Finding regularity in Tlingit verb prefixes

Marcel K. Goh

Abstract. We describe the results of a semi-computational search for regularity in the Tlingit verb prefix charts found in [4]. We present a set of twenty-eight rewrite rules that underlie phonological and morphological changes in the verb, and give an explicit sequence of rewrite rules that resolves every entry in the charts.

1. Introduction

The Tlingit language is a Na-Dene language spoken in Alaska, British Columbia, and Yukon. It is an endangered language, with fewer than 200 living native speakers. As in many other Na-Dene languages, Tlingit verbs have complex internal structures that pack a lot of meaning into relatively few syllables, when contrasted with verbs in other languages, or even Tlingit nouns [3]. At face value, the Tlingit verb looks like a single word, but when translated to English, a single verb may correspond to a whole sentence. This is because information such as subject, aspect, mood, etc. which are indicated by entire words in English, are instead indicated by verb prefixes in Tlingit. In particular, they undergo various phonological and morphological changes. For example, the perfective aspect `wu-` may appear under the guise of `u-` or `w-`, depending on the phonological context. In this paper, we describe a semi-computational attempt to find patterns in these transformations and develop rules that allow one to decompose an inflected verb into sequences of underlying prefixes.

Computational linguistic research has largely steered clear of the Na-Dene languages, for reasons that we will not attempt to speculate at here. Exceptions are [5], which presents a computational parsing tool for the Navajo verb, and [1], which models the verbs of Tsuut'ina. Both of these projects used finite-state methods that are bidirectional, meaning they can both parse and generate verbs. The program we present in this paper is not a parser, and in fact does not produce any usable output at all. Instead, the program simply reports when regularity is found, allowing one to test the robustness of various rewrite rules.

2. Sequences of states

This section will characterise the verb charts [4] as a function from sequences of underlying morphemes to sequences of observed prefixes. The main problem this paper addresses is an inverse one. Given a sequence of prefixes, is it possible to exploit various patterns in the data to determine the sequence(s) of underlying morphemes that it represents?

Strings and tags. Formally, our program deals with the monoid of all strings over a finite alphabet A (the set of all letters in the Tlingit orthography), with the binary operation of concatenation; strings in this set will be written in a fixed-width typeface to distinguish them from the ordinary text of the paper. As an exception to this convention, we use ϵ to denote the identity element (the empty string). The data found in [4] is labelled, in the sense that every string has been partitioned into substrings, each tagged with the type of prefix they represent. These tags are important to our program, so we will list all of them here. We have included page references to [3], where these prefixes are described in detail.

i) A *disjunct prefix* (p. 8) is tagged with Q. This may be a qualifier prefix, an incorporated noun prefix, or an object prefix.

ii) The *irrealis prefix* (p. 633) is tagged with R. The only two possibilities here are $R(\mathtt{u})$ and $R(\mathtt{w})$.

iii) *Aspect prefixes* are tagged with A. An example is the perfective prefix (p. 541), which can either be $A(\mathtt{wu})$ or $A(\mathtt{u})$.

iv) The *modality prefix* (p. 496) is tagged with M. The underlying form is always $M(\underline{\mathtt{g}})$.

v) *Subject prefixes* are tagged with S.

vi) The passive (p. 402), antipassive (p. 379), or middle (p. 415) voice is indicated by a `d-` *prefix*, which is tagged with D.

vii) Any of the `s-`, `l-`, *or* `sh-` *prefixes* are classified under the F tag, and we only deal with the `s` case (p. 239), though it is noted in [4] that the phonological patterns are more or less analogous in the other two cases.

viii) The stative `i`-*prefix* (p. 179) is given the I tag.

ix) *Epenthesis* is indicated by the E tag.

