Online Learning of ITSL Grammars

Jacob K. Johnson
Kahlert School of Computing
University of Utah
jacob@nnnNNnnn.info

Aniello De Santo
Dept. of Linguistics
University of Utah
aniello.desanto@utah.edu

Abstract

This paper presents the first incremental learning algorithm for input-sensitive TSL languages (ITSL). We leverage insights from De Santo and Aksënova (2021)'s ITSL batch-learner to generalize Lambert (2021)'s string extension learning approach to online learning of TSL. We discuss formal properties of the extension, and evaluate the effectiveness of both the original TSL learner and the new ITSL learner on a variety of phonotactic patterns.

1 Introduction

In the mathematical study of linguistic dependencies, the subregular (McNaughton and Papert, 1971; Heinz, 2011a,b; Chandlee and Heinz, 2016) class of Tier-based Strictly Local languages (TSL; Heinz et al., 2011) has gained prominence due to its ability to account for a variety of local and long-distance phonotactic phenomena. TSL as a formal class draws its linguistic inspiration from autosegmental phonology (Goldsmith, 1976), and it is characterized by two components: i) strictly local constraints on adjacent segments, and ii) a tier projection mechanism selecting string elements from a subset of the alphabet over which to enforce such constraints. Long-distance dependencies are thus thought of as local dependencies over strings where irrelevant segments (i.e. segments not part of the alphabet subset) are masked out.

From a typological perspective, the relativized adjacency at the core of the tier-based local constraints has made the TSL class fruitful in characterizing a vast amount of both local and unbounded phonotactic phenomena (McMullin, 2016, a.o.), and tier-locality has been proposed as a general mechanism to account for unbounded processes across linguistic domains (Aksënova et al., 2016; Vu et al., 2019; Graf, 2022a,b). Additionally, a variety of extensions of TSL have been proposed that take advantage of the relativized adjacency intuition while enriching the way elements of the tier are selected (Mayer and Major, 2018; Graf and Mayer, 2018; De Santo and Graf, 2019). In particular, De Santo and Graf (2019) observe that by conditioning tier-membership not just on the identity of a symbol, but also on its local contexts (which symbols precede or follow it) it is possible to generalize TSL to a class of languages (*input-sensitive* TSL, or ITSL) capturing the interaction of local and non-local processes simultaneously.

From a learnability perspective, TSL has been shown to be efficiently learnable in the limit from positive data only (Gold, 1967), assuming a batch-learning set-up where all data are fed to the learner at once (Jardine and Heinz, 2016; Jardine and McMullin, 2017) and more recently also in an online learning setting (Lambert, 2021). As Lambert (2021) observes, online learning of subregular classes seems to be a fundamental step in exploiting mathematical insights to develop learning algorithms that are plausible from a human perspective — given that batch learning assumes simultaneous access to all prior input. Moving beyond TSL, De Santo and Aksënova (2021) propose an efficient batch learning algorithm for (multiple) ITSL grammars with tier-constraints bounded to $k=2$, extending the TSL learner of Jardine and Heinz (2016) and the multiple TSL learner of McMullin et al. (2019). In this work, we leverage the insights of De Santo and Aksënova (2021), and we show how a minor modification to the definition of symbol allows us to extend Lambert (2021)'s TSL learner to an online ITSL learner. We thus contribute the first online learning algorithm for ITSL, including an open source Python 3 implementation of both the new ITSL extension and the original online TSL learner. We also follow the lead of Aksënova (2020) and Johnson and De Santo (2023), and offer a preliminary evaluation of the performance of both algorithms on data representing a variety of phonotactic patterns.

We start with some formal preliminaries (Section 2) necessary to ground our modification of the work in Lambert (2021), and provide some background on TSL and ITSL in Section 3. Section 4 presents the core intuitions behind the existing online TSL

Proceedings of the Society for Computation in Linguistics (SCiL) 2024, pages 257-267.
Irvine, California, June 27-29, 2024

learner, and then our extension to an ITSL learner. Finally, we conduct a preliminary evaluation of both algorithms on natural and artificial datasets (Section 5), and conclude with a broader discussion of current results and future steps.

2 Notation and Terminology

We assumes familiarity with set notation, specifically the union operator $\cup$, the element operator $\in$, the subset operator $\subseteq$, and the power set function $\mathcal{P}(\cdot)$. Sets are denoted as surrounded by curly braces $\{\}$, whereas angle brackets $\langle\rangle$ are used for ordered tuples.

Σ is used to denote some finite set of symbols, the alphabet. Σ^* is the set of all strings of finite length that can be formed using 0 or more instances of symbols from Σ. $\Sigma^k \subseteq \Sigma^*$ denotes the set of all strings that can be formed using exactly k instances of symbols from Σ. Likewise, $\Sigma^{\leq k} \subset \Sigma^*$ denotes the set of all strings that can be formed using k or fewer instances of symbols from Σ.

A language L is some subset of Σ^*. A grammar G can be thought of as a way to determine membership of a string in a stringset. If we denote the language associated with a grammar G as $L(G)$, G can be defined as a function to determine, for any string w, whether $w \in L(G)$.

In this paper, strings are denoted in `monospace` font, and ε denotes the empty or 0-length string. $|w|$ indicates the length of, or number of symbols in, a string w. The variable σ is commonly used to represent individual symbols, while the variables u,v,w,x,y are commonly used to represent strings. The concatenation of strings u and v, denoted uv, is the simple concatenation of the sequence of characters making up that string. That is, given $u = \texttt{ab}$ and $v = \texttt{cd}$, the concatenation $uv = \texttt{abcd}$. Concatenation is notated identically for individual symbols: given $\sigma_1 = \texttt{e}, \sigma_2 = \texttt{f}$, $uv\sigma_1\sigma_2 = \texttt{abcdef}$.

A string u is a substring of a string w iff $\exists x,y \in \Sigma^*$ such that $xuy = w$. Intuitively, this means that u is a substring of w if w contains u within it, without skipping or reusing symbols. A string $u = \sigma_1 \cdots \sigma_{|u|}$ is a subsequence of a string w iff $\exists x_1,\cdots,x_{|u|+1} \in \Sigma^*$ such that $\sigma_1 x_1 \cdots x_{|u|} \sigma_{|u|} x_{|u|+1} = w$. Intuitively, this means that u is a subsequence of w if w contains u within it, without reusing symbols.

