Proceedings of the Society for Computation in Linguistics

Volume 2 Article 26

2019

A Logical and Computational Methodology for Exploring Systems of Phonotactic Constraints

Dakotah Lambert
Earlham College, djlambert11@earlham.edu

James Rogers
Earlham College, jrogers@cs.earlham.edu

A logical and computational methodology for exploring systems of phonotactic constraints

Dakotah Lambert
Earlham College
Richmond, Indiana, USA
djlambe11@earlham.edu

James Rogers
Earlham College
Richmond, Indiana, USA
jrogers@cs.earlham.edu

Abstract

We introduce a methodology built around a logical analysis component based on a hierarchy of classes of Subregular constraints characterized by the kinds of features of a string a mechanism must be sensitive to in order to determine whether it satisfies the constraint, and a computational component built around a publicly-available interactive workbench that implements, based on the equivalence between logical formulae and finite-state automata, a theorem prover for these logics (even algorithmically extracting certain classes of constraints), wherein the alternation between these logical and computational analyses can provide useful insight more easily than using either in isolation.

We demonstrate this methodology by exploring a series of examples drawn from the StressTyp2 database (Goedemans et al., 2015) of patterns of lexical (suprasegmental) stress. Along the way we provide justification for the underlying model-theoretic approach to formalizing phonotactic patterns and demonstrate the ways in which this dual methodology can be applied to a range of phonological issues.

1 Introduction

In this paper, we introduce a set of logical (model-theoretic) and computational (automata-theoretic) tools along with a methodology for exploring systems of constraints on strings that combines them. We have incorporated these into a computational workbench, which we use to demonstrate the application of this methodology to the study of lexical (suprasegmental) stress patterns obtained from StressTyp2 (Goedemans et al., 2015).

Although the formal tools we employ here are tailored to phonotactics, the methodology we are using — alternating between the logical and computational tools in exploring the data — is applicable, with suitably modified tools, to other aspects of phonology, syntax and semantics, as well as many non-linguistic applications.

In our specific application, the phenomena we must account for are phonotactic stress patterns of human languages. Our facts are descriptions of these patterns drawn from the literature, and our methods are intended to support identification and generalization of regularities and variations both within and across languages. When we say "constraints," we are referring to these observed facts.[1] Often there will be constraints that are true of some words in a language but not others. This variation will be listed among the observed facts. There may also be constraints true of all or most languages. The expected universality of these constraints will also be among the observed facts.

With this notion of constraints, we identify the following desiderata for systems of phonotactic constraints:

1. The constraints should distinguish possible unmarked words from those that are not possible. They must be complete, licensing all and only the unmarked words, and consistent in the logical sense.

2. The constraints should provide a useful foundation for generalizing across languages. A system should provide a means to compare phonotactic constraints across languages and, in particular, a means to identify potential universals. At the same time the system must provide a means of identifying unnecessary constraints. Consequently, a system must provide a means of determining if a set of

[1] Our use of the term "constraint" here should not be confused with the formalized notion of constraint in Optimality Theoretic accounts of phonology, in which the violability of constraints is fundamental to the formal framework. In the sense we mean here, the constraints are the regularities of the surface structure of language that an OT account is intended to account for.

Proceedings of the Society for Computation in Linguistics (SCiL) 2019, pages 247-256.
New York City, New York, January 3-6, 2019

constraints implicitly satisfies constraints that are not explicitly stated. Finally, the system should, in principle, be capable of fully describing the entire range of attested human languages in a uniform way.

3. The phenomena the constraints describe are a consequence of actual human behavior. Thus, there actually exists some physical mechanism that can, in principle, determine whether a given set of constraints are satisfied by a given word. Further, these mechanisms do this quickly. Hence the constraints must be feasible in the sense that they lie at a relatively low level of complexity with respect to some computational model (but not necessarily all such models). Finally, the constraints are learnable. There are physical mechanisms that in principle, given some (possibly empty) set of universal constraints and sufficiently many (possibly positive or negative) examples, can generalize to an equivalent set of constraints.

In the next section we sketch the model-theoretic foundations of our methodology. We then proceed, in Sections 3 and 4 to introduce the analytic component of the methodology by applying it to examples drawn from StressTyp2.

In Section 5 we begin to explore the computational component by looking at the computational and cognitive complexity of simple propositional systems of constraints based on adjacency. We follow that, in Section 6, with a similar exploration of simple constraints based solely on precedence ("long distance" constraints).

In Section 7 we describe the algorithmic mechanisms underlying the computational component of the methodology. We then turn, in Section 8, to sketch the space of classes of stringsets definable with more powerful logical mechanisms beyond the propositional logic characterizing these simple classes.

In Sections 9 and 10 we lay out the full methodology and sketch the results of its application to Yidin. Finally, we conclude by revisiting the desiderata we list in the introduction. We then summarize some of the results that have been obtained using this methodology and close by noting some of the ways the model-theoretic approach has been generalized beyond phonotactics, all of which are candidates for potential extensions of our computational tools.

