Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020), pages 2676–2684
Marseille, 11–16 May 2020

Improved Finite-State Morphological Analysis for St. Lawrence Island Yupik Using Paradigm Function Morphology

Emily Chen, Hyunji Hayley Park, Lane Schwartz
University of Illinois Urbana-Champaign
Champaign, IL
{echen41, hpark129, lanes}@illinois.edu

Abstract

St. Lawrence Island Yupik is an endangered polysynthetic language of the Bering Strait region. While conducting linguistic fieldwork between 2016 and 2019, we observed substantial support within the Yupik community for language revitalization and for resource development to support Yupik education. To that end, Chen and Schwartz (2018) implemented a finite-state morphological analyzer as a critical enabling technology for use in Yupik language education and technology. Chen and Schwartz (2018) reported a morphological analysis coverage rate of approximately 75% on a dataset of 60K Yupik tokens, leaving considerable room for improvement. In this work, we present a re-implementation of the Chen and Schwartz (2018) finite-state morphological analyzer for St. Lawrence Island Yupik that incorporates new linguistic insights; in particular, in this implementation we make use of the Paradigm Function Morphology (PFM) theory of morphology. We evaluate this new PFM-based morphological analyzer, and demonstrate that it consistently outperforms the existing analyzer of Chen and Schwartz (2018) with respect to accuracy and coverage rate across multiple datasets.

Keywords: computational morphology, morphological analysis, linguistic resource, Yupik languages, language revitalization

1. Introduction

St. Lawrence Island Yupik (hereafter *Yupik*) is an endangered polysynthetic language spoken on St. Lawrence Island, Alaska and on the Chukotka Peninsula of Russia. While conducting linguistic fieldwork between 2016 and 2019, we observed substantial support within the Yupik community for language revitalization and for resource development to support Yupik language education. Yupik exhibits highly productive derivational and inflectional morphology (§2.), having been observed to allow up to seven derivational morphemes per word (de Reuse, 1994, p.53). Because of this high degree of morphological complexity, a high-coverage morphological analyzer is a necessary enabling technology that is a prerequisite for the development of basic language resources and technologies such as a searchable electronic dictionary or a mobile text completion system. Chen and Schwartz (2018) presented the first such morphological analyzer for Yupik (§3.). In this work, we apply the Paradigm Function Morphology (PFM) theory of morphology (§4.) to Yupik, and present a re-implementation of the Chen and Schwartz (2018) finite-state morphological analyzer. To our knowledge, this paper represents the first attempt at using PFM to inform the design of a morphological analyzer for Yupik, although some previous work exists in this regard for other languages, namely Lingala (Karttunen, 2003). We find that incorporating PFM into our implementation results in a higher-coverage second-generation morphological analyzer for St. Lawrence Island Yupik (§5.).

2. Yupik Language Description

The most thorough sources which describe St. Lawrence Island Yupik are the Jacobson (2001) reference grammar and the Yupik-English dictionary of Badten et al. (2008). Other sources include a Russian-language Yupik grammar (Menovshchikov, 1962), a description of Yupik phonology (Krauss, 1975), and a description of Yupik morphology and syntax by de Reuse (1994).

2.1. Yupik Morphology

The Yupik lexicon includes approximately 8,000 roots (including noun roots, verb roots, and roots relating to emotion and posture), approximately 600 derivational morphemes, and a rich set of demonstratives, pronouns, and particles. Suffixing a derivational morpheme to a root yields a *stem*, to which another derivational morpheme may suffix to yield yet another stem. Since stems are defined as any uninflected form, all roots are stems but not all stems are roots. Derivational morphemes in Yupik are thus categorized by the type of stem to which they attach, along with the type of the resulting stem. The four most common categories of derivational morphemes in Yupik are:

1. N→N Suffix to noun stems and yield noun stems
2. N→V Suffix to noun stems and yield verb stems
3. V→N Suffix to verb stems and yield noun stems
4. V→V Suffix to verb stems and yield verb stems

Noun stems inflect for case, possessor, and number, while verb stems inflect for mood, valency, person, and number. Fully-inflected Yupik words typically adhere to the following template, where parentheses indicate optionality:

Root + (0 -7 *Derivational Suffixes*) + *Inflectional Suffixes*

2.2. Yupik Morphophonology

Suffixation in Yupik involves various phonological alternation patterns. As a result, underlying characters near morpheme boundaries rarely map one-to-one with their surface form counterparts as demonstrated in (1).

(1) **panilek**
panig- -leg- -Ø
daughter- -one.with- -ABS.SG
'*one with a daughter*' (Jacobson, 2001, p.25)

The Jacobson (2001) reference grammar and the Badten et al. (2008) dictionary treat all phonological alternation as lexicalized, using a custom set of symbols to annotate the lexical entry of each morpheme. For instance, consider the second morpheme **-leg-** in (1), which is an N→N derivational morpheme meaning '*one with N[oun]*'. This morpheme is typically annotated as **–leg**, where the symbol **–** signals that any stem-final consonant is deleted when this derivational morpheme suffixes. Therefore, when **–leg** suffixes to the root **panig-** '*daughter*' in (1), the stem-final consonant *-g* is deleted, yielding the stem **panileg-** *'one with a daughter'*. This eventually surfaces as **panilek** upon being inflected for the absolutive singular.