3 Background: TSL and ITSL

As mentioned, TSL (Heinz et al., 2011) formalizes the linguistic notion of a phonological tier (Goldsmith, 1976). We can think of a tier T as a subset (e.g., only sibilants) of the original alphabet available to a language. Then, given a string w, tier projection can be understood as forming a relativized locality domain by "masking out" all segments in w that do not belong to the tier alphabet, while preserving the ordering relations among segments in T. Long-distance dependencies (restrictions over segments that are non adjacent in the original string) can then be characterized as local dependencies within such relativized domain, and can thus be enforced by strictly local constraints of width k (i.e. k-grams). In terms of its fundamental components then, TSL is parameterized by the width (k) of the tier-constraints *and* by T — which defined the elements that are relevant to the dependencies. The interested reader is referred to Lambert and Rogers (2020) for a detailed characterization of this class in terms of model and automata theory, as well as to De Santo and Graf (2019) and Lambert and Rogers (2020) for a discussion of its closure properties.

While TSL has been shown to provide insightful characterizations for a variety of unbounded dependencies (Heinz et al., 2011; McMullin, 2016; Graf, 2017, a.o.), phonotactic studies cross-linguistically have revealed substantial limits to its expressivity tied to its projection mechanism — how tier-membership is evaluated (McMullin, 2016; Mayer and Major, 2018; Baek, 2017; Graf and Mayer, 2018; De Santo and Graf, 2019).

For example, in the Ineseño Chumash language of Southern California, a regressive sibilant harmony with unbounded locality ([s] and [ʃ] may not co-occur anywhere within the same word, cf. a) overrides a restriction against string-adjacent *st, *sn, *sl that results in a pattern of dissimilation (Applegate, 1972; McMullin, 2016). For instance, /sn/ surfaces as [ʃn] (cf. b, c) unless there is an [s] following in the string, in which case it surfaces as [sn] (cf. d):

1) Unbounded sibilant harmony

a. /k-su-ʃojin/	kʃuʃojin	"I darken it"

2) /s/ → [ʃ] when preceding (adjacent) [t, n, l]

b. /s-niʔ/	ʃniʔ	"his neck"
c. /s-nanʔ/	ʃnanʔ	"he goes"

3) Long-distance harmony overides palatalization

d. /s-net-us/	snetus	"he does it to him"

Figure 1 exemplifies why this overall pattern, involving an interaction of local and non-local constraints, is not TSL. Since [sn] is sometimes observed in a string-adjacent context (as in d), it must

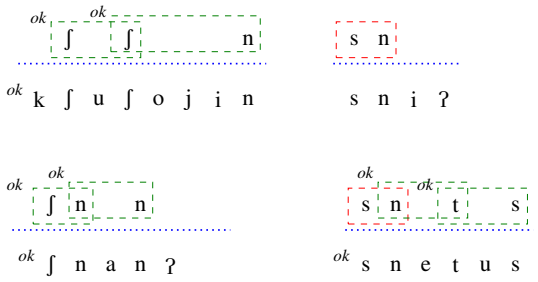

Figure 1: Example of a failed TSL analysis of Ineseño Chumash, adapted from De Santo and Graf (2019). Sibilants and [t,n,l] are tier symbols.

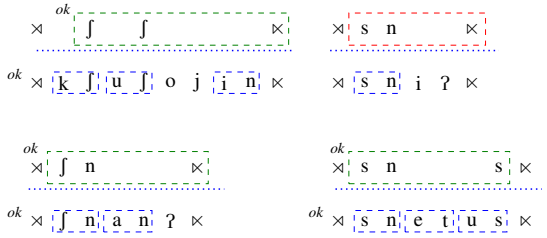

Figure 2: Example of an ITSL analysis of Ineseño Chumash, adapted from De Santo and Graf (2019). Assume a 2-local projection and 3-local tier constraints. [n,t,l] are projected on the tier only when immediately preceded by [s,ʃ]. ⋊ and ⋉ are word-boundaries.

be permitted as a 2-gram on a tier — even though it is only allowed when a segment such as [s] follows them later in the string. But then, a TSL grammar would have no means of banning *sn when there is no subsequent [s] in the string. Vice-versa, if we ban *sn on T, then the grammar will not be able to allow it when another [s] follows on the tier. Additionally, we might point out that the difference between (b) and (d) could be resolved by extending the tier-grammar to consider 3-grams. However, in order to ban *sn, every occurrence of [n] in the string must projected on the tier (and in fact, to really capture the generalization, every occurrence of t and l too). Since the number of [n,t,l] segments between two sibilants is potentially unbounded, no TSL grammar can generally account for this pattern, independently of the dimension of the tier k-grams.

In light of this, De Santo and Graf (2019) suggest to approach such limit by extending the locality window of the TSL projection. The m-Input-sensitive Tier-based Strictly k-Local (m-ITSL$_k$) class is thus defined by allowing the projection mechanism to consider the m-local context of a segment (i.e., its local surrounding environment) before projecting it on a tier.

See Figure 2, adapted from De Santo and Graf (2019), for a sketch of how this approach allows us to characterize the Ineseño Chumash pattern: by increasing the locality of the projection to 2 the grammar is allowed to project [n] iff it is immediately preceded by a sibilant in the input string, and then use 3-local tier constraints to ban $\{^*sn(\neg s), ^*\int ns\}$, in addition to the factors needed to enforce the usual sibilant harmony patterns. Thus, the possible unboundedness of /n/ is not a problem, since/n/ is now relevant for the projection only when adjacent to a sibilant.

More formally, ITSL is characterized by establishing tier-projection as an input strictly local function (Chandlee and Heinz, 2018) over m segments. This modification takes full advantage of the original definition of tier-projection by Heinz et al. (2011), and extends TSL to a class that retains all its well-behaved subregular properties while significantly increasing its typological coverage.

From a learnability perspective, learning a TSL grammar from data alone implies being able to infer not only the relevant constraints, but also (or especially) the content of T. This is even more challenging for ITSL, since it relies on a more complex mechanism to establish tier membership. However, TSL$_k$ and m-ITSL$_k$ grammars have both been shown to be efficiently (polynomial in time and input) learnable in the limit from positive data in the sense of Gold (1967), even when the tier-alphabet is not known a priori (Jardine and Heinz, 2016; Jardine and McMullin, 2017; De Santo and Aksënova, 2021). Additionally, Lambert (2021) has recently proposed an algorithm for incremental learning of TSL. In what follows, we overview Lambert (2021)'s TSL learner and show how it can be extended to ITSL.

4 Online Learning Algorithms

Following Lambert (2021), we restrict this discussion to learning in the limit in the sense of Gold (1967). In this sense, given a set of strings $W \subseteq L(G)$, for some target grammar G, a learner function φ should output a learned grammar G' which is equivalent to G for sufficient data. This discussion is also restricted to incremental learning specifically. That is, the learner does not consider all the input at once, but instead evaluates a single item from the input (and a previously proposed grammar) at each step.

