, and under what conditions.

The rules module can be viewed as the sum of several separate transducers, each of which represents some sort regular alternation in the language. For example, one rule might be /s/ $\rightarrow$ [z] / C_{vd}_ (as in (5)). By then joining these rules and the lexicon, a final machine is created which can join morphemes together and apply any applicable alternation rules to those joined morphemes.

Polysynthetic First Nations languages may require either more complex lexicon modules, or more complex operators for implementing some morpho-phonological rules than have been used with other languages. There is also some question of how much of the alternation should be viewed as a result of morpho-phonological rule derivations and how much should be viewed as lexical (see §6.1.1.1). I will attempt to discuss those parts of Cayuga that may

cause problems for either the lexicon module or the rules module, given current methodology for implementing those modules. The following chapter contains the Cayuga data which was implemented in my program as well as a description of its morphology and morpho-phonological processes.

Chapter 4

The Cayuga Language

Cayuga is an Iroquoian language, closely related to Oneida, Seneca and Mohawk. Traditionally, the Cayuga and Seneca peoples lived in modern day Cayuga County, New York, but they have since relocated and Cayuga is currently spoken by approximately 100 people at Six Nations (Froman *et al.*, 2002). Cayuga is a polysynthetic language, characterized by "morphologically complex, long word-forms" (Crystal, 2003:359). Combining affixes and roots often results in morphophonemic changes, making the final word difficult to decompose, and the stem hard to identify even for native speakers. If one cannot identify the stem, then it becomes impossible to search for a stem in a dictionary.

Given that Cayuga has numerous obligatory prefixes, there are too many forms to feasibly list in a dictionary. For this reason, dictionaries are organised by stems, or using a designated type of citation form. This means that a program that segments word-forms automatically would be invaluable in that it would make Iroquoian dictionaries accessible for everyone; the machine could determine the stem and prefix and look them up for the user. As it stands, there are a variety of ways to encode dictionaries for obligatorily prefixing polysynthetic languages but they are neither simple nor concise (See §5.1).

As a subset of the words of Cayuga the nouns are less complex than verbs, especially when considering that verbs can incorporate nouns. Despite being less complex, they are still rather complex relative to most common Indo-European languages and demonstrate important characteristics of Cayuga words in general, like obligatory prefixation. This combination of general complexity, yet relative simplicity compared to Cayuga verbs makes the nouns an ideal place to start work on the morphological parsing and segmentation of Cayuga words using FSTs.

This chapter will briefly discuss the orthography and spelling conventions of Cayuga, followed by a description of the morphology of the four classes of nouns that are treated by the program and a brief discussion of some of the morpho-phonological processes undergone by Cayuga nouns.

4.1 Orthography

There exist two commonly used orthographies for Cayuga: the standard Henry Orthography and the linguistic orthography. The sound-to-symbol combinations for each are described in table 4.1 below. The primary dissimilarities lie in the representation of the plosives and the affricates.

Table 4.1: Phonemic Inventory and Spellings

Phonetic Realizations	**Henry Orth.**	**Linguistic Orth.**
[d, t]	d	t
[t^h]	t	th
[ds, ʤ, ts, ʧ]	j	ts, tsy
[ts^h]	ts	tsh
[g, k]	g	k
[s]	s	s
[n]	n	n
[r]	r	r
[h]	h	h
[ʔ]	ˀ	ˀ

Phonetic Realizations	**Henry Orth.**	**Linguistic Orth.**
[i]	i	i
[ɑ]	a	a
[e]	e	e
[o]	o	o
[u]	u	u
[ɛ̃]	ę	ę
[õ]	ǫ	ǫ

For this project I have used a modified orthography to represent surface forms of words rather than either of the standards. The standard orthographies represent several predictable processes, including accent placement, and laryngeal metathesis.[1]

Table 4.2 demonstrates some of the differences between the orthographies. Note that the unmodified orthography essentially omits diacritics.

The advantage of such a modified orthography is that it allows me to focus on implementing the morpho-phonemic rules component, without needing to worry about processes such as accent placement and laryngeal metathesis. The accent placement rule, despite being predictable, is rather complex, and is not relevant within the scope of this project. Furthermore, an implementation of laryngeal metathesis would require a working stress accent

[1]The unmodified Henry orthography and the linguistic orthography use diacritics and spelling metathesis to encode LM. LM is not actually an instance of metathesis; it is a process of coalescence which affects metrically weak syllables.

Table 4.2: Alternate Orthographies

Modified Representation	hęnahsi˥da˥geh	sahsi˥da˥geh
Henry Orthography	hęnahs˥idá˥geh	sa̲hsí˥d˥ageh
Linguistic Orthography	hęnahs˥itá˥keh	sa̲hsí˥t˥ageh

placement algorithm which in turn requires a syllabification algorithm, including a means of syllable counting. The drawback to using a modified orthography such as this, however, is that it means that users will not be able to type words directly into this version of the machine since the spelling will differ from the standard spelling. At a later date, however, it is entirely feasible to create modules which translate to/from the modified representation into (a) the actual Henry orthography, or (b) the Linguistic orthography and vice versa.

4.2 Morphology of Cayuga Nouns

This discussion of the morphology of Cayuga is based on data from (Froman *et al.*, 2002). The nouns of Cayuga can be divided into five basic classes or types of nouns: defective nouns, basic or "regular" nouns, inalienably possessed nouns, de-verbal nouns and instrumental nouns.[2]

4.2.1 Defective Nouns

The simplest of the nouns are the defective nouns, which have no internal morphological structure. They do not take the regular affixes of other nouns as described below; instead they are composed of single lexicalised chunk:

(1) gwisgwis[3]
pig
pig

[2]Instrumental nouns display near identical properties to the de-verbal nouns and will therefore not be treated by this project.

[3]Not all defective nouns are apparent reduplicated forms e.g., tehtǫ˥*(ground hog, woodchuck, gopher)*

These nouns are computationally the simplest because they can each be stored as a single unit with no need for the finite-state transducer[4] to try to segment them. They are also the most accessible for users of traditional dictionaries since their meanings are invariable and they have no root form or alternate prefixes and are therefore easily located in a dictionary.

4.2.2 Basic Nouns

Basic nouns in Cayuga can take one of two forms; either possessed or unpossessed. All basic nouns consist of a prefix, a noun stem and a noun stem former. The prefix varies depending on whether the noun is possessed or not, and the type of the noun.

Unpossessed Basic Nouns

As described in example (2) below, unpossesssed basic nouns take either *ga-*, *o-* or *a-* as their prefix. The choice between the three prefixes is arbitrary and must be learned by the speakers.

(2) $\left\{ \begin{array}{l} \text{ga- (3znA)} \\ \text{o- (3znP)} \\ \text{a- (3znA)} \end{array} \right\}$ + noun stem +a˺ (NSF)

Example (3) gives a representative sample of nouns stems taking each of the three unpossessed prefixes.

(3) Sample Basic Nouns - Unpossessed

a. ga+ ˺wahsa: +a˺
3znA earring NSF
earring(s)

b. o+ ˺nhǫhs +a˺
3znP egg NSF
egg(s)

[4]See §3.3 for more info on FSTs

c. a+ ahdahditr +aˀ
3znA sock NSF
sock(s)

Some basic nouns have a choice to take either *ga-* or *o-* in their unpossessed form such as the word for in example (4-a):

(4) a. ga- jihoha: -aˀ
3znA- straight pin(s) -NSF
straight pin(s)

b. o- jihoha: -aˀ
3znP- straight pin(s) -NSF
straight pin(s)

Similarly, some basic nouns have the option to drop the 3znP prefix *o* as in example (5)

(5) a. ohǫnaˀdaˀ
potato(es)

b. hǫnaˀdaˀ
potato(es)

Possessed Basic Nouns

The possessed basic nouns take one of 12 prefixes called 'patient prefixes' that denote the gender, number and person of the possessor(s). All basic nouns take the same noun stem forming suffix *-aˀ*. Possessed basic nouns have the form described in example (6-a):

(6) a. Patient Prefix+ noun stem +aˀ (NSF)

b. age+ tsgoˀd +aˀ
1sP balsam fir NSF
my balsam fir

c. ǫkni+ tsgoˀd +aˀ
1dP balsam fir NSF
our (dual) balsam fir

In examples (6-b) and (6-c) the noun stem begins with a sequence of characters that matches the "elsewhere" column in table 4.3. Therefore it attaches prefixes such as *age-* or *yǫkni-* from that column.

Prefixes undergo both morpheme-initial and morpheme-final alternations, resulting in a great deal of allomorphy. The morpheme-final alternations are dependant upon the following stem-initial phones (as shown in tables 4.3 and 4.4). Depending on the final phones of the prefix and the initial phones of the stem, phones from either the stem or the prefix may be deleted or altered.

Table 4.3: Basic Nouns Patient Prefix Allomorphs - C-stems

Gloss	**Prefix UR**	**ˀCV**	**hV**	**hCV**	**nV**	**r**	**y/w**	**Elsewhere**[5]
1sP	(w)ag+	ag-	ak-*[6]	age-	ak-	ag-	ag-	age-
1dP	(y)ǫkni+	ǫkni-	ǫkni-	ǫkni-	ǫkni-	ǫkni-	ǫkni-	ǫkni-
1pP	(y)ǫgwa+	ǫgwa-	ǫgwa-	ǫgwa-	ǫgwa-	ǫgwa-	ǫgwa-	ǫgwa-
2sP	sa+	sa-	sa-	sa-	sa-	sa-	sa-	sa-
2dP	sni+	sni-	sni-	sni-	sni-	sni-	sni-	sni-
2pP	swa+	swa-	swa-	swa-	swa-	swa-	swa-	swa-
3msP	ho+	ho-	ho-	ho-	ho-	ho-	ho-	ho-
3fisP	(ya)go+	go-	go-	go-	go-	go-	go-	go-
3znsP	(y)o+	o-	o-	o-	o-	o-	o-	o-
3mdpP	hodi+	hodi-	hodi-	hodi-	hodi-	hodi-	hodi-	hodi-
3fidpP	(ya)godi+	godi-	godi-	godi-	godi-	godi-	godi-	godi-
3zndpP	(y)odi+	odi-	odi-	odi-	odi-	odi-	odi-	odi-

[5]The C-stem conditioning environments listed here (as well as for inalienable nouns below) are partial. For discussion of why this is the case, please see §7.2.1

[6]An * indicates that the first segment of the stem is deleted

Table 4.4: Basic Nouns Patient Prefix Allomorphs - V-stems

Gloss	**Prefix UR**	**i**	**a**	**e/ę**	**o/ǫ**
1sP	(w)ag+	ag-	ag-	ag-	ag-
1dP	(y)ǫkni+	ǫkn-	ǫgy-	ǫkn-	ǫkn-
1pP	(y)ǫgwa+	ǫgwę-*	ǫgw-	ǫgw-	ǫgy-
2sP	sa+	sę-*	s-a	s-	s-
2dP	sni+	sn-	j-	sn-	sn-
2pP	swa+	swę-*	sw-	sw-	j-
3msP	ho+	ho-*	ho-*	haw-	h-
3fisP	(ya)go+	go-*	go-*	gaw-	g-
3znsP	(y)o+	o-*	o-*	aw-	-
3mdpP	hodi+	hod-	hon-	hon-	hon-
3fidpP	(ya)godi+	god-	gon-	gon-	gon-
3zndpP	(y)odi+	od-	on-	on-	on-

The obligatory prefixation of the basic nouns, both possessed and unpossessed, can obscure the initial vowel segment of the noun stem. This means that these stems can be very difficult to locate in a dictionary. For example, when combined with the first person possessive prefix, the noun stem for 'egg' loses the glottal stop: [*aknhǫhsa*ˀ]. If naïve speakers were trying to look for the root in the dictionary, they might well search for a root looking like [-*nhǫhs*-] rather than [-ˀ*nhǫhs*-], assuming that they were even able to recognize that there were in fact two entities, the stem and the prefix.

In contrast to the prefix-final alternations, the prefix-initial alternations exist because of certain 'deleting phones' that are deleted word-initially. Example (7) below gives an example of a prefix in a position where its initial [y] is deleted (7-a) and in a position where the initial [y] remains intact (7-b).

(7) a. SR: o- ˀnhǫhs +aˀ
UR: (y)o+ ˀnhǫhsaˀ
3znP+ egg + NSF *eggs(s)*

b. SR: de- yo- ˀnhǫhs +a:ge:
UR: de+ (y)o+ ˀnhǫhs +age:
dualic+ 3znP+ egg +more-than-2 *two eggs*

The prefix-initial and prefix-final allomorphy described above is true of all pronominal prefix types including not only basic noun prefixes, but also inalienable noun and verbal prefixes.

4.2.3 Inalienably Possessed Nouns (Body Part Nouns)

Inalienably possessed nouns occur in three forms, a normal possessed form, a basic noun type of unpossessed form and a lexicalised unpossessed form that occurs only rarely and is idiomatic.

Possessed Inalienables

The structure of the inalienable nouns is similar to that of the possessed basic noun. Inalienable nouns take an obligatory prefix denoting the gender and person of the possessor, and an obligatory locative suffix. Rather than take patient prefixes, as the basic nouns do, they instead take one of 14 agent prefixes (see table 4.5). Example (8) below illustrates the structure of possessed inalienable nouns:

(8) Agent Prefix+ inalienable noun stem +aˀgeh (locative suffix meaning '*on*')

Example (9) gives two examples of inalienably possessed nouns:

(9) Inalienably Possessed Nouns (Body Parts)

a. s+ nɐ̨ts +aˀgeh
   2sA arm on
   (on) your (sg) arm
b. e+ nɐ̨ts +aˀgeh
   3f(i)A arm on
   (on) her arm

Agent prefixes display the same two types of allomorphy as described above for the possessed basic noun patient prefixes, including deletion of word-initial segments and prefix-

Table 4.5: Inalienably Possessed Noun Agent Prefix Allomorphs - C-Stems

Gloss	**Prefix UR**	**ˀCV**	**hV**	**hCV**	**nV**	**r**	**y/w**	**Elsewhere**
1sA	g+	k-*	k-*	k-*	k-	g-	g-	ge-
1idA	(e)kni+	kni-	kni-	kni-	kni-	kni-	kni-	kni-
1idA	(e)tni+	tni-	tni-	tni-	tni-	tni-	tni-	tni-
1edA	(y)akni+	akni-	akni-	akni-	akni-	akni-	akni-	akni-
1ipA	(e)dwa+	dwa-	dwa-	dwa-	dwa-	dwa-	dwa-	dwa-
1epA	(y)agwa+	agwa-	agwa-	agwa-	agwa-	agwa-	agwa-	agwa-
2sA	(h)s+	se-	s-*	se-	s-	s-	s-	se-
2dA	(h)sni+	sni-	sni-	sni-	sni-	sni-	sni-	sni-
2pA	(h)swa+	swa-	swa-	swa-	swa-	swa-	swa-	swa-
3msA	ha+	ha-	ha-	ha-	ha-	ha-	ha-	ha-
3f(i)A	(y)[ǫ/ę/e/ag]+	e-	e-	e-	e-	e-	e-	e-
3znsA	ga/(y)/w+	ga-	ga-	ga-	ga-	ga-	ga-	ga-
3mdpA	hadi+	hadi-	hadi-	hadi-	hadi-	hadi-	hadi-	hadi-
3f(i)dpA	gaǫ/gae/ga:g+	gae-	gae-	gae-	gae-	gae-	gae-	gae-
3zndpA	gadi+	gadi-	gadi-	gadi-	gadi-	gadi-	gadi-	gadi-

Table 4.6: Inalienably Possessed Noun Agent Prefix Allomorphs - V-Stems

Gloss	**Prefix UR**	**i**	**a**	**e/ę**	**o/ǫ**
1sA	g+	g-	g-	g-	g-
1idA	(e)kni+	kn-	gy-	kn-	kn-
1idA	(e)tni+	tn-	gy-	tn-	tn-
1edA	(y)akni+	akn-	agya-	akn-	akn-
1ipA	(e)dwa+	dwę-	dw-	dw-	gy-
1epA	(y)agwa+	agwę-	agw-	agw-	agy-
2sA	(eh)s+	s-	s-	s-	s-
2dA	(eh)sni+	sn-	j-	sn-	sn-
2pA	(eh)swa+	swę-*	sw-	sw-	sw-
3msA	ha+	hę-*	h-	h-	h-
3f(i)A	(y)[ǫ/ę/e/ag]+	ę-*	ǫ-*	ag-	ag-
3znsA	ga/(y)/w+	gę-	w-	w-	ø-
3mdpA	hadi+	had-	hęn-	hęn-	hęn-
3f(i)dpA	gaǫ/gae/ga:g+	gae-	gaǫ-*	ga:g-	ga:g-
3zndpA	gadi+	gad-	gęn-	gęn-	gęn-

final/stem-initial alternation and deletion processes. In (9) the stem begins with **nę** which fits the **nV** template in tables 4.5 and 4.6. In order to find the appropriate prefix form merely cross-reference the prefix type with the **nV** column in table 4.5 to determine the appropriate form of the prefix.

From tables 4.5 and 4.6 we see that there are two kinds of prefixes. Most have obviously related allomorphs, but some others have lexicalised prefix allomorphs. The **3fiA** (*(y)ǫ /(y)ɛ/(y)e/(y)ag*), **3fidpA** (*gaǫ/gae/ga:g*) and **3znsA** (*ga/(y)/w*) prefixes each have allomorphs that are historically unrelated, and cannot be derived using morpho-phonological rules; therefore each UR would be listed separately in a dictionary.

'Basic' Unpossessed Body Parts

Inalienable nouns can appear in basic noun form taking the **3znP** prefix *o-* and having a "detached" meaning (Froman *et al.*, 2002). These nouns are structured as in example (10):

(10) a. o+ inalienable noun root +a˥

b. o- nǫ˥a: -a˥
3znP head NSF
A (detached) head

These formations are considered semantically odd, except when used in compounds as in example (11):

(11) gwisgwis onǫ˥a:˥
pig head

Lexicalised Unpossessed Body Parts

Inalienable nouns may also occasionally appear in forms that take the **3znA** prefix *ga-* and have a lexicalised meaning as in (12-a). For comparison, see (12-b) which shows the corresponding inalienable possessed noun.

(12) a. ga- ya˺d -a˺
3znA body NSF
doll (basic noun)

b. g- ya˺d -a˺geh
1sA body NSF
on my body (inalienable body part)

4.2.4 De-verbal Nouns

The de-verbal nouns are formed by a nominal prefix, a verb root, a nominalizing suffix and a noun stem former. Essentially, the unit composed of verb stem plus nominalizing suffix acts exactly as the basic nouns described earlier.

(13) De-verbal Nouns
Prefix+ Verb stem +Nominalizing Suffix +NSF

a. ga+ tki +tr +a˺
3znA ugly Nominalizer NSF
'junk' - **(to be ugly)**

b. o+ yę +hsr +a˺
3znP lie.on.the.ground Nominalizer NSF
'blanket' - **(to be lying on the ground)**

The choice of nominalizing suffix is unpredictable, so it seems reasonable to treat these forms separately from actual nominal morphology. By this I mean that I will not be decomposing the derivational morphology within the de-verbal nouns, and I will be treating the combination of verb-stem + nominalizer as a regular nominal root that is not decomposed (-yęhsr- & -tkihsr- would each be considered a single unit in my system and act as a regular nominal root) (See §7.1). It would, however, be possible to replace such non-decomposed forms in the future with a module that correctly decomposes verbs and their derivational morphology.

4.3 Overview of Cayuga Morpho-phonological Variation

As has been alluded to in this chapter already, Cayuga words undergo a great deal of change when moving from the abstract underlying form to the surface form or vice versa. The morpho-phonological rules required to make these changes are listed fully in Appendix C, but I will briefly discuss a subset of them here.

A major feature of both Cayuga and other Iroquoian languages is a set of vowel hierarchy deletion rules (Hopkins, 1989). Vowels in these languages are ranked on a "strength hierarchy" according to which vowels are deleted when adjacent to a weaker vowel at morpheme boundaries.

(14) a. ǫ > o > ę > e > a > i

b. UR: gaǫ+ ahǫhd +aˀgeh
SR: gaǫ- hǫhd -aˀgeh

Example (14-a) describes the vowel hierarchy and (14-b) gives an example of a deletion caused by the hierarchy. As is clear from the example, this can easily cause the initial vowel of the stem or the final vowel of a prefix to be obscured.

There are two other additional rules which state that /w/ → [y] preceding [o, ǫ] and that /d/ → [g] preceding [y]. These two rules, in addition to the vowel strength deletion rules can all apply to a single prefix+stem combination; as shown in table 4.7.

Table 4.7: Multiple Rule Applications

UR:	(e)dwa+ǫts+aˀgeh
Hierachy deletions:	(e)dw+ǫts+aˀgeh
/w/ → [y]:	(e)dy+ǫts+aˀgeh
/d/ → [g]:	(e)gy+ǫts+aˀgeh
Other Rules→SR:	gyǫtsaˀgeh

Here three rules have applied to a single prefix, rendering it almost completely opaque. One can easily imagine that in cases of multiple prefixes, even more rules could apply to a single word. Other First Nations languages, especially those in the Athapaskan family, which

have very high numbers of prefixes, can undergo an even greater number of rule applications to a single word form, producing even more variation and opacity. Implementing an FSM for Cayuga should thus demonstrate the general applicability of the FSTs to other polysynthetic languages.

4.4 Implications of the Data

The morphology and morpho-phonology of Cayuga nouns is quite complex, as has been shown from the brief overview contained in this chapter. Not only are there four types of nouns, but there are two important issues with the basic and inalienable nouns: firstly the number of possessive prefixes each basic or inalienable noun stem may take, and secondly the number allomorphs for each prefix.

This second problem can be addressed in two ways. It is possible to assume that speakers have learned a small set of abstract prefix morphemes and an extensive set of rules to derive the actual spoken forms of those prefixes. It is also possible, however, to instead assume a more limited set of rules and a much larger set of prefix morphemes listed in the lexicon that more concretely resemble the spoken form.

The implementation of both an abstract (rule-based) approach to the segmentation of Cayuga nouns as well as a more concrete (lexical) approach will be discussed in greater detail in 6.1.1.2. The following chapter will discuss in greater detail the implications of Cayuga for the development of dictionaries and the development in the finite-state framework.

Chapter 5

Issues in Implementation

The structure of Cayuga nouns raises two types of problems. The first and most important problem is that of dictionary access. Given the quantity of variation that occurs within prefixes and stems of Cayuga, it is difficult to learn to use a dictionary for Cayuga. Although this program is a direct implementation of the linguistic description it is also a computer program and the constraints of processing power and hard drive space entail that some potential problems ought to be avoided : **non-determinism**, and **long-distance dependancies**. These two computational problems will be discussed following a discussion of the linguistic problem of dictionary access.

5.1 Dictionary Access

There have long been problems regarding dictionary access for speakers of obligatorily prefixing languages, especially polysynthetic languages such as Iroquoian and Athapaskan languages. Because these languages prefix obligatorily and undergo a high degree of morphophonological variation at prefix boundaries it can be difficult for speakers to properly segment morphemes in order to find them in a standard dictionary. Example (1) presents a sample of the difficulties that might arise.

(1) a. UR: hadi+ihn+aˀgeh
SR: hadi-hn-aˀgeh
(on) their (male) skin

b. UR: hadi+ahǫhd+aˀgeh
SR: hęn-ahǫhd-aˀgeh
(on) their (male) ears

In (1-a) the underlying form of the prefix (*hadi+*) is clearly related to the surface pronunciation (*hadi+*). However, in (1-b) the surface and underlying forms for the same prefix are

different (*hadi+* vs. *hǫn+*).

The problem of dictionary access is not a new one. To date there are essentially two types of paper dictionaries for highly prefixing languages such as Cayuga: "base dictionaries" and "root dictionaries" (Foster *et al.*, 1991).

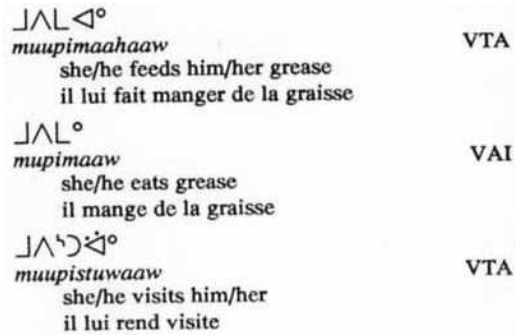
A "base dictionary" is a dictionary in which forms are organized by meaning. A base is defined as stem or combination of stem plus affix(es) that has an at least partly lexicalised meaning. Morphologically related forms are related by cross-references. This is in contrast to the stem dictionaries in which forms are related by morphology. Such dictionaries are similar to traditional Indo-European dictionaries, but include rules for deriving related forms. Unlike European dictionaries, however, citation forms are not necessarily whole words.

The drawback to both of these approaches is that they require users to look up forms which lack the obligatory prefixes. This requires that the user be able to perform some amount of segmentation, in order to use the dictionary.

For some polysynthetic languages, this is not an issue. For example, because the morphophonology of Algonquian prefixes is generally straightforward, it is not too difficult to organize a traditional dictionary that is easily usable. Often, stems can just be listed without affixes; prefixation is not obligatory in some forms. Another alternative, as in (MacKenzie and Jancewicz, 1994), is to choose a specific prefix and list all verbs with that prefix.

In example (2) we see how the initial 'mu' was used to standardise the entries in the Innu dictionary.

(2) An example of a stem dictionary entry (MacKenzie and Jancewicz, 1994:p. 39):



Slightly more complex languages, such as Oneida or Cayuga, however, pose a greater problem to the user. There are often a large number of forms whose semantic meaning is unpredictable directly from their morphology and these need to be listed separately. Michelson and Doxtator (2002), to alleviate this problem have used the base approach to dictionary construction, to group semantically related items together.[1]

In example (3) we see how a single base is listed along with several varieties of forms related to that base as well as its component parts.

(3) An example of a base dictionary entry of a verb (Michelson and Doxtator, 2002:p. 465):

-ka?tatye- v.m. have a lot of while going along. With t- cislocative: *tyakoka?táti?* she had a lot of things with her there. With n- partitive: *niswakka?táti?* I have so much with me as I'm going along. With -khw- food, and nis- partitive and repetitive: *nitsyuknikhwaka?táti?* we two have all this food along with us again. With -hle?n- bundle, and nis- partitive and repetitive: *nitsyakohle?naka?táti?* she has such a bundle along with her again. With -itsy- fish, and n- partitive: *nisʌtsyaka?táti?* what a lot of fish you have got with you, *nihotsyaka?táti?* he has such a lot of fish with him. With -yʌt- wood, cord of wood: *loyʌtaka?táti?* he has a lot of wood with him.

•*Né· s thikʌ́ nʌ ʌ́shlawe? loyʌtaka?táti? a?é· nihohlé·na? thikʌ́.* When he gets home he has a lot of wood with him, a great big bundle of it. (G1)

COMPOSED OF: **-ka?t(e)-** be or have a lot of, -tye- progressive.

NOTE: Many speakers have -ka?tati- before a final ?.

[1] In the finite-state approach the semantics of lexicalised forms must also be listed separately. This is straightforward to accomplish so there is no particular gain or loss with respect to this problem and the FS framework.

(4) An example of a base dictionary entry of a noun (Michelson and Doxtator, 2002:p. 465):

ka?nhehsatʌ́sha V > N silk, taffeta.
NOTE: A slightly different form,
ka?nheksatʌ́sha, is also attested. It
has been suggested that
ka?nhehsatʌ́sha refers to silk, while
ka?nheksatʌ́sha refers to taffeta.
Both forms include the ka- neuter
agent prefix, the noun base **-?nheks-/**
-?nhehs- ribbon, strap, and the verb
base **-tʌs-** be thick; but it's not clear
what the final -ha is.

Unfortunately, this approach does not alleviate a second and equally important problem: that of how to appropriately list forms in a logical, accessible manner such that users can easily find a written entry from a spoken form. When Michelson and Doxtator (2002) compiled their dictionary of Oneida, a hybrid approach was used with nouns listed as whole words (including obligatory pronominal prefixes such as 'ka' in exaple (4)) and verbs were listed without the obligatory prefixes such that morphologically related forms would be grouped together. Compare examples (3) and (4) to see the difference in noun and verb listings. In contrast the Cayuga dictionary of Froman *et al.* (2002) lists all forms without prefixes.

Neither of these approaches is superior to the other; both present considerable usage difficulties. If users wish to look up a word in a whole word dictionary they must first know the conventions used to prefix each type of word (e.g., basic nouns are prefixed with the appropriate 3zn prefix ('*ga*', '*o*' or '*a*') but inalienable nouns with the 1sP prefix ('*ak*', '*age*', or '*ag*'). Second, they must know the correct form of the prefix for the word that they wish to look up. In the case of a dictionary listing bare stems, users must know the form of the bare stem to be looked up: they must be able to properly segment the word. So either the users must know enough about the grammatical parts of speech to know how to determine the proper prefix for whole word forms or they must have a thorough knowledge of how to segment their language.

Unfortunately, speakers of a language often do not have conscious access to the underlying forms of morphemes. Mithun (1979) notes that in Mohawk, speakers cannot distinguish between epenthetic stem-initial 'e' and underlying stem-initial 'e'. If speakers cannot determine an epenthetic stem-initial sound, they may misconstrue the apparent stem they heard and be unable to look it up. Alternately, a stem-initial sound could have been deleted (de-

pending on the form that the user heard), and again they would have difficulty locating the form.

Often dictionaries are required to contain rather detailed explanations of the morphology of the language in order for users to be able to access them properly and even then they are still quite complex. For example, see (Froman *et al.*, 2002:pp. xvi-xxxix, §8.2-8.3). This is a major barrier to the usability of dictionaries for these languages. The problems are only compounded when dealing with Athapaskan languages which tend have an even greater number of prefixes (Young and Morgan, 1987).

A finite-state application with a simple interface consisting of a web-based form would make dictionaries easily accessible to all speakers. One would need only type a word into the form to access its morphology and semantics and not have to learn the morpho-phonology of the language.[2]

5.2 Computational Problems

Some problems have, in the past, been considered non-computable in the finite-state. Recently work-arounds have been developed that solve some of these problems. Unfortunately, such solutions introduce complexity (§2.2).

When constructing an FST for Cayuga nominals, it is important to be careful to avoid using too many complexity-heavy operations as they can introduce significant complexity into the machine. Non-determinism and long-distance dependancies are two such problems that will need to be avoided. These two operations when coupled with composition, which already introduces complexity, can lead to a very high degree of complexity very quickly.

5.2.1 Non-determinism

Non-determinism (Beesley and Karttunen, 2003:§9.2.3) refers to an ambiguity within the finite-state machine. I distinguish two types of ambiguity, external ambiguity and internal ambiguity. A case of external ambiguity might be a surface form which could be derived

[2]A guesser could even be incorporated to make spelling easier for users who might have difficulties in that area.

from either of two or more underlying forms. This ambiguity is external because it is visible to the programmer and the user. Internal ambiguity is a case where at a given point the FSA can follow either one of two different paths, which are disambiguated later on.

Both types are undesirable, but external ambiguity is necessary and internal ambiguity can be difficult to locate or prevent. Examples (5)-(6) illustrate an externally ambiguous FST:

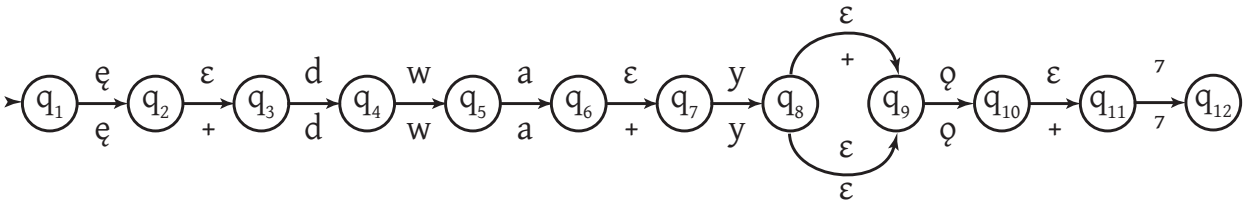
(5) a. ɋdwayǫ˥
we all will give it to someone OR *we all will arrive*

b. ɋ+ dwa +y +ǫ +˥[3]
future 1idpA give epenthetic [y] punctual
we all (inclusive) will give it to someone

c. ɋ+ dwa +yǫ +˥
future 1idpA arrive punctual
we all (inclusive) will arrive

The FST that would recognize the forms in (5) is described in example (6).

(6)



The symbol 'ε' is used to denote an empty string in the FS framework, and is conceptually similar to the linguistic formalism of using '$\emptyset$'. It is generally used when one needs to output on one side more symbols than have been inputted. However, as we will see, the ε-transition can lead to non-determinism.

The FST in (6) is fine so long as we are only trying to recognize an underlying form or remove the morpheme boundaries from a segmented form. Unfortunately, if we try to go the

[3]The literature generally reports that the form meaning *we all (incl) will give it to someone* is [ɋgyǫ˥] from /ɋ-dwa-ǫ-˥/→/ɋ-dw-ǫ-˥/→/ɋ-dy-ǫ-˥/→/ɋ-gy-ǫ-˥/. However, speakers currently prefer the more transparent form in (5-b) (Froman *et al.*, 2002:Appendix J).

other direction, we reach a point of non-determinism at q_8. At that state, when converting from SR to UR, the machine has no way of determining which is the correct path to take.

In this case, we would want the transducer to output both forms, but this would require that the machine explore all possible paths and output any that could be correct. In some cases, the FST may run quite far down a path before hitting a roadblock of some sort and aborting that path. If there are dozens of such forks the machine may have to explore tens, hundreds or thousands of paths to figure out where it is going. Obviously, such a program would run significantly more slowly than a program that only has one valid transition out of each state.

These examples show how certain operations can introduce non-determinism into an FST. In an FST representing an entire language, non-determinism is virtually unavoidable, but one must be careful to avoid needless non-determinism since it will significantly slow down the processing (Nederhof, 1996; Beesley and Karttunen, 2003).

5.2.2 Long-distance Dependancies

A long-distance dependancy is when the occurrence of a morpheme or morpho-phonological variant is governed by a factor that is not immediately adjacent to it. An FSM has no memory; it can only know the immediately preceding segment and the current segment. It is often costly, therefore, to model long-distance dependancies.

In example (7) we have a case of a long-distance dependancy where the suffix and prefix are co-dependant. The prefix *(e)kni-* requires the suffix *-ageh˺* (compare (7-a) and (7-d)) and *o-* requires the suffix *-a˺* (compare (7-b) and (7-c)).

(7) a. (e)kni + gahgwaos + a˺geh
1IncDu + eyebrow + on
(on) our (two people including listener and speaker) eyebrow(s)

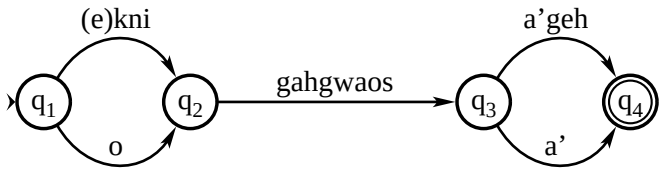
b. o + gahgwaos + a˺
3NP + eyebrow + NSF
an eyebrow (detached)

c. *o + gahgwaos + a˺geh
3NP + eyebrow + on

d. *(e)kni + gahgwaos + a˺
1IncDu + eyebrow + NSF

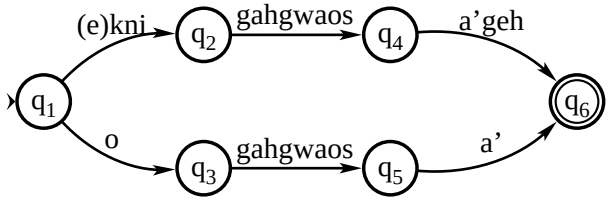
Example (8) shows the most parsimonious way to model the production of the forms in (7). However, the FSA in (8) overgenerates, producing not only (7-a) and (7-b) but also the invalid forms (7-c) and (7-d):

(8)



To model these forms correctly one would actually need to include an arc for the root form twice. This implementation, however, is relatively inefficient, in this case, doubling the number of arcs per root form, thereby increasing storage space required:

(9)



The program I will be using to construct the FST has a built-in methodology for programming long-distance dependancies in the lexicon called "flag diacritics" (Beesley and Karttunen, 2003:339-373). These are used at run-time to constrain relationships. While the size of the machine is unaffected, actual run-time processing is slower.

Several types of flag diacritics exist for a variety of purposes, but for the construction of my machine the type that is of interest are the "u-type" flag diacritics. These flags can be placed anywhere within the lexicon, as needed. If a string is generated with mismatched flags, that string is discounted as invalid at run-time (10-c). This means that prefixes that

can only take certain suffixes can be marked with the same flags whereas the other prefixes and suffixes are marked with different flags.

The FST program would recognize the words in examples (10-a) and (10-b) as valid because the two flags in the words are identical. In (10-c), however, there is an 'UNPOSS' and a 'POSS' flag, which do not match up, so the word would be marked as invalid.

(10) a. @U.INALIEN.POSS@ g+ nǫˀa: @U.INALIEN.POSS@ +ageh˥
Possessive Flag 1sA head Possessive Flag LOC
'*On my head*'

b. @U.INALIEN.UNPOSS@ o+ nǫˀa: @U.INALIEN.UNPOSS@ +a˥
Unpossessed Flag 3znP head Unpossessed Flag LOC
'*My detached head*'

c. *@U.INALIEN.POSS@ g+ nǫˀa: @U.INALIEN.UNPOSS@ +a˥
Possessive Flag 3znP head Unpossessed Flag LOC

It is possible for the FST programming tool to automatically remove flags and re-construct an equivalent FST which does not require flags. This means that I can automatically convert from an FST like that in example (8) that uses flags to the equivalent machine as in example (9) that does not use flags. The advantage to such a conversion is generally an increase in processing speed, but the one drawback is that this can drastically increase the size of the transducer.

Flags have the format X.FEATURE.VALUE where X is the flag type, feature is a feature name and value is the value of the feature. Other flag diacritics are P-type flags that allow the user to set a certain value to a feature (e.g.: P.INALIEN.POSS flag following a U.INALIEN.UNPOSS would act as if the user had initially set a U.INALIEN.POSS flag); N-type flags that set the value of a flag to the complement of a given value (e.g.: N.INALIEN.POSS will match with any INALIEN.* where * is not equal to POSS); R-type flags that check if a feature has a certain value (e.g.: if P.INALIEN.POSS is set then R.INALIEN.POSS will succeed but R.INALIEN.* will fail if * is not equal to POSS); D-type flags that only succeed if the feature is not yet set; and C-type flags that reset a feature to a neutral value.(Beesley and Karttunen, 2003:pp. 353-356)

Please note that flag diacritics do not confer any greater-than-finite-state power to an FSM. They are merely for convenience of composing lexicons. More in-depth discussion can

be found in (Beesley and Karttunen, 2003:pp. 339-341)

5.3 Summary

The problem of dictionary access can be greatly alleviated by having a computer program that can do all the morpho-phonological heavy lifting. A finite-state solution seems almost ideal, except for some of the potential computational problems mentioned above. These problems however, can be satisfactorily addressed, as the following two chapters explain.

Chapter 6

Methodology

This chapter will give an overview of the practical and technical requirements and specifications for the design of the FST, as well as a discussion of the tools that I used to implement the machine.

6.1 Requirements

There were two requirements that a successful morphological analyser/generator for Cayuga needs to fulfill. It needs to (a) produce useful dictionary output; (b) be easy to use. For the purposes of producing useful output it will also be valuable to (c) test the morpho-phonological rules posited for Cayuga nominals. Finally there are two other empirical requirements for the consideration of future work: (d) determine the usefulness of the finite-state framework for such polysynthetic languages; and also (e) serve as a way of comparing machines of varying degrees of linguistic elegance for their usefulness and complexity.

6.1.1 Ideal Dictionary Output

Given that speakers do not have conscious access to their underlying linguistic knowledge regarding morpho-phonology (Mithun, 1979), it is unclear what sort of information would be the most useful output for a speaker attempting to access a dictionary: would a set of underlying prefixes not necessarily obviously related to the actual pronunciation be useful; or would the output of a prefix not obviously related to other prefixes with a similar meaning be useful? To return to the example (1) (originally from §5.1 and repeated below): is it more useful for the speaker entering the words in (1) to be told that (a) there is a single underlying prefix (*hadi+*) whose meaning is **3mdpA** and can be pronounced in several ways including both (*hadi*) and (*hę́n*); or (b) there are several prefixes which all mean **3mdpA** and are produced in several different situations? Speakers tend to prefer the second alternative

(Dyck, 2006).

(1) a. UR: hadi+ihn+aˀgeh
SR: hadi-hn-aˀgeh
"(on) their (male) skin"
b. UR: hadi+ahǫhd+aˀgeh
SR: hęn-ahǫhd-aˀgeh
"(on) their (male) ears"

As has been described above (§4.2), the majority of nouns in Cayuga take some form of obligatory prefix, often with a stem change, leading to difficulties in segmentation and stem identification. This problem is discussed more fully below.

6.1.1.1 Abstractness vs. Concreteness

One problem that I will need to resolve is that it is not clear whether all prefix variants should be listed in the lexicon (concrete), or whether they should instead each be derived by rules from a smaller set of prefixes (abstract). Firstly, some prefix combinations (as in example (2) are lexicalised, or non-transparent.

(2) a. gaǫ+
3fidpA+a-root
b. gae+
3fidpA+i-root OR C-root
c. ga:g+
3fidpA+other root

The relationship between these alternants is opaque. This means that they should be listed separately in the lexicon to represent the speaker's intuition, especially since they are not easily derivable from morpho-phonological rules.

Other prefixes, such as those in example (3), however, are derivable, but are still not fully accessible to speakers in their underived forms. In this case, it still might be more useful to list these prefixes in the lexicon, again to represent speaker intuition, despite the

fact that they can be easily derived.

(3) a. UR: -(y)ǫkni- N
SR: -(y)ǫkni- N
1DuP [c/i]-initial root

b. UR: -(y)ǫkni- -a
SR: -(y)ǫgy- -a
1DuP [a]-initial root

It is possible to represent underlying /-(y)ǫkni-/ as two separate affixes in the lexicon, one, the SR (3-a), that precedes most nouns and another that precedes noun stems beginning with [a], the SR in(3-b). This is the concrete approach. In contrast it is possible to list only /-(y)ǫkni-/, the UR in both (3-a) and (3-b), in the lexicon, using a rule (4) to derive [-(y)ǫgy-]. This is the abstract approach.

(4) kni+a → gya

Linguists such as Froman *et al.* (2002); MacKenzie and Jancewicz (1994); Michelson and Doxtator (2002); Chafe (1967); Young and Morgan (1987) and Foster (1986) tend to represent the most abstract form when constructing dictionaries so that (3) (a) and (b) would be listed as a single underlying prefix '*(y)ǫkni-*'.

For a user it would likely be more useful to have a more concrete representation of affixes that are easily relatable to the actual pronunciation. It may however become clear once the project is underway that the specification of all affixes and base forms is not the most efficient method for the structuring of the project.

6.1.1.2 Computation of Abstractness vs. Concreteness

Considering that there are two possible linguistic solutions (listing allomorphs in the lexicon or deriving them from rules), it is also important to determine if both these approaches are equally computationally viable. If one method results in a machine that is too large or slow, it does not particularly matter if it is more useful to users since it will be unimplementable.

Adding morpho-phonological rule transducers will require additional compositions of FSTs which can very quickly lead to significant increases in the size of the final automaton and its running time. It might well be reasonable, therefore, to implement allomorphy in the lexicon rather than in the rules module.

I therefore decided to design my program in two separate versions. One version assumes fairly abstract prefix morphemes while the other assumes more concrete lexicalised prefix allomorphs as the UR.

In the abstract approach, the forms that are output to users are more linguistically and descriptively elegant, but are less useful to the users, who would require more specialized knowledge of the morpho-phonemics of the language to interpret the output (§5.1). The concrete approach, however, attempts to design with the end user in mind such that it will generate a set of prefixes that the user will be able to consciously relate to the actual pronounced surface forms.