Our program makes crucial use of these tags to detect when a sequence of observed prefixes corresponds exactly to a sequence of morphemes. We denote by A^* the set of all words formed from letters in the orthography, and if the nine tags above are viewed as functions, we can let B be the union of images of $A^* \setminus \{\epsilon\}$ under each of the nine functions above. We will call the elements of B *states*; an example is $S(\texttt{yi})$. Note that we never tag the empty string ϵ. Next, we consider the be the set of all nonempty sequences of states, denoted B^+; for instance, the sequence $Q(\texttt{ka})S(\underline{\texttt{x}})D(\texttt{d})I(\texttt{i})$ is an element of this set. We will let $S = B^+ \cup \{\epsilon\}$, where ϵ is not tagged. This identity element will be important later when we start defining rewrite rules as functions from S to itself.

Hidden and observed sequences. Although the data that the program uses is formatted as a table, it is simpler to think of it as a map (or dictionary) from a set of sequences of underlying morphemes to a set of actual observed sequences of prefixes. When sequences of tags are the same, it is clear which prefixes are represented by each part of the verb. For example, one of the entries in the dictionary is

$$Q(\texttt{ji})A(\texttt{wu})S(\texttt{du})D(\texttt{d})F(\texttt{s})I(\texttt{i}) \mapsto Q(\texttt{ji})A(\texttt{m})S(\texttt{du})D(\texttt{d})F(\texttt{z})I(\texttt{i}) \tag{1}$$

and it is clear which substrings of the actual verb correspond to respective prefixes. We will say that such an entry is *resolved* and *unresolved* otherwise. An example of an unresolved entry is

$$Q(\texttt{du})R(\texttt{w})A(\texttt{g})M(\underline{\texttt{g}})S(\texttt{du}) \mapsto Q(\texttt{du})A(\texttt{g})E(\texttt{a})M(\underline{\texttt{x}})S(\texttt{du}), \tag{2}$$

where the correspondence between states is not immediately obvious. Looking at more entries in the dictionary, we see that the transformation

$$A(\texttt{g})E(\texttt{a})M(\underline{\texttt{x}}) \to R(\texttt{w})A(\texttt{g})M(\underline{\texttt{g}}) \tag{3}$$

occurs a fair few times, so we might consider applying this transformation to every string in the dictionary and see if it causes more entries to be resolved. This suggests a semi-computational approach to finding regularity in the charts, for if we can cook up a relatively small set of transformations that seem to underlie all of the phonological and morphological irregularity in the Tlingit verb, we can use a computer program to verify if the entirety of the charts can be covered by the rules.

3. Rewrite rules

For $x, y \in S$, $x \neq \epsilon$, we formally define a *transformation* $x \to y$ to be a function from S to itself that replaces the *first* instance of x in a word $w \in S$ with y. A *rewrite rule* is a finite sequence of transformations. Although transformations are formally functions and functions are applied from right to left, we will apply transformations in a rewrite rule from left to right, to preserve the sanity of the (presumably English-speaking) reader. So if T_1, T_2, and T_3 are transformations, the rewrite rule $f : S \to S$ given by $f = T_1 \circ T_2 \circ T_3$ will be written T_3, T_2, T_1. This section will enumerate the twenty-eight rewrite rules that we will use to resolve every entry in the dictionary obtained from the charts [4].

Epenthesis and irregular disjuncts. With the luxury of having tagged data, an obvious thing to do is to remove all instances of epenthesis in the dictionary, since, more or less by definition, epenthesis has no underlying meaning. So we would like a rule along the lines of $E(*) \to \epsilon$ repeated infinitely many times, where the star stands for any possible string, but this does not satisfy our requirements for a well-defined

rewrite rule. Thankfully, in our data the only letters that are ever epenthesised are a, i, and o. In the case of a, there are at most four instances of epenthesis in a given word, and in the other two cases there is at most one instance, so the rewrite rule

i) $E(\mathtt{a}) \to \epsilon, E(\mathtt{a}) \to \epsilon, E(\mathtt{a}) \to \epsilon, E(\mathtt{a}) \to \epsilon, E(\mathtt{i}) \to \epsilon, E(\mathtt{o}) \to \epsilon$