3. Chen and Schwartz (2018) First-Generation Morphological Analyzer

The morphological analyzer of Chen and Schwartz (2018) implemented the morphophonological rules documented in the Jacobson (2001) reference grammar, and was designed with the following assumptions:

1. Following Jacobson (2001) and Badten et al. (2008), all phonological alternation was treated as lexicalized.

2. The resolution of phonological alternation in succeeding morphemes is processed one morpheme at a time.

We use the following example word from Jacobson (2001) to illustrate these assumptions. It consists of a noun root, a derivational morpheme, and two inflectional morphemes:

(2) **aghnaaguq**
aghnagh- -∼:(ng)u- -∼$_{f}$(g/t)u- -q
woman- -to.be- -INTR.IND- -3SG
'*she is a woman*' (Jacobson, 2001, p.25-26)

Under Assumption 1, each morpheme is annotated with zero or more symbols. Each of these symbols is listed in Table 1, followed by a description of the phonological alternation pattern it represents.[1] Under Assumption 2, the surface string is derived by suffixing each morpheme one at a time, resolving all phonological alternation from the first morpheme before proceeding to the next. The derivation of (2) proceeds as follows:

Suffix **∼:(ng)u** to root **aghnagh-**:

i) Allomorph symbol (ng) surfaces as Ø because the root *aghnagh-* does not end in a vowel.
ii) ∼ does not apply when there is no penultimate or stem-final *-e*.
iii) : deletes stem-final *-gh*, yielding *aghnau-*.
iv) Yupik phonology prohibits diphthongs and assimilates *-u* to *-a*, yielding stem *aghnaa-*.

Suffix **∼$_{f}$(g/t)u** to stem **aghnaa-**:

[1]Following Jacobson (2001), the interlinear gloss in (2) lists all of the symbols associated with each morpheme, even though not all apply in this example.

Symbol	Morphophonological Rule
∼	Deletes penultimate (semi-final) or stem-final *-e* and lengthens the preceding vowel under certain conditions
:	Deletes stem-final multicharacter consonant *-gh* between two single vowels
(ng)	Surfaces as *-ng* if the stem ends in a vowel, Ø otherwise
(g/t)	Surfaces as *-g* if the stem ends in two vowels, *-t* if the stem ends in a consonant
∼$_{f}$	Deletes stem-final *-e* and lengthens the preceding vowel uner certain conditions

Table 1: Description of the symbols that appear in (2) and the phonological alternations they represent.

v) Allomorph symbol (g/t) surfaces as *-g* because the derived stem *aghnaa-* ends in two vowels, yielding stem *aghnaagu-*. Observe that Assumption 2 requires Steps iii) and iv) to resolve first in order for this step to correctly resolve the allomorph (g/t) into a *-g*.
vi) ∼$_{f}$ does not apply when there is no stem-final *-e*.

Suffix **-q** to stem **aghnaagu-**:

vii) This results in the final surface form **aghnaaguq**.

3.1. Challenges Resulting From Assumption 2

The morphological analyzer of Chen and Schwartz (2018) was implemented using the `foma` finite-state toolkit (Hulden, 2009), which requires the implementation of two files: (1) the `lexc` file, in which the programmer codes the lexicon, and (2) the `foma` file, in which the programmer implements conditioned *rewrite rules* of the form $\alpha \rightarrow \beta \,||\, \gamma \_ \delta$. These rules are composed together with the lexicon into a finite-state transducer capable of transforming lexical strings into surface strings and vice versa. In this formalism, rules are ordered and applied across all conditionally relevant parts of a lexical string. This behavior, however, violates Assumption 2, which requires that all symbols at a given morpheme boundary be completely resolved prior to the resolution of symbols at subsequent boundaries. This is shown in the derivation of **aghnaaguq**:

Steps i–iv) aghnagh - ∼:(ng)u - ∼$_{f}$(g/t)u - q
Steps v–vi) aghnagh - ∼:(ng)u - ∼$_{f}$(g/t)u - q
Step vii) aghnagh - ∼:(ng)u - ∼$_{f}$(g/t)u - q

In order to achieve such behavior conforming to Assumption 2, Chen and Schwartz (2018) introduced two devices:

- an explicit morpheme boundary symbol ^
- multiple iterations of the rewrite rule cascade

During construction of the lexical string, the Chen and Schwartz (2018) Analyzer inserts a morpheme boundary symbol ^ between each pair of adjacent morphemes:

(3) aghnagh ^ ∼:(ng)u ^ ∼$_{f}$(g/t)u ^ q

```
Multichar_Symbols
gh (ng) (g/t) ~f
[Intr] [Ind] [3Sg]

LEXICON Root
  NounRoot;

LEXICON NounRoot
aghnagh NounSuffix;   !  woman
kufi    NounSuffix;   !  coffee
uqugh   NounSuffix;   !  smell; aroma

LEXICON NounSuffix
~%:(ng)u VerbSuffix; !  to be N

LEXICON VerbSuffix
0:0 VerbInfl;

LEXICON VerbInfl
[Intr][Ind]:~f(g/t)u PrsNum;

LEXICON PrsNum
[3Sg]:q #;
```

Figure 1: `Lexc` file that models the construction of the lexical string for **aghnaaguq**. An exclamation point denotes a comment. A percent sign denotes an escape character.

Each rewrite rule in the Chen and Schwartz (2018) Analyzer is then constrained to operate only on the symbols adjacent to the leftmost morpheme boundary; this ensures that rewrite rules only apply to the leftmost unprocessed morpheme, thus complying with Assumptions 1 and 2. The entire rewrite rule cascade in the Chen and Schwartz (2018) Analyzer is repeated eight times to allow for up to seven potential derivational morphemes and an obligatory inflectional morpheme (de Reuse, 1994, p.53). Figures 1 and 2 depict the Chen and Schwartz (2018) `foma` implementation for the above example. In particular, Figure 1 defines the lexical items such as roots, derivational suffixes, and inflectional suffixes, while Figure 2 defines the rewrite rules and the repeated rule cascade.