Lambert (2021) develops an incremental learner for TSL in the style of Heinz (2010). Heinz (2010) overviews several subregular classes of string languages sharing the common property that each string

in the language can be mapped to an element in the grammar, in terms of general String Extension Learners (McNaughton and Papert, 1971; Simon, 1975; Rogers and Pullum, 2011). The fundamental intuition centers around the notion of a *factor*, a connected substructure of a string. For strictly local languages (McNaughton and Papert, 1971), for example, a factor is an adjacent sequence of symbols. A stricly local language is the set of all strings containing only allowed (or not containing any forbidden) factors — and it is fully characterized by the set of all factors. Lambert (2021) exploits the fact that this property extends to TSL, where factors need to be defined over a subset of *salient* symbols in the alphabet (the tier), to provide a structural representation of TSL grammars that naturally lends itself to online learning efficienty in the learning paradigm of Gold (1967).

4.1 Learning TSL Online (Lambert, 2021)

Lambert (2021) identifies two core components of a TSL_k grammar that need to be identified by the learner, when T is not provided a priori: a) the set of underlying constraints (the strictly local factors of width k and b) the set of symbols that are *salient* to such constraints — the elements in T. In order to infer which elements belong to T then, Lambert (2021) relies two necessary and sufficient properties of any element *not* in T: free insertion and free deletion (Lambert and Rogers, 2020). Intuitively, if a segment is not salient to tier-constraints, that it is essentially invisible to the grammar: that is, there should be no way to restrict its distribution. Thus, such elements should be freely insertable and freely deletable in all strings without chance of affecting the well-formedness of such strings with respect to the grammar. Once the set of salient symbols has been defined, the learner can then infer strictly local constraints over the input filtered of irrelevant symbols.

Formalizing these observations, a TSL grammar is represented as the pair $\langle G_\ell, G_s \rangle$, where G_ℓ is the set of *attested factors* of width bounded above by $k+1$ and G_s is the set of *augmented subsequences* of length bounded above by k.

For a given k, the set of attested factors can be used to define salience, and it is bound to $k+1$ so to allow for an evaluation of both free insertability (so adding one symbol to k) and free deletability. For instance, the set of attested factors for a TSL_2 grammar for the string `cabacba` is: {ε,a,b,c,ab,ac,ba,ca,cb,aba, acb,bac,cab,cba}.

As mentioned, once the set of salient symbols has been detected, the next step is to infer the relevant k-local constraints. In batch learning, it would be possible to do a first pass over the input to infer tier-membership, and then a second pass over the same input with all non-salient symbols masked out in order to select constraints. However, in an online setting performing a second pass on the input would require to retaining every observed item, thus resulting in unbounded space requirements. Lambert (2021) gets around this obstacle by relying on the notion of *augmented subsequences*.

Subsequence	Intervener Sets
ε	{{}}
a	{{}}
b	{{}}
c	{{}}
aa	{{b},{b,c}}
ab	{{},{c}}
ac	{{}}
ba	{{}}
bb	{{a,c}}
bc	{{a}}
ca	{{},{b}}
cb	{{},{a}}
cc	{{a,b}}

Table 1: Augmented Subsequences extracted from `cabacba` by a TSL learner with $k = 2$, example adapted from Lambert (2021).

A subsequence is a factor over relativized adjacency: that is, a sequence of symbols that appear in order but not necessarily adjacent to each other. An augmented subsequence is a pair consisting of an attested subsequence, and a set of symbols that is attested to intervene among elements of such subsequence. Importantly, the same symbol cannot be both part of the subsequence and of the the intervening set. To illustrate this concept, the set of attested augmented subsequences for a TSL_2 grammar for the string `abbacb` is in Table 1.

For a given width k the space requirement to store all possible subsequences would still be exponential in the size of the alphabet and k. However, Lambert (2021) observes that storing all augmentations is in fact not necessary, due to subsumption relations between interveners. Consider for example the subsequence `ab` as attested in `cabacba`. Possible interveners for `cabacba` are both {} and {`c`}. But for {} to be in the intervener set, it means that `a` and `b` can be immediately adjacent to each other: then, it does not matter how adjacency is relativized. That is, if {} is an intervener then {`c`} trivially also is, and we do

not need to maintain both. This observation generalizes to any subset/superset relation between intervener sets, so that the learner only has to maintain the smallest observed ones (partially-ordered by subset).

We can now overview the full procedure of the Online TSL learner. Initially, the learner assumes an empty grammar (represented as $\langle\{\},\{\}\rangle$). For each input string w from the language, the learner updates the grammar, making use of the following functions:

- $f:\Sigma^*\rightarrow\mathcal{P}\left(\Sigma^{\leq k+1}\right)$ extracts all factors of w of width$\leq k+1$
- $x:\Sigma^*\rightarrow\mathcal{P}\left(\Sigma^{\leq k}\times\mathcal{P}(\Sigma)\right)$ extracts the valid augmented subsequences of width $\leq k$
- $r:\mathcal{P}\left(\Sigma^{\leq k}\times\mathcal{P}(\Sigma)\right)\rightarrow\mathcal{P}\left(\Sigma^{\leq k}\times\mathcal{P}(\Sigma)\right)$ removes all augmented subsequences that are already entailed by other augmented subsequences, that is, $r(S)\subseteq S$

The learning function $\varphi:\langle\mathcal{P}\left(\Sigma^{\leq k+1}\right)\times\mathcal{P}\left(\Sigma^{\leq k}\times\mathcal{P}(\Sigma)\right)\rangle\times\Sigma^*\rightarrow\langle\mathcal{P}\left(\Sigma^{\leq k+1}\right)\times\mathcal{P}\left(\Sigma^{\leq k}\times\mathcal{P}(\Sigma)\right)\rangle$ is as follows:

$$\varphi(\langle G_\ell,G_s\rangle,w)=\langle G_\ell\cup f(w),r(G_s\cup x(w))\rangle$$

To recap, for each string w as input to the learner, G_ℓ is updated to $G_\ell\cup f(w)$ (that is, the factors of w of width $\leq k+1$ are added to G_ℓ), and G_s is updated to $r(G_s\cup x(w))$ (that is, the augmented subsequences of w of width $\leq k$ are added to G_s and any redundancy is removed). Table 2 shows an example of how the grammar is updated as the learner receives strings one by one.

4.2 Generalizing to Online ITSL Learning

To generalize the algorithm presented above to ITSL grammars, it is worth contrasting the formal definitions of the projection function for TSL and ITSL. As discussed, TSL languages have k-local constraints only apply to elements of a tier $T\subseteq\Sigma$. A projection function (also called erasing function) is thus defined as deleting (or masking) all symbols that are not in T.

