VILNIUS UNIVERSITY
INSTITUTE OF COMPUTER SCIENCE
DEPARTMENT OF SOFTWARE ENGINEERING

Algorithmic Learning of Formal Languages of the MTSL$_k$ Class

Unofficial translation of a bachelor's thesis

Presented by:	Ignas Rudaitis
Advisor:	Valdas Dičiūnas, PhD
Reviewer:	Vytautas Valatis, PhD

Vilnius – 2021

Contents

Introduction

Extrapolation of string-sets

There may be a range of reasons to extrapolate sets of strings; in other words, to apply a mapping from smaller sets to larger ones that are supersets of the former.

For instance, we may let our mapping be indifferent to the number of times a piece of the string is repeated: $\{\mathsf{abc}, \mathsf{abbc}, \mathsf{abccc}\} \mapsto \{\mathsf{ab}^m\mathsf{c}^n \mid m, n \in \mathbb{N}\}$.

Problems of this kind demand solutions in a variety of contexts: machine learning, grammatical inference, pattern recognition, and predictive analytics, to name a few. In some cases, distributions or fuzzy sets replace usual sets, especially when it is a deep learning model carrying out the extrapolation.

In the present thesis, we shall discuss a specific problem of string-set extrapolation, one that is characteristic of one particular domain, namely, linguistics, or, more concretely, theoretical phonology. In spite of recent success stories generated by some neural network architectures (especially transformers and GPT in particular (Brown et al., 2020)), linguistics demonstrates a continuing demand for easily interpretable learning models, ones that would allow the scholar to embody hypotheses about first-language acquisition. The extent to which neural networks meet this expectation is insufficient – irrespective of their remarkable receptivity to patterns.

Phonotactics as a use case

Phonotactics is one of the demarcations that one observes in natural languages between well-formed and ill-formed strings. Alongside phonotactics, the reader is more likely to be acquainted with well-formedness at the syntactic or sentence level, also known as *grammaticality*. It subsumes mandatory agreement of words (by case etc.), obligatory complements (cf. *a question regarding the event* : $^{\times}$*a question regarding*), constraints on word order, and others. On the other hand, similar restrictions apply to the combinatorial relations between speech sounds. Such restrictions are observed on a level that is comparatively close to the surface, and are independent of such concepts as word form, root, phrase and the like. It is precisely such restrictions that are termed phonotactics.

Local phonotactics is the most common. For instance, in Lithuanian (Ambrazas et al., 1994), adjacent consonants must agree in palatalization; that is, they must be either both palatalized (◌$^{\text{j}}$, e. g. *Te*[$l^{j}\check{s}^{j}$]*iai*, a toponym), or both plain (◌$^{\text{ɣ}}$, e. g. *pa*[$l^{\text{ɣ}}\check{s}^{\text{ɣ}}$]*as* ‘pale gray’). Importantly, each language has its own rules of phonotactics: consider Russian, where *da*[$l^{j}\check{s}^{\text{ɣ}}$]*e* ‘further’ (where l^{j} and $\check{s}^{\text{ɣ}}$ stand next to each other) is not only a possible string, but an actual word. Some languages exhibit *long-distance* phonotactics as well, where speech sounds need not be adjacent to interact. For example, in so-called *vowel harmony*, all vowels in a word may interact, irrespectively of the consonants that separate them. Of languages of Europe, Finnish, Hungarian, Turkish and some others have this phenomenon.

As a matter of fact, parents and guardians do not teach their children phonotactics. Just as well, they do not segregate well-formed strings from ill-formed ones and then supply them to their children in equal measure. What is more, even if it is common to reward children for learning new skills, that would be an inconceivable scenario for phonotactics. All of this leads us to conclude that phonotactics is learned in an unsupervised manner. This is a commonplace claim in linguistic literature, such as (Chomsky, 1959) or (Yang, 2006).

Local phonotactics can be readily modeled using bigrams. Such models can also be construed as SL_2 (*strictly* 2-*local*) string-sets, if one is to take a formal language theory perspective. If one is provided with a finite sample of positive examples, it is a very simple task to extrapolate it to a minimal SL_2 superset.

SL_2 string-sets do not suffice for long-distance phonotactics. Having surveyed the limits of variation that phonotactics shows in the world's languages, Aksënova et al. (2020) conclude that one sufficient class of string-sets is MTSL_2 (*multiple tier-based strictly* 2-*local*), previously described by McMullin, Aksënova and De Santo (McMullin et al., 2019).

A MTSL_2 set is such set of strings that leaves a SL_2 set when any subset of its alphabet is completely erased. In other words, there is a bigram model operating for each subset of the alphabet.

Description of contribution

From the standpoint of algorithm design, this thesis generalizes an existing algorithm from (McMullin et al., 2019). This precursor algorithm is capable of extrapolating finite samples of positive examples to their minimal N-MTSL_2 supersets. The N-MTSL_2 class of string-sets is more restrictive than MTSL_2, since it incurs an additional constraint of *non-overlapping tiers*.

In the version presented here, the constraint is lifted. Importantly, greater generality notwithstanding, there is no change in asymptotic time complexity. Moreover, it supports any of the MTSL_k classes for arbitary k, which is the span of k-grams. However, it must be noted that the degree of the polynomial that bounds its running time grows in proportion to this k.

It seems that the generalized version *does not* become more inclusive for actual phonotactic patterns in its capacity to extrapolate them. One of the authors of the original algorithm, A. Aksënova, states in a paper co-authored with S. Deshmukh that all MTSL_2 phonotactic patterns are also N-MTSL_2 patterns (Aksënova & Deshmukh, 2018).

Despite that, the generalization presented here maintains its relevance in at least two ways.

Firstly, the idea for the generalization is based on novel propositions on *k-paths*, which are a specific-purpose structure with a commonplace presence in TSL_k and MTSL_k literature (see also 2.5). Generalizing 2-paths to k-paths had been positioned as an open problem by previous scholars. It is conceivable that solving it would contribute to the development of related future algorithms.

Secondly, having surveyed the resulting MTSL_k extrapolation algorithm in a qualitative manner, we easily arrive at a pumping lemma for MTSL_2 languages.

Later on in this thesis, the lemma is stated and proved. This, in turn, leads us to an important question: perhaps this lemma is sufficient of itself to define an interesting class of formal languages, if we are to obtain the closure of its iterated application? Preliminarily, we answer in the affirmative. What corroborates this belief is the remarkable overlap between minimal PUMP_3 supersets (whereby we shall refer to this class) and MTSL_2 supersets, observed empirically. In a simulation, around 95% of strings match, if the we limit their length to 8 (for reasons of efficiency), and owing to its origin in a pumping lemma, $\text{MTSL}_2 \subset \text{PUMP}_3$. Not only that, the lemma essentially constitutes a statement of the PUMP_3 class as a rewriting system, in which one proceeds from “axioms” to “theorems”.

This definition of PUMP_3 is especially simple and intuitive, enough so as to facilitate certain linguistic research and discussion, to the author’s mind. It is conceivable that it would raise the question to what extent, in current theoretical machinery of phonotactic generalization, the complexity that one is met with actually inheres in the phenomenon and not the approach. Also, rewrite systems have not seen much use in linguistics in any similar context as the one presented here, to the best of the author’s knowledge. They seem to only have appeared in this field once, as the transformations of transformational generative grammar (Chomsky, 1956), where they exist to augment context-free grammars. With that in mind, modeling language acquistion using rewriting systems has the potential to pave the way for a new kind of study.

Relevance

The preceding section has already stated the bulk of the present work’s relevance.

In broad strokes, as a work of computer science, this thesis is an applied one. Its entire relevance builds on a demand arising in linguistics.

Goals and objectives

It should be noted that a review of phonotactic typology and theory is intentionally left out of this thesis, and no goal is made to cover these topics beyond strict necessity. Instead, the author would prefer to direct the interested reader to (Aksënova et al., 2020).

