Thomas Graf*

Subregular linguistics: bridging theoretical linguistics and formal grammar

https://doi.org/10.1515/tl-2022-2037
Received October 2, 2020; accepted January 28, 2022

Abstract: Subregular linguistics is a fairly new approach that seeks a deeper understanding of language by combining the rigor of formal grammar with the empirical sophistication of theoretical linguistics. The approach started in phonology but has since branched out to morphology and even syntax, unearthing unexpected parallels between these three domains of language. In this paper, I argue based on these results that subregular linguistics has a lot to offer to both fields. Subregular linguistics may be the ideal conduit for knowledge transfer between these two communities.

Keywords: first-order logic; formal grammar; islands; minimalist grammars; phonology; subregular; syntax

1 Introduction

While *Syntactic Structures* (Chomsky 1957) is mostly remembered as a milestone of theoretical linguistics, it is also one of the inaugural works in formal language theory. Kickstarted by the desire to develop a mathematical understanding of language that is built on theorems and proofs, formal language theory produced many results that now form the foundation of computer science and related fields such as bioinformatics. In this respect, formal language theory has been an undeniable success story. Whether it has accomplished its original goal of illuminating the nature of language, however, is a contentious issue. There is a plethora of results, for instance that syntax is (at least) mildly context-sensitive, but their impact on theoretical linguistics has been limited. This is evidenced by the scarcity of formal language theory in linguistics papers, textbooks, and curricula. There seems to be a marked split, then, between formal language theory as a highly influential subfield of computer science, and its more isolated variant in linguistics, which I will refer to as *formal grammar*.

***Corresponding author: Thomas Graf**, Department of Linguistics, Stony Brook University, Stony Brook, NY, USA, E-mail: mail@thomasgraf.net

This paper presents a specific vision of how formal grammar and theoretical linguistics may fruitfully interact in the future. Even though theoretical linguistics and formal grammar have not always walked in tight lockstep, they actually share a lot of common goals that make knowledge transfer highly desirable. *Subregular linguistics* makes this eminently possible. Subregular linguistics builds on the accomplishments of formal grammar and adopts a lot of its methodology. In particular, its core idea is still that a computational/mathematical understanding of natural language provides new answers to long-standing linguistic issues. But subregular linguistics operates at a finer level of granularity than previous work in formal grammar, and thanks to that it can engage more directly with issues in theoretical linguistics, be they conceptual or empirical in nature. In addition, matters of linguistic analysis that formal grammarians could usually remain agnostic about may have major implications for subregular linguistics. Subregular linguistics thus exhibits the same feedback loop between theory and data that characterizes linguistic inquiry.

As concrete examples of what subregular linguistics brings to the table, I will discuss: how subregular linguistics provides a unified view of phonology, morphology, syntax, and possibly even semantics, allowing us to transfer insights between linguistic domains; how it can be combined with learnability considerations to derive typological restrictions of harmony systems; how it interacts with seemingly minor differences in the analysis of inflectional markers in Noon; and how it can derive the existence of islands directly from the computational nature of movement. A lot has been accomplished since the beginnings of the program 10 years ago, but even more remains to be done. Theoretical linguists and formal grammarians alike are already in an excellent position to contribute to this program.

The paper proceeds as follows. I first discuss key differences between formal grammar and theoretical linguistics that currently impede knowledge transfer between the two (Section 2). Subregular linguistics presents an intermediate position from which one can engage more easily with both fields. Readers without philosophical inclinations may want to skip this section and proceed immediately to Section 3, which surveys the last ten years of subregular linguistics in phonology and morphology. Particular emphasis is put on the notion of tiers and relativized locality in both domains. Section 4 then shows that these ideas—suitably generalized from strings to trees—also hold in syntax. This suggests a great degree of computational similarity between phonology, morphology, and syntax.

2 Differences between formal grammar and theoretical linguistics

Theoretical linguistics and formal grammar are united by common goals, as they both seek a deeper understanding of: the nature of grammatical knowledge; what languages have in common; to what extent languages may differ from each other; why some logically conceivable variants do not occur; how grammatical knowledge can be acquired from limited input; and how it feeds performance, e.g. in parsing. In light of their unity of goals, it is surprising that the two fields rarely interact.

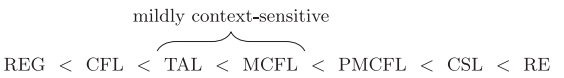
This is a multi-causal issue that cannot be conclusively resolved here, nor does it need to be. I want to highlight only two factors that I consider particularly important: the trade-off between robustness and precision (Section 2.1), and what constitutes the core of a linguistic theory (Section 2.2). On both issues, subregular linguistics assumes a position that lies between the two extreme poles of formal grammar and theoretical linguistics, and this allows it to act as a conduit between the two.