Definition 1 (TSL Proj.; Heinz et al. (2011))

$$E_T(\sigma):=\begin{cases}\sigma & if\ \sigma\in T\\ \varepsilon & otherwise\end{cases}$$

In order to extend the notion of tier in TSL languages to consider local properties of the segments in the input string, De Santo and Graf (2019) follow Chandlee and Heinz (2018) and define an input-sensitive projection function in terms of local contexts (segments adjecent to a target symbol within a context window of width m).

Definition 2 (Contexts; De Santo and Graf (2019)) *A* m-context c *over alphabet* Σ *is a triple* σ,u,v *such that* $\sigma\in\Sigma$, $u,v\in\Sigma^*$ *and* $|u|+|v|\leq m$. *A* m-context set *is a finite set of* m*-contexts.*

Definition 3 (ISL Proj.; De Santo and Graf (2019)) *Let* C *be a* m*-context set over* Σ *(where* Σ *is an arbitrary alphabet also containing edge-markers* $\rtimes,\ltimes$*). Then the* input strictly m-local *(ISL-*m*) tier projection* π_C *maps every* $s\in\Sigma^*$ *to* $\pi'_C(\rtimes^{m-1},s\ltimes^{m-1})$*, where* $\pi'_C(u,\sigma v)$ *is defined as follows, given* $\sigma\in\Sigma\cup\varepsilon$ *and* $u,v\in\Sigma^*$*:*

$$\begin{array}{ll}\varepsilon & \text{if } \sigma uv=\varepsilon,\\ \sigma\pi'_C(u\sigma,v) & \text{if } \sigma,u,v\in C,\\ \pi'_C(u\sigma,v) & \text{otherwise.}\end{array}$$

In essence, the notion of tier in ITSL is expressed by the set of contexts C, which is the set of tier segments augmented with locality conditions necessary for them to be salient to the tier constraints. Note also that an ISL-1 tier projection only determines projection of σ based on σ itself, showing that this projection function is really just an extension of what happens for TSL languages.

From an algorithmic perspective then, De Santo and Aksënova (2021) observe that having to evaluate salience of tier-segments based on m-local contexts (thus a segment plus its $m-1$ left or right context) can be understood as treating m-grams as unitary elements of the language. Thus, if we characterize every structure previously defined over individual segments over this more complex definition of symbol instead, we can directly lift the rest of the inference procedures for TSL. With this in mind, we can generalize the existing TSL online learning to ISTL in the same way De Santo and Aksënova (2021) generalized TSL batch learning.

For an m-ITSL learner, rather than considering w to be a string of $|w|$ symbols $\sigma_1,\cdots,\sigma_{|w|}\in\Sigma$, we consider it a string of width-m overlapping substrings of w: $\sigma_{1\cdots m},\sigma_{2\cdots m+1},\cdots,\sigma_{|w|-m+1\cdots|w|}\in\Sigma^m$. We can then apply the TSL learning algorithm as sketched above, unchanged.

To illustrate these concepts, consider an ITSL grammar with $m=2$ (the contexts) and $k=2$ (the tier constraints): the string `cabacba` is represented as $\langle$`ca, ab, ba, ac, cb, ba`$\rangle$. Thus the set of attested factors for $k=2$ becomes: $\{\langle\rangle,\langle$`ab`$\rangle,\langle$`ac`$\rangle,\langle$`ba`$\rangle,\langle$`ca`$\rangle,\langle$`cb`$\rangle,\langle$`ab, ba`$\rangle,\langle$`ac, cb`$\rangle,\langle$`ba, ac`$\rangle,\langle$`ca, ab`$\rangle,\langle$`cb, ba`$\rangle,\langle$`ab,ba,ac`$\rangle,\langle$`ac,cb,ba`$\rangle,\langle$`ba,ac,cb`$\rangle,\langle$`ca,ab,ba`$\rangle,\}$ (recall again that at this step we collect factors up to width $k+1$). Note that each unary symbol is now actually

w	G_ℓ	G_s
	{}	{}
cabacba	{ε,a,b,c,ab,ac,ba,ca,cb,aba,acb,bac,cab,cba}	{⟨ε,{}⟩,⟨a,{}⟩,⟨b,{}⟩,⟨c,{}⟩,⟨aa,{b}⟩,⟨ab,{}⟩, ⟨ac,{}⟩,⟨ba,{}⟩,⟨bb,{a,c}⟩,⟨bc,{a}⟩,⟨ca,{}⟩, ⟨cb,{}⟩,⟨cc,{a,b}⟩}
abca	{ε,a,b,c,ab,ac,ba,bc,ca,cb,aba,abc,acb,bac, bca,cab,cba}	{⟨ε,{}⟩,⟨a,{}⟩,⟨b,{}⟩,⟨c,{}⟩,⟨aa,{b}⟩,⟨ab,{}⟩, ⟨ac,{}⟩,⟨ba,{}⟩,⟨bb,{a,c}⟩,⟨bc,{}⟩,⟨ca,{}⟩, ⟨cb,{}⟩,⟨cc,{a,b}⟩}
abbacc	{ε,a,b,c,ab,ac,ba,bb,bc,ca,cb,cc,aba,abb,abc, acb,acc,bac,bca,bba,cab,cba}	{⟨ε,{}⟩,⟨a,{}⟩,⟨b,{}⟩,⟨c,{}⟩,⟨aa,{b}⟩,⟨ab,{}⟩, ⟨ac,{}⟩,⟨ba,{}⟩,⟨bb,{}⟩,⟨bc,{}⟩,⟨ca,{}⟩, ⟨cb,{}⟩,⟨cc,{}⟩}
baba	{ε,a,b,c,ab,ac,ba,bb,bc,ca,cb,cc,aba,abb,abc, acb,acc,bab,bac,bca,bba,cab,cba}	{⟨ε,{}⟩,⟨a,{}⟩,⟨b,{}⟩,⟨c,{}⟩,⟨aa,{b}⟩,⟨ab,{}⟩, ⟨ac,{}⟩,⟨ba,{}⟩,⟨bb,{}⟩,⟨bc,{}⟩,⟨ca,{}⟩, ⟨cb,{}⟩,⟨cc,{}⟩}

Table 2: Progression of Lambert (2021)'s Online TSL learner over an handful of presented strings. The first row includes the empty grammar as initially assumed by the learning algorithm.