To sum up the preceding claims, the goal of this thesis is twofold: (1) to present a novel algorithm for extrapolating MTSL$_k$ string-sets and (2) to present the PUMP$_3$ class of formal languages, also outlining its relevance and relation to MTSL$_2$.

These goals are then reduced to the following objectives and methods:

- to notice and to document an extension point in the existing N-MTSL$_2$ extrapolation algorithm,
- to design an algorithm that builds on this extension point (rational algorithm design and choice of data structures),
- to implement the algorithm in pseudocode and Python (procedural programming),
- to argue for its correctness and efficiency (axiomatic-deductive method),
- to survey this algorithm qualitatively and to note the possibility of a pumping lemma (qualitative study),
- to state and to prove a pumping lemma for MTSL$_2$ languages (axiomatic-deductive method),
- to discuss the significance of the results.

1. Terminology and notation

1.1. Strings and substrings

[Left out because of limited relevance, since the translation eliminates the homonymy between *string* and *word*.]

1.2. String-sets, formal languages, and alphabets

In some contexts, string-sets are synonymously referred to as *formal languages* or just *lanugages*. Both finite and infinite string-sets are possible. An example of a finite string-set (with three elements) would be $\{\mathsf{a}, \mathsf{ab}, \mathsf{abc}\}$. For one that is infinite, consider $\{\mathsf{a}^n\mathsf{b}^n \mid n \in \mathbb{N}\} = \{\mathsf{ab}, \mathsf{aabb}, \mathsf{aaabbb}, \dots\}$.

Each formal language L has its *alphabet*, denoted $\text{alphabet}(L)$. It is the set of all symbols that appear in one or more strings. For instance, $\text{alphabet}(\{\mathsf{a}^n\mathsf{b}^n \mid n \in \mathbb{N}\}) = \{\mathsf{a}, \mathsf{b}\}$.

1.3. Bigrams, *k*-grams and SL$_2$ sets

A *bigram* is a substring of length 2. For example, if our alphabet is $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$, and we use $\rtimes$ and $\ltimes$ as dedicated beginning-of-string and end-of-string symbols, respectively, then $\text{bigrams}(\{\rtimes\mathsf{aba}\ltimes, \rtimes\mathsf{abc}\ltimes\}) = \{\rtimes\mathsf{a}, \mathsf{ab}, \mathsf{ba}, \mathsf{bc}, \mathsf{a}\ltimes, \mathsf{c}\ltimes\}$.

If we were to memorize all bigrams from a set of strings, and declare the rest of possible bigrams forbidden, we will have obtained a minimal SL$_2$ extrapolation. The minimal SL$_2$ extrapolation of the preceding set is $\{\rtimes\mathsf{abc}\ltimes, \rtimes\mathsf{aba}\ltimes\} \mapsto \{\rtimes(\mathsf{ab})^m\mathsf{a}\ltimes \mid m \in \mathbb{N}\} \cup \{\rtimes(\mathsf{ab})^n\mathsf{c}\ltimes \mid n \in \mathbb{N}\}$.

Analogously, n-grams are substrings of length n. Even if k-grams are a less common synonym in general, we prefer this notation here because some relevant names of formal language classes, such as SL$_k$, TSL$_k$, and MTSL$_k$, support it.

1.4. Extrapolation, learning, and inference

The author expects *extrapolation* to be both a fitting and a widely understood term. Here, it refers to mapping from subsets to supersets (or equal sets). The subsets are *inputs* or *data*; they are normally finite. The supersets come from a class chosen in advance, and may be infinite.

Extrapolating means adding new members to a set. In some cases, these additions could be expected to complete a pattern. When it is these cases that concern us, we may also synonymously speak of *learning* the pattern.

Inference is yet another synonym, suggesting one more point of departure. One might indeed construe extrapolation as recovering an original superset that the data *originates* from; what kind of superset had to have existed until the bulk of it was discarded.

1.5. MTSL$_2$ sets

In Slovenian, one says ‘you sleep’, ‘you forget’ and ‘you build’ as *spiš*, *pozabiš* and *zidaš*, respectively. Importantly, the first two words have alternate pronunciations *špiš* and *požabiš*, but one never hears $^{\times}$*židaš*.

This is not accidental: all words with *s, š, z, ž* are part of this variational pattern. (Ozburn & Jurgec, 2015) supply a generalization of this phenomenon, where it is viewed as an optional consonant harmony.

One can artificially endow this harmony with the appearance of a *mandatory* harmony, which is a more common occurrence. In doing so, one can introduce three symbols for sibilants: primary *s* and *š*, and a secondary *ṣ*, which can be freely realized either as *s* or *š*. Analogously, the voiced sibilants become *z*, *ž*, *ẓ*. Now we will write *ṣpiš*, *poẓabiš*, but *zidaš*.

According to (Ozburn & Jurgec, 2015), a *š* or *ž* that is present later on in the word causes a change of *s, z* $\longrightarrow$ *ṣ, ẓ* earlier on in the same word. This change is, however, blocked by the consonants *t, d* that appear in between.

Let us erase all symbols from the words, save for $T = \{$*s, š, ṣ, z, ž, ẓ, t, d*$\}$. We shall make use of an implicit function erase$(y,\ \Sigma \setminus T)$:

- erase(ṣpiš, $\Sigma \setminus T$) = ṣš,
- erase(ẓabiš, $\Sigma \setminus T$) = ẓš,
- erase(zidaš, $\Sigma \setminus T$) = zdš.

To put it differently, we will only leave the subset T of the alphabet in place. In light of such formulation, we can state the harmony as a bigram model on the subset T:

- the bigrams *ṣš*, *ṣž*, *ẓš*, *ẓž* are allowed (these come from the changes induced by the harmony),
- the bigrams *šš*, *šž*, *žš*, *žž* are allowed (recall that the harmony has no effect on primary *š*, *ž*),
- the bigram *st*, *št*, *sd*, *šd*, *zt*, *žt*, *zd*, *žd* are allowed (due to blocking),
- the bigrams $^{\times}$*sš*, $^{\times}$*sž*, $^{\times}$*zš*, $^{\times}$*zž* are forbidden (the harmony must have taken effect).

For this pattern, there exists a MTSL$_2$ string-set – one that would include, among other members, *ṣpiš*, *poẓabiš*, and *zidaš*, but not *spiš*, *pozabiš*, or *ẓidaš*.

Traditionally, one defines MTSL$_k$ sets as the extensions of MTSL$_k$ grammars. A MTSL$_k$ grammar, in turn, is a function $\mathcal{P}(\Sigma) \to \mathcal{P}(\Sigma^k)$ that maps from subsets of the alphabet to the k-grams allowed on it. (Here and onwards, $\mathcal{P}(X)$ will denote the power-set of X.)

It must be noted that some natural languages have simultaneous harmonies on multiple subsets of the alphabet; specifically, on ones that do not overlap. For instance, it is the case in Tamashek Tuareg Berber (Heath, 2005).

1.6. Mathematical notation

In this thesis, one will find the usual notation of elementary set theory, propositional logic, and first-order logic. Of notation that commonly varies, set difference is denoted here by $\setminus$, not $-$, and conjunction by &, not $\wedge$. $\subset$ always refers to the strict superset relation; $\subseteq$ is also used when necessary. The angle brackets $\langle$ and $\rangle$ surround tuples. For zero, $0 \notin \mathbb{N}$, and the empty set is denoted $\emptyset$. $|X|$ is either the cardinality of the set X, or the length of the string X.

Multiplication ($\times$ or juxtaposition) means either the arithmetic or the Cartesian product. Depending on context, the latter operation may return either tuples or strings. Iterated Cartesian products are sometimes denoted as exponentiation, as in Σ^2.

Exponentiation may also refer to iterated concatenation, such as $(\mathsf{ab})^n(\mathsf{cd})^n = \mathsf{abcd}, \mathsf{ababcdcd}, \mathsf{abababcdcdcd}, \ldots$

A $\mathsf{sans\text{-}serif}$ typeface is used exclusively for symbolic constants, which are only equal if spelled equally: the notation alone implies $\mathsf{a} \neq \mathsf{b}$. In pseudocode, function names appear in a non-italic mathematical typeface, as in $\mathrm{alphabet}(X)$). Mathematical $italics$ is for various other uses.