successfully eliminates all epenthesis from the dictionary. Another rewrite rule that simply cleans up the data is the regularisation of certain irregular disjuncts:

ii) $Q(\mathtt{e}) \to Q(\mathtt{a}), Q(\mathtt{k}) \to Q(\mathtt{ka}), Q(\underline{\mathtt{x}}') \to Q(\underline{\mathtt{x}}'\mathtt{e}), Q(\mathtt{j}) \to Q(\mathtt{ji}), Q(\mathtt{t}) \to Q(\mathtt{tu})$

Labialised velars and uvulars. When a velar or uvular stop or fricative is followed by a w, the underlying form often has a wu or u (or variants thereof) *preceding* the consonant. These represent the perfective aspect and irrealis mood, respectively. In the case of the subject $\underline{\mathtt{x}}$, the two cases are handled by the rule

iii) $S(\underline{\mathtt{x}})A(\mathtt{w}) \to A(\mathtt{wu})S(\underline{\mathtt{x}}), S(\underline{\mathtt{x}})R(\mathtt{w}) \to R(\mathtt{u})S(\underline{\mathtt{x}}), S(\underline{\mathtt{k}})R(\mathtt{w}) \to R(\mathtt{u})S(\underline{\mathtt{k}})$.

Note that we didn't change $S(\underline{\mathtt{k}})$ to $S(\underline{\mathtt{x}})$ in the third transformation; this will be done later. In the case that the consonant indicates aspect or modality, the w can only indicate the irrealis mood:

iv) $A(\mathtt{k})R(\mathtt{w}) \to R(\mathtt{u})A(\mathtt{g}), A(\underline{\mathtt{g}})R(\mathtt{w}) \to R(\mathtt{u})A(\underline{\mathtt{g}})$

v) $M(\underline{\mathtt{g}})R(\mathtt{w}) \to R(\mathtt{u})M(\underline{\mathtt{g}})$

Elision. It is quite difficult to deal with elision using rewrite rules, since one has to guess where the missing state should be inserted. An example of this annoyance is the systematic elision of the d- prefix when it is followed by the s- prefix. A solitary s- prefix is indicated by $F(\mathtt{s})E(\mathtt{a})$, but since it is convenient to delete epenthesis early in the rewriting process, we are left with a situation where $F(\mathtt{s})$ may stand for either $D(\mathtt{d})F(\mathtt{s})$ or $F(\mathtt{s})$. Thus we have the rule

vi) $F(\mathtt{s}) \to D(\mathtt{d})F(\mathtt{s})$,

as well as its "inverse" $D(\mathtt{d})F(\mathtt{s}) \to F(\mathtt{s})$. (Another approach to this problem would be to introduce a dummy state, call it $\vdash$, marking the end of a sequence, and then transforming $F(\mathtt{s})E(\mathtt{a})$ into $F(\mathtt{s})$ and $F(\mathtt{s}) \vdash$ into $D(\mathtt{d})F(\mathtt{s})$.) Ambiguity due to elision also occurs with the prefixes $R(\mathtt{u})$ and $A(\mathtt{wu})$. We will defer a more general discussion of resolving this ambiguity by "chaining" rewrites to the next section.

