Advantages of constituency: computational perspectives on Samoan word prosody*

Abstract. 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'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. Some speculations are offered about whether these results are robust and will hold as the scope of the formal theory is enlarged beyond the small fragment considered here.

Keywords: phonology, stress, finite state, optimality theory

1 Introduction

There is substantial evidence that referring to phonological constituents—units such as feet, prosodic words and higher-level constituents—helps us uncover and describe generalizations in phonological patterns. Referring to constituents can help us succinctly describe what part of a word gets reduplicated, restrictions on minimal words and allowed stress patterns, and the domain of segmental processes [13,29,35]. Constituents can also help us state systematic patterns about where certain tones are placed and how segmental durations and the "strength" of articulatory gestures vary [37]. However—and strikingly—computational descriptions of phonological patterns have revealed strong structural universals without referring to constituents [15,16]. Thus, although "we [phonologists] tend to help ourselves to prosodic domains without further comment" [38], the role of constituents in phonological grammars bears further scrutiny: the driving motivation for positing constituents—*Do constituents make phonological grammars more succinct?*—hasn't been carefully explored computationally.[1]

In this work, we implement and compare phonological grammars of stress patterns in Samoan monomorphs to assess whether 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 [6,28,12,36,3,31], among other work. We define all grammars in xfst [2], 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's finite state Optimality Theory (OT) [23], which maps underlying forms to surface forms without an intermediate mapping to violations, by using a special

* An earlier version of this paper has been submitted to MOL 2017.

[1] Some computational work has defined phonological patterns in terms of tiers [24,4,7] from autosegmental theory [10], but tiers aren't properly nested like constituents, e.g. it's generally assumed that feet don't straddle prosodic words.

composition operator "lenient composition", and (b) a "direct" approach which directly describes the surface patterns, e.g. [18,15]. Our work thus addresses not only the role of constituents in phonology, but also the advantages and disadvantages of different formalisms. The code implementing and testing the grammars is available at `https://github.com/fg2017-prepub/smo-mono-succinct`.

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 simplified language of stress patterns of 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). The discussion follows in §3, and we conclude with §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.

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 [5,33]. 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 determines the structural complexity of the grammar. A standard definition of a (right) regular grammar says that it is a grammar where the rules are restricted to the form $A \to aB$, and $A \to \epsilon$, where A and B are non-terminal categories and a is a terminal symbol. This restriction results in grammars where the only constituents of length > 1 are suffixes. But prosodic constituents in phonological theory include both prefixes and suffixes. For instance, given an alphabet of light and heavy syllables $\Sigma = \{L, H\}$, suppose we defined a (right) regular grammar that could derive the string $LLHLL$. Then the suffix-constituents derived by the grammar would never be able to pick out the initial LL or the medial H as feet.

A regular language is one that can be derived by a regular grammar. It has been shown that (almost) all phonological patterns are regular [20,22]. We are left with an apparent contradiction: if phonology has constituents that are both prefixes and suffixes, then how is it that phonology is regular? The critical point is that it's the *language*—the set of admissible surface patterns—that's 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; supra-regular grammars—with fewer restrictions on rewrite rules than regular ones—can define

regular languages. A language that can be defined by a grammar with suffix-constituents can also be defined by one with non-suffix constituents. This paper shows how we might assess which kind of 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, too, e.g. feet, and it compiles our high-level grammars to machine-level finite state transducers. The language was intentionally designed by linguists to make it 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. Since xfst grammars are compiled into standard finite state representations, an xfst definition establishes that the phenomenon described is regular (see [9]) and gives us a common formalism in which we can define all four grammars. It is not possible to define "standard OT" [30] fully in xfst: we could define Gen, which generates the set of candidate outputs, and the assignment of violation marks according to the set of violable constraints in Con—but not the non-regular Eval relation for computing the optimal candidate, since the number of states required to define Eval can't be bounded. However, we can define Karttunen's finite state formulation of OT [23] in xfst, as it avoids Eval by mapping underlying forms directly to surface forms with "lenient composition".