2 Logical Foundations

We interpret the descriptions of phonotactic stress patterns collected in the StressTyp2 database as sets of strings over an alphabet, Σ, of symbols, each of which denotes a particular syllable type. In this paper, we take Σ to represent syllable weight: light (L) or heavy (H);[2] along with diacritics denoting stress level: none (no mark), secondary stress (`), or primary stress (´). For our logical formulae we include symbols denoting arbitrary weight (σ) and stress marks denoting arbitrary stress (*), some stress (i.e., not unstressed) ($^+$) and non-primary stress (i.e., either secondary or unstressed) ($^-$). So $\sigma\overset{+}{\mathrm{H}}\overline{\mathrm{L}}\grave{\sigma}$ would denote a word consisting of an arbitrary unstressed syllable, followed by a heavy syllable with either primary or secondary stress, followed by a light syllable with either secondary or no stress and followed by any syllable with secondary stress.

The semantics of our logic is built on two relations between strings. The first, based on adjacency, is the substring relation. A string, v, is a substring of another, w, if and only if (iff) the symbols of v occur in order as a contiguous block in w. Formally

$$v \preccurlyeq w \quad \overset{\text{def}}{\iff} \quad w = w_1 v w_2,\ w_1, w_2 \in \Sigma^*$$

The second relation is based on precedence. A string, v, is a subsequence of another, w, iff the symbols of v occur in w in order but not necessarily contiguously. Formally

$$\begin{aligned} v \sqsubseteq w \quad \overset{\text{def}}{\iff} \quad & v = v_1 v_2 \cdots v_n, \text{ and} \\ & w = w_1 v_1 w_2 v_2 \cdots v_n w_{n+1}, \\ & v_i, w_i \in \Sigma^* \end{aligned}$$

2.1 Logical formulae

Our focus, in this paper, is on a propositional logic in which the atoms are either substrings or subsequences of a string. The set of substrings occurring in a string is traditionally referred to as its set of factors. We extend this to subsequences, specifying substring factor or subsequence factor only when necessary to avoid confusion.

Substring factors are denoted with just the sequence of symbols that comprise the factor. Subsequence factors are distinguished by placing the connective '. .' between the symbols of the factor.

[2] The full database includes five levels of weight but we need only these two here.

φ	$\mathcal{W} \models \varphi$
$\sigma_1\sigma_2\cdots\sigma_n \in \{\rtimes\}\cdot\Sigma^*\cdot\{\ltimes\}$	$\sigma_1\sigma_2\cdots\sigma_n \preccurlyeq \{\rtimes\}\cdot\mathcal{W}\cdot\{\ltimes\}$
$\sigma_1 .. \sigma_2 .. \cdots .. \sigma_n, \quad \sigma_1\sigma_2\cdots\sigma_n \in \Sigma^*$	$\sigma_1\sigma_2\cdots\sigma_n \sqsubseteq \mathcal{W}$
$\neg\varphi$	$\mathcal{W} \not\models \varphi$
$\varphi_1 \wedge \varphi_2$	$\mathcal{W} \models \varphi_1$ and $\mathcal{W} \models \varphi_2$

Figure 1: The semantics of our logical formulae

Note that we do not license atoms that mix substring and subsequence, although complex formulae may well include atoms of both types. In addition, substring factors may include either a left-end marker ('$\rtimes$') or a right-end marker ('$\ltimes$') or both, in which case we say that they are anchored. In addition to these atoms the language includes the full set of Boolean connectives with their semantics derived from the semantics of negation ('$\neg$') and conjunction ('$\wedge$').

The model-theoretic semantics of this logic are given formally in Figure 1. An atom is true of a string iff it is a factor of the appropriate kind of that string. By augmenting the models with endmarkers in defining the semantics of substring models, anchored substring factors are required to occur at the either the left end or right end of the string or to span the entire string, as appropriate. The semantics of subsequence factors is insensitive to string boundaries. The Boolean operators are defined canonically. We use the notation '$\mathcal{W} \models \varphi$' to denote the satisfaction relation between strings and formulae; a string $\mathcal{W}$ satisfies a formula φ iff φ evaluates to TRUE with respect to the string $\mathcal{W}$.

Note that, while our models (i.e., strings) have internal structure which come into play when we add quantifiers to the logic, that structure is only significant here in assigning truth values to the atoms. Each string is simply a Boolean valuation of the set of all atoms occurring in a formula; this is truly a propositional logic.

3 An example: Cambodian

Let us take for example the Cambodian language, as described in StressTyp2:

(1) a. In words of all sizes, primary stress falls on the final syllable.
b. In words of all sizes, secondary stress falls on all heavy syllables.
c. Light syllables occur only immediately following heavy syllables.
d. Light monosyllables do not occur.