2.1 Robustness versus precision

Scientific research invariably involves a tension between robustness and precision. Robust claims are stable in the presence of new data, and they abstract away from the specifics of any given theory. But the more robust a claim, the less likely that it has anything to say about a specific scientific issue. Precise claims are deeply intertwined with the current theory and thus create a tight coupling between empirical data and the theory. However, precise claims must be revisited with every change to the theory or the empirical landscape, which makes them less likely to stand the test of time. Formal grammarians heavily favor robustness, theoretical linguists heavily favor precision.

As a concrete example, let us consider one of the best-known findings of formal grammar, namely that syntax is at least *mildly context-sensitive* (Huybregts 1984; Shieber 1985). Suppose we regard the syntax of a given language as a set of well-formed strings (rather than trees or graphs). One may ask, then, how much computational power is needed to generate this string set. The Chomsky hierarchy (Chomsky 1956, 1959) distinguishes four types of increasingly complex string languages, which has been subsequently refined and extended in the area between context-free languages (CFLs) and context-sensitive languages (Joshi 1985; Seki et al. 1991):

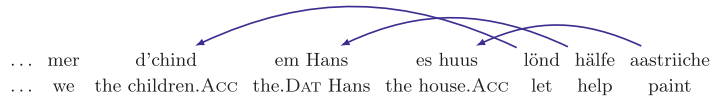
(1) **The revised Chomsky hierarchy**



What exactly all these acronyms stand for, and how these classes of string languages are defined, is largely orthogonal to the issue at hand (the interested reader may consult the appendix for additional background on these classes). The crucial difference is that CFLs can exhibit unbounded nested dependencies but not unbounded crossing dependencies, whereas mildly context-sensitive languages (TALs and MCFLs) can display both.

Shieber (1985, 335) observed that verb-final embedding constructions in Swiss German give rise to unbounded crossing dependencies:

(2)



'... we let the children help Hans paint the house.'

Just as with center embedding in English, there is no principled upper bound on the number of VPs and NPs in (2), so that the number of crossing dependencies can be pushed higher and higher, without ever reaching a finite cutoff point of, say, at most 27 crossing dependencies. Through a sequence of mathematical steps, Shieber reduces this empirical phenomenon to the formal language $a^m b^n c^m d^n$, which contains the strings abcd, aabccd, aaabcccd, abbcdd, abbbcddd, aabbccdd, aaabbcccdd, and so on. This language is not context-free, but it belongs to the class TAL and is thus mildly context-sensitive.

A theoretical linguist might see several shortcomings with this claim. First, it operates with strings instead of trees, so it is at odds with standard assumptions about the nature of syntactic representations. It also does not tell us much about the syntactic mechanisms that produce these patterns, as it is only about the output of grammar rather than grammar itself. In addition, the class of mildly context-sensitive languages is very large and contains infinitely many languages that have little to do with natural languages—for instance the MIX-language (Kanazawa and Salvati 2012; Salvati 2011), which is a formal counterpart to Swiss German where the NPs and VPs can occur in any arbitrary order. Mild context-sensitivity thus is too permissive a lower bound to shed new light on linguistic issues, for instance island effects.

A formal grammarian would concede all three points, but would not agree that they are weaknesses. The focus on strings does not reject the notion that syntax actually operates with trees or graphs, it is driven by the desire to find results that

are independent of representational choices. Even if syntax uses richly structured representations, this cannot lower the complexity of the string language. The mild context-sensitivity of syntax holds irrespective of one's theory of syntax, and this independence of theoretical assumptions makes the result stronger, not weaker. Similarly, claims about the generated language are very powerful because they quantify over infinitely many grammars that generate this language. And while it is true that the class of mildly context-sensitive languages is not a perfect fit for natural language, it does provide a hard requirement that any syntactic formalism has to pass. If a formalism fails to generate at least some mildly context-sensitive string languages, then it can never hope to provide a fully accurate description of syntax. In fact, this is exactly why the publication of Shieber (1985) marked the beginning of the end for GPSG (Gazdar and Pullum 1982; Gazdar et al. 1985), which was built on the assumption that syntax is at most context-free. So mild context-sensitivity may be a loose characterization of natural language, but it still provides meaningful lower bounds for the grammar formalism, learnability, and parsing.

These two opposing positions illustrate the dichotomy of precision and robustness. Our theoretical linguist prizes specificity, they want the work to draw from the full body of existing knowledge and to contribute to this body in a very direct manner with very detailed claims. Crossing dependencies in Dutch, for instance, are not viewed in a vacuum, but are embedded in a web of assumptions about headedness, clause structure, locality, and how these assumptions ought to be encoded in the grammar. The formal grammarian insists on robustness and wants results that are independent of the theoretical formalism *du jour*. There are pros and cons to each position, and both are worth pursuing. Transfer of knowledge between the two, however, is difficult. The specifics of omnivorous number (Nevins 2011), differential object marking (Kalin 2017), or the Williams cycle (Williams 2003), to name but a few, do not obviously narrow down the space of options that formal grammarians tend to work with. On the flip side, the complexity of Chinese number names (Radzinski 1991), case stacking in Old Georgian (Michaelis and Kracht 1997), or relativization in Yoruba (Kobele 2006) does not provide strong guidance in how these phenomena ought to be analyzed in Minimalism, HPSG, LFG, CCG, or other approaches.