Succinctness Having xfst as a common formalism for defining all four grammars allows us to make a controlled comparison of the succinctness of the grammars. We define the succinctness of a grammar as its size—the number of symbols it takes to write it down (in xfst), under the conventions specified in §2. We define size over the high-level xfst grammar rather than at the machine level because it's the high-level language that we can express and detect generalizations in; the machines that xfst compiles are big and redundant by comparison.

Our metric for succinctness is a special case of minimum description length (MDL) [32]. 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. The alphabet over which the grammars are defined (primary, secondary, and unstressed light and heavy syllables) is constant across grammars.[2] Thus, there is no difference between measuring xfst grammar size with symbol counting and with bits. The data is also the same across the comparisons (the set of stress patterns in words elicited from linguistic consultants, plus some predicted ones up to 5 syllables). More-

[2] The foot-based accounts also introduce `(`, `)`, `X` , (`X` is an unparsed syllable), but: (i) it's not clear these should be included in the alphabet since they come in only in the the calculation of stress, (ii) if they are included, they make a negligible difference.

over, 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. We accordingly consider just the size of the four grammars, all expressed in the common xfst formalism.

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 [39], and the Samoan stress patterns described here are typologically representative. [39]'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 (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 [39]. LSmo is defined over light and heavy syllables rather than segments, and thus ignores complications from diphthongization and the interaction of stress with epenthesis. In [39]'s description, Samoan outputs LL for HL-final words to avoid heavy-light (HL) feet, and also presumably for L (content) words, to satisfy minimal word constraints. Since our OT grammars don't change the syllable weights in the input, we model this in OT by mapping LL and HL-final inputs to a special Null symbol denoting a null output [30]. Our direct models simply define transductions that don't accept HL and LL.

The basic primary stress pattern in LSmo (and Samoan) is moraic trochees at the right edge [39, (4)], e.g. la(ˈvaː) 'energized', (ˈmanu) 'bird', i(ˈŋoa) 'name'; exactly like better-known Fijian [13, §6.1.5.1], "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". 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" [13, §6.1.5.1, p. 142], e.g. (ˌmaː)(ˌloː)(ˈloː) 'rest' [39, (7)]. However, LSmo has an initial dactyl effect: initial LLL sequences are initially stressed, e.g. (ˈmini)si(ˈtaː) 'minister' (cf. Fijian mi(ˌnisi)(ˈtaː)), (ˈtemo)ka(ˈlasi) (cf. Fijian pe(ˌresi)(ˈtẽndi) 'president').[3]

[39] provides data on only two monomorphs that are longer than 5 moras: [(ˌmaː)(ˌloː)(ˈloː)] and a 7-mora all light loanword for Afghanistan. As noted in [39, 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', which may be due to faithfulness to stress in the source word and a dispreference for stressing epenthetic vowels, as

[3] This ignores [39]'s evidence from LLLLL loan words showing that an initial weak-strong-weak (WSW) pattern can occur if the first vowel in the word is epenthetic.

in Samoan [13, 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). Also, we allow a more general distribution of dactyls, which permits both initial or medial dactyls. This allows both (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). We define the transductions in the grammar in xfst, with symbol counts in square brackets to the right of the command. All of the xfst expressions we use are definitions, which get compiled into transducers associated with a variable. These have the syntax define *variable-name xfst-expression*. Our conventions for writing xfst expressions and counting symbols are as follows: (a) xfst expressions are delimited by square brackets in a define command. (b) Auxiliary terms are defined for any expression that appears more than once in the grammar. (c) A conjunct or disjunct longer than one symbol is enclosed in brackets. (d) Semicolons ending a line counts as a symbol; spaces do not count. (e) A number or a character enclosed in double quotes, e.g. "(", counts as one symbol. (f) Each variable name, command, operator, and atomic expression counts as a single symbol, i.e. define, Heavy, WeakLight, etc., ?, *, +, ^, .#., [,], (,), |, &, ~, \, $, ->, ,, =>, ->@ _, ..., .o., .O. (g) Symbols used to define Gen (constant across the grammars) don't contribute to the symbol count.