We begin by examining Constraint 1a. Using our logical notation, we write this constraint as a positive literal:

(1a) $\acute{\sigma}\ltimes$ (Preliminary)

Let us now continue to Constraint 1b. Here, we may be tempted to say something like this:

(1b) $\overset{*}{\text{H}} \rightarrow \grave{\text{H}}$ (Wrong)

However, this would mean "If a heavy syllable happens, then a heavy syllable with secondary stress also happens." What Constraint 1b really says is "No heavy syllable without secondary stress occurs," which in turn implies "No unstressed heavy syllable occurs, and no heavy syllable with primary stress occurs." So we write instead a conjunction of two negative literals:

(1b) $\neg\text{H} \wedge \neg\acute{\text{H}}$ (Preliminary)

However, this is still not quite the correct constraint, as it would prevent primary stress on a heavy syllable, which Cambodian requires if it is final. As is not unusual, many of the descriptions of stress patterns rely on common phonological assumptions and are inconsistent or ambiguous in their absence. Indeed, the constraint we actually want is "No unstressed heavy syllables occur."

(1b) $\neg\text{H}$ (Final)

This is a single negative literal. In fact, we can rewrite Constraint 1a as a negative literal as well:

(1a) $\neg\bar{\sigma}\ltimes$ (Preliminary)

Note that, since the semantics '$\bar{\sigma}$' is disjunctive, this expands into a conjunction of four negative literals:

(1a) $\neg\text{L}\ltimes \wedge \neg\grave{\text{L}}\ltimes \wedge \neg\text{H}\ltimes \wedge \neg\grave{\text{H}}\ltimes$ (Preliminary)

Next, we look at Constraint 1c. Its implication is that a light syllable cannot immediately follow another light syllable. Further, any light syllable must be preceded by some (heavy) syllable, so they do not begin a word:

(1c) $\neg \overset{*}{\mathrm{L}}\overset{*}{\mathrm{L}} \wedge \neg \rtimes \overset{*}{\mathrm{L}}$ (Final)

Finally, we have Constraint 1d, that light monosyllables do not occur:

(1d) $\neg \rtimes \overset{*}{\mathrm{L}} \ltimes$ (Final)

It is clear at this point that any word satisfying Constraint 1c also satisfies Constraint 1d, so including the latter is logically unnecessary. Therefore the following conjunction suffices to describe these constraints:

(1) $\neg \bar{\sigma} \ltimes \wedge \neg \mathrm{H} \wedge \neg \overset{*}{\mathrm{L}}\overset{*}{\mathrm{L}} \wedge \neg \rtimes \overset{*}{\mathrm{L}}$ (Preliminary)

This will be refined again at the end of the next section.

As we will see shortly, this form, a conjunction of negative literals, is significant from a cognitive and complexity-theoretic perspective.

4 Phonotactic regularities

Constraint 1d is logically unnecessary in the description of Cambodian, but it is still interesting phonologically. For example, the stress pattern of Alawa, a language with only one syllable weight and two levels of stress, is given in StressTyp2 as

(2) In words of all sizes, primary stress falls on the penultimate syllable.

This constraint, of course, cannot apply to monosyllables, which have no penultimate syllable. Again, the description relies on common phonological assumptions; in this case the meaning is ambiguous. There are four distinct possibilities for what might happen to monosyllables in Alawa:

1. Monosyllables do not occur (similar to Constraint 1d).
2. Monosyllables occur, and are always stressed.
3. Monosyllables occur, and are never stressed.
4. Monosyllables occur, and may or may not be stressed.

In the case of Alawa, monosyllables do occur and are always stressed, so Constraint 1d is not satisfied by Alawa, even though the original description suggests it would be.

This leads us naturally to consider what kinds of constraints do, in fact, occur universally across languages. One such putative universal is that all words contain exactly one syllable with primary stress (which we will call 1-Stress). Hyman (2009) argues, in the context of unifying the analysis of stress and tone, that 1-Stress is better analyzed as the conjunction of two constraints, one of which is that every word contains *some* syllable with primary stress (which, following Hyman, we will refer to as obligatoriness). The other is that no word contains *more than one* syllable with primary stress (culminativity).

Between the description of Alawa and our observation regarding monosyllables, we see that Alawa satisfies this constraint. We also see that, in Cambodian, Constraint 1a logically implies obligatoriness for words of at least one syllable. If we wanted to be truly accurate, we might rule out words of less than one syllable:

(0) When a human says something, they actually say something.

Formally:

(0) $\neg \rtimes \ltimes$ (Final)

This may be overly pedantic, but it explicitly states that every word contains at least one syllable. Such things matter when using logical machinery to test hypotheses.

Looking at Cambodian's English description, culminativity is a (pragmatic) implicature of Constraint 1a, but it is neither explicitly stated nor logically implied. If this constraint says that primary stress falls on the final syllable and we assume culminativity, then primary stress may not occur on any non-final syllable. We obtain a complete description by augmenting this constraint to account for obligatoriness and culminativity.