Alphabets are always denoted Σ. Variables that range over symbols are denoted by lower-case Greek letters κ, σ, τ, υ. In other places, lower-case Greek letters have other meanings. Capital letters T, K, Q and others denote sets. An important exception is M and N, which are always integers.

Only the most typical syntactic constructs of structured imperative languages appear in pseudocode. Observe, however, that the loop counters of **for** loops can vary in complex ways, regardless of succinct notation, such as **for** $\sigma_1\sigma_2\cdots\sigma_N \in \Sigma^N$.

2. Analysis of MTSL2IA

2.1. General remarks

McMullin et al. (2019) present an N-MTSL$_2$ extrapolation algorithm, named MTSL2IA. The initialism is expanded as *MTSL$_2$ Inference Algorithm*. Importantly, even though its hypothesis space is restricted to N-MTSL$_2$ by imposing the constraint of non-overlapping tiers, this fact receives only cursory mention. In the work cited, the MTSL$_2$ and N-MTSL$_2$ classes are not distinguished by terminology. The latter abbreviation, N-MTSL$_2$, is a coinage of the present thesis.

2.2. Computational problem

2.2.1. Initial statement

MTSL2IA solves the problem of extrapolating finite string-sets to their N-MTSL$_2$ supersets.

2.2.2. Supersets: minimal, maximal, and others

It must be noted that in stating the problem the way it is done in 1.2.1, trivial solutions are not excluded. Concretely, referring to the problem as extrapolation does not prevent the solution from always returning the same superset, which is *maximal*.

Each N-MTSL$_2$ set is underlain by a numerous ensemble of bigram models. The maximal superset is one where every one of these models declares every bigram *allowed*. Indeed, this kind of extrapolation would be identical to the maximal SL$_2$ superset, which, in turn, would equal the Kleene closure $\Sigma^* = \Sigma \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$, where Σ is the alphabet.

To prevent this definitional loophole, one can amend the problem statement with a criterion of correctness. One common criterion, namely the criterion of *identification in the limit* proposed by E. M. Gold, is described in the following section.

Importantly, if a class Λ of string-sets is partially ordered by its subset relation ($\subset$), then Λ is a lattice class, which (Heinz et al., 2012) prove to be identifiable in the limit simply by always extrapolating to the *minimal* superset.

2.2.3. Criterion of correctness

McMullin et al. (2019) make no mention of a correctness criterion, but the closely related works (Jardine & Heinz, 2016) and (Jardine & McMullin, 2017), in which the classes TSL_2 and TSL_k (which MTSL_k is a intersection closure of) are studied, apply the criterion of identification in the limit, which is due to E. M. Gold (1967).

(Gold, 1967) is presented in a different setting than extrapolation would suggest, but it is eventually equivalent. Following Gold, we assume a specific string-set L from the class Λ. An agent will attempt to guess (identify) the set L. In advance, the agent is only informed about the class Λ. After that, an unlimited number of discrete guessing steps is made by the agent.

In every step i the agent receives a message S_i, which supplies it with some clues about the set L. The nature of the clues is specified independently, and the order in which they are presented is arbitrary; the agent must not make any assumptions regarding it. For every message S_i, the agent must supply a response R_i.

We consider the agent to have identified L in the limit (with respect to the class Λ), if there exists an N, such that $R_N = R_{N+1} = R_{N+2} = \cdots = L$; in other words, if at some step, the agent starts responding with L and does not change their mind in the steps to follow.

Of possible types of clues, (Gold, 1967) distinguishes:

- positive examples only (*TxtEx* setting),
- examples explicitly annotated as positive or negative,
- a chance to ask an oracle if a string is in the set.

The class of regular string-sets, widely known as the extension of regular expressions and finite automata, cannot be extrapolated in a TxtEx setting. (Gold, 1967) supplies a proof. On the other hand, MTSL_2 (as well as N-MTSL_2) are a *subregular* class of string-sets. This denomination is sometimes reserved to subclasses of the regular string-sets that are immune to the negative identifiability result for regular languages due to Gold.

2.3. Exponential-time algorithm

2.3.1. Problem reduction

It is certain that once we have solved the minimal-superset MTSL_2 extrapolation problem for a set X, we can apply the constraint of non-overlapping tiers and hereby obtain an N-MTSL_2 set in a simple way. For this reason, we can make the distinction between MTSL_2 and N-MTSL_2 optional for now.

2.3.2. Exhaustive search over subsets of the alphabet

Let us content ourselves with exponential running time, for the time being. It will be bounded by $\mathcal{O}(2^N)$, or, more strictly $\mathcal{O}(2^{|\Sigma|}N)$, where N is the sum length of the input strings, and because of that, $|\Sigma| \leq N$. This way, we will be able to design a simple minimal-superset MTSL_2 extrapolation algorithm with little effort.

The pseudocode follows:

```
function grammar($X$) is
    $\Sigma \leftarrow \mathrm{alphabet}(X)$
    $\Gamma \leftarrow \emptyset$
    for $T \in \mathcal{P}(X)$ do
        $X' \leftarrow \mathrm{erase}(X,\ \Sigma \setminus T)$
        for $\beta \in \mathrm{bigram}(X')$ do
            $\Gamma \leftarrow \Gamma \cup \{\langle T, \beta \rangle\}$
        end for
    end for
    return $\Gamma$
end function

function member($y, \Gamma$) is
    return $\mathrm{grammar}(\{y\}) \subseteq \Gamma$
end function
```

Of course, the algorithm cannot return the superset in an explicit manner, as it is often infinite. For this reason, a *grammar* serves as a proxy, and is returned instead.

The implied set, which is the result of extrapolating, is called the extension of the grammar. It will be denoted $\ell(\Gamma)$.

If the extrapolation is described by the grammar Γ, then the function $\mathrm{member}(y, \Gamma)$ can answer a membership query for y in $\mathcal{O}(2^{|y|})$ time.

2.4. Non-overlapping subsets of the alphabet

An MTSL$_2$ grammar consists of pairs that have the shape $\langle T, \alpha\beta \rangle = \langle \{\sigma_1, \sigma_2, \dots, \sigma_M\}, \sigma_1\sigma_2 \rangle$. In MTSL$_k$ literature, one often speaks of *tiers* when referring to subsets of the alphabet $T = \{\sigma_1, \sigma_2, \dots, \sigma_M\}$.

Two tiers that allow or restrict the same bigram $\sigma_i\sigma_j$ are said to *not overlap*, if they are nested or disjoint, save for both containing the bigram itself. Symbolically, T_i and T_j do not overlap, if $T_i \subseteq T_j$ or $T_i \cap T_j = \{\sigma_1, \sigma_2\}$.

We will say that a grammar $\Gamma = \{\langle T_1, \alpha_1\beta_1 \rangle, \langle T_2, \alpha_2\beta_2 \rangle, \dots, \langle T_N, \alpha_N\beta_N \rangle\}$ complies with the constraint of non-overlapping tiers, if $\forall i \forall j ((i \neq j \ \& \ \alpha_i\beta_i = \alpha_j\beta_j) \Rightarrow (T_i \subseteq T_j \lor T_i \cap T_j = \{\alpha_i, \beta_i\}))$.

If a set L can be described by a grammar Γ that observes this constraint, then L is not only a MTSL$_2$ set, but also an N-MTSL$_2$ set.

Employing the exponential-time algorithm of 2.3, once the function $\mathrm{grammar}(X)$ has halted and returned a grammar Γ, one can follow up with a post-processing step and prune the grammar. Many strategies are possible, and enforcing the constraint of non-overlapping tiers is one of them.