The project should provide a means for comparison in terms of computational elegance and computational complexity of the two approaches described above. I compared the two machines to each other rather than to a separate set of arbitrary criteria. This was done because I am only attempting to determine which machine is most efficient. As mentioned earlier (§6.1.3) it is beyond the scope of this project to determine the applicability of the framework to these languages for a full implementation, so for the moment a comparative method will have to suffice. The criteria for computational elegance and complexity are listed below.

Computational Elegance:

- Time/difficulty of creation of machine
- Time/difficulty of modification of machine

Computational complexity:

- Relative numbers of states
- Relative numbers of transitions

- Relative run-time
- Relative machine size in kB

Computational elegance will be viewed loosely in terms of the difficulty of creation and modification of the design. If it is difficult or tedious to update the program or requires significantly more time to effect changes, one version will be deemed less computationally elegant than the other. In terms of complexity, this will be judged by comparing the numbers of states and transitions that each machine contains, as well as average running times and final machine sizes in kB.

6.1.2 Thorough Testing of Morpho-phonological Rules

The finite-state framework allows for a high speed testing of the rule formalisms relating surface and underlying forms that have been already posited for Cayuga. The machine is designed to relate the surface and underlying forms of the words; and rule formalisms are the most straightforward way to generalize that relationship so that the program must naturally represent a testing environment for the rule formalisms.

It is often difficult to test the ordering of large numbers of rules against large numbers of word-forms to ensure that there are no incorrect forms generated or inconsistencies in the output. Also, applying large numbers of rules by hand to large data sets leaves a margin for human error. Having a computerized version of the rule formalism, however, solves this problem.

It becomes possible to easily and quickly test very large sets of data against the machine to ensure that the output is correct for all cases. One can design a set of input data knowing what the output should be and automatically compare that to the actual output of the machine (§7.2.1).

6.1.3 Applicability of FS Framework to Morphologically Complex Languages

The project must demonstrate that the finite-state framework is capable of handling the complex morphological and morpho-phonological processes of First Nations languages efficiently. This means that the program must be able to quickly process words containing

long-distance dependancies (§5.2.2) and prefixes which may be opaque, semi-opaque or ambiguous (§4.2.2-4.2.3).

Despite the fact that nouns display the majority of interesting morphological and morpho-phonological properties of Cayuga, such as large degrees of allomorphy and long-distance dependancies, there are a few caveats to keep in mind. Verbs have more prefixes and more prefix combinations than the nouns. Furthermore, verbs also have a much higher incidence of long-distance dependancies. It is possible therefore, that despite the fact that the machine described in later chapters is computationally adequate, that the expansion of the machine may pose efficiency problems. In particular, a large number of additional long-distance dependancies could pose a problem. My machine models only a single long-distance dependancy whereas a model that handles verbs would require several long-distance dependancies (where the greatest complexity is likely to be introduced).

6.2 Specifications

I will here briefly overview the specific requirements of the abstract and concrete FSTs. They each follow essentially the same design, so differences are only mentioned where applicable. A fully detailed description of the programs can be found in §7.1. In light of the motivations described above the programs must conform to several technical requirements.

1. In order to satisfy the needs of dictionary users, the program must **generate and segment all the basic noun types** described in §4.2:

    - **Unpossessed Basic Nouns** - A 3rd person neuter prefix followed by a basic noun root and a noun stem forming suffix.
    - **Possessed Basic Nouns** - A patient prefix followed by a basic noun root and a noun stem forming suffix.
    - **Possessed Inalienable Nouns** - A agent prefix followed by an inalienable noun root and an external locative suffix.
    - **Unpossessed Inalienable Nouns** - A single 3rd person neuter singular patient /(y)o-/ prefix followed by an inalienable noun root and a noun stem forming suffix.

- **De-verbal Nouns** -A third person neuter prefix followed by a verb root and deverbal suffix and noun stem former.
- **Defective Nouns** - A single defective noun root.

2. The program must **generate and segment only these word forms**. In order to be appropriate for a dictionary tool and to properly represent the language, the machine must not give false positives (i.e., no incorrect forms should be judged as correct).

3. The program must **provide basic semantic output** for all word form morphemes. To satisfy the needs of a dictionary application, semantic output must be generated for the user (My program only generates a basic semantic representation, however, to demonstrate that it is possible).

4. The program must encode an **ordered rule formalism** that is adequate for testing the set of morpho-phonological rules and their ordering as currently posited for Cayuga.

5. The program must be able to have an **easy to use interface** created such that users need only type a word to receive its morphology and semantics.

6. The program must satisfy the requirement of testing the adequacy of the finite-state framework for applications in First Nations languages. The final machine must present data on the comparative levels of efficiency of the two approaches.

6.3 Tools and Data Structure

To implement the program I used XFST (Beesley and Karttunen, 2003), a finite-state development environment developed by Xerox for use by computational linguists. XFST makes the creation of finite-state machines somewhat more intuitive. It is a tool that allows a computational linguist with only modest knowledge of programming and computational theory to create and modify finite state machines.

Without a tool like XFST one might need to know how to create a 'regular expression' that would look something like:

(5) /(h|c)at/

A regular expression is a formalism for describing a language, in this case the language consisting of the strings 'hat' and 'cat' (Beesley and Karttunen, 2003; Nederhof, 1996). However if one were attempting to encode an entire language, the resulting regular expression could easily become very long.

(6) /((h|r|c)at)|((c|b)ar)(e|ε)/
ε denotes the 'empty' character

Example (6) describes a very small language containing: 'hat', 'rat', 'cat', 'car', 'bar', 'hate', 'rate', 'care', 'bare' and *'cate' (The 'cate' example shows how it can be quite easy to make a mistake with regular expressions that need to match many forms). In contrast, using XFST, one can simply create a lexicon file which details (a) the morphemes that exist in the language and (b) how they can join together. XFST can then turn that result into an FST that correctly encodes a regular expression.[1]

Another positive aspect of XFST is that it is constantly compiling and optimizing the machine as it is being created by the user. This means that it is significantly more efficient and simple than writing a finite-state machine and then optimizing it afterwards; the machine will run more quickly and take up less space overall. Additionally, users can check their work as it progresses, with no need to wait until programming is complete.

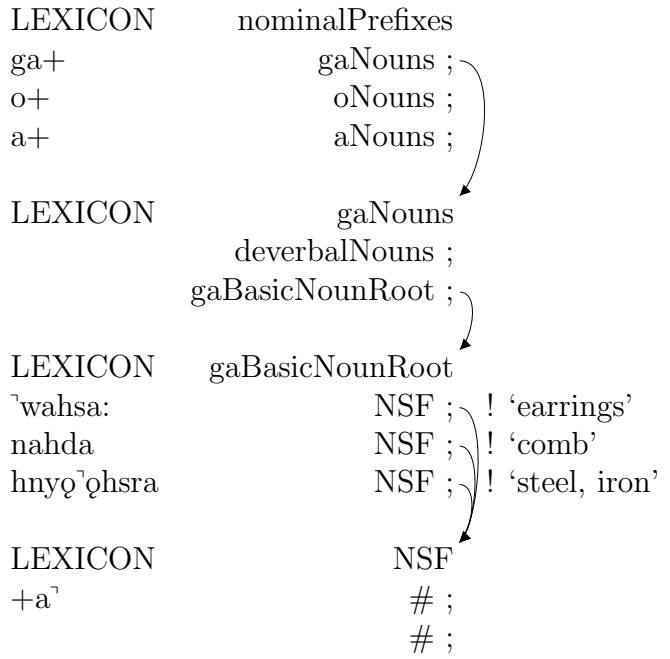
Finally, XFST also includes several specialized tools for creating linguistic descriptions. Two of these tools are described in the following two sections.

6.3.1 Lexc

The **Lexc** language is an XFST formalism developed specifically for designing lexicons. Lexc allows the user to design several "sub-lexicons" (e.g., a prefix lexicon and a stem lexicon) and define how the lexicons each join to each other in terms of what are called "continuation classes". (Beesley and Karttunen, 2003:§4)

[1]Regular expressions and FSMs are generally equivalent in that a regular expression can be represented as an FSM and vice-versa, but there is often a loss of efficiency in translation (Nederhof, 1996). There are also other formalisms (such as finite-state grammars) for specifying FSM, but generally regular expressions are the most popular.

Table 6.1: Sample Cayuga lexc Lexicon.



Each sub-lexicon specifies a morpheme in the left column and the "continuation" class (if any) in the right-hand column. The continuation class is simply the name of another sub-lexicon whose morphemes are allowed to follow after the given morpheme in this sub-lexicon.

In the table 6.1, four sample lexicons are given in lexc format. The first lexicon specifies all the nominal prefixes (in the left column) as well as the name of the lexicon that specifies what can follow each prefix (on the right). So, the prefix *ga* can be followed by anything found in the lexicon named *gaNouns* which in turn specifies two further lexicons that can follow it. This allows the linguist to break down the forms into as many parts as are necessary to describe the URs of all words in the language.

Example (7) gives a sample possible output from the FST define in table 6.1.

(7) nominalPrefixes+ gaNouns+ gaBasicNounRoot +NSF
ga+ ø+ ˀwahsa: +aˀ#
ga+ˀwahsa:+aˀ *earrings*

Because *ga+* was the output from the *'nominalPrefixes'* sub-lexicon, the next segment has to come from the *'gaNouns'* sub-lexicon. The output of either option in that sub-lexicon is ø followed by either of the applicable sub-lexicons (*'deverbalNouns'* or *'gaBasicNounRoot'*). After an item from *'gaBasicNounRoot'*, an item from the sub-lexicon *'NSF'* must follow. This sub-lexicon specifies # as the continuation class, which signals to the compiler that it is a valid end state (§3.2).[2]

The lexicon description in *lexc* format is converted into an FST by the *lexc* compiler. This FST can then be composed with other FSTs that represent other modules of the final program (§3.6), including a rules module or a semantics module. Such a methodology facilitates the modification of either module independantly from the others.

6.3.2 Rule-like Notation

Just as XFST provides the lexc formalism for defining lexicons, it also provides a number of built-in rule-formalism shortcuts that make the writing of morpho-phonological rules much simpler for linguists. In the same way that a *lexc* lexicon is compiled into an FST, so too are these rules compiled into an FST. The two resulting modules can then be composed together.

Rules are defined using a notation very similar to standard phonological rule notation. For example a vowel coalescence rule written as:

(8) / a + i / → [+ę]

would instead be written as:

(9) define aię [a %+ i − > %+ ę];[3]

The rules are composed together to form a module of rules. The order in which the rules are composed is identical to the order in which they will apply, meaning that the rule ordering

[2]When using lexc, one need not have a word-initial word boundary marker if one is only working with single words. The transducer will report an error if it encounters a multi-word token.

[3]In the Lexc formalism, the + symbol is a character with special properties, and to encode the + as a normal symbol it must be prefixed by a % symbol.

in the finite-state machine is logically identical to the posited morpho-phonological rule orderings.

6.4 Summary

Having described the basic types of structures from which my machines were formed, I will now go on in the following chapter to specifically describe the structure of the machines and what findings were determined during their construction.

Chapter 7

Results

Here I will discuss the structure of the final program, giving a description of each module and of the construction of both the concrete and the abstract machines. Following that, I will analyse the ability of the machines to meet the requirements as set out in §6.1 and compare their performances.

7.1 Final Program Components

The final machine was composed of several parts; firstly, the morphological analyser, secondly the semantics module and thirdly, the user interface. The morphological analyser itself was constructed of several modules, which are explained below. An explanation of the semantics module and the interface follows the discussion of the morphological analyser

Since the finite-state framework is inherently modular (§3.6), I developed my programs each in three separate modules. I developed a lexicon module, a rules module and a basic semantics module. The semantics module was produced only to make a more satisfying final product and demonstrate that a semantic component could easily be added. It does not lie within the primary focus of the project to output more than a basic semantic gloss.

7.1.1 Lexicon Module

As explained in §6.3.1 the lexicon module is constructed by the concatenation of multiple sub-lexicons using **lexc**. These lexicons concatenate noun morphemes together, but do not perform any rule-based operations.

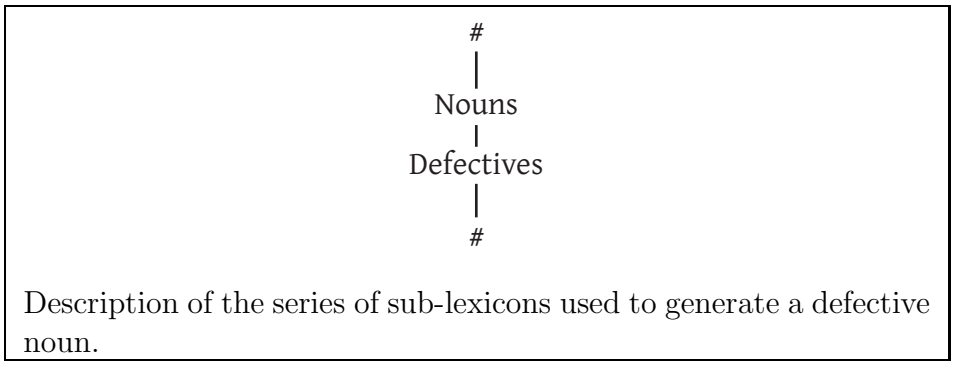
There are four separate sub-lexicons for 'defectives', 'deverbals', 'basics' and, 'inalienables' which in turn point to their own set of specialised sub-lexicons. The sets of sub-lexicons for generating each noun type in the abstract lexicon will be discussed below.

I will be only discussing the specifics of the abstract approach as it is much simpler in terms of sub-lexicons than the concrete approach. The concrete approach is essentially similar, but with more sub-lexicons. This aspect of the concrete approach will be discussed in more detail later.

7.1.1.1 Defective Nouns

The defective nouns are the most straightforward and are illustrated in fig. 7.1. The main over-arching sub-lexicon "nouns" points to a sub-lexicon named "defectives" that in turn merely lists all defective nouns. Since there is no internal morphology, there is no need for further sub-lexicons; so the "defectives" sub-lexicon is merely followed by a word-boundary.[1]

Figure 7.1: Abstract Defective Noun Sub-lexicon Structure



Description of the series of sub-lexicons used to generate a defective noun.

A sample defective noun as generated by the machine by joining the constituents of the appropriate sub-lexicons is shown below in example (1):

(1) Nouns+ Defectives
ø+ sgwa:gwaʠdʠˀ#
sgwa:gwaʠdʠ ˀ *toad*

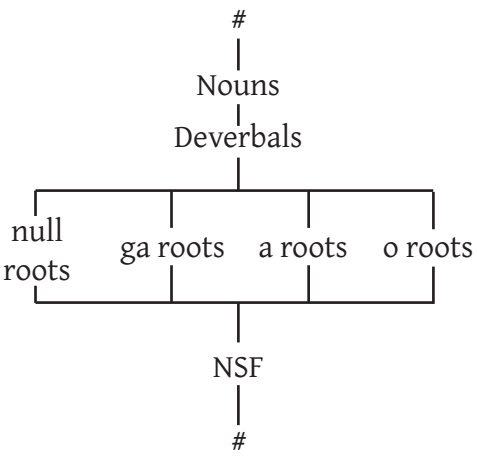
7.1.1.2 De-verbal Nouns

De-verbal nouns are slightly more complex than the defective nouns. As described in figure 7.2, the sub-lexicon named "deverbals" contains a list of prefixes that can attach to deverbal

[1]Word boundary symbols (#) are used to represent the start and end of the machine but do not appear in the input or output.

noun roots such as *ga+, a+, o+, Ø*.

Figure 7.2: Abstract De-verbal Noun Sub-lexicon Structure



Description of the series of sub-lexicons used to generate a defective noun.

Each prefix points to the appropriate sub-lexicon containing roots that may attach to it; so for example, the *ga+* entry in the "deverbals" sub-lexicon would point to a further sub-lexicon named "ga roots" which contains a list of all roots which may take *ga+* as a prefix. (These nouns roots were divided in the lexicon because the prefixes that they take are fully lexicalised and do not depend on any morpho-phonological factors).

Example (2) gives an example of a deverbal noun as constructed by the FST from the constituents of each sub-lexicon.

(2) Nouns+ Deverbals+ a roots +NSF
ø+ a+ atshoˀkdǫhsr +aˀ#
a+atshoˀkdǫhsr+aˀ *hoe*

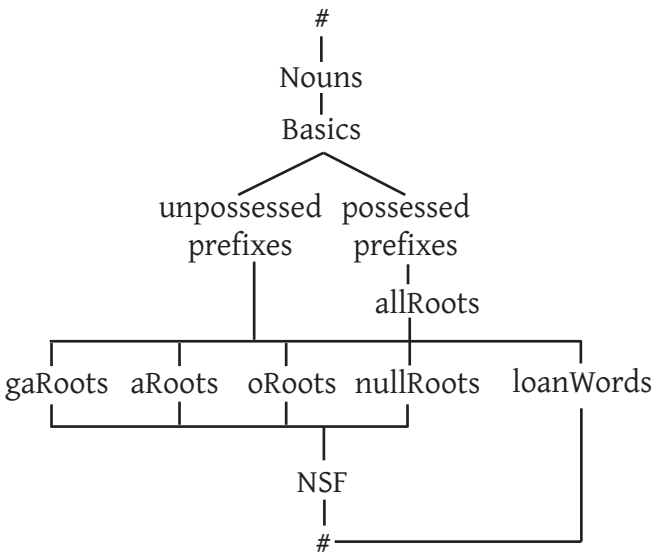
7.1.1.3 Basic Nouns

The structure of the sub-lexicons to generate the basic nouns is significantly more complex than that for generating the deverbals. There are two primary branches in this series of sub-lexicons. The first branch is a sub-lexicon containing the unpossessed prefixes and their appropriate sub-lexicons, the second branch contains the possessed prefixes and their

appropriate sub-lexicon.

The sub-lexicon for unpossessed basic nouns ("unpossessed prefixes") acts identically to the de-verbal lexicon described above. For this reason as in the deverbal lexicon, the basic noun roots have been sub-divided according to the unpossessed 3rd person neuter prefixes that they can take. The prefixes in the "unpossessed prefixes" sub-lexicon each point to the appropriate sub-lexicons of roots that take either *ga-* as a prefix ("gaRoots"), *a-* as a prefix ("aRoots"), *o-* as a prefix ("oRoots"), or ø- ("nullRoots").[2]

Figure 7.3: Abstract Basic Noun Sub-lexicon Structure



Description of the series of sub-lexicons used to generate a defective noun.

The other main branch of the basic noun sub-lexicons is the "possessed prefixes" series. This sub-lexicon contains all the possessed prefixes that may attach to a basic noun root and each prefix then points to a sub-lexicon named "allRoots". "allRoots" is empty except for a reference to each of the other sub-lexicons. Instead of attaching specific prefixes to specific basic noun roots, it allows all possessive prefixes to attach to all basic noun roots.

As can be seen from table 7.1, the "allRoots" sub-lexicon contains no morphological

[2]As described in §4.2.2 some basic noun roots have a choice of two prefixes. These roots are listed in sub-lexicons that are linked up with both appropriate prefixes. These (very small) sub-lexicons have been omitted from the diagram for the sake of simplicity.

Table 7.1: Basic Noun "allRoots" Sub-lexicon

LEXICON	allRoots
Ø	gaRoots ;
Ø	oRoots ;
Ø	aRoots ;
Ø	nullRoots ;

data[3], merely references to the other sub-lexicons. This enables me to reference the sub-lexicon "allRoots" once rather than to reference all the other sub-lexicon for each prefix. If I did not have this structure, each prefix would have to be listed once for each sub-lexicon to which it could attach, rather than being listed as merely attaching to the "allRoots" sub-lexicon.

Two examples of these structures can be found below in example (3):

(3) a. Nouns+ Basics+ Possessed Prefixes+ allRoots +nullRoots +NSF
ø+ ø+ (w)ag+ ø+ ɐ̓nhotr +a̓#
(w)ag+ɐ̓nhotr+a̓ *my ball*

b. Nouns+ Basics+ Unpossessed Prefixes+ oRoots +NSF
ø+ ø+ o+ ̓dɋdr +a̓#
o+̓dɋdr+a̓ *gristle*

Once a root from one of the 5 root sub-lexicons has been joined to its prefix one of two things is done. The "gaRoots", the "aRoots", the "oRoots" and "nullRoots" lexicons link to the "NSF" sub-lexicon, which adds the NSF suffix and terminates the word-formation process. Alternately, if a root from the "loanWords" sub-lexicon was added, no NSF suffix is attached to the word, it leads directly to the terminal # marker.

7.1.1.4 Inalienable Nouns

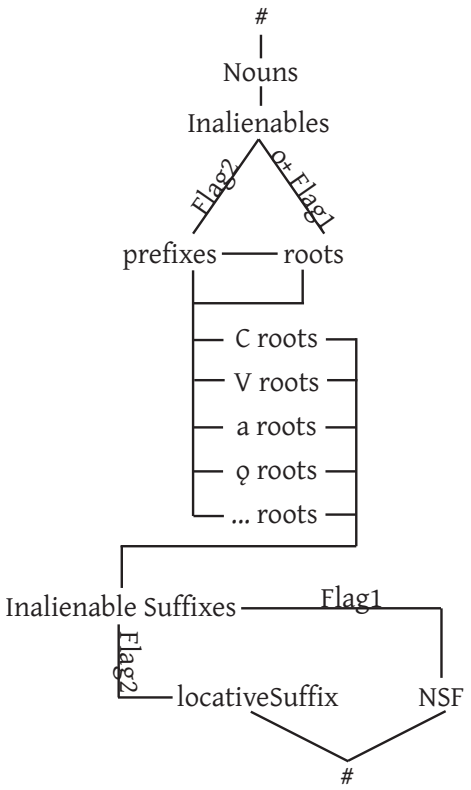
Inalienably possessed nouns have the most complex structure in the abstract lexicon. This is for two reasons: (a) they make use of several lexicalised prefixes that required a sub-division

[3]the Ø symbol is added for clarity, but is not part of the *lexc* formalism

of the lexicons (see §4.2.3, and §6.1.1.1 for more discussion on this topic) and (b) they contain long-distance dependancies that are handled with flag diacritics.

As with the basic nouns, there are two primary divisions in the nouns generated by this lexicon. As shown in 7.4 the first type are the "Flag1"[4] inalienables and the second type "Flag2" inalienables. "Flag1" and "Flag2" designate the attachment of flag diacritics, that eventually ensure that the agent prefixes (Flag2) co-occur with the locative suffix (*-aʾgeh*) while the **3znS** prefix (*(y)o-*, Flag1) co-occurs with the NSF suffix *-aʾ*. These flags (and the **3znS** prefix) are added in the initial "inalienables" sub-lexicon.

Figure 7.4: Abstract Inalienable Noun Sub-lexicon Structure



Description of the series of sub-lexicons used to generate an inalienable noun.

[4]Please note that "Flag1" used here is a short form of @U.INALIEN.POSS@ used in §5.2.2 and "Flag2" a short form of @U.INALIEN.UNPOSS@.

Paths With 'o+Flag1'

Forms with "o+Flag1" are followed by the sub-lexicon named "roots". This sub-lexicon is a directory that outputs nothing (ø) and points to all of the sub-lexicons which contain the various inalienable noun roots ("C roots", "V roots", "a roots", etc. There are 10 sub-lexicons, not all of which are listed; "... roots" is an abbreviation for these sub-lexicons).

A noun root from any of these sub-lexicons is added on the *(y)o+Flag1* construction previously output by the "inalieneables" sub-lexicon. The result is *(y)o+Flag1+any specific noun root*. This construction then points to the "Inalienable Suffixes" sub-lexicon.

The "Inalienable Suffixes" sub-lexicon either attaches "Flag1" and points to the "NSF" sub-lexicon (containing *+aˀ*) or it attaches nothing (ø) and points to the "locativeSuffix" sub-lexicon (containing *Flag2+aˀgeh*). The end result can generate both well-formed and ill-formed constructions as in example (4) below.

(4) a. Nouns+ Inalienables+ Roots+ a roots +Inalienable Suffixes +NSF
ø+ (y)o-Flag1+ ø+ ahǫhd +Flag1 +aˀ#
(y)o+Flag1+ahǫhd+Flag1+aˀ *ears* (detached)

b. *Nouns+ Inalienables+ Roots+ a roots +Inalienable Suffixes +locativeSuffix
ø+ (y)o-Flag1+ ø+ ahǫhd +Flag2 +agehˀ#
(y)o+Flag1+ahǫhd+Flag1+aˀ*Nonsensical form*

Paths With 'Flag2'

Forms with "Flag2" are followed by the "prefixes" sub-lexicon. This sub-lexicon lists two types of possessed agentive prefixes (§4.2.3, pg. 27): (a) the transparent prefixes whose allomorphs can be derived by rules and (b) the opaque prefixes whose allomorphs are not synchronically morpho-phonologically related.

The transparent prefixes point to the "roots" sub-lexicon and proceed identically from there, as do the 'o+Flag1' inalienables described above, resulting in a form of the type *Flag2+any specific noun root+Flag2+aˀgeh* or *Flag2+any specific noun root+Flag1+aˀ*, the latter being ill-formed.

The opaque prefixes (e.g., **3znsA** or **3fiA**), however, are derived differently. Each of

these prefixes points to a specific sub-lexicon containing a sub-set of noun roots. For example the **3fisA** prefix */(y)e-/* points to the "C roots" sub-lexicon while the **3fisA** prefix */(y)ǫ-/* points to the "a roots" sub-lexicon. The results then are *Flag2+(y)e+consonant-initial root* and *Flag2+(y)ǫ+[a]-initial root* as show in example (5) below.

The output from these root sub-lexicons then proceeds as previously described, giving both valid forms such as *Flag2+(y)e+consonant-initial root+Flag2+aˀgeh* and invalid forms such as *(y)o+Flag1+any specific noun root+Flag2+aˀgeh.* These invalid forms are then removed at run-time or during later composition as described in §5.2.2.

(5) a. Nouns+ Inalienables+ Prefixes+ a roots +Inalienable Suffixes +NSF
ø+ Flag2+ (y)ǫ+ ahǫhd +Flag2 +ageh$^{\text{ˀ}}$#
(y)o+Flag1+ahǫhd+Flag1+a$^{\text{ˀ}}$ *ears* (detached)

b. *Nouns+ Inalienables+ Prefixes+ a roots +Inalienable Suffixes +locativeSuffix
ø+ Flag2+ (y)ǫ+ ahǫhd +Flag1 +a$^{\text{ˀ}}$#
(y)o+Flag1+ahǫhd+Flag1+a$^{\text{ˀ}}$

7.1.1.5 Flag Diacritics

Figure 7.4 (p. 60) also illustrates how flag diacritics (§5.2.2) operate. Flag1 attaches with the prefix "*o+*" and the NSF suffix for the formation of basic noun style inalienables 4.2.3. Flag2 is inserted with all other agent prefixes and with the locative suffix. This allows the machine to generate illegal forms in which conflicting flags co-occur, as described earlier. These illegal forms as in example (6), are then filtered out at a later point.

(6) *o+ hsohgw +aˀgeh
3znP+ lip +LOC

The form in example (6) is not valid because the prefix 'o+' with an inlienable noun stem requires the suffix '*+aˀ*' and cannot take the suffix '*+ageh*ˀ". In order to avoid having to define the inalienable noun root sublexicon twice I decided to have a machine that overgenerates by producing even the invalid prefix and suffix combinations.

To constrain the output to only the valid word forms I used flag diacritics (Beesley and Karttunen, 2003:339). A "u-type" flag was used to mark the possessive or unpossessive

prefix and suffix. At runtime the machine then checks these flags to make sure that the prefix flag corresponds to the suffix flag. If they do not the machine returns an "invalid" response for the word. In the final version of the program these flags were removed and an equivalent FST was automatically generated by XFST (as described in §5.2.2).

7.1.1.6 Possible Modifications of Inalienables Nouns Sub-lexicons

When constructing the inalienable noun lexicon for the abstract approach I decided to create sublexicons to handle the alternations of the **3fiA**, **3znsA** and **3fidpA** prefixes for inalienable nouns. These prefixes are not morpho-phonologically related to each other. Being historically un-related they are traditionally listed as separate underlying prefixes in a dictionary. For this reason, I chose to list them as separate underlying prefixes and specify the noun roots to which they could attach in the lexicon, mimicking the structure used in the concrete approach.

The use of sub-lexicons makes the construction of the abstract semantics module somewhat difficult and inconsistent. The basic function of such a module is unimpaired, but it now has a concrete component: noun roots now must be compartmentalized into sub-lexicons based on their initial phones. As well the complexity of the lexicon is greatly increased and this complexity will hamper any future modifications to that lexicon.

Fortunately, the opaque prefix alternations are conditioned by their environment, so they are not strictly arbitrary. Despite being lexicalised, their alternation is phonologically governed. That is, each prefix only occurs before a regular sub-set of roots (e.g., roots beginning in [a] or roots beginning in a consonant). This means that it will be possible to rework the more "concrete" portion of the abstract approach, as described below.

There are two viable ways in which this series of sub-lexicons can be remodeled. The machine can be modified to either (a) over-generate with all abstract prefix allomorphs which can then be constrained using a rule, or to (b) replace the allomorphs with a single abstract prefix that can then be transformed into the appropriate surface form with a rule on the output side and into the appropriate underlying form with a rule on the input side.

The first approach, overgeneration, works by attaching all prefixes to all stems, regardless of whether or not this generates valid URs. Then, in the rules module, filter rules are

added that remove the invalid forms that were generated in the lexicon. In this approach both example (7-a) and example (7-b) would be generated, but a rule would filter out the incorrect form (7-b).

(7) a. UR: (y)ǫ+ ahsi˺d +ageh˺
Sem: 3fiA+ foot +LOC

b. *UR: (y)ę+ ahsi˺d +ageh˺
Sem: 3fiA+ foot +LOC

This approach is exponential in generation of incorrect forms and would require more rules for each valid surface form. This would mean a greater degree composition and subsequent minimisation. Compile-time would therefore be adversely affected.

The second, more abstract approach would replace all three prefixes with a single abstract prefix, possibly denoting the prefix's semantics as in examples (8-a) and (8-b).

(8) a. Lexical: 3fiA+ ahsi˺d +ageh˺
Surface: (y)ǫ+ ahsi˺d +ageh˺

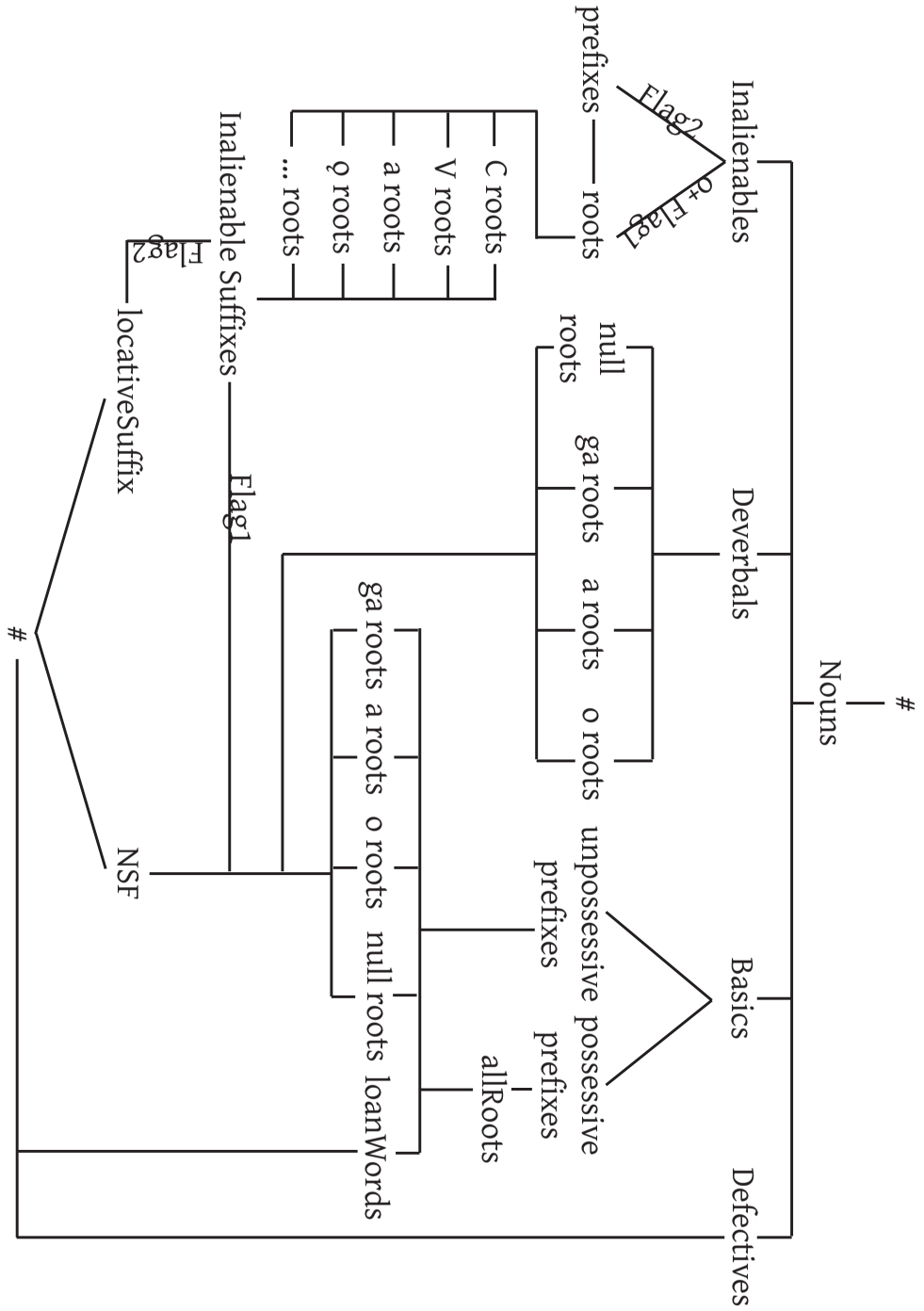
b. Lexical: 3fiA+ ǫts +ageh˺
Surface: (y)ag+ ǫts +ageh˺

Spell-out rules (such as in (9)) would then be created. These would change this abstract prefix, as in examples (8-a) and (8-b), into one of the appropriate surface forms.

(9) 3fiA→(y)ǫ/_+[a]
3fiA→(y)ag/_+[ǫ]

This approach is deterministic and therefore puts less of a burden on the machine at compile time as it does not generate a vast number of incorrect forms. It is therefore somewhat more preferable despite producing the same final result.

Figure 7.5: Abstract Lexicon Structure



Description of the continuation classes of the abstract noun lexicon including subdivisions based on lexicalised prefixes (*ga roots*, *o roots*...), subdivision based on stem-initial phones (*C roots*, *V roots*, *ϱ roots*...) and flag diacritics (Flag1 & Flag2).

7.1.1.7 Structural Overview of the Lexicons

Having described each part of the lexicon in detail, I will now briefly compare the concrete lexicon to the abstract lexicon. (The latter is diagrammed in full in figure 7.5). It is important to note that while in several cases there are redundant paths that would cause the same word form to be defined multiple times, these do not actually affect the final number of words in the FST. Identical forms are automatically removed by the XFST compiler.

In some respects, the two versions of the FST are identical. For example, both the concrete and the abstract machine process unpossessed basic nouns as in example (10).

(10) Nouns+ Basic+ Unpossessed prefixes+ gaRoots +NSF
ø+ ø+ ga+ ˺wahsa: +a˺#

These nouns are represented identically because the prefix that is used is lexicalized. The major difference between the concrete FST and the abstract FST resides in the lexicon in the processing of the regular affixes.

The abstract model was straightforward to develop; the noun stems needed to be subdivided into classes depending on the lexicalised unpossessed prefix that they take and rules were used to generate all other prefix allomorphs as in example (12).

(11) a. Nouns+ Basic+ Possessed prefixes+ allRoots+ gaRoots +NSF
ø+ ø+ ag+ ø+ ˺wahsa: +a˺#

b. /ag+˺wahsa:+a˺/→/age˺wahsa:˺/

Example (11-a) represents the output of the abstract lexicon module and (11-b) represents the application of the rules module the converts the UR to a valid SR.

In the case of the concrete approach, however, there were two methods that could be used to design the machine: either a fully lexicalised approach could be taken, or a machine could be designed that overgenerates and uses rules to filter out the unnecessary forms.

Part of the basis for the decision to create a concrete and abstract approach was to test if there was a computational cost for the addition of extra rules; it therefore seemed

counter-intuitive to design that concrete approach with a large number of additional rules to filter out over-generating forms; such an approach would be essentially analogous to the abstract approach itself.

The fully lexicalised (concrete) approach requires defining specific sub-lexicons for the noun stems. These sub-lexicons contain noun roots grouped by their stem-initial phones (graphs) and the lexicalised unpossessed prefixes. In example (12) the prefix "age-" must be followed by a noun root that begins with ˀCV.

(12) Nouns+ Basic+ Unpossessed prefixes+ ˀCV roots +NSF
ø+ ø+ age+ ˀwahsa: +aˀ#

This approach means that for each type of noun the stems were sub-divided into groups based upon their initial phones and the unpossessed prefix form they take making for a total of 73 lexicons in the concrete approach versus 34 in the abstract approach. The very high number of sub-lexicons results from the need to repeat all sub-lexicons for each stem type for each class of unpossessed prefix, as explained below. A small sub-set of the necessary sub-lexicons are listed in examples (13)-(16) here while the full spectrum is detailed in figure 7.6.

(13) Basic nouns taking /ga-/ when unpossessed but /age-/ when possessed in the first person singular.

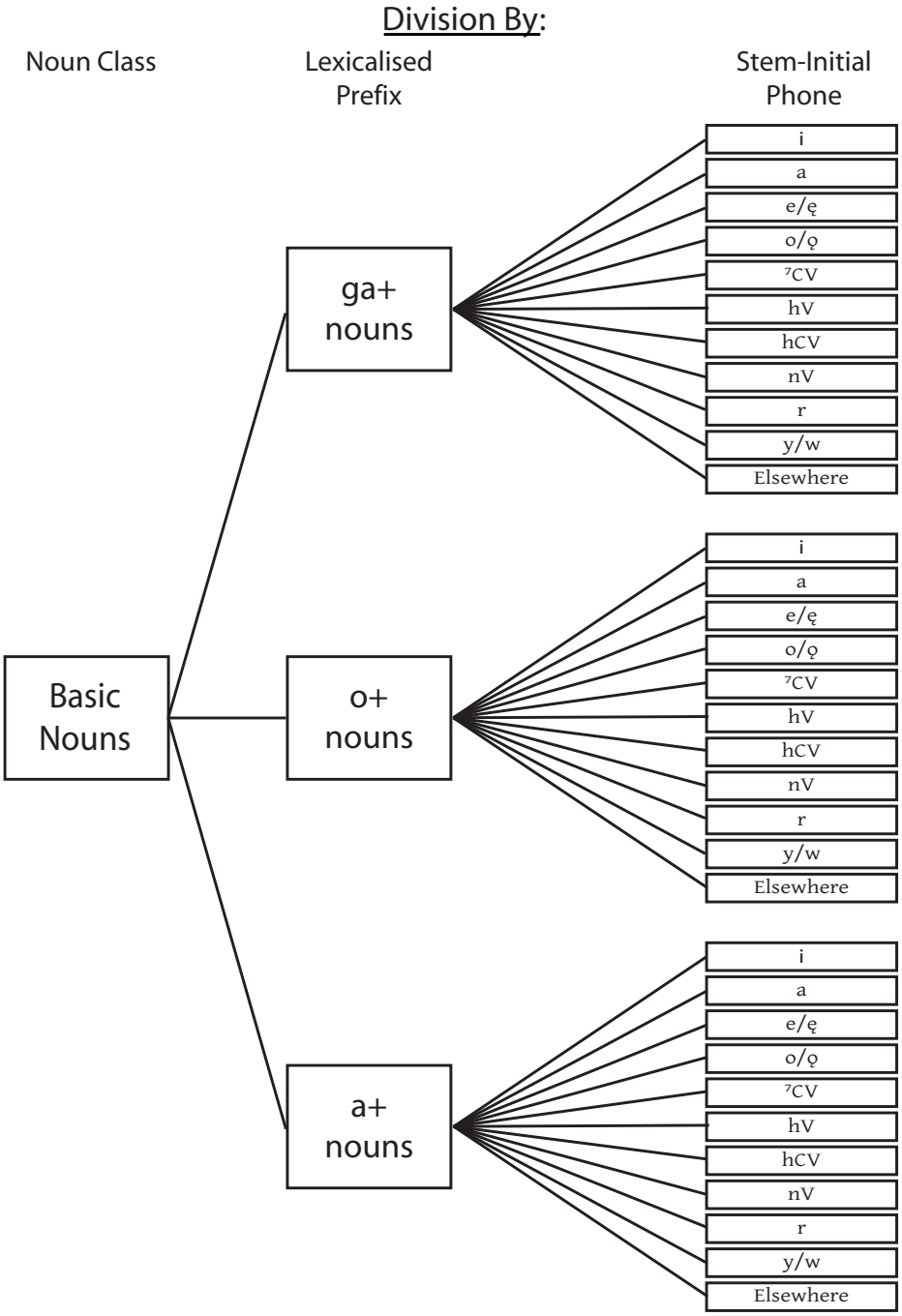
(14) Basic nouns taking /ga-/ when unpossessed but /ag-/ when possessed in the first person singular.

(15) Basic nouns taking /o-/ when unpossessed but /age-/ when possessed in the first person singular.

(16) Basic nouns taking /o-/ when unpossessed but /ag-/ when possessed in the first person singular.

Figure 7.6 clearly demonstrates the two types of subdivisions for the basic nouns: lexicalised prefixes (3: *ga+, o+, a+*) and based on initial phone(s) (11 conditioning environ-

Figure 7.6: Concrete Approach Basic Noun Lexicons



Multiple subdividing of nouns based on lexicalised prefix and stem-initial phones results in a very high number of sub-lexicons (potentially 33 for possessed basic nouns alone) that can quickly become difficult to manage.

ments). Examples (13) and (14) show two types of words which share the same unpossessed prefix, but different possessed prefixes, and hence need to appear in different sub-lexicons.

The words are first sub-divided into 3 sub-lexicons according to the lexicalised unpossessed prefixes they take. Then, the forms in these sub-lexicons were again sub-divided based on initial phone. This makes for a total of 33 possible lexicons for just the basic noun roots. Although there are currently no attested forms for some of the possible sub-lexicons, nouns that appear in these categories could theoretically exist. The manner in which this creates a large number of sub-lexicons is clear.

Now having thoroughly described how the lexicons concatenate morphemes and some possible alternative approaches that were not used in my project, I will go on to discuss the rules module that turns segmented morpheme sequences into surface forms and vice versa.

7.1.2 Rules Module

Unlike in the case of the lexicon module, the rules modules for the two versions contain nearly identical rulesets. The set of rules for the concrete approach is a simple subset of the rules for the abstract approach. Rules for the concrete approach implement almost none of the prefix rules, just stem+suffix alternations and clean-up rules. This made the generation of the rules module for the concrete machine more straightforward.

The variety of vowel changes in Cayuga is rather large, and the juncture of two morphemes is rarely just a case of A + B = AB. The rules module of the FST replicates the morpho-phonological processes that occur at morpheme boundaries in Cayuga (see §6.1.1):

Example (17) demonstrates just a few possible vowel sandhi rules that occur in Cayuga morphology.

(17) Example Morpho-phonemic Alternations

a. o + idǫhgwa + a˥
   odǫhgwa˥
   3NP flame NSF *flame*

b. o + adeshę + a˥
   odeshę˥

3NP cocoon NSF *cocoon*

c. ga + itsga: + a˺

gę̨tsga:˺

3NA mattress NSF *mattress*

We see that o + i/a at a prefix/stem boundary gives o (see examples (17-a) and (17-b)). The data in (17-c) also show that a + i at a prefix/stem boundary produces ę̨ and that V + a˺ produces V˺ at the stem/NSF boundary.

7.1.2.1 Rules Module Components

The rules module consists of three component parts: (a) a module that removes deleting prefix segments (§4.2.2), (b) the actual morpho-phonological rules and (c) a set of "clean-up" rules. The entire set of rules contained in the rules module are contained in appendix C. Each of these rule components were only separated for clarity. Some grammars have separate rules components for nouns and for verbs.[5]

In XFST, rules are generally specified only in terms of phones and morpheme boundaries. Any symbols other than letters, vowel length markers, or morpheme boundary markers can block the application of rules. For this reason, the first module to apply removes any word-initial deleting prefix segments (denoted by parentheses) as in the **(y)** in *(y)o+* (**3znsP**). Then for deleting segments that are not word-initial, it removes the parentheses surrounding the segment.[6]

Table 7.2 gives examples of word-initial segment deletions.