Modality prefix. The modality prefix $M(\underline{\mathtt{g}})$ causes various shuffling of prefixes, when it coincides with the aspect prefix $A(\mathtt{g})$ and its variants. The first thing to do is to simplify the problem a bit by modifying the following two strings (but not their tags):

vii) $M(\underline{\mathtt{x}}) \to M(\underline{\mathtt{g}}), A(\mathtt{k})M(\underline{\mathtt{g}}) \to A(\mathtt{g})M(\underline{\mathtt{g}})$

This rule will not cause the further resolution of any entries, but reduces the number of transformations we need in the following few rules. The first of these is a rule that removes an irrealis prefix in the vicinity.

viii) $A(\mathtt{g})M(\underline{\mathtt{g}})R(\mathtt{w}) \to A(\mathtt{g})M(\underline{\mathtt{g}}), A(\mathtt{g})R(\mathtt{w})M(\underline{\mathtt{g}}) \to A(\mathtt{g})M(\underline{\mathtt{g}})$

This prefix may need to be added back later, but it will precede the aspect prefix:

ix) $A(\mathtt{g})M(\underline{\mathtt{g}}) \to R(\mathtt{w})A(\mathtt{g})M(\underline{\mathtt{g}}), A(\underline{\mathtt{g}})M(\underline{\mathtt{g}}) \to R(\mathtt{w})A(\underline{\mathtt{g}})M(\underline{\mathtt{g}})$

Next, we have two "cleanup" rules that handle the mess that happens when the modality prefix clashes with a $S(\underline{\mathtt{x}})$ prefix:

x) $A(\mathtt{k})S(\underline{\mathtt{k}})R(\mathtt{w}) \to R(\mathtt{w})A(\mathtt{g})M(\underline{\mathtt{g}})S(\underline{\mathtt{x}}), A(\mathtt{k})R(\mathtt{u})S(\underline{\mathtt{k}}) \to R(\mathtt{u})A(\mathtt{g})M(\underline{\mathtt{g}})S(\underline{\mathtt{x}}),$
$A(\mathtt{k})R(\mathtt{w})S(\underline{\mathtt{k}}) \to R(\mathtt{w})A(\underline{\mathtt{g}})M(\underline{\mathtt{g}})S(\underline{\mathtt{x}})$

xi) $A(\mathtt{k})S(\underline{\mathtt{k}}) \to A(\mathtt{g})M(\underline{\mathtt{g}})S(\underline{\mathtt{x}}), A(\mathtt{g})S(\underline{\mathtt{k}}) \to A(\mathtt{g})M(\underline{\mathtt{g}})S(\underline{\mathtt{x}}), A(\underline{\mathtt{k}})S(\underline{\mathtt{k}}) \to A(\underline{\mathtt{g}})M(\underline{\mathtt{g}})S(\underline{\mathtt{x}})$

Velar aspect prefixes. The next few rules handle the irrealis prefix in the vicinity of a velar aspect prefix. In most cases the irrealis is elided, so we must guess where it originally stood. First we have the rule

xii) $A(\mathtt{g})R(\mathtt{o}) \to R(\mathtt{u})A(\mathtt{g})$,

and to make our life easier, we will also regularise some of the subject prefixes:

xiii) $S(\texttt{y}) \to S(\texttt{i}), S(\texttt{ee}) \to S(\texttt{i}), S(\texttt{yee}) \to S(\texttt{yi}), S(\texttt{ye}) \to S(\texttt{yi}), S(\texttt{too}) \to S(\texttt{tu}), S(\underline{\texttt{k}}) \to A(\underline{\texttt{g}})S(\underline{\texttt{x}})$

This allows us to systematically add $R(\texttt{u})$ before the aspect prefix:

xiv) $A(\texttt{k})S(\texttt{du}) \to R(\texttt{u})A(\texttt{g})S(\texttt{du}), A(\texttt{g})S(\texttt{du}) \to R(\texttt{u})A(\texttt{g})S(\texttt{du}),$
$A(\underline{\texttt{x}})S(\texttt{du}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{du}), A(\underline{\texttt{g}})S(\texttt{du}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{du})$

xv) $A(\texttt{g})S(\texttt{i}) \to R(\texttt{u})A(\texttt{g})S(\texttt{i})$

xvi) $A(\texttt{g})S(\texttt{yi}) \to R(\texttt{u})A(\texttt{g})S(\texttt{yi}), A(\underline{\texttt{x}})S(\texttt{yi}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi}), A(\underline{\texttt{g}})S(\texttt{yi}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi}),$
$A(\underline{\texttt{g}})S(\texttt{i}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi}), A(\underline{\texttt{g}})S(\texttt{ee}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{yi})$