Even when formal grammarians look beyond strings, the chasm does not always disappear. Hunter and Frank (2021) suggest that multiple wh-movement could establish a computational lower bound on what kind of tree structures a grammar formalism must be able to generate—a tree-based follow-up to Shieber's argument against context-free formalisms. But this view of multiple wh-

movement is completely independent of the issues that make up the bulk of Bošković (2002) and related works on wh-movement, namely its licensing conditions and what factors give rise to exceptions. And the central piece of data needed by Hunter and Frank (2021)—whether any language has movement of an unbounded number of wh-phrases such that each wh-phrase targets a different landing site—seems to be missing from the linguistic literature on multiple wh-movement.

Once we start our discussion of subregular linguistics, we will see how it attempts to shrink this chasm by increasing precision without sacrificing too much robustness. A lot of subregular work is very similar in flavor to Shieber (1985), but since it focuses on much weaker classes the findings are more dependent on how exactly one analyzes the data. As we will see, this is particularly true in the case of subregular syntax (Section 4), but it also holds for phonology and morphology.

2.2 What is a linguistic theory?

Besides the issue of precision and robustness, formal grammar and theoretical linguistics also differ in what they consider the core of a linguistic theory, and this has immediate repercussions for the status of empirical analyses.

To a formal grammarian, the heart of a linguistic theory is the grammatical machinery. The meaningful claims about language are what can be derived directly from the formalism. A linguistic analysis of an empirical phenomenon in formalism F provides evidence that F is linguistically adequate, but it does not directly inform the central questions of complexity, parsing, and learnability. These issues are addressed by studying F from a mathematically informed perspective. To understand the formalism is to understand language.

This is why the formal grammar literature is filled to the brim with proofs that show i) how to convert between different formalisms (e.g. the translations between TAG, CCG, Head Grammar, and Linear Indexed Grammar in Joshi et al. (1991), or the equivalence proof for Minimalist grammars and Multiple Context-Free Grammars in Michaelis (2001)), or ii) how adding new operations or removing existing ones does or does not change the formalism's power (e.g. the finding in Kobele (2010) that the power of Minimalist grammars drops to that of context-free grammars if one forbids remnant movement, whereas Graf (2012b) showed that the power of Minimalist grammars with remnant movement does not increase if one adds sideward movement, affix hopping, or any other movement operation

proposed in the literature), or iii) how a well-known formalism can be reconceptualized in a completely different manner without changing any of its properties (for instance, Rogers (2003) reanalyzes TAG as a finite-state system that operates with three-dimensional trees). This is the reason why formal grammarians want linguistic theories to be defined in a rigorous manner. Rigorous definitions enable mathematical investigation, which deepens our understanding of the formalism, which tells us what the linguistic theory is truly saying about language.

But there are multiple ways in which a linguistic theory is arguably more than its formalism. Consider first the case of Peters and Ritchie (1971, 1973), where it was shown that the version of Transformational Grammar developed in Chomsky (1965) is an unrestricted formalism that can generate any computable string language. Consequently, there can be no efficient parsing or learning algorithms that work for every Transformational Grammar, which undermines its status as a plausible theory of syntax. But Peters and Ritchie's theorem hinges on a particularly liberal use of deletion rules that, albeit technically allowed, does not match the way the formalism was actually used by linguists at the time.[1] This raises the question whether a linguistic theory is defined by its formalism or by how the formalism is used. As we will see in Section 3.1, the seminal result that phonology is regular—which is a precondition for considering subregular phonology—firmly adopts the latter stance.

Another salient issue is the contrast between the formalism and the concrete analyses that have been articulated using this formalism. This dichotomy is best illustrated by HPSG, which is known to be an unrestricted formalism like Transformational Grammar. The unrestricted nature of HPSG is not regarded as a major

1 The core insight of Peters and Ritchie (1973) is that transformations bridge the gap between context-sensitive grammars and unrestricted rewrite systems (see also the appendix for additional background). An unrestricted rewrite system is one where rewrite rules have the form $\alpha \rightarrow \beta \mid \phi \_ \psi$—if a string α appears between strings ϕ and ψ, then it is rewritten as string β. Context-sensitive grammars use the same format but with the extra stipulation that β must not be shorter than α. In other words, context-sensitive grammars cannot shrink a string. Now suppose that we take an unrestricted rewrite system G and replace every rule $\alpha \rightarrow \beta \mid \phi \_ \psi$ where β is n symbols shorter than α with a padded counterpart $\alpha \rightarrow \beta B^n \mid \phi \_ \psi$, where B is some new non-terminal symbol that is not used by G. The result is a modified version of G where no rewrite step ever shrinks the string, which means that this is a context-sensitive grammar. We can then use a transformation to delete all reflexes of the useless padding symbol B, leaving us with exactly the strings generated by G. The full construction requires a few extra steps, in particular if one wants the modified counterpart of G to be a context-free grammar, not just a context-sensitive one. But the central trick is to introduce new, linguistically vacuous symbols whose sole purpose is to push the shrinking abilities of unrestricted rewrite systems out of the grammar and into the transformational component.

issue by its community because the formalism is treated as a convenient description language for what really matters, the collection of analyses of linguistic phenomena. Unless a formal analysis of HPSG incorporates the specifics of raising, passive, binding, NPI-licensing, and so on, it is not really making claims about HPSG as a linguistic theory.[2] From this vantage point, a linguistic theory is not just its formalism, nor how this formalism is used, but the collection of analyses that are built on top of the formalism.

