Advantages of constituency: computational perspectives on Samoan word prosody

Anonymous

Abstract

Prosodic constituents are ubiquitous in phonologist's descriptions of a wide range of phenomena. However, whether constituents make phonological grammars more succinct is a question that has not been carefully explored computationally. Moreover, computational descriptions of phonological patterns have revealed strong structural universals without referring to constituents. In this paper, we computationally implement and compare grammars of Samoan stress patterns that refer to feet and that refer only to syllables in Karttunen (1998)'s finite state formalization of Optimality Theory, and in grammars that directly state restrictions on surface stress patterns. While succinctness (size of the grammar) is not affected by referring to feet in the direct grammars, in the OT formalism, the grammar with feet is clearly more succinct.

1 Introduction

Zuraw (2009, p. 1) states: "Phonologists are often explicit about whether they subscribe to level ordering or output-output correspondence (rarely both). But we tend to help ourselves to prosodic domains without further comment." Indeed, there is substantial evidence that phonological constituents are useful: referring to units such as syllables, feet, prosodic words and higher-level constituents (Nespor and Vogel, 1986; Selkirk, 1986),i.a., helps us uncover and describe generalizations in phonological patterns. As stated in Hayes (1995, p. 41, §3.8), referring to prosodic constituents can help phonologists succinctly describe what part of a word gets reduplicated and restrictions on minimal words (McCarthy and Prince, 1986/1996), as well as restrictions on allowed stress patterns[1](Liberman and Prince, 1977) and the domain of segmental processes (Selkirk, 1980; Nespor and Vogel, 1986). Prosodic constituents can also help us state systematic patterns about where certain tones are placed (Pierrehumbert, 1980) and variation in duration (e.g. final lengthening) (Wightman et al., 1992) and the "strength" of articulatory gestures (Fougeron and Keating, 1997). However, the driving motivation for positing constituents—*Do constituents make phonological grammars more succinct?*—has not been carefully explored computationally. Moreover, computational descriptions of phonological patterns have revealed strong structural universals without referring to constituents (Heinz, 2009, 2010, 2011b,a).[2]

[1]In this paper, when we say "stress pattern", we mean a set of sequence of syllables specified for degree of stress and weight.

[2]Some computational work has defined phonological patterns in terms of tiers (Kornai, 1991; Bird and Ellison, 1994; Eisner, 1997) from autosegmental theory (Goldsmith, 1976, 1990), but the architecture of tiers doesn't make them properly nested like constituents: for instance, it's generally assumed that no feet straddle prosodic words.

In this work, we implement and compare phonological grammars of stress patterns in Samoan monomorphs to show that grammars referring to feet are more succinct than grammars that refer only to syllables. Succinctness comparisons between grammars of the same ilk have appeared in Chomsky (1965); Chomsky and Halle (1968); Meyer and Fischer (1971); Hartmanis (1980); Stabler (2013); Berwick (2015); Rasin and Katzir (To appear), among other work. We define all grammars in `xfst` (Beesley and Karttunen, 2003), software for computing with finite state machines, and our comparison is a 2×2 experimental design because we compare grammars with and without feet in two formalisms: (a) Karttunen (1998)'s finite state Optimality Theory (OT) (Prince and Smolensky, 1993, 2004), which maps underlying forms to surface forms without an intermediate mapping to violations using a special composition operator "lenient composition", and (b) a "direct" approach which directly describes the surface patterns like in Hulden (2006) and Heinz (2009). Our work thus addresses not only the utility of constituents in phonology, but also the advantages and disadvantages of different formalisms.

The code implementing the grammars is available at AUTHOR'S LINK VIA EDITORS.

The rest of the paper is structured as follows: the remainder of this introductory section operationalizes *Do constituents make phonological grammars more succinct?* (§1.1) and describes the language of stress patterns of Samoan, Little Samoan, that our grammars are designed to capture (§1.2). We describe the four grammars in §2: the direct account with feet (§2.1), the direct account with syllables only (§2.2), the OT account with feet (§2.3), and the OT account with syllables (§2.4). Discussion of the comparison follows in §3, and the conclusion is in §4.

1.1 Operationalizing the research question

In this section, we precisely define each of the concepts in our research question: phonological grammars, constituents, and succinctness.

1.1.1 Phonological grammars and constituents

In Formal Language Theory, a grammar is defined by a finite alphabet of terminal symbols, a finite set of non-terminal categories (which shares no members in common with the alphabet), a finite set of rewrite rules, and a start symbol that initiates the derivation (Chomsky, 1956; Salomaa, 1973). The set of strings that can be derived by the grammar is defined to be the language derived by the grammar. In a tree derived by the grammar, nodes are labeled with categories, and a set of nodes form a constituent if they are exhaustively dominated by a common node.

The nature of restrictions on the structure of rewrite rules determine the structural complexity of the grammar. A standard definition of a regular grammar says that it is a grammar where the rules are restricted to the form $A \rightarrow aB$, and $A \rightarrow \epsilon$, where A and B are non-terminal categories and a is a terminal symbol. This restriction results in grammars where the only constituents are suffixes. For instance, given an alphabet of heavy and light syllables $\Sigma = \{L, H\}$, suppose we defined a regular grammar that could derive the string $LLHLL$. Then the suffix-constituents derived by the grammar would pick out $\{L, LL, HLL, LHLL\}$; we would never be able to pick out the initial LL or the medial H as units, e.g. as feet: prosodic constituents in phonological theory are not just suffixes.

A regular language is a language that can be derived by a regular grammar. It has been shown that (almost) all phonological patterns are regular (Johnson, 1972; Kaplan and Kay,

1994). We are left with an apparent contradiction: if phonology has constituents that aren't just suffixes, then how is it that phonology is regular? The critical point is that it is the *language*—the set of admissible surface patterns—that has been shown to be regular in phonology, not the grammar that could derive it. There are infinitely many grammars that can define the same language, and non-regular grammars—with fewer restrictions on rewrite rules than regular ones—can define regular languages. That is, a language that can be defined by a grammar with suffix-constituents can also be defined by a grammar with non-suffix constituents. Our work in this paper assesses which grammar is, in a certain sense, better.

Let us call a class of non-regular grammars, such that every grammar in that class defines a regular string language, an *extended regular grammar*. `xfst` allows us to define extended regular grammars. It includes pre-defined complex operators which allow us to write grammars at a very high-level. For example, we can write SPE-style rules using replacement rules with the syntax `A -> B || L _ R`, where we simply need to specify the focus, change, and the structural description: this rule clearly does not meet the form $A \rightarrow aB$. `xfst` allows us to define our own operators and units, e.g. feet, too, and it does the hard work of compiling our high-level grammars to machine-level finite state transducers. This is advantageous because it can be easy for us to express generalizations in the high-level language which are hard to express or detect at the level of a regular grammar or a finite state machine.

By defining our grammars with `xfst`, we have a proof by construction that our grammars define regular languages, cf. Gainor et al. (2012). We also have a common formalism in which we can define all four grammars. We could define GEN and the assignment of violation marks in standard OT[3] using `xfst`, but the number of states required for EVAL in standard OT cannot be bounded. Using Karttunen (1998)'s formulation of OT, which maps directly from underlying to surface forms, we can avoid computing the non-regular EVAL relation.