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 (4) as the composition of Input (1), SWParse (2), and ElevateProm (3). Input generates 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,[4] (see Parse in [23] and also [2, p. 68]), e.g. the input L has the output S[L] (strong/stressed), W[L] (weak/unstressed). Finally, ElevateProm optionally replaces any strong syllable S[ ] with a primary stressed syllable, P[ ], so that S[ ] now stands for secondary stress. As an example, input LL is mapped to {P[L]P[L], P[L]W[L], P[L]S[L], W[L]P[L], W[L]W[L], W[L]S[L], S[L]P[L], S[L]W[L], S[L]S[L]}.

Next, ParseFoot (6) parses the output from Gen into feet; it refers to auxiliary terms for heavy [H] and light [L] syllables, Heavy and Light (5). We restrict a foot to being bimoraic: either a LL or a H, so ParseFoot wraps parentheses pairs around any LL or H, e.g. (P[L]W[L]), regardless of the stress pattern. We then define types of feet to express restrictions on stress patterns. Foot defines a

[4] We assume that syllable splitting feet do not occur [13, §5.6.2, p. 121].

foot as string of non-parentheses enclosed in parentheses (7); PrimaryFoot defines a primary stressed foot as a string accepted by Foot that includes P (8), and WeakLight defines a weak light syllable (9) using Light. We define trochaic feet with Trochee (12), which accepts a strong-weak LL sequence LLFoot (10), or a strong H HFoot (11). 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 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)
define Heavy [ "[" "H" "]" ]; define Light [ "[" "L" "]" ]; [16]          (5)
define ParseFoot [ [ [ ? Light ]^2 | [ ? Heavy ] ]  -> "(" ... ")" ]; [22] (6)
define Foot ["(" \\["(" | ")"]]*  ")"]; [16]                               (7)
define PrimaryFoot [ Foot & $["P"] ]; [11]                                (8)
define WeakLight [ "W" Light ]; [7]                                       (9)
define LLFoot [ "(" \"W" Light WeakLight ")" ]; [11]                      (10)
define HFoot [ "(" \"W" Heavy ")" ]; [10]                                 (11)
define Trochee [ LLFoot | HFoot ]; [8]                                    (12)
define TrocheesOnly [ Trochee | WeakLight ]*; [9]                         (13)
define PrimaryFootRight [ \"P"* PrimaryFoot ]; [9]                        (14)
define InitialDactyl ~[ WeakLight LLFoot ?+ ]; [10]                       (15)
define LSmoDirFt [ ParseFeet .o. TrocheesOnly
                   .o. PrimaryFootRight .o. InitialDactyl ]; [12]         (16)
```

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 (13) forces feet 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 parses for LL to just (S[L]W[L]) and (P[L]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 (14), e.g. eliminating (S[L]W[L]). Note that this transduction eliminates lone Ls and HL-final strings in the language, since they do not have parses with final primary stressed feet. Finally, we implement the "initial dactyl effect" in LSmo with InitialDactyl (15), 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], [39, (8)]), but a WSW pattern W[L](P[L]W[L)] for LLL sequences [i(ˈŋoa)], [39, (4)]. This also allows outputs for HLLLH and 7Ls with medial dactyls. 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 (16).[5] All together, excluding Gen, the grammar (5-16) costs 141 symbols.

2.2 Direct account referring to syllables only

The definition of Gen in this account is the same as in §2.1, but then we state restrictions on stress patterns in terms of syllables, rather than over feet. In addition to the previously defined auxiliary terms Heavy and Light (5), and WeakLight (9), we also define: PrimaryLight and SecondaryLight, a primary P[L] and secondary stressed light syllable S[L] (17); StressedSyll, a syllable of any weight that is not weak (18); and W2, a sequence of two weak lights W[L]W[L] (a lapse) (19).