4. Paradigm Function Morphology & Yupik

Paradigm Function Morphology or PFM (Stump, 2001) is a theory of inflectional morphology, and in this section, we argue that it provides for an elegant analysis of Yupik morphology that can serve as a basis for an improved finite-state morphological analyzer (§5.).

PFM organizes all of the inflected forms of a given root in a language into a PARADIGM, such that each inflected form occupies one CELL in that paradigm (see Table 2). Each inflected form is then derived from the root by way of *rules*. Formally, these rules comprise the core operation of PFM: the *paradigm function* PF, which takes as arguments, a root X and a morphosyntactic property set σ that contains properties such as as CASE, PERSON, NUMBER, etc. The paradigm function outputs an inflected form Y, which when paired with σ (see Equation 1) occupies one cell in

```
define InsertBoundary
  [..]  -> ^ || Alph _ Symbol ;

define CleanupBoundary
  ^ -> 0 || .#.  Alph+ _ ;

define ResolveAllomorphy
  (ng)  -> ng || Alph+ Vow ^ _     .o.
  (g/t) -> g  || Alph+ Vow Vow ^ _.o.
  (g/t) -> t  || Alph+ Cns ^ _     .o.
  (ng)  -> 0  || .#.  Alph+ ^ _    .o.
  (g/t) -> 0  || .#.  Alph+ ^ _    ;

define DeleteGH
  gh -> 0 || Alph+ Vow _ ^ :  Vow  .o.
  :  -> 0 || .#.  Alph+ ^ _        ;

⋮

define Grammar [
  Lexicon                          .o.
  InsertBoundary                   .o.
  !!  ITERATION 1 !!
  ResolveAllomorphy                .o.
  SemiAndFinalE                    .o.
  DeleteGH                         .o.
  FinalE                           .o.
  AssimilateVowels                 .o.
  CleanupBoundary                  .o.
  ⋮
  !!  ITERATION 8 !!
  ResolveAllomorphy                .o.
  SemiAndFinalE                    .o.
  DeleteGH                         .o.
  FinalE                           .o.
  AssimilateVowels                 .o.
  CleanupBoundary
];
```

Figure 2: `Foma` file that correctly derives the surface string **aghnaaguq**. **.#.** denotes a word boundary, the **.o.** operator denotes rule composition, and the **+** regex operator denotes "at least one".

the inflectional paradigm of the root (Stump, 2001, p.43).

$$\text{PF}(\langle \text{X}, \sigma \rangle) = \langle \text{Y}, \sigma \rangle \quad (1)$$

The highlighted cell in Table 2 can therefore be described as follows:

$$\text{PF}(\langle \textit{kufi}, \{\text{UNPD}, \text{PL}\} \rangle) = \langle \textit{kufit}, \{\text{UNPD}, \text{PL}\} \rangle \quad (2)$$

One significant feature of PFM is the fact that the same PF may apply to multiple roots. This results in identical morphological exponents in their paradigms, where an *exponent* is defined as the surface realization of a morphosyntactic property set. All roots with matching paradigms can be said to belong to the same INFLECTION CLASS or simply, CLASS. Compare Tables 2 & 3 and contrast them with

ABSOLUTIVE CASE PARADIGM for '*kufi-*'			
	SG	PL	DU
UNPOSSESSED	kufi**Ø**	kufi**t**	kufi**k**
3SG POSSESSOR	kufi**nga**	kufi**ngi**	kufi**kek**
3PL POSSESSOR	kufi**ngat**	kufi**ngit**	kufi**gket**
3DU POSSESSOR	kufi**ngak**	kufi**ngik**	kufi**gkek**

Table 2: Part of the ABSOLUTIVE CASE PARADIGM for the Yupik noun root **kufi**- '*coffee*'. The exponent of each cell's morphosyntactic property set is in bold, e.g. the exponent of the morphosyntactic property set {UNPOSSESSED, PL} is -**t**.

Table 4. Since the noun roots **kufi**- and **nuna**- share all of their morphological exponents and the noun root **uqugh**- does not, we can conclude from these tables that **kufi**- and **nuna**- belong to the same class while **uqugh**- belongs to another.

ABSOLUTIVE CASE PARADIGM for '*nuna-*'			
	SG	PL	DU
UNPOSSESSED	nuna**Ø**	nuna**t**	nuna**k**
3SG POSSESSOR	nuna**nga**	nuna**ngi**	nuna**kek**
3PL POSSESSOR	nuna**ngat**	nuna**ngit**	nuna**gket**
3DU POSSESSOR	nuna**ngak**	nuna**ngik**	nuna**gkek**

Table 3: Part of the ABSOLUTIVE CASE PARADIGM for the Yupik noun root **nuna**- '*land*'.

ABSOLUTIVE CASE PARADIGM for '*uqugh-*'			
	SG	PL	DU
UNPOSSESSED	uqu**q**	uqu**t**	uqu**k**
3SG POSSESSOR	uq**aa**	uq**ii**	uqu**kek**
3PL POSSESSOR	uq**aat**	uq**iit**	uqu**gket**
3DU POSSESSOR	uq**aak**	uq**iik**	uqu**gkek**

Table 4: Part of the ABSOLUTIVE CASE PARADIGM for the Yupik noun root **uqugh**- '*fat; blubber*'.

What this ultimately signifies is that the various symbols in an annotated Yupik morpheme can be interpreted as instructions that dictate how all classes of roots inflect. For instance, recall that the morpheme that marks the intransitive indicative verbal mood is $\sim_{\mathbf{f}}$**(g/t)u**. The Chen and Schwartz (2018) Analyzer attempts to apply all of the morphophonological rules associated with this morpheme indiscriminate of the stem (see the derivation of **aghnaaguq** in § 3.). PFM, however, interprets these symbols as four differing sets of instructions that pertain to various classes. In other words, the intransitive indicative morpheme in Yupik has four morphological exponents:

Exponent	Instructions
-u-	For roots that end in *-e*, delete *-e* and add *-u*
-gu-	For roots that end in two vowels, add *-gu*
-tu-	For roots that end in a consonant, add *-tu*
-u-	For roots in all other classes, just add *-u*

Thus, given a verb root, as long as one can identify its class, one can derive its intransitive indicative form using one of the four instructions given above. This precludes the need to annotate every Yupik suffix.