1.1.2 Succinctness

Having `xfst` as a common formalism for defining all four grammars also allows us to make a controlled comparison of the succinctness of the grammars. We define the succinctness of a grammar as the size of the grammar—the number of symbols it takes to write it down (in `xfst`), under the conventions specified in (2.1). Our metric for succinctness is a special case of minimum description length (MDL) (Rissanen, 1989), which is also the metric used in the recent work of Rasin and Katzir (To appear). MDL as a metric for succinctness balances the minimization of the size of the grammar, which favors simple grammars that often overgenerate, with minimization of the size of the data encoded by the grammar, which favors restrictive but often overly memorized grammars.

For our comparisons, the data is the same across the comparisons (the set of stress patterns in words elicited from linguistic consultants, plus some predicted ones up to 5 syllables), as is the alphabet over which the grammars are defined (primary, secondary, and unstressed light and heavy syllables). Moreover, as we show in §3, all the grammars admit exactly the same set of stress patterns up to 5 syllables (with one exception). Thus, the MDL metric reduces to the size of the grammar. That is, the size of their encodings of the sequences up to 5 syllables is exactly the same, since the possibilities allowed by the grammars in that range is identical (cf. e.g. Rasin and Katzir (To appear)). We accordingly consider just the size of the four grammars, all expressed in the common `xfst` formalism.

[3]We use the term "standard OT" only to draw the comparison between OT that maps underlying forms to violation vectors and Karttunen (1998)'s formalization which does not.

1.2 Description of the language Little Samoan

Samoan stress presents a good case study for a first comparison of the succinctness of grammars with and without feet. A recent detailed foot-based OT analysis of Samoan stress based on a rich set of data is available (Zuraw et al., 2014), and Samoan stress patterns are for the most part not typologically exotic. Zuraw et al. (2014)'s analysis also extends to morphologically complex words parsed into multiple prosodic words, and in future work, we plan to pursue comparisons of grammars with higher-level prosodic constituents than feet.

We define Little Samoan (henceforth, LSmo), a language of strings of syllables marked for stress and weight, as a simplified version of the description of Samoan stress in monomorphs provided in Zuraw et al. (2014). Unlike that description, LSmo is defined over light and heavy syllables rather than segments, and thus ignores both complications from diphthongization as well as the interaction of stress with epenthesis. In Zuraw et al. (2014)'s description, Samon outputs LL for HL-final words because of avoidance of heavy-light (HL) feet, as well as for L, due to minimal word constraints. Since our grammars don't change the weights of the input, we model this in OT by mapping LL and HL-final inputs to a special `Null` symbol denoting a null output (Prince and Smolensky, 2004). In our direct models which accept only a subset of input strings, we ban them.

The basic primary stress pattern in LSmo is the same as in Samoan: moraic trochees at the right edge, exemplified in (1.1), adapted from Zuraw et al. (2014, (4)): exactly like Fijian (Hayes, 1995, §6.1.5.1, p. 142), "if the final syllable is light, main stress falls on the penult; if the final syllable is heavy, main stress falls on the final syllable".

(1.1) Basic primary stress pattern: moraic trochee at the right edge

. . .ˈH#	la(ˈvaː)	'energised'
. . .ˈLL#	(ˈmanu)	'bird'
. . .ˈLLL#	i(ˈŋoa)	'name'

Secondary stress in LSmo is almost like in Fijian, where "secondary stress falls on the remaining [non-final] heavy syllables, and on every other light syllable before another stress, counting from right to left" (Hayes, 1995, §6.1.5.1, p. 142), e.g. (ˌmaː)(ˌloː)(ˈloː) 'rest' (Zuraw et al., 2014, (7)). However, LSmo has an initial dactyl effect: initial LLL sequences are stressed as SWW.[4] The difference between LSmo and Fijian can be observed in (1.2).

(1.2) Secondary stress in LSmo vs. Fijian (Zuraw et al., 2014, (8)), (Hayes, 1995, p. 143–4, (44a, 47))

String	LSmo	Fijian	Gloss
LLLH	(ˈmini)si(ˈtaː)	mi(ˌnisi)(ˈtaː)	'minister'
LLLLL	(ˈtemo)ka(ˈlasi)	pe(ˌresi)(ˈtend͡i)	'democracy', 'president'

Zuraw et al. (2014) provides data on only two monomorphs that are longer than five moras: [(ˌmaː)(ˌloː)(ˈloː)] and a seven-mora all light loanword for Afghanistan. As noted in Zuraw et al. (2014, p. 281,fn. 2) the consultant produced (ˌʔafa)(ˌkani)si(ˈtana), while a true initial dactyl pattern would yield [(ˌʔafa)ka(ˌnisi)(ˈtana)]. Little data is available for longer words in Fijian, too, and there is variability [te(ˌreni)(ˈsisi)(taː)] 'transistor' vs. [(ˌkeː)(ˌmisi)ti(ˈriː)] 'chemistry',

[4]This ignores Zuraw et al. (2014)'s evidence from LLLLL loan words showing that an initial WSW pattern can occur if the first vowel in the word is epenthetic.

which may be due to faithfulness to stress in the source word and a dispreference for stressing epenthetic vowels, as in Samoan (Hayes, 1995, p. 144, (44b, 47)). Due to the lack of data on longer words, we limit testing empirical coverage of LSmo to monomorphs of 5 syllables (although of course our grammars can accept input strings of arbitrary length). In addition, we allow a more general distribution of dactyls, which permits both initial or medial dactyls. This allows both patterns for Afghanistan: (ˌLL)(ˌLL)L(ˈLL) and (ˌLL)L(ˌLL)(ˈLL), and predicts, for example, that HLLLH may be (ˌH)(ˌLL)L(ˈH) or (ˌH)L(ˌLL)(ˈH).

2 Four grammars for Little Samoan

In this section, we describe our grammars for LSmo: the direct account with feet (§2.1), the direct account with syllables only (§2.2), the OT account with feet (§2.3), and the OT account with syllables (§2.4). These descriptions include the definition of the transductions in the grammar as `xfst` commands, with numbers in square brackets at the right hand side of a command giving the number of symbols it took to write down the command. All of the `xfst` commands we use are definitions, which store a regular expression in a variable. These have the syntax `define` *variable-name regular-expression*. Our conventions for writing `xfst` expressions and counting symbols are given in (2.1).

(2.1) Conventions for xfst expressions and symbol counting

a. In a `define` command, regular expressions are always delimited by square brackets

b. Auxiliary terms are defined for any regular expression that appears more than once in the set of transduction rules

c. A conjunct or disjunct which is longer than one symbol is enclosed in brackets

d. Semicolons ending a line counts as a symbol; spaces do not count

e. A character enclosed in double quotes, e.g. `"("` counts as one symbol, a number counts as one symbol

f. Each variable name, command, operator, and atomic expression counts as a single symbol: `define`, `Heavy`, `WeakLight`, etc., `?`, `*`, `+`, `^`, `.#.`, `[`, `]`, `(`, `)`, `|`, `\&`, `~`, `\`, `$`, `->`, `,`, `=>`, `->@ _`, `...`, `.o.`, `.O.` are the symbols that appear in our transducers.

g. Symbols that are used in the specification of the input (GEN) do not contribute to the symbol count; transductions that introduce foot structure do.