As a class, N-MTSL$_2$ $\subset$ MTSL$_2$, and as such, such a pruning step could be treated as a second extrapolation in series: finite set $\rightarrow$ MTSL$_2$ $\rightarrow$ N-MTSL$_2$.

2.5. MTSL2IA in pseudocode

The MTSL2IA of McMullin et al. (2019) does no such post-processing. Instead, it builds up N-MTSL$_2$ grammars from the very outset.

The pseudocode of MTSL2IA follows:

```
function mtsl2ia($X$) is
    $\Sigma \leftarrow \mathrm{alphabet}(X)$
    $K \leftarrow \mathrm{paths}(X)$  ❶
    $\Gamma \leftarrow \emptyset$
    for $\sigma_1\sigma_2 \in \Sigma^2$ do  ❷
        $T \leftarrow \Sigma$
        for $\tau \in \Sigma \setminus \{\sigma_1, \sigma_2\}$ do
            $\phi \leftarrow \top$  ❸
            for $\langle \sigma_1, \{\kappa_1, \kappa_2, \ldots, \kappa_N,\ \tau\}, \sigma_2 \rangle \in K$ do  ❹
                if not $\langle \sigma_1, \{\kappa_1, \kappa_2, \ldots, \kappa_N\}, \sigma_2 \rangle \in K$ then  ❺
                    $\phi \leftarrow \bot$
                    break
                end if
            end for
            if $\phi$ then
                $T \leftarrow T \setminus \{\tau\}$
            end if
        $\Gamma \leftarrow \Gamma \cup \{\langle T, \sigma_1\sigma_2 \rangle\}$
        end for
    end for
end function
```

The cross-reference comments ❶, ❷, ❸, ❹, ❺ mark several important places in the pseudocode.

Paths (❶) are, broadly speaking, fragments of the strings in X. They are returned by the function paths(X), which is omitted in the pseudocode. For every string $x \in X$, there are $\mathcal{O}(|x|^2)$ paths. In essence, these are the substrings $\alpha_1\alpha_2 \cdots \alpha_{|x|-1}\alpha_{|x|}$ of x, but structured in a specific way: we speak of the same path, when the two outermost symbols $\alpha_1, \alpha_{|x|}$ match, and so does the set $\{\alpha_2, \ldots, \alpha_{|x|-1}\}$ with the rest of the symbols, irrespective of their order. As such, a path is represented by the tuple $\langle \alpha_1, \{\alpha_2, \ldots, \alpha_{|x|-1}\}, \alpha_{|x|} \rangle$.

On lines ❹ and ❺, the algorithm runs the *path test*: it checks whether removing a symbol τ from a path $\langle \sigma_1, \{\kappa_1, \kappa_2, \dots, \kappa_N,\ \tau\}, \sigma_2 \rangle$ will result in another attested path. This is an important aspect of how MTSL2IA operates, which we discuss in more detail in Section 2.6.

Observe that the outer loop ❷ iterates first over bigrams, not tiers (subsets of the alphabet), unlike in the exponential-time solution. In other words, the algorithm traverses every bigram that is possible with the given alphabet, and looks for tiers on which it can be *restricted*. Let us note that:

- the grammar returned by MTSL2IA, Γ, unlike in the exponential-time algorithm, is a *negative-polarity* grammar: the more elements there are in it, the more restrictions apply, and, consequently, the fewer strings remain in the extrapolated set;
- each bigram can only bear a restriction on one tier; we shall soon show that this is equivalent to restricting it on any number of non-overlapping tiers.

We also remark that the loop on line ❹ does not traverse every single path: it has a search query implicit in it, by which it picks out paths with their outermost elements equal to the current bigram σ_1, σ_2.

Finally, for a symbol τ to pass the path test in general, it must pass it in particular, that is, to pass it with every path $\langle \sigma_1, I \cup \{\tau\}, \sigma_2 \rangle$, varying I. The flag ϕ (❸) keeps track of this. $\top$ and $\bot$ are truth and falsehood, the Boolean constants.

The running time of the algorithm is $\mathcal{O}(N^2)$, where $N = \sum_{x \in X} |x|$.

2.6. Grounds for correctness of MTSL2IA

2.6.1. Relating grammars to data

MTSL2IA starts by restricting each bigram β_i on the maximal tier $T_i = \Sigma$. At first, it holds that $\Gamma_0 = \{\langle \Sigma, \beta_1 \rangle, \langle \Sigma, \beta_2 \rangle, \dots, \langle \Sigma, \beta_N \rangle\}$.

Later on, the tiers T_i become distinct, as the *path test* is repeatedly applied. Any symbol that passes the path test is subsequently removed from the tier.

Let us verify the path test. Consider the Finnish word *päivä* 'day'. Now, $X = \{\text{päivä}\}$. What can be made out of the substring -*äivä* that is attested in the data? Concretely, what does it imply about the tiers, on which the bigram *ää* is allowed or restricted?

On the tier {*ä*}, it is imperative that we allow the bigram, if we are to maintain consistency with the data. After all, the letter *ä* does appear in the word twice, and other symbols would be erased by applying the tier.

Aiming to extrapolate to a minimal superset, let us consider the proposition that has to hold true of a tier, so that we could restrict $^{\times}$*ää* on it. In the data, there are letters *i* and *v* separating the bigram; consequently, on the tiers {*ä*, *i*} and {*ä*, *v*} there would no longer be a direct attestation of the bigram. It would be kept apart by *i* or *v*. In the latter case, applying the tier {*ä*, *v*} would introduce the bigrams *äv* and *vä* – but not *ää*. At this point, we can in fact restrict $^{\times}$*ää*: what matters is that the tier must also have *i* or *v* in it. Therefore, for each T_0, we can include a restriction $\langle T_0 \cup \{\mathsf{ä}, \mathsf{i}\}, \mathsf{ää}\rangle$, and $\langle T_0 \cup \{\mathsf{ä}, \mathsf{v}\}, \mathsf{ää}\rangle$ as well.

-*äivä* is a *minimal path*, given that its outermost symbols (the two letters *ä*) and the rest (*i*, *v*) do not share any elements.

Lemma 1. (*Reading of a minimal path*) From an attestation of a substring of the shape $\sigma_1\,\tau_1\tau_2\cdots\tau_N\,\sigma_2$, for which $\sigma_1, \sigma_2 \notin \{\tau_1, \tau_2, \ldots, \tau_N\}$, one can derive the proposition $\langle T, \sigma_1\sigma_2\rangle \in \Gamma \;\Rightarrow\; \tau_1 \in T \,\vee\, \tau_2 \in T \,\vee\, \cdots \,\vee\, \tau_N \in T$.

Proof idea. Make the running example general and rigorous. □

2.6.2. Validity of the path test

Lemma 2. (*Path test*) Suppose that in a negative-polarity (N-)MTSL$_2$ grammar, only T alone of all tiers has a restriction for $^{\times}\sigma_1\sigma_2$. Let us fix a specific symbol of the alphabet, τ. Now, if for each attested path $\langle \sigma_1, K \cup \{\tau\}, \sigma_2\rangle$ there exists another path $\langle \sigma_1, K, \sigma_2\rangle$, then $\tau \notin T$.

Proof. For all paths that have the shape $\langle \sigma_1, Q, \sigma_2\rangle$, it must be true that either $\tau \in Q$, or $\tau \notin Q$. Note that considering only minimal paths (for them, $\sigma_1, \sigma_2 \notin Q$) does not result in a loss of generality. The interested reader will be able to verify this without difficulty.

We now group the paths into three sets (more precisely, we group those that are minimal) in the following manner:

- the set $\mathbb{K}^+$ will contain the paths $\langle \sigma_1, Q, \sigma_2\rangle$, for which $\tau \in Q$;
- $\mathbb{K}^-$ will contain all such $\langle \sigma_1, Q', \sigma_2\rangle$, for which $\tau \notin Q'$ and $\langle \sigma_1, Q' \cup \{\tau\}, \sigma_2\rangle \in \mathbb{K}^+$;
- finally, $\mathbb{K}^\circ$ will contain the rest of minimal paths $\langle \sigma_1, Q'', \sigma_2\rangle$; for these, $\tau \notin Q''$ will also hold by way of exclusion.