xvii) $A(\texttt{k})S(\texttt{tu}) \to R(\texttt{u})A(\texttt{g})S(\texttt{tu}), A(\texttt{g})S(\texttt{tu}) \to R(\texttt{u})A(\texttt{g})S(\texttt{tu}),$
$A(\underline{\texttt{x}})S(\texttt{tu}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{tu}), A(\underline{\texttt{g}})S(\texttt{tu}) \to R(\texttt{u})A(\underline{\texttt{g}})S(\texttt{tu}),$

Lastly, because the $\underline{\texttt{gx}}$ pattern performs double duty to indicate both aspect and modality, we will need to a rule to switch from aspect prefixes to modality prefixes:

xviii) $A(\underline{\texttt{g}})S(\underline{\texttt{x}}) \to M(\underline{\texttt{g}})S(\underline{\texttt{x}})$

Perfective aspect. The perfective aspect is elided before $S(\texttt{i})$, so we can resolve a bunch of entries with the rule

xix) $Q(\texttt{a})S(\texttt{i}) \to Q(\texttt{a})A(\texttt{wu})S(\texttt{i}), Q(\texttt{ka})S(\texttt{i}) \to Q(\texttt{ka})A(\texttt{wu})S(\texttt{i}), Q(\underline{\texttt{x}}'\texttt{a})S(\texttt{i}) \to Q(\underline{\texttt{x}}'\texttt{e})A(\texttt{wu})S(\texttt{i}),$
$Q(\texttt{ji})S(\texttt{i}) \to Q(\texttt{ji})A(\texttt{wu})S(\texttt{i}), Q(\texttt{tu})S(\texttt{i}) \to Q(\texttt{tu})A(\texttt{wu})S(\texttt{i}), A(\texttt{e})S(\texttt{e}) \to Q(\texttt{a})A(\texttt{wu})S(\texttt{i}).$

The last of these transformations handles only a single odd case.

Aspect n- prefix. A couple of problems arise with the irrealis and modality prefixes in the vicinity of the $A(\texttt{n})$ prefix. First, we guess the existence of an irrealis prefix with

xx) $A(\texttt{n}) \to R(\texttt{u})A(\texttt{n}),$

then we insert a missing modality prefix with the transformations

xxi) $A(\texttt{n})S(\texttt{yi}) \to A(\texttt{n})M(\underline{\texttt{g}})S(\texttt{yi}), A(\texttt{n})S(\texttt{du}) \to A(\texttt{n})M(\underline{\texttt{g}})S(\texttt{du}).$

Irrealis cleanup. The last few rules have to do with the pesky irrealis prefix, and are ever so slightly more "destructive". First, we have the rule

xxii) $Q(\texttt{a}) \to Q(\texttt{a})R(\texttt{u}), Q(\texttt{ka}) \to Q(\texttt{ka})R(\texttt{u}), Q(\underline{\texttt{x}}'\texttt{a}) \to Q(\underline{\texttt{x}}'\texttt{e})R(\texttt{u}),$
$Q(\texttt{ji}) \to Q(\texttt{ji})R(\texttt{u}), Q(\texttt{tu}) \to Q(\texttt{tu})R(\texttt{u}),$

which dishes out irrealis prefixes wholesale. To counter the possible doubling of irrealis prefixes in the previous rule, we have the rule

xxiii) $R(\texttt{e})R(\texttt{u}) \to R(\texttt{u}), R(\texttt{i})R(\texttt{u}) \to R(\texttt{u}), R(\texttt{o})R(\texttt{u}) \to R(\texttt{u}), R(\texttt{u})R(\texttt{u}) \to R(\texttt{u}),$