(1a) $\neg \bar{\sigma} \ltimes \wedge \neg \rtimes \ltimes \wedge \neg \acute{\sigma} \overset{*}{\sigma}$ (Final)

(1) $\neg \bar{\sigma} \ltimes \wedge \neg \rtimes \ltimes \wedge \neg \acute{\sigma} \overset{*}{\sigma} \wedge \neg \mathrm{H} \wedge \neg \overset{*}{\mathrm{L}}\overset{*}{\mathrm{L}} \wedge \neg \rtimes \overset{*}{\mathrm{L}}$ (Final)

5 Local constraints

Conjunctions of negative literals, such as the final formula for Cambodian, characterize the lowest level of a strict hierarchy of logics explored by Rogers and Pullum (2011) and Rogers et al. (2012) (also see Section 8 of this paper), which characterize a range of sub-Regular classes of stringsets. These classes of stringsets are also characterized by classes of finite-state automata, grammars, and, more importantly, abstract properties of the sets in the class. The constrast between

classes corresponds to a ranking of cognitive complexity based on the nature of the information in a string: each class is characterized by the kinds of features of strings that a mechanism must be sensitive to in order to determine whether a string is in a given set.

Because the pattern of phonological stress in Cambodian is completely described by a conjunction of finitely many negative atomic formulae, a mechanism that is able to make judgments as to whether a word satisfies each constraint only needs to be sensitive to whether certain substrings of some fixed length occur in the word.

Specifically, '$\neg$H' requires sensitivity only to single syllables in isolation while the other constraints in Cambodian require a mechanism to be sensitive to pairs of consecutive syllables. Such a pattern is called "Strictly 2-Local," or "SL_2", The '2' in "SL_2" is a parameter giving the length of factors that a mechanism must necessarily be sensitive to in order to be able to make judgments about well-formedness.

From a procedural perspective, these judgments can be made by a *scanner*, a mechanism that simply scans a window of a fixed size across the input, one symbol at a time, looking at each factor in sequence. Since SL_k stringsets are defined by conjunctions of negative constraints, it suffices to look for unlicensed factors, rejecting the string if any are found.

This generalizes to any pattern completely described by a conjunction of finitely many negative literals: such a pattern is SL_k, where k is the length of the longest factor. For example, Alawa is SL_3:

(2) $\neg\bar{\sigma}\bar{\sigma}\ltimes \wedge \neg\acute{\sigma}\overset{*}{\sigma}\overset{*}{\sigma} \wedge \neg\overset{*}{\sigma}\acute{\sigma}\ltimes \wedge \neg\rtimes\bar{\sigma}\ltimes \wedge \neg\rtimes\ltimes$ (Final)

Since each k-factor can be extended to an equivalent $(k+1)$-factor by extending it with all possible next syllables, a pattern that is SL_k must also be SL_{k+1}.

If we say 'SL' with no explicit factor length, what we really mean is "SL_k for some k." A constraint can be shown to be SL simply by giving a formula that witnesses this. Of course, this merely gives an upper bound.

Generally, lower bounds can be found by appealing to the abstract properties of classes of stringsets. In this case we can show that Alawa is not SL_2 using the fact that SL_k stringsets are characterized by a *suffix substitution closure* property: if some $(k-1)$ factor occurs in two strings that satisfy the constraint then the results of swapping suffixes of the two strings that start with that factor will also satisfy the constraint. (Intuitively, recognizing the penult requires noting that it is followed by a single syllable and no more: $\rtimes\sigma\sigma\acute{\sigma}\sigma\ltimes$ and $\rtimes\sigma\acute{\sigma}\sigma\sigma\ltimes$ share the same 2-factors, so a mechanism needs a window of size at least three.)

In SL stringsets, each constraint must be satisfied at every position in the word. Since a single violation suffices to rule out a word, a mechanism only needs to be sensitive to factors in isolation. Logical complements of SL constraints (coSL), which are defined by disjunctions of positive literals, also only require sensitivity to factors in isolation (accepting if any required factor occurs) but combinations of positive and negative literal constraints are not so simple: in order to verify that certain factors occur somewhere in the word while others do not a mechanism must be sensitive to the entire set of factors in the word. Constraints requiring sensitivity to this entire set are "Locally Testable," or "LT". Much as factor-width may be specified in SL stringsets, we may say a pattern or constraint is LT_k, where the salient factors have a length of at most k. Again like SL patterns, every LT_k pattern is LT_{k+1}.

Since a mechanism capable of making judgments based on the set of factors in a word must necessarily be able to check these factors individually, any constraint that is SL_k or coSL_k must also be LT_k. Further, since the entire set of factors is available, any Boolean combination of factors may be used as an LT constraint. Giving a constraint in this form is sufficient to show that it is LT, and again we can use abstract properties of the LT stringsets to show that a constraint is not in the class.

Obligatoriness by itself is coSL_1, hence LT_1, as witnessed by its expression as a single positive atomic constraint: '$\acute{\sigma}$'. It is not, in itself, SL_k for any k. Obligatoriness can only be satisfied by a system of SL constraints if they require primary stress to fall within some fixed distance from either end of the word.

Similarly, culminativity is not SL. It can only be satisfied by a system of SL constraints if they require *all* syllables with primary stress to occur within a fixed distance of either end of the word. Since satisfaction of an LT constraint depends only on the set of factors of a string, LT constraints cannot distinguish be-

tween strings that have the same factorization, thus we can see by example that culminativity is also not LT_3, as 'LLH́LL' and 'LLH́LLH́LL' contain the same set of substring 3-factors: $\{\rtimes\mathrm{LL}, \mathrm{LLH\acute{}}, \mathrm{LH\acute{}L}, \mathrm{H\acute{}LL}, \mathrm{LL}\ltimes\}$. This counterexample generalizes to any factor-width, so culminativity is not LT (*a fortiori* also not coSL).