There is no universally accepted answer as to which of these stances one should adopt, leaving us with three distinct notions of linguistic theory: the formalism as defined, the formalism as used, and the formalism as a collection of analyses. Formal grammar prioritizes the first because the others are harder to study and yield less robust results. Usage of a formalism is difficult to determine and may well change over time. Analyses are constantly being revised and replaced, making them a moving target. Distinct analyses may actually contradict each other, yielding a mathematically inconsistent theory. And the sheer number of interacting parts may be beyond what formal grammar can handle at this point, just like one cannot use classical mechanics to effectively model hundreds of leaves falling at once.[3]

Subregular linguistics is a noteworthy departure in this respect. Rather than the linguistic formalism, it is the linguistic analyses that matter most to the mathematical inquiry. We will see this soon during the discussion of subregular phonology in Section 3.2. Its claims are not about SPE, Government Phonology, OT, or Harmonic Grammar, but about phonology under a specific mode of analysis that may borrow from these theories. Similarly, subregular syntax is not a formalization of GB, Minimalism, or TAG, but a mathematical investigation of syntax that builds directly on the insights that have grown out of these syntactic theories as well as what formal grammarians have learned about them.

---

2 Interestingly, Kasper et al. (1995) argue that many HPSG analyses can be expressed in a more restricted formalism that generates at most mildly context-sensitive string languages. HPSG has also been widely used in computational applications, which suggests that parsing performance is not an insurmountable challenge in practice.

3 Of course this does not mean that things have to stay this way. Government-and-binding theory was notoriously difficult to analyze from a mathematical perspective. For the longest time, Stabler (1992) was the only in-depth analysis of GB, and it relied on an automatic theorem prover to keep track of the many tightly interwoven components of GB. But a few years later, Rogers (1998) showed how the model-theoretic ideas of Blackburn (1993) and Backofen et al. (1995) could be expanded to yield a remarkably intuitive analysis of the formal properties of GB. These techniques are now widely used by formal grammarians, and one can only imagine how future methodological revolutions may broaden the scope of inquiry for formal grammar.

3 A subregular program for linguistics

Subregular linguistics starts from the observation that key domains of language seem to be at most regular (Section 3.1), that is to say, they fall into the class REG at the very bottom of the Chomsky hierarchy in (1). This prompts the question if language might be even more limited. There is robust evidence that first-order logic provides a tighter, albeit still loose upper bound for phonology, morphology, and syntax (Sections 3.2 and 4.1). Subregular linguistics tries to drill even deeper than that, identifying very limited yet linguistically natural classes that cover a wide range of phenomena. This has worked tremendously well for phonology in the last 10 years, with most phenomena belonging to the formal classes strictly local (SL), tier-based strictly local (TSL), or small extensions thereof (Section 3.3). More recently, these classes have also been shown to play a role in morphology and morphosemantics (Section 3.4). It seems, then, that subregular linguistics provides a notion of complexity that cuts across the linguistic domains and allows us to identify deep commonalities between them. As we will see in Section 4, this even includes syntax.

3.1 Regularity of phonology and morphology

While the best-known work in formal grammar is about syntax, other language modules have also been studied for a long time. There is now a broad consensus that phonology and morphology are at most regular, which means that they fall into the weakest class of the Chomsky hierarchy. Let us briefly consider how the upper bounds on phonology and morphology were established.

Kaplan and Kay (1994) observed that phonologists working with SPE did not actually make use of the full power of the formalism. In principle, SPE rewrite rules can apply to the part of the string that they have just modified. Consider for example a rule that inserts *ab* between *a* and *b*. This rule could turn the underlying representation *ab* into *aabb*, which is then rewritten as *aaabbb*, which becomes *aaaabbbb*, and so on, until the grammar decides to no longer apply the rule. But rules of this kind do not occur in the literature. Intuitively, SPE rewrite rules consume their input in the sense that the trigger for the application of a rule must consist of more than just the material that was inserted by the very same rule. Based on this insight about how SPE is used, rather than how it is defined, Kaplan and Kay (1994) were able to model SPE as a formal device called a *finite-state transducer*. In combination with other plausible assumptions about the nature of

the phonological lexicon, this entails that the set of surface forms generated by any given SPE grammar is a regular string language. This is a claim about SPE, not phonology, but the theorem about the former lets us draw inferences about the latter. While SPE has been largely abandoned by phonologists, this was prompted by SPE's lack of explanatory power, not because there were some phonological phenomena that it was unable to describe. If SPE is a descriptively adequate theory of phonology, and if SPE generates at most regular string languages, then phonology generates string languages that are at most regular.