2.1 Direct account with feet

Let us call the function that generates all possible sequences of syllables marked with degree of stress and weight GEN. We define GEN as a composition of three transducers (4): `Input` (1), `SWParse` (2), and `ElevateProm` (3). We define the `Input` to GEN as Σ^* over the alphabet of light (L) and heavy (H) syllables, $\Sigma = \{L, H\}$. Then, `SWParse` marks the degree of stress on a syllable by inserting labeled brackets around each syllable,[5] e.g. the input element *L* has the output {`S[L]` (strong/stressed), `W[L]` (weak/unstressed)}, much like Karttunen (1998)'s syllable parser `Parse` (see also (Beesley and Karttunen, 2003, p. 68)).

[5]We assume that syllable splitting feet do not occur Hayes (1995, §5.6.2, p. 121).

Finally, `ElevateProm` optionally replaces any strong syllable `S[ ]` with a primary stressed syllable, `P[ ]`.

```
                                define Input [ "L" | "H" ]*; [9]    (1)
    define SWParse [ ? -> [ "S" "[" | "W" "[" ] ... "]" ]; [15]    (2)
                      define ElevateProm [ "S" (->) "P" ]; [10]    (3)
         define Gen [ Input .o. SWParse .o. ElevateProm ]; [10]    (4)
```

As an example, `Gen` assigns the input `LH` to the set of outputs `P[H]P[L]`, `P[H]W[L]`, `P[H]S[L]`, `W[H]P[L]`, `W[H]W[L]`, `W[H]S[L]`, `S[H]P[L]`, `S[H]W[L]`, and `S[H]S[L]`.

Next, `ParseFoot` (7) parses the output from `Gen` into feet. We restrict a foot to being binary: either a LL or a H, so `ParseFoot` wraps parentheses pairs around any `LL` or `H`, regardless of the stress pattern. For readability, we define two auxiliary terms `Light` (5) and `Heavy` (6) and refer to these in the definition of `ParseFoot`.

```
                              define Heavy [ "[" "H" "]" ]; [8]    (5)
                              define Light [ "[" "L" "]" ]; [8]    (6)
  define ParseFoot [ [[ ? Light ]^2 | [ ? Heavy ]]  -> "(" ... ")" ]; [22]    (7)
```

We then define types of feet in preparation for use in restricting stress patterns to those in LSmo in terms of feet.

```
          define Foot ["(" \\["(" | ")"]]*  ")"]; [16]        (8)
           define PrimaryFoot [ Foot & $["P"] ]; [11]         (9)
                  define WeakLight [ "W" Light ]; [7]        (10)
```

`Foot` defines a foot as string of non-parentheses enclosed in parentheses (8); `PrimaryFoot` defines a primary stressed foot as a string accepted by `Foot` that includes `P` (9), and `WeakLight` defines a weak light syllable (10) using the auxiliary term `Light`.

We define trochaic feet with `Trochee` (13), which accepts a strong-weak LL sequence `LLFoot` (11), or a strong H `HFoot` (12). The sequence `\"W"` indicates the negation of the character `W`, i.e. any character but `W` such as `P` or `S`, which mark strong syllables.

```
    define LLFoot [ "(" \"W" Light WeakLight ")" ]; [11]     (11)
              define HFoot [ "(" \"W" Heavy ")" ]; [10]      (12)
                 define Trochee [ LLFoot | HFoot ]; [8]      (13)
```

With parsing the input into feet and definitions of types of feet behind us, the payoff comes as we can express LSmo's restrictions on stress patterns in terms of feet. `{TrocheesOnly` (14) forces feet in LSmo to be trochaic: it only accepts strings that are sequences of trochees that may be interspersed with unparsed syllables (weak lights), e.g. it winnows down the set of outputs `Gen` assigns to `LH` to just `(S[H]W[L])` and `(P[H]W[L])`. Additionally, `PrimaryFootRight` accepts only strings which terminate in a foot bearing primary stress and whose final foot is not preceded by any other primary stresses (15), e.g. eliminating

`(S[H]W[L])`. Note that this transduction eliminates lone Ls and HL-final strings in the language. Finally, we implement the "initial dactyl effect" in LSmo with `Initial Dactyl` (16), which forbids a sequence of a weak light followed by a LL foot at the beginning of the word, if the LLL sequence is non-final, i.e. followed by at least one character (`?+`). This yields the SWW-initial output `(S[L]W[L])W[L](P[H])` for LLLH sequences ([(ˌmini)si(ˈtaː], (Zuraw et al., 2014, (8))), but a WSW pattern `W[L](P[L]W[L)]` for LLL sequences [i(ˈŋoa)], (Zuraw et al., 2014, (4)). This also allows for multiple inputs for HLLLH and 7Ls, with medial dactyls.

```
define TrocheesOnly [ Trochee | WeakLight ]*; [9]          (14)
define PrimaryFootRight [ \"P"* PrimaryFoot ]; [9]         (15)
define InitialDactyl ~[ WeakLight LLFoot ?+ ]; [10]        (16)
```

The final transduction going from all possible sequences of stressed and weighted syllables to only those in LSmo composes the foot parser with the restrictions on words in terms of feet (17)[6]:

```
define LSmoDirFt [ ParseFeet .o. TrocheesOnly
          .o. PrimaryFootRight .o. InitialDactyl ]; [12]   (17)
```

All together, excluding the definition of GEN, the definition of the grammar costs 141 symbols.

2.2 Direct account referring to syllables only

The definition of GEN in this account is the same as in the previous account: the composition (4) of `Input` (1), `SWParse` (2), and `ElevateProm` (3). We state restrictions on words in LSmo by restricting stress patterns in terms of light and heavy syllables directly, rather than over feet. For expressing these restrictions, we define the auxiliary terms: `Heavy` (6), `Light` (5), `WeakLight` (10), as before, as well as `PrimaryLight` (18), a primary stressed light syllable , `SecondaryLight` (19), a secondary stress light syllable, `StressedSyll` (20), a syllable of any weight that is not weak, and `W2`, a sequence of two weak lights (a lapse).

```
define PrimaryLight ["P" Light]; [7]                       (18)
define SecondaryLight ["S" Light]; [7]                     (19)
define StressedSyll [ \"W" [ Heavy | Light ] ]; [12]       (20)
define W2 [ WeakLight WeakLight ]; [7]                     (21)
```

`StressHeavy` (22) requires that heavy syllables receive primary stress if word-final and otherwise, secondary stress. It uses the restriction operator => (Beesley and Karttunen, 2003, §2.4.1, p. 64), where `[A => L _ R]` expresses that the substring `A` must appear in the context `L _ R`, where `L` and `R` are regular expressions;[7] we can also define a list of allowed

[6] `LSmoDirFt` can also be composed with a transduction that replaces `W` in unparsed syllables with `X`.

[7] `[A => L _ ]` allows `A` to be followed by any string.

contexts separated by commas. `StressHeavy` restricts the substring `Heavy` to either be preceded by `P` and string-final (the end of the string is indicated by the boundary symbol `.#.`), or preceded by `S`.