6 Piecewise constraints

All local constraints are given in terms of adjacency. If we instead use precedence, then culminativity is quite a simple constraint:

(3) $\neg\acute{\sigma}..\acute{\sigma}$ (Final)

A constraint described by a negative atomic formula of length k is "Strictly k-Piecewise", or "SP_k". A mechanism must be sensitive to subsequences of length k in isolation in order to determine whether a string is in an SP_k stringset.

Culminativity in particular is SP_2. Another SP constraint comes from Nubian, this one SP_3. It is also LT_2, but not SL:

(4) a. If a word contains a non-final heavy syllable, then no light syllable with primary stress occurs in that word.

(4a) $\neg(\overset{*}{\mathrm{H}}\overset{*}{\sigma} \wedge \acute{\mathrm{L}})$ [or: $\overset{*}{\mathrm{H}}\overset{*}{\sigma} \rightarrow \neg\acute{\mathrm{L}}$] (Adjacency)

(4a) $\neg\acute{\mathrm{L}}..\overset{*}{\mathrm{H}}..\overset{*}{\sigma} \wedge \neg\overset{*}{\mathrm{H}}..\acute{\mathrm{L}}$ (Precedence)

Although Constraint 4a is both LT and SP, It is not the case that every LT constraint is SP. If a string satisfies an SP constraint, then every string formed by deleting finitely many symbols from that string also satisfies that same constraint (Rogers et al., 2010). Knowing that, it is impossible for obligatoriness to be satisfied by a system of only SP constraints.

In order to capture obligatoriness using precedence, a mechanism must be sensitive to the set of all subsequences in a string. These constraints are "Piecewise Testable", or "PT". Just as Boolean combinations of SL constraints are LT, Boolean combinations of SP constraints are PT. As with the local classes, stringsets that are the complement of SP stringsets (i.e., that are coSP), while properly PT are effectively no harder to recognize than SP stringsets. Obligatoriness is coSP as witnessed by '$\acute{\sigma}$', the same formula that witnesses that it is coSL; SL_k and SP_k converge for $k = 1$.

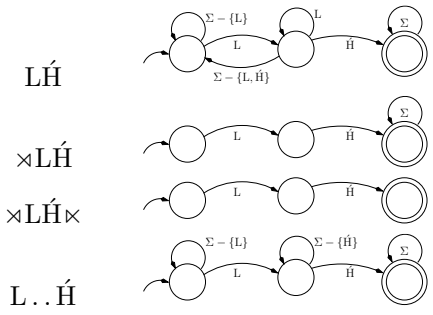

Figure 2: Automata for substring and subsequence factors.

7 Computational Foundations

Our computational workbench is a theorem prover (written in Haskell) based on the equivalence between logical formulae for finite sequences (successor models) and finite state automata established originally by Medvedev (1964); Büchi (1960) and Elgot (1961). The underlying idea is quite simple. One constructs automata for each of the atomic formulae of the logic which accept exactly those strings which satisfy that atom. Since our logic (so far) is quantifier free and our models are simply strings, the automata work directly on the models.

Examples of automata for anchored and unanchored substring formulae and for a subsequence formula are given in Figure 2. The Boolean connectives are implemented via the corresponding automaton constructions: conjunction corresponds to intersection, negation to complement.

Using this procedure on a formula for a stress pattern yields an automaton that recognizes the set of all strings that satisfy that pattern. The StressTyp2 database includes minimal deterministic finite state automata for nearly all of the lects[3] it includes and these can be read directly into the workbench. Hence we can establish the correctness of a formula relative to the automata-theoretic interpretation given in the database by testing the equivalence of that automaton and the one constructed from our formula. One way of doing this (although not the most efficient way) is to construct the automaton between the sets recognized that recognizes the set of strings that the formula incorrectly excludes (undergenerates), and another that recognizes the set that the formula incorrectly

[3] StressTyp2 adopts the convention of using the bare combining form 'lect' to denote a distinct pattern and, by extension, the class of languages that exhibit it.

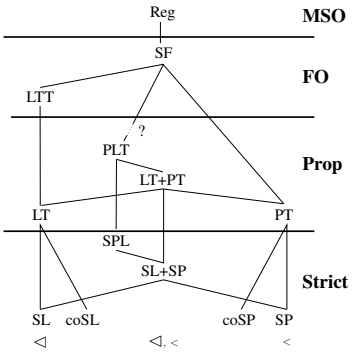

Figure 3: A cognitive complexity hierarchy.

includes (overgenerates); the formula is correct if both sets are empty.