A very similar argument also holds for morphology. Finite-state transducers have been used by computational linguists for a long time to construct efficient systems for morphological analysis. This work runs under the banner of *finite-state morphology* and *two-level morphology* (Koskenniemi 1983; Karttunen et al. 1992; Karttunen and Beesley 2005; Roark and Sproat 2007), and it has achieved an impressive degree of descriptive coverage. Since the computational machinery being used is limited to regular languages, its descriptive adequacy suggests that morphology is at most regular.[4]

These findings about phonology and morphology form the backbone of many practical applications, but from a linguistic perspective there is room for improvement. Just like we saw in Section 2.1 that mild context-sensitivity is a loose lower bound for syntax, regularity is a loose upper bound for phonology and morphology. It does entail computational efficiency, but it is too permissive regarding what kind of linguistic patterns are (im)possible. Consider a language where a phonological word is well-formed iff the number of vowels in the word exceeds the number of consonants. The resulting string language is not regular, so regularity as an upper bound on phonology explains why no attested language displays such a constraint. There are constraints, though, that are equally implausible yet the resulting string language is regular. Suppose, for instance, that each well-formed string must receive a point total that is an even number and that the point total is calculated as follows: each vowel in a word is worth 1 point, whereas each consonant is worth 2 points if it is adjacent to a vowel and worth 3

4 One clear-cut counterexample to this is unbounded reduplication, which is also discussed in the appendix as an example of a mildly context-sensitive phenomenon. It would be more accurate to say, then, that morphology *modulo* unbounded reduplication is regular. Reduplication is an active research area, see for instance Dolatian and Heinz (2020) for a recent approach that incorporates insights from subregular morphology.

It has also been argued that the *anti-X-missile* construction, as in *anti-air-missile*, yields the string language anti^n-*X*-missile^n (Halle 1973). Since this string language is context-free, morphology cannot be regular. However, there are several conceptual and empirical problems with this argument that greatly diminish its force compared to unbounded reduplication (see Langendoen 1981; Carden 1983).

points otherwise. Intuitively, the second example seems even more complex than the first one, yet the second stays within the realm of regular string languages and the first one does not. A tight formal characterization of phonology and morphology ought to be able to rule out both constraints; next we will see how this has been done for phonology.

3.2 Subregular phonology

If regularity is too loose an upper bound for phonology, then we have to find a more restrictive subclass of the regular string languages that provides a better fit. But the Chomsky hierarchy in (1) lists the regular languages as the weakest class, so one might think that looking for even more restricted characterizations of phonology would mean charting unexplored territory. Fortunately, this is not the case. Subclasses of the regular languages have been defined and studied for over 50 years now (McNaughton 1974; McNaughton and Papert 1971; Pin 1997; Ruiz et al. 1998; Schützenberger 1965; Simon 1975).

For example, the class of regular languages contains the proper subclass of *star-free* string languages, which are exactly those string languages that can be defined in first-order logic (with relations for successor and precedence and a predicate $\sigma(x)$ for each alphabet symbol σ).[5] The first-order formula below defines a string language that displays a simple form of long-distance sibilant harmony—s and ʃ cannot occur in the same string, no matter how far apart they are. This is expressed by quantifying over all nodes x and y in the string to stipulate that if one of them is labeled s, the other one must not be labeled ʃ, and if one of them is labeled ʃ, then the other one must not be labeled s.

(3) **First-order formula for long-distance sibilant harmony**
$\forall x, y[(\mathrm{s}(x) \rightarrow \neg\int(y)) \wedge (\int(x) \rightarrow \neg\mathrm{s}(y))]$

Note that the formula treats the harmony as a phonotactic constraint on surface forms, there is no actual rewriting of a non-harmonic underlying representation into a harmonic surface-form. This is not an ontological claim about whether harmonies are surface-constraints or rewriting processes, it is merely a formal description of one specific way of construing the phenomenon.

---

5 Originally, the star-free string languages were defined as those string sets that can be defined by regular expressions with all Boolean operations and concatenation, but not the Kleene star. They are also equivalent to the counter-free languages, and the string languages with an aperiodic syntactic monoid. But the first-order perspective of star-free languages is the most accessible.

Let us consider intervocalic voicing as another example. For the sake of convenience, we first define two new predicates *vowel* and *voiceless*, with the assumption that the only vowels are a, i, and u, and that the only voiceless segments are s and f.