In table 7.2 we see the application of two rules. Rule '**Delete Opt**' deletes any deleting segments that are word-initial. The second rule, '**Delete Parentheses**' then removes any other parenthesis that remain in the words. These two rules together remove all parentheses while keeping non-word-initial prefix segments and allowing further rules to apply unobstructed.

[5]In this case, however, all rules apply to all word forms, so there is no need to separate the separate modules.

[6]The symbols '(' and ')' were not used while formulating the morpho-phonological rules; therefore they could block rule application. A rule stating that /e/→Ø/u+_ would not apply to *du+(e)tni+...* because the parentheses are not stated in rule.

Table 7.2: Two Rules for Optional Prefix Segment Removal

Rule	UR: (y)o+˺nhǫhs+a˺	UR: de+(y)o+˺nhǫhs+age:
Delete Opt	o+ ˺nhǫhs+a˺	de+(y)o+˺nhǫhs+age:
Delete Parentheses	o+ ˺nhǫhs+a˺	de+yo+˺nhǫhs+age:
Output	o+ ˺nhǫhs+a˺	de+yo+˺nhǫhs+age:

The second module applies the remaining morpho-phonological rules, including vowel hierarchy deletions, vowel coalescence processes, epentheses and so forth. Table 7.3 shows the application of a subset of rules to an abstract and a concrete underlying form.

Table 7.3: Morpho-phonological Rule Application

	Concrete	Abstract
UR:	(h)j+ahyagwiy+ageh˺	(h)sni+ahyagwiy+ageh˺
sn→j/_i+a	n/a	(h)ji+ahyagwiy
Vowel Hierarchy	n/a	(h)j+ahyagwiy
Output:	(h)j+ahyagwiy+ageh	(h)ji+ahyagwiy

This module contains 23 rules in the abstract version and 11 in the concrete approach. The concrete approach only contains basic rules such as vowel hierarchy deletion rules; vowel lengthening rules and so forth. It does not specify most forms of coalescence; voicing or devoicing rules because, as can be seen in table 7.3, these rules are already encoded in the UR.

The third module applies clean-up processes such as the deletion of multiple identical vowels at morpheme boundaries, the removal of morpheme boundaries and the removal of abstract consonants. Some roots contain abstract phones ([C]) which block the application of some coalescence and vowel hierarchy rules. Example (18) shows the actual (18-a) UR and SR of a form with an abstract C versus the expected (18-a) UR and SR.

(18) a. UR: ga+ Cisra +a˺
SR: ga- isr -a˺

b. UR: ga+ Cisra +a˺
SR: *gę- sr -a˺

The abstract C in (18-a) blocks the application of the vowel coalescence rule that turns / a + i / into [ę]. In (18-b) since there is no abstract C to block the process, the vowels incorrectly coalesce into [ę].

7.1.3 Semantics Modules

I created a simple semantics modules that glosses the stems, prefixes and suffixes with a very basic English gloss. The semantics module was generated by slightly modifying the lexicon module such that the transductions between a morpheme and its semantics were directly encoded into a lexc file. Essentially the semantics module is an enhanced lexicon that contains not only morphemes and how they can connect, but also the semantics for each morpheme.

Table 7.4 compares a regular lexicon used for segmentation (left) with a modified lexicon used for generation for semantics (right).

Table 7.4: lexc samples for semantics module and lexicon module

Regular Lexicon	
LEXICON	inalienablePrefixes
(y)agwa+	inalienableStems ;
LEXICON	inalienableStems
ahǫhd	NSF ;
LEXICON	NSF
+a˺	# ;

Semantics Lexicon	
LEXICON	inalienablePrefixes
1epA+ : (y)agwa+	inalienableStems ;
LEXICON	inalienableStems
on your ears : ahǫhd	NSF ;
LEXICON	NSF
+NSF : +a˺	# ;

As can be seen in the right-hand columnn of table 7.4, the basic semantic definition of a form is given to its left separated by a colon (for example, **+NSF :** +a˺). An alternative method of implementing the semantics module would have been a list of transformation rules that would contain the semantics for all stems, prefixes and suffixes. Either method would produce the same results; however, a major drawback of the method I applied was that when the basic lexicon was changed, the semantic lexicon had to be changed to match. A rule-based approach (in which semantic glosses are added by rules) would not need such changes.

It is also important to note that I could not just use the semantics lexicon in place of the regular lexicon because then the intermediate output (the segmented morpheme) would not be generated, just the semantics.

7.1.4 Interface

A user interface was not part of the original set of specifications, but I decided that it was more satisfying to create a basic one. I designed a simple web based interface for the program using Python[7] and PHP.[8] This interface allows users to enter words into a simple form field and then receive the segmented form of the word, the semantics and the set of related prefixes. Currently this interface uses the concrete version of the FST to give users concrete underlying forms. Additionally the interface gives a brief explanation of the prefix semantics.

Figure 7.7 outlines the entire process in the form of a flow chart explicitly stating the intermediate tasks assigned to the python script. At point (**A**), the user is presented with a form field in which to input a word. This word is passed to the Python script which renders the word to lower-case[9] and ensures that only a single word has been entered (**B**).

This single lowercase word is then passed by the script to point (**C**) the segmenting FST (the composition of the lexicon and rules components as described above). The output of the segmenting FST is returned to the script (**D**) which outputs the result to the user and also passes the information to (**E**): the semantics FST.

The semantics FST in turn returns its result to the script (**F**). The script then outputs this result, and then separates the prefix semantics to be passed back to the semantics FST in reverse (**G**). The result of (**G**) is then passed to the script (**H**) which outputs that result to the user.

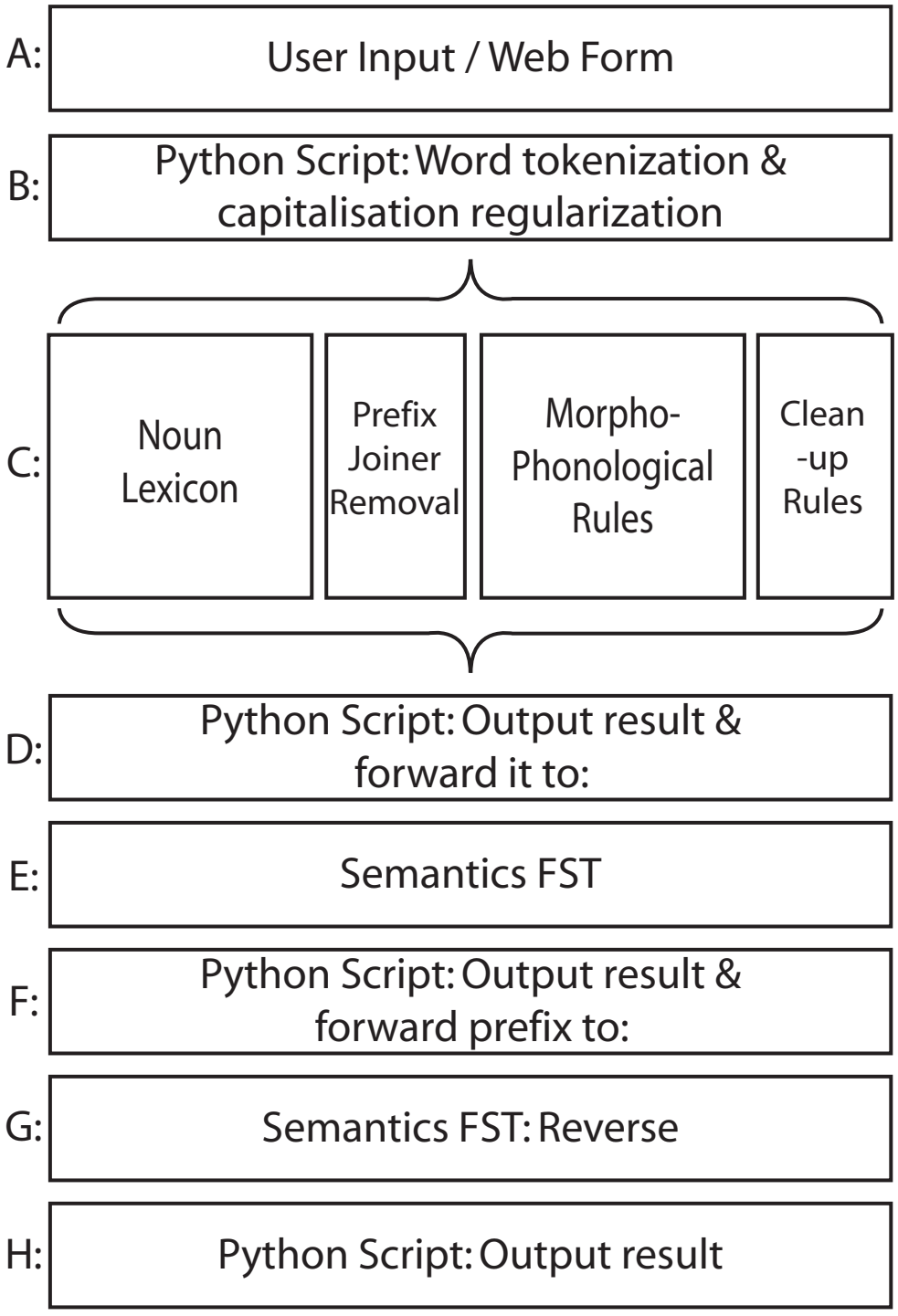
Table 7.5 describes the information flow through the interface.

[7]Python is a standard procedural programming language. For more information please visit http://www.python.org

[8]PHP is a standard procedural programming language. For more information please visit http://www.php.net. The PHP scripting performs no task other than to pass the user-input to the python script for processing.

[9]The FST has been designed in lower-case, it can however, be modified to be case insensitive.

Figure 7.7: Complete Program Flow Chart



Flow chart demonstrating how user input is passed through to each module of the machine using the python script as an intermediary.

Table 7.5: Stages of Output from User Interface

a	**User Input to Segmentation**	ohǫnaˀdaˀ
b	**Segmentation Output**	(y)o+hǫnaˀd+aˀ
c	**Input to Semantics**	(y)o+hǫnaˀd+aˀ
d	**Semantics Output**	3znsP+potato+NSF
e	**Input to Reverse Semantics**	3znsP+
f	**Reverse Semantics Output**	*List of prefix allomorphs with conditioning environments, see example (19)*

The interface actually runs three FSTs in order to produce the results. First the word entered by the user is run through the combined lexicon module and the rules module (table 7.5 **a**). This FST outputs the segmented form (table 7.5 **b**) or an error if the word does not exist in the machine.

The output from that FST is then run through the semantics module to generate the semantics for the form (table 7.5 **c** and **d**).

Finally, the prefix from the output of the semantics module is run through the semantics module again, this time **in reverse** (table 7.5 **e** and **f** and example (19)), exploiting the bidirectionality of FSTs. At both stages the output is saved and displayed to the user.

The output of the second pass through the semantics (Table 7.5 **f**) module is formatted as such:

(19) Prefixes meaning **3zndpA**:

- gę+inalienable noun root beginning with +a
- gad+inalienable noun root beginning with +i
- gadi+inalienable noun root beginning with +ˀC
- ...

This interface was quite straight-forward to design, demonstrating that XFST is worthwhile for developing usable applications.

7.2 Addressing the requirements

There were several requirements that each machine should meet in order to determine which approach was most useful. The machines should (a) provide an adequate arena for testing the morpho-phonological rules of Cayuga; (b) properly generate and segment all and only Cayuga nouns; (c) help investigate the usefulness of the FS framework for polysynthetic languages; (d) provide ideal dictionary access; and (e) be efficient and elegant. Each of these requirements is discussed below.

7.2.1 Testing of Rules

Determination of morpho-phonological rules and their orderings can often be difficult for a variety of reasons. Applying rules to a large number of forms can be very time consuming and difficult if done manually. Finite-state machines allow the user to run large corpora of data against a given ruleset very quickly, and furthermore allow the changing and updating of those rules with ease.

I compiled a set of test-case files included in Appendix C in order to ensure that the rule-orderings were producing the appropriate results for all noun types. This required firstly determining the set of conditioning environments (e.g., Table 7.7) and then selecting a set of words exemplifying each underlying prefix form with each stem type.

Two lists were created, one containing the valid underlying forms and one containing valid surface forms of those same words for comparison with the machine's output. A second set of underlying forms had to be generated for the concrete approach since the prefix alternants are lexicalised.

After running the list of underlying forms through the machine I compared its output automatically with my list of correct surface forms. The automatic comparison listed forms that should not have been output as well as forms that should have been, for example, see table 7.6 (b):

Table 7.6: Rule-testing Samples

	Input	**Output**	**Expected Output**	**Result**
(a)	(ya)godi+hǫna˺d+a˺	godihǫna˺da˺	godihǫna˺da˺	**OK**
(b)	(w)ag+hsgwaę˺d+a˺	aksgwaę˺da˺	agehsgwaę˺da˺	**The SR "aksgwaę˺da˺" was produced which is not a valid noun of type X. The SR "agehsgwaę˺da˺" was not produced.**

Finally, I ran the same process in "reverse" to check that the bidirectionality of the machine was intact: surface forms gave the proper underlying forms and vice versa. As described in the following section, some errors in the posited rules were revealed.

7.2.1.1 Errors Discovered

One problem was discovered with the rule formulations: that the **1sP** basic noun prefix was not always generating the correct allomorph. The **1sP** prefix was listed as having the following C-stem allomorphs:

Table 7.7: Basic Nouns Patient Prefix Allomorphs - C-stems

Gloss	**Prefix UR**	**˺CV**	**hV**	**hCV**	**nV**	**r**	**y/w**	**Elsewhere**
1sP	(w)ag+	ag-	ak-*	age-	ak-	ag-	ag-	age-

In the course of testing it became clear that this description was not fully adequate. The current description describes nouns beginning in [˺CC] as a single environment. However, as can be seen from example (20), not all [˺CC] stems in fact pattern identically.

(20) a. age+ ˺drehd +a˺
1sP car; truck; vehicle NSF
My car(s)

b. *age+ ˺nhǫhs +a˺

c. ak+ ˀnhǫhs +aˀ
1sP egg NSF
my egg(s)

According to the current description of morph-phonological alternations, the stems -ˀ*drehd*- and -ˀ*nhǫhs*- should both pattern identically and take the same **1sP** allomorph *age+*. However, as can be seen in (20-b) and (20-c), this is not the proper allomorph for the stem -ˀ*nhǫhs*-. This demonstrates that the machine description, by systematically applying itself to all nouns, can detect inconsistencies in rule formulation/conditioning environment description.[10]

7.2.2 Generation & Segmentation

The above method for testing covered all regular alternation types. If there are any unrecognized forms still in the machine they are then a fault of either an error in the initial data or caused by human error in the creation of the lexicon file. The machines each generate and segment over 4000 nouns with almost 100% coverage, with the only errors resulting from the occasional mis-analysis of [e]-epenthesis described in the preceding section.

7.2.3 FS Applicability to Polysynthetic Languages

In general, it does not seem that any inordinate amount of complexity arises as a result of the large number of phonological alternation rules or as a result of (a limited number of) long-distance dependancies. In fact the two machines that I created were each quite small in final size. However, because this machine only implements a single long-distance dependancy it is not clear whether or not a fully implemented language FST would encounter difficulties arising from that issue.

While there were no complexity issues in the development of either machine, there may be some issues that arise in future work due almost solely to the morphological complexity

[10]There are two ways to approach the modelling of [e]-epenthesis: (a) a linear description (ø→[e]/C_CC), or (b) a non-linear description (apply the rule in (a) only in the case where some of the Cs would remain unsyllabified otherwise). I have assumed a linear approach as a non-linear approach would require either a complex environment that accounts for the differing syllabifications of continuants and non-continuants, or a syllabification module. It is beyond the scope of this paper to model the complex environments of [e]-epenthesis.

found in First Nations languages. It seems likely that highly synthetic[11] languages, such as those in the Athapaskan family, may require special care for the designing of morphological segmenters. There will be not only many prefixes but also many more cases of long-distance dependancies and morpho-phonological variation will be much higher.

7.2.4 Ideal Dictionary Access

The machine produces a basic semantic gloss for all valid noun forms and has a straightforward interface. The user need only type a word into a text box to retrieve its morphological and semantic information. The output of the machine is very accessible.

A potential drawback of the current version of the machine is that it is somewhat restricted by its use of an intermediate (non-standard) orthography lacking accent-related diacritics (§4.1). In its current version therefore it does not meet all the criteria of an accessible dictionary. This problem, however, can be easily resolved by the introduction of additional transducers to take input in either of the Henry or linguistic orthographies.

7.2.5 Efficiency, Elegance and Usability - Abstract vs. Concrete Versions

Both the abstract and the concrete versions are almost identical in the terms of computational efficiency initially laid out in §6.1.1.2. The two machines have statistically insignificant differences in the number of states, transitions and in file size. As can be seen in table 7.8 the abstract machine is slightly larger (by 0.4Kb), has 6 additional states (out of 1054), and 20 additional transitions (out of 1943). The comparative evaluation of the two machines, therefore, must rest on other criteria, as discussed below.

Table 7.8: Sizes of Final Segmenter FSTs

	File Size (Kb)	States	Transitions	Paths	Rules	Roots	Prefixes
Abstract	40.5	1054	1943	4009	28	413	45
Concrete	40.1	1048	1923	4006	19	413	149

[11]Synthetic languages "typically contain more than one morpheme". In a highly synthetic language the words would typically contain more than 3 or 4 morphemes, as in, for example, Totonac. (Crystal, 2003)

In terms of processing speeds, the abstract and concrete versions are, again, nearly identical. I timed 5 sets of 100 iterations of segmentation and combination of morphemes using the test cases described above (713 words per iteration for the abstract machine and 712 for the concrete machine). As can be seen from the data in tables 7.9 and 7.10, the speed of running the combination of an underlying form to produce a surface form was very slightly slower than segmenting a surface form into an underlying form. However, both machines show little difference in speed and on average segment single words in about 0.0002145s and combine single words in about 0.2315s.

Table 7.9: Time (in seconds) for 100 iterations of morpheme combination of the test corpora

Machine	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**	**Average/word**
Abstract	15.325	15.626	15.203	15.102	15.110	15.273	0.0214
Concrete	15.870	15.329	15.749	14.979	14.784	15.342	0.0215

Table 7.10: Time (in seconds) for 100 iterations of segmentation of the test corpora

Machine	**Run 1**	**Run 2**	**Run 3**	**Run 4**	**Run 5**	**Average**	**Average/word**
Abstract	16.570	16.317	16.504	16.528	17.319	16.648	0.0233
Concrete	16.633	16.245	16.254	16.284	16.325	16.348	0.0230

The two machines contain both internal and external ambiguity (non-determinism). The external ambiguity was necessary because some lexical items have more than one potential meaning and some surface prefixes can be analysed as one of 2 or more underlying prefixes. This non-determinism does not seem to impede the performance of the final machine, most likely as it occurs early on in the words processing time is only marginally affected and as there are only a small number of cases machine size is not greatly affected either.

Initially, the concrete approach shows some distinct advantages over the abstract approach. It is easier to have it generate useful output for the users. As well, there are some possessive prefixes that need to be listed lexically regardless of the approach. This somewhat mitigates the argument of linguistic elegance as an advantage of the abstract approach. Since some continuation classes are necessary regardless, and the concrete machine uses them throughout, the concrete approach is more consistent. (See §4.2.3, §6.1.1.1 and §7.1.1 for a further discussion of the opaque prefixes).

The abstract approach has more composition operations than the concrete approach and to expand the machine to include verbs and other word-forms would certainly require even more rules to be added (resulting in more composition operations). However, there are unlikely to be many more rules that need to be added to account for other word-forms and several of those rules would still need to be applied to the concrete machine, so it seems unlikely that there will be a substantial growth in the size of the abstract machine relative to the concrete machine with the addition of further roots and prefixes. So, in terms of future machine growth, the two approaches seem equal.

The distinct advantage to the abstract approach that makes it the best choice for future work, however, is that of elegance (not just linguistic elegance, but equally importantly, computational elegance). The abstract approach is significantly simpler to design, implement, maintain and update than the concrete approach, as described below.

When a bug was found in the abstract approach, either a single form in the lexicon needed to be edited, or a small number of rules needed to be modified or rearranged. In the case of the concrete approach, however, all major bugs were in the lexicon; if there was an initial design error, then going back to fix it was significantly more difficult. In other words, the concrete approach was significantly less computationally elegant than the abstract approach. For example, suppose it had been incorrectly assumed that /(e)dwa+ǫ.../ become /(e)dw+ǫ.../. To fix this problem in the concrete approach, all inalienable roots beginning with [ǫ] would have to be relocated to new sub-lexicons. Since the roots beginning with [ǫ] are distributed across multiple sub-lexicons, and it is necessary to keep their other groupings intact, this would be a lengthy endeavour. To fix this problem in the abstract approach, however, one would need only to modify a single rule to effect the change.

Finally, while the abstract machine in its current form is slightly less usable than the concrete machine, it can be modified to provide more user-friendly output. Currently, the abstract machine cannot provide a list of related prefix forms for an underlying prefix, while the concrete machine can. Such a list is of great use to speakers who might have difficulty identifying related prefix allomorphs. This problem can easily be solved at a later date by modifying the semantics module and combining it with a modified version of the rules module.

The abstract approach, then, is more computationally and linguistically elegant, meaning it will be easier to upgrade and modify in the future. Any drawbacks that it may have

compared to the concrete approach can be satisfied by the creation of a number of further modules, at an apparently low cost in terms of computational efficiency.

Chapter 8

Conclusion

8.1 Future Work

There is a wide array of uses to which my application can be put. The innate modularity of the FS framework means that my work can be easily extended to include enhancements such as orthographic conversion between the three orthographies or recognition of more complex prefix combinations and more word types.

Dictionaries can be constructed that will be more accessible to their users. The user will no longer need to have a complex grasp of the morphology and morpho-phonology of their language, simply to find the definition of a word (§5.1).

This machine could be expanded to include all Cayuga word-forms. Almost all of the rules that are necessary for analyzing verbal forms are independently needed to model nominal forms, so very few new rules would need to be added. In principle this approach should work for verbs, but it remains to be fully tested. It would also be useful to implement a more rigorous semantics module that could more fully implement dictionary-like semantic readings.

Furthermore the simple parser that I have described can be enhanced by the addition of a variety of other modules such as spell-checkers, syllabifiers and the ability to use multiple orthographies (something that is of particular use in the context of First Nations languages which typically make use of multiple orthographies). A spell-checking dictionary that could guess what a user meant if they mis-typed a word would be ideal for many First Nations languages where the spelling system and the morphology are often complex and confusing to speakers.

8.2 Summary Conclusions

In this document I have detailed my implementation of a morphological parser for Cayuga using the finite-state framework. This parser demonstrates that the finite-state framework can generally handle problems typical of polysynthetic languages such as obligatory prefixing and long-distance dependancies and that the finite-state framework is a versatile tool for NLP applications.

Much future work will need to be done to fully establish the finite-state framework's usefulness for full-scale implementations of morphologically complex languages but my machine as a proof-of-concept is the first step towards such a full-scale implementation.

BIBLIOGRAPHY

Alegria, Iñaki, Maxux Aranzabe, Nerea Ezeiza, Aitzol Ezeiza, and Ruben Urizar. 2002. Using Finite State Technology in Natural Language Processing of Basque. In *Implementation and Application of Automata: 6th International Conference in the Lecture Notes in Computer Science Series*, ed. by B.W. Watson and D. Wood, volume 2494, 1–11. Berlin, Heidelberg: Springer-Verlag GmbH.

Beesley, R. Kenneth, and Lauri Karttunen. 2000. Finite-State non-concatenative morphotactics. In *Special Interest Group in Computational Phonetics*, volume (2000), 1–12.

Beesley, R. Kenneth., and Lauri Karttunen. 2003. *Finite State Morphology*. Stanford, CA.: Center for the Study of Language and Information Publications.

Chafe, Wallace L. 1967. *Seneca Morphology and Dictionary*. Number 4 in Smithsonian Contributions to Anthropology. Washington D.C.: Smithsonian Press.

Chomsky, Noam. 1956. Three Models for the Description of Language. *IRE Transactions on Information Theory* 2: 113–124.

Crystal, David. 2003. *A Dictionary of Linguistics and Phonetics*. Malden, MA: Blackwell Publishing, 5th edition.

Dyck, Carrie. 2006. Speakers prefer concrete prefixes. Personal Communication.

Foster, Michael. 1986. Updating the terminology of tense, mood, and aspect in Northern Iroquoian descriptions. *International Journal of American Linguistics* 52: 65–72.

——, Karin Michelson, and Hanni Woodbury. 1991. *Base and affix dictionary for Iroquoian languages*. N.Y.: Ms. Snyder.

Frank, Robert, and Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24.2: 307–315.

Froman, Frances, Alfred Keye, Lottie Keye, and Carrie Dyck. 2002. *English-Cayuga / Cayuga-English Dictionary*. Toronto: University of Toronto Press.

Hopkins, Alice W. 1989. *Theoretical Perspectives on Native American Linguistics*, chapter Vowel Doubling in Mohawk, 445–459. Albany: State University of New York press.

Kaplan, Ronald M., and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20.3: 331–378.

Karttunen, Lauri. 2001. Applications of Finite-State Transducers in Natural Language Processing. In *Implementation and Application of Automata : 5th International Conference*, ed. by S. Yu and A. Paun, volume 2088, 33–46, Berlin, Heidelberg. CIAA, Springer-Verlag GmbH.

Kiraz, George Anton, and Bernd Möbius. 1998. Multilingual syllabification using weighted Finite-State Transducers. In *Proceedings of the Third ESCA Workshop on Speech Synthesis*, Australia. Jenolan Caves.

Koskenniemi, Kimmo. 1997. Representations and Finite-State components in Natural Language. In *Finite-State Language Processing*, ed. by Emmanuel Roche and Yves Schabes, 99–110. Cambridge: MIT Press.

——, and Kenneth Ward Church. 1988. Complexity, Two-Level Morphology and Finnish. In *COLING*, 335–340.

MacKenzie, Marguerite, and Bill Jancewicz (eds.) 1994. *Naskapi Lexicon*. Kawawachikamach: Naskapi Development Corporation.

Michelson, Karin, and Mercy Doxtator. 2002. *Oneida-English / English-Oneida Dictionary*. Toronto: University of Toronto Press.

Mithun, Marianne. 1979. The Consciousness of Levels of Phonological Structure. *International Journal of American Linguistics* 45.4: 343–348.

Mohri, Mehryar. 1997. Finite-State Transducers in language and speech processing. In *Association for Computational Linguists*, volume 23, 269–311.

Nederhof, Mark-Jan. 1996. Introduction to Finite-State techniques. Lecture notes.

Oflazer, Kemal. 1994. Two-level description of Turkish morphology. *Literary and Linguistic Computing* 9.2.

Reichel, Uwe, and Karl Weilhammer. 2004. *Automated Morphological Segmentation and Evaluation*. Lisbon, Portugal: n/a.

Roche, Emmanuel, and Yves Schabes. 1997. *Finite-State Language Processing*. Cambridge, Mass.: MIT Press.

Sproat, Richard W. 1992. *Morphology and Computation*. Cambridge, Mass.: MIT Press.

Tinsley, John. Accessed: 2007 02 13. Spanish Morphological Analyser/Generator. *http://www.redbrick.dcu.ie/ tinsley/* .

Tzoukerman, Evelyn, and Mark Y. Liberman. 1990. A finite-state morphological processor for Spanish. In *COLING-90: Papers Presented to the 13th International Conference of Computational Linguistics*, ed. by Hans Karlgren, 277–282, Helsinki. Helsingiensis Universitas.

Vilares, M., J. Otero, F.M. Barcala, and J. Dominguez. 2004. Automatic Spelling Correction in Galician. In *Lecture Notes in Artificial Intelligence*, number 3230 in España for Natural Language Processing 2004, 45–57. Berlin: Springer-Verlag.

Wikipedia. Accessed: 2006 03 24. Finite State Machines. *http://en.wikipedia.org/wiki/Finite_state_machine* .

Young, Robert W., and William Morgan. 1987. *The Navajo Language, A Grammar and Colloquial Dictionary*. Albuquerque: University of New Mexico Press, revised edition.

Appendix A

Morpho-phonological and Clean-up rules

This section contains the morpho-phonological rules and the clean-up rules as described in §7.1.2.1. The optional prefix segment rules were left out as they are relatively straightforward.

Rules prefixed by * appear only in the abstract version of the program. Other rules appear in both versions[1]. Finally rules prefix by a † implement minor spelling variations characteristic of the Henry Orthography.

1. *† /g/ → [k] /__ + $\left\{ \begin{array}{l} \text{h(C)V} \\ \text{ˀCV} \\ \text{nV} \end{array} \right\}$
2. † / h / → ø / s +__V
3. / h, ˀ/ → ø / k __ + C
4. *ø → [e] / $\left\{ \begin{array}{l} \text{s} \\ \text{g} \end{array} \right\}$ __ $\left\{ \begin{array}{l} \text{C}_o \\ \text{ˀV} \\ \text{CC} \end{array} \right\}$
   (C_o consists of [k, g, t, d, h, s, j])
5. Some prefixes have additional initial vowels that are only pronounced when preceded by another phone, these are deleted as necessary here, and actually consists of 3 separate rules.
6. /a + i/ → [ę]
7. */o/ → [aw] / _ + $\left\{ \begin{array}{l} \text{e} \\ \text{ę} \end{array} \right\}$
8. /V/ → ø / $\left\{ \begin{array}{l} V:+ \\ V+V \\ VV+ \end{array} \right\}$__
   (The final vowel in a tri-moraic vowel sequence at a morpheme boundary is deleted)
9. */ni/ → [y] / _ + a
10. */adi/ → [ęn] / __ + $V_{\text{-i}}$
    ($V_{\text{-i}}$ is the set of all vowels except [i])

[1] In some cases I have have combined rules together for clarity where they are actually stated as separate in the rules module.

11. */di/ → [n] / _ + V_{-i}

    Rules (12 - 16) represent vowel hierarchy deletions. The hierarchy from strongest to weakest is [ǫ, o, ę, e, a, i]

12. /V/ → ø/ $\left\{ \begin{array}{l} \_\_ + ǫ \\ ǫ + \_\_ \end{array} \right\}$

13. /o, ę, e, a, i/ → ø/ $\left\{ \begin{array}{l} \_ + o \\ o + \_ \end{array} \right\}$

14. /ę, e, a, i/ → ø/ $\left\{ \begin{array}{l} \_\_ + ę \\ ę + \_ \end{array} \right\}$

15. /e, a, i/ → ø/ $\left\{ \begin{array}{l} \_ + e \\ e + \_\_ \end{array} \right\}$

16. /a, i/ → ø/ $\left\{ \begin{array}{l} \_ + a \\ a + \_ \end{array} \right\}$

17. */w/ → [y]/ $\left\{ \begin{array}{l} \_ + o \\ o + \_ \\ \_ + ǫ \\ ǫ + \_\_ \end{array} \right\}$

18. */k, t, d/ → [g] / __ y +[2]

19. */sy/ → [j] / __ +

20. /V_1+V_1/ → [V_1:]

21. /r/ → ø / V__V

22. /C_A/ → ø
    Some roots require abstract consonants that block morpho-phonological processes such as the coalescence rule in (6). These abstract consonants are deleted here.

23. /V_1/ → ø / V_1(:) __ +
    Some rules leave behind groups of identical vowels at morpheme boundaries such as '...a:+a...' or 'ǫ+ǫ...'. All such sequences are cleaned up here.

24. /+/ → ø
    θhis rule deletes the morpheme boundary marker.

[2]This rule implements a difference between the lower Cayuga dialect dealt with in this study, and the upper Cayuga dialect.

Appendix B

Code

This appendix contains the XFST Lexc code used to generate the abstract and concrete lexicons. This code specifies the valid morpheme combinations, but does not apply any rules. The design of this code is outlined in §7.1.1. This code is written in Lexc as described in §6.3.1.

There are four sections to this appendix. The broad division is between the abstract and concrete version and within those sections is the code for both the semantic lexicon and the regular lexicon.

B.1 Abstract Version

B.1.1 Abstract Lexicon

```
Multichar_Symbols @U.INALIEN.POSS@ @U.INALIEN.UNPOSS@


LEXICON                     ROOT
                            allNouns ;


LEXICON                     allNouns
                            inalienableNouns ;
                            basicNouns ;
                            deverbalNouns ;
                            defectiveNouns ;


LEXICON                     defectiveNouns
sgwa:gwaǫdǫ̃       # ;
dago:s                     # ;
da:gu:s                     # ;
dakshae̓dohs       # ;
so:wa:s                     # ;
twę:twę:t              # ;
hǫ:ga:k                     # ;
dogę:t                     # ;
gwihsgwihs               # ;
```

```
gwaˀyǫˀ                    # ;
sohǫ:t                     # ;
gyo:gyo:ˀ                # ;
jogrihs                      # ;
gwidoˀgwidoˀ          # ;
diˀdi:ˀ                    # ;
jikjiye:ˀ                    # ;
gaˀga:ˀ                    # ;
hihi:                     # ;
gwiyęˀgwiyęˀ             # ;
dihsdihs                     # ;
jiˀnhǫwę:se:             # ;
duwisduwi:ˀ              # ;
saˀsaˀ                     # ;
gwę:dihs                     # ;
gwe:sęˀ                    # ;
tsahgo:wah              # ;
jihsgogoˀ               # ;
gwaoh                      # ;
johwęˀsdagaˀ             # ;
gwęˀgohnyęˀ             # ;
hnyagwaiˀ               # ;
gǫ:deh                     # ;
tgwiyo:gęˀ               # ;
jinhǫhgwahęh          # ;
jiˀnǫhdo:yaˀ              # ;
jiˀdana:wę:              # ;
jinǫhsanǫh               # ;
jihsda:                      # ;
jiˀao:yę:                    # ;
jinǫhyahae:              # ;
degriyaˀgǫˀ              # ;
jihnyoˀgęˀ               # ;
hehshai:                     # ;
sgwaˀahdaˀ               # ;
tehtǫˀ                     # ;
jǫˀdagaˀ                   # ;
jino:wę:                     # ;
tea:Cǫt                    # ;
sa:no:ˀ                    # ;
drę:na:                    # ;
dre:na:                    # ;
joni:tsgrǫ:t             # ;
kdagǫˀ                     # ;
do:dihs                    # ;
```

```
sgwa:yęh              # ;
gwiyo:gę̃             # ;
jǫ:nyǫ:̃                    # ;
nǫhsodai:yǫ:             # ;
gwa̰da:                   # ;
jidę:̃ęh                  # ;
jǫ̰dae:ya:̃             # ;
jḭdrǫ:wę:             # ;
onohotsgę̰ę̃            # ;
teo:jḭ                  # ;
tsa̰gę:da̰             # ;
yahgęhda̰             # ;
tsinyohgwa:k             # ;
gihę:k                  # ;
nawę̰da̰            # ;
jihsǫ:dahk               # ;
otahyǫ:ni:               # ;
tahyǫ:ni:                  # ;
jihsgę:                  # ;
jḭo:                       # ;
grahe:t                  # ;


LEXICON                  deverbalNouns
o+                         deverbalORoots ;
a+                         deverbalARoots ;
ga+                          deverbalGaRoots ;
                         deverbalNullRoots ;


LEXICON                  deverbalNullRoots
ędehsr                         NSF ;
ę̰nyotr                        NSF ;
ę̰nhotr                        NSF ;


LEXICON                  deverbalGaRoots
idehsra                        NSF ;
yęnawahsr                    NSF ;
ya̰dowehdahsr              NSF ;
ya̰dagenhahsr              NSF ;
atgwęnya̰tr                  NSF ;
atgǫnya̰tr                 NSF ;
tgḭtr                        NSF ;
nohǫkdehsr                  NSF ;
nhehsr                         NSF ;
```

```
naˀjowiˀtr                    NSF ;
risr                           NSF ;
rihwiyohsdęhsr               NSF ;
rihwaneˀaksra               NSF ;
rihoˀdęhsr                     NSF ;
hyadǫhsr                     NSF ;
hshahsdęhsr                NSF ;
Cahǫhsr                      NSF ;


LEXICON                      deverbalORoots
yęhsr                      NSF ;
atgahnǫnihsr               NSF ;
nrahdǫdahsr                NSF ;
nǫnheˀdr                     NSF ;
niga:hęhsr                     NSF ;
hshahsdęhsr                NSF ;
adǫtgadǫhsr                NSF ;
adǫtgadehsr                NSF ;
ˀdrohsr                          NSF ;
iˀdaihęhdr                     NSF ;


LEXICON                      deverbalARoots
atshoˀkdǫhsr               NSF ;
atnaˀtsotr                     NSF ;
atnaˀgwihdr                NSF ;
atgahnyehtr                NSF ;
nahaotr                          NSF ;
agyaˀdawiˀtr               NSF ;
adrihwagyaǫhsr               NSF ;
adraˀwihsd                     NSF ;
adǫnhehsr                    NSF ;
adiˀgrǫhsra                    NSF ;
adęnaˀtr                     NSF ;
adekwahahsra               NSF ;
adaoˀtra                         NSF ;
ahdahdiˀtr                     NSF ;


LEXICON                      inalienableNouns
@U.INALIEN.POSS@                          inalienablePrefixes ;
@U.INALIEN.UNPOSS@o+               inalienableStems ; ! Unpossessed inalienables
 end with the NSF
```

```
LEXICON                    inalienablePrefixes
g+                           inalienableStems ; ! 1S
(e)tni+                    inalienableStems ; ! 1ID       (This and the following
                 are interchangeable)
(e)kni+                    inalienableStems ; ! 1ID       (This and the preceding
                are interchangeable)
(y)akni+                    inalienableStems ; ! 1ED
(e)dwa+                    inalienableStems ; ! 1IPL
(y)agwa+                inalienableStems ; ! 1EPL
(                           inalienablePrefixes2 ;
ha+                          inalienableStems ; ! 3MS
(y)                          inalienable3FIPrefixes ;
ga+                          inalienableICStems ; ! 3N       (For I-stems and C-stems)
(y)+                          inalienableOǫStems ; ! 3N       (For o-stems)
w+                           inalienableNotIOǫStems ; ! 3N       (For other stems)
hadi+                    inalienableStems ; ! 3MPL
gaǫ+                    inalienableAStems ; ! 3FIPL       (For a-stems)
gae+                    inalienableICStems ; ! 3FIPL       (For i-stems and C-Stems)
ga:g+                    inalienableNotAIStems ; ! 3FIPL       (For other stems)
gadi+                    inalienableStems ; ! 3NPL


LEXICON                    inalienablePrefixes2
h)s+                           inalienableVStems ; ! 2S
h)sni+                    inalienableVStems ; ! 2D
h)swa+                    inalienableVStems ; ! 2P
eh)s+                    inalienableCStems ; ! 2S
eh)sni+                    inalienableCStems ; ! 2D
eh)swa+                    inalienableCStems ; ! 2P


LEXICON                    inalienable3FIPrefixes
ǫ+                           inalienableAStems ; ! 3FI       (For a-stems)
ę+                           inalienableIStems ; ! 3FI       (For i-stems)
ag+                           inalienableNotAIStems ; ! 3FI       (For other V-stems)
e+                           inalienableCStems ; ! (For C-stems)


LEXICON                    inalienableICStems
                           inalienableIStems ; ! 3FIPL (For i-stems)
                           inalienableCStems ;  ! 3FIPL (For C-stems)


LEXICON                    inalienableVStems
                           inalienableAStems ;
                           inalienableIStems ;
```

```
!                          inalienableEStems ;
!                          inalienableęStems ;
!                          inalienableOStems ;
                          inalienableǫStems ;


LEXICON                   inalienableStems
                          inalienableAStems ;
                          inalienableIStems ;
                          inalienableNotAIStems ;
                          inalienableCStems ;

LEXICON                   inalienableNotAIStems
!                          inalienableEStems ;
!                          inalienableęStems ;
!                          inalienableOStems ;
                          inalienableǫStems ;

LEXICON                   inalienableNotIOǫStems
                          inalienableAStems ;
!                          inalienableEStems ;
!                          inalienableęStems ;


LEXICON                   inalienableOǫStems
!                          inalienableOStems ;
                          inalienableǫStems ;


LEXICON                     inalienableAStems
ahǫhd                     inalienableSuffix ;
ahsi˺d                    inalienableSuffix ;
ahyagwiy              inalienableSuffix ;


LEXICON                     inalienableIStems
ihn                           inalienableSuffix ;


!LEXICON                    inalienableEStems


!LEXICON                    inalienableęStems


!LEXICON                    inalienableOStems
```

```
LEXICON                    inalienableǫStems
ǫts                          inalienableSuffix ;


LEXICON                    inalienableCStems
wę̓nahs              inalienableSuffix ;
wę̓nohs              inalienableSuffix ;
gah                          inalienableSuffix ;
gahehd                     inalienableSuffix ;
gahgwaohs                inalienableSuffix ;
gǫhs                       inalienableSuffix ;
gǫhstǫ̓                     inalienableSuffix ;
gǫ̓d                        inalienableSuffix ;
gę̓sd                       inalienableSuffix ;
ha̓d                        inalienableSuffix ;
han                          inalienableSuffix ;
hdega:                     inalienableSuffix ;
hetga̓                      inalienableSuffix ;
hna̓ts                      inalienableSuffix ;
hnęs                       inalienableSuffix ;
hnya̓s                      inalienableSuffix ;
hnyędahs                 inalienableSuffix ;
hsgwa:                       inalienableSuffix ;
hsin                       inalienableSuffix ;
hsǫhga:                      inalienableSuffix ;
hsna̓d                      inalienableSuffix ;
hsohd                      inalienableSuffix ;
hsohgw                       inalienableSuffix ;
hswa̓n                      inalienableSuffix ;
hswe̓n                      inalienableSuffix ;
hyohs                      inalienableSuffix ;
̓nhǫhsga:                inalienableSuffix ;
̓nyǫhs                       inalienableSuffix ;
̓yohgw                       inalienableSuffix ;
̓ahs                       inalienableSuffix ;
kse̓d                       inalienableSuffix ;
nǫ̓a:                       inalienableSuffix ;
nǫ̓gw                       inalienableSuffix ;
nǫnhe̓dr                  inalienableSuffix ;
no̓j                        inalienableSuffix ;
nr                               inalienableSuffix ;
nętsh                      inalienableSuffix ;
nyęd                       inalienableSuffix ;
```

```
rad                          inalienableSuffix ;
ragwahd                 inalienableSuffix ;
wɋˀyǫhga:                inalienableSuffix ;
jaohoˀgw                inalienableSuffix ;
jiˀohd                       inalienableSuffix ;
jiˀehd                       inalienableSuffix ;
jisgoˀgw                inalienableSuffix ;
yaˀd                       inalienableSuffix ;
yaˀga:                       inalienableSuffix ;
yoˀd                       inalienableSuffix ;
yoˀgw                       inalienableSuffix ;
yoˀts                       inalienableSuffix ;
yuˀts                       inalienableSuffix ;


LEXICON                      inalienableSuffix
@U.INALIEN.POSS@                            locativeSuffix ;
@U.INALIEN.UNPOSS@                            NSF ;


LEXICON                      locativeSuffix
+aˀgeh                      # ;


LEXICON                      basicNouns
                             basicNounUnpossessedPrefixes ;
                             basicNounPossessedPrefixes ;


LEXICON                      basicNounUnpossessedPrefixes
ga+                             gaBNouns ;
o+                             oBNouns ;
a+                             aBNouns ;
                             nullBNouns ;

LEXICON                      basicNounPossessedPrefixes
(w)ag+                      allBasicNounStems ; ! 1st person singular
(y)ǫkni+                      allBasicNounStems ; ! 1st person dual
(y)ǫgwa+                allBasicNounStems ; ! 1st person plural
sa+                            allBasicNounStems ; ! 2nd person singular
sni+                            allBasicNounStems ; ! 2nd person dual
swa+                       allBasicNounStems ; ! 2nd person plural
ho+                            allBasicNounStems ; ! 3rd person masculine singular
(ya)go+                      allBasicNounStems ; ! 3rd person feminine indefinite?
(y)o+                       allBasicNounStems ; ! 3rd person neuter
hodi+                       allBasicNounStems ; ! 3rd person masculine plural
```

```
(ya)godi+              allBasicNounStems ; ! 3rd person feminine indefinite plural
(y)odi+                     allBasicNounStems ; ! 3rd person neuter plural


LEXICON                      allBasicNounStems
                            gaBNouns ;
                            oBNouns ;
                            aBNouns ;
                            nullBNouns ;


LEXICON                     gaBNouns
                            normalGaBNouns ;
                            gaoBNouns ;


LEXICON                     oBNouns
                            normalOBNouns ;
                            gaoBNouns ;
                            aoBNouns;
                            nullBNouns;
                            oLoanWords;


LEXICON                     aBNouns
                            normalABNouns ;
                            aoBNouns ;


LEXICON                     oLoanWords
di:                            # ;
ji:s                            # ;


LEXICON                     nullBNouns
hǫnaˀd                    NSF ;
hsgwaęˀd                    NSF ;

LEXICON                     gaoBNouns
ˀyohgw                    NSF ;
ˀwahsd                    NSF ;
hehn                     NSF ;
hnyędahs                    NSF ;
hǫˀjihsd                    NSF ;
hsdagw                    NSF ;
jiˀgw                    NSF ;
```

```
jihoha:                    NSF ;


LEXICON                    aoBNouns
adǫhneˀts              NSF ;


LEXICON                    normalABNouns
adęhę                    NSF ;
adehsw                    NSF ;
adęnaˀtr                    NSF ;
adoˀjin                    NSF ;
adoda:                    NSF ;
adogę                    NSF ;
adowadǫ:              NSF ;
adraˀsw                    NSF ;
ahdahgw                    NSF ;
ahgwęny                    NSF ;
ahsgw                    NSF ;
atrǫniˀd                    NSF ;
atsogę                    NSF ;
awęhę                    NSF ;
awęnohgr              NSF ;


LEXICON                    normalOBNouns
ˀdǫdr                    NSF ;
ˀga:                         NSF ;
ˀgęhę                    NSF ;
ˀgr                         NSF ;
ˀnehs                    NSF ;
ˀnest                    NSF ;
ˀnost                    NSF ;
ˀnhahgy              NSF ;
nhahd                    NSF ;
ˀnhęhts                    NSF ;
ˀnhwęhts                    NSF ;
ˀnhǫhs                    NSF ;
ˀnǫhs                    NSF ;
ˀnihsda:                    NSF ;
ˀnhǫhd                    NSF ;
ˀnhǫhs                    NSF ;
ˀǫhgwa:                    NSF ;
ˀǫhs                         NSF ;
aˀęn                         NSF ;
adehshę                    NSF ;
```