The automata from the database are useful in analysis as well. The workbench includes algorithms that, given a finite state automaton, can determine whether the stringset it recognizes is Strictly Local (Caron, 2000) or if it is Strictly Piecewise (Rogers et al., 2010) and, if it is, determines the factor width parameter k. In addition, it includes the algorithms given by Rogers and Lambert (2017) and Rogers and Lambert (to appear) which can, given an automaton, extract the set of forbidden substring or subsequence factors which suffice to characterize the stringset recognized by that automaton if it is SL or SP (or a conjunction of $\mathrm{SL} + \mathrm{SP} + \mathrm{coSL} + \mathrm{coSP}$ constraints) and can produce an automaton that recognizes an $\mathrm{SL} + \mathrm{SP} + \mathrm{coSL} + \mathrm{coSP}$ approximation of that stringset if it is not.

In each case the approximation, if it is not exact, overgenerates and, so, by taking the difference between the approximation and the original automaton one gets a stringset that provides guidance in formulating the non-strict constraints necessary to completely characterize the original pattern.

8 Higher level constraints

Culminativity is neither SL nor LT. It is possible to enforce culminativity using only adjacency but in order to do so a mechanism must distinguish occurrences of primary stress by their position in the string. The logic we are working with here is the propositional fragment of the hierarchy of logics explored by Rogers et al. (2012). The higher levels (i.e. regular stringsets that are not $\mathrm{LT} + \mathrm{PT}$) represent strings as the same first-order models, with binary predicates for adjacency and precedence, but also allow for quantification over positions (first-order) or sets of positions (monadic second-order). Recognizing whether a string satisfies a quantified formula requires inferring information that is not explicit in the string which changes the nature of the cognitive task, not just its magnitude.

First-order formulae that employ only adjacency characterize the Locally Threshold Testable ($\mathrm{LTT}_{k,t}$) class. A mechanism recognizing such a stringset can count occurrences of k-factors, but only up to a threshold: the second parameter, t. It cannot distinguish t or more distinct occurrences.

If, on the other hand, these first-order formulae make use of precedence ('$<$'), then the result is something strictly stronger, the Star Free (SF) stringsets. Such stringsets are defined much like Regular stringsets, except they are closed under complement rather than the Kleene star.

Finally, if we allow monadic second-order quantification we can define all and only the Regular stringsets. This completes the sub-Regular hierarchy shown in Figure 3; the classes in the middle of the diagram are the stringsets definable by combinations of local and piecewise constraints (conjunctions at the restricted level, any Boolean combinations at the full propositional level and atoms that mix adjacency and precedence in the PL classes).

Some of the constraints listed in StressTyp2 are most naturally expressed in terms of the quantified logics but these turn out to never be necessary for stress patterns.

9 Methodology

We explore patterns by alternating between analysis based on linguistic and logical knowledge and computational analysis and synthesis using the tools of the workbench. In this paper, we began with an analytic phase, transforming the English description of a pattern into a logical description. During this translation process, one can use the abstract characterizations of the complexity classes as a guide to the form needed to express a given constraint. Assuming this translation is successful, these constraints provide an upper-bound on the complexity of the pattern as a whole.

Alternatively, for patterns already described by automata, one can start with a computational analytic phase, using the workbench to extract systems of SL, SP, coSL and coSP constraints from

those automata (Rogers and Lambert, to appear). If the pattern is not simply $\mathrm{SL}+\mathrm{SP}+\mathrm{coSL}+\mathrm{coSP}$ the result of these computational methods is an approximation that is.

Another alternative is to work directly from a corpus of annotated examples using learning algorithms based on Chandlee et al. (2018), which are currently being incorporated into the workbench. Again, the result is a (possibly exact) approximation that is $\mathrm{SL} + \mathrm{SP} + \mathrm{coSL} + \mathrm{coSP}$.

In all three cases, the workbench implements a computational synthesis phase which represents these systems of constraints as automata. If an automaton is provided for the pattern the correctness and completeness of the constraints can be checked computationally against this automaton by constructing an automaton that recognizes the symmetric difference between the two, which can either be examined directly or used to generate examples of strings that satisfy the constraints but should be excluded or those that should not be excluded but fail to satisfy the constraints. If there is no existing automaton to work against, one can generate strings up to a given length-bound and look for inconsistencies. In any case, if there is under- or overgeneration the structures of these residues guide a return to the analytical phase, adding or modifying constraints in order to account for the differences.

Once the conjunction of logical constraints correctly describes the pattern in question, the workbench can minimize the description by removing constraints that are logically implied by others. Because these subregular classes form a proper hierarchy and are all closed under intersection, the complexity of the stringset is simply the maximum of the complexity of the constraints that describe it.

Our workbench can find all of the minimal independent subsets of a set of constraints constraints that describe the same pattern as the full set. However, if a constraint of higher complexity is implied by a set of lower-complexity constraints a smaller subset in which the higher-complexity constraint is explicit will be not be an accurate indication of the complexity of the stringset, rather a larger set of lower-complexity constraints would be preferred. So the workbench can also check sets of constraints provided by the user, such as the constraints identified earlier in the process either from analysis of other patterns or from theoretical analysis of the linguistic phenomena under study, against the rest of the set and find independent subsets that are minimally complex. It should be noted that minimal descriptions from a linguistic perspective may well not be the same as the minimal descriptions from a complexity-theoretic perspective. But as long as they are logically equivalent, the complexity result is still valid.