(4) **Two helper predicates for phonology**

$$\text{vowel}(x) \Leftrightarrow \text{a}(x) \vee \text{i}(x) \vee \text{u}(x)$$
$$\text{voiceless}(x) \Leftrightarrow \text{s}(x) \vee \text{f}(x)$$

We then use those predicates to ensure that if a node n is flanked by nodes labeled with vowels, then n is not a voiceless segment. The formula makes use of the successor relation $\lhd$ to express the position of the voiceless segment relative to the vowels: $x \lhd y$ holds iff y is immediately to the right of x.

(5) **First-order formula for intervocalic voicing**

$$\forall x, y, z[x \lhd y \wedge y \lhd z \wedge \text{vowel}(x) \wedge \text{vowel}(z) \rightarrow \neg\text{voiceless}(y)]$$

A minor variation of (5) yields unbounded tone plateauing, which does not allow any low tone L to occur between two high tones H, no matter how far away those two H are. In this case, the precedence relation $\lhd^+$ is used instead of the successor relation $\lhd$. Precedence, in contrast to the successor relation, holds across arbitrary distances.

(6) **First-order formula for unbounded tone plateauing**

$$\forall x, y, z[x \lhd^+ y \wedge y \lhd^+ z \wedge \text{H}(x) \wedge \text{H}(z) \rightarrow \neg\text{L}(y)]$$

With the predicates and relations introduced so far, one can write first-order formulas for many other phenomena, e.g. word-final devoicing (construed as the phonotactic constraint that the last segment must not be a voiced consonant). The basic building blocks used above are very simple, yet they can be recombined in elaborate ways to describe some of the most complicated aspects of phonology. All evidence points towards first-order logic covering a large part of phonology, perhaps even all of it; and since the first-order definable string languages (i.e. the star-free languages) are a proper subclass of the regular string languages, we can possibly lower the complexity bound for phonology from regular to star-free.

While certainly an abstract claim, limiting phonology to star-free languages makes clear typological predictions and can guide empirical inquiry. If phonology is at most star-free, this solves the issue with counting constraints that we encountered at the end of Section 3.1. Neither one of the two counting constraints listed there is star-free. The star-free languages thus manage to provide a more restricted characterization of phonology without losing empirical ground relative to regular languages. As discussed in Graf (2010), the only potential counterexample in the literature is primary stress assignment in Cairene Arabic and Creek.

Stress in Cairene Arabic has been described in the literature (Hayes 1995) as using an elsewhere condition such that if primary stress cannot be assigned using a more specific rule, it falls on the rightmost syllable that is an even number of steps away from the left edge of the word. This is not a star-free language for the same reason that our example language requiring an even point total is not star-free. Star-free languages are incapable of this kind of *modulo* counting. Primary stress in Cairene Arabic as usually described in the literature thus is not star-free. Interestingly, Becker (2019) argues that the pattern has been mischaracterized, and Becker's new description of the phenomenon is much simpler, making it star-free.[6] This is a nice example of how the search for more restricted classes can push certain patterns to the forefront of theoretical linguistics. It is still unclear whether Creek has been similarly misanalyzed. With only one putative counterexample, which has not been thoroughly vetted by linguists yet, it is likely that the star-free languages constitute a viable upper bound on phonology.

But just like regular languages allow for all kinds of unnatural constraints, so do the star-free languages. One example of this is a language where sibilant harmony is limited to the first and last segment of the string, while all sibilants in between can be non-harmonic. This *first-last harmony* is unattested.

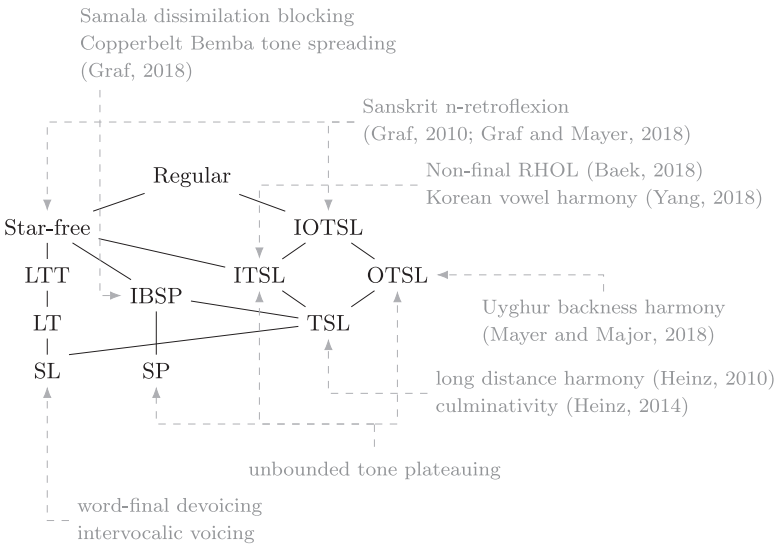
(7) **First-order formulas for first-last harmony**

$$\begin{gathered}\text{first}(x) \Leftrightarrow \neg\, \exists\, y[y \lhd x] \\ \text{last}(x) \Leftrightarrow \neg\, \exists\, y[x \lhd y] \\ \forall x, y, z[\text{first}(x) \wedge \text{last}(y) \rightarrow (\text{s}(x) \rightarrow \neg \int(y)) \wedge (\int(x) \rightarrow \neg \text{s}(y))]\end{gathered}$$