```
define PrimaryLight ["P" Light]; define SecondaryLight ["S" Light]; [14]   (17)
define StressedSyll [ \"W" [ Heavy | Light ] ]; [12]                       (18)
define W2 [ WeakLight WeakLight ]; [7]                                     (19)
define StressHeavy [ Heavy => "P" _ .#., "S" _ ]; [13]                     (20)
define StressPLight [ PrimaryLight => _ WeakLight .#. ]; [10]              (21)
define StressSLight [SecondaryLight =>
[.#. | W2 | [StressedSyll WeakLight] | Heavy] _ WeakLight ? ]; [19]         (22)
define RestrictLapse [ W2 => SecondaryLight _ StressedSyll ]; [10]         (23)
define NoFinHL ~[?* ? Heavy ? Light]; [10] define NoLoneL ~[Light] ; [7]   (24)
define LSmoDirSyll [ Gen .o. [ StressHeavy & StressPLight &
StressSLight & RestrictLapse ] .o. NoFinHL .o. NoLoneL ]; [20]             (25)
```

StressHeavy (20) requires that a heavy syllable (Heavy) receive primary stress if word-final (preceded by P and string-final, as indicated by .#.) and otherwise, secondary stress (preceded by S). It uses the restriction operator => [2, §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.[6] StressPLight (21) states that a light syllable can bear primary stress only in the penult, followed by a final unstressed light. Together, StressHeavy and StressPLight enforce Culminativity: that every word must have exactly one primary stress [25,13].

Restrictions on the distribution of secondary and weak lights are more complex and are expressed as a series of cases. StressSLight (22) 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. We must further restrict the position of weak lights, because the transducer that is the intersection of (20), (21), and (22) admits strings with lapses

[5] LSmoDirFt can also be composed with a transduction that replaces W in unparsed syllables with X, to match notation for the OT footed account in §2.3.

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

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 (23) 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. 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.[7] Thus, we must define transducers to ban HL-final sequences and lone Ls: NoFinHL and NoLoneL (24). All together 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 [39]; we have removed constraints that are only relevant for segments, morphologically complex words and multiple prosodic words. The partial ranking was computed with OTSoft[8] [14] based on monomorphemic candidates used in [39] and is: *Stratum 1* (i) FootBinarity (FtBin) 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; 'Ft,R) (Edgemost-R) The end of the prosodic word must coincide with the end of a primary-stressed foot; *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. Constraints in a earlier stratum are ranked higher than those in a later one, but constraints within a stratum are not ranked with respect to one another.

We compute constraints referring to a prosodic word edge with respect to the edge of the input string since the input never contains more than a single prosodic word. With the exception of Edgemost-R, our constraint definitions are identical to those in [39, (5), (12)]. 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 (39). Other constraints, called gradient, "measure the extent of a candidate's deviance from some norm", and can assign multiple violations even if there is a single locus of violation in the input [26, p. 75]. [39] computes Align(Pwd;L,Ft,L) as a categorical constraint, e.g. 1 violation for *[sika('lamu)] 'scrum'. However, Edgemost-R in [39] is computed gradiently, e.g. 2 violations for *[('pi:)niki]. We define it instead as categorical: since it's undominated in our constraint set, whether it is assessed categorically or gradiently makes no difference. 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" [27]. As we'll discuss in §3, whether a constraint is categorical

[7] HL-final sequences are allowed in [15]'s acceptor for Fijian stress (`http://phonology.cogsci.udel.edu/dbs/stress/language.php?id=109`), based on [13] basic description of Fijian stress, but [13, p. 145, §6.1.5.2]'s more detailed description reveals that they should not be accepted.

[8] All OTSoft input and output files are in the github repository.

vs. gradient markedly impacts the succinctness of the grammar in Karttunen's formalism, as does whether a constraint can be multiply or only singly violated.