Subsequence	Intervener Sets
⟨⟩	{{}}
⟨ab⟩	{{}}
⟨ac⟩	{{}}
⟨ba⟩	{{}}
⟨ca⟩	{{}}
⟨cb⟩	{{}}
⟨ab,ac⟩	{{ba}}
⟨ab,ba⟩	{{}}
⟨ab,cb⟩	{{ac,ba}}
⟨ac,ba⟩	{{cb}}
⟨ac,cb⟩	{{}}
⟨ba,ac⟩	{{}}
⟨ba,ba⟩	{{ac,cb}}
⟨ba,cb⟩	{{ac}}
⟨ca,ab⟩	{{}}
⟨ca,ac⟩	{{ab,ba}}
⟨ca,ba⟩	{{ab}}
⟨ca,cb⟩	{{ab,ac,ba}}
⟨cb,ba⟩	{{}}

Table 3: Augmented Subsequences extracted from cabacba by an ITSL learner with $k=2$ and $m=2$.

a width-2 string over the original alphabet (a width-2 substring of the input string), and thus we represent ITSL factors as tuples, with unary ITSL symbols separated by commas $\langle\sigma_1\sigma_2,\sigma_3\sigma_4,...\rangle$. Then, the set of attested augmented subsequences is as listed in Table 3. Finally, Table 4 exemplifies a run of the new ITSL learner on the same example strings as in Table 2.

In terms of space/time complexity, the original TSL learner total time complexity is $\mathcal{O}(n^k/(k-1)!\cdot|\Sigma|\log|\Sigma|)$, and its space complexity $\mathcal{O}\left(\binom{|\Sigma|}{|\Sigma|/2}\right)$. These results generalize to the ITSL learner, with an additional variable tied to the need of extracting subsequences and interveners defined over complex input symbols. However, Lambert (2021) observes how the TSL learner as defined above can be thought of two separate learners run in parallel and that, thanks to free deletability of the non salient symbols, in most situations the left component of this composite grammar (G_l) is sufficient to both determine salience and function as an acceptor. Thus, an optimization is presented such that the TSL learner can converge in $\mathcal{O}(nk\log|\Sigma|)$ time and $\mathcal{O}(|\Sigma|^{k+1})$ space, where n is the number of strings to learn over, k is the width of the dependencies within the tier and $|\Sigma|$ is the size of the alphabet.

This optimization generalizes as is to ITSL, since nothing was changed in the structure of the learning procedure itself, and thus we only have to incorporate the additional complexity in deriving the salience of the contextually enriched "symbols". Accordingly, the ITSL learner learns in $\mathcal{O}(nk\log(|\Sigma|^m))$ time and $\mathcal{O}((|\Sigma|^m)^{k+1})$ space, preserving the linear time and constant space requirements of the TSL version relative to the input size. Additionally, time and space complexity for the ITSL learner are exponential in the context width, and context and factor width, respectively.

5 Evaluating Online TSL and ITSL

The learning algorithm presented above offers formal convergence guarantees tied to the representation of ITSL and its impact on possible structural restrictions on the hypothesis space of the learner, assuming an input sample fully representative of the target language. In this last part of the paper, we offer a preliminary evaluation of the empirical performance of both the new ITSL learner and of Lambert (2021)'s TSL learner, in terms of consistency with the grammar generating the input (Aksënova, 2020). In particular,