If $|\mathbb{K}^+| \neq |\mathbb{K}^-|$, then one cannot maintain the part of the lemma whereby each $\langle \sigma_1, K \cup \{\tau\}, \sigma_2 \rangle$ has a counterpart $\langle \sigma_1, K, \sigma_2 \rangle$. We remove this possiblity from our consideration, therefore, $|\mathbb{K}^+| = |\mathbb{K}^-| =: \ell$.

Now, we make a pairing $\mathbb{K}^\pm := \{\langle p_1, n_1 \rangle, \langle p_2, n_2 \rangle, \ldots, \langle p_\ell, n_\ell \rangle\}$ so that:

- $\bigcup_i \{p_i\} = \mathbb{K}^+$,
- $\bigcup_i \{n_i\} = \mathbb{K}^-$,
- denoting $p_i = \langle \sigma_1, Q_i, \sigma_2 \rangle$ and $n_i = \langle \sigma_1, Q'_i, \sigma_2 \rangle$, for each i it is true that $Q_i = \{\tau\} \cup Q'_i$.

For the sake of clarity, let us observe that $\mathbb{K}^\pm \subset \mathbb{K}^+ \times \mathbb{K}^-$.

Now, while traversing each $\langle p, n \rangle \in \mathbb{K}^\pm$, we expand $p = \langle \sigma_1, \{\tau,\ v_1, v_2, \ldots, v_N\}, \sigma_2 \rangle$, and analogously for n, except for the absence of τ. Then we apply Lemma 1 and derive the following pair of propositions:

- $\langle T, \sigma_1\sigma_2 \rangle \in \Gamma \ \Rightarrow\ (v_1 \in T \ \vee\ v_2 \in T \ \vee\ \cdots \ \vee\ v_N \in T) \ \vee\ \tau \in T$,
- $\langle T, \sigma_1\sigma_2 \rangle \in \Gamma \ \Rightarrow\ (v_1 \in T \ \vee\ v_2 \in T \ \vee\ \cdots \ \vee\ v_N \in T)$.

The truth of both propositions follows from each $\langle p, n \rangle$; as such, the truth conditions are equal for the two.

Moreover, the lemma demands that a T exist, such that $\langle T, \sigma_1\sigma_2 \rangle \in \Gamma$, and that there be exactly one such T. By *modus ponens*, we cut off the left-hand side of the implication and equate the variables T on both right-hand sides:

$$\begin{gathered}(v_1 \in T \ \vee\ v_2 \in T \ \vee\ \cdots \ \vee\ v_N \in T) \\ \Leftrightarrow \\ (v_1 \in T \ \vee\ v_2 \in T \ \vee\ \cdots \ \vee\ v_N \in T) \ \vee\ \tau \in T.\end{gathered}$$

Hence, $\tau \notin T$. □

2.6.3. Sufficiency of a single tier

Let us also note that for each bigram, MTSL2IA constructs only one tier, in which the bigram can be restricted. We shall argue that any two tiers T_1, T_2 assigned to the same bigram $\sigma_1\sigma_2$, that are either nested ($T_1 \subseteq T_2$) or disjoint ($T_1 \cap T_2 = \{\sigma_1, \sigma_2\}$), can be replaced with their intersection $T_1 \cap T_2$ or their union $T_1 \cup T_2$, respectively.

Lemma 3. (*Nested tiers to intersection*) If Γ is a negative-polarity (N-)MTSL$_2$ grammar, which has the (N-)MTSL$_2$ set as its extension L, and in addition, both $\langle T, \sigma_1\sigma_2 \rangle \in \Gamma$ and $\sigma_1\sigma_2 \in T$ hold, then the augmented negative-polarity grammar $\Gamma' = \Gamma \cup \{\langle A \cup T, \sigma_1\sigma_2 \rangle\}$ also has L as its extension, regardless of the tier augment $A \subset \Sigma$, provided that $A \cap T = \emptyset$.

Proof. When $A = \emptyset$, the proof is trivial. Let us then concentrate on $A \neq \emptyset$. As we do not yet know that the extension of Γ' is L, let us instead denote it L' and then attempt to demonstrate that $L' = L$.

An additional restriction can only make an extension smaller, therefore, $L' \subseteq L$. Assume the contrary, anticipating a contradiction: there exists an x, such that $x \in L$, but $x \notin L'$.

Let $y = \text{erase}(x, \Sigma \setminus T)$ and $y' = \text{erase}(x, \Sigma \setminus (A \cup T))$. Then let $B = \text{bigram}(y)$ and $B' = \text{bigram}(y')$. Importantly, it cannot be the case that $B = B'$, since our construction of Γ' did not involve adding new bigrams; therefore, the differing membership of x in the sets L and L' would remain unexplained. For this reason, there must exist a $b \notin B$, $b \in B'$. From our strategy of constructing y and y', it must follow that $b = \alpha\beta$ or $b = \beta\alpha$, where $\alpha \in A$.

As a consequence, $b \neq \sigma_1\sigma_2$, and as such, if $x \in L$ is true, then so is $x \in L'$; for Γ and Γ' only differ in their restrictions for ${}^{\times}\sigma_1\sigma_2$. Our derivation of both $x \in L'$ and $x \notin L'$ at once is absurd, therefore, the assumption that x exists must have been false.

We thus have that no x, for which $x \in L$, but $x \notin L'$, can exist. In other words for all x, it is the case that $x \in L \Rightarrow x \in L'$, and from this, $L \subseteq L'$. At the same time, we have already derived $L' \subseteq L$. Hence, $L' = L$. □

Lemma 4. (*Disjoint tiers to union*) Assume a negative-polarity (N-)MTSL$_2$ grammar Γ. In it, the bigram ${}^{\times}\sigma_1\sigma_2$ bears no restrictions. Let the restrictions $d_1 = \langle T_1, \sigma_1\sigma_2 \rangle$, $d_2 = \langle T_2, \sigma_1\sigma_2 \rangle$, $d_\cup = \langle T_1 \cup T_2, \sigma_1\sigma_2 \rangle$, none of which are present in the grammar: $d_1, d_2, d_\cup \notin \Gamma$. For their respective tiers, we have $T_1 \cap T_2 = \{\sigma_1, \sigma_2\}$. It then follows that the grammars $\Gamma_+ = \Gamma \cup \{d_1, d_2\}$ and $\Gamma_\cup = \Gamma \cup \{d_\cup\}$ have the same extension L.

Proof. Let us denote the extension of a negative-polarity (N-)MTSL$_2$ grammar X by $\ell(X)$.

From Lemma 3, it follows that $\ell(\Gamma_+) = \ell(\Gamma_+ \cup \Gamma_\cup)$. If we augment a negative-polarity grammar with additional restrictions, its extension can only shrink or remain intact, therefore, $\ell(\Gamma_\cup) \subseteq \ell(\Gamma_+ \cup \Gamma_\cup)$. Combining with the equality obtained earlier, we have $\ell(\Gamma_\cup) \subseteq \ell(\Gamma_+)$. It now remains to show that $\ell(\Gamma_+) \subseteq \ell(\Gamma_\cup)$ is also true.

Assume the contrary: indeed, that $\ell(\Gamma_\cup) \subset \ell(\Gamma_+)$, since we do know that the sets are commensurable. Then, there must exist a string $x \in \ell(\Gamma_+)$, for which, at the same time $x \notin \ell(\Gamma_\cup)$. For this reason, there must also exist a restriction $d \in \Gamma_\cup$, which is violated by x; on the other hand, necessarily, $d \notin \Gamma_+$.

We already know that the restriction d is responsible for this differing membership of x, therefore, we insert it into Γ_+, obtaining $x \notin \ell(\Gamma_+ \cup \{d\})$. Hence, $\ell(\Gamma_+) \neq \ell(\Gamma_+ \cup \{d\})$.