```
define MarkUnparsed [ "W" (->) "X" ]; [10]                                   (26)
define FtParse [ [ \"X" [ Heavy | Light ] ]+ -> "(" ... ")" ]; [19]          (27)
define GenFt [ Gen .o. MarkUnparsed .o. FtParse]; [10]                       (28)
define Culminativity [ $.P ]; [8]                                            (29)
define NullFinHL [  [?* Heavy [?]^<4 Light (")") ] ->@ "Null" ]; [27]        (30)
define NullLoneL [ [ ("(") ? Light (")") ] -> "Null" ]; [21]                 (31)
define Unparsed [ "X" "[" ? "]" ]; [8]                                       (32)
define FtBinH [ "(" ? Heavy ")" ]; [7] define FtBinLL [ "("[ ? Light]^2 ")" ]; [13]  (33)
define FtBin [ FtBinH | FtBinLL | Unparsed ]*; [11]                          (34)
define Stressed [ "S" | "P" ]; [8]                                           (35)
define RhTypeTrocheeH [ [ Heavy ")" ] => "("Stressed _ ]; [13]               (36)
define RhTypeTroLL [ "[" "L" => "(" Stressed _ ,  "]" "W" _ , "X" _ ] ; [18] (37)
define RhTypeTrochee [ RhTypeTrocheeH & RhTypeTrocheeLL ]; [8]               (38)
define ParseSyll ~[$"X"]; [8]                                                (39)
define ParseSyll1 ~[[$"X"]^>1]; [13] define ParseSyll2 ~[[$"X"]^>2]; [13]    (40)
define AlignWdLFtL [ "(" ?* ]; [8]                                           (41)
define LSmoMonoFtOT [ GenFt .O. NullFinHL .O. NullLoneL .O. Culminativity
.O. FtBin .O. RhTypeTrochee .O. EdgemostR .O. ParseS .O. ParseS1 .O. ParseS2
.O. ParseS3 .O. ParseS4 .O. ParseS5 .O. AlignWdLFtL ]; [28]                   (42)
```

We define the candidate set in the OT footed account with GenFt (28) as the composition of Gen (4), MarkUnparsed (26), and FtParse (27). MarkUnparsed optionally replaces W[ ] with X[ ] to mark syllables unparsed into feet. FtParse parses the input into feet by wrapping parentheses around a non-empty sequence of syllables that are not unparsed, e.g. X[L](P[L]W[L]). We define three undominated constraints not included in the OTSoft ranking: Culminativity (29), NullFinHL (30), and NullLoneL (31). We define Culminativity (every word must contain exactly one primary stress) as a constraint rather than a property of Gen to keep Gen constant across grammars. NullFinHL and NullLoneL map HL-final and L inputs to Null. These definitions allow the HL-final and L inputs to be footed or unfooted and use directed replacement operators [2, p. 73]. For the -@> operator, replacement strings are selected right to left, and only the longest match is replaced.

We then define constraints on feet. FtBin (34) restricts footed sequences to be any sequence of heavy feet and LL feet (FtBinH, FtBinLL, (33)), and unparsed syllables (Unparsed, (32)). We define RhyTypeTrochee (38) as the conjunction of RhyTypeTrocheeH (36) and RhyTypeTrocheeLL (37). 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 restricts a light syllable to be foot-initial and stressed (defined with Stressed (35)), or non-initial in a foot and weak, or unparsed. It accepts SWW dactyls. EdgemostR (not shown) is identical to PrimaryFootRight (14).

ParseSyll (39) 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, e.g. $N = 1$ in ParseSyll1 (40). We follow the implementation in [23, p. 10-11][9] 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. AlignWdLFtL (41) states that the beginning of the input string must coincide with the beginning of a foot and then may be followed by any string. The final transduction LSmoMonoFtOT is defined as a "lenient composition" (42).[10] [23] defines this (.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. 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. In total the OT footed grammar costs 306 symbols, including 73 from the ParseSyll family and 16 from Light and Heavy.

2.4 Karttunen OT with syllables only

The grammar using Karttunen OT with syllables only is by far the biggest grammar: it includes not only 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" [1,11,21], and we drew on constraints from them, but failed to generate only the allowed stress patterns up to 5 syllable words without introducing ad-hoc constraints that referenced feet without naming them. The partial ranking was computed with OTSoft [14] and is: *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 musn't be followed by two unstressed syllables. (iv) NoInitialWS A word musn't begin with an unstressed-stressed sequence. *Stratum 2.* Align(x2,R,x0,PWd) Assign a violation for every grid mark of level 2 that doesn't coincide with the right edge of level 0 grid marks in a prosodic word. *Stratum 3.* *Clash Assign a violation for every sequence of two stressed syllables. *Stratum 4.* *Lapse Assign a violation for every sequence of two unstressed syllables. *Stratum 5.* Align(x1;L,x0,PWd) Assign a violation for every grid mark of level 1 that doesn't coincide with the right edge of level 0 grid marks in a prosodic word.

Our constraint set includes the undominated constraints WeightToStress, NonfinalityL (nonfinality restricted to light syllables) (49) [19], and two ad-hoc constraints: (i) NoLapseFollowingHeavy (50) essentially enforces that a LL sequence after a heavy should be footed, and thus receive stress and allows the general

[9] But there's a typo in [23]; Parse is defined as ~$["X["]; but should be ~["$"X["];.

[10] We abbreviate ParseSyll as ParseS for space; see github repository for definitions of ParseSyllN for $N > 2$.

*Lapse constraint to be ranked lower, (ii) NoInitialWS (51) essentially bans iambic feet word-initially. To achieve the initial dactyl effect, we used grid-based gradient Align-x constraints drawn from the schema in [11, (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. While WSP (48) and NoLapseFollowingHeavy may have multiple loci of violation, because they are undominated, we were able to implement them as if they could only be singly violated. However, we had to implement a family of constraints to effectively count multiple violations for *Clash, *Lapse, and Align(x2,R,x0,PWd). We used Culminativity (29) to filter out strings with multiple primary stresses, so we could implement Align(x2,R,x0,PWd) similarly to the ParseSyll family. But the implementation of Align(x1;L,x0,PWd) required doing arithmetic because the same number of violations could be incurred by multiple stress patterns. We present a selection of the grammar below (see the github repository for the full grammar); Gen (4) is the same as before. Previously defined transductions repeated here (not shown) are: Culminativity (29), Heavy and Light (5), and W2 (19). We also defined Weak (43) and Stressed (44) syllables, and clash S2 (45), two adjacent stressed syllables. Transductions similar to NullFinHL and NullLoneL map HL-final and L inputs to Null (not shown).