`StressPLight` (23) states that if the final syllable is light, main stress falls on the penult. It restricts a `PrimaryLight` to be followed by a string-final `WeakLight`. Note that together, `StressHeavy` and `StressPLight` have the effect of enforcing CULMINATIVITY: that every word must have exactly one primary stress (Liberman and Prince, 1977, p. 262; Hayes, 1995, p. 24).

```
define StressHeavy [ Heavy => "P" _ .#., "S"  _ ]; [13]
```
(22)

```
define StressPLight [ PrimaryLight =>  _ WeakLight .#. ]; [10]
```
(23)

Restrictions on the distribution of secondary and weak lights are more complex and are expressed as a series of cases. `StressSLight` (24) restricts a secondary light to be followed by a non-final weak light (so a penult light cannot receive secondary stress). In addition, a secondary light must be string-initial, preceded by a lapse `[W[L]W[L]`, or preceded by a `S[L]W[L]`, i.e. in terms of feet, start a new foot.

```
define StressSLight [ SecondaryLight =>
        [.#. | W2 | [StressedSyll WeakLight] | Heavy] _ WeakLight ? ]; [19]
```
(24)

Note that we must further restrict the position of weak lights, because the transducer that is the composition of (22), (23), and (24) admits strings with lapses anywhere, e.g. it accepts both `P[L]W[L]` and `W[L]W[L]` from the set of sequences generated from input `LL`.

`RestrictLapse` (25) restricts a lapse `W[L]W[L]` to be preceded by a secondary light and followed by a stressed syllable, allowing a lapse just in case it is in a dactyl `S[L]W[L]W[L]` and not string-final.

```
define RestrictLapse [ W2 => SecondaryLight  _ StressedSyll ]; [10]
```
(25)

The intersection of `StressPLight`, `StressSLight` eliminates a lone stressed L, since `StressPLight` and `StressSLight` restrict stressed lights to being non-final. But `RestrictLapse` does not eliminate a lone weak L. Moreover, no transduction thus far eliminates HL-final sequences.[8] Thus, we must define transducers to ban HL-final sequences and lone Ls: `NoFinHL` (26) and `NoLoneL` (27).

```
define NoFinHL ~[?* ? Heavy ? Light]; [10]
```
(26)

```
define NoLoneL ~[Light] ; [7]
```
(27)

The final transduction is the composition defined by `LSmoDirSyll` :

[8] HL-final sequences are allowed in Heinz (2009)'s acceptor for Fijian stress (`http://phonology.cogsci.udel.edu/dbs/stress/language.php?id=109`), based on Hayes (1995) basic description of Fijian stress, but Hayes (1995, p. 145, §6.1.5.2)'s more detailed description reveals that they should not be accepted.

```
define LSmoDirSyll [ Gen .o.
 [ StressHeavy & StressPLight & StressSLight & RestrictLapse ]
                                .o. NoFinHL .o. NoLoneL ]; [20]   (28)
```

All together (including `Heavy` (6), `Light` (5), `WeakLight` (10), the transducers already defined in the previous section §2.1 ($8 + 8 + 7$)), the definition of the grammar costs 145 symbols.

2.3 Karttunen OT with feet

Our constraint set for this account is a subset of the constraints used in Zuraw et al. (2014); we have removed constraints that are only relevant for segments, morphologically complex words and multiple prosodic words. The remaining constraints are given in (2.2). The partial ranking was computed with OTSoft (Hayes et al., 2016) based on monomorphemic candidates used in Zuraw et al. (2014).

(2.2) Partial ranking of constraint set for Karttunen OT, using feet

I. Stratum 1

i. FOOTBINARITY (FOOTBIN) A foot must contain exactly two moras.

ii. RHYTHMTYPE=TROCHEE (RHTYPE=TROCHEE) A foot must have stress on its initial mora, and its initial mora only.

iii. ALIGN(PWD,R; ˈFT,R) (EDGEMOST-R) The end of the prosodic word must coincide with the end of a primary-stressed foot.

II. Stratum 2

i. PARSE-σ Every syllable must be included in a foot.

ii. ALIGN(PWD;L,FT,L) The beginning of the prosodic word must coincide with the beginning of a foot.

We assume that constraints referring to a prosodic word edge are computed with respect to the edge of the string provided by GEN. With the exception of EDGEMOST-R, our constraint definitions are identical to those in Zuraw et al. (2014, (5), (12)). Zuraw et al. (2014) computes ALIGN(PWD;L,FT,L) as a categorical constraint, e.g. 1 violation for [sika(ˈlamu)] 'scrum'. However, EDGEMOST-R is computed gradiently, e.g. 2 violations for [(piː)niki] and called EDGEMOST(ˈFt, R; Wd, R), with the same plain English description as in (2.2). We choose to define the constraint to instead be categorical. To be clear, we can paraphase the constraint definition as "assign a violation for every PWd where there exists a primary-stressed foot such that the right edge of the PWd and the right edge of the primary-stressed foot do not coincide" (McCarthy and Prince, 1993; McCarthy 2008, p. 214). Since the constraint is undominated in our constraint set, whether it is assessed categorically or gradiently has no effect on comparing candidates. As we will discuss in §3, whether a constraint is categorical vs. gradient strongly impacts the succinctness of the grammar in Karttunen's formalism, as does whether a constraint can be multiply or only singly violated (Karttunen, 1998).

The definition of GEN in this account is almost the same as in the previous ones, but we replace `SWParse` with `SWXParse` (29), which marks light and heavy syllables as strong, weak, or unparsed `X[ ]`.[9]

```
define SWXParse [ ? -> ["S" "[" | "W" "[" | "X" "[" ] ... "]" ]; [17]
```
(29)

In addition, we include `FtParse` (30) into the composition of GEN, `GenFt` (31). This parses the input into feet by wrapping parentheses around a non-empty sequence of syllables that are not unparsed. It refers to auxiliary terms for heavy and light syllables `Heavy` (6) and `Light` (5).

```
define FtParse [ [ \"X" [ Heavy | Light ] ]+ -> "(" ... ")" ]; [19]
```
(30)

```
define GenFt [ Input .o. SWXParse .o. ElevateProm .o. FtParse]; [12]
```
(31)

`FtParse` counts towards the symbol count since footing isn't part of the input in the unfooted accounts.