w	G_ℓ	G_s
	{}	{}
cabacba	{⟨⟩, ⟨ab⟩, ⟨ac⟩, ⟨ba⟩, ⟨ca⟩, ⟨cb⟩, ⟨ab, ba⟩, ⟨ac, cb⟩, ⟨ba, ac⟩, ⟨ca, ab⟩, ⟨cb, ba⟩, ⟨ab, ba, ac⟩, ⟨ac, cb, ba⟩, ⟨ba,ac,cb⟩,⟨ca,ab,ba⟩,}.	{⟨⟨⟩, {}⟩, ⟨⟨ab⟩, {}⟩, ⟨⟨ac⟩, {}⟩, ⟨⟨ba⟩, {}⟩, ⟨⟨ca⟩, {}⟩, ⟨⟨cb⟩,{}⟩,⟨⟨ab,ac⟩,{ba}⟩,⟨⟨ab,ba⟩,{}⟩,⟨⟨ab,cb⟩,{ac,ba}⟩, ⟨⟨ac, ba⟩, {cb}⟩, ⟨⟨ac, cb⟩, {}⟩, ⟨⟨ba, ac⟩, {}⟩, ⟨⟨ba, ba⟩, {ac, cb}⟩, ⟨⟨ba, cb⟩, {ac}⟩, ⟨⟨ca, ab⟩, {}⟩, ⟨⟨ca,ac⟩,{ab,ba}⟩,⟨⟨ca,ba⟩,{ab}⟩,⟨⟨ca,cb⟩,{ab,ac,ba}⟩, ⟨⟨cb,ba⟩,{}⟩}
abca	{⟨⟩, ⟨ab⟩, ⟨ac⟩, ⟨ba⟩, ⟨bc⟩, ⟨ca⟩, ⟨cb⟩, ⟨ab, ba⟩, ⟨ab, bc⟩, ⟨ac, cb⟩, ⟨ba, ac⟩, ⟨bc,ca⟩,⟨ca,ab⟩,⟨cb,ba⟩,⟨ab,ba,ac⟩, ⟨ab, bc, ca⟩, ⟨ac, cb, ba⟩, ⟨ba, ac, cb⟩, ⟨ca,ab,ba⟩,}.	{⟨⟨⟩, {}⟩, ⟨⟨ab⟩, {}⟩, ⟨⟨ac⟩, {}⟩, ⟨⟨ba⟩, {}⟩, ⟨⟨bc⟩, {}⟩, ⟨⟨ca⟩, {}⟩, ⟨⟨cb⟩, {}⟩, ⟨⟨ab, ac⟩, {ba}⟩, ⟨⟨ab, ba⟩, {}⟩, ⟨⟨ab, bc⟩, {}⟩, ⟨⟨ab, ca⟩, {bc}⟩, ⟨⟨ab, cb⟩, {ac, ba}⟩, ⟨⟨ac, ba⟩, {cb}⟩, ⟨⟨ac, cb⟩, {}⟩, ⟨⟨ba, ac⟩, {}⟩, ⟨⟨ba, ba⟩, {ac, cb}⟩, ⟨⟨ba, cb⟩, {ac}⟩, ⟨⟨bc, ca⟩, {}⟩, ⟨⟨ca, ab⟩, {}⟩, ⟨⟨ca, ac⟩, {ab, ba}⟩, ⟨⟨ca, ba⟩, {ab}⟩, ⟨⟨ca,cb⟩,{ab,ac,ba}⟩,⟨⟨cb,ba⟩,{}⟩}
abbacc	{⟨⟩, ⟨ab⟩, ⟨ac⟩, ⟨ba⟩, ⟨bb⟩, ⟨bc⟩, ⟨ca⟩, ⟨cb⟩, ⟨cc⟩, ⟨ab, ba⟩, ⟨ab, bb⟩, ⟨ab, bc⟩, ⟨ac, cb⟩, ⟨ac, cc⟩, ⟨ba, ac⟩, ⟨bb, ba⟩, ⟨bc,ca⟩,⟨ca,ab⟩,⟨cb,ba⟩,⟨ab,ba,ac⟩, ⟨ab, bc, ca⟩, ⟨ac, cb, ba⟩, ⟨ba, ac, cb⟩, ⟨ca,ab,ba⟩}.	{⟨⟨⟩, {}⟩, ⟨⟨ab⟩, {}⟩, ⟨⟨ac⟩, {}⟩, ⟨⟨ba⟩, {}⟩, ⟨⟨bb⟩, {}⟩, ⟨⟨bc⟩,{}⟩,⟨⟨ca⟩,{}⟩,⟨⟨cb⟩,{}⟩,⟨⟨cc⟩,{}⟩,⟨⟨ab,ac⟩,{ba}⟩, ⟨⟨ab,ba⟩,{}⟩,⟨⟨ab,bb⟩,{}⟩,⟨⟨ab,bc⟩,{}⟩,⟨⟨ab,ca⟩,{bc}⟩, ⟨⟨ab, cb⟩, {ac, ba}⟩, ⟨⟨ab, cc⟩, {ab, ac, ba, bb}⟩, ⟨⟨ac,ba⟩,{cb}⟩,⟨⟨ac,cb⟩,{}⟩,⟨⟨ac,cc⟩,{}⟩,⟨⟨ba,ac⟩,{}⟩, ⟨⟨ba, ba⟩, {ac, cb}⟩, ⟨⟨ba, cb⟩, {ac}⟩, ⟨⟨ba, cc⟩, {ac}⟩, ⟨⟨bb, ac⟩, {ba}⟩, ⟨⟨bb, ba⟩, {}⟩, ⟨⟨bb, cc⟩, {ac, ba}⟩, ⟨⟨bc, ca⟩, {}⟩, ⟨⟨ca, ab⟩, {}⟩, ⟨⟨ca, ac⟩, {ab, ba}⟩, ⟨⟨ca,ba⟩,{ab}⟩,⟨⟨ca,cb⟩,{ab,ac,ba}⟩,⟨⟨cb,ba⟩,{}⟩}
baba	{⟨⟩, ⟨ab⟩, ⟨ac⟩, ⟨ba⟩, ⟨bb⟩, ⟨bc⟩, ⟨ca⟩, ⟨cb⟩, ⟨cc⟩, ⟨ab, ba⟩, ⟨ab, bb⟩, ⟨ab, bc⟩, ⟨ac, cb⟩, ⟨ac, cc⟩, ⟨ba, ab⟩, ⟨ba, ac⟩, ⟨bb, ba⟩, ⟨bc, ca⟩, ⟨ca, ab⟩, ⟨cb, ba⟩, ⟨ab, ba, ac⟩, ⟨ab, bc, ca⟩, ⟨ac, cb, ba⟩, ⟨ba,ac,cb⟩,⟨ba,ab,ba⟩,⟨ca,ab,ba⟩}.	{⟨⟨⟩, {}⟩, ⟨⟨ab⟩, {}⟩, ⟨⟨ac⟩, {}⟩, ⟨⟨ba⟩, {}⟩, ⟨⟨bb⟩, {}⟩, ⟨⟨bc⟩,{}⟩,⟨⟨ca⟩,{}⟩,⟨⟨cb⟩,{}⟩,⟨⟨cc⟩,{}⟩,⟨⟨ab,ac⟩,{ba}⟩, ⟨⟨ab,ba⟩,{}⟩,⟨⟨ab,bb⟩,{}⟩,⟨⟨ab,bc⟩,{}⟩,⟨⟨ab,ca⟩,{bc}⟩, ⟨⟨ab, cb⟩, {ac, ba}⟩, ⟨⟨ab, cc⟩, {ab, ac, ba, bb}⟩, ⟨⟨ac,ba⟩,{cb}⟩,⟨⟨ac,cb⟩,{}⟩,⟨⟨ac,cc⟩,{}⟩,⟨⟨ba,ab⟩,{}⟩, ⟨⟨ba, ac⟩, {}⟩, ⟨⟨ba, ba⟩, {ab}⟩, ⟨⟨ba, ba⟩, {ac, cb}⟩, ⟨⟨ba, cb⟩, {ac}⟩, ⟨⟨ba, cc⟩, {ac}⟩, ⟨⟨bb, ac⟩, {ba}⟩, ⟨⟨bb, ba⟩, {}⟩, ⟨⟨bb, cc⟩, {ac, ba}⟩, ⟨⟨bc, ca⟩, {}⟩, ⟨⟨ca, ab⟩, {}⟩, ⟨⟨ca, ac⟩, {ab, ba}⟩, ⟨⟨ca, ba⟩, {ab}⟩, ⟨⟨ca,cb⟩,{ab,ac,ba}⟩,⟨⟨cb,ba⟩,{}⟩}

Table 4: Progression of the Online ITSL grammar over an handful of presented strings. The first row includes the empty grammar as initially assumed by the learning algorithm.

we implement the "two parallel learners" version of each algorithm as presented above in Python 3, both for the TSL learner and for our ITSL generalization.[1] We then evaluate performance when trained on input samples representative of patterns corresponding to various subregular classes and designed to mimic natural phonotatic phenomena. [2]

In particular, we conduct evaluations on 11 different training datasets, eight of which were artificially generated from a defined target grammar, and three were word-lists extracted from three natural language corpora with simplified alphabets (see Aksënova, 2020, for details). Specifically, our testing suite includes: word-final devoicing (strictly local); two vowel harmony patterns with a single constraint type (TSL); two vowel harmony patterns with multiple constraints to be evaluated over a single tier (TSL); and three types of ITSL patterns. For the ITSL dependencies, we consider an unbounded tone-plateauing pattern (Hyman and Katamba, 2010; Jardine, 2016) and a pattern of local dissimilation in which the tier consists of `o`, `e`, and `a`, where `oe` is a restricted bigram over the tier, but instances of `o` are only projected to the tier when followed by `x`. [3] We also test a first-last harmony pattern, which

[1] A Haskell implementation of the TSL learner is available as part of the Language Toolkit at https://github.com/vvulpes0/Language-Toolkit-2.

[2] Our code repository, with data for training and testing of both algorithms is available at https://github.com/jacobkj314/online_itsl.

[3] This pattern was originally inspired by the ITSL analysis of Yaka nasal harmony (Hyman, 1995; Walker, 2000) as presented in De Santo and Aksënova (2021). Such analysis hinges on Yaka having nasal-stop clusters. A reviewer points out that it might

establishes a harmonic dependency between the first and the last element in the string. While this pattern has been argued to be unattested in natural languages (Lai, 2015; Avcu, 2017), it is a dependency worth testing in addition to the ones above, as it requires both elements in the constraint to be sensitive to their local context (the end and start symbols, respectively).