```
define Weak "W" [ Heavy | Light ]; [9]                                   (43)
define Stressed ["S"|"P"] [ Heavy | Light ] ; [13]                       (44)
define S2 [ Stressed Stressed ]; [7]                                     (45)
define No1Clash ~[$[S2]]; [10]                                           (46)
define No2Clash ~[$[Stressed]^3] & ~[[$[S2]]^2]; [25]                    (47)
define WSP [ Heavy  => \W _ ]; [10]                                      (48)
define NonfinalityL ~[?* \"W" Light]; [11]                               (49)
define NoLapseFollowingHeavy ~[$[Heavy W2]]; [11]                        (50)
define NoInitWS ~["W" Light "S" Light ?*]; [12]                          (51)
define AnySyll [ ? "[" ? "]" ]; [9]                                      (52)
define Alignx2R0 [ Primary => _ .#. ]; [9]                               (53)
define Alignx2R1 [ Primary => _ [AnySyll]^<2 .#. ]; [15]                 (54)
define SWStar [Stressed [Weak]*]; [10]                                   (55)
define Alignx1L1 ~[AnySyll SWStar]; define Alignx1L2 ~[AnySyll Weak SWStar];   (56)
define Alignx1L3 ~[AnySyll W2 SWStar] & ~[AnySyll Stressed SWStar]; [16] (57)
```

The NoNClash family of constraints 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 [23]'s ParseSyll family. We implemented NoNClash constraints (for $N = 1, 2, 3, 4$) as conjunctions, 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 (47). The higher N is, the more conjuncts in the definition; specifically, it is 2^{N-1}. Already for No3Clash, we require 4 conjuncts: 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. The definition of NoNLapse is identical to that of NoNClash, but replaces S2 with W2 and Stressed with Weak.[11]

Similarly, we implement Align(x2,R,x0,PWd) as a categorical constraint, as the family Alignx2RN. Each transducer Alignx2RN restricts primary stress to be at most N syllables, for any syllable (AnySyll (52)) from the right edge. All Alignx2RN transducers for $N > 1$ have the same form as Alignx2R1 (54); the languages accepted by the transducers are in a strict subset relation, and for words only up to 5 syllables, Alignx2RN for $N > 4$ may be omitted. Align(x1;L,x0,PWd), though, must be implemented as a gradient constraint. We do it 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 (57) 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 definition refers to auxiliary term SWStar, the language of a stressed syllable followed by any number of unstressed syllables (55). Definitions for $N = 1, 2, 3$ are given in (56,57).

We then take 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 constraints, 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. 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 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 the candidate with 9 violations is in the Alignx1Lg7 language, while the candidate with 8 isn't: the transduction outputs the wrong candidate. To output the correct one, we must have $k \geq 10$, with $M \geq 9$: even for only up to 5-syllable words, we must have $k \geq 10$. Minimally we must include Alignx1L10 in the conjunction

[11] See the github code repository for definitions of No3Clash (61 symbols) and No4Clash (131 symbols) and the NoNLapse constraint family.

that defines Alignx1Lg9. 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 language derived by the OT syllable grammar has a small difference from the others: 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]. But [39] doesn't actually include elicited data for HLLLH, so we don't know which pattern(s) our consultants would accept. The striking difference about the OT syllable grammar is in its size. It is much larger than any of the other grammars, and the growth of the size of the grammar increases rapidly with the length of the input string: the definition of just the gradient constraint Align(x1;L,x0,PWd) takes