Lastly, while PFM is recognized as a theory of *inflectional* morphology, Stump (2001) observes that there is no reason it cannot also apply to derivational morphology as well. Derivational morphemes can consequently be organized into classes, and share the same morphological exponents as other members of their class.

5. Improved Finite-State Morphological Analysis for Yupik Using PFM

We now introduce our re-implementation of the Yupik analyzer that incorporates both PFM (§4.) and other linguistic insights we have gained as a result of our fieldwork. We refer to this analyzer as the PFM Analyzer[2] and deviate from Chen and Schwartz (2018) in two major ways:

- We restructure the lexicon to accord with Paradigm Function Morphology (§4.)
- We simplify the analyzer architecture to require only a single cascade of rewrite rules rather than eight (§3.)

5.1. Improved `lexc` File

The use of PFM necessitated a radical overhaul of the organization of the lexicon (the `lexc` files). The Chen and Schwartz (2018) `lexc` file was structured like the one depicted in Figure 1, where one constructs a lexical string by beginning at the `Root LEXICON` and concatenating morphemes from successive `LEXICONS` (Beesley and Karttunen, 2003). Observe that each morpheme in a lexicon is followed by the name of another lexicon, also known as its *continuation class*. After selecting a morpheme from a lexicon, one proceeds to its continuation class to select another morpheme to concatenate to the lexical string, until one reaches the end-of-string marker #. This forms a *path* through the `lexc` file that begins at the `Root LEXICON` and ends at #. In the PFM-based `lexc` file, however, there exists a separate path for every class of roots through the `lexc` file.

Through manual trial and error, we identified ten classes for Yupik noun roots and seven classes for Yupik verb roots for the PFM Analyzer. Since suffixing derivational morphemes in Yupik results in noun stems and verb stems (see § 2.1.), they can be organized into the same classes as the noun roots and verb roots. Figures 3 and 4 depict portions of the PFM-based `lexc` file whose lexicons and continuation classes were constructed by hand, but made use of Python scripts to automate the addition of new lexical entries. Figure 3 depicts the class of noun roots that end in *-a*, *-i*, or *-u*, designated CLASS 1, and Figure 4 depicts the class of noun roots that end in *-gh*, designated CLASS 2. Each class has its own unique `Root LEXICON`, `Suffix LEXICON`, and `Inflection LEXICON`.

Restructuring the `lexc` files to accord with PFM consequently results in the construction of much simpler lexical strings. Contrast the lexical string constructed by the `lexc`

[2] `https://github.com/chenemile/yupik-foma-v2`

```
LEXICON NounClass1Root
kufi NounClass1Suffix; !  coffee
nuna NounClass1Suffix; !  land

LEXICON NounClass1Suffix
ngu  VerbClass1Suffix; !  to be
ligh VerbClass2Suffix; !  to provide

LEXICON VerbClass1Suffix
0:0 VerbClass1Infl;

LEXICON VerbClass1Infl
[Intr][Ind]:(g)u PrsNum;
```

Figure 3: Portion of the `lexc` file that depicts a potential path through the file for noun roots in CLASS 1.

```
LEXICON NounClass2Root
aghnagh NounClass2Suffix;!  woman
uqugh   NounClass2Suffix;!  aroma

LEXICON NounClass2Suffix
%:u   VerbClass1Suffix;   !  to be
-ligh VerbClass2Suffix;   !  to provide

LEXICON VerbClass2Suffix
0:0 VerbClass2Infl;

LEXICON VerbClass2Infl
[Intr][Ind]:tu PrsNum;
```

Figure 4: Portion of the `lexc` file that depicts a potential path through the file for noun roots in CLASS 2.

file of the Chen and Schwartz (2018) Analyzer with the one constructed by the PFM Analyzer.

Chen and Schwartz (2018) lexical string for **aghnaaguq**:

aghnagh - ∼:(ng)u - ∼$_f$(g/t)u - q

The PFM Analyzer's lexical string for **aghnaaguq**:

aghnagh - :u - (g)u - q

5.2. Improved `foma` File

Applying the tenets of PFM simplified the `foma` file by enabling the removal of several rules altogether. One such rule is the `ResolveAllomorphy` rule which, in the Chen and Schwartz (2018) Analyzer, handled over 20 cases of Yupik allomorphy, a subset of which is shown in Figure 5. Consider, however, just the allomorph *-ng-* highlighted in Figure 5 that features in the derivational morpheme meaning '*to be*': -∼**:(ng)u**-. According to PFM, this morpheme has four exponents, and as indicated by the third exponent, it is entirely predictable when the allomorph *-ng-* surfaces:

```
define ResolveAllomorphy
  (g/t) -> g || Vow Vow _          .o.
  (g/t) -> t || Cns _              .o.
  (s/z) -> s || Cns _              .o.
  (s/z) -> z || Vow _              .o.
  (p/v) -> p || Cns _              .o.
  (p/v) -> v || Vow _              .o.
  (t/y) -> t || Cns _              .o.
  (t/y) -> y || Vow _              .o.
  (i/u) -> i || t e _              .o.
  (i/u) -> u || [ e | Cns ] _      .o.
  (ng)  -> ng || Vow _             .o.
  (te)  -> t e || Cns _            .o.
  (a)   -> a || e _                .o.
  (i1)  -> i || [ e | Cns ] _      .o.
  (i2)  -> i || [ Cns - t ] e _    .o.
  (u)   -> u || [ e | Cns ] _      .o.
  (s)   -> s || Vow _              .o.
  (g/t) -> 0                       .o.
  (s/z) -> 0                       .o.
  (p/v) -> 0                       .o.
  (t/y) -> 0                       .o.
  (i/u) -> 0                       .o.
  (ng)  -> 0                       .o.
  (te)  -> 0                       .o.
  (a)   -> 0                       .o.
  (i1)  -> 0                       .o.
  (i2)  -> 0                       .o.
  (u)   -> 0                       .o.
  (s)   -> 0                       .o.
  ⋮
```

Figure 5: Partial depiction of the `foma` rule to resolve Yupik allomorphy in the Chen and Schwartz (2018) Analyzer.