	$\lhd$	$<$
one $\acute{\sigma}$	$\mathrm{LTT}_{1,2}$	$\boxed{\mathrm{PT}_2}$
obligatoriness	$\boxed{\mathrm{coSL}_1}$	$\boxed{\mathrm{PT}_1}$
culminativity	$\mathrm{LTT}_{1,2}$	$\boxed{\mathrm{SP}_2}$
no $\bar{\mathrm{H}}$ before $\acute{\mathrm{H}}$	SF	$\boxed{\mathrm{SP}_2}$
no $\overset{*}{\mathrm{H}}$ with $\acute{\mathrm{L}}$	LT_1	$\boxed{\mathrm{SP}_2}$
nothing before $\acute{\mathrm{L}}$	$\boxed{\mathrm{SL}_2}$	$\boxed{\mathrm{SP}_2}$
alternation	$\boxed{\mathrm{SL}_2}$	SF
no light monosyllables	$\boxed{\mathrm{SL}_3}$	PT_2

Table 1: Constraints in Yidin expressed locally and piecewise.

One can use this same mechanism to determine whether a pattern satisfies a putative universal constraint by merely checking whether this constraint is implied by those that describe the pattern in question. When the corpus of patterns of StressTyp2 was tested for both obligatoriness and culminativity, it was discovered that while every lect satisfies culminativity, there are two that do not satisfy obligatoriness, namely Seneca and Cayuga. These are languages that Hyman identifies as not satisfying "the more accent-like properties of obligatoriness and culminativity" (Hyman, 2009).

When working with a set of lects, one will collect a library of non-strict constraints. The workbench can use this to automatically determine if a new pattern can be completely described by a conjunction of strict constraints and some subset of this library. When the StressTyp2 corpus was analyzed, only five non-strict constraints were needed to describe the entire set of patterns.

10 Yidin: An example

Yidin is described as:

(5) a. In words of all sizes, primary stress falls on the left-most heavy syllable, else on the initial syllable.

b. In words of all sizes, secondary stress falls iteratively on every second syllable in both directions from the main

stress.

c. Light monosyllables do not occur.

Table 1 shows the constraints we derived from this description. The left column is an English gloss of the constraint, while the remaining two columns note the complexity class in which the constraint falls on both the local and piecewise branches of the hierarchy. Alternation, SF on the piecewise side, is only SL_2 on the local side. Similarly, "no $\bar{\mathrm{H}}$ before $\acute{\mathrm{H}}$" is SF on the local side, but only SP_2 on the piecewise side. Thus, considering either branch of the hierarchy in isolation brings the conclusion that the pattern is SF, but using a mix of constraints from both sides shows that the pattern is $\mathrm{SL}_3 + \mathrm{coSL}_1 + \mathrm{SP}_2$.

11 Conclusion

We have introduced an approach to formalizing linguistic patterns that is based on a model-theoretic analysis of strings. We work primarily with propositional formulae for which satisfaction depends the substrings or subsequences that occur in a string. Definability of sets of strings in these logics characterizes the lowest classes of the local and piecewise subregular hierarchies.

This foundation fulfills all of the desiderata for systems of phonotactic constraints that we identify in the introduction. The logical constraints are unambiguous and fully explicit. As we have seen, the process of translating constraints stated in English to an equivalent logical form can illuminate inconsistencies and ambiguities that are usually resolved by pragmatic linguistic assumptions.

By comparing systems of logical constraints along with the sets of their models one can generalize across languages, identifying constraints that are common to classes of languages and testing putative universals against those classes. Again, using standard model-theoretic tools, one can identify minimal independent sets of constraints, eliminating those that are logically implied by the others.

We have demonstrated the breadth of coverage of this approach with respect to one range of phonotactic phenomena by using it to fully characterize the phonological stress patterns gathered in the StressTyp2 database. In Heinz (2018), Heinz argues that essentially all of phonotactics falls within the the classes of this subregular hierarchy. Moreover there is a growing body of work (Chandlee and Heinz, 2018; Chandlee et al., 2015, 2014; Chandlee, 2014; Jardine, 2017, 2016a,b) that suggests that many phonological processes can be captured by functions based on these same sorts of logics.

Most importantly, all of these model-theoretic tools are effective in the sense of being computable. Moreover, the classes of the hierarchy correspond directly to the nature of the information about a string that any mechanism must be sensitive to in order to determine if it satisfies a constraint. Consequently, the hierarchy provides a measure of the relative complexity of constraints. They are also all learnable in the limit from positive data by algorithms of low computational complexity.

Our analysis of StressTyp2 demonstrates that phonological stress is extremely simple. Our focus is the methodology itself, not the phonological results that have been obtained using it. But we note that, as reported elsewhere, when this methodology was applied to a corpus of the 106 lects in StressTyp2 that have both English language and automata descriptions, it was found that all but eight (92.5%) were $\mathrm{SL} + \mathrm{SP} + \mathrm{coSL}$, six more needed some subset of a library of LT constraints consisting of obligatoriness and three other constraints, and the remaining two were properly Regular, sharing a constraint based on hidden stress alternation. If secondary stress were to surface in these latter two then they would simply be SL. This means that, aside from the two patterns involving the properly Regular constraint, no pattern in the corpus requires a mechanism to infer additional information beyond that which is present in the surface string. Moreover, the only piecewise constraints are SP, forbidding the coöccurrence of certain syllables as in the case of culmanitivity, confirming Heinz (2014).