more symbols than any of the other entire grammars, and the definitions of the clash and lapse constraints alone already grow exponentially 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 approximating these constraints require doing arithmetic and defining constraint families that have the effect of counting up violations to some finite bound. Finally, to just 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. And additional exploratory calculations from OTSoft showed that we still need gradient alignment constraints to fit the data, despite including additional categorical constraints from [21]'s Rhythmic Licensing Theory—designed to avoid gradient constraints (see github repository, `otsoft-files/syll/test-with-rlt`).

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 the one exception mentioned above: the OT syllable account admits only one stress pattern for HLLLH. 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; the OT footed grammar cost 306 symbols. The small differences between these is insignificant, compared to the qualitatively different character of exponential growth we saw in definition of the OT syllable grammar, with a count in the 1000s. Even if including a battery of constraints from Rhythmic Licensing Theory [21], we found that an OT syllable grammar would still need to include multiply violated clash and lapse constraints and gradient Align constraints; we'd

also expect this to be true in general beyond the Samoan case study here, such that the size of OT syllable phonological grammars would in general blow up.

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 isn't preferable to the other. Also, the size of the direct grammars is a few times smaller than even the OT footed grammar, so Karttunen OT grammars are certainly not preferable by succinctness. For the OT accounts, a grammar referencing feet is sizeably more succinct than one that references only syllables, showing the utility of feet. It's interesting that the direct footed account wasn't notably more succinct than the direct syllable one; this could be because of the narrowness of the scope of phonological phenomena considered here. For instance, patterns of stress shift in Samoan upon affixation can be generalized on the basis of constituents [39], but here we considered only monomorphs. The more phonological processes that reference constituents in the grammar, the more the savings from those constituents.

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" [6,34]. As [34] 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 [23], any constraint that can be multiply violated such as Parse-σ cannot be defined with a finite state transducer in [23]'s formalization of OT because a finite system cannot infinitely many degrees of well-formedness [8,26]. 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: define ParseSyllOT ["Unparsed" -> "1" , \"Unparsed" -> "0"]; [11]. 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. 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 [26] argues that OT constraints are categorical, even if that is the case, Eval isn't a finite state process. OT isn't regular if the number of violations is unbounded [8].

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 striking 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, or because the range of phonological phenomena considered here is too narrow. 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 [17, 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

1. Bailey, T.: Nonmetrical constraints on stress. Ph.D. thesis, U. Minnesota (1995)
2. Beesley, K.R., Karttunen, L.: Finite State Morphology. CSLI, Stanford, CA (2003)
3. Berwick, R.C.: Mind the gap. In: Gallego, A., Ott, D. (eds.) 50 Years Later, pp. 1–12. MITWPL77, MIT, Cambridge, Massachusetts (2015)