But the only restriction exclusive to $\Gamma_\cup$ is known to us: $d = d_\cup = \langle T_1 \cup T_2, \sigma_1\sigma_2 \rangle$. As a matter of fact, $\Gamma_+ \cup \Gamma_\cup = \Gamma_+ \cup \{d\}$; also, recall that $\ell(\Gamma_+) = \ell(\Gamma_+ \cup \Gamma_\cup)$, and as such, $\ell(\Gamma_+) = \ell(\Gamma_+ \cup \{d\})$.

We have derived an absurd result from the premise that $\ell(\Gamma_\cup) \subset \ell(\Gamma_+)$, namely the result that $\ell(\Gamma_+) \neq \ell(\Gamma_+ \cup \{d\})$, but also, $\ell(\Gamma_+) = \ell(\Gamma_+ \cup \{d\})$. TThe premise must thus have been false. From this contradiction it follows that $\ell(\Gamma_+) \subseteq \ell(\Gamma_\cup)$, since the sets are commensurable.

Lastly, combining $\ell(\Gamma_\cup) \subseteq \ell(\Gamma_+)$ with $\ell(\Gamma_+) \subseteq \ell(\Gamma_\cup)$, we obtain $\ell(\Gamma_\cup) = \ell(\Gamma_+)$. □

3. Generalized algorithm

3.1. A new look at the path reading lemma

Lemma 1. (*Reading of a minimal path*, reproduced from 2.6.1) From an attestation of a substring of the shape $\sigma_1\,\tau_1\tau_2\cdots\tau_N\,\sigma_2$, for which $\sigma_1, \sigma_2 \notin \{\tau_1, \tau_2, \ldots, \tau_N\}$, one can derive the proposition $\langle T, \sigma_1\sigma_2\rangle \in \Gamma \;\Rightarrow\; \tau_1 \in T \,\vee\, \tau_2 \in T \,\vee\, \cdots \,\vee\, \tau_N \in T$.

Let us derive an extrapolation algorithm from this lemma alone. As before, it would consist of a grammar-constructing function (grammar′) and a predicate (member′) for membership queries; only now, its grammars would be even more implicit.

Concretely, tiers, which as we recall, are subsets of the alphabet, will be represented not by enumerating the elements, but as formulae of propositional logic, that assert facts about the membership of symbols in the tier.

Following up the example with Finnish *päivä* ‘day’ (see also 2.6.1), we apply Lemma 1 to the substring $\sigma_1\,\tau_1\tau_2\,\sigma_2 = \mathsf{äivä}$. Put differently, $\sigma_1 = \sigma_2 = \mathsf{ä}$, $\tau_1 = \mathsf{i}$, $\tau_2 = \mathsf{v}$. We would then derive $\langle T, \mathsf{ää}\rangle \in \Gamma \;\Rightarrow\; \mathsf{i} \in T \,\vee\, \mathsf{v} \in T$.

Here, Γ *would not* be a grammar of the propositional variety we mentioned above. In fact, let us use G to denote the latter. Instead, Γ would be the MTSL$_2$ grammar characterized by G. We write this as $\chi(G) = \Gamma$.

Let us decree that $\chi(G) = \Gamma$ is a *negative-polarity* grammar – a set of restrictions, which, if expanded, shrinks the extension $\ell(\Gamma)$ or keeps it intact. We shall use the shorthand $\ell(G)$ for the extension $\ell(\Gamma) = \ell(\chi(G))$.

3.2. Example of an implicit grammar

Päivää is another case of the same Finnish word for 'day', namely, the partitive case. If $X = \{\text{päivää}\}$, then $G = \text{grammar}'(X)$ must equal $G = \{g_{\text{pä}}, g_{\text{pi}}, g_{\text{pv}}, g_{\text{äi}}, g_{\text{äv}}, g_{\text{ää}}, g_{\text{iv}}, g_{\text{iä}}, g_{\text{vä}}\}$, where:

- $g_{\text{pä}} = \bot_{\text{pä}}$ – there cannot exist tiers with a restriction for ${}^{\times}$*pä*, since the symbols *pä* even occur next to each other in the data,
- $g_{\text{äi}} = \bot_{\text{äi}}$,
  $g_{\text{iv}} = \bot_{\text{iv}}$,
  $g_{\text{vä}} = \bot_{\text{vä}}$ – analogously,
- $g_{\text{pi}} = \text{ä} \in T_{\text{pi}}$,
  $g_{\text{pv}} = \text{ä} \in T_{\text{pv}} \lor \text{i} \in T_{\text{pv}}$,
  $g_{\text{iä}} = \text{v} \in T_{\text{iä}}$, – from the minimal paths *päi*, *päiv*, *ivä*, respectively,
- $g_{\text{ää}} = (\text{i} \in T_{\text{ää}} \lor \text{v} \in T_{\text{ää}})\ \&\ \bot_{\text{ää}} = \bot_{\text{ää}}$ – by combining propositions that obtain from the paths *äivä* and *ää* (the conjunction is stored in the grammar in its unsimplified form).

$\bot_{\alpha\beta}$ means $T_{\alpha\beta} = \{\alpha, \beta\}$.

3.3. Unattested bigrams

For every bigram $\alpha\beta$ that is possible with the alphabet but absent from the data, we shall extend the grammar with $\langle \top_{\alpha\beta},\ \alpha\beta \rangle \in G$. This means that ${}^{\times}\alpha\beta$ is necessarily restricted in every possible tier.

Put differently, $\top_{\alpha\beta} = \bigvee_{\tau} \tau \in T_{\alpha\beta}$.

3.4. Membership queries

Answering membership queries can be reduced to testing the subset relation on two extensions. In fact, $\ell(G_1) \subseteq \ell(G_2)$ if and only if $\neg\text{SAT}(G_1\ \&\ \neg G_2)$. Here SAT is the Boolean satisfiability problem, and G_1 and G_2 are treated as propositions, not just sets. The way this is carried out will be discussed in section 3.5.

G_1 and G_2 are in conjunctive normal form (CNF), moreover, in both formulae, each clause κ_i only contains literals related to a single tier. Therefore $\text{SAT}(G_1\ \&\ \neg G_2)$ can be split up into $\text{SAT}(\kappa_1\ \&\ \neg\kappa'_1)\ \&\ \cdots\ \&\ \text{SAT}(\kappa_N\ \&\ \neg\kappa'_N)$. As κ_i is a clause of literals joined by disjunction, this problem can be solved in linear time.

3.5. Grammars as propositions

For each bigram $\alpha\beta$ and, at the same time, for each additional symbol τ on the tier, one can state the proposition $p^{\tau}_{\alpha\beta} = (\tau \in T_{\alpha\beta})$, which is a literal. Each of these literals admits two interpretations: $p^{\tau}_{\alpha\beta} = \top$ or $p^{\tau}_{\alpha\beta} = \bot$; in other words, either as truth or as falsehood.

The grammar G is a set of formulae, every one of which is either a CNF containing as literals only the propositions $p^{\tau}_{\alpha\beta}$ for varying τ, α, β, or the empty clause $\bot_{\alpha\beta}$ that bears a bigram annotation $\alpha\beta$.

The literals $p^{\tau}_{\alpha\beta}$ admit interpretations $\iota^{\tau}_{\alpha\beta}$, where $\iota^{\tau}_{\alpha\beta} \in \{\langle p^{\tau}_{\alpha\beta},\, \top\rangle,\, \langle p^{\tau}_{\alpha\beta},\, \bot\rangle\}$, which correspond directly to the tiers of the grammar being characterized, $\chi(G) = \Gamma$. As a first step towards this, one accepts the conjunction of all formulae in G. If $G = \{\phi_1, \phi_2, \ldots, \phi_N\}$, then one has to entertain $\phi_1 \;\&\; \phi_2 \;\&\; \cdots \;\&\; \phi_N$. Having picked an arbitrary bigram $\alpha\beta$, every set of interpretations $I_{\alpha\beta} = \{\iota^{\tau}_{\alpha\beta} \mid \tau \in \Sigma \setminus \{\alpha, \beta\}\}$ that is not precluded by the acceptance of G, corresponds to a tier T in a restriction $\langle T,\, \alpha\beta\rangle$ that exists within $\chi(G) = \Gamma$.