We train each learner on 1000 strings randomly sampled from the language generated by the target grammar for the artificial datasets, and on up to 130K words for the simplified natural language corpora (see Table 5). First, we set evaluation criteria defined according to the same pipeline as in Aksënova (2020), and comparable with the evaluation of the 2-ITSL$_2$ batch learner of De Santo and Aksënova (2021) as presented in Johnson and De Santo (2023). Embedding the learned grammar in an acceptor function, we filter strings from Σ^* in length-lexicographical order until 5000 strings are accepted. These 5000 strings are then additionally fed into an acceptor incorporating the original target grammar.[4] Therefore, the score reported for each learner/target grammar pair in Table 6 indicates what proportion of the strings generated by the learned grammar were accepted by the target grammar. For the artificial languages, both learning and testing were repeated over 10 separate trials using a different set of input strings, and we report the average score over these 10 iterations of the full algorithm.

As shown in Table 6, both learners output highly consistent grammars for each of the SL and TSL patterns, even considering the relatively small input size. These results extend to the ITSL learner's performance over the three ITSL patterns, fully consistently with theoretical expectations.[5]

Interestingly, the TSL learner shows (somewhat unexpected) differential performance on the ITSL data. As expected, this learner performs below or at chance for two of these patterns, but the consistency between learned and target grammars on the last ITSL pattern is strikingly high. Recall now that in an ITSL set-up, symbols are only relevant to the tier when conditioned by the appropriate local context. Thus, ITSL patterns viewed from a TSL perspective might look relatively unconstrained: that is, no symbol evaluated in isolation might fit the no free deletion/insertion requirements needed to be considered salient for the tier. A preliminary qualitative evaluation of the output of the TSL learner over this pattern reveals that this is probably the reason for the high acceptance performance: the learner has converged to a strictly local grammar with no tier constraints.

Slightly in contrast with this observation though, the evaluation metric adopted above does not penalize strings that are accepted by the target grammar but rejected by the learned grammar, thus potentially favoring over-restricting grammars (i.e. under-generalization). As a preliminary investigation of this issue, we conduct a second batch of experiments. Table 7 shows, for all the artificial datasets, the proportion of the first 5000 strings accepted by the target grammar that are also accepted by the learned grammar. Together with the results of the previous experiment, these results support the intuition that the ITSL learner converges to more restrictive grammars than the TSL one, as it needs more data to infer that a segment is involved in a dependency independently of context. Still, the high performance of the TSL learner on the ITSL patterns, as well as the performance of both learners on TSL patterns with and without blocking deserve further attention. A more careful investigation of the learned grammars is needed to fully gain insights into the different

	Mean Length (SD)
Word-final devoicing	
A	10
N_G	14.90 (3.70)
Single vowel harmony without blocking	
A	10
N_F	13.92 (3.82)
Single vowel harmony with blocking	
A	10
Several vowel harmonies without blocking	
A	10
Several vowel harmonies with blocking	
A	7.32 (1.08)
N_T	7.85 (2.48)
Unbounded tone plateauing	
A	5
First-Last Assimilation	
A	10
Locally-driven long-distance assimilations (ITSL restriction)	
A	6.20 (0.93)

Table 5: Mean length of the strings in the datasets used for training the learners, based on the union of all sets of strings used by each trial. N_G: German; N_F: Finnish; N_T: Turkish. Where omitted, $SD = 0$

be more appropriate to treat these not as sequences but as pre-nasalized stops and affricates, in which case the harmony pattern would simply be TSL. While getting the linguistic facts right is crucial for a subregular understanding of Yaka, for the sake of this paper what matters is that the abstract example is ITSL, and we keep it for comparison with Johnson and De Santo (2023).

[4]For the natural datasets, each acceptor function incorporates a grammar built to reproduce the underlying pattern, even if that grammar was not technically used to generate the input data.

[5]While omitted here because of space constraints, all learned grammars are available in the repository associated with this paper.

	TSL	ITSL
Word-final devoicing		
T	✓	✓
A	100%	100%
N_G	100%	100%
Single vowel harmony without blocking		
T	✓	✓
A	100%	100%
N_F	100%	100%
Single vowel harmony with blocking		
T	✓	✓
A	100%	100%
Several vowel harmonies without blocking		
T	✓	✓
A	100%	100%
Several vowel harmonies with blocking		
T	✓	✓
A	100%	100%
N_T	100%	100%
Unbounded tone plateauing		
T	✗	✓
A	9.97% (0.51%)	100%
First-Last Assimilation		
T	✗	✓
A	50.02%	100%
Locally-driven long-distance assimilation (ITSL restriction)		
T	✗	✓
A	94.88% (0.15%)	100%

Table 6: Results for Experiment 1. (T)heoretical expectations and performance as mean (and standard deviation) consistency (based on the first 5000 strings accepted by the learned grammar) of the grammars learned by Online TSL and Online ITSL learners on (A)rtificial and simplified (N)atural language input data-sets, measured over 10 iterations. N_G: German; N_F: Finnish; N_T: Turkish. Where omitted, $SD = 0$.

performance of these learners. Understanding the kind of grammars more/less expressive learners converge onto when trained on theoretically less/more expressive patterns might also offer predictions for learnability expectations in human experiments.

6 Conclusion

Formal language theoretical insights have been argued to help bridge typological observations to learnability considerations (Lambert et al., 2021; De Santo and Rawski, 2022). While ITSL offers a good account of phonotactic dependencies from a descriptive characterization perspective, its overall relevance to this broader enterprise is limited by the implausibility of batch learning for humans. In this paper we presented a straightforward generalization of Lambert (2021)'s TSL incremental learner to ITSL, leveraging a more complex definition of tier-symbols in order

	TSL	ITSL
Word-final devoicing		
A	99.96%	71.22% (2.64%)
Single vowel harmony without blocking		
A	9.24%	7.78%
Single vowel harmony with blocking		
A	86.54%	18.64% (1.25%)
Several vowel harmonies without blocking		
A	12.64%	10.26%
Several vowel harmonies with blocking		
A	99.82%	56.90% (1.53%)
Unbounded tone plateauing		
A	99.96%	99.86%
First-Last Assimilation		
A	78.14%	73.01% (0.81%)
Locally-driven long-distance assimilation (ITSL restriction)		
A	99.96%	59.79% (1.23%)

Table 7: Results for Experiment 2. Performance as mean (and standard deviation) completeness (based on the first 5000 strings accepted by the target grammar) of the grammars learned by Online TSL and Online ITSL learners on Artificial language input data-sets, measured over 10 iterations. Where omitted, $SD = 0$.

to determine salience. Taking into account the additional complexity brought by moving segments from unigrams to m-grams, this learner maintains the complexity constraints of the original TSL learner, and its convergence guarantees. An evaluation of learning performance over a variety of patterns also demonstrates the viability of this learning approach beyond theoretical guarantees. Moreover, as already suggested by Johnson and De Santo (2023), we argue that implemented grammatical inference algorithms allow to probe the information about target patterns present in phonotactic corpora, facilitating the study of the relation between data and learnability in humans and machines. In the future, it would be interesting to explore the extent to which this approach can be used to extend Lambert (2021)'s learner to the input-output languages of Graf and Mayer (2018), to stochastic counterparts to TSL and ITSL (Mayer, 2021), and to multiple independent TSL constraints (De Santo and Graf, 2019; McMullin et al., 2019; De Santo and Aksënova, 2021).