adęnihs NSF ;
ahshed NSF ;
ahy NSF ;
ga: NSF ;
gaˀd NSF ;
gahdr NSF ;
gahehd NSF ;
gahgwaohs NSF ;
gahoˀj NSF ;
ganyęˀd NSF ;
geˀa: NSF ;
gǫˀdr NSF ;
gwiy NSF ;
haˀd NSF ;
hah NSF ;
hakd NSF ;
heˀa: NSF ;
hehd NSF ;
hęhda: NSF ;
hehs NSF ;
hets NSF ;
hey NSF ;
hikd NSF ;
hjiˀgr NSF ;
hn NSF ;
hnya: NSF ;
hnye:h NSF ;
hnyoˀgw NSF ;
hnyǫhs NSF ;
hǫd NSF ;
hodr NSF ;
hohsgr NSF ;
hohwa: NSF ;
hsa: NSF ;
hsaheˀd NSF ;
hsda: NSF ;
hsdai NSF ;
hsdaoˀgw NSF ;
hsęhę NSF ;
hsgęˀdr NSF ;
ahsgęˀdr NSF ;
hsgeh NSF ;
hsgoh NSF ;
hsgwiˀdr NSF ;
hsgyęˀda: NSF ;

hsgyǫ̃w NSF ;
hshe̜ NSF ;
hsiy NSF ;
hsna̜d NSF ;
hstǫdr NSF ;
hswę̜d NSF ;
hwahd NSF ;
hwę̜ga: NSF ;
hwęhda: NSF ;
hwęhsd NSF ;
i̜d NSF ;
i̜da: NSF ;
i̜dǫhgw NSF ;
ijǫ̃d NSF ;
jaǫs NSF ;
ji̜a: NSF ;
ji̜drǫwahd NSF ;
ji̜nǫw NSF ;
jigwęd NSF ;
jihgw NSF ;
jihsgw NSF ;
jihsǫda: NSF ;
jihwęd NSF ;
jike̜d NSF ;
jinǫhgr NSF ;
jitgwa: NSF ;
kd NSF ;
kdeh NSF ;
kjin NSF ;
kw NSF ;
na̜da: NSF ;
na̜ga: NSF ;
na̜gwiy NSF ;
na̜sgw NSF ;
nawad NSF ;
nę̜d NSF ;
ne̜da: NSF ;
negrę̜d NSF ;
negw NSF ;
nę̜hę̜: NSF ;
nę̜noga: NSF ;
nę̜nyo̜gw NSF ;
nǫ̃gw NSF ;
nǫgę̜d NSF ;
nǫhgwę NSF ;

```
nǫny                    NSF ;
nrahd                    NSF ;
nregę̃d                    NSF ;
nrębę                   NSF ;
nya̰gw                    NSF ;
nyah                    NSF ;
nyęd                    NSF ;
rihw                          NSF ;
sehd                          NSF ;
shaihsd                     NSF ;
tḛtr                          NSF ;
tgo̰d                   NSF ;
tragwę̃d                   NSF ;
trę̃d                   NSF ;
tsad                          NSF ;
tsehsd                    NSF ;
tsgę̃ę:                    NSF ;
tsgo̰d                   NSF ;
tsgr                          NSF ;
wa:                          NSF ;
wa̰wihsd                   NSF ;
wajihsd                     NSF ;
way                          NSF ;
hwę̃hga:                   NSF ;
węn                          NSF ;
widr                          NSF ;
widrębd                     NSF ;
wiy                          NSF ;
y                           NSF ;
ya̰d                          NSF ;
yad                          NSF ;
yahgw                   NSF ;
yan                          NSF ;
yę:                          NSF ;
yę̃gw                   NSF ;
yęd                          NSF ;
yębsa:                    NSF ;
yǫ̃d                          NSF ;
yo̰gw                   NSF ;
yǫw                          NSF ;


LEXICON                    normalGaBNouns
̰ahdr                   NSF ;
̰drehd                   NSF ;
```

ˀdroda NSF ;
ˀka: NSF ;
ˀnaˀgw NSF ;
ˀnigǫh NSF ;
ˀwahsha: NSF ;
Cagwa: NSF ;
Catsgę̱ˀd NSF ;
Cidrę̱hd NSF ;
Cihsd NSF ;
Cisr NSF ;
gaˀd NSF ;
gahihsd NSF ;
gahwehs NSF ;
gawehs NSF ;
gehd NSF ;
gǫˀdr NSF ;
gǫhs NSF ;
had NSF ;
hǫga: NSF ;
hǫw NSF ;
hsdow NSF ;
hsę̱n NSF ;
hsgwa: NSF ;
hsǫwahd NSF ;
itsga: NSF ;
ję̱ NSF ;
jihay NSF ;
jihsd NSF ;
jihw NSF ;
kw NSF ;
naˀj NSF ;
naˀjohsgw NSF ;
nad NSF ;
nahd NSF ;
nahgw NSF ;
nahsgw NSF ;
naiˀd NSF ;
nakd NSF ;
nehsda: NSF ;
nehw NSF ;
nheˀd NSF ;
nhy NSF ;
nǫhs NSF ;
nǫny NSF ;
now NSF ;

```
nyod                    NSF ;
rẹn                       NSF ;
rihwihs                  NSF ;
rọd                       NSF ;
tgẹhets                  NSF ;
tgwẹˀd                   NSF ;
tseˀd                    NSF ;
itseˀd                    NSF ;
tsenẹ                    NSF ;
ya:                       NSF ;
yaˀd                       NSF ;


LEXICON        NSF
+aˀ                    # ;
```

B.1.2 Abstract Semantic Lexicon

```
Multichar_Symbols @U.INALIEN.POSS@ @U.INALIEN.UNPOSS@


LEXICON                 ROOT
                        allNouns ;
                        allRoots ;
                        allSuffixes ;

LEXICON                 stemArchetypes
0 : a% a                                    #;
0 : i% i                                    #;
0 : e% e                                    #;
0 : ẹ% ẹ                                    #;
0 : o% o                                    #;
0 : ọ% ọ                                    #;
0 : ˀCV% ˀCV                        #;
0 : hV% hV                                  #;
0 : hCV% hCV                         #;
0 : g% C/nC/ˀCC/hCC/ˀV      #;
0 : nV% nV                                  #;
0 : r% r                                    #;
0 : y% y/w                                  #;


LEXICON                 allRoots
```

```
                            deverbalRoots ;
                            inalienableStems ;
                            allBasicNounStems ;


LEXICON                     allSuffixes
                            NSF ;
                            locativeSuffix ;


LEXICON                     deverbalRoots
                            deverbalORoots ;
                            deverbalARoots ;
                            deverbalGaRoots ;
                            deverbalNullRoots ;


LEXICON                     allNouns
                            inalienableNouns ;
                            basicNouns ;
3zn : 0                     deverbalNouns ;
                            defectiveNouns ;


LEXICON                     defectiveNouns
toad : sgwa:gwaǫdǫˀ                         # ;
cat : dago:s                                  # ;
cat : da:gu:s                                  # ;
chicken : dakshaeˀdohs                # ;
dog : so:wa:s                                  # ;
duck : twę:twę:t                               # ;
goose : hǫ:ga:k                               # ;
guinea% hen : dogę:t                        # ;
pig : gwihsgwihs                               # ;
rabbit : gwaˀyǫˀ                              # ;
turkey : sohǫ:t                               # ;
Baltimore% oriole : gyo:gyo:ˀ         # ;
blackbird : jogrihs                          # ;
black% breasted% woodpecker : gwidoˀgwidoˀ                          # ;
blue% jay : diˀdi:ˀ                                  # ;
chickadee : jikjiye:ˀ                        # ;
crow,% raven : gaˀga:ˀ                         # ;
great% horned% owl : hihi:                  # ;
high% soaring% hawk : gwiyęˀgwiyęˀ                          # ;
house% woodpecker : dihsdihs         # ;
hummingbird : jiˀnhǫwę:se:         # ;
```

killdeer : duwisduwi:˥ # ;
mockingbird,% chatterbox : sa˥sa˥ # ;
night% hawk : gwę:dihs # ;
partridge : gwe:sę˥ # ;
pigeon : tsahgo:wah # ;
robin : jihsgogo˥ # ;
screech% owl : gwaoh # ;
seagull : johwę˥sdaga˥ # ;
whip-poor-will : gwę˥gohnyę˥ # ;
bear : hnyagwai˥ # ;
eel : gǫ:deh # ;
Channel% catfish : tgwiyo:gę˥ # ;
ants : jinhǫhgwahęh # ;
bed% bug : ji˥nǫhdo:ya˥ # ;
butterfly% (something% is% wet%;% refers% to% the% transformation) : ji˥dana:wę: # ;
cricket : jinǫhsanǫh # ;
grasshopper : jihsda: # ;
spider : ji˥ao:yę: # ;
garter% snake : jinǫhyahae: # ;
buffalo : degriya˥gǫ˥ # ;
chipmunk% (refers% to% the% stripe% on% the% chipmunk's% back) : jihnyo˥gę˥ # ;
fox : hehshai: # ;
frog : sgwa˥ahda˥ # ;
ground% hog,% woodchuck,% gopher : tehtǫ˥ # ;
mink : jǫ˥daga˥ # ;
mouse : jino:wę: # ;
muskrat : tea:ǫt # ;
raccoon : sa:no:˥ # ;
skunk : drę:na: # ;
skunk : dre:na: # ;
squirrel : joni:tsgrǫ:t # ;
grey% squirrel,% black% squirrel : kdagǫ˥ # ;
salamander : do:dihs # ;
otter : sgwa:yęh # ;
barn% swallow : gwiyo:gę˥ # ;
bluebird : jǫ:nyǫ:˥ # ;
mud% puppies,% dogfish : nǫhsodai:yǫ: # ;
flying% squirrel : gwa˥da: # ;
bird : jidę:˥ęh # ;
raspberries : jǫ˥dae:ya:˥ # ;
sea% shell : ji˥drǫ:wę: # ;
beech : onohotsgę˥ę˥ # ;
iron% wood% (tree)%;% red% oak : teo:ji˥ # ;
corn% tassel : tsa˥gę:da˥ # ;
morel,% black% type% of% mushroom : yahgęhda˥ # ;

```
wild% walnut : tsinyohgwa:k       # ;
river,% stream,% creek : gihę:k       # ;
sugar : nawę̃da̰                              # ;
strawberry : jihsǫ:dahk              # ;
wolf : otahyǫ:ni:                              #;
wolf : tahyǫ:ni:                              # ;
a% ghost : jihsgę:                              # ;
a% crab : jḭo:                                  # ;
tree : grahe:t                             # ;


LEXICON                      deverbalNouns
P+ : o+                             deverbalORoots ;
A+ : a+                             deverbalARoots ;
A+ : ga+                             deverbalGaRoots ;
                                    deverbalNullRoots ;


LEXICON                      deverbalGaRoots
deverbal% noun% root : 0                                              # ;
sexuality : idehsra                                        NSF ;
help : yę̃nawahsr                                             NSF ;
the% ability% to% think%;% thinking% skills : ya̰dowehdahsr       NSF ;
helpfulness : ya̰dagenhahsr                                  NSF ;
corn% bread% paddles%;% corn% soup% paddles : atgwę̃nya̰tr       NSF ;
corn% bread% paddles%;% corn% soup% paddles : atgǫnya̰tr       NSF ;
junk : tgḭtr                                                NSF ;
sickness : nohǫkdehsr                                    NSF ;
to% take% someone's% part%;% advocacy : nhehsr                      NSF ;
water% drum : na̰jowḭtr                    NSF ;
leggings : risr                                           NSF ;
religion%;% the% Christian% faith : rihwiyohsdę̃hsr        NSF ;
sin : rihwanḛaksra                                       NSF ;
work : riho̰dę̃hsr                         NSF ;
paper : hyadǫhsr                                          SF ;
power,% strength : hshahsdę̃hsr                           NSF ;
cradleboard : Cahǫhsr                                     NSF ;


LEXICON                      deverbalORoots
deverbal% noun% root : 0                                              # ;
blankets : yę̃hsr         NSF ;
flint% (stone) : tragwę̰̃d                  NSF ;
wealth : atgahnǫnihsr                        NSF ;
poplar : nrahdǫdahsr                          NSF ;
soother,% pacifier,% nipple : nǫnhḛdr                  NSF ;
```

```
material,% cloth : niga:hęhsr                        NSF ;
power,% strength : hshahsdęhsr                    NSF ;
fun : adǫtgadǫhsr            NSF ;
celebration : adǫtgadehsr          NSF ;
fat,% pig% rinds : ˀdrohsr               NSF ;
sweat : iˀdaihęhdr              NSF ;


LEXICON                      deverbalARoots
deverbal% noun% root : 0                                          # ;
hoe : atshoˀkdǫhsr        NSF ;
pants : atnaˀtsotr                        NSF ;
belt : atnaˀgwihdr              NSF ;
sports,% games : atgahnyehtr                          NSF ;
hat : nahaotr                                                 NSF ;
coat,% dress : agyaˀdawiˀtr          NSF ;
disaster : adrihwagyaǫhsr                             NSF ;
bat% (mammal) : adraˀwihsd                            NSF ;
birth : adǫnhehsr                                             NSF ;
shyness : adiˀgrǫhsra                                                 NSF ;
lunch,% groceries : adęnaˀtr                                          NSF ;
table : adekwahahsra                                          NSF ;
friendship%;% also% refers% to% a% ceremonial% friend : adaoˀtra  NSF ;
socks : ahdahdiˀtr                            NSF ;


LEXICON                      deverbalNullRoots
mittens : ęˀnyotr                             NSF ;
ball : ęˀnhotr                                 NSF ;
sexuality : ędehsr                               NSF ;


LEXICON                      inalienableNouns
0:@U.INALIEN.POSS@                              inalienablePrefixes ;
0:@U.INALIEN.UNPOSS@o+              inalienableStems ;


LEXICON                      inalienablePrefixes
1sA+ : g+                                    inalienableStems ; ! 1S
1idA+ : (e)tni+                      inalienableStems ; ! 1ID
1idA+ : (e)kni+                      inalienableStems ; ! 1ID
1edA+ : (y)akni+                      inalienableStems ; ! 1ED
1ipA+ : (e)dwa+                      inalienableStems ; ! 1IPL
1epA+ : (y)agwa+                      inalienableStems ; ! 1EPL
(                                          inalienablePrefixes2 ;
3msA+ : ha+                             inalienableStems ; ! 3MS
```

```
3fiA+ : (y)                                        inalienable3FIPrefixes ;
3znsA+ : ga+                              inalienableICStems ; ! 3N
3znsA+ : (y)+                              inalienableOǫStems ; ! 3N
3znsA+ : w+                               inalienableNotIOǫStems ; ! 3N
3mdpA+ : hadi+                          inalienableStems ; ! 3MPL
3fidpA+ : gaǫ+                             inalienableAStems ; ! 3FIPL
3fidpA+ : gae+                             inalienableICStems ; ! 3FIPL
3fidpA+ : ga:g+                         inalienableNotAIStems ; ! 3FIPL
3zndpA+ : gadi+                         inalienableStems ; ! 3NPL


LEXICON                         inalienablePrefixes2
2sA+ : h)s+                                     inalienableVStems ; ! 2S
2dA+ : h)sni+                             inalienableVStems ; ! 2D
2pA+ : h)swa+                             inalienableVStems ; ! 2P
2sA+ : eh)s+                             inalienableCStems ; ! 2S
2dA+ : eh)sni+                          inalienableCStems ; ! 2D
2pA+ : eh)swa+                          inalienableCStems ; ! 2P


LEXICON                         inalienable3FIPrefixes
0 : ǫ+                                inalienableAStems ; ! 3FI
0 : ę+                                inalienableIStems ; ! 3FI
0 : ag+                                inalienableNotAIStems ; ! 3FI
0 : e+                                inalienableCStems ; ! (For% C-stems)


LEXICON                         inalienableICStems
                                inalienableIStems ; ! 3FIPL
                                inalienableCStems ; ! 3FIPL


LEXICON                         inalienableVStems
                                inalienableAStems ;
                                inalienableIStems ;
!                                inalienableEStems ;
!                                inalienableęStems ;
!                                inalienableOStems ;
                                inalienableǫStems ;


LEXICON                         inalienableStems
                                inalienableAStems ;
                                inalienableIStems ;
                                inalienableNotAIStems ;
                                inalienableCStems ;
```

```
LEXICON                     inalienableNotAIStems
!                            inalienableEStems ;
!                            inalienableɋStems ;
!                            inalienableOStems ;
                            inalienableǫStems ;

LEXICON                     inalienableNotIOǫStems
                            inalienableAStems ;
!                            inalienableEStems ;
!                            inalienableɋStems ;


LEXICON                     inalienableOǫStems
!                            inalienableOStems ;
                            inalienableǫStems ;


LEXICON                      inalienableAStems
                                        stemArchetypes ;
inalienable% noun% root% beginning% with% %+a : 0              # ;
on% your% ears : ahǫhd                                    inalienableSuffix ;
on% your% foot : ahsiˀd                                    inalienableSuffix ;
on% my% toes : ahyagwiy                               inalienableSuffix ;


LEXICON                     inalienableIStems
                                        stemArchetypes ;
inalienable% noun% root% beginning% with% %+i : 0            # ;
(on)% my% skin : ihn                                  inalienableSuffix ;


!LEXICON                     inalienableEStems


!LEXICON                     inalienableɋStems


!LEXICON                     inalienableOStems


LEXICON                     inalienableǫStems
                                        stemArchetypes ;
inalienable% noun% root% beginning% with% %+ǫ : 0                # ;
on% your% knee : ǫts                                  inalienableSuffix ;
```

```
LEXICON                        inalienableCStems
                                        stemArchetypes ;
inalienable% noun% root% beginning% with% a% consonant : 0              # ;
on% your% tongue : wę̨ˀnahs                        inalienableSuffix ;
on% your% tongue : wę̨ˀnohs                        inalienableSuffix ;
on% your% thumb : wę̨ˀyǫhga:                        inalienableSuffix ;
on% your% eyes : gah                                      inalienableSuffix ;
on% your% eyelashes : gahehd                       inalienableSuffix ;
on% your% eyebrow : gahgwaohs                      inalienableSuffix ;
on% your% hairline,% upper% brow%;% forehead : gę̨ˀsd               inalienableSuffix ;
on% the% bridge% of% my% nose : gǫˀd               inalienableSuffix ;
on% your% face : gǫhs                                     inalienableSuffix ;
on% its% whiskers : gǫhstǫˀ                               inalienableSuffix ;
on% its% throat : haˀd                                    inalienableSuffix ;
on% your% groin : han                                     inalienableSuffix ;
on% your% ribs : hdega:                                     inalienableSuffix ;
on% your% anus : hetgaˀ                                inalienableSuffix ;
on% your% buttocks : hnaˀts                               inalienableSuffix ;
on% your% shoulders : hnę̨s                inalienableSuffix ;
on% your% neck% (front% of% the% neck) : hnyaˀs    inalienableSuffix ;
on% its% beak : hnyę̨dahs                           inalienableSuffix ;
his% testicles : hsgwa:                                   inalienableSuffix ;
on% your% leg : hsin                                      inalienableSuffix ;
on% your% calf% (of% leg) : hsnaˀd                        inalienableSuffix ;
on% your% hand : hsohd                                 inalienableSuffix ;
on% your% upper% lip : hsǫhga:                         inalienableSuffix ;
on% your% lip : hsohgw                                       inalienableSuffix ;
on% your% upper% back : hswaˀn                         inalienableSuffix ;
on% your% upper% back : hsweˀn                         inalienableSuffix ;
on% your% (p)% elbows : hyohs                          inalienableSuffix ;
on% my% chest : ˀahs                                       inalienableSuffix ;
on% my% inner% thigh : ˀnhǫhsga:               inalienableSuffix ;
on% your% nose : ˀnyǫhs                                inalienableSuffix ;
on% its% tail% (pertaining% to% birds) : ˀyohgw      inalienableSuffix ;
on% your% belly : kseˀd                                     inalienableSuffix ;
on% your% arm : nę̨tsh                                     inalienableSuffix ;
on% your% head : nǫˀa:                                     inalienableSuffix ;
on% your% breast : nǫˀgw                                inalienableSuffix ;
on% your% teeth : noˀj                                      inalienableSuffix ;
on% your% nipples : nǫnheˀdr                          inalienableSuffix ;
on% his% penis,% phallus : nr                               inalienableSuffix ;
on% your% shin : nyę̨d                                     inalienableSuffix ;
on% your% heel : rad                                       inalienableSuffix ;
on% the% ball% of% my% foot : ragwahd                  inalienableSuffix ;
on% your% ankle : jaohoˀgw                                 inalienableSuffix ;
```

```
on% my% nail : jiˀehd                                    inalienableSuffix ;
on% my% nail : jiˀohd                                    inalienableSuffix ;
on% your% hip : jisgoˀgw                               inalienableSuffix ;
on% your% body : yaˀd                                    inalienableSuffix ;
on% your% waist : yaˀga:                               inalienableSuffix ;
on% your% gums : yoˀd                                    inalienableSuffix ;
on% your% cheeks : yoˀgw                               inalienableSuffix ;
on% your% chin : yoˀts                                     inalienableSuffix ;
on% your% chin : yuˀts                                     inalienableSuffix ;


LEXICON                    inalienableSuffix
@U.INALIEN.POSS@                        locativeSuffix ;
@U.INALIEN.UNPOSS@                        NSF ;


LEXICON                    locativeSuffix
+Loc : +aˀgeh                   # ;
                                      # ;


LEXICON                    basicNouns
                           basicNounUnpossessedPrefixes ;
                           basicNounPossessedPrefixes ;


LEXICON                    basicNounUnpossessedPrefixes
3znA+ : ga+                          gaBNouns ;
3znP+ : o+                          oBNouns ;
3znA+ : a+                          aBNouns ;
                                        nullBNouns ;

LEXICON                    basicNounPossessedPrefixes
1sP+ : (w)ag+                     allBasicNounStems ;
1dP+ : (y)ǫkni+                     allBasicNounStems ;
1pP+ : (y)ǫgwa+                allBasicNounStems ;
2sP+ : sa+                          allBasicNounStems ;
2dP+ : sni+                          allBasicNounStems ;
2pP+ : swa+                      allBasicNounStems ;
3msP+ : ho+                          allBasicNounStems ;
3fisP+ : (ya)go+                    allBasicNounStems ;
3znsP+ : (y)o+                     allBasicNounStems ;
3mdpP+ : hodi+                     allBasicNounStems ;
3fidpP+ : (ya)godi+               allBasicNounStems ;
3zndpP+ : (y)odi+                     allBasicNounStems ;
```

```
LEXICON                     allBasicNounStems
                           gaBNouns ;
                           oBNouns ;
                           aBNouns ;
                           nullBNouns ;


LEXICON                     gaBNouns
                           normalGaBNouns ;
                           gaoBNouns ;


LEXICON                     oBNouns
                           normalOBNouns ;
                           gaoBNouns ;
                           aoBNouns ;
                           nullBNouns ;
                           oLoanWords ;


LEXICON                     aBNouns
                           normalABNouns ;
                           aoBNouns ;


LEXICON                     oLoanWords
                                    stemArchetypes ;
basic% noun% root : 0                                    # ;
tea : di:                        # ;
cheese : ji:s                        # ;


LEXICON                     nullBNouns
                                    stemArchetypes ;
basic% noun% root : 0                                    # ;
potato : hǫnaˀd                 NSF ;
colts% foot : hsgwaęˀd                  NSF ;

LEXICON                     gaoBNouns
                                    stemArchetypes ;
basic% noun% root : 0                                    # ;
clothespin : ˀwahsd                 NSF ;
cargo%;% bundle%;% load : hehn                    NSF ;
beak : hnyędahs                 NSF ;
a% motor%;% engine : hǫˀjihsd                   NSF ;
dirty% clothes : hsdagw                  NSF ;
```

```
nakedness%;% nudity : jiˀgw                          NSF ;
straight% pin%;% pin%;% brooch%;% safety% pin : jihoha:                        NSF ;


LEXICON                      aoBNouns
                                          stemArchetypes ;
basic% noun% root : 0                                          # ;
ladder%;% stairs : adǫhneˀts               NSF ;


LEXICON                      normalABNouns
                                          stemArchetypes ;
basic% noun% root : 0                                          # ;
fence : adęhę                        NSF ;
blouse%;% middy : adehsw                          NSF ;
lunch%;% groceries : adęnaˀtr                         NSF ;
skate : adoˀjin                       NSF ;
bow% (as% in% bow% and% arrow) : adoda:                        NSF ;
axe%;% tomahawk : adogę                           NSF ;
hunt : adowadǫ:                NSF ;
luck : adraˀsw                        NSF ;
shoes : ahdahgw                        NSF ;
clothing%;% clothes : ahgwęny                         NSF ;
roof : ahsgw                         NSF ;
closthes : atrǫniˀd                          NSF ;
calendar : atsogę                         NSF ;
flower : awęhę                        NSF ;
weeds : awęnohgr                NSF ;


LEXICON                      normalOBNouns
                                          stemArchetypes ;
basic% noun% root : 0                                          # ;
it% is% fat%;% gristle%;% rind : ˀdǫdr                         NSF ;
a% parable%;% tale%;% story%;% legend : ˀga:                                NSF ;
ashes%;% bullet%;% dust : ˀgęhę                        NSF ;
snow%;% snowflake : ˀgr                               NSF ;
sand : ˀnehs                         NSF ;
nudity : ˀnest                         NSF ;
nudity : ˀnost                         NSF ;
lumber% logs% (large)%;% timber : ˀnhahgy                  NSF ;
lumber% logs% (large)%;% timber : nhahd                         NSF ;
tail% of% an% animal : ˀnhęhts                        NSF ;
tail% of% an% animal : ˀnhwęhts                        NSF ;
eggs : ˀnhǫhs                         NSF ;
stem%;% hull% of% berries : ˀnihsda:                          NSF ;
```

```
bur : ˥nhǫhd                        NSF ;
onions : ˥nǫhs                       NSF ;
sod%;% moss : ˥ǫhgwa:                        NSF ;
vines : ˥ǫhs                              NSF ;
skirt%;% tail%;% feather : ˥yohgw                          NSF ;
snowsnake%;% pole : a˥ęn                                NSF ;
cocoon%;% nest%;% hive%;% bee-hive : adehshę                              NSF ;
wall : adęnihs                        NSF ;
number : ahshed                         NSF ;
fruit : ahy                               NSF ;
a% price% (on% it) : ga:                                  NSF ;
pants : ga˥d                        NSF ;
a% tear% (in% one's% eye) : gahdr                           NSF ;
eyelash%;% the% stem% of% a% berry%;% the% eye% of% the% corn% kernel : gahehd  NSF ;
eyebrow : gahgwaohs                 NSF ;
grass : gaho˥j                        NSF ;
cadaver%;% dead% body : ganyę˥d                          NSF ;
hair%;% a% rag%;% (it% is)% ragged%;% tattered : ge˥a:                    NSF ;
cotton% batting%;% q-tips : gǫ˥dr                          NSF ;
a% limb%;% twig%;% branch : gwiy                                  NSF ;
quill%;% plume%;% feather%;% voice%;% throat%;% larynx%;% esophagus : ha˥d
road : hah                              NSF ;
soot : hakd                         NSF ;
corn% husk : he˥a:                          NSF ;
dirt%;% earth%;% ground%;% land : hehd                          NSF ;
fur : hęhda:                          NSF ;
decayed% tree%;% log%;% wood%;% board : hehs                                  NSF ;
(raw)% sausage%;% bologna%;% wieners : hets                                 NSF ;
one% corn% stalk : hey                                  NSF ;
thorn%;% thistle : hikd                                  NSF ;
cloud : hji˥gr                        NSF ;
grease%;% oil : hn                                    NSF ;
nutmeat : hnya:                       NSF ;
flint% corn% soup : hnye:h                           NSF ;
nut : hnyo˥gw                         NSF ;
squash%;% melon : hnyǫhs                            NSF ;
a% bush%;% a% whip : hǫd                                  NSF ;
basswood : hodr                                 NSF ;
slippery% elm : hohsgr                          NSF ;
pelt : hohwa:                         NSF ;
mouth : hsa:                                  NSF ;
beans : hsahe˥d                       NSF ;
rain : hsda:                          NSF ;
scale% (of% a% fish) : hsda:                          NSF ;
scale% (of% a% fish) : hsdai                          NSF ;
```

necklace : hsdaoˀgw NSF ;
frost : hsęhę NSF ;
rust : hsgęˀdr NSF ;
rust : ahsgęˀdr NSF ;
louse : hsgeh NSF ;
branch : hsgoh NSF ;
wrinkles : hsgwiˀdr NSF ;
bone%;% bare% bones : hsgyęˀda: NSF ;
blue% beech% (tree) : hsgyǫˀw NSF ;
dough : hsheˀ NSF ;
thread%;% string%;% cord : hsiy NSF ;
hamstrings%;% calves% (of% the% legs)%;% outer% thighs : hsnaˀd NSF ;
straw : hstǫdr NSF ;
coal : hswęˀd NSF ;
maple : hwahd NSF ;
a% splint : hwęˀga: NSF ;
corn% ears : hwęhda: NSF ;
foam : hwęhsd NSF ;
feces%;% shit%;% excrement : iˀd NSF ;
clay%;% mud%;% mortar : iˀda: NSF ;
flame : iˀdǫhgw NSF ;
fish : ijǫˀd NSF ;
leaves% of% corn : jaǫs NSF ;
curtains%;% lace : jiˀa: NSF ;
the% brain : jiˀdrǫwahd NSF ;
bug%;% insect%;% worm : jiˀnǫw NSF ;
gonorrhea : jigwęd NSF ;
porridge%;% mush : jihgw NSF ;
mush : jihsgw NSF ;
cluster% of% stars%;% star : jihsǫda: NSF ;
bell : jihwęd NSF ;
salt : jikeˀd NSF ;
nasal% mucous : jinǫhgr NSF ;
yellow : jitgwa: NSF ;
a% nutshell : kd NSF ;
root%;% edible% roots% (pepper% roots%;% turnips%;% carrots) : kdeh NSF ;
stump%;% knots% in% a% tree : kjin NSF ;
its% food : kw NSF ;
bread : naˀda: NSF ;
horns%;% antlers : naˀga: NSF ;
cotton% batting : naˀgwiy NSF ;
a% mattress : naˀsgw NSF ;
clay%;% plaster%;% white-wash : nawad NSF ;
evergreen%;% conifer : nęˀd NSF ;
roe% (fish% eggs) : neˀda: NSF ;

morel% mushroom : negrę́d NSF ;
peas : negw NSF ;
corn : nę́hę́: NSF ;
hickory% wood%;% stick : nę́noga: NSF ;
pills : nę́nyoˀgw NSF ;
milk : nǫˀgw NSF ;
catfish : nǫgę́ˀd NSF ;
corn% cob : nǫhgwę́ NSF ;
a% husk : nǫny NSF ;
leaf : nrahd NSF ;
tripe% (cow% stomach% lining)%;% animal% stomache : nregę́ˀd NSF ;
tapeworm : nrę́hę́ NSF ;
vomit%;% vomitus : nyaˀgw NSF ;
native% mush% dishes% made% with% corn : nyah NSF ;
stem : nyę́d NSF ;
message%;% it% matters%;% it% is% its% fault%;% word%;% affair%;% business : rihw NSF ;
willow%;% nape% of% neck : sehd NSF ;
snake : shaihsd NSF ;
flour%;% powder : teˀtr NSF ;
sumac : tgoˀd NSF ;
housefly%;% fly : trę́ˀd NSF ;
mist%;% steam%;% fog : tsad NSF ;
syrup%;% honey%;% gum : tsehsd NSF ;
peach% pit : tsgę́ˀę́: NSF ;
balsam% fir : tsgoˀd NSF ;
saliva%;% spit%;% sputum : tsgr NSF ;
air%;% wind%;% a% moth : wa: NSF ;
a% peeling : waˀwihsd NSF ;
peelings%;% bark% of% a% tree : wajihsd NSF ;
fin% of% a% fish%;% wings : way NSF ;
wood% chips : hwę́ˀhga: NSF ;
word%;% voice%;% speech : wę́n NSF ;
ice : widr NSF ;
sleep%;% a% dream : widrę́hd NSF ;
young%;% offspring% (i.e.% of% an% animal)%;% baby : wiy NSF ;
other%;% another : y NSF ;
body : yaˀd NSF ;
basement%;% track%;% ditch : yad NSF ;
pants : yahgw NSF ;
tire%;% its% track%;% anything% that% leaves% tracks : yan NSF ;
beads : yę́: NSF ;
tobacco%;% cigarettes : yę́ˀgw NSF ;
wood%;% firewood : yę́d NSF ;
bandage : yę́hsa: NSF ;
a% dead% body%;% cadaver : yǫˀd NSF ;

```
cheeks : yoˀgw                      NSF ;
guts%;% intestines : yǫw                              NSF ;


LEXICON                      normalGaBNouns
                                          stemArchetypes ;
basic% noun% root : 0                                          # ;
basket : ˀahdr                             NSF ;
car%;% truck%;% vehicle : ˀdrehd                                NSF ;
diaper : ˀdroda                             NSF ;
skirt%;% slip : ˀka:                              NSF ;
marriage : ˀnaˀgw                            NSF ;
the% mind : ˀnigǫh                             NSF ;
earrings : ˀwahsha:                             NSF ;
Avocet% blue% stocking% (bird) : ˀyohgw                  NSF;
a% celestial% orb% (ie.% the% sun%;% the% moon) : Cagwa:              NSF ;
a% rope : Catsgę̀ˀd                              NSF ;
tin%;% metal : Cihsd                               NSF ;
leggings : Cisr                             NSF ;
sleep%;% a% dream : Cidrę̀hd                                NSF ;
white% oak : gaˀd                             NSF ;
eye% glasses : gahihsd                                NSF ;
shovel : gahwehs                            NSF ;
paddle : gawehs                             NSF ;
tie%;% scarf : gehd                               NSF ;
pillow%;% cushion : gǫˀdr                                 NSF ;
the% mask : gǫhs                              NSF ;
forest%;% bush : had                                NSF ;
elm : hǫga:                                NSF ;
boat : hǫw                                NSF ;
headdress : hsdow                                 NSF ;
a% name : hsę̀n                                NSF ;
stone%;% rock%;% boulder%;% bullet : hsgwa:                        NSF ;
nails%;% wire%;% needle : hsǫwahd                               NSF ;
mattress%;% sleeping% mat : itsga:                               NSF ;
dish%;% plate%;% bowl : ję̀                                NSF ;
the% devil : jihay                                 NSF ;
lamp : jihsd                               NSF ;
hammer : jihw                               NSF ;
food : kw                                NSF ;
pail : naˀj                               NSF ;
cup : naˀjohsgw                                NSF ;
town%;% community : nad                                     NSF ;
comb : nahd                                NSF ;
bass% drum : nahgw                                  NSF ;
```

```
tame% animal%;% pet%;% domestic% animal : nahsgw                    NSF ;
a% peacock%;% bride%;% boastfulness : naiˀd                      NSF ;
bed : nakd                              NSF ;
a% board : nehsda:                              NSF ;
leather%;% hide : nehw                                  NSF ;
porcupine : nheˀd                               NSF ;
stick : nhy                              NSF ;
a% house : nǫhs                                 NSF ;
a% dance : nǫny                                 NSF ;
guitar%; string% instrument%;% (refers% to% round% back% of% a% turtle) : now NSF ;
spoon%;% canoe%;% birch% bark% canoe : nyod                                NSF ;
song : rę̨n                              NSF ;
an% agreement : rihwihs                                    NSF ;
log : rǫd                               NSF ;
a% handle : tgę̨hets                                    NSF ;
wallet%;% purse%;% pocketbook%;% suitcase : tgwę̨ˀd  NSF ;
bottle%;% jar : tseˀd                                   NSF ;
bottle%;% jar : itseˀd                                   NSF ;
one% animal%;% pet : tsenę̨ˀ                                    NSF ;
bag%;% mattress%;% tick%;% pouch% (ie.%
         a% mattress% bag% into% which% straw% is% stuffed) : ya:          NSF ;
doll : yaˀd                              NSF ;


LEXICON        NSF
+NSF : +aˀ                      # ;
                                     # ;
```