In addition to the constraints defined in 2.2, we define an additional undominated constraint `Culminativity` which states that every word must not have more than one primary stress and must contain a primary stress. We define this as a constraint rather than a property of GEN to keep GEN as close to constant across grammars.

```
define Culminativity ~[[$"P"]^>1] \& $["P"]; [18]
```
(32)

We also define `NullFinHL` and `NullLoneL` to map HL-final and L inputs to `Null`. These definitions allow for the syllables to be optionally footed and use directed replacement operators (Beesley and Karttunen, 2003, p. 73). For the `-@>` operator, replacement strings are selected right to left, and only the longest match is replaced.

```
define NullFinHL [  [?* Heavy [?]^<4 Light (")") ] ->@ "Null" ]; [27]
```
(33)

```
define NullLoneL [ [ ("(") ? Light (")") ] -> "Null" || .#.  _ .#. ]; [21]
```
(34)

We define `FtBin` (38) as a restriction on footed sequences to be any sequence of heavy feet, LL feet, and unparsed syllables, which are defined in `Unparsed` (35). `FtBinH` (36) defines a heavy foot as a heavy syllable of any degree of stress enclosed in parentheses, and `FtBinLL` (37) defines a LL foot as a sequence of two lights of any degree of stress enclosed in parentheses.

```
define Unparsed [ "X" "[" ? "]" ]; [8]
```
(35)

```
define FtBinH [ "(" ? Heavy ")" ]; [7]
```
(36)

```
define FtBinLL [ "("[ ? Light]^2 ")" ]; [13]
```
(37)

```
define FtBin [ FtBinH | FtBinLL | Unparsed ]*; [11]
```
(38)

[9] We could alternatively define another separate transduction that replaced unfooted `W[ ]` with `X[ ]` parsing into feet, which would possibly alter the symbol count slightly, but as we will see, the difference in symbol count between the OT syllable and OT foot account is so vast, that small differences like this are negligible in comparison.

We define RhyTypeTrochee (42) as the conjunction of RhyTypeTrocheeH (40) and RhyTypeTrocheeLL (41). RhyTypeTrocheeH restricts a heavy syllable followed by a parentheses, i.e. a footed H, to be preceded by a parentheses and S (secondary) or P (primary): a footed H must be stressed. RhyTypeTrocheeLL doesn't assume binary feet, and restricts a light syllable to either by foot-initial and stressed (defined with Stressed (39), or non-initial in a foot and weak, or unparsed. It accepts SWW dactyls.

```
define Stressed [ "S" | "P" ]; [8]                                          (39)
define RhTypeTrocheeH [ [ Heavy ")" ] => "("Stressed _ ]; [13]              (40)
define RhTypeTrocheeLL [ "[" "L" => "(" Stressed _ ,  "]" "W" _ , "X" _ ] ; [18]   (41)
define RhTypeTrochee [ RhTypeTrocheeH & RhTypeTrocheeLL ]; [8]              (42)
```

EdgemostR is identical to PrimaryFootRight (15), just renamed for the OT constraint, and refers to the auxiliary terms Foot (8) and PrimaryFoot (9)

ParseSyll must be implemented as a family of constraints because it can be multiply violated; we discuss this further in §3. Each ParseSyllN constraint in the family restricts the input string to have no more than N substrings containing X. We follow the implementation in Karttunen (1998, p. 10-11)[10]. We must impose some finite k-bound on the family; here we set $k = 5$ since the range of patterns we want to account for are only as long as five syllables.

```
define ParseSyll ~[$"X"] ; [8]                                              (43)
define ParseSyll1 ~[[$"X"]^>1] ; [13]                                       (44)
define ParseSyll2 ~[[$"X"]^>2] ; [13]                                       (45)
```

AlignWdLFtL (46) states that the beginning of the input string must coincide with beginning of foot and then may be followed by any string.

```
define AlignWdLFtL [ "(" ?* ]; [8]                                          (46)
```

The final transduction for the footed OT account is defined as LSmoMonoFtOT (47). The order of "lenient" composition is important: the higher ranked a constraint is, the earlier it must enter the composition. Within a stratum, order of composition doesn't matter. Karttunen (1998) defines "lenient composition" .O. to be a special form of composition where input strings are held back from being eliminated to keep the set of output candidates from becoming empty.

```
define LSmoMonoFtOT [ GenFt .O. NullFinHL .O. NullLoneL
       .O. Culminativity .O. FtBin .O. RhTypeTrochee
        .O. EdgemostR .O. ParseSyll .O. ParseSyll1
      .O. ParseSyll2 .O. ParseSyll3 .O. ParseSyll4
          .O. ParseSyll5.O. AlignWdLFtL ]; [28]
```

(47)

[10]But there is a typo, the definition in Karttunen (1998) is `~[[$"X["]^>N]; .`

In total the footed OT transduction costs 335 symbols, including 16 symbols from the definition of `Heavy` and `Light` (previously introduced) and 39 symbols from the definition of `EdgemostR` (previously introduced as `PrimaryFootRight`), and $13 \times 5 + 8 = 73$ for the `ParseSyll` family.

2.4 Karttunen OT with syllables only

The grammar using Karttunen OT referring to syllables only is by far the biggest grammar. This is because it not only includes constraints that can be multiply violated, but also gradient alignment constraints. There are few OT analyses of stress patterns that are not based on feet, i.e. "grid-based" (Bailey, 1995; Gordon, 2002, 2011), and we drew on constraints from them and other rhythmic analyses, but failed to generate only the allowed stress patterns up to 5 syllable words without introducing unorthodox constraints that referenced feet without naming them. Our constraint set includes the undominated constraints WEIGHTTOSTRESS, NONFINALITYL (nonfinality restricted to light syllables) (Hyde, 2007), and two constraints that we sheepishly invented: NOLAPSEFOLLOWINGHEAVY essentially enforces that a LL sequence after a heavy should be footed, and thus receive stress and allows the general *LAPSE constraint to be ranked lower. NOINITIALWS essentially bans iambic feet word-initially. To help achieve the initial dactyl effect, we used grid-based gradient ALIGN-x constraints drawn from the schema in Gordon (2002, (2)). ALIGN(x2,R,x0,PWd) enforces primary stress towards the right, while ALIGN(x1;L,x0,PWD) is necessary to promote SWW initial candidates.

The partial ranking was computed with OTSoft (Hayes et al., 2016) and given in (2.3).

(2.3) Partial ranking of constraint set for Karttunen OT, using syllables only

I. Stratum 1
   i. WEIGHTTOSTRESS (WSP) A heavy syllable must be stressed.
   ii. NONFINALITYL A word-final light syllable must be unstressed.
   iii. NOLAPSEFOLLOWINGHEAVY A heavy syllable must not be followed by two unstressed syllables.
   iv. NOINITIALWS A word must not begin with an unstressed-stressed sequence.

II. Stratum 2, 3, 4, 5
   i. ALIGN(x2,R,x0,PWd) Assign a violation for every grid mark of level 2 that does not coincide with the right edge of level 0 grid marks in a prosodic word.
   ii. *CLASH Assign a violation for every sequence of two stressed syllables.
   iii. *LAPSE Assign a violation for every sequence of two unstressed syllables.
   iv. ALIGN(x1;L,x0,PWd) Assign a violation for every grid mark of level 1 that does not coincide with the right edge of level 0 grid marks in a prosodic word.

While WSP and NOLAPSEFOLLOWINGHEAVY may have multiple loci of violation, because they are undominated, we were able to implement them as if they can only be singly violated. However, we had to implement a family of constraints to effectively count multiple violations for *CLASH, *LAPSE, ALIGN(x2,R,x0,PWd). Like in the footed OT analysis, we added rules to map HL-final and lone Ls to null output and CULMINATIVITY as an undominated constraint, which filtered out strings with multiple primary stresses. Thus, we were able

to implement ALIGN(x2,R,x0,PWd) similarly to the ParseSyll family. However, the implementation of ALIGN(X1;L,X0,PWD) required doing arithmetic because the same number of violations can be incurred by multiple stress patterns.