Rather than incrementally lowering the upper bound on phonology, Heinz (2009, 2010) decided to go with a bottom-up approach to show that most phonological phenomena fall into remarkably simple classes. This increases robustness: although there may be a few outliers, if a large part of phonology belongs to some class C, then C provides insights into core parts of phonology even if C does not encompass all of phonology. Heinz (2009, 2010) was followed by a flurry of work that now makes up the program of *subregular phonology*. A number of new classes have been defined in the process (De Santo and Graf 2019; Graf 2017, 2018; Graf and Mayer 2018; Heinz et al. 2011; Rogers et al. 2010), prompted by various phonological phenomena.

6 Becker posits two distinct stress patterns that are conditioned by the word's morphological makeup and the weight of the last three syllables. In monomorphemic words, stress is penultimate by default, but ultimate if the last syllable is heavy. In polymorphemic words, stress is always assigned among the last three syllables depending on several conditions, most importantly which of those syllables are heavy. All of these properties are very local, which makes them fairly easy to define in first-order logic.

(8) **Hierarchy of subregular classes, with example phenomena**[7,8]



The figure above reveals a sprawling subregular landscape that can easily intimidate the uninitiated. But the majority of phonological phenomena fall in just two classes, SL (*strictly local*) and TSL (*tier-based strictly local*). Other classes like SP, ITSL, OTSL, IOTSL, and IBSP cover relatively few phenomena that aren't also in SL or TSL. As SL is a special case of TSL, I will primarily cover the latter. TSL will also be of central importance for the discussion of syntax in Section 4.

---

7 The full names for these acronyms are as follows:
- SL: strictly local
- SP: strictly piecewise
- LT: locally testable
- LTT: locally threshold-testable
- TSL: tier-based strictly local
- ITSL: input tier-based strictly local
- OTSL: output tier-based strictly local
- IOTSL: input output tier-based strictly local
- IBSP: interval-based strictly piecewise

8 Graf and Mayer (2018) contains the claim that IOTSL is a subclass of the star-free languages, but this remains to be shown. The alleged proof mentioned in the paper turned out to be erroneous, and it is currently an open question how the two classes are related. Dmitrii Zelenskii (p.c.) points out that the language $(abab)^n$ (with $n > 0$ even) is not star-free but can be described as the intersection of an SL language and an OTSL language. If OTSL is not star-free, then IOTSL cannot be either. However, unless OTSL can be shown to be closed under intersection with SL languages (which is unlikely), it is unclear how much Zelenskii's finding tells us about the power of OTSL by itself.

3.3 Tier-based strictly local string languages (TSL)

TSL is short for *tier-based strictly local*. The class was first defined in Heinz et al. (2011), building on ideas of autosegmental phonology (Goldsmith 1976). A tier projection mechanism masks out all segments in the string that are irrelevant for the phenomenon being modeled, and the remaining structure must not contain any forbidden substructures of a fixed size. Intuitively, TSL captures a notion of relativized locality.

Consider the requirement that every word has exactly one syllable with primary stress (*σ́*), also known as culminativity (Heinz 2014). The corresponding string language contains *σσ́σσ*, but not *σσσσ*, *σσ́σσ́*, or *σσ́σ́σ́*. Formally, we may denote the string language as $\sigma^{\star}\acute{\sigma}\sigma^{\star}$, where $\sigma^{\star}$ means "0 or more unstressed syllables". This language can be shown to be TSL. First, we construct a tier that contains every *σ́*—this is the same as masking out all *σ* in the string.

(9) **Tier projection for culminativity**

Tier:	*σ́*	*Tier:*		*Tier:*	*σ́*	*σ́*	*σ́*
	|				|	|	|
Word:	*σ σ́ σ σ*	*Word:*	**σ σ σ σ*	*Word:*	**σ σ́ σ σ́ σ σ́*		

We now have to identify a finite number of local constraints that separate the well-formed tier from the ill-formed ones. In TSL, local constraints are encoded as forbidden *n*-grams. A tier is ill-formed iff it contains at least one instance of at least one forbidden *n*-gram. Hence we can block the rightmost tier *σ́σ́σ́* in (9) by making *σ́σ́* a forbidden *n*-gram. Given our tier projection mechanism, every string with at least two primary stressed syllables will necessarily yield a tier that contains at least one instance of *σ́σ́*. This holds no matter how far apart the two stressed syllables are in the actual string because the material between them has been masked out by the tier projection. Combining a tier projection function that masks out all *σ* with a local constraint against *σ́σ́* thus rules out all strings that contain two or more syllables with primary stress.

This still leaves us with strings that contain no primary stress at all. To accommodate these cases, the special symbols ⋊ and ⋉ are used to mark the left and right edge of the tier. We can then make ⋊⋉ a forbidden bigram, which immediately rules out all empty tiers. Since a tier can only be empty if the string did not contain any syllables with primary stress, we have correctly ruled out all strings that do not contain at least one primary stress.