which also takes the opportunity to delete any funky renditions of this prefix. We also try adding irrealis prefixes before the modality prefix:

xxiv) $M(\underline{\texttt{g}}) \to R(\texttt{u})M(\underline{\texttt{g}})$

Some of these rules undoubtedly make a big mess of things, so we will include a rule that deletes an irrealis prefix, no questions asked:

xxv) $R(\texttt{u}) \to \epsilon$

We include a rule

xxvi) $Q(\texttt{a})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{a})R(\texttt{u})S(\texttt{i}), Q(\texttt{ka})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{ka})R(\texttt{u})S(\texttt{i}),$
$Q(\underline{\texttt{x}}'\texttt{e})A(\texttt{wu})S(\texttt{i}) \to Q(\underline{\texttt{x}}'\texttt{e})R(\texttt{u})S(\texttt{i}), Q(\texttt{ji})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{ji})R(\texttt{u})S(\texttt{i}),$
$Q(\texttt{tu})A(\texttt{wu})S(\texttt{i}) \to Q(\texttt{tu})R(\texttt{u})S(\texttt{i})$

that repurposes the pattern we created in rule (xix), and for subject prefixes not preceded by a `wu-`, we have the rule

xxvii) $S(\texttt{tu}) \to R(\texttt{u})S(\texttt{tu}), S(\texttt{du}) \to R(\texttt{u})S(\texttt{du}), S(\texttt{i}) \to R(\texttt{u})S(\texttt{i}),$
$S(\texttt{yeey}) \to R(\texttt{u})S(\texttt{yeey}), S(\texttt{yi}) \to R(\texttt{u})S(\texttt{yi}).$

A final oddity. There are two entries that require the creation of a $Q(\texttt{u})$ disjunct prefix, so our last rule is

xxviii) $S(\texttt{i}) \to Q(\texttt{u})S(\texttt{i}).$

4. An explicit sequence of rewrites

In the previous section we described twenty-eight rewrite rules and the claim is that these rules are enough to resolve every entry in the charts found in [4]. Given an entry, a naïve way to check which rules apply is to simply try every combination. However, because multiple rewrites may be needed and because composition of rewrite rules is not commutative (the identity $f \circ g = g \circ f$ does not hold in general), for every k rewrites we would like to apply, there are $k!$ distinct orderings that may each produce a different result. So *a priori* there are

$$\sum_{k=1}^{28} k! \binom{28}{k} \approx 8.29 \times 10^{29} \tag{4}$$

possibilities to check. Assuming each check takes a billionth of a second (which is rather optimistic), running through all of these possibilities will take several orders of magnitude longer than the current age of the universe to terminate. Of course, there are numerous ways one might be able to reduce these possibilities and make enumeration work, but we will take a different approach.

A function $f : S \to S$ is said to be *invertible* if there exists a function $f^{-1} : S \to S$ such that $(f^{-1} \circ f) = 1_S$, the identity function on S. The following negative result shows that nontrivial transformations are not invertible.

Lemma A. *Let S be the set of all sequences of tagged strings as defined above. Let $T : S \to S$ be a transformation such that for all $w \in S$, $f(w)$ is obtained by replacing the first occurrence of $x \in S$ in w with $y \in S$. Then T is invertible if and only if T is the identity transformation (which occurs precisely when $x = y$).*

Proof. If T is the identity 1_S, then we may take $T^{-1} = 1_S$. For the "only if" direction, suppose that T is not the identity transformation; that is, x is a nonempty string and $y \neq x$. We note that $T(x) = T(y) = y$, so that T is not injective and cannot have a well-defined inverse. ∎