While the interpretation sets multiply at an exponential pace, the size of the formulae grows at a mere linear rate. This way, by establishing a correspondence between interpretation sets and tiers, the exponential-time solution of section 2.3 is made polynomial.

Having supplied the pseudocode, we shall next argue for the equivalence of the two algorithms, slower and faster, in terms of the results that they return.

3.6. Pseudocode

```
**function** grammar′$(X)$ **is**
    $\Sigma \leftarrow \text{alphabet}(X)$
    $K \leftarrow \text{paths}(X)$
    $G \leftarrow \emptyset$
    **for** $\langle \sigma_1,\, Q,\, \sigma_2\rangle \in K$ **do**
        $G \leftarrow G \cup \big\{\textbf{quote}\ \bigvee_{\tau \in \textbf{unquote}\, Q} \tau \in T_{\sigma_1\sigma_2}\big\}$
    **end for**
    $\triangleright$ *Omitted*: add $\top_{\sigma_1\sigma_2}$ for each unattested bigram
    **return** $G$
**end function**

**function** member′$(y, G_1)$ **is**
    $G_2 \leftarrow \text{grammar}'(\{y\})$
    **return** $\neg\text{SAT}\Big(\textbf{quote}\ \big(\bigwedge_{g \in \textbf{unquote}\, G_1} g\big) \;\&\; \big(\neg \bigwedge_{g' \in \textbf{unquote}\, G_2} g'\big)\Big)$
**end function**
```

The **quote** construct, loosely inspired by its eponymous LISP analog, suppresses the evaluation of the expression that follows it; instead, it produces a data structure that mirrors the syntactic makeup of the expression. Correspondingly, **unquote** serves to locally suppress the effect of **quote**, and as such, its argument is evaluated eagerly.

The $\mathrm{paths}(X)$ function returns minimal paths, and only them.

Also, notice the notational peculiarity here, whereby $\bigwedge$ is a n-ary &.

3.7. Correctness

3.7.1. Fragments of a formal demonstration

Conjecture. MTSL$_2$ is a lattice class, as is defined by (Heinz et al., 2012).

In case this conjecture holds, then, as per (Heinz et al., 2012), we should make sure that the algorithm $\langle \mathrm{grammar}', \mathrm{member}' \rangle$ returns minimal-superset extrapolations, and that alone would suffice to warrant its correctness. This means that for a data-set X, it has to construct a grammar G, whose extension $L = \ell(G)$ is minimal along all such $L' \in \mathrm{MTSL}_2$, for which $X \subseteq L'$.

We shall denote the minimal-superset MTSL$_2$ extrapolation as $\mu(X)$. We now intend to demonstrate that $\ell(\mathrm{grammar}'(X)) = \mu(X)$ for any data-set X.

Lemma 7. (*Correctness*) For any data-set X, $\ell(\mathrm{grammar}'(X)) = \mu(X)$.

Proof idea. We do not pursue a proof to completion, but one is possible by way of induction from the base case $\ell(\mathrm{grammar}'(\emptyset)) = \mu(\emptyset) = \emptyset$. It remains to verify the inductive step: if $\ell(\mathrm{grammar}'(X_0)) = \mu(X_0)$, then $\ell(\mathrm{grammar}'(X_0 \cup \{x\})) = \mu(X_0 \cup \{x\})$ as well, for any new data point x.

Essentially, $\mathrm{grammar}'(X)$ is incremental: with each iteration, the **for** loop in the pseudocode will have made the grammar compatible with a greater share of the data. A minor detail is that the algorithm does not traverse the strings from the data-set proper; instead, it does so with their paths: each data point x is substituted with its path set $x \mapsto \{k_1, k_2, \ldots, k_N\}$.

For this reason, one can readily imagine a variation of the algorithm, $\mathrm{grammar}''(k, G_0)$, which modifies a grammar G_0 to accomodate a new path k:

```
function grammar''$(k, G_0)$ is
    $\langle \sigma_1, Q, \sigma_2 \rangle \leftarrow k$
    return $G_0 \cup \left\{ \textbf{quote} \bigvee_{\tau \in \textbf{unquote}\, Q} \tau \in T_{\sigma_1 \sigma_2} \right\}$
end function
```

It would remain to show that including the clause $\bigvee_{\tau \in Q} \tau \in T_{\sigma_1 \sigma_2}$ in G is inevitable. This could be achieved by inquiring if the grammar would still be consistent with the data if we failed to include it. □

3.7.2. Empirical support

By testing, one can also corroborate the belief that the algorithm indeed does produce minimal-superset extrapolations, or, more precisely, grammars that have these supersets as extensions. This undertaking is simplified by obtaining a reference solution, that is, another algorithm, of whose correctness we are certain.

A prudent candidate would be the exponential-time algorithm of section 2.3, since its structure greatly mirrors the very definition of MTSL$_2$ string-sets. Of course, it must be kept in mind that relying on a slow algorithm will reduce the extent to which our testing can be exhaustive.

The code that section 3.9 links to contains the test cases in question.

3.8. From bigrams to *k*-grams

Evidently, a string will never contain a trigram $\alpha\beta\gamma$, unless it also contains the bigrams $\alpha\beta$ and $\beta\gamma$. It would be equivalent to claim that if there is no $\alpha\beta$, then there cannot be an $\alpha\beta\gamma$ either, and analogously, if there is no $\beta\gamma$, then there is just as well no $\alpha\beta\gamma$. More generally, a k-gram is necessarily absent, if *any* of its $(k-1)$-gram substrings is absent.

By means of induction, one can show without difficulty that a k-gram is necessarily absent, if any of its *bigram* substrings $\alpha\beta$ is absent.

This is important for MTSL$_k$ sets in that a k-gram can be restricted on all those tiers, on which there is a symbol to break apart at least one of its bigram substrings.

This lets us restate Lemma 1 for k-grams with arbitrary k.

Lemma 8. (*Reading of a minimal path II*) Suppose that in the data-set X, the substring $\sigma_1 t_1\, \sigma_2 t_2\, \sigma_3 \cdots t_{N-1}\, \sigma_N$ is attested, where t_i are smaller substrings, for which $\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_N \notin \bigcup_i \text{alphabet}(\{t_i\})$. Then it is true that

$$\langle T,\ \sigma_1\sigma_2\sigma_3\cdots\sigma_N\rangle \in \Gamma \ \Rightarrow\ T \cap \Big(\bigcup_i \text{alphabet}\big(\{t_i\}\big)\Big) \neq \emptyset.$$

Here, Γ is a negative-polarity MTSL$_k$ grammar; $X \subseteq \ell(\Gamma)$.

Proof omitted. □

Once some necessary edits have been effected, each related lemma continues to hold. This also applies to the algorithm of section 3.6.

3.9. Source code

The algorithm of Section 3.6, modified to support k-grams for any fixed value of k, has been implemented by the present author in Python 3 and made available online, in the repository `https://github.com/antecedent/k-mtslia`.

The function `learn(X, k=2)` corresponds to the function $\mathrm{grammar}'(X)$ from the pseudocode, and likewise, `scan(y, G)` corresponds to $\mathrm{member}'(y, G)$.

By default, the implementation has an important mismatch with the pseudocode: it automatically surrounds the input strings with beginnning-of-string and end-of-string symbols, `>` and `<`, which correspond the $\rtimes$ and $\ltimes$ that we had already employed earlier. A stark consequence of this is that if L is an MTSL$_k$ set, and $x \in L$, then so are all the substrings $p_{ij} \in L$ of x, but if $\rtimes x \ltimes \in L$, then $\rtimes p_{ij} \ltimes \in L$ does not necessarily follow.