B.2 Concrete Version

B.2.1 Concrete Lexicon

```
Multichar_Symbols @U.INALIEN.POSS@ @U.INALIEN.UNPOSS@


LEXICON                     allNouns
                            inalienableNouns ;
                            allBasicNouns ;
                            deverbalNouns ;
                            defectiveNouns ;
```

```
LEXICON                    defectiveNouns
sgwa:gwaǫdǫ̃       # ;
dago:s                    # ;
da:gu:s                    # ;
dakshaeˀdohs       # ;
so:wa:s                    # ;
twę:twę:t              # ;
hǫ:ga:k                    # ;
dogę:t                    # ;
gwihsgwihs               # ;
gwaˀyǫ̃                    # ;
sohǫ:t                    # ;
gyo:gyo:̃              # ;
jogrihs                    # ;
gwidoˀgwidoˀ       # ;
diˀdi:̃                    # ;
jikjiye:̃                    # ;
gaˀga:̃                    # ;
hihi:                    # ;
gwiyęˀgwiyęˀ              # ;
dihsdihs                    # ;
jiˀnhǫwę:se:             # ;
duwisduwi:̃              # ;
saˀsaˀ                    # ;
gwę:dihs                    # ;
gwe:sę̃                    # ;
tsahgo:wah              # ;
jihsgogoˀ              # ;
gwaoh                    # ;
johwęˀsdagaˀ             # ;
gwęˀgohnyęˀ              # ;
hnyagwaiˀ              # ;
gǫ:deh                    # ;
tgwiyo:gę̃              # ;
jinhǫhgwahęh       # ;
jiˀnǫhdo:yaˀ              # ;
jiˀdana:wę:              # ;
jinǫhsanǫh              # ;
jihsda:                    # ;
jiˀao:yę:                    # ;
jinǫhyahae:             # ;
degriyaˀgǫ̃              # ;
jihnyoˀgę̃              # ;
hehshai:                    # ;
sgwaˀahdaˀ              # ;
```

```
tehtǫ̃                    # ;
jǫ̃daga̰                   # ;
jino:wę:                  # ;
tea:ǫt                    # ;
sa:no:̰                   # ;
drę:na:                   # ;
dre:na:                   # ;
joni:tsgrǫ:t              # ;
kdagǫ̃                    # ;
do:dihs                   # ;
sgwa:yęh                  # ;
gwiyo:gę̃                 # ;
jǫ:nyǫ:̰                  # ;
nǫhsodai:yǫ:              # ;
gwa̰da:                   # ;
jidę:̰ęh                  # ;
jǫ̃dae:ya:̰                # ;
jḭdrǫ:wę:                # ;
onohotsgę̃ę̃               # ;
teo:jḭ                   # ;
tsa̰gę:da̰                 # ;
yahgęhda̰                 # ;
tsinyohgwa:k              # ;
gihę:k                    # ;
nawę̃da̰                  # ;
jihsǫ:dahk                # ;
otahyǫ:ni:                # ;
tahyǫ:ni:                 # ;
jihsgę:                   # ;
jḭo:                     # ;
grahe:t                   # ;


LEXICON                   deverbalNouns
o+                          deverbalORoots ;
a+                          deverbalARoots ;
ga+                          deverbalGaRoots ;
                          deverbalNullRoots ;


LEXICON                   deverbalNullRoots
ędehsr                          NSF ;
ę̃nyotr                          NSF ;
ę̃nhotr                          NSF ;
```

```
LEXICON                    deverbalGaRoots
idehsra                           NSF ;
yęnawahsr                       NSF ;
yaˀdowehdahsr                 NSF ;
yaˀdagenhahsr                 NSF ;
atgwęnyaˀtr                     NSF ;
atgǫnyaˀtr                    NSF ;
tgiˀtr                           NSF ;
nohǫkdehsr                     NSF ;
nhehsr                            NSF ;
naˀjowiˀtr                     NSF ;
risr                              NSF ;
rihwiyohsdęhsr                NSF ;
rihwaneˀaksra                 NSF ;
rihoˀdęhsr                     NSF ;
hyadǫhsr                      NSF ;
hshahsdęhsr                 NSF ;
Cahǫhsr                       NSF ;


LEXICON                    deverbalORoots
yęhsr                       NSF ;
atgahnǫnihsr                NSF ;
nrahdǫdahsr                NSF ;
nǫnheˀdr                     NSF ;
niga:hęhsr                     NSF ;
hshahsdęhsr                NSF ;
adǫtgadǫhsr                NSF ;
adǫtgadehsr                NSF ;
ˀdrohsr                           NSF ;
iˀdaihęhdr                     NSF ;


LEXICON                    deverbalARoots
atshoˀkdǫhsr               NSF ;
atnaˀtsotr                      NSF ;
atnaˀgwihdr                     NSF ;
atgahnyehtr                     NSF ;
nahaotr                       NSF ;
agyaˀdawiˀtr               NSF ;
adrihwagyaǫhsr                NSF ;
adraˀwihsd                      NSF ;
adǫnhehsr                     NSF ;
adiˀgrǫhsra                     NSF ;
adęnaˀtr                      NSF ;
```

```
adekwahahsra              NSF ;
adaoˀtra                        NSF ;
ahdahdiˀtr                    NSF ;


LEXICON                     inalienableNouns
@U.INALIEN.POSS@                          inalienablePrefixes ;
@U.INALIEN.UNPOSS@o+                          inalienableStems ; !This is for all
                                        the unpossessed inalienables.
                                         These should get the NSF


LEXICON                     inalienablePrefixes
                            inalienableAPrefixes ;
                            inalienableIPrefixes ;
!                            inalienableEǫPrefixes ;
                            inalienableOǫPrefixes ;
                            inalienableHˀCPrefixes ;
                            inalienableHˀCCPrefixes ;
                            inalineableNPrefixes ;
                            inalienableYWRPrefixes ;
                            inalienableOtherCPrefixes ;
!                            inalienableUPrefixes ;


LEXICON                     inalienableAPrefixes
g+                            inalienableAStems ;
(e)gy+                      inalienableAStems ;
(y)agy+                      inalienableAStems ;
(e)dw+                      inalienableAStems ;
(y)agw+                      inalienableAStems ;
(h)s+                      inalienableAStems ;
(h)j+                      inalienableAStems ;
(h)sw+                      inalienableAStems ;
h+                             inalienableAStems ;
(y)ǫ+                      inalienableAStems ;
w+                             inalienableAStems ;
hǫn+                     inalienableAStems ;
gaǫ+                     inalienableAStems ;
gǫn+                     inalienableAStems ;


LEXICON                      inalienableIPrefixes
g+                            inalienableIStems ;
(e)kn+                      inalienableIStems ;
(e)tn+                      inalienableIStems ;
```

```
(y)akn+                    inalienableIStems ;
(e)dwę+                    inalienableIStems ;
(y)agwę+                inalienableIStems ;
(h)s+                     inalienableIStems ;
(h)sn+                     inalienableIStems ;
(h)swę+                    inalienableIStems ;
hę+                           inalienableIStems ;
(y)ę+                     inalienableIStems ;
gę+                           inalienableIStems ;
had+                     inalienableIStems ;
gae+                     inalienableIStems ;
gad+                     inalienableIStems ;


!LEXICON                    inalienableEęPrefixes
!g+
!(e)kn
!(e)tn+
!(y)akn+
!(e)dw+
!(y)agw+
!(h)s+
!(h)sn+
!(h)sw+
!h+
!(y)ag+
!w+
!hęn+
!ga:g+
!gęn+


LEXICON                     inalienableOǫPrefixes
g+                            inalienableOǫStems ;
(e)kn+                    inalienableOǫStems ;
(e)tn+                    inalienableOǫStems ;
(y)akn+                    inalienableOǫStems ;
(e)gy+                    inalienableOǫStems ;
(y)agy+                    inalienableOǫStems ;
(h)s+                     inalienableOǫStems ;
(h)sn+                     inalienableOǫStems ;
(h)j+                     inalienableOǫStems ;
h+                            inalienableOǫStems ;
(y)ag+                     inalienableOǫStems ;
(y)+                           inalienableOǫStems ;
```

```
hęn+                    inalienableOǫStems ;
ga:g+                    inalienableOǫStems ;
gęn+                    inalienableOǫStems ;


!LEXICON                    inalienableUPrefixes
!g+
!kn+, tn+
!akn+
!gy+
!agy+
!s+
!sn+
!j+
!h+
!ag+
!w+
!hęn+
!ga:g+
!gęn+


LEXICON                    inalienableH̃CPrefixes
k+                           inalienableH̃CVStems ;
(e)kni+                    inalienableH̃CVStems ;
(e)tni+                    inalienableH̃CVStems ;
(y)akni+                    inalienableH̃CVStems ;
(e)dwa+                    inalienableH̃CVStems ;
(y)agwa+              inalienableH̃CVStems ;
(eh)s+                     inalienableH̃CVStems ;
(eh)sni+                    inalienableH̃CVStems ;
(eh)swa+              inalienableH̃CVStems ;
ha+                          inalienableH̃CVStems ;
(y)e+                     inalienableH̃CVStems ;
ga+                          inalienableH̃CVStems ;
hadi+                     inalienableH̃CVStems ;
gae+                    inalienableH̃CVStems ;
gadi+                     inalienableH̃CVStems ;


LEXICON                    inalienableH̃CCPrefixes
ge+                           inalienableH̃CCStems ;
(e)kni+                    inalienableH̃CCStems ;
(e)tni+                    inalienableH̃CCStems ;
(y)akni+                    inalienableH̃CCStems ;
(e)dwa+                    inalienableH̃CCStems ;
```

```
(y)agwa+              inalienableH^CCStems ;
(eh)se+                     inalienableH^CCStems ;
(eh)sni+                     inalienableH^CCStems ;
(eh)swa+              inalienableH^CCStems ;
ha+                          inalienableH^CCStems ;
(y)e+                    inalienableH^CCStems ;
ga+                          inalienableH^CCStems ;
hadi+                    inalienableH^CCStems ;
gae+                    inalienableH^CCStems ;
gadi+                    inalienableH^CCStems ;


LEXICON                    inalineableNPrefixes
k+                           inalineableNStems ;
(e)kni+                    inalineableNStems ;
(e)tni+                    inalineableNStems ;
(y)akni+                    inalineableNStems ;
(e)dwa+                    inalineableNStems ;
(y)agwa+              inalineableNStems ;
(eh)s+                     inalineableNStems ;
(eh)sni+                     inalineableNStems ;
(eh)swa+              inalineableNStems ;
ha+                          inalineableNStems ;
(y)e+                    inalineableNStems ;
ga+                          inalineableNStems ;
hadi+                    inalineableNStems ;
gae+                    inalineableNStems ;
gadi+                    inalineableNStems ;


LEXICON                    inalienableYWRPrefixes
g+                           inalienableYWRStems ;
(e)kni+                    inalienableYWRStems ;
(e)tni+                    inalienableYWRStems ;
(y)akni+                    inalienableYWRStems ;
(e)dwa+                    inalienableYWRStems ;
(y)agwa+              inalienableYWRStems ;
(eh)s+                     inalienableYWRStems ;
(eh)sni+                     inalienableYWRStems ;
(eh)swa+              inalienableYWRStems ;
ha+                          inalienableYWRStems ;
(y)e+                    inalienableYWRStems ;
ga+                          inalienableYWRStems ;
hadi+                    inalienableYWRStems ;
gae+                    inalienableYWRStems ;
```

```
gadi+                    inalienableYWRStems ;


LEXICON                    inalienableOtherCPrefixes
ge+                           inalienableOtherCStems ;
(e)kni+                    inalienableOtherCStems ;
(e)tni+                    inalienableOtherCStems ;
(y)akni+                    inalienableOtherCStems ;
(e)dwa+                    inalienableOtherCStems ;
(y)agwa+              inalienableOtherCStems ;
(eh)se+                    inalienableOtherCStems ;
(eh)sni+                    inalienableOtherCStems ;
(eh)swa+              inalienableOtherCStems ;
ha+                           inalienableOtherCStems ;
(y)e+                     inalienableOtherCStems ;
ga+                           inalienableOtherCStems ;
hadi+                     inalienableOtherCStems ;
gae+                     inalienableOtherCStems ;
gadi+                     inalienableOtherCStems ;


LEXICON                    inalienableStems
                           inalienableAStems ;
                           inalienableIStems ;
                           inalienableOǫStems ;
                           inalienableH◌̓CVStems ;
                           inalienableH◌̓CCStems ;
                           inalineableNStems ;
                           inalienableYWRStems ;
                           inalienableOtherCStems ;


LEXICON                    inalienableAStems !3
ahǫhd                     inalienableSuffix ;
ahsi◌̓d                    inalienableSuffix ;
ahyagwiy              inalienableSuffix ;


LEXICON                    inalienableIStems !1
ihn                           inalienableSuffix ;


!LEXICON                    inalienableEęStems


LEXICON                    inalienableOǫStems !1
```

ǫts inalienableSuffix ;

LEXICON inalienableHˀCVStems !12
haˀd inalienableSuffix ;
han inalienableSuffix ;
hdega: inalienableSuffix ;
hetgaˀ inalienableSuffix ;
hnaˀts inalienableSuffix ;
hnęs inalienableSuffix ;
hsin inalienableSuffix ;
hsǫhga: inalienableSuffix ;
hsohd inalienableSuffix ;
hsohgw inalienableSuffix ;
hyohs inalienableSuffix ;
ˀyohgw inalienableSuffix ;

LEXICON inalienableHˀCCStems !8
hnyaˀs inalienableSuffix ;
hnyędahs inalienableSuffix ;
hsgwa: inalienableSuffix ;
ˀnhǫhsga: inalienableSuffix ;
ˀnyǫhs inalienableSuffix ;
hsnaˀd inalienableSuffix ;
hswaˀn inalienableSuffix ;
hsweˀn inalienableSuffix ;

LEXICON inalineableNStems !7
nǫˀa: inalienableSuffix ;
nǫˀgw inalienableSuffix ;
nǫnheˀdr inalienableSuffix ;
noˀj inalienableSuffix ;
nr inalienableSuffix ;
nętsh inalienableSuffix ;
nyęd inalienableSuffix ;

LEXICON inalienableYWRStems !10
yaˀd inalienableSuffix ;
yaˀga: inalienableSuffix ;
yoˀd inalienableSuffix ;
yoˀgw inalienableSuffix ;
yoˀts inalienableSuffix ;
yuˀts inalienableSuffix ;

```
wę̓nahs              inalienableSuffix ;
wę̓nohs              inalienableSuffix ;
wę̓yǫhga:              inalienableSuffix ;
rad                          inalienableSuffix ;
ragwahd              inalienableSuffix ;


LEXICON                    inalienableOtherCStems !14
gah                           inalienableSuffix ;
gahehd                     inalienableSuffix ;
gahgwaohs               inalienableSuffix ;
gǫhs                      inalienableSuffix ;
gǫhstǫ̓                     inalienableSuffix ;
gǫ̓d                       inalienableSuffix ;
gę̓sd                      inalienableSuffix ;
̓ahs                       inalienableSuffix ;
kse̓d                      inalienableSuffix ;
jaoho̓gw              inalienableSuffix ;
ji̓ohd                      inalienableSuffix ;
ji̓ehd                      inalienableSuffix ;
jisgo̓gw              inalienableSuffix ;


LEXICON                    inalienableSuffix
@U.INALIEN.POSS@                           locativeSuffix ;
@U.INALIEN.UNPOSS@                           NSF ;


LEXICON                    locativeSuffix
+a̓geh                     # ;


LEXICON                    allBasicNouns
                           basicNounPossessedPrefixes ;
                           basicNounUnpossessedPrefixes ;


LEXICON                    basicNounPossessedPrefixes
                           wagPrefixes ;
                           yǫkniPrefixes ;
                           yǫgwaPrefixes ;
                           saPrefixes ;
                           sniPrefixes ;
                           swaPrefixes ;
                           hoPrefixes ;
                           yagoPrefixes ;
```

```
                          yoPrefixes ;
                          hodiPrefixes ;
                          yagodiPrefixes ;
                          yodiPrefixes ;

LEXICON                    wagPrefixes
(w)ag+                    basicNounsAStems ;
(w)ag+                    basicNounsIStems ;
!                          basicNounsEStems ;
!                          basicNounsęStems ;
!                          basicNounsOStems ;
!                          basicNounsǫStems ;
(w)ak+                    basicNounshor˺CornStems ;
(w)ag+                    basicNounsyorwStems ;
(w)age+                    basicNounsCOtherStems ;

LEXICON                    yǫkniPrefixes
(y)ǫgy+                    basicNounsAStems ;
(y)ǫkn+                    basicNounsIStems ;
!(y)ǫkn+                    basicNounsEStems ;
!(y)ǫkn+                    basicNounsęStems ;
!(y)ǫkn+                    basicNounsOStems ;
!(y)ǫkn+                    basicNounsǫStems ;
(y)ǫkni+                    basicNounsOtherStems ;

LEXICON                          yǫgwaPrefixes
(y)ǫgw+                          basicNounsAStems ;
(y)ǫgwę+                  basicNounsIStems ;
!(y)ǫgw+                  basicNounsEStems ;
!(y)ǫgw+                  basicNounsęStems ;
!(y)ǫgy+                          basicNounsOStems ;
!(y)ǫgy+                          basicNounsǫStems ;
(y)ǫgwa+                  basicNounsOtherStems ;

LEXICON                          saPrefixes
s+                                 basicNounsAStems ;
sę+                                basicNounsIStems ;
!s+                                basicNounsEStems ;
!s+                                basicNounsęStems ;
!s+                                basicNounsOStems ;
!s+                                basicNounsǫStems ;
sa+                                basicNounsOtherStems ;
```

```
LEXICON                    sniPrefixes
j+                           basicNounsAStems ;
sn+                           basicNounsIStems ;
!sn+                           basicNounsEStems ;
!sn+                           basicNounsęStems ;
!sn+                           basicNounsOStems ;
!sn+                           basicNounsǫStems ;
sni+                           basicNounsOtherStems ;


LEXICON                    swaPrefixes
sw+                           basicNounsAStems ;
swę+                       basicNounsIStems ;
!sw+                           basicNounsEStems ;
!sw+                           basicNounsęStems ;
!j+                           basicNounsOStems ;
!j+                           basicNounsǫStems ;
swa+                      basicNounsOtherStems ;


LEXICON                    hoPrefixes
ho+                           basicNounsAStems ;
ho+                           basicNounsIStems ;
!haw+                      basicNounsEStems ;
!haw+                      basicNounsęStems ;
!h+                           basicNounsOStems ;
!h+                           basicNounsǫStems ;
ho+                           basicNounsOtherStems ;


LEXICON                    yagoPrefixes
(ya)go+                    basicNounsAStems ;
(ya)go+                    basicNounsIStems ;
!(ya)gaw+               basicNounsEStems ;
!(ya)gaw+               basicNounsęStems ;
!(ya)g+                    basicNounsOStems ;
!(ya)g+                    basicNounsǫStems ;
(ya)go+                    basicNounsOtherStems ;


LEXICON                    yoPrefixes
(y)o+                     basicNounsAStems ;
(y)o+                     basicNounsIStems ;
!(y)aw+                     basicNounsEStems ;
```

```
!(y)o+                    basicNounsęStems ;
!(y)+                           basicNounsOStems ;
!(y)+                           basicNounsǫStems ;
(y)o+                    basicNounsOtherStems ;


LEXICON                     hodiPrefixes
hon+                      basicNounsAStems ;
hod+                      basicNounsIStems ;
!hon+                      basicNounsEStems ;
!hon+                      basicNounsęStems ;
!hon+                      basicNounsOStems ;
!hon+                      basicNounsǫStems ;
hodi+                      basicNounsOtherStems ;


LEXICON                     yagodiPrefixes
(ya)gon+                basicNounsAStems ;
(ya)god+                basicNounsIStems ;
!(ya)gon+                basicNounsEStems ;
!(ya)gon+                basicNounsęStems ;
!(ya)gon+                basicNounsOStems ;
!(ya)gon+                basicNounsǫStems ;
(ya)godi+                basicNounsOtherStems ;


LEXICON                     yodiPrefixes
(y)on+                    basicNounsAStems ;
(y)od+                    basicNounsIStems ;
!(y)on+                    basicNounsEStems ;
!(y)on+                    basicNounsęStems ;
!(y)on+                    basicNounsOStems ;
!(y)on+                    basicNounsǫStems ;
(y)odi+                    basicNounsOtherStems ;


LEXICON                     basicNounUnpossessedPrefixes
ga+                           gaBNouns ;
o+                          oBNouns ;
a+                          aBNouns ;
                          nullBNouns ;


! Grouping by initial stem vowel for possessed basic nouns
LEXICON                     basicNounshor˺CornStems
                          nullBNounshVStems ;
```

```
                          gaohˀCNouns ;
                          normalOBNounshˀCnStems ;
                          normalGaBNounshˀCnStems ;


LEXICON                   basicNounsyorwStems
                          normalOBNounsYWStems ;
                          normalGaBNounsYWStems ;


LEXICON                   basicNounsCOtherStems
                          gaoJNouns ;
                          normalOBNounsOtherCStems ;
                          normalGaBNounsOtherCStems ;
                          oLoanWords ;
                          nullBNounshCStems ;


LEXICON                   basicNounsAStems
                          aoBNouns ;
                          normalABNouns ;
                          normalOBNounsAStems ;

LEXICON                   basicNounsIStems
                          normalOBNounsIStems ;
                          normalGaBNounsIStems ;

!LEXICON                   basicNounsEStems
!                          # ;

!LEXICON                   basicNounsęStems
!                          # ;


!LEXICON                   basicNounsǫStems
!                          # ;


LEXICON                   basicNounsOtherStems
                          oLoanWords ;
                          nullBNouns ;
                          gaoBNouns ;
                          normalOBNounsOtherStems ;
                          normalGaBNounsOtherStems ;

! End grouping by stem vowel for possessed basic nouns
```

```
LEXICON                    gaBNouns
                           normalGaBNouns ;
                           gaoBNouns ;

LEXICON                    oBNouns
                           normalOBNouns ;
                           gaoBNouns ;
                           aoBNouns;
                           nullBNouns;
                           oLoanWords;

LEXICON                    aBNouns
                           normalABNouns ;
                           aoBNouns ;

LEXICON                    oLoanWords
di:                           # ;
ji:s                           # ;

LEXICON                    nullBNouns
                           nullBNounshVStems ;
                           nullBNounshCStems ;

LEXICON                    nullBNounshVStems
hǫnaˀd                    NSF ;

LEXICON                    nullBNounshCStems
hsgwaęˀd             NSF ;

LEXICON                    gaoBNouns
                           gaohˀCNouns ;
                           gaoJNouns ;

LEXICON                    gaoJNouns
jiˀgw                     NSF ;
jihoha:                    NSF ;
hnyędahs                    NSF ;
```

```
hsdagw                      NSF ;


LEXICON                      gaohˀCNouns
ˀwahsd                     NSF ;
ˀyohgw                     NSF ;
hehn                      NSF ;
hǫˀjihsd                    NSF ;


LEXICON                      aoBNouns
adǫhneˀts               NSF ;


LEXICON                      normalABNouns
adęhę                     NSF ;
adehsw                     NSF ;
adęnaˀtr                    NSF ;
adoˀjin                    NSF ;
adoda:                     NSF ;
adogę                     NSF ;
adowadǫ:                NSF ;
adraˀsw                     NSF ;
ahdahgw                     NSF ;
ahgwęny                     NSF ;
ahsgw                     NSF ;
atrǫniˀd                    NSF ;
atsogę                    NSF ;
awęhę                     NSF ;
awęnohgr                NSF ;


LEXICON normalOBNouns
                            normalOBNounsAStems ;
                            normalOBNounsIStems ;
                            normalOBNounsOtherStems ;


LEXICON                      normalOBNounsOtherStems
                            normalOBNounshˀCnStems ;
                            normalOBNounsYWStems ;
                            normalOBNounsOtherCStems ;


LEXICON                      normalOBNounsOtherCStems
ˀǫhgwa:                     NSF ;
```

ˀǫhs NSF ;
ga: NSF ;
gaˀd NSF ;
gahdr NSF ;
gahehd NSF ;
gahgwaohs NSF ;
gahoˀj NSF ;
ganyęˀd NSF ;
geˀa: NSF ;
gǫˀdr NSF ;
gwiy NSF ;
jaǫs NSF ;
jiˀa: NSF ;
jiˀdrǫwahd NSF ;
jiˀnǫw NSF ;
jigwęd NSF ;
jihgw NSF ;
jihsgw NSF ;
jihsǫda: NSF ;
jihwęd NSF ;
jikeˀd NSF ;
jinǫhgr NSF ;
jitgwa: NSF ;
kd NSF ;
kdeh NSF ;
kjin NSF ;
kw NSF ;
sehd NSF ;
shaihsd NSF ;
teˀtr NSF ;
tgoˀd NSF ;
tragwęˀd NSF ;
tręˀd NSF ;
tsad NSF ;
tsehsd NSF ;
tsgęˀę: NSF ;
tsgoˀd NSF ;
tsgr NSF ;
normalOBNounshˀCCStems ;

LEXICON normalOBNounsYWStems
wa: NSF ;
waˀwihsd NSF ;
wajihsd NSF ;

```
way                          NSF ;
hwę̃hga:                NSF ;
wę̨n                       NSF ;
widr                        NSF ;
widrę̨hd                  NSF ;
wiy                          NSF ;
y                              NSF ;
yaˀd                       NSF ;
yad                          NSF ;
yahgw                       NSF ;
yan                          NSF ;
yę̨:                         NSF ;
yę̃gw                       NSF ;
yę̨d                         NSF ;
yę̨hsa:                     NSF ;
yǫ̃d                       NSF ;
yoˀgw                       NSF ;
yǫ̨w                       NSF ;
rihw                        NSF ;


LEXICON                        normalOBNounshˀCCStems
ˀnhahgy                  NSF ;
ˀnhę̨hts                    NSF ;
ˀnhwę̨hts                  NSF ;
ˀnhǫ̨hs                    NSF ;
ˀnhǫ̨hd                    NSF ;
ˀnǫ̨hs                     NSF ;
hnya:                       NSF ;
hnye:h                      NSF ;
hnyoˀgw                     NSF ;
hnyǫ̨hs                    NSF ;
hsda:                       NSF ;
hsdai                       NSF ;
hsdaoˀgw                    NSF ;
hsgę̃dr                     NSF ;
hsgeh                       NSF ;
hsgoh                       NSF ;
hsgwiˀdr                    NSF ;
hsgyę̃da:                   NSF ;
hsgyǫ̃w                    NSF ;
hsheˀ                       NSF ;
hsnaˀd                      NSF ;
hstǫ̨dr                     NSF ;
hswę̃d                      NSF ;
```

```
nrahd                    NSF ;
nregę̱ʔd                   NSF ;
nrę̱hę̱                    NSF ;
nyaʔgw                   NSF ;
nyah                    NSF ;
nyę̱d                    NSF ;
nhahd                    NSF ;
ʔgr                         NSF ;


LEXICON                    normalOBNounshʔCnStems
naʔda:                   NSF ;
naʔga:                   NSF ;
naʔgwiy                   NSF ;
naʔsgw                   NSF ;
nawad                   NSF ;
nę̱ʔd                         NSF ;
neʔda:                   NSF ;
negrę̱d                   NSF ;
negw                    NSF ;
nę̱hę̱:                    NSF ;
nę̱noga:                     NSF ;
nę̱nyoʔgw              NSF ;
nǫʔgw                    NSF ;
nǫgę̱ʔd                   NSF ;
nǫhgwę̱                   NSF ;
nǫny                    NSF ;
ʔdǫdr                    NSF ;
ʔnost                    NSF ;
ʔga:                          NSF ;
ʔgę̱hę̱                    NSF ;
ʔnehs                    NSF ;
ʔnest                    NSF ;
ʔnihsda:                    NSF ;
haʔd                         NSF ;
hah                         NSF ;
hakd                    NSF ;
heʔa:                    NSF ;
hehd                    NSF ;
hę̱hda:                     NSF ;
hehs                         NSF ;
hets                         NSF ;
hey                         NSF ;
hikd                         NSF ;
hjiʔgr                   NSF ;
```

```
hn                          NSF ;
hǫd                          NSF ;
hodr                          NSF ;
hohsgr                      NSF ;
hohwa:                      NSF ;
hsa:                          NSF ;
hsahe꞉d                     NSF ;
hsęhę                      NSF ;
hsiy                          NSF ;
hwahd                      NSF ;
hwę꞉ga:                     NSF ;
hwęhda:                     NSF ;
hwęhsd                     NSF ;


LEXICON                     normalOBNounsAStems
ahsgę꞉dr                    NSF ;
a꞉ęn                           NSF ;
adehshę                     NSF ;
adęnihs                     NSF ;
ahshed                     NSF ;
ahy                            NSF ;


LEXICON                     normalOBNounsIStems
i꞉d                           NSF ;
i꞉da:                          NSF ;
i꞉dǫhgw                    NSF ;
ijǫ꞉d                          NSF ;


LEXICON                     normalGaBNouns
                            normalGaBNounsIStems ;
                            normalGaBNounsOtherStems ;


LEXICON                     normalGaBNounsOtherStems
                            normalGaBNounsh꞉CnStems ;
                            normalGaBNounsYWStems ;
                            normalGaBNounsOtherCStems ;


LEXICON                     normalGaBNounsOtherCStems
꞉ahdr                      NSF ;
ga꞉d                           NSF ;
gahihsd                     NSF ;
```

```
gahwehs                    NSF ;
gawehs                    NSF ;
gehd                     NSF ;
gǫ̓dr                     NSF ;
gǫhs                          NSF ;
ję                          NSF ;
jihay                     NSF ;
jihsd                           NSF ;
jihw                          NSF ;
kw                          NSF ;
tgęhets                   NSF ;
tgwę̓d                    NSF ;
tse̓d                     NSF ;
tsenę                     NSF ;
                            normalGaBNounsh̓CCStems ;


LEXICON                     normalGaBNounsYWStems
ya:                            NSF ;
ya̓d                            NSF ;
Cagwa:                    NSF ;
Catsgę̓d                   NSF ;
Cidręhd                   NSF ;
Cihsd                     NSF ;
Cisr                           NSF ;
ręn                           NSF ;
rihwihs                   NSF ;
rǫd                            NSF ;


LEXICON                     normalGaBNounsh̓CCStems
hsdow                     NSF ;
̓drehd                    NSF ;
̓droda                    NSF ;
hsgwa:                    NSF ;
nhe̓d                     NSF ;
nhy                            NSF ;
nyod                     NSF ;


LEXICON                     normalGaBNounsh̓CnStems
had                            NSF ;
hǫga:                     NSF ;
hǫw                            NSF ;
hsęn                           NSF ;
hsǫwahd                   NSF ;
```

```
ˀka:                          NSF ;
ˀnaˀgw                      NSF ;
ˀnigǫh                      NSF ;
ˀwahsha:                      NSF ;
naˀj                           NSF ;
naˀjohsgw                NSF ;
nad                            NSF ;
nahd                        NSF ;
nahgw                       NSF ;
nahsgw                      NSF ;
naiˀd                       NSF ;
nakd                        NSF ;
nehsda:                        NSF ;
nehw                        NSF ;
nǫhs                           NSF ;
nǫny                        NSF ;
now                            NSF ;


LEXICON                       normalGaBNounsIStems
itseˀd                      NSF ;
itsga:                      NSF ;


LEXICON        NSF
+aˀ                      # ;
```