The definition of GEN (4) in this account is the same as for the direct accounts. The output of Gen is not submitted to the foot parser but is filtered by Culminativity (32). We define the NoClash family of constraints referencing the auxiliary term S2, a sequence of two stressed syllables, with StressedSyll defined in (20). NoNClash restricts the input from having N clashes. If an input is not accepted by NokClash, then it is also not accepted for any any $N > k$. The constraints define languages that are in a strict subset relation, like Karttunen (1998)'s ParseSyll family. We implement NoNClash constraints for any $N > 1$ as disjunctions, because there are multiple stress patterns that can result in the same number of clashes. For instance, an input may have two clashes because it has two nonadjacent clashes, or because it has a sequence of three stressed syllables, see No2Clash (50). The higher N is, the more disjuncts in the definition, and the number of symbols rapidly increases. We implemented up to $N = 4$. For No3Clash, we defined a disjunction of restrictions on strings containing a substring of 4 stressed syllables (SSSS, where S stands for stressed syllable), two substrings containing SSS sequences, a substring containing SSS followed by a substring containing SS, and a substring containing SS followed by a substring containing SSS. See the github code repository for definitions of No3Clash (61 symbols) and No4Clash (131 symbols).

```
define S2 [ StressedSyll StressedSyll ]; [7]
```
(48)

```
define No1Clash ~[$[S2]]; [10]
```
(49)

```
define No2Clash ~[$[StressedSyll]^3] & ~[[$[S2]]^2]; [25]
```
(50)

The definition of the NoNLapse family of constraints is identical in form, but defined over WW sequences, replacing S2 with W2 and StressedSyll with WeakSyll.

Since WSP is undominated, we implement it in (51) such that the final transduction does not differentiate between input strings with different numbers of unstressed heavy syllables.

```
define WSP [ Heavy  => \W _ ];  [10]
```
(51)

We define NonfinalityL (52) to accept strings where the final syllable cannot be a non-weak (i.e. stressed) light, NoLapseFollowingHeavy in (53), and NoInitWS in (54).

```
define NonfinalityL ~[?* \"W" Light]; [11]
```
(52)

```
define NoLapseFollowingHeavy ~[$[Heavy W2]];[11]
```
(53)

```
define NoInitWS ~["W" Light "S" Light ?*]; [12]
```
(54)

We do differentiate between inputs with primary stresses that are different distances from the right edge in the implementation of the Alignx2RN family. Each transducer Alignx2RN restricts primary stress to be at most N syllables, for any syllable (AnySyll (55)) from the

right edge. All Alignx2RN transducers for $N > 1$ have the same form as Alignx2R1, and the languages accepted by the transducers are in a strict subset relation.

```
define AnySyll [ ? "[" ? "]" ]; [9]                    (55)
define Alignx2R0 [ Primary => _ .#. ]; [9]             (56)
define Alignx2R1 [ Primary => _ [AnySyll]^<2 .#. ]; [15]   (57)
```

We implement ALIGN(x1;L,x0,PWd) in two parts. First we define the Alignx1LN family, a set of transducers where Alignx1LN restricts the input to have the sum of the distances that stressed syllables are away from the left edge to total N. For example, Alignx1L3 does not accept inputs that are in ?WWSW* (3 violations) or ?SSW* (1+2 violations), and Alignx1L5 does not accept inputs that are in ?WWWWSW* (5 violations), ?SWWSW* (1+4 violations), or ?WSSW* (2+3 violations), where W stands for a weak syllable, S for a stressed syllable, and ? for any character. The family is defined with the auxiliary term SWStar, which is the language of a stressed syllable followed by any number of unstressed syllables (58). The definitions for $N = 1, 2$ are given below.

```
define SWStar [StressedSyll [Weak]*];                  (58)
define Alignx1L1 ~[AnySyll SWStar];                     (59)
define Alignx1L2 ~[AnySyll Weak SWStar];                (60)
define Alignx1L3 ~[AnySyll W2 SWStar] & ~[AnySyll Stressed SWStar];   (61)
```

To convey a sense of how long these definitions get, here is our implementation of Alignx1L10 (62):

```
define Alignx1L10 ~[AnySyll Weak^9 SWStar] &            (62)
        ~[AnySyll Stressed Weak^7 SWStar] &             (63)
    ~[AnySyll Weak Stressed Weak^5 SWStar] &            (64)
      ~[AnySyll W2 Stressed Weak^3 SWStar] &            (65)
    ~[AnySyll Weak^3 Stressed Weak SWStar] &            (66)
            ~[AnySyll S2 Weak^4 SWStar] &               (67)
 ~[AnySyll Stressed Weak Stressed W2 SWStar]            (68)
          & ~[AnySyll Weak S2 Weak SWStar]              (69)
  & ~[AnySyll Stressed W2 Stressed SWStar];              (70)
```

We then take the intersections of the Alignx1LN languages to define languages Alignx1LgM that accept any number of violations less than M. For example Alignx1Lg5 is the intersection of Alignx1L5, Alignx1L6 , Alignx1L7, ..., Alignx1Lk, "don't have 5, 6, 7, ... k violations" where k is some finite upper bound. The Alignx1Lg5 language contains

`S[H]S[L]W[L]P[H]`, which has a total of $1+3 = 4$ violations, but not `S[H]W[L]S[L]P[H]`, which has a total of $2 + 3 = 5$ violations.

How high does k need to be? The output `S[H]W[L]S[L]S[H]P[H]` for HLLHH has $2 + 3 + 4 = 9$ violations in total, while the output `S[H]S[L]W[L]S[H]P[H]` has $1 + 3 + 4 = 8$ violations in total. With our constraint set, these two candidates have no other difference in their violation profiles. Thus, our transduction should admit the candidate with 8 violations, and not the one with 9 violations. The candidate with 8 violations should be in any `Alignx1LgM` language where $M > 8$, in particular, in the `Alignx1Lg9` language, while the candidate with 9 violations should not. However, if $k = 8$, and we define `Alignx1LgM` transducers up to $M = 7$, then `Alignx1Lg7` is the intersection of `Alignx1L7` and `Alignx1L8` "don't have 7 or 8 violations". Then in fact, the candidate with 9 violations is in the `Alignx1Lg7` language, while the candidate with 8 is not, and the transduction outputs the wrong candidate. To output the correct candidate, we must have $k >= 10$, with $M >= 9$: even for only up to 5-syllable words, we must have $k >= 10$. Minimally we must include `Alignx1L10` in the conjunction that defines `Alignx1Lg9`, as shown in (71):

```
define Alignx1Lg9 [Alignx1L9 & Alignx1L10];        (71)
```

In general, a n-syllable word can have a maximum of $\sum_{n=2}^{n-1} n$ violations of ALIGN(x1;L,x0,PWd) and k must be greater than that sum.