It has been verified that all patterns in this corpus satisfy culminativity, and the two patterns that do not satisfy obligatoriness, as expected by Hyman (2009), have been identified.

The logical and computational methods we use have applications beyond phonotactics. Similar techniques have proven useful in phonology in general. In particular, for exploring phonological functions (Chandlee and Heinz, 2018; Chandlee et al., 2014; Chandlee, 2014), morphology (Chandlee, 2017), and tone (Jardine, 2017, 2016a,b). These have also been used to analyze syntax (Graf, 2014; Rogers, 1998).

References

J. R. Büchi. 1960. Weak second-order arithmetic and finite automata. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 6:66–92.

Pascal Caron. 2000. Families of locally testable languages. *Theoretical Computer Science* 242:361–376.

Jane Chandlee. 2014. *Strictly Local Phonological Processes*. Ph.D. thesis, The University of Delaware.

Jane Chandlee. 2017. Computational locality in morphological maps. *Morphology* 27:599–641.

Jane Chandlee, Rémi Eryraud, Jeffery Heinz, Adam Jardine, and Jonathan Rawski. 2018. Learning with partially ordered representations. Submitted to ALT 2019.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics* 2:491–503.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2015. Output strictly local functions. In Marco Kuhlmann, Makoto Kanazawa, and Gregory M. Kobele, editors, *Proceedings of the 14th Meeting on the Mathematics of Language (MoL 2015)*. Chicago, USA, pages 112–125.

Jane Chandlee and Jeffrey Heinz. 2018. Strictly locality and phonological maps. *Linguistic Inquiry* 49(1):23–60.

Calvin C. Elgot. 1961. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society* 98:21–51.

R. W. Goedemans, Jeffrey Heinz, and Harry van der Hulst. 2015. `http://st2.ullet.net/files/files/st2-v1-archive-0415.tar.gz`. Retrieved 24 Jun 2015.

Thomas Graf. 2014. Beyond the apparent: Cognitive parallels between syntax and phonology. In Carson T. Schütze and Linnaea Stockall, editors, *Connectedness: Papers by and for Sarah van Wagenen*. volume 18 of *UCLA Working Papers in Linguistics*, pages 161–174.

Jeffrey Heinz. 2014. Culminativity times harmony equals unbounded stress. In Harry van der Hulst, editor, *Word Stress: Theoretical and Typological Issues*, Cambridge University Press, Cambridge, UK, chapter 8.

Jeffrey Heinz. 2018. The computational nature of phonological generalizations. In Larry Hyman and Frank Plank, editors, *Phonological Typology*, Mouton De Gruyter, volume 23 of *Phonetics and Phonology*, chapter 5, pages 126–195.

Larry M. Hyman. 2009. How (not) to do phonological typology: the case of pitch-accent. *Language Sciences* 31(2–3):213–238.

Adam Jardine. 2016a. Computationally, tone is different. *Phonology* 32(2):247–283.

Adam Jardine. 2016b. *Locality and non-linear representations in tonal phonology*. Ph.D. thesis, University of Delaware.

Adam Jardine. 2017. The local nature of tone-association patterns. *Phonology* 34:385–405.

Yu. T. Medvedev. 1964. On the class of events representable in a finite automaton. In Edward F. Moore, editor, *Sequential Machines; Selected Papers*, Addison-Wesley, pages 215–227. Originally published in Russian in Avtomaty, 1956, 385–401.

James Rogers. 1998. *A Descriptive Approach to Language-Theoretic Complexity*. (Monograph.) Studies in Logic, Language, and Information. CSLI/FoLLI.

James Rogers, Jeff Heinz, Margaret Fero, Jeremy Hurst, Dakotah Lambert, and Sean Wibel. 2012. Cognitive and sub-regular complexity. In Glyn Morrill and Mark-Jan Nederhof, editors, *Formal Grammar 2012*, Springer, volume 8036 of *Lecture Notes in Computer Science*, pages 90–108.

James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. 2010. On languages piecewise testable in the strict sense. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language: 10th and 11th Biennial Conference, MOL 10, Los Angeles, CA, USA, July 28-30, 2007, and MOL 11, Bielefeld, Germany, August 20-21, 2009, Revised Selected Papers*, Springer Berlin Heidelberg, Berlin, Heidelberg, pages 255–265.

James Rogers and Dakotah Lambert. 2017. Extracting forbidden factors from regular stringsets. In *Proceedings of the 15th Meeting on the Mathematics of Language*. Association for Computational Linguistics, pages 36–46. http://aclweb.org/anthology/W17-3404.

James Rogers and Dakotah Lambert. to appear. Extracting subregular constraints from regular stringsets. Under review.

James Rogers and Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20(3):329–342.