B.2.2 Concrete Semantic Lexicon

```
Multichar_Symbols @U.INALIEN.POSS@ @U.INALIEN.UNPOSS@


LEXICON                       root
                                        allNouns ;
                                        allRoots ;
                                        allSuffixes ;


LEXICON                       allSuffixes
                                        NSF ;
                                        locativeSuffix ;
```

```
LEXICON                    allRoots
                                        deverbalORoots ;
                                        deverbalARoots ;
                                        deverbalGaRoots ;
                                        deverbalNullRoots ;
                                        inalienableStems ;
                                        basicNounsAStems ;
                                        basicNounsIStems ;
!                                        basicNounsEStems ;
!                                        basicNounsǫStems ;
!                                        basicNounsOStems ;
!                                        basicNounsǫStems ;
                                       basicNounshor˥CornStems ;
                                       basicNounsyorwStems ;
                                       basicNounsCOtherStems ;


LEXICON                    allNouns
                                        inalienableNouns ;
                                        allBasicNouns ;
                                        deverbalNouns ;
                                        defectiveNouns ;


LEXICON                    defectiveNouns
toad : sgwa:gwaǫdǫ˥            # ;
cat : dago:s           # ;
cat : da:gu:s                                # ;
chicken : dakshae˥dohs                        # ;
dog : so:wa:s                          # ;
duck : twę:twę:t             # ;
goose : hǫ:ga:k                           # ;
guinea% hen : dogę:t                      # ;
pig : gwihsgwihs                           # ;
rabbit : gwa˥yǫ˥        # ;
turkey : sohǫ:t                         # ;
Baltimore% oriole : gyo:gyo:˥                    # ;
blackbird : jogrihs                  # ;
black% breasted% woodpecker : gwido˥gwido˥     # ;
blue% jay : di˥di:˥                                 # ;
chickadee : jikjiye:˥                                        # ;
crow,% raven : ga˥ga:˥                          # ;
great% horned% owl : hihi:                           # ;
high% soaring% hawk : gwiyę˥gwiyę˥        # ;
house% woodpecker : dihsdihs                    # ;
```

```
hummingbird : jiˀnhǫwę:se: # ;
killdeer : duwisduwi:ˀ                    # ;
mockingbird,% chatterbox : saˀsaˀ          # ;
night% hawk : gwę:dihs                          # ;
partridge : gwe:sęˀ                       # ;
pigeon : tsahgo:wah                                  # ;
robin : jihsgogoˀ                  # ;
screech% owl : gwaoh                          # ;
seagull : johwęˀsdagaˀ             # ;
whip-poor-will : gwęˀgohnyęˀ        # ;
bear : hnyagwaiˀ                           # ;
eel : gǫ:deh                                  # ;
Channel% catfish : tgwiyo:gęˀ                    # ;
ants : jinhǫhgwahęh                     # ;
bed% bug : jiˀnǫhdo:yaˀ              # ;
butterfly% (something% is% wet%;% refers% to% the% transformation) : jiˀdana:wę:   # ;
cricket : jinǫhsanǫh                    # ;
grasshopper : jihsda:                     # ;
spider : jiˀao:yę:                           # ;
garter% snake : jinǫhyahae:       # ;
buffalo : degriyaˀgǫˀ                     # ;
chipmunk% (refers% to% the% stripe% on% the% chipmunk's% back) : jihnyoˀgęˀ     # ;
fox : hehshai:                            # ;
frog : sgwaˀahdaˀ                      # ;
ground% hog,% woodchuck,% gopher : tehtǫˀ                      # ;
mink : jǫˀdagaˀ                           # ;
mouse : jino:wę:                            # ;
muskrat : tea:ǫt                            # ;
raccoon : sa:no:ˀ                           # ;
skunk : drę:na:                           # ;
skunk : dre:na:                           # ;
squirrel : joni:tsgrǫ:t              # ;
grey% squirrel,% black% squirrel : kdagǫˀ                      # ;
salamander : do:dihs                    # ;
otter : sgwa:yęh                            # ;
barn% swallow : gwiyo:gęˀ              # ;
bluebird : jǫ:nyǫ:ˀ                     # ;
mud% puppies,% dogfish : nǫhsodai:yǫ:                    # ;
flying% squirrel : gwaˀda:              # ;
bird : jidę:ˀęh                           # ;
raspberries : jǫˀdae:ya:ˀ              # ;
sea% shell : jiˀdrǫ:wę:                     # ;
beech : onohotsgęˀęˀ                     # ;
iron% wood% (tree)%;% red% oak : teo:jiˀ                      # ;
corn% tassel : tsaˀgę:daˀ              # ;
```

```
morel,% black% type% of% mushroom : yahgęhdaˀ                    # ;
wild% walnut : tsinyohgwa:k       # ;
river,% stream,% creek : gihę:k       # ;
sugar : nawęˀdaˀ                          # ;
strawberry : jihsǫ:dahk             # ;
wolf : otahyǫ:ni:                          #;
wolf : tahyǫ:ni:                          # ;
a% ghost : jihsgę:                          # ;
a% crab : jiˀo:                                 # ;
tree : grahe:t                           # ;


LEXICON                     deverbalNouns
3znP+ : o+                     deverbalORoots ;
3znA+ : a+                     deverbalARoots ;
3znA+ : ga+                     deverbalGaRoots ;
                                  deverbalNullRoots ;


LEXICON                     deverbalNullRoots
mittens : ęˀnyotr                                    NSF ;
ball : ęˀnhotr                                      NSF ;
sexuality : ędehsr                                     NSF ;


LEXICON                     deverbalGaRoots
0 : deverbal% noun% root                                  # ;
deverbal% noun% root : 0                                  # ;
sexuality : idehsra                                              NSF ;
help : yęnawahsr                                      NSF ;
the% ability% to% think%;% thinking% skills : yaˀdowehdahsr      NSF ;
helpfulness : yaˀdagenhahsr                               NSF ;
corn% bread% paddles%;% corn% soup% paddles : atgwęnyaˀtr     NSF ;
corn% bread% paddles%;% corn% soup% paddles : atgǫnyaˀtr       NSF ;
junk : tgiˀtr                                        NSF ;
sickness : nohǫkdehsr                                    NSF ;
to% take% someone's% part%;% advocacy : nhehsr                       NSF ;
water% drum : naˀjowiˀtr                     NSF ;
leggings : risr                                                NSF ;
religion%;% the% Christian% faith : rihwiyohsdęhsr              NSF ;
sin : rihwaneˀaksra                                             NSF ;
work : rihoˀdęhsr                                        NSF ;
paper : hyadǫhsr                                  NSF ;
power,% strength : hshahsdęhsr                        NSF ;
cradleboard : Cahǫhsr                                      NSF ;
```

```
LEXICON                     deverbalORoots
0 : deverbal% noun% root                                    # ;
deverbal% noun% root : 0                                    # ;
blankets : yęhsr          NSF ;
flint% (stone) : tragwę̃d                      NSF ;
wealth : atgahnǫnihsr                                          NSF ;
poplar : nrahdǫdahsr                                           NSF ;
soother,% pacifier,% nipple : nǫnhe̋dr           NSF ;
material,% cloth : niga:hęhsr                            NSF ;
power,% strength : hshahsdęhsr                          NSF ;
fun : adǫtgadǫhsr                   NSF ;
celebration : adǫtgadehsr                                    NSF ;
fat,% pig% rinds : ̋drohsr                                     NSF ;
sweat : i̋daihęhdr          NSF ;


LEXICON                     deverbalARoots
0 : deverbal% noun% root                                                    # ;
deverbal% noun% root : 0                                                    # ;
hoe : atshőkdǫhsr                            NSF ;
pants : atna̋tsotr                                    NSF ;
belt : atna̋gwihdr                                   NSF ;
sports,% games : atgahnyehtr                                          NSF ;
hat : nahaotr                                       NSF ;
coat,% dress : agya̋dawi̋tr            NSF ;
disaster : adrihwagyaǫhsr                     NSF ;
bat% (mammal) : adra̋wihsd                              NSF ;
birth : adǫnhehsr                                  NSF ;
shyness : adi̋grǫhsra               NSF ;
lunch,% groceries : adęna̋tr            NSF ;
table : adekwahahsra                                          NSF ;
friendship%;% also% refers% to% a% ceremonial% friend : adaőtra   NSF ;
socks : ahdahdi̋tr                                       NSF ;


LEXICON                     inalienableNouns
0 : @U.INALIEN.POSS@                           inalienablePrefixes ;
0 : @U.INALIEN.UNPOSS@o+                           inalienableStems ;


LEXICON                     inalienablePrefixes
                            inalienableAPrefixes ;
                            inalienableIPrefixes ;
!                            inalienableEęPrefixes ;
                            inalienableOǫPrefixes ;
                            inalienableH̋CPrefixes ;
```

```
                          inalienableHˀCCPrefixes ;
                          inalineableNPrefixes ;
                          inalienableYWRPrefixes ;
                          inalienableOtherCPrefixes ;
!                          inalienableUPrefixes ;


LEXICON                   inalienableAPrefixes
1sA+ : g+                          inalienableAStems ;
1idA+ : (e)gy+                    inalienableAStems ;
1edA+ : (y)agy+                    inalienableAStems ;
1ipA+ : (e)dw+                    inalienableAStems ;
1epA+ : (y)agw+                    inalienableAStems ;
2sA+ : (h)s+                     inalienableAStems ;
2dA+ : (h)j+                     inalienableAStems ;
2pA+ : (h)sw+                     inalienableAStems ;
3msA+ : h+                            inalienableAStems ;
3fisA+ : (y)ǫ+                    inalienableAStems ;
3znsA+ : w+                           inalienableAStems ;
3mdpA+ : hęn+                     inalienableAStems ;
3fidpA+ : gaǫ+                    inalienableAStems ;
3zndpA+ : gęn+                    inalienableAStems ;


LEXICON                   inalienableIPrefixes
1sA+ : g+                          inalienableIStems ;
1idA+ : (e)kn+                    inalienableIStems ;
1idA+ : (e)tn+                    inalienableIStems ;
1edA+ : (y)akn+                    inalienableIStems ;
1ipA+ : (e)dwę+                    inalienableIStems ;
1epA+ : (y)agwę+                inalienableIStems ;
2sA+ : (h)s+                     inalienableIStems ;
2dA+ : (h)sn+                     inalienableIStems ;
2pA+ : (h)swę+                    inalienableIStems ;
3msA+ : hę+                           inalienableIStems ;
3fisA+ : (y)ę+                    inalienableIStems ;
3znsA+ : gę+                           inalienableIStems ;
3mdpA+ : had+                     inalienableIStems ;
3fidpA+ : gae+                    inalienableIStems ;
3zndpA+ : gad+                    inalienableIStems ;


!LEXICON                   inalienableEęPrefixes
!1sA+ : g+
!1idA+ : (e)kn
!1idA+ : (e)tn+
```

```
!1edA+ : (y)akn+
!1ipA+ : (e)dw+
!1epA+ : (y)agw+
!2sA+ : (h)s+
!2dA+ : (h)sn+
!2pA+ : (h)sw+
!3msA+ : h+
!3fisA+ : (y)ag+
!3znsA+ : w+
!3mdpA+ : hǫn+
!3fidpA+ : ga:g+
!3zndpA+ : gǫn+


LEXICON                    inalienableOǫPrefixes
1sA+ : g+                           inalienableOǫStems ;
1idA+ : (e)kn+                    inalienableOǫStems ;
1idA+ : (e)tn+                    inalienableOǫStems ;
1edA+ : (y)akn+                    inalienableOǫStems ;
1ipA+ : (e)gy+                    inalienableOǫStems ;
1epA+ : (y)agy+                    inalienableOǫStems ;
2sA+ : (h)s+                    inalienableOǫStems ;
2dA+ : (h)sn+                    inalienableOǫStems ;
2pA+ : (h)j+                    inalienableOǫStems ;
3msA+ : h+                         inalienableOǫStems ;
3fisA+ : (y)ag+                    inalienableOǫStems ;
3znsA+ : (y)+                           inalienableOǫStems ;
3mdpA+ : hǫn+                    inalienableOǫStems ;
3fidpA+ : ga:g+                    inalienableOǫStems ;
3zndpA+ : gǫn+                    inalienableOǫStems ;


!LEXICON                    inalienableUPrefixes
!1sA+ : g+
!1idA+ : kn+
!1idA+ : tn+
!1edA+ : akn+
!1ipA+ : gy+
!1epA+ : agy+
!2sA+ : s+
!2dA+ : sn+
!2pA+ : j+
!3msA+ : h+
!3fisA+ : ag+
!3znsA+ : w+
```

```
!3mdpA+ : hęn+
!3fidpA+ : ga:g+
!3zndpA+ : gęn+


LEXICON                    inalienableH^CPrefixes
1sA+ : k+                                  inalienableH^CStems ;
1idA+ : (e)kni+                    inalienableH^CStems ;
1idA+ : (e)tni+                    inalienableH^CStems ;
1edA+ : (y)akni+             inalienableH^CStems ;
1ipA+ : (e)dwa+                    inalienableH^CStems ;
1epA+ : (y)agwa+             inalienableH^CStems ;
2sA+ : (eh)s+                           inalienableH^CStems ;
2dA+ : (eh)sni+                    inalienableH^CStems ;
2pA+ : (eh)swa+                    inalienableH^CStems ;
3msA+ : ha+                           inalienableH^CStems ;
3fisA+ : (y)e+                           inalienableH^CStems ;
3znsA+ : ga+                           inalienableH^CStems ;
3mdpA+ : hadi+                   inalienableH^CStems ;
3fidpA+ : gae+                   inalienableH^CStems ;
3zndpA+ : gadi+                    inalienableH^CStems ;


LEXICON                    inalienableH^CCPrefixes
ge+                           inalienableH^CCStems ;
(e)kni+                    inalienableH^CCStems ;
(e)tni+                    inalienableH^CCStems ;
(y)akni+                    inalienableH^CCStems ;
(e)dwa+                    inalienableH^CCStems ;
(y)agwa+             inalienableH^CCStems ;
(eh)se+                    inalienableH^CCStems ;
(eh)sni+                    inalienableH^CCStems ;
(eh)swa+             inalienableH^CCStems ;
ha+                           inalienableH^CCStems ;
(y)e+                    inalienableH^CCStems ;
ga+                           inalienableH^CCStems ;
hadi+                    inalienableH^CCStems ;
gae+                   inalienableH^CCStems ;
gadi+                    inalienableH^CCStems ;


LEXICON                    inalineableNPrefixes
1sA+ : k+                                  inalineableNStems ;
1idA+ : (e)kni+                    inalineableNStems ;
1idA+ : (e)tni+                    inalineableNStems ;
1edA+ : (y)akni+             inalineableNStems ;
```

```
1ipA+ : (e)dwa+                    inalineableNStems ;
1epA+ : (y)agwa+                 inalineableNStems ;
2sA+ : (eh)s+                          inalineableNStems ;
2dA+ : (eh)sni+                    inalineableNStems ;
2pA+ : (eh)swa+                    inalineableNStems ;
3msA+ : ha+                           inalineableNStems ;
3fisA+ : (y)e+                           inalineableNStems ;
3znsA+ : ga+                          inalineableNStems ;
3mdpA+ : hadi+                      inalineableNStems ;
3fidpA+ : gae+                      inalineableNStems ;
3zndpA+ : gadi+                    inalineableNStems ;


LEXICON                         inalienableYWRPrefixes
1sA+ : g+                                  inalienableYWRStems ;
1idA+ : (e)kni+                    inalienableYWRStems ;
1idA+ : (e)tni+                    inalienableYWRStems ;
1edA+ : (y)akni+                 inalienableYWRStems ;
1ipA+ : (e)dwa+                    inalienableYWRStems ;
1epA+ : (y)agwa+                 inalienableYWRStems ;
2sA+ : (eh)s+                          inalienableYWRStems ;
2dA+ : (eh)sni+                    inalienableYWRStems ;
2pA+ : (eh)swa+                    inalienableYWRStems ;
3msA+ : ha+                           inalienableYWRStems ;
3fisA+ : (y)e+                           inalienableYWRStems ;
3znsA+ : ga+                          inalienableYWRStems ;
3mdpA+ : hadi+                      inalienableYWRStems ;
3fidpA+ : gae+                      inalienableYWRStems ;
3zndpA+ : gadi+                    inalienableYWRStems ;


LEXICON                         inalienableOtherCPrefixes
1sA+ : ge+                               inalienableOtherCStems ;
1idA+ : (e)kni+                    inalienableOtherCStems ;
1idA+ : (e)tni+                    inalienableOtherCStems ;
1edA+ : (y)akni+                  inalienableOtherCStems ;
1ipA+ : (e)dwa+                    inalienableOtherCStems ;
1epA+ : (y)agwa+                 inalienableOtherCStems ;
2sA+ : (eh)se+                       inalienableOtherCStems ;
2dA+ : (eh)sni+                    inalienableOtherCStems ;
2pA+ : (eh)swa+                  inalienableOtherCStems ;
3msA+ : ha+                           inalienableOtherCStems ;
3fisA+ : (y)e+                       inalienableOtherCStems ;
3znsA+ : ga+                          inalienableOtherCStems ;
3mdpA+ : hadi+                      inalienableOtherCStems ;
```

```
3fidpA+ : gae+                    inalienableOtherCStems ;
3zndpA+ : gadi+                    inalienableOtherCStems ;


LEXICON                     inalienableStems
                            inalienableAStems ;
                            inalienableIStems ;
                            inalienableOǫStems ;
                            inalienableHˀCStems ;
                            inalienableHˀCCStems ;
                            inalineableNStems ;
                            inalienableYWRStems ;
                            inalienableOtherCStems ;


LEXICON                      inalienableAStems
0 : inalienable% noun% root% beginning% with% %+a                # ;
inalienable% noun% root% beginning% with% %+a : 0                # ;
on% your% ears : ahǫhd                                    inalienableSuffix ;
on% your% foot : ahsiˀd                                    inalienableSuffix ;
on% my% toes : ahyagwiy                                inalienableSuffix ;


LEXICON                      inalienableIStems
0 : inalienable% noun% root% beginning% with% %+i                 # ;
inalienable% noun% root% beginning% with% %+i : 0                 # ;
(on)% my% skin : ihn                                       inalienableSuffix ;


!LEXICON                      inalienableEǫStems


LEXICON                      inalienableOǫStems
0 : inalienable% noun% root% beginning% with% %+ǫ           # ;
inalienable% noun% root% beginning% with% %+ǫ : 0           # ;
on% your% knee : ǫts                                       inalienableSuffix ;


LEXICON                      inalienableHˀCStems !12
0 : inalienable% noun% root% beginning% with% %+h                # ;
0 : inalienable% noun% root% beginning% with% %+ˀC            # ;
inalienable% noun% root% beginning% with% %+h : 0                # ;
inalienable% noun% root% beginning% with% %+ˀC : 0        # ;
on% its% throat : haˀd                                     inalienableSuffix ;
on% your% groin : han                                      inalienableSuffix ;
on% your% ribs : hdega:                                     inalienableSuffix ;
```

```
on% your% anus : hetgaˀ                                    inalienableSuffix ;
on% your% buttocks : hnaˀts                               inalienableSuffix ;
on% your% shoulders : hnǫs                                inalienableSuffix ;
on% your% leg : hsin                                              inalienableSuffix ;
on% your% hand : hsohd                                    inalienableSuffix ;
on% your% upper% lip : hsǫhga:                           inalienableSuffix ;
on% your% lip : hsohgw                                    inalienableSuffix ;
on% your% (p)% elbows : hyohs                            inalienableSuffix ;
on% its% tail% (pertaining% to% birds) : ˀyohgw    inalienableSuffix ;


LEXICON                      inalineableNStems !7
0 : inalienable% noun% root% beginning% with% %+n                  # ;
inalienable% noun% root% beginning% with% %+n : 0                   # ;
on% your% arm : nǫtsh                                     inalienableSuffix ;
on% your% head : nǫˀa:                                    inalienableSuffix ;
on% your% breast : nǫˀgw                                 inalienableSuffix ;
on% your% teeth : noˀj                                    inalienableSuffix ;
on% your% nipples : nǫnheˀdr                             inalienableSuffix ;
on% his% penis,% phallus : nr                            inalienableSuffix ;
on% your% shin : nyǫd                                     inalienableSuffix ;


LEXICON                      inalienableYWRStems !10
inalienable% noun% root% beginning% with% %+y : 0              # ;
inalienable% noun% root% beginning% with% %+w : 0             # ;
inalienable% noun% root% beginning% with% %+r : 0              # ;
0 : inalienable% noun% root% beginning% with% %+y              # ;
0 : inalienable% noun% root% beginning% with% %+w             # ;
0 : inalienable% noun% root% beginning% with% %+r              # ;
on% your% body : yaˀd                                     inalienableSuffix ;
on% your% waist : yaˀga:                                     inalienableSuffix ;
on% your% gums : yoˀd                                     inalienableSuffix ;
on% your% cheeks : yoˀgw                                 inalienableSuffix ;
on% your% chin : yoˀts                                    inalienableSuffix ;
on% your% chin : yuˀts                                    inalienableSuffix ;
on% your% tongue : wǫˀnahs                               inalienableSuffix ;
on% your% tongue : wǫˀnohs                               inalienableSuffix ;
on% your% thumb : wǫˀyǫhga:                             inalienableSuffix ;
on% your% heel : rad                                             inalienableSuffix ;
on% the% ball% of% my% foot : ragwahd                    inalienableSuffix ;


LEXICON                      inalienableOtherCStems !14
inalienable% noun% root% beginning% with% %+g : 0                  # ;
inalienable% noun% root% beginning% with% %+ˀV : 0           # ;
```

```
inalienable% noun% root% beginning% with% %+k : 0                    # ;
inalienable% noun% root% beginning% with% %+w% : 0              # ;
inalienable% noun% root% beginning% with% %+j : 0               # ;
0 : inalienable% noun% root% beginning% with% %+g               # ;
0 : inalienable% noun% root% beginning% with% %+ˀV         # ;
0 : inalienable% noun% root% beginning% with% %+k                   # ;
0 : inalienable% noun% root% beginning% with% %+w               # ;
0 : inalienable% noun% root% beginning% with% %+j                   # ;
on% your% eyes : gah                                      inalienableSuffix ;
on% your% eyelashes : gahehd                          inalienableSuffix ;
on% your% eyebrow : gahgwaohs                          inalienableSuffix ;
on% your% hairline,% upper% brow%;% forehead : gǫˀsd         inalienableSuffix ;
on% the% bridge% of% my% nose : gǫˀd                 inalienableSuffix ;
on% your% face : gǫhs                                       inalienableSuffix ;
on% its% whiskers : gǫhstǫˀ                                inalienableSuffix ;
on% my% chest : ˀahs                                       inalienableSuffix ;
on% your% belly : kseˀd                                       inalienableSuffix ;
on% your% ankle : jaohoˀgw                                 inalienableSuffix ;
on% my% nail : jiˀehd                                        inalienableSuffix ;
on% my% nail : jiˀohd                                        inalienableSuffix ;
on% your% hip : jisgoˀgw                                  inalienableSuffix ;


LEXICON                        inalienableHˀCCStems !8
on% my% inner% thigh : ˀnhǫhsga:                 inalienableSuffix ;
on% your% nose : ˀnyǫhs                                       inalienableSuffix ;
on% your% neck% (front% of% the% neck) : hnyaˀs          inalienableSuffix ;
on% its% beak : hnyǫdahs                                  inalienableSuffix ;
his% testicles : hsgwa:                                         inalienableSuffix ;
on% your% calf% (of% leg) : hsnaˀd                 inalienableSuffix ;
on% your% upper% back : hswaˀn                        inalienableSuffix ;
on% your% upper% back : hsweˀn                        inalienableSuffix ;


LEXICON                        inalienableSuffix
0 : @U.INALIEN.POSS@                                      locativeSuffix ;
0 : @U.INALIEN.UNPOSS@                                NSF ;


LEXICON                        locativeSuffix
+loc : +aˀgeh                       # ;
                                      # ;


LEXICON                        allBasicNouns
                               basicNounPossessedPrefixes ;
```

```
                                basicNounUnpossessedPrefixes ;


LEXICON                         basicNounPossessedPrefixes
1sP+ :  0                               wagPrefixes ;
1dP+ :  0                               yǫkniPrefixes ;
1pP+ :  0                               yǫgwaPrefixes ;
2sP+ :  0                               saPrefixes ;
2dP+ :  0                               sniPrefixes ;
2pP+ :  0                               swaPrefixes ;
3msP+ :  0                               hoPrefixes ;
3fisP+ :  0                               yagoPrefixes ;
3znsP+ :  0                               yoPrefixes ;
3mdpP+ :  0                               hodiPrefixes ;
3fidpP+ :  0                               yagodiPrefixes ;
3zndpP+ :  0                               yodiPrefixes ;


LEXICON                         wagPrefixes
0 : (w)ag+                         basicNounsAStems ;
0 : (w)ag+                         basicNounsIStems ;
!                                       basicNounsEStems ;
!                                       basicNounsęStems ;
!                                       basicNounsOStems ;
!                                       basicNounsǫStems ;
0 : (w)ak+                         basicNounshor¬CornStems ;
0 : (w)ag+                         basicNounsyorwStems ;
0 : (w)age+                         basicNounsCOtherStems ;


LEXICON                         yǫkniPrefixes
0 : (y)ǫgy+                        basicNounsAStems ;
0 : (y)ǫkn+                        basicNounsIStems ;
!0 : (y)ǫkn+                        basicNounsEStems ;
!0 : (y)ǫkn+                        basicNounsęStems ;
!0 : (y)ǫkn+                        basicNounsOStems ;
!0 : (y)ǫkn+                        basicNounsǫStems ;
0 : (y)ǫkni+                        basicNounsOtherStems ;


LEXICON                         yǫgwaPrefixes
0 : (y)ǫgw+                        basicNounsAStems ;
0 : (y)ǫgwę+                        basicNounsIStems ;
!0 : (y)ǫgw+                        basicNounsEStems ;
!0 : (y)ǫgw+                        basicNounsęStems ;
!0 : (y)ǫgy+                        basicNounsOStems ;
```

```
!0 : (y)ǫgy+                    basicNounsǫStems ;
0 : (y)ǫgwa+                    basicNounsOtherStems ;


LEXICON                         saPrefixes
0 : s+                           basicNounsAStems ;
0 : sę+                           basicNounsIStems ;
!0 : s+                           basicNounsEStems ;
!0 : s+                           basicNounsęStems ;
!0 : s+                           basicNounsOStems ;
!0 : s+                           basicNounsǫStems ;
0 : sa+                           basicNounsOtherStems ;


LEXICON                         sniPrefixes
0 : j+                           basicNounsAStems ;
0 : sn+                           basicNounsIStems ;
!0 : sn+                           basicNounsEStems ;
!0 : sn+                           basicNounsęStems ;
!0 : sn+                           basicNounsOStems ;
!0 : sn+                           basicNounsǫStems ;
0 : sni+                           basicNounsOtherStems ;


LEXICON                         swaPrefixes
0 : sw+                           basicNounsAStems ;
0 : swę+                        basicNounsIStems ;
!0 : sw+                           basicNounsEStems ;
!0 : sw+                           basicNounsęStems ;
!0 : j+                           basicNounsOStems ;
!0 : j+                           basicNounsǫStems ;
0 : swa+                           basicNounsOtherStems ;


LEXICON                         hoPrefixes
0 : ho+                           basicNounsAStems ;
0 : ho+                           basicNounsIStems ;
!0 : haw+                       basicNounsEStems ;
!0 : haw+                       basicNounsęStems ;
!0 : h+                           basicNounsOStems ;
!0 : h+                           basicNounsǫStems ;
0 : ho+                           basicNounsOtherStems ;


LEXICON                         yagoPrefixes
0 : (ya)go+                     basicNounsAStems ;
```

```
0 : (ya)go+                    basicNounsIStems ;
!(0 : ya)gaw+                    basicNounsEStems ;
!0 : (ya)gaw+                    basicNounsęStems ;
!0 : (ya)g+                    basicNounsOStems ;
!0 : (ya)g+                    basicNounsǫStems ;
0 : (ya)go+                    basicNounsOtherStems ;


LEXICON                    yoPrefixes
0 : (y)o+                          basicNounsAStems ;
0 : (y)o+                          basicNounsIStems ;
!0 : (y)aw+                     basicNounsEStems ;
!0 : (y)o+                    basicNounsęStems ;
!0 : (y)+                          basicNounsOStems ;
!0 : (y)+                          basicNounsǫStems ;
0 : (y)o+                          basicNounsOtherStems ;


LEXICON                    hodiPrefixes
0 : hon+                          basicNounsAStems ;
0 : hod+                          basicNounsIStems ;
!0 : hon+                    basicNounsEStems ;
!0 : hon+                    basicNounsęStems ;
!0 : hon+                    basicNounsOStems ;
!0 : hon+                    basicNounsǫStems ;
0 : hodi+                    basicNounsOtherStems ;


LEXICON                    yagodiPrefixes
0 : (ya)gon+              basicNounsAStems ;
0 : (ya)god+              basicNounsIStems ;
!0 : (ya)gon+              basicNounsEStems ;
!0 : (ya)gon+              basicNounsęStems ;
!0 : (ya)gon+              basicNounsOStems ;
!0 : (ya)gon+              basicNounsǫStems ;
0 : (ya)godi+              basicNounsOtherStems ;


LEXICON                    yodiPrefixes
0 : (y)on+                    basicNounsAStems ;
0 : (y)od+                    basicNounsIStems ;
!0 : (y)on+                     basicNounsEStems ;
!0 : (y)on+                     basicNounsęStems ;
!0 : (y)on+                     basicNounsOStems ;
!0 : (y)on+                     basicNounsǫStems ;
0 : (y)odi+                     basicNounsOtherStems ;
```

```
LEXICON                     basicNounUnpossessedPrefixes
3znA+ : ga+                          gaBNouns ;
3znP+ : o+                          oBNouns ;
3znA+ : a+                          aBNouns ;
                                        nullBNouns ;


! Grouping by initial stem vowel for possessed basic nouns
LEXICON                     basicNounshor^CornStems
                            nullBNounshVStems ;
                            gaoh^CNouns ;
                            normalOBNounsh^CnStems ;
                            normalGaBNounsh^CnStems ;


LEXICON                     basicNounsyorwStems
                            normalOBNounsYWStems ;
                            normalGaBNounsYWStems ;


LEXICON                     basicNounsCOtherStems
                            gaoJNouns ;
                            normalOBNounsOtherCStems ;
                            normalGaBNounsOtherCStems ;
                            oLoanWords ;
                            nullBNounshCStems ;


LEXICON                     basicNounsAStems
                            aoBNouns ;
                            normalABNouns ;
                            normalOBNounsAStems ;

LEXICON                     basicNounsIStems
                            normalOBNounsIStems ;
                            normalGaBNounsIStems ;

!LEXICON                     basicNounsEStems
!                            # ;

!LEXICON                     basicNounsęStems
!                            # ;
```

```
!LEXICON                    basicNounsǫStems
!                           # ;


LEXICON                     basicNounsOtherStems
                            oLoanWords ;
                            nullBNouns ;
                            gaoBNouns ;
                            normalOBNounsOtherStems ;
                            normalGaBNounsOtherStems ;

! End grouping by stem vowel for possessed basic nouns


LEXICON                     gaBNouns
                            normalGaBNouns ;
                            gaoBNouns ;


LEXICON                     oBNouns
                            normalOBNouns ;
                            gaoBNouns ;
                            aoBNouns;
                            nullBNouns;
                            oLoanWords;


LEXICON                     aBNouns
                            normalABNouns ;
                            aoBNouns ;


LEXICON                     oLoanWords
0 : basic% noun% root% (loan% words)                        # ;
basic% noun% root% (loan% words) : 0                         # ;
tea : di:                                                              # ;
cheese : ji:s                                                  # ;


LEXICON                     nullBNouns
0 : basic% noun% root% (unprefixed)                           # ;
basic% noun% root% (unprefixed) : 0                             # ;
                            nullBNounshVStems ;
                            nullBNounshCStems ;
```

```
LEXICON                     nullBNounshVStems
basic% noun% root% (beginning% in% %+hV) : 0                  # ;
potato : hǫnaˀd                     NSF ;


LEXICON                     nullBNounshCStems
basic% noun% root% (beginning% in% %+hCC) : 0             # ;
colts% foot : hsgwaęˀd                NSF ;

LEXICON                     gaoBNouns
                            gaohˀCNouns ;
                            gaoJNouns ;


LEXICON                     gaoJNouns
basic% noun% root% (beginning% in% %+j% or% hCC% that% takes% ga%+% or% o%+) : 0 # ;
0 : basic% noun% root% (beginning% in% %+j% or% hCC% that% takes% ga%+% or% o%+) # ;
nakedness%;% nudity : jiˀgw                                          NSF ;
straight% pin%;% pin%;% brooch%;% safety% pin : jihoha:                    NSF ;
dirty% clothes : hsdagw                                          NSF ;
beak : hnyędahs                                                    NSF ;


LEXICON                     gaohˀCNouns
0 : basic% noun% root% (beginning% in% %+hV% or% ˀCV% that% takes% ga%+% or% o%+)
basic% noun% root% (beginning% in% %+hV% or% ˀCV% that% takes% ga%+% or% o%+) : 0
clothespin : ˀwahsd                                  NSF ;
cargo%;% bundle%;% load : hehn                NSF ;
a% motor%;% engine : hǫˀjihsd         NSF ;


LEXICON                     aoBNouns
0 : basic% noun% root% (beginning% in% %+a% that% takes% o%+% or% a%+)
basic% noun% root% (beginning% in% %+a% that% takes% o%+% or% a%+) : 0
ladder%;% stairs : adǫhneˀts      NSF ;


LEXICON                     normalABNouns
0 : basic% noun% root% (beginning% in% %+a% that% takes% a%+)              # ;
basic% noun% root% (beginning% in% %+a% that% takes% a%+) : 0               # ;
fence : adęhę                  NSF ;
blouse%;% middy : adehsw                       NSF ;
lunch%;% groceries : adęnaˀtr           NSF ;
skate : adoˀjin                                          NSF ;
bow% (as% in% bow% and% arrow) : adoda:                      NSF ;
axe%;% tomahawk : adogę                       NSF ;
```

```
hunt : adowadǫ:                                                NSF ;
luck : adraˀsw                                                NSF ;
shoes : ahdahgw                                                NSF ;
clothing%;% clothes : ahgwęny                                NSF ;
roof : ahsgw                                                        NSF ;
closthes : atrǫniˀd                                            NSF ;
calendar : atsogę                                                   NSF ;
flower : awęhę                                                NSF ;
weeds : awęnohgr                                             NSF ;


LEXICON normalOBNouns
                            normalOBNounsAStems ;
                            normalOBNounsIStems ;
                            normalOBNounsOtherStems ;


LEXICON                     normalOBNounsOtherStems
                            normalOBNounshˀCnStems ;
                            normalOBNounsYWStems ;
                            normalOBNounsOtherCStems ;


LEXICON                     normalOBNounsOtherCStems
0 : basic% noun% root% (beginning% with% a% C% other% than% %+y/w/r/% taking% o%+)   # ;
basic% noun% root% (beginning% with% a% C% other% than% %+ˀC/y/w% taking% o%+) : 0  # ;
sod%;% moss : ˀǫhgwa:                        NSF ;
vines : ˀǫhs                                NSF ;
a% price% (on% it) : ga:                                NSF ;
pants : gaˀd                          NSF ;
a% tear% (in% one's% eye) : gahdr                         NSF ;
eyelash%;% the% stem% of% a% berry%;% the% eye% of% the% corn% kernel : gahehd  NSF ;
eyebrow : gahgwaohs                NSF ;
grass : gahoˀj                       NSF ;
cadaver%;% dead% body : ganyęˀd                          NSF ;
hair%;% a% rag%;% (it% is)% ragged%;% tattered : geˀa:          NSF ;
cotton% batting%;% q-tips : gǫˀdr                         NSF ;
a% limb%;% twig%;% branch : gwiy                                 NSF ;
leaves% of% corn : jaǫs                                 NSF ;
curtains%;% lace : jiˀa:                                 NSF ;
the% brain : jiˀdrǫwahd                NSF ;
bug%;% insect%;% worm : jiˀnǫw                          NSF ;
gonorrhea : jigwęd                        NSF ;
porridge%;% mush : jihgw                             NSF ;
mush : jihsgw                         NSF ;
cluster% of% stars%;% star : jihsǫda:                          NSF ;
```

```
bell : jihwęd                       NSF ;
salt : jikeˀd                       NSF ;
nasal% mucous : jinǫhgr                       NSF ;
yellow : jitgwa:                       NSF ;
a% nutshell : kd                              NSF ;
root%;% edible% roots% (pepper% roots%;% turnips%;% carrots) : kdeh        NSF ;
stump%;% knots% in% a% tree : kjin                                NSF ;
its% food : kw                                NSF ;
willow%;% nape% of% neck : sehd                                   NSF ;
snake : shaihsd                         NSF ;
flour%;% powder : teˀtr                                   NSF ;
sumac : tgoˀd                       NSF ;
housefly%;% fly : tręˀd                       NSF ;
mist%;% steam%;% fog : tsad                                   NSF ;
syrup%;% honey%;% gum : tsehsd                        NSF ;
peach% pit : tsgęˀę:                      NSF ;
balsam% fir : tsgoˀd                      NSF ;
saliva%;% spit%;% sputum : tsgr                                   NSF ;
                           normalOBNounshˀCCStems ;


LEXICON                       normalOBNounsYWStems
0 : basic% noun% root% (beginning% with% %+y/w/r% that% takes% o%+)
basic% noun% root% (beginning% with% %+y/w/r% that% takes% o%+) : 0
air%;% wind%;% a% moth : wa:                                  NSF ;
a% peeling : waˀwihsd                       NSF ;
peelings%;% bark% of% a% tree : wajihsd                         NSF ;
fin% of% a% fish%;% wings : way                                 NSF ;
wood% chips : hwęˀhga:                        NSF ;
word%;% voice%;% speech : węn                                   NSF ;
ice : widr                                  NSF ;
sleep%;% a% dream : widręhd                           NSF ;
young%;% offspring% (i.e.% of% an% animal)%;% baby : wiy                  NSF ;
other%;% another : y                                  NSF ;
body : yaˀd                                 NSF ;
basement%;% track%;% ditch : yad                                  NSF ;
pants : yahgw                         NSF ;
tire%;% its% track%;% anything% that% leaves% tracks : yan                  NSF ;
beads : yę:                                 NSF ;
tobacco%;% cigarettes : yęˀgw                           NSF ;
wood%;% firewood : yęd                                NSF ;
bandage : yęhsa:                        NSF ;
a% dead% body%;% cadaver : yǫˀd                                   NSF ;
cheeks : yoˀgw                        NSF ;
guts%;% intestines : yǫw                                    NSF ;
```

message%;% it% matters%;% it% is% its% fault%;% word%;% affair%;% business : rihw

LEXICON normalOBNounshˀCCStems
leaf : nrahd NSF ;
tripe% (cow% stomach% lining)%;% animal% stomache : nregɋˀd NSF ;
tapeworm : nrɋhɋ NSF ;
vomit%;% vomitus : nyaˀgw NSF ;
native% mush% dishes% made% with% corn : nyah NSF ;
stem : nyɋd NSF ;
lumber% logs% (large)%;% timber : nhahd NSF ;
snow%;% snowflake : ˀgr NSF ;
lumber% logs% (large)%;% timber : ˀnhahgy NSF ;
tail% of% an% animal : ˀnhɋhts NSF ;
tail% of% an% animal : ˀnhwɋhts NSF ;
egg : ˀnhǫhs NSF ;
bur : ˀnhǫhd NSF ;
onion : ˀnǫhs NSF ;
nutmeat : hnya: NSF ;
flint% corn% soup : hnye:h NSF ;
nut : hnyoˀgw NSF ;
squash%;% melon : hnyǫhs NSF ;
rain : hsda: NSF ;
scale% (of% a% fish) : hsda: NSF ;
scale% (of% a% fish) : hsdai NSF ;
necklace : hsdaoˀgw NSF ;
rust : hsgɋˀdr NSF ;
louse : hsgeh NSF ;
branch : hsgoh NSF ;
wrinkles : hsgwiˀdr NSF ;
bone%;% bare% bones : hsgyɋˀda: NSF ;
blue% beech% (tree) : hsgyǫˀw NSF ;
dough : hsheˀ NSF ;
hamstrings%;% calves% (of% the% legs)%;% outer% thighs : hsnaˀd NSF ;
straw : hstǫdr NSF ;
coal : hswɋˀd NSF ;

LEXICON normalOBNounshˀCnStems
0 : basic% noun% root% (beginning% with% %+ˀCV/hCV/n% that% takes% o%+)
basic% noun% root% (beginning% with% %+ˀCV/hCV/n% that% takes% o%+) : 0
bread : naˀda: NSF ;
horns%;% antlers : naˀga: NSF ;
cotton% batting : naˀgwiy NSF ;
a% mattress : naˀsgw NSF ;

clay%;% plaster%;% white-wash : nawad NSF ;
evergreen%;% conifer : nę̓d NSF ;
roe% (fish% eggs) : ne̓da: NSF ;
morel% mushroom : negręd NSF ;
peas : negw NSF ;
corn : nęhę: NSF ;
hickory% wood%;% stick : nęnoga: NSF ;
pills : nęnyo̓gw NSF ;
milk : nǫ̓gw NSF ;
catfish : nǫgę̓d NSF ;
corn% cob : nǫhgwę NSF ;
a% husk : nǫny NSF ;
% is% fat%;% gristle%;% rind : ̓dǫdr NSF ;
a% parable%;% tale%;% story%;% legend : ̓ga: NSF ;
ashes%;% bullet%;% dust : ̓gęhę NSF ;
sand : ̓nehs NSF ;
nudity : ̓nest NSF ;
nudity : ̓nost NSF ;
stem%;% hull% of% berries : ̓nihsda: NSF ;
skirt%;% tail%;% feather : ̓yohgw NSF ;
quill%;% plume%;% feather%;% voice%;% throat%;% larynx%;% esophagus : ha̓d NSF ;
road : hah NSF ;
soot : hakd NSF ;
corn% husk : he̓a: NSF ;
dirt%;% earth%;% ground%;% land : hehd NSF ;
fur : hęhda: NSF ;
decayed% tree%;% log%;% wood%;% board : hehs NSF ;
(raw)% sausage%;% bologna%;% wieners : hets NSF ;
one% corn% stalk : hey NSF ;
thorn%;% thistle : hikd NSF ;
cloud : hji̓gr NSF ;
grease%;% oil : hn NSF ;
a% bush%;% a% whip : hǫd NSF ;
basswood : hodr NSF ;
slippery% elm : hohsgr NSF ;
pelt : hohwa: NSF ;
mouth : hsa: NSF ;
beans : hsahe̓d NSF ;
frost : hsęhę NSF ;
thread%;% string%;% cord : hsiy NSF ;
maple : hwahd NSF ;
a% splint : hwę̓ga: NSF ;
corn% ears : hwęhda: NSF ;
foam : hwęhsd NSF ;

```
LEXICON                    normalOBNounsAStems
0 : basic% noun% root% (beginning% with% %+a% that% takes% o%+)          # ;
basic% noun% root% (beginning% with% %+a% that% takes% o%+) : 0            # ;
rust  : ahsgę̓dr                                          NSF ;
snowsnake%;% pole : a̓ęn                    NSF ;
cocoon%;% nest%;% hive%;% bee-hive : adehshę        NSF ;
wall : adęnihs                                  NSF ;
number : ahshed
fruit : ahy


LEXICON                    normalOBNounsIStems
0 : basic% noun% root% (beginning% with% %+i% that% takes% o%+)       # ;
basic% noun% root% (beginning% with% %+i% that% takes% o%+) : 0              # ;
feces%;% shit%;% excrement : i̓d                                        NSF ;
clay%;% mud%;% mortar : i̓da:                               NSF ;
flame : i̓dǫhgw                                  NSF ;
fish : ijǫ̓d                                                    NSF ;


LEXICON                    normalGaBNouns
                           normalGaBNounsIStems ;
                           normalGaBNounsOtherStems ;


LEXICON                    normalGaBNounsOtherStems
                           normalGaBNounsh̓CnStems ;
                           normalGaBNounsYWStems ;
                           normalGaBNounsOtherCStems ;


LEXICON                    normalGaBNounsOtherCStems
0 : basic% noun% root% (beginning% with% a% C% other% than% %+̓C/y/w% taking% ga%+)    # ;
basic% noun% root% (beginning% with% a% C% other% than% %+̓C/y/w% taking% ga%+) : 0   # ;
basket : ̓ahdr                              NSF ;
white% oak : ga̓d                             NSF ;
eye% glasses : gahihsd                              NSF ;
shovel : gahwehs                            NSF ;
paddle : gawehs                             NSF ;
tie%;% scarf : gehd                             NSF ;
pillow%;% cushion : gǫ̓dr                                  NSF ;
the% mask : gǫhs                            NSF ;
dish%;% plate%;% bowl : ję                                  NSF ;
the% devil : jihay                            NSF ;
lamp : jihsd                              NSF ;
hammer : jihw                              NSF ;
```

```
food : kw                                NSF ;
a% handle : tgęhets                                NSF ;
wallet%;% purse%;% pocketbook%;% suitcase : tgwę̓d  NSF ;
bottle%;% jar : tse̓d                                NSF ;
one% animal%;% pet : tsenę̓                                NSF ;


LEXICON                        normalGaBNounsYWStems
0 : basic% noun% root% (beginning% with% %+y/w/r% that% takes% ga%+)
basic% noun% root% (beginning% with% %+y/w/r% that% takes% ga%+) : 0
bag%;% mattress%;% tick%;% pouch% (ie.% a% mattress% bag% into%
                  which% straw% is% stuffed) : ya:   NSF ;
doll : ya̓d                                                                NSF ;
a% celestial% orb% (ie.% the% sun%;% the% moon) : Cagwa:                NSF ;
a% rope : Catsgę̓d                                NSF ;
tin%;% metal : Cihsd                                   NSF ;
leggings : Cisr                                NSF ;
sleep%;% a% dream : Cidręhd                                    NSF ;
song : ręn                                NSF ;
an% agreement : rihwihs                                   NSF ;
log : rǫd                                NSF ;


LEXICON                        normalGaBNounsh̓CnStems
0 : basic% noun% root% (beginning% with% %+̓C/n/h% taking% ga%+)  # ;
basic% noun% root% (beginning% with% %+̓C/n/h% taking% ga%+) : 0 # ;
forest%;% bush : had                                                    NSF ;
elm : hǫga:                           NSF ;
boat : hǫw                             NSF ;
headdress : hsdow                                    NSF ;
a% name : hsęn                              NSF ;
stone%;% rock%;% boulder%;% bullet : hsgwa:                          NSF ;
nails%;% wire%;% needle : hsǫwahd        NSF ;
car%;% truck%;% vehicle : ̓drehd     NSF ;
diaper : ̓droda                                  NSF ;
skirt%;% slip : ̓ka:                                  NSF ;
marriage : ̓na̓gw                 NSF ;
the% mind : ̓nigǫh               NSF ;
earrings : ̓wahsha:                                                 NSF ;
Avocet% blue% stocking% (bird) : ̓yohgw           NSF ;
pail : na̓j                                                  NSF ;
cup : na̓johsgw                            NSF ;
town%;% community : nad                                                 NSF ;
comb : nahd                                                                NSF ;
bass% drum : nahgw                                                    NSF ;
```

```
tame% animal%;% pet%;% domestic% animal : nahsgw                                NSF ;
a% peacock%;% bride%;% boastfulness : naiˀd       NSF ;
bed : nakd                                                          NSF ;
a% board : nehsda:                                          NSF ;
leather%;% hide : nehw                                                NSF ;
porcupine : nheˀd              NSF ;
stick : nhy                                                                 NSF ;
a% house : nǫhs                    NSF ;
a% dance : nǫny      NSF ;
guitar%;% any% string% instrument%;% (refers% to%rounded% back% of% a% turtle) : now NSF ;
spoon%;% canoe%;% birch% bark% canoe : nyod              NSF ;


LEXICON                       normalGaBNounsIStems
0 : basic% noun% root% (beginning% with% %+i% that% takes% ga%+)              # ;
basic% noun% root% (beginning% with% %+i% that% takes% ga%+) : 0        # ;
bottle%;% jar : itseˀd                                      NSF ;
mattress%;% sleeping% mat : itsga:                                          NSF ;


LEXICON        NSF
+NSF : +aˀ                     # ;
                                    # ;
```

Appendix C

Test Cases

This appendix contains tables listing all the surface and underlying forms that were used to the validity of my machine. There are two sets of data, one for use with testing the abstract approach and one for testing the concrete approach. Each data set consists of six series of tables; one for each type of noun (the tables are divided by prefix type where applicable: ie. a separate table for possessed an unpossessed basic nouns).

As described in §7.2.1 the data from the **surface** column of each table was run through the machine, and compared to the expected output from the **underlying** column. If there was a mismatch or missing word, the program reported an error. This process was then repeated taking the data from the **underlying** column, running it through the machine and comparing the output with the expected result from the **surface** column.

C.1 Data for the Abstract Approach

Table C.1: Unpossessed Basic Nouns

Surface	**Underlying**
hǫna˥da˥	hǫna˥d+a˥
odi:	o+di:
ga˥wahsda˥	ga+˥wahsd+a˥
o˥wahsda˥	o+˥wahsd+a˥
gahsdagwa˥	ga+hsdagw+a˥
ohsdagwa˥	o+hsdagw+a˥
adǫhne˥tsa˥	a+adǫhne˥ts+a˥
awęnohgra˥	a+awęnohgr+a˥
o˥nhahgya˥	o+˥nhahgy+a˥
ohya˥	o+ahy+a˥
ohwęhsda˥	o+hwęhsd+a˥

Surface	**Underlying**
hsgwaę˥da˥	hsgwaę˥d+a˥
oji:s	o+ji:s
gahehna˥	ga+hehn+a˥
ohehna˥	o+hehn+a˥
gaji˥gwa˥	ga+ji˥gw+a˥
oji˥gwa˥	o+ji˥gw+a˥
odǫhne˥tsa˥	o+adǫhne˥ts+a˥
adra˥swa˥	a+adra˥sw+a˥
onhahda˥	o+nhahd+a˥
oga:˥	o+ga:+a˥
o˥da˥	o+i˥d+a˥

[0]This is an incorrect form that was used during the testing. This underscores the fact that despite using a computer to check for errors, work is always subject to human error. Please see the discussion in §7.2.1 for more information.

Unpossessed Basic Nouns Continued

Surface	Underlying
okwa˺	o+kw+a˺
oihwa˺	o+rihw+a˺
otsgra˺	o+tsgr+a˺
owiya˺	o+wiy+a˺
oya˺da˺	o+ya˺d+a˺
ga:gwa:˺	ga+Cagwa:+a˺
gahsǫwahda˺	ga+hsǫwahd+a˺
gajɛ˺	ga+jɛ+a˺
gana˺ja˺	ga+na˺j+a˺
gatgɛhetsa˺	ga+tgɛhets+a˺

Surface	Underlying
ona˺da:˺	o+na˺da:+a˺
osehda˺	o+sehd+a˺
owa:˺	o+wa:+a˺
oya˺	o+y+a˺
ga˺wahsha:˺	ga+˺wahsha:+a˺
gaga˺da˺	ga+ga˺d+a˺
gɛtsga:˺	ga+itsga:+a˺
gakwa˺	ga+kw+a˺
gaǫda˺	ga+rǫd+a˺
gaya:˺	ga+ya:+a˺

Table C.2: Possessed Basic Nouns

Surface	Underlying
akǫna˺da˺	(w)ag+hǫna˺d+a˺
ǫgwahǫna˺da˺	(y)ǫgwa+hǫna˺d+a˺
snihǫna˺da˺	sni+hǫna˺d+a˺
hohǫna˺da˺	ho+hǫna˺d+a˺
ohǫna˺da˺	(y)o+hǫna˺d+a˺
godihǫna˺da˺	(ya)godi+hǫna˺d+a˺
agehsgwaɛ˺da˺	(w)ag+hsgwaɛ˺d+a˺
ǫgwahsgwaɛ˺da˺	(y)ǫgwa+hsgwaɛ˺d+a˺
snihsgwaɛ˺da˺	sni+hsgwaɛ˺d+a˺
hohsgwaɛ˺da˺	ho+hsgwaɛ˺d+a˺
ohsgwaɛ˺da˺	(y)o+hsgwaɛ˺d+a˺
godihsgwaɛ˺da˺	(ya)godi+hsgwaɛ˺d+a˺
akwahsda˺	(w)ag+˺wahsd+a˺
ǫgwa˺wahsda˺	(y)ǫgwa+˺wahsd+a˺
sni˺wahsda˺	sni+˺wahsd+a˺
ho˺wahsda˺	ho+˺wahsd+a˺
o˺wahsda˺	(y)o+˺wahsd+a˺
godi˺wahsda˺	(ya)godi+˺wahsd+a˺
akehna˺	(w)ag+hehn+a˺
ǫgwahehna˺	(y)ǫgwa+hehn+a˺
snihehna˺	sni+hehn+a˺

Surface	Underlying
ǫknihǫna˺da˺	(y)ǫkni+hǫna˺d+a˺
sahǫna˺da˺	sa+hǫna˺d+a˺
swahǫna˺da˺	swa+hǫna˺d+a˺
gohǫna˺da˺	(ya)go+hǫna˺d+a˺
hodihǫna˺da˺	hodi+hǫna˺d+a˺
odihǫna˺da˺	(y)odi+hǫna˺d+a˺
ǫknihsgwaɛ˺da˺	(y)ǫkni+hsgwaɛ˺d+a˺
sahsgwaɛ˺da˺	sa+hsgwaɛ˺d+a˺
swahsgwaɛ˺da˺	swa+hsgwaɛ˺d+a˺
gohsgwaɛ˺da˺	(ya)go+hsgwaɛ˺d+a˺
hodihsgwaɛ˺da˺	hodi+hsgwaɛ˺d+a˺
odihsgwaɛ˺da˺	(y)odi+hsgwaɛ˺d+a˺
ǫkni˺wahsda˺	(y)ǫkni+˺wahsd+a˺
sa˺wahsda˺	sa+˺wahsd+a˺
swa˺wahsda˺	swa+˺wahsd+a˺
go˺wahsda˺	(ya)go+˺wahsd+a˺
hodi˺wahsda˺	hodi+˺wahsd+a˺
odi˺wahsda˺	(y)odi+˺wahsd+a˺
ǫknihehna˺	(y)ǫkni+hehn+a˺
sahehna˺	sa+hehn+a˺
swahehna˺	swa+hehn+a˺

Possessed Basic Nouns Continued

Surface	Underlying
hohehnaˀ	ho+hehn+aˀ
ohehnaˀ	(y)o+hehn+aˀ
godihehnaˀ	(ya)godi+hehn+aˀ
agejihoha:ˀ	(w)ag+jihoha:+aˀ
ǫgwajihoha:ˀ	(y)ǫgwa+jihoha:+aˀ
snijihoha:ˀ	sni+jihoha:+aˀ
hojihoha:ˀ	ho+jihoha:+aˀ
ojihoha:ˀ	(y)o+jihoha:+aˀ
godijihoha:ˀ	(ya)godi+jihoha:+aˀ
agadǫhneˀtsaˀ	(w)ag+adǫhneˀts+aˀ
ǫgwadǫhneˀtsaˀ	(y)ǫgwa+adǫhneˀts+aˀ
jadǫhneˀtsaˀ	sni+adǫhneˀts+aˀ
hodǫhneˀtsaˀ	ho+adǫhneˀts+aˀ
odǫhneˀtsaˀ	(y)o+adǫhneˀts+aˀ
gonadǫhneˀtsaˀ	(ya)godi+adǫhneˀts+aˀ
*ageˀnhǫhsaˀ[1]	(w)ag+ˀnhǫhs+aˀ
ǫgwaˀnhǫhsaˀ	(y)ǫgwa+ˀnhǫhs+aˀ
sniˀnhǫhsaˀ	sni+ˀnhǫhs+aˀ
hoˀnhǫhsaˀ	ho+ˀnhǫhs+aˀ
oˀnhǫhsaˀ	(y)o+ˀnhǫhs+aˀ
godiˀnhǫhsaˀ	(ya)godi+ˀnhǫhs+aˀ
ageˀǫhgwa:ˀ	(w)ag+ˀǫhgwa:+aˀ
ǫgwaˀǫhgwa:ˀ	(y)ǫgwa+ˀǫhgwa:+aˀ
sniˀǫhgwa:ˀ	sni+ˀǫhgwa:+aˀ
hoˀǫhgwa:ˀ	ho+ˀǫhgwa:+aˀ
oˀǫhgwa:ˀ	(y)o+ˀǫhgwa:+aˀ
godiˀǫhgwa:ˀ	(ya)godi+ˀǫhgwa:+aˀ
agenhahdaˀ	(w)ag+nhahd+aˀ
ǫgwanhahdaˀ	(y)ǫgwa+nhahd+aˀ
sninhahdaˀ	sni+nhahd+aˀ
honhahdaˀ	ho+nhahd+aˀ
onhahdaˀ	(y)o+nhahd+aˀ
godinhahdaˀ	(ya)godi+nhahd+aˀ
agegaˀdaˀ	(w)ag+gaˀd+aˀ
ǫgwagaˀdaˀ	(y)ǫgwa+gaˀd+aˀ
snigaˀdaˀ	sni+gaˀd+aˀ
hogaˀdaˀ	ho+gaˀd+aˀ
ogaˀdaˀ	(y)o+gaˀd+aˀ

Surface	Underlying
gohehnaˀ	(ya)go+hehn+aˀ
hodihehnaˀ	hodi+hehn+aˀ
odihehnaˀ	(y)odi+hehn+aˀ
ǫknijihoha:ˀ	(y)ǫkni+jihoha:+aˀ
sajihoha:ˀ	sa+jihoha:+aˀ
swajihoha:ˀ	swa+jihoha:+aˀ
gojihoha:ˀ	(ya)go+jihoha:+aˀ
hodijihoha:ˀ	hodi+jihoha:+aˀ
odijihoha:ˀ	(y)odi+jihoha:+aˀ
ǫgyadǫhneˀtsaˀ	(y)ǫkni+adǫhneˀts+aˀ
sadǫhneˀtsaˀ	sa+adǫhneˀts+aˀ
swadǫhneˀtsaˀ	swa+adǫhneˀts+aˀ
godǫhneˀtsaˀ	(ya)go+adǫhneˀts+aˀ
honadǫhneˀtsaˀ	hodi+adǫhneˀts+aˀ
onadǫhneˀtsaˀ	(y)odi+adǫhneˀts+aˀ
ǫkniˀnhǫhsaˀ	(y)ǫkni+ˀnhǫhs+aˀ
saˀnhǫhsaˀ	sa+ˀnhǫhs+aˀ
swaˀnhǫhsaˀ	swa+ˀnhǫhs+aˀ
goˀnhǫhsaˀ	(ya)go+ˀnhǫhs+aˀ
hodiˀnhǫhsaˀ	hodi+ˀnhǫhs+aˀ
odiˀnhǫhsaˀ	(y)odi+ˀnhǫhs+aˀ
ǫkniˀǫhgwa:ˀ	(y)ǫkni+ˀǫhgwa:+aˀ
saˀǫhgwa:ˀ	sa+ˀǫhgwa:+aˀ
swaˀǫhgwa:ˀ	swa+ˀǫhgwa:+aˀ
goˀǫhgwa:ˀ	(ya)go+ˀǫhgwa:+aˀ
hodiˀǫhgwa:ˀ	hodi+ˀǫhgwa:+aˀ
odiˀǫhgwa:ˀ	(y)odi+ˀǫhgwa:+aˀ
ǫkninhahdaˀ	(y)ǫkni+nhahd+aˀ
sanhahdaˀ	sa+nhahd+aˀ
swanhahdaˀ	swa+nhahd+aˀ
gonhahdaˀ	(ya)go+nhahd+aˀ
hodinhahdaˀ	hodi+nhahd+aˀ
odinhahdaˀ	(y)odi+nhahd+aˀ
ǫknigaˀdaˀ	(y)ǫkni+gaˀd+aˀ
sagaˀdaˀ	sa+gaˀd+aˀ
swagaˀdaˀ	swa+gaˀd+aˀ
gogaˀdaˀ	(ya)go+gaˀd+aˀ
hodigaˀdaˀ	hodi+gaˀd+aˀ