Acknowledgements

We are grateful to SCiL's anonymous reviewers for their valuable feedback on this manuscript.

References

Alëna Aksënova. 2020. *Tool-assisted induction of subregular languages and mappings*. Ph.D. thesis, State University of New York at Stony Brook.

Alëna Aksënova, Thomas Graf, and Sedigheh Moradi. 2016. Morphotactics as tier-based strictly local dependencies. In *Proceedings of SIGMorPhon 2016*.

Richard B. Applegate. 1972. *Ineseno Chumash grammar*. Ph.D. thesis, UC Berkeley.

Enes Avcu. 2017. Experimental investigation of the subregular hierarchy. In *Proceedings of the 35st West Coast Conference on Formal Linguistics at Calgary, Alberta Canada*.

Hyunah Baek. 2017. Computational representation of unbounded stress patterns: tiers with structural features. In *Proceedings of the 53rd Meeting of the Chicago Linguistic Society (CLS53)*.

Jane Chandlee and Jeffrey Heinz. 2016. Computational phonology. Ms., Haverford College and University of Delaware.

Jane Chandlee and Jeffrey Heinz. 2018. Strict locality and phonological maps. *Linguistic Inquiry*, 49:23–60.

Aniello De Santo and Alëna Aksënova. 2021. Learning interactions of local and non-local phonotactic constraints from positive input. *Proceedings of the Society for Computation in Linguistics*, 4(1):167–176.

Aniello De Santo and Thomas Graf. 2019. Structure sensitive tier projection: Applications and formal properties. In *International conference on formal grammar*, pages 35–50. Springer.

Aniello De Santo and Jonathan Rawski. 2022. Mathematical linguistics and cognitive complexity. In *Handbook of Cognitive Mathematics*, pages 1–38. Springer.

E Mark Gold. 1967. Language identification in the limit. *Information and control*, 10(5).

John Goldsmith. 1976. *Autosegmental phonology*. Ph.D. thesis, MIT, Cambridge, MA.

Thomas Graf. 2017. The power of locality domains in phonology. *Phonology*, 34:385–405.

Thomas Graf. 2022a. Subregular linguistics: bridging theoretical linguistics and formal grammar. *Theoretical Linguistics*, 48(3-4):145–184.

Thomas Graf. 2022b. Typological implications of tier-based strictly local movement. In *Proceedings of the Society for Computation in Linguistics 2022*, pages 184–193.

Thomas Graf and Connor Mayer. 2018. Sanskrit n-retroflexion is input-output tier-based strictly local. In *Proceedings of SIGMorPhon 2018*.

Jeffrey Heinz. 2010. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 897–906.

Jeffrey Heinz. 2011a. Computional phonology – part 1: Foundations. *Language and Linguistics Compass*, 5(4):140–152.

Jeffrey Heinz. 2011b. Computional phonology – part 2: Grammars, learning, and the future. *Language and Linguistics Compass*, 5(4):153–168.

Jeffrey Heinz, Chetan Rawal, and Herbert G Tanner. 2011. Tier-based strictly local constraints for phonology. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human language technologies*, pages 58–64.

Larry M Hyman. 1995. Nasal consonant harmony at a distance the case of yaka. *Studies in African Linguistics*, 24(1):6–30.

Larry M Hyman and Francis X Katamba. 2010. Tone, syntax, and prosodic domains in luganda. *ZAS Papers in Linguistics*, 53:69–98.

Adam Jardine. 2016. Computationally, tone is different. *Phonology*.

Adam Jardine and Jeffrey Heinz. 2016. Learning tier-based strictly 2-local languages. *Transactions of the ACL*, 4:87–98.

Adam Jardine and Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In *Language and Automata Theory and Applications, 11th International Conference*, LNCS, pages 64–76. Springer.

Jacob K Johnson and Aniello De Santo. 2023. Evaluating a phonotactic learner for MITSL-(2, 2) languages. *Proceedings of the Society for Computation in Linguistics*, 6(1):379–382.

Regine Lai. 2015. Learnable vs. unlearnable harmony patterns. *LI*, 46(3):425–451.

Dakotah Lambert. 2021. Grammar interpretations and learning tsl online. In *Proceedings of the Fifteenth International Conference on Grammatical Inference*, volume 153 of *Proceedings of Machine Learning Research*, pages 81–91. PMLR.

Dakotah Lambert, Jonathan Rawski, and Jeffrey Heinz. 2021. Typology emerges from simplicity in representations and learning. *Journal of Language Modelling*, 9.

Dakotah Lambert and James Rogers. 2020. Tier-based strictly local stringsets: Perspectives from model and automata theory. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 159–166.

Connor Mayer. 2021. Capturing gradience in long-distance phonology using probabilistic tier-based strictly local grammars. In *Proceedings of the Society for Computation in Linguistics 2021*, pages 39–50.

Connor Mayer and Travis Major. 2018. A challenge for tier-based strict locality from Uyghur backness harmony. In *Formal Grammar 2018. Lecture Notes in Computer Science, vol. 10950*, pages 62–83. Springer, Berlin, Heidelberg.

Kevin McMullin. 2016. *Tier-based locality in long-distance phonotactics?: learnability and typology*. Ph.D. thesis, U. of British Columbia.

Kevin McMullin, Alëna Aksënova, and Aniello De Santo. 2019. Learning phonotactic restrictions on multiple tiers. *Proccedings of SCiL 2019*, 2(1):377–378.

Robert McNaughton and Seymour Papert. 1971. *Counter-Free Automata*. MIT Press, Cambridge.

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.

Imre Simon. 1975. Piecewise testable events. In *Automata Theory and Formal Languages 2nd GI Conference*, volume 33 of *Lectures Notes in Computer Science*, pages 214–222, Berlin. Springer.

Mai Ha Vu, Nazila Shafiei, and Thomas Graf. 2019. Case assignment in TSL syntax: A case study. In *Proccedings of SCiL 2019*, pages 267–276.

Rachel Walker. 2000. Yaka nasal harmony: Spreading or segmental correspondence? *Annual Meeting of the Berkeley Linguistics Society*, 26(1):321–332.