Exponent	Instructions
-u-	For roots that end in *-e* or penultimate *-e*, delete *-e* and add *-u*
-u-	For roots that end in *-gh*, delete *-gh* if it is preceded by a vowel and add *-u*
-ngu-	For roots that end in a vowel, add *-ngu-*
-u-	For roots in all other classes, just add *-u*

This consequently suggests that the realization of the allomorph *-ng-* can be handled in `lexc` rather than in `foma`. Rather than encoding the derivational morpheme meaning '*to be*' as ∼**:(ng)u** as was done in the Chen and Schwartz (2018) Analyzer (see Figure 1), it can now be encoded as one of the four exponents listed prior. For the class of noun roots that end in a vowel, the encoding of the derivational morpheme meaning '*to be*' is **-ngu-**, as depicted in Figure 3. Contrast this with Figure 4 which depicts the class of nouns roots that end in *-gh*. Since the morphological exponent of the derivational morpheme meaning '*to be*' is **-u-** for this particular class of roots, allomorph *-ng-* is not coded in this part of the `lexc` file. In this way, we successfully handle the realization of allomorph *-ng-* in `lexc` and forgo the use of rewrite rules in `foma`.

Since all of the allomorphy depicted in Figure 5 can be handled in this way, the `ResolveAllomorphy` rule was removed entirely and replaced with the much simpler rule depicted in Figure 6. The reason the allomorph *-g-* must still

```
define ResolveG
  (g) -> g || Vow Vow ^ _  .o.
  (g) -> 0;
```

Figure 6: PFM-based `foma` rule that conditions when the Yupik allomorph *-g-* surfaces.

be handled via a `foma` rule rather than in `lexc` is due to the fact that it is not always predictable when a stem will end in two vowels, permitting allomorph *-g-* to surface. This is because *derived* stems can end in two vowels as well (e.g. **aghnaa-** in § 3.). Therefore, we account for the unpredictable realization of allomorph *-g-* by maintaining it as a `foma` rule.

Our second objective for the re-implementation of the Yupik analyzer was to remove the repeated rule cascade, illustrated in Figure 2. Recall that the principal motivation for implementing the iterations in the Chen and Schwartz (2018) Analyzer was to ensure that the behavior of the analyzer conformed to Assumption 2. We have found, however, that from a computational perspective this assumption need not be true. By rewriting the `foma` rewrite rules such that they capture all of the instances when a given phonological alternation occurs, it is possible to avoid having to implement a repeated rule cascade.

For instance, the `ResolveAllomorphy` rule in the Chen and Schwartz (2018) Analyzer, shown in Figure 5 does not currently permit allomorph *-g-* to surface after a derived stem that ends in two vowels. We account for this in the PFM Analyzer by relaxing the environment coded in Figure 6 to that seen in Figure 7. By inserting an optional morpheme boundary marker (ˆ), the environment in Figure 7 now captures all of the environments in which allomorph *-g-* surfaces.

```
define ResolveG
  (g) -> g || Vow (^) Vow ^ _  .o.
  (g) -> 0;
```

Figure 7: Edited `foma` rule that conditions when the Yupik allomorph *-g-* surfaces, taking stems into account.

Removing the repeated rule cascade constitutes a major improvement to the size and efficiency of the Yupik morphological analyzer. The PFM Analyzer is nearly half the size of the Chen and Schwartz (2018) Analyzer: 4,787 states with 8,626 arcs versus 8,277 states with 17,055 arcs. It also processed a dataset containing 91K tokens three times as fast, finishing in approximately 15.03 seconds versus 47.12 seconds for the Chen and Schwartz (2018) Analyzer.

In summary, the changes to the `lexc` and `foma` files afforded to us by applying the tenets of PFM can be summarized as follows. We have shifted the burden of work from the `foma` file to the `lexc` file, such that linguistic phenomena such as allomorphy that was once handled in `foma` are now handled in `lexc`. By simplifying and reducing the number of rules we must implement in `foma`, we consequently minimize the likelihood of rule conflicts as well as coding errors on behalf of the programmer. Any coding errors would presumably be more egregious in `foma`, since the `foma` rewrite rules affect *all* lexical items coded in the `lexc` files, while errors in `lexc` would only affect the lexical items in a particular path.

6. Evaluating the Chen and Schwartz (2018) Analyzer vs. the PFM Analyzer

Of the 18 chapters in the Jacobson (2001) reference grammar, we have successfully re-implemented the morphophonological rules documented in the first 10 chapters and are currently implementing the rest. While this means that the PFM Analyzer is not yet complete, we can still judge the performance of the implementation thus far, and compare it to the performance of an earlier version of the Chen and Schwartz (2018) Analyzer that had also implemented the morphophonological rules described up to Chapter 10.

As will be shown in this section, the PFM Analyzer consistently outperforms this earlier version of the Chen and Schwartz (2018) Analyzer (abbreviated to **Ch10 C&S** as needed) with respect to coverage across several datasets.