4. Bird, S., Ellison, T.M.: One level phonology: autosegmental representations and rules as finite automata. Computational Linguistics 20, 55–90 (1994)
5. Chomsky, N.: Three descriptions of language. IRE Transactions in Information Theory 2(3), 113–124 (1956)
6. Chomsky, N., Halle, M.: The sound pattern of English. The MIT Press (1968)
7. Eisner, J.: Efficient generation in Primitive Optimality Theory. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (1997)

8. Frank, R., Satta, G.: Optimality Theory and the generative complexity of constraint violability. Computational Linguistics 24(2), 307–315 (1998)
9. Gainor, B., Lai, R., Heinz, J.: Computational characterizations of vowel harmony patterns and pathologies. In: Choi, J., et al. (eds.) WCCFL 29. pp. 63–71. Cascadilla Proceedings Project, Somerville, MA (2012)
10. Goldsmith, J.A.: Autosegmental phonology. Ph.D. thesis, MIT (1976)
11. Gordon, M.: A factorial typology of quantity insensitive stress. NLLT 20, 491–552 (2002)
12. Hartmanis, J.: On the succinctness of different representations of languages. SIAM Journal on Computing 9, 114–120 (1980)
13. Hayes, B.: Metrical stress theory. U. Chicago Press (1995)
14. Hayes, B., Tesar, B., Zuraw, K.: Otsoft 2.4. Software package (2016), `http://www.linguistics.ucla.edu/people/hayes/otsoft/`
15. Heinz, J.: On the role of locality in learning stress patterns. Phonology 26(02), 303–351 (2009)
16. Heinz, J.: Learning long-distance phonotactics. LI 41(4), 623–661 (2010)
17. Heinz, J.: Computational phonology – part I: foundations. Language and Linguistics Compass 5(4), 140–152 (2011)
18. Hulden, M.: Finite state syllabification. In: Yli-Jyrä, A., Karttunen, L., Karhumäki, J. (eds.) FSMNLP05, pp. 120–131. Springer-Verlag, Berlin (2006)
19. Hyde, B.: Non-finality and weight sensitivity. Phonology 24(2), 287–334 (2007)
20. Johnson, C.D.: Formal aspects of phonological description. Mouton (1972)
21. Kager, R.: Rhythmic licensing theory: an extended typology. In: Proceedings of the 3rd Seoul International Conference on Phonology. pp. 5–31 (2005)
22. Kaplan, R.M., Kay, M.: Regular models of phonological rule systems. Computational Linguistics 20(3), 331–378 (1994)
23. Karttunen, L.: The proper treatment of optimality in computational phonology. In: FSMNLP'98 (1998)
24. Kornai, A.: Formal phonology. Ph.D. thesis, Stanford University (1991)
25. Liberman, M., Prince, A.: On stress and linguistic rhythm. LI 8(2), 249–336 (1977)
26. McCarthy, J.J.: OT constraints are categorical. Phonology 20(1), 75–138 (2003)
27. McCarthy, J.J., Prince, A.S.: Generalized alignment. Morph. pp. 79–153 (1993)
28. Meyer, A., Fischer, M.: Economy of description by automata, grammars, and formal systems. In: SWAT 1971. pp. 188–191 (1971)
29. Nespor, M., Vogel, I.: Prosodic phonology. Foris Publications, Dordrecht (1986)
30. Prince, A., Smolensky, P.: Optimality Theory: Constraint interaction in generative grammar. Blackwell Publishing, Malden, Massachusetts (2004)
31. Rasin, E., Katzir, R.: On evaluation metrics in Optimality Theory. LI (To appear)
32. Rissanen, J.: Stochastic complexity in statistical inquiry theory (1989)
33. Salomaa, A.: Formal languages. Academic Press, New York, New York (1973)
34. Selkirk, E.: Prosodic domains in phonology: Sanskrit revisited. In: Aronoff, M., Keans, M.L. (eds.) Juncture. Anma Libri, Saratoga, California (1980)
35. Selkirk, E.O.: Phonology and syntax. MIT Press, Cambridge, MA (1986)
36. Stabler, E.P.: Two models of minimalist, incremental syntactic analysis. Topics in Cognitive Science 5(3), 611—633 (2013)
37. Wagner, M., Watson, D.G.: Experimental and theoretical advances in prosody: a review. Language and Cognitive Processes 25, 905–945 (2010)
38. Zuraw, K.: Prosodic domains for segmental processes? (June 2009), `https://www.mcgill.ca/linguistics/files/linguistics/Handout_RevisedForMcGill.pdf`
39. Zuraw, K., Yu, K.M., Orfitelli, R.: The word-level prosody of Samoan. Phonology 31(2), 271–327 (2014)