The final transduction is the lenient composition. For words only up to 5 syllables, `Alignx2RN` for $N > 4$ may be omitted.

```
define LSmoMonoSyllOT [ Gen .O. SNullFinHL .O.
          SNullLoneL  .O. WSP .O. NonfinalityL .O.
     NoLapseFollowingHeavy .O. NoInitWS .O. Alignx2R0 .O.
   Alignx2R1 .O. Alignx2R2 .O. Alignx2R3 .O. Alignx2R4 .O.
    Alignx2R5 .O. Alignx2R6 .O. Alignx2R7 .O. No1Clash .O.
      No2Clash .O. No3Clash .O. No4Clash .O. No1Lapse .O.
     No2Lapse .O. No3Lapse .O. No4Lapse .O. Alignx1Lg1 .O.
 Alignx1Lg2 .O. Alignx1Lg3 .O. Alignx1Lg4 .O. Alignx1Lg5 .O.
Alignx1Lg6 .O. Alignx1Lg7 .O. Alignx1Lg8.O. Alignx1Lg9 ];  (72)
```

It is obvious that the size of the grammar for the OT syllable account is much larger than that any of the other grammars and that the growth of the size of the grammar increases rapidly with the size of the input. Moreover, all of the constraints which can be multiply violated cannot be implemented as finite state transducers, and our implementations of these constraints require doing arithmetic and defining constraint families that have the effect of counting up violations. Finally, to even get near-coverage of the data without feet, we needed to define ad-hoc constraints that referenced feet without revealing generalizations in the structural restrictions on stress patterns.

3 Discussion

We examined the set of possible stress patterns from our grammars by composing our final transducers with an identity transducer that was defined as a disjunction of elicited/expected stress patterns for LSmo up to 5 syllables. We also defined an identity transducer for all possible light-heavy inputs up to 5 syllables and composed that with our final transducers. Then we checked that the set of strings defined by these two compositions was identical for each grammar and across grammars. Our four grammars for LSmo admit exactly the same set of stress patterns up to 5 syllables, with one exception: while all the other grammars admit two outputs for HLLLH: `S[H]S[L]W[L]W[L]P[H]` or `S[H]W[L]S[L]W[L]P[H]`, the OT syllable account admits only `S[H]W[L]S[L]W[L]P[H]`. Thus, the MDL metric reduces to the size of the grammar, although that's not quite the case for the OT syllable account. The direct accounts were almost the same in size: 145 symbols for the syllable account and 141 symbols for the footed account. However, the OT syllable account was much larger than the OT footed one, and was by far the largest grammar. Even if a more standard constraint set was found that could account for the data, it's likely that an OT syllable grammar would still need to include gradient ALIGN constraints.

Our results show that with a direct account, a grammar referencing feet in the description of stress patterns in LSmo is about as succinct as a grammar that does not. By this metric, one is not preferable to the other. For the OT accounts, a grammar referencing feet is sizeably more succinct than one that references only syllables and is thus preferable. While the two direct grammars do not differ noticeably in succinctness, once we have defined restrictions on feet, defining restrictions on stress in terms of feet allows us to state those generalizations about allowed stress patterns in three transducers, which refer to just three units (restricted feet) and weak lights. Structural generalizations are not as apparent in the grammar referencing syllables: for instance we must outright ban final HL and lone L sequences, while their ungrammaticality falls out of restrictions on feet and stress in the footed account, and stress patterns are defined in terms of weak lights, secondary lights, primary lights, stressed syllables, heavy syllables, and lapses—a larger set of units which are not clearly related to one another.

One thing to stress about all of the grammars, is that although they place boundaries (parentheses) in the string language, they are very different from SPE-style "boundary symbol theory" (Chomsky, 1965; Selkirk, 1980). As Selkirk (1980) points out, compared to a grammar which references nested units in the prosodic hierarchy, grammars with boundary symbols may be expressive enough to fit the data, but the lack of a well-defined relation between the different boundary symbols makes the grammars much too expressive. Moreover, boundary symbol theory places the boundary symbols in the alphabet and allows their placement to be restricted only by general restrictions on possible rewrite rules. The LSmo grammars do not place boundary symbols in the alphabet: these grammars place parentheses in the string language so we can refer to the units that they enclose. What restricts their placement is the phonological generalizations defined in the grammar.

Comparing the direct grammars to the OT grammars, a number of the transducers defined are identical or similar, e.g. `EdgemostR` appears in both footed accounts. This suggests that structural regularities we notice in phonological patterns can be well-described in both types of grammars. One advantage of the direct grammars is that we can define them with finite state transducers. Defining phonology with FSTs is not only helpful as a common formalism with comparison of syntax (and other patterns), but also enables us to compose phonological transducers with syntactic ones to model the syntax-phonology interface. In contrast, the expressive power of finite state transducers is not enough to define OT grammars where underlying forms

are mapped directly to surface forms rather than violation vectors. As noted by Karttunen (1998), any constraint that can be multiply violated such as PARSE-σ cannot be defined with a finite state transducer in Karttunen (1998)'s formalization of OT because a finite system cannot infinitely many degrees of well-formedness (Frank and Satta, 1998; McCarthy, 2003). If, instead we do define relations that map from underlying forms to violations as in standard OT, we can easily use `xfst` to implement PARSE-σ as `ParseSyllOT` (73). The FST is trivial: a 2-state machine, where one state maps any syllable that is not unparsed to `0` and the other maps an unparsed syllable `X[?]` to `1`, e.g. it maps `X[L]P[L]X[L]` to `101`. This shows an advantage to mapping to violations.

```
define ParseSyllOT [ "Unparsed" -> "1" , \"Unparsed" -> "0" ]; [11]
```
(73)

Some constraints, called categorical, assign multiple violations to a candidate iff there are multiple places where the constraint is violated in the candidate, e.g. PARSE-σ; or, they can be assigned by constraints, called gradient, that "measure the extent of a candidate's deviance from some norm", even if these is a single place in the string that is the locus of violation, e.g. ALIGN(PWD,R; 'FT,R) (McCarthy, 2003, p. 75). In Karttunen's formalization, both gradient and categorical constraints that can assign multiple violations cannot be defined with a FST. Moreover, as we saw with ALIGN(x1;L,x0,PWd), the implementation of gradient constraints is even more cumbersome. Not only are the machines that `xfst` compiles them into much too big and redundant to discover generalizations in; even the high-level language description are, too. When OT transduces instead to violation marks, only gradient constraints cannot be defined with a FST. While McCarthy (2003) argues that OT constraints are categorical, even if that is the case, EVAL is not a finite state process. OT is not regular if the number of violations is unbounded (Frank and Satta, 1998).

4 Conclusion

In this paper, we computationally implemented and compared grammars of Samoan stress patterns that refer to feet or only to syllables. We made this comparison in Karttunen's finite state formalization of OT, and in grammars directly describing restrictions of the surface patterns. In the OT formalism, having the prosodic constituents of feet clearly allows our grammar to be much more succinct. However, whether or not we have feet in the direct account does not impact succinctness of the grammar. It is interesting that direct finite-state descriptions of phonological patterns have revealed strong structural universals without referring to constituents, while the advantage of having constituents is clear in the OT formalism. The difference in the comparison between the two types of grammars may simply be because our measure of succinctness is not appropriate, and also may not hold in general. But even if direct accounts do no better in describing and revealing phonological generalizations with constituents, it does not follow that direct accounts should ignore phonological constituency: just because they can describe the patterns doesn't mean they are the correct description (Heinz, 2011a, p. 146) If phonological patterns are restricted in terms of constituents, then grammars with constituents are exactly the right ones we need to understand how phonological patterns are learned.