6.1. Evaluation Datasets

We evaluated the Chapter 10 Chen and Schwartz (2018) Analyzer and the PFM Analyzer on two datasets, the first of which consisted of the end-of-chapter translation exercises from Chapters 3–10 of the Jacobson (2001) reference grammar[3]. This included 187 Yupik sentences, comprising 470 tokens and 347 types. To this, we added 264 tokens and 228 types from Chapters 11–17, whose grammatical rules were introduced in the chapters that had already been implemented. This increased our token and type count to 734 tokens and 575 types, respectively. The Jacobson (2001) test set is the only evaluation set at present with gold standard morphological analyses.

Our second test set will subsequently be referred to as the *Stories Test Set*, since all of the texts included therein may be considered stories. This evaluation set encompassed three short story collections, five anthologies of orally-narrated Yupik folk tales, the Yupik translation of the New Testament (Wycliffe, 2018), and 51 elementary readers from the school in Gambell, Alaska on St. Lawrence Island. The three short story collections (Apassingok et al., 1993; Apassingok et al., 1994; Apassingok et al., 1995) form a reading series designed for fluent Yupik speakers in grades 4–6, and are ordered with respect to difficulty as indicated by their levels, 1–3. Of the five anthologies of oral narrations, three comprise a trilogy known as *Sivuqam Nangaghnegha* (Apassingok et al., 1985; Apassingok et al., 1987; Apassingok et al., 1989). The fourth anthology is a standalone text (Nagai, 2001), as is the fifth, *Ungipaghaghlanga* (Koonooka, 2003).

[3]Only Chapters 3–17 of the reference grammar offer end-of-chapter translation exercises.

6.2. Evaluation Results for the Jacobson (2001) Test Set

The gold standard morphological analyses for the Jacobson (2001) test set were developed with the assistance of a native Yupik speaker trained in linguistics. In Table 5, we report the precision, recall, and f-measures scores achieved by each analyzer on this test set, calculated using Equations 3–5.

$$Precision = \frac{\text{\# of Items with a Correct Analysis}}{\text{\# of Items with an Analysis}} \quad (3)$$

$$Recall = \frac{\text{\# of Items with a Correct Analysis}}{\text{\# of Items}} \quad (4)$$

$$F\text{-}Measure = 2 * \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (5)$$

As can be seen, the PFM Analyzer reports higher recall for types and tokens than the Chapter 10 Chen and Schwartz (2018) Analyzer without losing any points of precision. This consequently results in a higher f-measure score.

Types	**Precision**	**Recall**	***F*-Measure**
Ch10 C&S	99.63	94.43	96.96
PFM	99.63	94.78	97.06

Tokens	**Precision**	**Recall**	***F*-Measure**
Ch10 C&S	99.71	95.23	97.42
PFM	99.71	95.50	97.50

Table 5: Precision, recall, and f-measure scores for each analyzer over *types* and *tokens* in the Jacobson (2001) end-of-chapter exercises.

Furthermore, upon closely examining the words that each analyzer failed to analyze, we observed that there was in fact significant overlap in these word lists. This suggests that the PFM Analyzer is not only handling all of the words that the Chapter 10 Chen and Schwartz (2018) Analyzer handled, but given its higher recall, is also analyzing words that the Chapter 10 Chen and Schwartz (2018) Analyzer could not.

Two words of particular note are **tengegkayugteggun** and **qergesengi**, which may be considered exceptions to typical inflection patterns. Specifically, the *-teggun* inflectional morpheme of **tengegkayugteggun** is an alternative form of the *perlative plural* that suffixes to roots that end in consonants only. Since the PFM Analyzer recognizes this class of roots: those that end in a consonant, we can readily account for this exception by encoding this extra inflectional morpheme in the appropriate `lexc` files.

Likewise, the inflectional morpheme *-ngi* in **qergesengi** ordinarily deletes stem-final *-e*, but deviates from the usual patterns of *e*-deletion in that it will not delete *-e* if the root exceeds two syllables in length, as it does here: **qer.ge.se**. Again, the PFM Analyzer recognizes roots that end in *-e* and exceed two syllables as constituting their own class, so we can account for this exception to *e*-deletion in `lexc`.

In contrast, the Chen and Schwartz (2018) Analyzer would have had to handle these exceptions by way of `foma` rewrite rules that realize the atypical behaviors of *-teggun* and *-ngi*. Since **tengegkayugteggun** and **qergesengi** received no analysis from the Chapter 10 Chen and Schwartz (2018) Analyzer, it is assumed that such rules had not been successfully implemented.

6.3. Evaluation Results for the Stories Test Set

Since all of the texts in this second evaluation set lack gold standard morphological analyses, we simply report each analyzer's *coverage* over these datasets. We define coverage as follows, and report each analyzer's coverage rate over types in Table 6 and over tokens in Table 7.

$$Coverage = \frac{\text{\# of Items with an Analysis}}{\text{\# of Items}} \quad (6)$$

Text	**% Coverage**		**# Types**
	Ch10 C&S	PFM	
Nagai (2001)	10.91	13.09	1146
Level 1	15.96	17.04	2318
Level 2	13.91	14.70	2660
Level 3	14.70	15.75	3149
Sivuqam Vol.1	9.13	10.07	4241
Sivuqam Vol.2	10.52	11.31	7188
Sivuqam Vol.3	10.79	11.62	7649
Ungipaghaghlanga	12.94	13.64	6290
Readers	13.04	13.77	12,945
New Testament	7.54	9.27	31,878
TOTAL	**10.32**	**11.52**	79,464

Table 6: Coverage rates of the two analyzers over types for each collection of texts in the *Stories Test Set*. The last line reports the *total* coverage over all types in this test set.