4. Pumping lemma

4.1. On pumping lemmata in general

If additional qualifications are absent, the term *pumping lemma* as such has no mathematically rigorous definition. However, for instance, the pumping lemma for regular sets and the pumping lemma for context-free sets do have one (Hopcroft et al., 2006).

For regular string-sets, the respective lemma states that once a substring long enough, xy, is chosen, then the string xy^nz, for which $n = 1$, remains in the set L, even if n is increased to $2, 3, \ldots$

This becomes clear once one recalls that for each regular string-set, there is a finite state automaton that recognizes it. Its states are necessarily finite in number, therefore, having repeated the substring y a sufficient number of times and still observing $xy^nz \in L$, one can be certain, that it is a loop in the automaton that is being traversed. To put it metaphorically, the automaton loses count.

MTSL$_k$ string-sets are also regular: as classes, MTSL$_k$ ⊂ REG, for each k. Because of this, naturally, the aforementioned pumping lemma also applies to them, but there is also a stronger proposition that holds.

4.2. Pumping lemma for MTSL$_2$ sets

We shall restrict ourselves to $k = 2$.

Lemma 9. (*Pumping lemma for MTSL$_2$ sets*) If L is an MTSL$_2$ language, and xy, yz, $xz \in L$, then necessarily, $xyz \in L$ as well.

Proof idea. Let us denote $\ell(\Gamma) = L$. It is entirely impossible to violate a restriction within the grammar Γ, unless it had already been violated in the constituent parts xy, yz, xz. This is because no new bigrams are introduced, and as for paths, pre-existing ones can only expand.

□

4.3. The PUMP$_3$ class of string-sets

We now see that MTSL$_2$ languages admit a rewriting rule

$$\frac{xy \quad yz \quad xz}{xyz}.$$

If K is an MTSL$_2$ set, then the statement of the lemma implies that no new productions $xyz \notin K$ can be derived. On the other hand, if $K \subset L$, and it is only L that is an MTSL$_2$ set, one can gradually come closer to L.

This means that this pumping lemma is a device of extrapolation of its own, even if it is likely that its extrapolations do not reach their respective MTSL_2 supersets. Let us denote the novel target of extrapolation as the PUMP_3 languages, since the productions are "glued" from 3 pieces.

If, unlike in the provided Python implementation, we do not make use of $\rtimes$ and $\ltimes$ or equivalent symbols to mark the edges of a string, then it follows that whenever $x \in L$, where L is an MTSL_2 language, then all x's substrings $p_{ij} \in L$ as well. For this reason, let us decree that a PUMP_3 extrapolation is defined as the closure of iterated application of the pumping lemma, and the seed set X that one begins with is made up in such way, that if a string is a member, all of its substrings are members too. If that is not the case naturally, it can be enforced artificially.

Empirically, in a preliminary simulation, by generating sets $\{x, y\}$ of 2 strings each, for which $|x|, |y| \leq 8$, and then additionally including all substrings, MTSL_2 and PUMP_3 sets showed an overlap of members of around 95%.

4.4. Phonotactics as a rewriting system

Such a high level of overlap (of which we are, however, only preliminarily certain) between MTSL_2 and PUMP_3 suggests that the acquisition of phonotactics might be modeled using PUMP_3 supersets as well. In fact, this has important advantages over MTSL_2:

- PUMP_3 has a near-trivial definition, which can facilitate the exchange of ideas,
- the mechanism of PUMP_3's operation is highly intuitive, making it neurocognitively plausible,
- one can seemingly derive from PUMP_3 a corresponding probabilistic model with much ease, in contrast to MTSL_2,
- PUMP_3 extrapolations are even easier to interpret that the ones of MTSL_2; for the productions can be readily traced to the input strings that they originate from.

Indeed, it would be even preposterous to consider the PUMP_3 class a significant finding at all, owing to its triviality. Such a finding, however, might be constituted by the fact that this class seems equally apt with MTSL_2 in its domain of application, even if the latter is a much more elaborate class.

It is crucial that rewriting systems might be even more general: they might be suitable for studying other kinds of generalizations that people arrive at from limited samples of observed language use. It is conceivable that similar systems exist for morphology or syntax. It is even possible that in linguistics, they could begin to fill the theoretical chasm between usage and competence.

Findings

An algorithm for extrapolating MTSL$_k$ supersets, which is the main topic of this thesis, has been, as the author hopes, successfully motivated, designed, implemented and had arguments put forward in favor of its correctness.

A pumping lemma for MTSL$_2$ has been derived as a secondary finding. Its ramifications are not clear at the moment, but it is conceivable that it could be this very lemma that would place the PUMP$_3$ class in competition with MTSL$_2$ in its current domain of application.

Conclusion

Phonotactics can be modeled as a very simple rewriting system. Its simplicity notwithstanding, it seems that it is equally powerful as the highly elaborate MTSL model.

A question should be raised, whether the complexity that one observes in commonplace models of theoretical phonology, is really inherent in the phenomenon and not the explanation. In fact, the pioneers of MTSL have already contributed towards this direction by comparing MTSL with such competing models as Optimality Theory.

Since the finding highlighted in this section is merely secondary for this thesis, it would be relevant to corroborate it with follow-up research. If rewriting systems are to bring new insights into the realm of phonotactics research (and of phonology in general), it might be interesting to examine the applicability of the model in other levels of analysis.

References

A. Aksënova, S. Deshmukh. Formal restrictions on multiple tiers. In: G. Jarosz, B. O'Connor, J. Pater (eds.) Proceedings of the Society for Computation in Linguistics (SCiL) 2018, 2018, pp. 64-73.

V. Ambrazas, K. Garšva, A. Girdenis. Dabartinė lietuvių kalbos gramatika. Mokslo and enciklopedijų leidykla, Vilnius, 1994.

A. Aksënova, J. Rawski, T. Graf, J. Heinz. The computational power of harmony. Unpublished manuscript, 2020.

T. B. Brown, B. Mann, N. Ryder, M. Subbiah *et al.* Language models are few-shot learners. Advances in Neural Information Processing Systems, 33, 2020, pp. 1877-1901.

N. Chomsky. Three models for the description of language. IRE Transactions on Information Theory 2 (3), 1956, pp. 113-124.

N. Chomsky. A Review of B. F. Skinner's Verbal Behavior. Language, 35 (1), 1959, pp. 26–58.

E. M. Gold. Language identification in the limit. Information and Control 10 (5), 1967, pp. 447-474.

J. Heath. A grammar of Tamashek (Tuareg of Mali). De Gruyter Mouton, 2005.

J. Heinz, A. Kasprzik, T. Kötzing. Learning in the limit with lattice-structured hypothesis spaces. Theoretical Computer Science 457, 2012, pp. 111-127.

J. E. Hopcroft., R. Motwani, J. D. Ullman. Introduction to Automata Theory, Languages, and Computation (3rd Edition). Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

A. Jardine, J. Heinz. Learning tier-based strictly 2-local languages. Transactions of the Association for Computational Linguistics 4, 2016, pp. 87-98.

A. Jardine, K. McMullin. Efficient learning of tier-based strictly k-local languages. In: F. Drewes, C. Martín-Vide, B. Truthe (eds.) Language and Automata Theory and Applications. 11th International Conference, LATA 2017, Umeå, Sweden, March 6-9, 2017, Proceedings, Springer, Cham, 2017, pp. 64-76.

K. McMullin, A. Aksënova, A. De Santo. Learning phonotactic restrictions on multiple tiers. In: G. Jarosz, M. Nelson, B. O'Connor, J. Pater (eds.) Proceedings of the Society for Computation in Linguistics (SCiL) 2019, pp. 377-378.

A. Ozburn, P. Jurgec. Blocking in Slovenian sibilant harmony: a perception experiment. Poster presented at the Annual Meeting on Phonology, 2015.

C. Yang. The infinite gift: How children learn and unlearn the languages of the world. Simon and Schuster, 2006.