References

Bailey, Todd. 1995. Nonmetrical constraints on stress. Doctoral Dissertation, University of Minnesota, Minneapolis, Minnesota.

Beesley, Kenneth R., and Lauri Karttunen. 2003. Finite state morphology. Stanford, California: CSLI Publications.

Berwick, Robert C. 2015. Mind the gap. In 50 years later: Reflections on Chomsky's Aspects, ed. Á. Gallego and D. Ott, MITWPL77, 1–12. Cambridge, Massachusetts: MIT.

Bird, Steven, and T. Mark Ellison. 1994. One level phonology: autosegmental representations and rules as finite automata. Computational Linguistics 20:55–90.

Chomsky, Noam. 1956. Three descriptions of language. IRE Transactions in Information Theory 2:113–124.

Chomsky, Noam. 1965. Aspects of a theory of grammar. Cambridge, MA: MIT Press.

Chomsky, Noam, and Morris Halle. 1968. The sound pattern of English. The MIT Press.

Eisner, Jason. 1997. Efficient generation in Primitive Optimality Theory. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics.

Fougeron, Cécile, and Patricia A. Keating. 1997. Articulatory strengthening at edges of prosodic domains. Journal of Acoustical Society of America 101:3728–3740.

Frank, Robert, and Giorgio Satta. 1998. Optimality Theory and the generative complexity of constraint violability. Computational Linguistics 24:307–315.

Gainor, Brian, Regine Lai, and Jeffrey Heinz. 2012. Computational characterizations of vowel harmony patterns and pathologies. In Proceedings of the 29th West Coast Conference on Formal Linguistics, ed. Jaehoon Choi et al., 63–71. Somerville, MA: Cascadilla Proceedings Project.

Goldsmith, John A. 1990. Autosegmental and metrical phonology. Basil Blackwell.

Goldsmith, John Anton. 1976. Autosegmental phonology. Doctoral Dissertation, Massachusetts Institute of Technology.

Gordon, Matthew. 2002. A factorial typology of quantity insensitive stress. Natural Language & Linguistic Theory 20:491–552.

Gordon, Matthew. 2011. Stress sytems. In The handbook of phonological theory, ed. John A. Goldsmith, Jason Riggle, and Alan C. L. Yu, chapter 5, 141–163. Malden, Massachusetts: Blackwell Publishing Ltd, 2nd edition.

Hartmanis, Juris. 1980. On the succinctness of different representations of languages. SIAM Journal on Computing 9:114–120.

Hayes, Bruce. 1995. Metrical stress theory: principles and case studies. University of Chicago Press.

Hayes, Bruce, Bruce Tesar, and Kie Zuraw. 2016. Otsoft 2.4. Software package. URL http://www.linguistics.ucla.edu/people/hayes/otsoft/.

Heinz, Jeffrey. 2009. On the role of locality in learning stress patterns. Phonology 26:303–351.

Heinz, Jeffrey. 2010. Learning long-distance phonotactics. Linguistic Inquiry 41:623–661.

Heinz, Jeffrey. 2011a. Computational phonology – part i: Foundations. Language and Linguistics Compass 5:140–152. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1749-818X.2011.00269.x/abstract.

Heinz, Jeffrey. 2011b. Computational phonology – part II: grammars, learning, and the future. Language and Linguistics Compass 5:153–168. URL http://onlinelibrary.wiley.com/doi/10.1111/j.1749-818X.2011.00268.x/abstract.

Hulden, Mans. 2006. Finite state syllabification. In Finite-state methods and natural language processing, FSMNLP05, ed. Anssi Yli-Jyrä, Lauri Karttunen, and Juhani Karhumäki, 120–131. Berlin: Springer-Verlag.

Hyde, Brett. 2007. Non-finality and weight sensitivity. Phonology 24:287–334.

Johnson, C. Douglas. 1972. Formal aspects of phonological description. The Hague: Mouton.

Kaplan, Ronald M., and Martin Kay. 1994. Regular models of phonological rule systems. Computational Linguistics 20:331–378.

Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. In Proceedings of the International Workshop on Finite-State Methods in Natural Language Processing, FSMNLP'98.

Kornai, András. 1991. Formal phonology. Doctoral Dissertation, Stanford University, Stanford, California.

Liberman, Mark, and Alan Prince. 1977. On stress and linguistic rhythm. Linguistic Inquiry 8:249–336.

McCarthy, John J. 2003. OT constraints are categorical. Phonology 20:75–138. URL http://www.jstor.org/stable/4420242, ArticleType: research-article / Full publication date: 2003 / Copyright © 2003 Cambridge University Press.

McCarthy, John J. 2008. Doing optimality theory. Malden, Massachusetts: Blackwell Publishing.

McCarthy, John J., and Alan Prince. 1986/1996. Prosodic morphology. Technical report, Rutgers University Center for Cognitive Science, New Brunswick, NJ.

McCarthy, John J., and Alan S. Prince. 1993. Generalized alignment. Yearbook of Morphology 79–153.

Meyer, A.R., and M.J. Fischer. 1971. Economy of description by automata, grammars, and formal systems. In 12th Annual IEEE Symposium on Switching and Automata THeory, 188–191.

Nespor, Marina, and Irene Vogel. 1986. Prosodic phonology. Dordrecht, The Netherlands: Foris Publications.

Pierrehumbert, J.B. 1980. The phonology and phonetics of English intonation. Doctoral Dissertation, MIT.

Prince, Alan, and Paul Smolensky. 1993. Optimality theory: Constraint interaction in generative grammar. ROA version, 8/2002, Rutgers University Center for Cognitive Science.

Prince, Alan, and Paul Smolensky. 2004. Optimality theory: Constraint interaction in generative grammar. Malden, Massachusetts: Blackwell Publishing.

Rasin, Ezer, and Roni Katzir. To appear. On evaluation metrics in Optimality Theory. Linguistic Inquiry .

Rissanen, Jorma. 1989. Stochastic complexity in statistical inquiry theory. River Edge, New Jersey: World Scientific Publishing Co.

Salomaa, Arto. 1973. Formal languages. New York, New York: Academic Press.

Selkirk, Elisabeth. 1980. Prosodic domains in phonology: Sanskrit revisited. In Juncture, ed. M. Aronoff and M. L. Keans. Saratoga, California: Anma Libri.

Selkirk, Elisabeth O. 1986. Phonology and syntax: the relationship between sound and structure. Cambridge, MA: MIT Press.

Stabler, Edward P. 2013. Two models of minimalist, incremental syntactic analysis. Topics in Cognitive Science 5:611—633.

Wightman, Colin W., Stefanie Shattuck-Hufnagel, Mari Ostendorf, and Patti J. Price. 1992. Segmental durations in the vicinity of prosodic phrase boundaries. The Journal of the Acoustical Society of America 91:1707–1717.

Zuraw, Kie. 2009. Prosodic domains for segmental processes? evidence from some Austronesian languages. URL `https://www.mcgill.ca/linguistics/files/linguistics/Handout_RevisedForMcGill.pdf`, talk at Syntax Interfaces Research Group, McGill University, June 2009.

Zuraw, Kie, Kristine M. Yu, and Robyn Orfitelli. 2014. The word-level prosody of Samoan. Phonology 31:271–327.