Since individual transformations are not invertible and every rewrite rule is a composition of transformations, nontrivial rewrite rules are not invertible. However, many rewrite rules f we defined above are "invertible enough" for our purposes, in the sense that for a convenient set $S' \subseteq S$ (for instance, S' might be the set of entries in the dictionary at a certain point in time), there exists another rewrite rule f^{-1} such that $(f \circ f^{-1} \circ f)(w) = f(w)$ for all w in S'; that is, applying f and f^{-1} in succession on the entire set eventually causes it to oscillate between two different sets. For example, consider the function f_6 given in rule (vi) that replaces $F(\texttt{s})$ with $D(\texttt{d})F(\texttt{s})$. Because no entry in the dictionary contains more than one instance of $F(\texttt{s})$, applying f_6 causes every $F(\texttt{s})$ to be preceded by $D(\texttt{d})$. Applying the rule $D(\texttt{d})F(\texttt{s}) \to F(\texttt{s})$ does not return the dictionary to its original state, because now the sequence $D(\texttt{d})F(\texttt{s})$ has completely disappeared. But now applying f and f^{-1} in succession will cause the entries to oscillate between only two possible distinct dictionaries. For a rewrite rule f that consists of a single transformation $x \to y$, it will be convenient to call the transformation $y \to x$ its *pseudoinverse* and write f^{-1}, where it is understood that f is only invertible in the very restricted sense outlined above. The pseudoinverse of a rewrite rule $x_1 \to y_1, \ldots, x_n \to y_n$ is the rewrite rule $y_n \to x_n, \ldots, y_1 \to x_1$.

The algorithm we use to resolve entries in the dictionary runs as follows.

Algorithm R (*Find regularity*). Given a dictionary $D : S \to S$ and a nonempty ordered list `R` of rewrite rules, this algorithm tests if the rewrite rules in the list `R` successfully resolve every entry in the dictionary D. We use the notation $x \Leftarrow \texttt{R}$ to indicate removing the first element of the list `R` and assigning it to the variable x.

R1. [Apply rewrite.] Set $f \Leftarrow \mathtt{R}$. For every key-value pair (k, v) in D, apply the function f to v.

R2. [Resolved?] For every key-value pair $(k, v) \in D$, check if the entry is resolved; that is, verify if there are the same number of states in each sequence and if the tags of respective states match. If so, remove (k, v) from D.

R3. [Done?] If $\mathtt{R}$ is nonempty, return to step R1. Otherwise, output "successful" if D is empty, "unsuccessful" if D is nonempty, and terminate. ∎

It remains to describe the list of rewrites $\mathtt{R}$ that, in our experiment, successfully resolved every entry in the dictionary. We will write f_k to indicate the function represented by rule k (in Roman numerals) in the previous section; so rule (xix) corresponds to f_{19}. As before, the notation 1_S indicates the identity function on the set S. In list notation, the sequence of forty-five rewrites is

$$\begin{aligned} &1_S, f_1, f_2, f_3, f_4, f_5, f_6, {f_5}^{-1}, {f_6}^{-1}, f_7, f_8, f_9, f_{10}, f_{11}, f_6, f_6{-}1, f_{12}, f_{13}, f_{14}, f_{15}, f_{16}, f_{17}, f_{18}, f_{19}, \\ &\quad f_6, f_{6-1}, f_{20}, f_6, {f_6}^{-1}, f_{21}, {f_9}^{-1}, f_{22}, f_{23}, f_{24}, {f_6}^{-1}{f_{24}}^{-1}, {f_{18}}^{-1}f_6, f_{25}, f_{26}, f_{27}, {f_6}^{-1}, f_{25}, f_{28}. \end{aligned} \tag{5}$$

Table 1, which appears at the end of this paper, shows the number of entries resolved by each rewrite rule. Note that the same rule is often applied and inverted multiple times to test different combinations. In particular, the pattern $f_i f_j {f_i}^{-1} {f_j}^{-1}$ tests all four combinations of ${f_i}^{\pm 1}$ and ${f_j}^{\pm 1}$. This mechanism allows us to resolve some ambiguities in the surface forms. Although there are 4773 entries in the original dictionary, there are only 2621 distinct surface forms, so there are plenty of instances of the same inflected form corresponding to multiple possible underlying forms. We make no claim that (5) is the shortest sequence of rewrite rules that resolves the entire dictionary, and we suspect that many changes could be made to the rewrite rules that would decrease the number of transformations needed significantly.