Text	**% Coverage**		**# Tokens**
	Ch10 C&S	PFM	
Nagai (2001)	14.56	17.46	1827
Level 1	20.46	22.09	3377
Level 2	18.27	19.60	4341
Level 3	22.90	24.75	4682
Sivuqam Vol.1	16.39	19.83	6743
Sivuqam Vol.2	16.26	18.72	11,809
Sivuqam Vol.3	17.61	20.33	12,907
Ungipaghaghlanga	20.50	21.51	14,412
Readers	22.42	23.75	27,213
New Testament	15.90	21.29	90,856
TOTAL	**17.74**	**21.39**	178,167

Table 7: Coverage rates of the two analyzers over tokens for each collection of texts in the *Stories Test Set*. The last line reports the *total* coverage over all tokens in this test set.

As expected, the coverage of both analyzers is significantly lower for this second evaluation set when compared to the Jacobson (2001) test set, since the phonological alternations introduced in Chapters 11–18 of the reference grammar have not yet been implemented as `foma` rewrite rules.

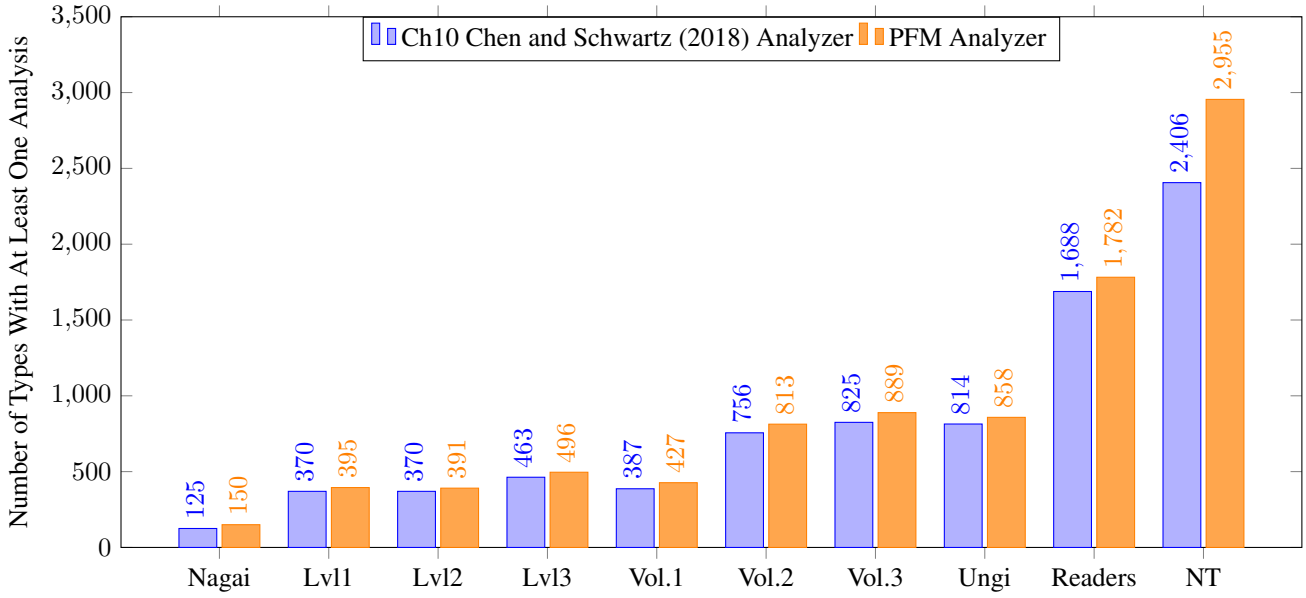

Figure 8: For each collection of texts in the *Stories Test Set*, depicts the number of types for which each analyzer returned at least one analysis.

Likewise, a substantial portion of the Yupik lexicon has not yet been added to either analyzer's `lexc`. Nevertheless, it is clear from both Tables 6 and 7 that the PFM Analyzer *consistently* reports better coverage than the Chapter 10 Chen and Schwartz (2018) Analyzer for every digitized text over types and tokens respectively. As shown in Figure 8, these improvements are at times slight, but they amplify as the size of the dataset increases. For instance, the PFM Analyzer returns an analysis for approximately *500* more types than the Chapter 10 Chen and Schwartz (2018) Analyzer on the New Testament dataset alone. While not all of these analyses may be correct, having one available provides a valuable starting point, especially as we work towards developing a gold standard corpus of morphological analyses to evaluate future iterations of the analyzer.

6.4. Other Notable Improvements

To close, we have also improved other aspects of the Yupik morphological analyzer that have not yet been discussed. In particular, there are two words **siqinghem** and **uyviinghet** that not even the final version of the Chen and Schwartz (2018) Analyzer can analyze. This is due to the substring *-ngh-*, which can parse as either *-ng-* and *-h-* or *-n-* and *-gh-*. This issue arose since the multicharacter Yupik graphemes *-ng-* and *-gh-* were implemented as MULTICHARACTER SYMBOLS in the Chen and Schwartz (2018) Analyzer, a design decision known to interfere with morphological analysis in `foma`. Having reduced our usage of multicharacter symbols in the PFM Analyzer by more careful construction of the `foma` rewrite rules, we can now successfully analyze **siqinghem** and **uyviinghet** and other word forms of this type.

7. Discussion

Despite some remaining work, the PFM Analyzer has already demonstrated many advantages over the Chen and Schwartz (2018) Analyzer. Most importantly, the PFM Analyzer successfully applies the tenets of PFM to Yupik morphology and reduces the number of morphophonological symbols that appear in the lexical string. This consequently simplified the `foma` rewrite rules to reduce programming-related errors and to promote programming efficiency by removing the need for the repeated rule cascade. As a result, the improved coverage will help us build a gold standard corpus of morphological analyses. Though we will still need human annotators to validate the PFM Analyzer's outputs, its higher coverage rate will still equate to more efficient use of available resources.