Possessed Basic Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
godigaˀdaˀ	(ya)godi+gaˀd+aˀ		odigaˀdaˀ	(y)odi+gaˀd+aˀ
akakdaˀ	(w)ag+hakd+aˀ		ǫknihakdaˀ	(y)ǫkni+hakd+aˀ
ǫgwahakdaˀ	(y)ǫgwa+hakd+aˀ		sahakdaˀ	sa+hakd+aˀ
snihakdaˀ	sni+hakd+aˀ		swahakdaˀ	swa+hakd+aˀ
hohakdaˀ	ho+hakd+aˀ		gohakdaˀ	(ya)go+hakd+aˀ
ohakdaˀ	(y)o+hakd+aˀ		hodihakdaˀ	hodi+hakd+aˀ
godihakdaˀ	(ya)godi+hakd+aˀ		odihakdaˀ	(y)odi+hakd+aˀ
agiˀdǫhgwaˀ	(w)ag+iˀdǫhgw+aˀ		ǫkniˀdǫhgwaˀ	(y)ǫkni+iˀdǫhgw+aˀ
ǫgwęˀdǫhgwaˀ	(y)ǫgwa+iˀdǫhgw+aˀ		sęˀdǫhgwaˀ	sa+iˀdǫhgw+aˀ
sniˀdǫhgwaˀ	sni+iˀdǫhgw+aˀ		swęˀdǫhgwaˀ	swa+iˀdǫhgw+aˀ
hoˀdǫhgwaˀ	ho+iˀdǫhgw+aˀ		goˀdǫhgwaˀ	(ya)go+iˀdǫhgw+aˀ
oˀdǫhgwaˀ	(y)o+iˀdǫhgw+aˀ		hodiˀdǫhgwaˀ	hodi+iˀdǫhgw+aˀ
godiˀdǫhgwaˀ	(ya)godi+iˀdǫhgw+aˀ		odiˀdǫhgwaˀ	(y)odi+iˀdǫhgw+aˀ
agejihsǫda:ˀ	(w)ag+jihsǫda:+aˀ		ǫknijihsǫda:ˀ	(y)ǫkni+jihsǫda:+aˀ
ǫgwajihsǫda:ˀ	(y)ǫgwa+jihsǫda:+aˀ		sajihsǫda:ˀ	sa+jihsǫda:+aˀ
snijihsǫda:ˀ	sni+jihsǫda:+aˀ		swajihsǫda:ˀ	swa+jihsǫda:+aˀ
hojihsǫda:ˀ	ho+jihsǫda:+aˀ		gojihsǫda:ˀ	(ya)go+jihsǫda:+aˀ
ojihsǫda:ˀ	(y)o+jihsǫda:+aˀ		hodijihsǫda:ˀ	hodi+jihsǫda:+aˀ
godijihsǫda:ˀ	(ya)godi+jihsǫda:+aˀ		odijihsǫda:ˀ	(y)odi+jihsǫda:+aˀ
agekdehaˀ	(w)ag+kdeh+aˀ		ǫknikdehaˀ	(y)ǫkni+kdeh+aˀ
ǫgwakdehaˀ	(y)ǫgwa+kdeh+aˀ		sakdehaˀ	sa+kdeh+aˀ
snikdehaˀ	sni+kdeh+aˀ		swakdehaˀ	swa+kdeh+aˀ
hokdehaˀ	ho+kdeh+aˀ		gokdehaˀ	(ya)go+kdeh+aˀ
okdehaˀ	(y)o+kdeh+aˀ		hodikdehaˀ	hodi+kdeh+aˀ
godikdehaˀ	(ya)godi+kdeh+aˀ		odikdehaˀ	(y)odi+kdeh+aˀ
agrihwaˀ	(w)ag+rihw+aˀ		ǫkni:hwaˀ	(y)ǫkni+rihw+aˀ
ǫgwaihwaˀ	(y)ǫgwa+rihw+aˀ		saihwaˀ	sa+rihw+aˀ
sni:hwaˀ	sni+rihw+aˀ		swaihwaˀ	swa+rihw+aˀ
hoihwaˀ	ho+rihw+aˀ		goihwaˀ	(ya)go+rihw+aˀ
oihwaˀ	(y)o+rihw+aˀ		hodi:hwaˀ	hodi+rihw+aˀ
godi:hwaˀ	(ya)godi+rihw+aˀ		odi:hwaˀ	(y)odi+rihw+aˀ
agesehdaˀ	(w)ag+sehd+aˀ		ǫknisehdaˀ	(y)ǫkni+sehd+aˀ
ǫgwasehdaˀ	(y)ǫgwa+sehd+aˀ		sasehdaˀ	sa+sehd+aˀ
snisehdaˀ	sni+sehd+aˀ		swasehdaˀ	swa+sehd+aˀ
hosehdaˀ	ho+sehd+aˀ		gosehdaˀ	(ya)go+sehd+aˀ
osehdaˀ	(y)o+sehd+aˀ		hodisehdaˀ	hodi+sehd+aˀ
godisehdaˀ	(ya)godi+sehd+aˀ		odisehdaˀ	(y)odi+sehd+aˀ
agetsehsdaˀ	(w)ag+tsehsd+aˀ		ǫknitsehsdaˀ	(y)ǫkni+tsehsd+aˀ
ǫgwatsehsdaˀ	(y)ǫgwa+tsehsd+aˀ		satsehsdaˀ	sa+tsehsd+aˀ

Possessed Basic Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
snitsehsdaˀ	sni+tsehsd+aˀ		swatsehsdaˀ	swa+tsehsd+aˀ
hotsehsdaˀ	ho+tsehsd+aˀ		gotsehsdaˀ	(ya)go+tsehsd+aˀ
otsehsdaˀ	(y)o+tsehsd+aˀ		hoditsehsdaˀ	hodi+tsehsd+aˀ
goditsehsdaˀ	(ya)godi+tsehsd+aˀ		oditsehsdaˀ	(y)odi+tsehsd+aˀ
agwaˀwihsdaˀ	(w)ag+waˀwihsd+aˀ		ǫkniwaˀwihsdaˀ	(y)ǫkni+waˀwihsd+aˀ
ǫgwawaˀwihsdaˀ	(y)ǫgwa+waˀwihsd+aˀ		sawaˀwihsdaˀ	sa+waˀwihsd+aˀ
sniwaˀwihsdaˀ	sni+waˀwihsd+aˀ		swawaˀwihsdaˀ	swa+waˀwihsd+aˀ
howaˀwihsdaˀ	ho+waˀwihsd+aˀ		gowaˀwihsdaˀ	(ya)go+waˀwihsd+aˀ
owaˀwihsdaˀ	(y)o+waˀwihsd+aˀ		hodiwaˀwihsdaˀ	hodi+waˀwihsd+aˀ
godiwaˀwihsdaˀ	(ya)godi+waˀwihsd+aˀ		odiwaˀwihsdaˀ	(y)odi+waˀwihsd+aˀ
agyǫˀdaˀ	(w)ag+yǫˀd+aˀ		ǫkniyǫˀdaˀ	(y)ǫkni+yǫˀd+aˀ
ǫgwayǫˀdaˀ	(y)ǫgwa+yǫˀd+aˀ		sayǫˀdaˀ	sa+yǫˀd+aˀ
sniyǫˀdaˀ	sni+yǫˀd+aˀ		swayǫˀdaˀ	swa+yǫˀd+aˀ
hoyǫˀdaˀ	ho+yǫˀd+aˀ		goyǫˀdaˀ	(ya)go+yǫˀd+aˀ
oyǫˀdaˀ	(y)o+yǫˀd+aˀ		hodiyǫˀdaˀ	hodi+yǫˀd+aˀ
godiyǫˀdaˀ	(ya)godi+yǫˀd+aˀ		odiyǫˀdaˀ	(y)odi+yǫˀd+aˀ
agihsdaˀ	(w)ag+Cihsd+aˀ		ǫkni:hsdaˀ	(y)ǫkni+Cihsd+aˀ
ǫgwaihsdaˀ	(y)ǫgwa+Cihsd+aˀ		saihsdaˀ	sa+Cihsd+aˀ
sni:hsdaˀ	sni+Cihsd+aˀ		swaihsdaˀ	swa+Cihsd+aˀ
hoihsdaˀ	ho+Cihsd+aˀ		goihsdaˀ	(ya)go+Cihsd+aˀ
oihsdaˀ	(y)o+Cihsd+aˀ		hodi:hsdaˀ	hodi+Cihsd+aˀ
godi:hsdaˀ	(ya)godi+Cihsd+aˀ		odi:hsdaˀ	(y)odi+Cihsd+aˀ
agatsgęˀdaˀ	(w)ag+Catsgęˀd+aˀ		ǫkniatsgęˀdaˀ	(y)ǫkni+Catsgęˀd+aˀ
ǫgwa:tsgęˀdaˀ	(y)ǫgwa+Catsgęˀd+aˀ		sa:tsgęˀdaˀ	sa+Catsgęˀd+aˀ
sniatsgęˀdaˀ	sni+Catsgęˀd+aˀ		swa:tsgęˀdaˀ	swa+Catsgęˀd+aˀ
hoatsgęˀdaˀ	ho+Catsgęˀd+aˀ		goatsgęˀdaˀ	(ya)go+Catsgęˀd+aˀ
oatsgęˀdaˀ	(y)o+Catsgęˀd+aˀ		hodiatsgęˀdaˀ	hodi+Catsgęˀd+aˀ
godiatsgęˀdaˀ	(ya)godi+Catsgęˀd+aˀ		odiatsgęˀdaˀ	(y)odi+Catsgęˀd+aˀ
ageˀahdraˀ	(w)ag+ˀahdr+aˀ		ǫkniˀahdraˀ	(y)ǫkni+ˀahdr+aˀ
ǫgwaˀahdraˀ	(y)ǫgwa+ˀahdr+aˀ		saˀahdraˀ	sa+ˀahdr+aˀ
sniˀahdraˀ	sni+ˀahdr+aˀ		swaˀahdraˀ	swa+ˀahdr+aˀ
hoˀahdraˀ	ho+ˀahdr+aˀ		goˀahdraˀ	(ya)go+ˀahdr+aˀ
oˀahdraˀ	(y)o+ˀahdr+aˀ		hodiˀahdraˀ	hodi+ˀahdr+aˀ
godiˀahdraˀ	(ya)godi+ˀahdr+aˀ		odiˀahdraˀ	(y)odi+ˀahdr+aˀ
agitsga:ˀ	(w)ag+itsga:+aˀ		ǫknitsga:ˀ	(y)ǫkni+itsga:+aˀ
ǫgwętsga:ˀ	(y)ǫgwa+itsga:+aˀ		sętsga:ˀ	sa+itsga:+aˀ
snitsga:ˀ	sni+itsga:+aˀ		swętsga:ˀ	swa+itsga:+aˀ
hotsga:ˀ	ho+itsga:+aˀ		gotsga:ˀ	(ya)go+itsga:+aˀ
otsga:ˀ	(y)o+itsga:+aˀ		hoditsga:ˀ	hodi+itsga:+aˀ

Possessed Basic Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
goditsga:˺	(ya)godi+itsga:+a˺		oditsga:˺	(y)odi+itsga:+a˺
agrẹna˺	(w)ag+rẹn+a˺		ǫknięna˺	(y)ǫkni+rẹn+a˺
ǫgwaęna˺	(y)ǫgwa+rẹn+a˺		saęna˺	sa+rẹn+a˺
snięna˺	sni+rẹn+a˺		swaęna˺	swa+rẹn+a˺
hoęna˺	ho+rẹn+a˺		goęna˺	(ya)go+rẹn+a˺
oęna˺	(y)o+rẹn+a˺		hodięna˺	hodi+rẹn+a˺
godięna˺	(ya)godi+rẹn+a˺		odięna˺	(y)odi+rẹn+a˺
agrihwihsa˺	(w)ag+rihwihs+a˺		ǫkni:hwihsa˺	(y)ǫkni+rihwihs+a˺
ǫgwaihwihsa˺	(y)ǫgwa+rihwihs+a˺		saihwihsa˺	sa+rihwihs+a˺
sni:hwihsa˺	sni+rihwihs+a˺		swaihwihsa˺	swa+rihwihs+a˺
hoihwihsa˺	ho+rihwihs+a˺		goihwihsa˺	(ya)go+rihwihs+a˺
oihwihsa˺	(y)o+rihwihs+a˺		hodi:hwihsa˺	hodi+rihwihs+a˺
godi:hwihsa˺	(ya)godi+rihwihs+a˺		odi:hwihsa˺	(y)odi+rihwihs+a˺
agrǫda˺	(w)ag+rǫd+a˺		ǫkniǫda˺	(y)ǫkni+rǫd+a˺
ǫgwaǫda˺	(y)ǫgwa+rǫd+a˺		saǫda˺	sa+rǫd+a˺
sniǫda˺	sni+rǫd+a˺		swaǫda˺	swa+rǫd+a˺
hoǫda˺	ho+rǫd+a˺		goǫda˺	(ya)go+rǫd+a˺
oǫda˺	(y)o+rǫd+a˺		hodiǫda˺	hodi+rǫd+a˺
godiǫda˺	(ya)godi+rǫd+a˺		odiǫda˺	(y)odi+rǫd+a˺

Table C.3: Deverbal Nouns

Surface	**Underlying**
ęde̱hsra˺	ęde̱hsr+a˺
ę˺nhotra˺	ę˺nhotr+a˺
gayęnawahsra˺	ga+yęnawahsr+a˺
gaya˺dagenhahsra˺	ga+ya˺dagenhahsr+a˺
gatgǫnya˺tra˺	ga+atgǫnya˺tr+a˺
ganohǫkdehsra˺	ga+nohǫkdehsr+a˺
gana˺jowi˺tra˺	ga+na˺jowi˺tr+a˺
gaihwiyohsdę̱hsra˺	ga+rihwiyohsdę̱hsr+a˺
gaiho˺dę̱hsra˺	ga+riho˺dę̱hsr+a˺
gahshahsdę̱hsra˺	ga+hshahsdę̱hsr+a˺
oyę̱hsra˺	o+yę̱hsr+a˺
onrahdǫdahsra˺	o+nrahdǫdahsr+a˺
oniga:hę̱hsra˺	o+niga:hę̱hsr+a˺
odǫtgadǫhsra˺	o+adǫtgadǫhsr+a˺
o˺drohsra˺	o+˺drohsr+a˺
atsho˺kdǫhsra˺	a+atsho˺kdǫhsr+a˺
atna˺gwihdra˺	a+atna˺gwihdr+a˺
anahaotra˺	a+nahaotr+a˺
adrihwagyaǫhsra˺	a+adrihwagyaǫhsr+a˺
adǫnhehsra˺	a+adǫnhehsr+a˺
adęna˺tra˺	a+adęna˺tr+a˺
adao˺tra˺	a+adao˺tra+a˺

Surface	**Underlying**
ę˺nyotra˺	ę˺nyotr+a˺
gędehsra˺	ga+idehsra+a˺
gaya˺dowehdahsra˺	ga+ya˺dowehdahsr+a˺
gatgwęnya˺tra˺	ga+atgwęnya˺tr+a˺
gatgi˺tra˺	ga+tgi˺tr+a˺
ganhehsra˺	ga+nhehsr+a˺
gaisra˺	ga+risr+a˺
gaihwane˺aksra˺	ga+rihwane˺aksra+a˺
gahyadǫhsra˺	ga+hyadǫhsr+a˺
ga:hǫhsra˺	ga+Cahǫhsr+a˺
otgahnǫnihsra˺	o+atgahnǫnihsr+a˺
onǫnhe˺dra˺	o+nǫnhe˺dr+a˺
ohshahsdę̱hsra˺	o+hshahsdę̱hsr+a˺
odǫtgadehsra˺	o+adǫtgadehsr+a˺
o˺daihę̱hdra˺	o+i˺daihę̱hdr+a˺
atna˺tsotra˺	a+atna˺tsotr+a˺
atgahnyehtra˺	a+atgahnyehtr+a˺
agya˺dawi˺tra˺	a+agya˺dawi˺tr+a˺
adra˺wihsda˺	a+adra˺wihsd+a˺
adi˺grǫhsra˺	a+adi˺grǫhsra+a˺
adekwahahsra˺	a+adekwahahsra+a˺
ahdahdi˺tra˺	a+ahdahdi˺tr+a˺

Table C.4: Defective Nouns

Surface	Underlying
sgwa:gwaǫdǫ˥	sgwa:gwaǫdǫ˥
da:gu:s	da:gu:s
so:wa:s	so:wa:s
hǫ:ga:k	hǫ:ga:k
gwihsgwihs	gwihsgwihs
sohǫ:t	sohǫ:t
jogrihs	jogrihs
di˥di:˥	di˥di:˥
ga˥ga:˥	ga˥ga:˥
gwiyę˥gwiyę˥	gwiyę˥gwiyę˥
ji˥nhǫwę:se:	ji˥nhǫwę:se:
sa˥sa˥	sa˥sa˥
gwe:sę˥	gwe:sę˥
jihsgogo˥	jihsgogo˥
johwę˥sdaga˥	johwę˥sdaga˥
hnyagwai˥	hnyagwai˥
tgwiyo:gę˥	tgwiyo:gę˥
ji˥nǫhdo:ya˥	ji˥nǫhdo:ya˥
jinǫhsanǫh	jinǫhsanǫh
ji˥ao:yę:	ji˥ao:yę:
degriya˥gǫ˥	degriya˥gǫ˥
hehshai:	hehshai:
tehtǫ˥	tehtǫ˥
jino:wę:	jino:wę:
sa:no:˥	sa:no:˥
dre:na:	dre:na:
kdagǫ˥	kdagǫ˥
sgwa:yęh	sgwa:yęh
jǫ:nyǫ:˥	jǫ:nyǫ:˥
gwa˥da:	gwa˥da:
jǫ˥dae:ya:˥	jǫ˥dae:ya:˥
onohotsgę˥ę˥	onohotsgę˥ę˥
tsa˥gę:da˥	tsa˥gę:da˥
tsinyohgwa:k	tsinyohgwa:k
nawę˥da˥	nawę˥da˥
otahyǫ:ni:	otahyǫ:ni:
jihsgę:	jihsgę:
grahe:t	grahe:t

Surface	Underlying
dago:s	dago:s
dakshae˥dohs	dakshae˥dohs
twę:twę:t	twę:twę:t
dogę:t	dogę:t
gwa˥yǫ˥	gwa˥yǫ˥
gyo:gyo:˥	gyo:gyo:˥
gwido˥gwido˥	gwido˥gwido˥
jikjiye:˥	jikjiye:˥
hihi:	hihi:
dihsdihs	dihsdihs
duwisduwi:˥	duwisduwi:˥
gwę:dihs	gwę:dihs
tsahgo:wah	tsahgo:wah
gwaoh	gwaoh
gwę˥gohnyę˥	gwę˥gohnyę˥
gǫ:deh	gǫ:deh
jinhǫhgwahęh	jinhǫhgwahęh
ji˥dana:wę:	ji˥dana:wę:
jihsda:	jihsda:
jinǫhyahae:	jinǫhyahae:
jihnyo˥gę˥	jihnyo˥gę˥
sgwa˥ahda˥	sgwa˥ahda˥
jǫ˥daga˥	jǫ˥daga˥
tea:ǫt	tea:ǫt
drę:na:	drę:na:
joni:tsgrǫ:t	joni:tsgrǫ:t
do:dihs	do:dihs
gwiyo:gę˥	gwiyo:gę˥
nǫhsodai:yǫ:	nǫhsodai:yǫ:
jidę:˥ęh	jidę:˥ęh
ji˥drǫ:wę:	ji˥drǫ:wę:
teo:ji˥	teo:ji˥
yahgęhda˥	yahgęhda˥
gihę:k	gihę:k
jihsǫ:dahk	jihsǫ:dahk
tahyǫ:ni:	tahyǫ:ni:
ji˥o:	ji˥o:

Table C.5: Inalienable Nouns

Surface	**Underlying**
gahǫhdaˀgeh	g+ahǫhd+aˀgeh
gyahǫhdaˀgeh	(e)kni+ahǫhd+aˀgeh
dwahǫhdaˀgeh	(e)dwa+ahǫhd+aˀgeh
sahǫhdaˀgeh	(h)s+ahǫhd+aˀgeh
swahǫhdaˀgeh	(h)swa+ahǫhd+aˀgeh
ǫhǫhdaˀgeh	(y)ǫ+ahǫhd+aˀgeh
hęnahǫhdaˀgeh	hadi+ahǫhd+aˀgeh
gęnahǫhdaˀgeh	gadi+ahǫhd+aˀgeh
gyahsiˀdaˀgeh	(e)tni+ahsiˀd+aˀgeh
agyahsiˀdaˀgeh	(y)akni+ahsiˀd+aˀgeh
agwahsiˀdaˀgeh	(y)agwa+ahsiˀd+aˀgeh
jahsiˀdaˀgeh	(h)sni+ahsiˀd+aˀgeh
hahsiˀdaˀgeh	ha+ahsiˀd+aˀgeh
wahsiˀdaˀgeh	w+ahsiˀd+aˀgeh
gaǫhsiˀdaˀgeh	gaǫ+ahsiˀd+aˀgeh
gihnaˀgeh	g+ihn+aˀgeh
knihnaˀgeh	(e)kni+ihn+aˀgeh
dwęhnaˀgeh	(e)dwa+ihn+aˀgeh
sihnaˀgeh	(h)s+ihn+aˀgeh
swęhnaˀgeh	(h)swa+ihn+aˀgeh
ęhnaˀgeh	(y)ę+ihn+aˀgeh
hadihnaˀgeh	hadi+ihn+aˀgeh
gadihnaˀgeh	gadi+ihn+aˀgeh
tnǫtsaˀgeh	(e)tni+ǫts+aˀgeh
aknǫtsaˀgeh	(y)akni+ǫts+aˀgeh
agyǫtsaˀgeh	(y)agwa+ǫts+aˀgeh
snǫtsaˀgeh	(h)sni+ǫts+aˀgeh
hǫtsaˀgeh	ha+ǫts+aˀgeh
ǫtsaˀgeh	(y)+ǫts+aˀgeh
ga:gǫtsaˀgeh	ga:g+ǫts+aˀgeh
gwęˀnahsaˀgeh	g+węˀnahs+aˀgeh
kniwęˀnahsaˀgeh	(e)kni+węˀnahs+aˀgeh
dwawęˀnahsaˀgeh	(e)dwa+węˀnahs+aˀgeh
swęˀnahsaˀgeh	(eh)s+węˀnahs+aˀgeh
swawęˀnahsaˀgeh	(eh)swa+węˀnahs+aˀgeh
ewęˀnahsaˀgeh	(y)e+węˀnahs+aˀgeh
hadiwęˀnahsaˀgeh	hadi+węˀnahs+aˀgeh
gadiwęˀnahsaˀgeh	gadi+węˀnahs+aˀgeh
tnigahaˀgeh	(e)tni+gah+aˀgeh

Surface	**Underlying**
gyahǫhdaˀgeh	(e)tni+ahǫhd+aˀgeh
agyahǫhdaˀgeh	(y)akni+ahǫhd+aˀgeh
agwahǫhdaˀgeh	(y)agwa+ahǫhd+aˀgeh
jahǫhdaˀgeh	(h)sni+ahǫhd+aˀgeh
hahǫhdaˀgeh	ha+ahǫhd+aˀgeh
wahǫhdaˀgeh	w+ahǫhd+aˀgeh
gaǫhǫhdaˀgeh	gaǫ+ahǫhd+aˀgeh
gahsiˀdaˀgeh	g+ahsiˀd+aˀgeh
gyahsiˀdaˀgeh	(e)kni+ahsiˀd+aˀgeh
dwahsiˀdaˀgeh	(e)dwa+ahsiˀd+aˀgeh
sahsiˀdaˀgeh	(h)s+ahsiˀd+aˀgeh
swahsiˀdaˀgeh	(h)swa+ahsiˀd+aˀgeh
ǫhsiˀdaˀgeh	(y)ǫ+ahsiˀd+aˀgeh
hęnahsiˀdaˀgeh	hadi+ahsiˀd+aˀgeh
gęnahsiˀdaˀgeh	gadi+ahsiˀd+aˀgeh
tnihnaˀgeh	(e)tni+ihn+aˀgeh
aknihnaˀgeh	(y)akni+ihn+aˀgeh
agwęhnaˀgeh	(y)agwa+ihn+aˀgeh
snihnaˀgeh	(h)sni+ihn+aˀgeh
hęhnaˀgeh	ha+ihn+aˀgeh
gęhnaˀgeh	ga+ihn+aˀgeh
gaehnaˀgeh	gae+ihn+aˀgeh
gǫtsaˀgeh	g+ǫts+aˀgeh
knǫtsaˀgeh	(e)kni+ǫts+aˀgeh
gyǫtsaˀgeh	(e)dwa+ǫts+aˀgeh
sǫtsaˀgeh	(h)s+ǫts+aˀgeh
jǫtsaˀgeh	(h)swa+ǫts+aˀgeh
agǫtsaˀgeh	(y)ag+ǫts+aˀgeh
hęnǫtsaˀgeh	hadi+ǫts+aˀgeh
gęnǫtsaˀgeh	gadi+ǫts+aˀgeh
tniwęˀnahsaˀgeh	(e)tni+węˀnahs+aˀgeh
akniwęˀnahsaˀgeh	(y)akni+węˀnahs+aˀgeh
agwawęˀnahsaˀgeh	(y)agwa+węˀnahs+aˀgeh
sniwęˀnahsaˀgeh	(eh)sni+węˀnahs+aˀgeh
hawęˀnahsaˀgeh	ha+węˀnahs+aˀgeh
gawęˀnahsaˀgeh	ga+węˀnahs+aˀgeh
gaewęˀnahsaˀgeh	gae+węˀnahs+aˀgeh
gegahaˀgeh	g+gah+aˀgeh
knigahaˀgeh	(e)kni+gah+aˀgeh

Inalienable Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
aknigahaˀgeh	(y)akni+gah+aˀgeh		dwagahaˀgeh	(e)dwa+gah+aˀgeh
agwagahaˀgeh	(y)agwa+gah+aˀgeh		segahaˀgeh	(eh)s+gah+aˀgeh
snigahaˀgeh	(eh)sni+gah+aˀgeh		swagahaˀgeh	(eh)swa+gah+aˀgeh
hagahaˀgeh	ha+gah+aˀgeh		egahaˀgeh	(y)e+gah+aˀgeh
gagahaˀgeh	ga+gah+aˀgeh		hadigahaˀgeh	hadi+gah+aˀgeh
gaegahaˀgeh	gae+gah+aˀgeh		gadigahaˀgeh	gadi+gah+aˀgeh
ketgaˀaˀgeh	g+hetgaˀ+aˀgeh		tnihetgaˀaˀgeh	(e)tni+hetgaˀ+aˀgeh
knihetgaˀaˀgeh	(e)kni+hetgaˀ+aˀgeh		aknihetgaˀaˀgeh	(y)akni+hetgaˀ+aˀgeh
dwahetgaˀaˀgeh	(e)dwa+hetgaˀ+aˀgeh		agwahetgaˀaˀgeh	(y)agwa+hetgaˀ+aˀgeh
setgaˀaˀgeh	(eh)s+hetgaˀ+aˀgeh		snihetgaˀaˀgeh	(eh)sni+hetgaˀ+aˀgeh
swahetgaˀaˀgeh	(eh)swa+hetgaˀ+aˀgeh		hahetgaˀaˀgeh	ha+hetgaˀ+aˀgeh
ehetgaˀaˀgeh	(y)e+hetgaˀ+aˀgeh		gahetgaˀaˀgeh	ga+hetgaˀ+aˀgeh
hadihetgaˀaˀgeh	hadi+hetgaˀ+aˀgeh		gaehetgaˀaˀgeh	gae+hetgaˀ+aˀgeh
gadihetgaˀaˀgeh	gadi+hetgaˀ+aˀgeh		geˀnhǫhsgaːˀgeh	g+ˀnhǫhsgaː+aˀgeh
tniˀnhǫhsgaːˀgeh	(e)tni+ˀnhǫhsgaː+aˀgeh		kniˀnhǫhsgaːˀgeh	(e)kni+ˀnhǫhsgaː+aˀgeh
akniˀnhǫhsgaːˀgeh	(y)akni+ˀnhǫhsgaː+aˀgeh		dwaˀnhǫhsgaːˀgeh	(e)dwa+ˀnhǫhsgaː+aˀgeh
agwaˀnhǫhsgaːˀgeh	(y)agwa+ˀnhǫhsgaː+aˀgeh		seˀnhǫhsgaːˀgeh	(eh)s+ˀnhǫhsgaː+aˀgeh
sniˀnhǫhsgaːˀgeh	(eh)sni+ˀnhǫhsgaː+aˀgeh		swaˀnhǫhsgaːˀgeh	(eh)swa+ˀnhǫhsgaː+aˀgeh
haˀnhǫhsgaːˀgeh	ha+ˀnhǫhsgaː+aˀgeh		eˀnhǫhsgaːˀgeh	(y)e+ˀnhǫhsgaː+aˀgeh
gaˀnhǫhsgaːˀgeh	ga+ˀnhǫhsgaː+aˀgeh		hadiˀnhǫhsgaːˀgeh	hadi+ˀnhǫhsgaː+aˀgeh
gaeˀnhǫhsgaːˀgeh	gae+ˀnhǫhsgaː+aˀgeh		gadiˀnhǫhsgaːˀgeh	gadi+ˀnhǫhsgaː+aˀgeh
geˀahsaˀgeh	g+ˀahs+aˀgeh		tniˀahsaˀgeh	(e)tni+ˀahs+aˀgeh
kniˀahsaˀgeh	(e)kni+ˀahs+aˀgeh		akniˀahsaˀgeh	(y)akni+ˀahs+aˀgeh
dwaˀahsaˀgeh	(e)dwa+ˀahs+aˀgeh		agwaˀahsaˀgeh	(y)agwa+ˀahs+aˀgeh
seˀahsaˀgeh	(eh)s+ˀahs+aˀgeh		sniˀahsaˀgeh	(eh)sni+ˀahs+aˀgeh
swaˀahsaˀgeh	(eh)swa+ˀahs+aˀgeh		haˀahsaˀgeh	ha+ˀahs+aˀgeh
eˀahsaˀgeh	(y)e+ˀahs+aˀgeh		gaˀahsaˀgeh	ga+ˀahs+aˀgeh
hadiˀahsaˀgeh	hadi+ˀahs+aˀgeh		gaeˀahsaˀgeh	gae+ˀahs+aˀgeh
gadiˀahsaˀgeh	gadi+ˀahs+aˀgeh		gekseˀdaˀgeh	g+kseˀd+aˀgeh
tnikseˀdaˀgeh	(e)tni+kseˀd+aˀgeh		knikseˀdaˀgeh	(e)kni+kseˀd+aˀgeh
aknikseˀdaˀgeh	(y)akni+kseˀd+aˀgeh		dwakseˀdaˀgeh	(e)dwa+kseˀd+aˀgeh
agwakseˀdaˀgeh	(y)agwa+kseˀd+aˀgeh		sekseˀdaˀgeh	(eh)s+kseˀd+aˀgeh
snikseˀdaˀgeh	(eh)sni+kseˀd+aˀgeh		swakseˀdaˀgeh	(eh)swa+kseˀd+aˀgeh
hakseˀdaˀgeh	ha+kseˀd+aˀgeh		ekseˀdaˀgeh	(y)e+kseˀd+aˀgeh
gakseˀdaˀgeh	ga+kseˀd+aˀgeh		hadikseˀdaˀgeh	hadi+kseˀd+aˀgeh
gaekseˀdaˀgeh	gae+kseˀd+aˀgeh		gadikseˀdaˀgeh	gadi+kseˀd+aˀgeh
knǫˀaːˀgeh	g+nǫˀaː+aˀgeh		tninǫˀaːˀgeh	(e)tni+nǫˀaː+aˀgeh
kninǫˀaːˀgeh	(e)kni+nǫˀaː+aˀgeh		akninǫˀaːˀgeh	(y)akni+nǫˀaː+aˀgeh

Inalienable Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
dwanǫ̃ʔa:ʔgeh	(e)dwa+nǫ̃ʔa:+aʔgeh		agwanǫ̃ʔa:ʔgeh	(y)agwa+nǫ̃ʔa:+aʔgeh
snǫ̃ʔa:ʔgeh	(eh)s+nǫ̃ʔa:+aʔgeh		sninǫ̃ʔa:ʔgeh	(eh)sni+nǫ̃ʔa:+aʔgeh
swanǫ̃ʔa:ʔgeh	(eh)swa+nǫ̃ʔa:+aʔgeh		hanǫ̃ʔa:ʔgeh	ha+nǫ̃ʔa:+aʔgeh
enǫ̃ʔa:ʔgeh	(y)e+nǫ̃ʔa:+aʔgeh		ganǫ̃ʔa:ʔgeh	ga+nǫ̃ʔa:+aʔgeh
hadinǫ̃ʔa:ʔgeh	hadi+nǫ̃ʔa:+aʔgeh		gaenǫ̃ʔa:ʔgeh	gae+nǫ̃ʔa:+aʔgeh
gadinǫ̃ʔa:ʔgeh	gadi+nǫ̃ʔa:+aʔgeh		gragwahdaʔgeh	g+ragwahd+aʔgeh
tniagwahdaʔgeh	(e)tni+ragwahd+aʔgeh		kniagwahdaʔgeh	(e)kni+ragwahd+aʔgeh
akniagwahdaʔgeh	(y)akni+ragwahd+aʔgeh		dwa:gwahdaʔgeh	(e)dwa+ragwahd+aʔgeh
agwa:gwahdaʔgeh	(y)agwa+ragwahd+aʔgeh		sragwahdaʔgeh	(eh)s+ragwahd+aʔgeh
sniagwahdaʔgeh	(eh)sni+ragwahd+aʔgeh		swa:gwahdaʔgeh	(eh)swa+ragwahd+aʔgeh
ha:gwahdaʔgeh	ha+ragwahd+aʔgeh		eagwahdaʔgeh	(y)e+ragwahd+aʔgeh
ga:gwahdaʔgeh	ga+ragwahd+aʔgeh		hadiagwahdaʔgeh	hadi+ragwahd+aʔgeh
gaeagwahdaʔgeh	gae+ragwahd+aʔgeh		gadiagwahdaʔgeh	gadi+ragwahd+aʔgeh
gwę̃ʔyǫ̃hga:ʔgeh	g+wę̃ʔyǫ̃hga:+aʔgeh		tniwę̃ʔyǫ̃hga:ʔgeh	(e)tni+wę̃ʔyǫ̃hga:+aʔgeh
kniwę̃ʔyǫ̃hga:ʔgeh	(e)kni+wę̃ʔyǫ̃hga:+aʔgeh		akniwę̃ʔyǫ̃hga:ʔgeh	(y)akni+wę̃ʔyǫ̃hga:+aʔgeh
dwawę̃ʔyǫ̃hga:ʔgeh	(e)dwa+wę̃ʔyǫ̃hga:+aʔgeh		agwawę̃ʔyǫ̃hga:ʔgeh	(y)agwa+wę̃ʔyǫ̃hga:+aʔgeh
swę̃ʔyǫ̃hga:ʔgeh	(eh)s+wę̃ʔyǫ̃hga:+aʔgeh		sniwę̃ʔyǫ̃hga:ʔgeh	(eh)sni+wę̃ʔyǫ̃hga:+aʔgeh
swawę̃ʔyǫ̃hga:ʔgeh	(eh)swa+wę̃ʔyǫ̃hga:+aʔgeh		hawę̃ʔyǫ̃hga:ʔgeh	ha+wę̃ʔyǫ̃hga:+aʔgeh
ewę̃ʔyǫ̃hga:ʔgeh	(y)e+wę̃ʔyǫ̃hga:+aʔgeh		gawę̃ʔyǫ̃hga:ʔgeh	ga+wę̃ʔyǫ̃hga:+aʔgeh
hadiwę̃ʔyǫ̃hga:ʔgeh	hadi+wę̃ʔyǫ̃hga:+aʔgeh		gaewę̃ʔyǫ̃hga:ʔgeh	gae+wę̃ʔyǫ̃hga:+aʔgeh
gadiwę̃ʔyǫ̃hga:ʔgeh	gadi+wę̃ʔyǫ̃hga:+aʔgeh		gejaohoʔgwaʔgeh	g+jaohoʔgw+aʔgeh
tnijaohoʔgwaʔgeh	(e)tni+jaohoʔgw+aʔgeh		knijaohoʔgwaʔgeh	(e)kni+jaohoʔgw+aʔgeh
aknijaohoʔgwaʔgeh	(y)akni+jaohoʔgw+aʔgeh		dwajaohoʔgwaʔgeh	(e)dwa+jaohoʔgw+aʔgeh
agwajaohoʔgwaʔgeh	(y)agwa+jaohoʔgw+aʔgeh		sejaohoʔgwaʔgeh	(eh)s+jaohoʔgw+aʔgeh
snijaohoʔgwaʔgeh	(eh)sni+jaohoʔgw+aʔgeh		swajaohoʔgwaʔgeh	(eh)swa+jaohoʔgw+aʔgeh
hajaohoʔgwaʔgeh	ha+jaohoʔgw+aʔgeh		ejaohoʔgwaʔgeh	(y)e+jaohoʔgw+aʔgeh
gajaohoʔgwaʔgeh	ga+jaohoʔgw+aʔgeh		hadijaohoʔgwaʔgeh	hadi+jaohoʔgw+aʔgeh
gaejaohoʔgwaʔgeh	gae+jaohoʔgw+aʔgeh		gadijaohoʔgwaʔgeh	gadi+jaohoʔgw+aʔgeh
gyaʔgaːʔgeh	g+yaʔga:+aʔgeh		tniyaʔgaːʔgeh	(e)tni+yaʔga:+aʔgeh
kniyaʔgaːʔgeh	(e)kni+yaʔga:+aʔgeh		akniyaʔgaːʔgeh	(y)akni+yaʔga:+aʔgeh
dwayaʔgaːʔgeh	(e)dwa+yaʔga:+aʔgeh		agwayaʔgaːʔgeh	(y)agwa+yaʔga:+aʔgeh
syaʔgaːʔgeh	(eh)s+yaʔga:+aʔgeh		sniyaʔgaːʔgeh	(eh)sni+yaʔga:+aʔgeh
swayaʔgaːʔgeh	(eh)swa+yaʔga:+aʔgeh		hayaʔgaːʔgeh	ha+yaʔga:+aʔgeh
eyaʔgaːʔgeh	(y)e+yaʔga:+aʔgeh		gayaʔgaːʔgeh	ga+yaʔga:+aʔgeh
hadiyaʔgaːʔgeh	hadi+yaʔga:+aʔgeh		gaeyaʔgaːʔgeh	gae+yaʔga:+aʔgeh
gadiyaʔgaːʔgeh	gadi+yaʔga:+aʔgeh			

Table C.6: Unpossessed Inalienable Nouns

Surface	**Underlying**
ohǫhda˺	o+ahǫhd+a˺
ohna˺	o+ihn+a˺
owę˺nahsa˺	o+wę˺nahs+a˺
ohetga˺a˺	o+hetga˺+a˺
o˺ahsa˺	o+˺ahs+a˺
onǫ˺a:˺	o+nǫ˺a:+a˺
owę˺yǫhga:˺	o+wę˺yǫhga:+a˺
oya˺ga:˺	o+ya˺ga:+a˺

Surface	**Underlying**
ohsi˺da˺	o+ahsi˺d+a˺
ǫtsa˺	o+ǫts+a˺
ogaha˺	o+gah+a˺
o˺nhǫhsga:˺	o+˺nhǫhsga:+a˺
okse˺da˺	o+kse˺d+a˺
oagwahda˺	o+ragwahd+a˺
ojaoho˺gwa˺	o+jaoho˺gw+a˺

C.2 Data for the Concrete Approach

Table C.7: Unpossessed Basic Nouns

Surface	Underlying
hǫnaˀdaˀ	hǫnaˀd+aˀ
odi:	o+di:
gaˀwahsdaˀ	ga+ˀwahsd+aˀ
oˀwahsdaˀ	o+ˀwahsd+aˀ
gahsdagwaˀ	ga+hsdagw+aˀ
ohsdagwaˀ	o+hsdagw+aˀ
adǫhneˀtsaˀ	a+adǫhneˀts+aˀ
awęnohgraˀ	a+awęnohgr+aˀ
oˀnhahgyaˀ	o+ˀnhahgy+aˀ
ohyaˀ	o+ahy+aˀ
ohwęhsdaˀ	o+hwęhsd+aˀ
okwaˀ	o+kw+aˀ
oihwaˀ	o+rihw+aˀ
otsgraˀ	o+tsgr+aˀ
owiyaˀ	o+wiy+aˀ
oyaˀdaˀ	o+yaˀd+aˀ
ga:gwa:ˀ	ga+Cagwa:+aˀ
gahsǫwahdaˀ	ga+hsǫwahd+aˀ
gajęˀ	ga+ję+aˀ
ganaˀjaˀ	ga+naˀj+aˀ
gatgęhetsaˀ	ga+tgęhets+aˀ

Surface	Underlying
hsgwaęˀdaˀ	hsgwaęˀd+aˀ
oji:s	o+ji:s
gahehnaˀ	ga+hehn+aˀ
ohehnaˀ	o+hehn+aˀ
gajiˀgwaˀ	ga+jiˀgw+aˀ
ojiˀgwaˀ	o+jiˀgw+aˀ
odǫhneˀtsaˀ	o+adǫhneˀts+aˀ
adraˀswaˀ	a+adraˀsw+aˀ
onhahdaˀ	o+nhahd+aˀ
oga:ˀ	o+ga:+aˀ
oˀdaˀ	o+iˀd+aˀ
onaˀda:ˀ	o+naˀda:+aˀ
osehdaˀ	o+sehd+aˀ
owa:ˀ	o+wa:+aˀ
oyaˀ	o+y+aˀ
gaˀwahsha:ˀ	ga+ˀwahsha:+aˀ
gagaˀdaˀ	ga+gaˀd+aˀ
gętsga:ˀ	ga+itsga:+aˀ
gakwaˀ	ga+kw+aˀ
gaǫdaˀ	ga+rǫd+aˀ
gaya:ˀ	ga+ya:+aˀ

Table C.8: Possessed Basic Nouns

Surface	Underlying
akǫnaˀdaˀ	(w)ak+hǫnaˀd+aˀ
ǫgwahǫnaˀdaˀ	(y)ǫgwa+hǫnaˀd+aˀ
snihǫnaˀdaˀ	sni+hǫnaˀd+aˀ
hohǫnaˀdaˀ	ho+hǫnaˀd+aˀ
ohǫnaˀdaˀ	(y)o+hǫnaˀd+aˀ
godihǫnaˀdaˀ	(ya)godi+hǫnaˀd+aˀ
agehsgwaęˀdaˀ	(w)age+hsgwaęˀd+aˀ
ǫgwahsgwaęˀdaˀ	(y)ǫgwa+hsgwaęˀd+aˀ
snihsgwaęˀdaˀ	sni+hsgwaęˀd+aˀ
hohsgwaęˀdaˀ	ho+hsgwaęˀd+aˀ
ohsgwaęˀdaˀ	(y)o+hsgwaęˀd+aˀ
godihsgwaęˀdaˀ	(ya)godi+hsgwaęˀd+aˀ
akwahsdaˀ	(w)ak+ˀwahsd+aˀ
ǫgwaˀwahsdaˀ	(y)ǫgwa+ˀwahsd+aˀ
sniˀwahsdaˀ	sni+ˀwahsd+aˀ
hoˀwahsdaˀ	ho+ˀwahsd+aˀ
oˀwahsdaˀ	(y)o+ˀwahsd+aˀ
godiˀwahsdaˀ	(ya)godi+ˀwahsd+aˀ
akehnaˀ	(w)ak+hehn+aˀ
ǫgwahehnaˀ	(y)ǫgwa+hehn+aˀ
snihehnaˀ	sni+hehn+aˀ
hohehnaˀ	ho+hehn+aˀ
ohehnaˀ	(y)o+hehn+aˀ
godihehnaˀ	(ya)godi+hehn+aˀ
agejihoha:ˀ	(w)age+jihoha:+aˀ
ǫgwajihoha:ˀ	(y)ǫgwa+jihoha:+aˀ
snijihoha:ˀ	sni+jihoha:+aˀ
hojihoha:ˀ	ho+jihoha:+aˀ
ojihoha:ˀ	(y)o+jihoha:+aˀ
godijihoha:ˀ	(ya)godi+jihoha:+aˀ
agadǫhneˀtsaˀ	(w)ag+adǫhneˀts+aˀ
ǫgwadǫhneˀtsaˀ	(y)ǫgw+adǫhneˀts+aˀ
jadǫhneˀtsaˀ	j+adǫhneˀts+aˀ
hodǫhneˀtsaˀ	ho+adǫhneˀts+aˀ
odǫhneˀtsaˀ	(y)o+adǫhneˀts+aˀ
gonadǫhneˀtsaˀ	(ya)gon+adǫhneˀts+aˀ
ageˀnhǫhsaˀ	(w)age+ˀnhǫhs+aˀ
ǫgwaˀnhǫhsaˀ	(y)ǫgwa+ˀnhǫhs+aˀ
sniˀnhǫhsaˀ	sni+ˀnhǫhs+aˀ