5. Further directions

We presented a very restricted algorithm that tests for regularity in Tlingit verb charts. In principle, the information that our program supplies regarding the twenty-eight rewrite rules could be used by a *bona fide* computational parser in the future. In particular, it would be interesting to see if one could perform a similar exercise using strings that have not yet been tagged, perhaps using contextual information surrounding particular substrings. Another approach would be probabilistic, perhaps employing the well-known Viterbi parsing algorithm [6] and using "real-world" data from a text corpus (e.g. [2]) to statistically guess the underlying forms of inflected verbs.

Acknowledgements

The author would like to thank Prof. James Crippen for his continuous feedback throughout the semester on this draft, as well as Emi Baylor for numerous valuable discussions.

References

[1] Antti Arppe, Christopher Cox, Mans Hulden, Jordan Lachler, Sjur N. Moshagen, Miikka Silfverberg, and Trond Trosterud, "Computational modeling of the verb in Dene languages. The case of Tsuut'ina," *Working papers in Athabascan Linguistics* Red Book (2017), 51–68.

[2] James A. Crippen, *Tlingit text corpus* (GitHub repository, `https://github.com/jcrippen/tlingit-corpus`, 2015).

[3] James A. Crippen, *The syntax in Tlingit verbs*, Ph.D. thesis (University of British Columbia, Vancouver, B.C., 2019).

[4] James A. Crippen, *Tlingit verb prefixes* (GitHub repository, `https://github.com/jcrippen/tlingit-verb-prefixes`, 2020).

[5] Mans Hulden and Sharon T. Bischoff, "An experiment in computational parsing of the Navajo verb," *Coyote Papers* **16** (2008), 110–118.

[6] Andrew J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory* **13** (1967), 260–269.

Table 1

EFFECT OF SUCCESSIVE APPLICATIONS OF REWRITE RULES ON THE ENTIRE DICTIONARY

Rewrite rule	Inverse entries	Unresolved entries	Entries resolved in this step
1_S	575	4040	733
f_1	1756	2626	1414
f_2	1756	2626	0
f_3	1798	2528	98
f_4	1829	2497	31
f_5	1829	2490	7
f_6	2110	2107	383
${f_5}^{-1}$	2110	2107	0
${f_6}^{-1}$	2110	2107	0
f_7	2110	2107	0
f_8	2110	2107	0
f_9	2137	1737	370
f_{10}	2182	1677	60
f_{11}	2301	1558	119
f_6	2338	1451	107
${f_6}^{-1}$	2338	1451	0
f_{12}	2414	1345	106
f_{13}	2498	1261	84
f_{14}	2498	1225	36
f_{15}	2498	1147	78
f_{16}	2498	1057	90
f_{17}	2498	973	84
f_{18}	2589	822	151
f_{19}	2589	770	52
f_6	2621	709	61
${f_6}^{-1}$	2621	709	0
f_{20}	2621	421	288
f_6	2621	379	42
${f_6}^{-1}$	2621	379	0
f_{21}	2621	379	0
f_6	2621	373	6
${f_9}^{-1}$	2621	373	0
f_{22}	2621	205	168
f_{23}	2621	205	0
f_{24}	2621	183	22
${f_6}^{-1}$	2621	165	18
${f_{24}}^{-1}$	2621	135	30
${f_{18}}^{-1}$	2621	135	0
f_6	2621	128	7
f_{25}	2621	92	36
f_{26}	2621	54	38
f_{27}	2621	26	28
${f_6}^{-1}$	2621	18	8
f_{25}	2621	2	16
f_{28}	2621	0	2