The improvements in the coverage of the PFM Analyzer also lends support to the efficacy of the PFM theory in the context of Yupik derivational and inflectional morphology. The successful implementation so far suggests that other languages with rich morphology, especially other Inuit-Yupik languages or polysynthetic languages, may benefit from adopting this alternative way of understanding complex morphology. This may even result in the implementation of better-performing computational systems, as we have demonstrated herein.

Moreover, adopting PFM to model Yupik morphology has important pedagogical implications. While the Jacobson (2001) reference grammar has been the standard for teaching Yupik morphology in a high school or university setting, PFM provides an alternative way to teach Yupik and by extension, an alternative way of *learning* Yupik. By recognizing the existence of inflection classes and their patterns of suffixation, the PFM approach not only fits into an established pedagogical framework of inflection classes, paradigms, and memorization of these paradigms, as is evident in modern approaches to teaching Latin and the Romance languages, it may also be more effective for younger learners and students who are used to learning a language this way. While the Jacobson (2001) reference grammar

provides a compact way of representing a morpheme and all the possible phonological alternations it may involve, PFM offers a different way of teaching morphology based on multiple examples and pattern discovery. This has great potential for future pedagogical materials and other downstream applications that can assist with the revitalization of this endangered language.

8. Conclusion

This paper reports our ongoing effort to implement a PFM-based morphological analyzer for St. Lawrence Island Yupik. It demonstrates the PFM Analyzer's improvements over an earlier version of the Chen and Schwartz (2018) Analyzer in terms of coverage, parsimony and efficiency, as well as improvements over the final Chen and Schwartz (2018) Analyzer (§ 6.4.). In sum, the implementation of the new analyzer thus far provides support for PFM in the context of both derivational and inflectional Yupik morphology, and has significant implications for research and downstream applications.

9. Bibliographical References

Anders Apassingok, (Iyaaka), et al., editors. (1985). *Sivuqam Nangaghnegha — Siivanllemta Ungipaqellghat / Lore of St. Lawrence Island — Echoes of our Eskimo Elders*, volume 1: Gambell. Bering Strait School District, Unalakleet, Alaska.

Anders Apassingok, (Iyaaka), et al., editors. (1987). *Sivuqam Nangaghnegha — Siivanllemta Ungipaqellghat / Lore of St. Lawrence Island — Echoes of our Eskimo Elders*, volume 2: Savoonga. Bering Strait School District, Unalakleet, Alaska.

Anders Apassingok, (Iyaaka), et al., editors. (1989). *Sivuqam Nangaghnegha — Siivanllemta Ungipaqellghat / Lore of St. Lawrence Island — Echoes of our Eskimo Elders*, volume 3: Southwest Cape. Bering Strait School District, Unalakleet, Alaska.

Anders Apassingok, (Iyaaka), et al., editors. (1993). *Kallagneghet / Drumbeats*. Bering Strait School District, Unalakleet, Alaska.

Anders Apassingok, (Iyaaka), et al., editors. (1994). *Akiingqwaghneghet / Echoes*. Bering Strait School District, Unalakleet, Alaska.

Anders Apassingok, (Iyaaka), et al., editors. (1995). *Suluwet / Whisperings*. Bering Strait School District, Unalakleet, Alaska.

Badten, L. W., Kaneshiro, V. O., Oovi, M., and Koonooka, C. (2008). *St. Lawrence Island / Siberian Yupik Eskimo Dictionary*. Alaska Native Language Center, University of Alaska Fairbanks.

Beesley, K. R. and Karttunen, L. (2003). *Finite State Morphology*. CLSI Publications, Palo Alto, California.

Chen, E. and Schwartz, L. (2018). A morphological analyzer for St. Lawrence Island / Central Siberian Yupik. In *Proceedings of the 11th Language Resources and Evaluation Conference*, Miyazaki, Japan, May.

de Reuse, W. J. (1994). *Siberian Yupik Eskimo — The Language and Its Contacts with Chukchi*. Studies in Indigenous Languages of the Americas. University of Utah Press, Salt Lake City, Utah.

Hulden, M. (2009). Foma: A finite-state compiler and library. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 29–32. Association for Computational Linguistics.

Jacobson, S. A. (2001). *A Practical Grammar of the St. Lawrence Island / Siberian Yupik Eskimo Language, Preliminary Edition*. Alaska Native Language Center, Fairbanks, Alaska, 2nd edition.

Karttunen, L. (2003). Computing with realizational morphology. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 203–214. Springer.

Koonooka, C. P. (2003). *Ungipaghaghlanga: Let Me Tell You A Story*. Alaska Native Language Center.

Krauss, M. (1975). St. Lawrence Island Eskimo phonology and orthography. *Linguistics: An International Review*, 13(152):39–72, January.

Menovshchikov, G. A. (1962). *Grammatika iazyka aziatskikh eskimosov (Grammar of the language of Asian Eskimos)*, volume 1. Izdatel'stvo akademii Nauk (Academy of Sciences of the USSR), Moscow and Leningrad.

Nagai, K. (2001). *Mrs. Della Waghiyi's St. Lawrence Island Yupik Texts with Grammatical Analysis*. Number A2-006 in Endangered Languages of the Pacific Rim. Nakanishi Printing, Kyoto, Japan.

Stump, G. T. (2001). *Inflectional morphology: A theory of paradigm structure*, volume 93. Cambridge University Press.

Wycliffe. (2018). *Yupik New Testament*. Wycliffe Bible Translators, Saint Lawrence Island, Alaska.