Surface	Underlying
ǫknihǫnaˀdaˀ	(y)ǫkni+hǫnaˀd+aˀ
sahǫnaˀdaˀ	sa+hǫnaˀd+aˀ
swahǫnaˀdaˀ	swa+hǫnaˀd+aˀ
gohǫnaˀdaˀ	(ya)go+hǫnaˀd+aˀ
hodihǫnaˀdaˀ	hodi+hǫnaˀd+aˀ
odihǫnaˀdaˀ	(y)odi+hǫnaˀd+aˀ
ǫknihsgwaęˀdaˀ	(y)ǫkni+hsgwaęˀd+aˀ
sahsgwaęˀdaˀ	sa+hsgwaęˀd+aˀ
swahsgwaęˀdaˀ	swa+hsgwaęˀd+aˀ
gohsgwaęˀdaˀ	(ya)go+hsgwaęˀd+aˀ
hodihsgwaęˀdaˀ	hodi+hsgwaęˀd+aˀ
odihsgwaęˀdaˀ	(y)odi+hsgwaęˀd+aˀ
ǫkniˀwahsdaˀ	(y)ǫkni+ˀwahsd+aˀ
saˀwahsdaˀ	sa+ˀwahsd+aˀ
swaˀwahsdaˀ	swa+ˀwahsd+aˀ
goˀwahsdaˀ	(ya)go+ˀwahsd+aˀ
hodiˀwahsdaˀ	hodi+ˀwahsd+aˀ
odiˀwahsdaˀ	(y)odi+ˀwahsd+aˀ
ǫknihehnaˀ	(y)ǫkni+hehn+aˀ
sahehnaˀ	sa+hehn+aˀ
swahehnaˀ	swa+hehn+aˀ
gohehnaˀ	(ya)go+hehn+aˀ
hodihehnaˀ	hodi+hehn+aˀ
odihehnaˀ	(y)odi+hehn+aˀ
ǫknijihoha:ˀ	(y)ǫkni+jihoha:+aˀ
sajihoha:ˀ	sa+jihoha:+aˀ
swajihoha:ˀ	swa+jihoha:+aˀ
gojihoha:ˀ	(ya)go+jihoha:+aˀ
hodijihoha:ˀ	hodi+jihoha:+aˀ
odijihoha:ˀ	(y)odi+jihoha:+aˀ
ǫgyadǫhneˀtsaˀ	(y)ǫgy+adǫhneˀts+aˀ
sadǫhneˀtsaˀ	s+adǫhneˀts+aˀ
swadǫhneˀtsaˀ	sw+adǫhneˀts+aˀ
godǫhneˀtsaˀ	(ya)go+adǫhneˀts+aˀ
honadǫhneˀtsaˀ	hon+adǫhneˀts+aˀ
onadǫhneˀtsaˀ	(y)on+adǫhneˀts+aˀ
ǫkniˀnhǫhsaˀ	(y)ǫkni+ˀnhǫhs+aˀ
saˀnhǫhsaˀ	sa+ˀnhǫhs+aˀ
swaˀnhǫhsaˀ	swa+ˀnhǫhs+aˀ

Basic Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
ho˥nhǫhsa˥	ho+˥nhǫhs+a˥		go˥nhǫhsa˥	(ya)go+˥nhǫhs+a˥
o˥nhǫhsa˥	(y)o+˥nhǫhs+a˥		hodi˥nhǫhsa˥	hodi+˥nhǫhs+a˥
godi˥nhǫhsa˥	(ya)godi+˥nhǫhs+a˥		odi˥nhǫhsa˥	(y)odi+˥nhǫhs+a˥
age˥ǫhgwa:˥	(w)age+˥ǫhgwa:+a˥		ǫkni˥ǫhgwa:˥	(y)ǫkni+˥ǫhgwa:+a˥
ǫgwa˥ǫhgwa:˥	(y)ǫgwa+˥ǫhgwa:+a˥		sa˥ǫhgwa:˥	sa+˥ǫhgwa:+a˥
sni˥ǫhgwa:˥	sni+˥ǫhgwa:+a˥		swa˥ǫhgwa:˥	swa+˥ǫhgwa:+a˥
ho˥ǫhgwa:˥	ho+˥ǫhgwa:+a˥		go˥ǫhgwa:˥	(ya)go+˥ǫhgwa:+a˥
o˥ǫhgwa:˥	(y)o+˥ǫhgwa:+a˥		hodi˥ǫhgwa:˥	hodi+˥ǫhgwa:+a˥
godi˥ǫhgwa:˥	(ya)godi+˥ǫhgwa:+a˥		odi˥ǫhgwa:˥	(y)odi+˥ǫhgwa:+a˥
agenhahda˥	(w)age+nhahd+a˥		ǫkninhahda˥	(y)ǫkni+nhahd+a˥
ǫgwanhahda˥	(y)ǫgwa+nhahd+a˥		sanhahda˥	sa+nhahd+a˥
sninhahda˥	sni+nhahd+a˥		swanhahda˥	swa+nhahd+a˥
honhahda˥	ho+nhahd+a˥		gonhahda˥	(ya)go+nhahd+a˥
onhahda˥	(y)o+nhahd+a˥		hodinhahda˥	hodi+nhahd+a˥
godinhahda˥	(ya)godi+nhahd+a˥		odinhahda˥	(y)odi+nhahd+a˥
agega˥da˥	(w)age+ga˥d+a˥		ǫkniga˥da˥	(y)ǫkni+ga˥d+a˥
ǫgwaga˥da˥	(y)ǫgwa+ga˥d+a˥		saga˥da˥	sa+ga˥d+a˥
sniga˥da˥	sni+ga˥d+a˥		swaga˥da˥	swa+ga˥d+a˥
hoga˥da˥	ho+ga˥d+a˥		goga˥da˥	(ya)go+ga˥d+a˥
oga˥da˥	(y)o+ga˥d+a˥		hodiga˥da˥	hodi+ga˥d+a˥
godiga˥da˥	(ya)godi+ga˥d+a˥		odiga˥da˥	(y)odi+ga˥d+a˥
akakda˥	(w)ak+hakd+a˥		ǫknihakda˥	(y)ǫkni+hakd+a˥
ǫgwahakda˥	(y)ǫgwa+hakd+a˥		sahakda˥	sa+hakd+a˥
snihakda˥	sni+hakd+a˥		swahakda˥	swa+hakd+a˥
hohakda˥	ho+hakd+a˥		gohakda˥	(ya)go+hakd+a˥
ohakda˥	(y)o+hakd+a˥		hodihakda˥	hodi+hakd+a˥
godihakda˥	(ya)godi+hakd+a˥		odihakda˥	(y)odi+hakd+a˥
agi˥dǫhgwa˥	(w)ag+i˥dǫhgw+a˥		ǫkni˥dǫhgwa˥	(y)ǫkn+i˥dǫhgw+a˥
ǫgwę˥dǫhgwa˥	(y)ǫgwę+i˥dǫhgw+a˥		sę˥dǫhgwa˥	sę+i˥dǫhgw+a˥
sni˥dǫhgwa˥	sn+i˥dǫhgw+a˥		swę˥dǫhgwa˥	swę+i˥dǫhgw+a˥
ho˥dǫhgwa˥	ho+i˥dǫhgw+a˥		go˥dǫhgwa˥	(ya)go+i˥dǫhgw+a˥
o˥dǫhgwa˥	(y)o+i˥dǫhgw+a˥		hodi˥dǫhgwa˥	hod+i˥dǫhgw+a˥
godi˥dǫhgwa˥	(ya)god+i˥dǫhgw+a˥		odi˥dǫhgwa˥	(y)od+i˥dǫhgw+a˥
agejihsǫda:˥	(w)age+jihsǫda:+a˥		ǫknijihsǫda:˥	(y)ǫkni+jihsǫda:+a˥
ǫgwajihsǫda:˥	(y)ǫgwa+jihsǫda:+a˥		sajihsǫda:˥	sa+jihsǫda:+a˥
snijihsǫda:˥	sni+jihsǫda:+a˥		swajihsǫda:˥	swa+jihsǫda:+a˥
hojihsǫda:˥	ho+jihsǫda:+a˥		gojihsǫda:˥	(ya)go+jihsǫda:+a˥
ojihsǫda:˥	(y)o+jihsǫda:+a˥		hodijihsǫda:˥	hodi+jihsǫda:+a˥

Basic Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
godijihsǫda:ˀ	(ya)godi+jihsǫda:+aˀ		odijihsǫda:ˀ	(y)odi+jihsǫda:+aˀ
agekdehaˀ	(w)age+kdeh+aˀ		ǫknikdehaˀ	(y)ǫkni+kdeh+aˀ
ǫgwakdehaˀ	(y)ǫgwa+kdeh+aˀ		sakdehaˀ	sa+kdeh+aˀ
snikdehaˀ	sni+kdeh+aˀ		swakdehaˀ	swa+kdeh+aˀ
hokdehaˀ	ho+kdeh+aˀ		gokdehaˀ	(ya)go+kdeh+aˀ
okdehaˀ	(y)o+kdeh+aˀ		hodikdehaˀ	hodi+kdeh+aˀ
godikdehaˀ	(ya)godi+kdeh+aˀ		odikdehaˀ	(y)odi+kdeh+aˀ
agrihwaˀ	(w)ag+rihw+aˀ		ǫkni:hwaˀ	(y)ǫkni+rihw+aˀ
ǫgwaihwaˀ	(y)ǫgwa+rihw+aˀ		saihwaˀ	sa+rihw+aˀ
sni:hwaˀ	sni+rihw+aˀ		swaihwaˀ	swa+rihw+aˀ
hoihwaˀ	ho+rihw+aˀ		goihwaˀ	(ya)go+rihw+aˀ
oihwaˀ	(y)o+rihw+aˀ		hodi:hwaˀ	hodi+rihw+aˀ
godi:hwaˀ	(ya)godi+rihw+aˀ		odi:hwaˀ	(y)odi+rihw+aˀ
agesehdaˀ	(w)age+sehd+aˀ		ǫknisehdaˀ	(y)ǫkni+sehd+aˀ
ǫgwasehdaˀ	(y)ǫgwa+sehd+aˀ		sasehdaˀ	sa+sehd+aˀ
snisehdaˀ	sni+sehd+aˀ		swasehdaˀ	swa+sehd+aˀ
hosehdaˀ	ho+sehd+aˀ		gosehdaˀ	(ya)go+sehd+aˀ
osehdaˀ	(y)o+sehd+aˀ		hodisehdaˀ	hodi+sehd+aˀ
godisehdaˀ	(ya)godi+sehd+aˀ		odisehdaˀ	(y)odi+sehd+aˀ
agetsehsdaˀ	(w)age+tsehsd+aˀ		ǫknitsehsdaˀ	(y)ǫkni+tsehsd+aˀ
ǫgwatsehsdaˀ	(y)ǫgwa+tsehsd+aˀ		satsehsdaˀ	sa+tsehsd+aˀ
snitsehsdaˀ	sni+tsehsd+aˀ		swatsehsdaˀ	swa+tsehsd+aˀ
hotsehsdaˀ	ho+tsehsd+aˀ		gotsehsdaˀ	(ya)go+tsehsd+aˀ
otsehsdaˀ	(y)o+tsehsd+aˀ		hoditsehsdaˀ	hodi+tsehsd+aˀ
goditsehsdaˀ	(ya)godi+tsehsd+aˀ		oditsehsdaˀ	(y)odi+tsehsd+aˀ
agwaˀwihsdaˀ	(w)ag+waˀwihsd+aˀ		ǫkniwaˀwihsdaˀ	(y)ǫkni+waˀwihsd+aˀ
ǫgwawaˀwihsdaˀ	(y)ǫgwa+waˀwihsd+aˀ		sawaˀwihsdaˀ	sa+waˀwihsd+aˀ
sniwaˀwihsdaˀ	sni+waˀwihsd+aˀ		swawaˀwihsdaˀ	swa+waˀwihsd+aˀ
howaˀwihsdaˀ	ho+waˀwihsd+aˀ		gowaˀwihsdaˀ	(ya)go+waˀwihsd+aˀ
owaˀwihsdaˀ	(y)o+waˀwihsd+aˀ		hodiwaˀwihsdaˀ	hodi+waˀwihsd+aˀ
godiwaˀwihsdaˀ	(ya)godi+waˀwihsd+aˀ		odiwaˀwihsdaˀ	(y)odi+waˀwihsd+aˀ
agyǫˀdaˀ	(w)ag+yǫˀd+aˀ		ǫkniyǫˀdaˀ	(y)ǫkni+yǫˀd+aˀ
ǫgwayǫˀdaˀ	(y)ǫgwa+yǫˀd+aˀ		sayǫˀdaˀ	sa+yǫˀd+aˀ
sniyǫˀdaˀ	sni+yǫˀd+aˀ		swayǫˀdaˀ	swa+yǫˀd+aˀ
hoyǫˀdaˀ	ho+yǫˀd+aˀ		goyǫˀdaˀ	(ya)go+yǫˀd+aˀ
oyǫˀdaˀ	(y)o+yǫˀd+aˀ		hodiyǫˀdaˀ	hodi+yǫˀd+aˀ
godiyǫˀdaˀ	(ya)godi+yǫˀd+aˀ		odiyǫˀdaˀ	(y)odi+yǫˀd+aˀ
agihsdaˀ	(w)ag+Cihsd+aˀ		ǫkni:hsdaˀ	(y)ǫkni+Cihsd+aˀ

Basic Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
ǫgwaihsda˥	(y)ǫgwa+Cihsd+a˥		saihsda˥	sa+Cihsd+a˥
sni:hsda˥	sni+Cihsd+a˥		swaihsda˥	swa+Cihsd+a˥
hoihsda˥	ho+Cihsd+a˥		goihsda˥	(ya)go+Cihsd+a˥
oihsda˥	(y)o+Cihsd+a˥		hodi:hsda˥	hodi+Cihsd+a˥
godi:hsda˥	(ya)godi+Cihsd+a˥		odi:hsda˥	(y)odi+Cihsd+a˥
agatsgę˥da˥	(w)ag+Catsgę˥d+a˥		ǫkniatsgę˥da˥	(y)ǫkni+Catsgę˥d+a˥
ǫgwa:tsgę˥da˥	(y)ǫgwa+Catsgę˥d+a˥		sa:tsgę˥da˥	sa+Catsgę˥d+a˥
sniatsgę˥da˥	sni+Catsgę˥d+a˥		swa:tsgę˥da˥	swa+Catsgę˥d+a˥
hoatsgę˥da˥	ho+Catsgę˥d+a˥		goatsgę˥da˥	(ya)go+Catsgę˥d+a˥
oatsgę˥da˥	(y)o+Catsgę˥d+a˥		hodiatsgę˥da˥	hodi+Catsgę˥d+a˥
godiatsgę˥da˥	(ya)godi+Catsgę˥d+a˥		odiatsgę˥da˥	(y)odi+Catsgę˥d+a˥
age˥ahdra˥	(w)age+˥ahdr+a˥		ǫkni˥ahdra˥	(y)ǫkni+˥ahdr+a˥
ǫgwa˥ahdra˥	(y)ǫgwa+˥ahdr+a˥		sa˥ahdra˥	sa+˥ahdr+a˥
sni˥ahdra˥	sni+˥ahdr+a˥		swa˥ahdra˥	swa+˥ahdr+a˥
ho˥ahdra˥	ho+˥ahdr+a˥		go˥ahdra˥	(ya)go+˥ahdr+a˥
o˥ahdra˥	(y)o+˥ahdr+a˥		hodi˥ahdra˥	hodi+˥ahdr+a˥
godi˥ahdra˥	(ya)godi+˥ahdr+a˥		odi˥ahdra˥	(y)odi+˥ahdr+a˥
agitsga:˥	(w)ag+itsga:+a˥		ǫknitsga:˥	(y)ǫkn+itsga:+a˥
ǫgwętsga:˥	(y)ǫgwę+itsga:+a˥		sętsga:˥	sę+itsga:+a˥
snitsga:˥	sn+itsga:+a˥		swętsga:˥	swę+itsga:+a˥
hotsga:˥	ho+itsga:+a˥		gotsga:˥	(ya)go+itsga:+a˥
otsga:˥	(y)o+itsga:+a˥		hoditsga:˥	hod+itsga:+a˥
goditsga:˥	(ya)god+itsga:+a˥		oditsga:˥	(y)od+itsga:+a˥
agrękna˥	(w)ag+rękn+a˥		ǫknięna˥	(y)ǫkni+rękn+a˥
ǫgwaękna˥	(y)ǫgwa+rękn+a˥		saękna˥	sa+rękn+a˥
snięna˥	sni+rękn+a˥		swaękna˥	swa+rękn+a˥
hoękna˥	ho+rękn+a˥		goękna˥	(ya)go+rękn+a˥
oękna˥	(y)o+rękn+a˥		hodięna˥	hodi+rękn+a˥
godięna˥	(ya)godi+rękn+a˥		odięna˥	(y)odi+rękn+a˥
agrihwihsa˥	(w)ag+rihwihs+a˥		ǫkni:hwihsa˥	(y)ǫkni+rihwihs+a˥
ǫgwaihwihsa˥	(y)ǫgwa+rihwihs+a˥		saihwihsa˥	sa+rihwihs+a˥
sni:hwihsa˥	sni+rihwihs+a˥		swaihwihsa˥	swa+rihwihs+a˥
hoihwihsa˥	ho+rihwihs+a˥		goihwihsa˥	(ya)go+rihwihs+a˥
oihwihsa˥	(y)o+rihwihs+a˥		hodi:hwihsa˥	hodi+rihwihs+a˥
godi:hwihsa˥	(ya)godi+rihwihs+a˥		odi:hwihsa˥	(y)odi+rihwihs+a˥
agrǫda˥	(w)ag+rǫd+a˥		ǫkniǫda˥	(y)ǫkni+rǫd+a˥
ǫgwaǫda˥	(y)ǫgwa+rǫd+a˥		saǫda˥	sa+rǫd+a˥
sniǫda˥	sni+rǫd+a˥		swaǫda˥	swa+rǫd+a˥

Basic Nouns Continued

Surface	Underlying		Surface	Underlying
hoǫda˺	ho+rǫd+a˺		goǫda˺	(ya)go+rǫd+a˺
oǫda˺	(y)o+rǫd+a˺		hodiǫda˺	hodi+rǫd+a˺
godiǫda˺	(ya)godi+rǫd+a˺		odiǫda˺	(y)odi+rǫd+a˺

Table C.9: Deverbal Nouns

Surface	Underlying		Surface	Underlying
ędehsra˺	ędehsr+a˺		ę˺nyotra˺	ę˺nyotr+a˺
ę˺nhotra˺	ę˺nhotr+a˺		gędehsra˺	ga+idehsra+a˺
gayęnawahsra˺	ga+yęnawahsr+a˺		gaya˺dowehdahsra˺	ga+ya˺dowehdahsr+a˺
gaya˺dagenhahsra˺	ga+ya˺dagenhahsr+a˺		gatgwęnya˺tra˺	ga+atgwęnya˺tr+a˺
gatgǫnya˺tra˺	ga+atgǫnya˺tr+a˺		gatgi˺tra˺	ga+tgi˺tr+a˺
ganohǫkdehsra˺	ga+nohǫkdehsr+a˺		ganhehsra˺	ga+nhehsr+a˺
gana˺jowi˺tra˺	ga+na˺jowi˺tr+a˺		gaisra˺	ga+risr+a˺
gaihwiyohsdęhsra˺	ga+rihwiyohsdęhsr+a˺		gaihwane˺aksra˺	ga+rihwane˺aksra+a˺
gaiho˺dęhsra˺	ga+riho˺dęhsr+a˺		gahyadǫhsra˺	ga+hyadǫhsr+a˺
gahshahsdęhsra˺	ga+hshahsdęhsr+a˺		ga:hǫhsra˺	ga+Cahǫhsr+a˺
oyęhsra˺	o+yęhsr+a˺		otgahnǫnihsra˺	o+atgahnǫnihsr+a˺
onrahdǫdahsra˺	o+nrahdǫdahsr+a˺		onǫnhe˺dra˺	o+nǫnhe˺dr+a˺
oniga:hęhsra˺	o+niga:hęhsr+a˺		ohshahsdęhsra˺	o+hshahsdęhsr+a˺
odǫtgadǫhsra˺	o+adǫtgadǫhsr+a˺		odǫtgadehsra˺	o+adǫtgadehsr+a˺
o˺drohsra˺	o+˺drohsr+a˺		o˺daihęhdra˺	o+i˺daihęhdr+a˺
atsho˺kdǫhsra˺	a+atsho˺kdǫhsr+a˺		atna˺tsotra˺	a+atna˺tsotr+a˺
atna˺gwihdra˺	a+atna˺gwihdr+a˺		atgahnyehtra˺	a+atgahnyehtr+a˺
anahaotra˺	a+nahaotr+a˺		agya˺dawi˺tra˺	a+agya˺dawi˺tr+a˺
adrihwagyaǫhsra˺	a+adrihwagyaǫhsr+a˺		adra˺wihsda˺	a+adra˺wihsd+a˺
adǫnhehsra˺	a+adǫnhehsr+a˺		adi˺grǫhsra˺	a+adi˺grǫhsra+a˺
adęna˺tra˺	a+adęna˺tr+a˺		adekwahahsra˺	a+adekwahahsra+a˺
adao˺tra˺	a+adao˺tra+a˺		ahdahdi˺tra˺	a+ahdahdi˺tr+a˺

Table C.10: Defective Nouns

Surface	Underlying
sgwa:gwaǫdǫ˥	sgwa:gwaǫdǫ˥
da:gu:s	da:gu:s
so:wa:s	so:wa:s
hǫ:ga:k	hǫ:ga:k
gwihsgwihs	gwihsgwihs
sohǫ:t	sohǫ:t
jogrihs	jogrihs
di˥di:˥	di˥di:˥
ga˥ga:˥	ga˥ga:˥
gwiyę˥gwiyę˥	gwiyę˥gwiyę˥
ji˥nhǫwę:se:	ji˥nhǫwę:se:
sa˥sa˥	sa˥sa˥
gwe:sę˥	gwe:sę˥
jihsgogo˥	jihsgogo˥
johwę˥sdaga˥	johwę˥sdaga˥
hnyagwai˥	hnyagwai˥
tgwiyo:gę˥	tgwiyo:gę˥
ji˥nǫhdo:ya˥	ji˥nǫhdo:ya˥
jinǫhsanǫh	jinǫhsanǫh
ji˥ao:yę:	ji˥ao:yę:
degriya˥gǫ˥	degriya˥gǫ˥
hehshai:	hehshai:
tehtǫ˥	tehtǫ˥
jino:wę:	jino:wę:
sa:no:˥	sa:no:˥
dre:na:	dre:na:
kdagǫ˥	kdagǫ˥
sgwa:yęh	sgwa:yęh
jǫ:nyǫ:˥	jǫ:nyǫ:˥
gwa˥da:	gwa˥da:
jǫ˥dae:ya:˥	jǫ˥dae:ya:˥
onohotsgę˥ę˥	onohotsgę˥ę˥
tsa˥gę:da˥	tsa˥gę:da˥
tsinyohgwa:k	tsinyohgwa:k
nawę˥da˥	nawę˥da˥
otahyǫ:ni:	otahyǫ:ni:
jihsgę:	jihsgę:
grahe:t	grahe:t

Surface	Underlying
dago:s	dago:s
dakshae˥dohs	dakshae˥dohs
twę:twę:t	twę:twę:t
dogę:t	dogę:t
gwa˥yǫ˥	gwa˥yǫ˥
gyo:gyo:˥	gyo:gyo:˥
gwido˥gwido˥	gwido˥gwido˥
jikjiye:˥	jikjiye:˥
hihi:	hihi:
dihsdihs	dihsdihs
duwisduwi:˥	duwisduwi:˥
gwę:dihs	gwę:dihs
tsahgo:wah	tsahgo:wah
gwaoh	gwaoh
gwę˥gohnyę˥	gwę˥gohnyę˥
gǫ:deh	gǫ:deh
jinhǫhgwahęh	jinhǫhgwahęh
ji˥dana:wę:	ji˥dana:wę:
jihsda:	jihsda:
jinǫhyahae:	jinǫhyahae:
jihnyo˥gę˥	jihnyo˥gę˥
sgwa˥ahda˥	sgwa˥ahda˥
jǫ˥daga˥	jǫ˥daga˥
tea:ǫt	tea:ǫt
drę:na:	drę:na:
joni:tsgrǫ:t	joni:tsgrǫ:t
do:dihs	do:dihs
gwiyo:gę˥	gwiyo:gę˥
nǫhsodai:yǫ:	nǫhsodai:yǫ:
jidę:˥ęh	jidę:˥ęh
ji˥drǫ:wę:	ji˥drǫ:wę:
teo:ji˥	teo:ji˥
yahgęhda˥	yahgęhda˥
gihę:k	gihę:k
jihsǫ:dahk	jihsǫ:dahk
tahyǫ:ni:	tahyǫ:ni:
ji˥o:	ji˥o:

Table C.11: Inalienable Nouns

Surface	Underlying
gahǫhdaˀgeh	g+ahǫhd+aˀgeh
agyahǫhdaˀgeh	(y)agy+ahǫhd+aˀgeh
agwahǫhdaˀgeh	(y)agw+ahǫhd+aˀgeh
jahǫhdaˀgeh	(h)j+ahǫhd+aˀgeh
hahǫhdaˀgeh	h+ahǫhd+aˀgeh
wahǫhdaˀgeh	w+ahǫhd+aˀgeh
gaǫhǫhdaˀgeh	gaǫ+ahǫhd+aˀgeh
gahsiˀdaˀgeh	g+ahsiˀd+aˀgeh
gyahsiˀdaˀgeh	(e)gy+ahsiˀd+aˀgeh
dwahsiˀdaˀgeh	(e)dw+ahsiˀd+aˀgeh
sahsiˀdaˀgeh	(h)s+ahsiˀd+aˀgeh
swahsiˀdaˀgeh	(h)sw+ahsiˀd+aˀgeh
ǫhsiˀdaˀgeh	(y)ǫ+ahsiˀd+aˀgeh
hęnahsiˀdaˀgeh	hęn+ahsiˀd+aˀgeh
gęnahsiˀdaˀgeh	gęn+ahsiˀd+aˀgeh
tnihnaˀgeh	(e)tn+ihn+aˀgeh
aknihnaˀgeh	(y)akn+ihn+aˀgeh
agwęhnaˀgeh	(y)agwę+ihn+aˀgeh
snihnaˀgeh	(h)sn+ihn+aˀgeh
hęhnaˀgeh	hę+ihn+aˀgeh
gęhnaˀgeh	gę+ihn+aˀgeh
gaehnaˀgeh	gae+ihn+aˀgeh
gǫtsaˀgeh	g+ǫts+aˀgeh
knǫtsaˀgeh	(e)kn+ǫts+aˀgeh
gyǫtsaˀgeh	(e)gy+ǫts+aˀgeh
sǫtsaˀgeh	(h)s+ǫts+aˀgeh
jǫtsaˀgeh	(h)j+ǫts+aˀgeh
agǫtsaˀgeh	(y)ag+ǫts+aˀgeh
hęnǫtsaˀgeh	hęn+ǫts+aˀgeh
gęnǫtsaˀgeh	gęn+ǫts+aˀgeh
tniwęˀnahsaˀgeh	(e)tni+węˀnahs+aˀgeh
akniwęˀnahsaˀgeh	(y)akni+węˀnahs+aˀgeh
agwawęˀnahsaˀgeh	(y)agwa+węˀnahs+aˀgeh
sniwęˀnahsaˀgeh	(eh)sni+węˀnahs+aˀgeh
hawęˀnahsaˀgeh	ha+węˀnahs+aˀgeh
gawęˀnahsaˀgeh	ga+węˀnahs+aˀgeh
gaewęˀnahsaˀgeh	gae+węˀnahs+aˀgeh
gegahaˀgeh	ge+gah+aˀgeh
knigahaˀgeh	(e)kni+gah+aˀgeh

Surface	Underlying
gyahǫhdaˀgeh	(e)gy+ahǫhd+aˀgeh
dwahǫhdaˀgeh	(e)dw+ahǫhd+aˀgeh
sahǫhdaˀgeh	(h)s+ahǫhd+aˀgeh
swahǫhdaˀgeh	(h)sw+ahǫhd+aˀgeh
ǫhǫhdaˀgeh	(y)ǫ+ahǫhd+aˀgeh
hęnahǫhdaˀgeh	hęn+ahǫhd+aˀgeh
gęnahǫhdaˀgeh	gęn+ahǫhd+aˀgeh
gyahsiˀdaˀgeh	(e)gy+ahsiˀd+aˀgeh
agyahsiˀdaˀgeh	(y)agy+ahsiˀd+aˀgeh
agwahsiˀdaˀgeh	(y)agw+ahsiˀd+aˀgeh
jahsiˀdaˀgeh	(h)j+ahsiˀd+aˀgeh
hahsiˀdaˀgeh	h+ahsiˀd+aˀgeh
wahsiˀdaˀgeh	w+ahsiˀd+aˀgeh
gaǫhsiˀdaˀgeh	gaǫ+ahsiˀd+aˀgeh
gihnaˀgeh	g+ihn+aˀgeh
knihnaˀgeh	(e)kn+ihn+aˀgeh
dwęhnaˀgeh	(e)dwę+ihn+aˀgeh
sihnaˀgeh	(h)s+ihn+aˀgeh
swęhnaˀgeh	(h)swę+ihn+aˀgeh
ęhnaˀgeh	(y)ę+ihn+aˀgeh
hadihnaˀgeh	had+ihn+aˀgeh
gadihnaˀgeh	gad+ihn+aˀgeh
tnǫtsaˀgeh	(e)tn+ǫts+aˀgeh
aknǫtsaˀgeh	(y)akn+ǫts+aˀgeh
agyǫtsaˀgeh	(y)agy+ǫts+aˀgeh
snǫtsaˀgeh	(h)sn+ǫts+aˀgeh
hǫtsaˀgeh	h+ǫts+aˀgeh
ǫtsaˀgeh	(y)+ǫts+aˀgeh
ga:gǫtsaˀgeh	ga:g+ǫts+aˀgeh
gwęˀnahsaˀgeh	g+węˀnahs+aˀgeh
kniwęˀnahsaˀgeh	(e)kni+węˀnahs+aˀgeh
dwawęˀnahsaˀgeh	(e)dwa+węˀnahs+aˀgeh
swęˀnahsaˀgeh	(eh)s+węˀnahs+aˀgeh
swawęˀnahsaˀgeh	(eh)swa+węˀnahs+aˀgeh
ewęˀnahsaˀgeh	(y)e+węˀnahs+aˀgeh
hadiwęˀnahsaˀgeh	hadi+węˀnahs+aˀgeh
gadiwęˀnahsaˀgeh	gadi+węˀnahs+aˀgeh
tnigahaˀgeh	(e)tni+gah+aˀgeh
aknigahaˀgeh	(y)akni+gah+aˀgeh

Inalienable Nouns Continued

Surface	**Underlying**		**Surface**	**Underlying**
dwagahaˀgeh	(e)dwa+gah+aˀgeh		agwagahaˀgeh	(y)agwa+gah+aˀgeh
segahaˀgeh	(eh)se+gah+aˀgeh		snigahaˀgeh	(eh)sni+gah+aˀgeh
swagahaˀgeh	(eh)swa+gah+aˀgeh		hagahaˀgeh	ha+gah+aˀgeh
egahaˀgeh	(y)e+gah+aˀgeh		gagahaˀgeh	ga+gah+aˀgeh
hadigahaˀgeh	hadi+gah+aˀgeh		gaegahaˀgeh	gae+gah+aˀgeh
gadigahaˀgeh	gadi+gah+aˀgehk+het		ketgaˀaˀgeh	gaˀ+aˀgeh
tnihetgaˀaˀgeh	(e)tni+hetgaˀ+aˀgeh		knihetgaˀaˀgeh	(e)kni+hetgaˀ+aˀgeh
aknihetgaˀaˀgeh	(y)akni+hetgaˀ+aˀgeh		dwahetgaˀaˀgeh	(e)dwa+hetgaˀ+aˀgeh
agwahetgaˀaˀgeh	(y)agwa+hetgaˀ+aˀgeh		setgaˀaˀgeh	(eh)s+hetgaˀ+aˀgeh
snihetgaˀaˀgeh	(eh)sni+hetgaˀ+aˀgeh		swahetgaˀaˀgeh	(eh)swa+hetgaˀ+aˀgeh
hahetgaˀaˀgeh	ha+hetgaˀ+aˀgeh		ehetgaˀaˀgeh	(y)e+hetgaˀ+aˀgeh
gahetgaˀaˀgeh	ga+hetgaˀ+aˀgeh		hadihetgaˀaˀgeh	hadi+hetgaˀ+aˀgeh
gaehetgaˀaˀgeh	gae+hetgaˀ+aˀgeh		gadihetgaˀaˀgeh	gadi+hetgaˀ+aˀgeh
geˀnhǫhsgaːˀgeh	ge+ˀnhǫhsgaː+aˀgeh		tniˀnhǫhsgaːˀgeh	(e)tni+ˀnhǫhsgaː+aˀgeh
kniˀnhǫhsgaːˀgeh	(e)kni+ˀnhǫhsgaː+aˀgeh		akniˀnhǫhsgaːˀgeh	(y)akni+ˀnhǫhsgaː+aˀgeh
dwaˀnhǫhsgaːˀgeh	(e)dwa+ˀnhǫhsgaː+aˀgeh		agwaˀnhǫhsgaːˀgeh	(y)agwa+ˀnhǫhsgaː+aˀgeh
seˀnhǫhsgaːˀgeh	(eh)se+ˀnhǫhsgaː+aˀgeh		sniˀnhǫhsgaːˀgeh	(eh)sni+ˀnhǫhsgaː+aˀgeh
swaˀnhǫhsgaːˀgeh	(eh)swa+ˀnhǫhsgaː+aˀgeh		haˀnhǫhsgaːˀgeh	ha+ˀnhǫhsgaː+aˀgeh
eˀnhǫhsgaːˀgeh	(y)e+ˀnhǫhsgaː+aˀgeh		gaˀnhǫhsgaːˀgeh	ga+ˀnhǫhsgaː+aˀgeh
hadiˀnhǫhsgaːˀgeh	hadi+ˀnhǫhsgaː+aˀgeh		gaeˀnhǫhsgaːˀgeh	gae+ˀnhǫhsgaː+aˀgeh
gadiˀnhǫhsgaːˀgeh	gadi+ˀnhǫhsgaː+aˀgeh		geˀahsaˀgeh	ge+ˀahs+aˀgeh
tniˀahsaˀgeh	(e)tni+ˀahs+aˀgeh		kniˀahsaˀgeh	(e)kni+ˀahs+aˀgeh
akniˀahsaˀgeh	(y)akni+ˀahs+aˀgeh		dwaˀahsaˀgeh	(e)dwa+ˀahs+aˀgeh
agwaˀahsaˀgeh	(y)agwa+ˀahs+aˀgeh		seˀahsaˀgeh	(eh)se+ˀahs+aˀgeh
sniˀahsaˀgeh	(eh)sni+ˀahs+aˀgeh		swaˀahsaˀgeh	(eh)swa+ˀahs+aˀgeh
haˀahsaˀgeh	ha+ˀahs+aˀgeh		eˀahsaˀgeh	(y)e+ˀahs+aˀgeh
gaˀahsaˀgeh	ga+ˀahs+aˀgeh		hadiˀahsaˀgeh	hadi+ˀahs+aˀgeh
gaeˀahsaˀgeh	gae+ˀahs+aˀgeh		gadiˀahsaˀgeh	gadi+ˀahs+aˀgeh
gekseˀdaˀgeh	ge+kseˀd+aˀgeh		tnikseˀdaˀgeh	(e)tni+kseˀd+aˀgeh
knikseˀdaˀgeh	(e)kni+kseˀd+aˀgeh		aknikseˀdaˀgeh	(y)akni+kseˀd+aˀgeh
dwakseˀdaˀgeh	(e)dwa+kseˀd+aˀgeh		agwakseˀdaˀgeh	(y)agwa+kseˀd+aˀgeh
sekseˀdaˀgeh	(eh)se+kseˀd+aˀgeh		snikseˀdaˀgeh	(eh)sni+kseˀd+aˀgeh
swakseˀdaˀgeh	(eh)swa+kseˀd+aˀgeh		hakseˀdaˀgeh	ha+kseˀd+aˀgeh
ekseˀdaˀgeh	(y)e+kseˀd+aˀgeh		gakseˀdaˀgeh	ga+kseˀd+aˀgeh
hadikseˀdaˀgeh	hadi+kseˀd+aˀgeh		gaekseˀdaˀgeh	gae+kseˀd+aˀgeh
gadikseˀdaˀgeh	gadi+kseˀd+aˀgeh		knǫˀaːˀgeh	k+nǫˀaː+aˀgeh
tninǫˀaːˀgeh	(e)tni+nǫˀaː+aˀgeh		kninǫˀaːˀgeh	(e)kni+nǫˀaː+aˀgeh
akninǫˀaːˀgeh	(y)akni+nǫˀaː+aˀgeh		dwanǫˀaːˀgeh	(e)dwa+nǫˀaː+aˀgeh

Inalienable Nouns Continued

Surface	Underlying		Surface	Underlying
agwanǫ̃ˀa:ˀgeh	(y)agwa+nǫ̃ˀa:+aˀgeh		snǫ̃ˀa:ˀgeh	(eh)s+nǫ̃ˀa:+aˀgeh
sninǫ̃ˀa:ˀgeh	(eh)sni+nǫ̃ˀa:+aˀgeh		swanǫ̃ˀa:ˀgeh	(eh)swa+nǫ̃ˀa:+aˀgeh
hanǫ̃ˀa:ˀgeh	ha+nǫ̃ˀa:+aˀgeh		enǫ̃ˀa:ˀgeh	(y)e+nǫ̃ˀa:+aˀgeh
ganǫ̃ˀa:ˀgeh	ga+nǫ̃ˀa:+aˀgeh		hadinǫ̃ˀa:ˀgeh	hadi+nǫ̃ˀa:+aˀgeh
gaenǫ̃ˀa:ˀgeh	gae+nǫ̃ˀa:+aˀgeh		gadinǫ̃ˀa:ˀgeh	gadi+nǫ̃ˀa:+aˀgeh
gragwahdaˀgeh	g+ragwahd+aˀgeh		tniagwahdaˀgeh	(e)tni+ragwahd+aˀgeh
kniagwahdaˀgeh	(e)kni+ragwahd+aˀgeh		akniagwahdaˀgeh	(y)akni+ragwahd+aˀgeh
dwa:gwahdaˀgeh	(e)dwa+ragwahd+aˀgeh		agwa:gwahdaˀgeh	(y)agwa+ragwahd+aˀgeh
sragwahdaˀgeh	(eh)s+ragwahd+aˀgeh		sniagwahdaˀgeh	(eh)sni+ragwahd+aˀgeh
swa:gwahdaˀgeh	(eh)swa+ragwahd+aˀgeh		ha:gwahdaˀgeh	ha+ragwahd+aˀgeh
eagwahdaˀgeh	(y)e+ragwahd+aˀgeh		ga:gwahdaˀgeh	ga+ragwahd+aˀgeh
hadiagwahdaˀgeh	hadi+ragwahd+aˀgeh		gaeagwahdaˀgeh	gae+ragwahd+aˀgeh
gadiagwahdaˀgeh	gadi+ragwahd+aˀgeh		gwę̃ˀyǫ̃hga:ˀgeh	g+wę̃ˀyǫ̃hga:+aˀgeh
tniwę̃ˀyǫ̃hga:ˀgeh	(e)tni+wę̃ˀyǫ̃hga:+aˀgeh		kniwę̃ˀyǫ̃hga:ˀgeh	(e)kni+wę̃ˀyǫ̃hga:+aˀgeh
akniwę̃ˀyǫ̃hga:ˀgeh	(y)akni+wę̃ˀyǫ̃hga:+aˀgeh		dwawę̃ˀyǫ̃hga:ˀgeh	(e)dwa+wę̃ˀyǫ̃hga:+aˀgeh
agwawę̃ˀyǫ̃hga:ˀgeh	(y)agwa+wę̃ˀyǫ̃hga:+aˀgeh		swę̃ˀyǫ̃hga:ˀgeh	(eh)s+wę̃ˀyǫ̃hga:+aˀgeh
sniwę̃ˀyǫ̃hga:ˀgeh	(eh)sni+wę̃ˀyǫ̃hga:+aˀgeh		swawę̃ˀyǫ̃hga:ˀgeh	(eh)swa+wę̃ˀyǫ̃hga:+aˀgeh
hawę̃ˀyǫ̃hga:ˀgeh	ha+wę̃ˀyǫ̃hga:+aˀgeh		ewę̃ˀyǫ̃hga:ˀgeh	(y)e+wę̃ˀyǫ̃hga:+aˀgeh
gawę̃ˀyǫ̃hga:ˀgeh	ga+wę̃ˀyǫ̃hga:+aˀgeh		hadiwę̃ˀyǫ̃hga:ˀgeh	hadi+wę̃ˀyǫ̃hga:+aˀgeh
gaewę̃ˀyǫ̃hga:ˀgeh	gae+wę̃ˀyǫ̃hga:+aˀgeh		gadiwę̃ˀyǫ̃hga:ˀgeh	gadi+wę̃ˀyǫ̃hga:+aˀgeh
gejaohoˀgwaˀgeh	ge+jaohoˀgw+aˀgeh		tnijaohoˀgwaˀgeh	(e)tni+jaohoˀgw+aˀgeh
knijaohoˀgwaˀgeh	(e)kni+jaohoˀgw+aˀgeh		aknijaohoˀgwaˀgeh	(y)akni+jaohoˀgw+aˀgeh
dwajaohoˀgwaˀgeh	(e)dwa+jaohoˀgw+aˀgeh		agwajaohoˀgwaˀgeh	(y)agwa+jaohoˀgw+aˀgeh
sejaohoˀgwaˀgeh	(eh)se+jaohoˀgw+aˀgeh		snijaohoˀgwaˀgeh	(eh)sni+jaohoˀgw+aˀgeh
swajaohoˀgwaˀgeh	(eh)swa+jaohoˀgw+aˀgeh		hajaohoˀgwaˀgeh	ha+jaohoˀgw+aˀgeh
ejaohoˀgwaˀgeh	(y)e+jaohoˀgw+aˀgeh		gajaohoˀgwaˀgeh	ga+jaohoˀgw+aˀgeh
hadijaohoˀgwaˀgeh	hadi+jaohoˀgw+aˀgeh		gaejaohoˀgwaˀgeh	gae+jaohoˀgw+aˀgeh
gadijaohoˀgwaˀgeh	gadi+jaohoˀgw+aˀgeh		gyaˀga:ˀgeh	g+yaˀga:+aˀgeh
tniyaˀga:ˀgeh	(e)tni+yaˀga:+aˀgeh		kniyaˀga:ˀgeh	(e)kni+yaˀga:+aˀgeh
akniyaˀga:ˀgeh	(y)akni+yaˀga:+aˀgeh		dwayaˀga:ˀgeh	(e)dwa+yaˀga:+aˀgeh
agwayaˀga:ˀgeh	(y)agwa+yaˀga:+aˀgeh		syaˀga:ˀgeh	(eh)s+yaˀga:+aˀgeh
sniyaˀga:ˀgeh	(eh)sni+yaˀga:+aˀgeh		swayaˀga:ˀgeh	(eh)swa+yaˀga:+aˀgeh
hayaˀga:ˀgeh	ha+yaˀga:+aˀgeh		eyaˀga:ˀgeh	(y)e+yaˀga:+aˀgeh
gayaˀga:ˀgeh	ga+yaˀga:+aˀgeh		hadiyaˀga:ˀgeh	hadi+yaˀga:+aˀgeh
gaeyaˀga:ˀgeh	gae+yaˀga:+aˀgeh		gadiyaˀga:ˀgeh	gadi+yaˀga:+aˀgeh

Table C.12: Unpossessed Inalienable Nouns

Surface	**Underlying**
ohǫhda˺	o+ahǫhd+a˺
ohna˺	o+ihn+a˺
owę˺nahsa˺	o+wę˺nahs+a˺
ohetga˺a˺	o+hetga˺+a˺
o˺ahsa˺	o+˺ahs+a˺
onǫ˺a:˺	o+nǫ˺a:+a˺
owę˺yǫhga:˺	o+wę˺yǫhga:+a˺
oya˺ga:˺	o+ya˺ga:+a˺

Surface	**Underlying**
ohsi˺da˺	o+ahsi˺d+a˺
ǫtsa˺	o+ǫts+a˺
ogaha˺	o+gah+a˺
o˺nhǫhsga:˺	o+˺nhǫhsga:+a˺
okse˺da˺	o+kse˺d+a˺
oagwahda˺	o+ragwahd+a˺
ojaoho˺gwa˺	o+jaoho˺gw+a˺