(10) **Tier projection for culminativity, with explicit edge symbols**

Tier:	⋊	*σ́*	⋉	*Tier:*	⋊	⋉
		|				
Word:		*σ σ́ σ σ*		*Word:*	**σ σ σ σ*	

Combining the forbidden bigrams ⋊⋉ and *σ́σ́* leaves *σ́* as the only well-formed tier, so that only strings with exactly one primary stress are well-formed. This shows that the language $\sigma^* \acute{\sigma} \sigma^*$ is TSL. In fact, it is *tier-based strictly 2-local* (TSL-2) because none of the forbidden *n*-grams exceed a length of 2.

This is all there is to TSL, it is a very simple class of string languages.

(11) **Definition of TSL over strings**[9]
Given some alphabet Σ, a *tier alphabet* is some fixed subset *T* of Σ. For every finite string *s* over Σ and node *i* in *s*, *T*(*i*) is true iff *i* has a label in *T*. We define the *tier successor* relation $\lhd_T$ as follows ($\lhd^+$ denotes precedence):

$$x \lhd_T y \Leftrightarrow T(x) \wedge T(y) \wedge x \lhd^+ y \wedge \neg \exists z [T(z) \wedge x \lhd^+ z \wedge z \lhd^+ y]$$

The *T-tier projected from s* consists of all nodes of *s* with a label in *T*, ordered by $\lhd_T$. We stipulate that the first node on the *T*-tier is the $\lhd_T$-successor of the left edge marker ⋊. The last node on the *T*-tier has the right edge marker ⋉ as its $\lhd_T$-successor.
A string language *L* is *tier-based strictly k-local* (TSL-*k*) iff there is a tier alphabet *T* and finite set *G* of *n*-grams with $n \leq k$ such that a string *s* is a member of *L* iff no *n*-gram listed in *G* is a substring of the *T*-tier projected from *s*. It is *tier-based strictly local* (TSL) iff it is TSL-*k* for some $k \geq 0$.

The simplicity of TSL also holds from a cognitive perspective. It isn't even necessary to literally construct a tier. TSL allows for a simple online recognition process: i) memorize the most recently seen symbols $s_1 s_2 \cdots s_n$ that belong to the tier alphabet *T*, and check this sequence against the finite list of forbidden *n*-grams; ii) if the sequence is forbidden, reject the string, otherwise read in the next symbol *s*; iii) if *s* is in the tier alphabet *T*, update the memorized sequence to $s_2 \cdots s_n s$, and proceed from step i. The only potential challenge is memorizing long sequences in cases where *n* is large, but since most TSL phenomena in phonology are TSL-2 or TSL-3, *n* rarely exceeds 3. This underscores how simple TSL phenomena are from a computational perspective: the human brain needs to furnish only very simple data structures and inference mechanisms to support TSL dependencies in language (cf. Rogers and Pullum 2011).

In linguistic terms TSL captures a notion of relativized locality that ignores irrelevant segments in the string. In cases where all segments matter, one winds up with the more limited class SL, which is short for *strictly local*. SL is the special case

9 In contrast to the original definition in Heinz et al. (2011), the one given here does not require all *n*-grams to have the same length. This is merely a matter of convenience and has no formal consequences. I also use first-order logic to directly define tier successor over strings—the standard definition of TSL instead uses a function that deletes all non-tier symbols from any given string, leaving us with just its tier.

of TSL where the tier alphabet is identical to the full alphabet. This is the same as saying that there is no tier and the forbidden n-grams are checked against the string itself.

(12) **Definition of SL over strings**
A string language L over alphabet Σ is *strictly k-local* (SL-k) iff it is TSL-k with tier alphabet $T = \Sigma$. It is *strictly local* (SL) iff it is SL-k for some $k \geq 0$.

SL provides a formal model of local phenomena, where the affected segments cannot be arbitrarily far away from each other. For example, the intervocalic voicing example from Section 3.2 is SL-3: we forbid all trigrams of the form V_1SV_2, with $V_1, V_2 \in \{a, i, u\}$ and $S \in \{s, f\}$. Some cases of vowel harmony are also SL, but TSL tends to provide a more succinct description. This is best illustrated with a toy example. Suppose we have a language with a strict CV syllable template and two vowels a and i that can never occur in the same word. This language is SL: we forbid all trigrams of the form aCi and iCa, where C is a consonant. So if there are k distinct consonants, then there are $2k$ trigrams that must be forbidden. From the TSL perspective, on the other hand, we simply project a vowel tier on which we ban only two bigrams, which are ai and ia. Not only is the TSL grammar much smaller than the SL grammar, its size is independent of the number of consonants in the language. This shows that the subregular perspective is not limited to complexity, as both succinctness and naturalness can be incorporated as insightful criteria.

It is worth keeping in mind that TSL involves a maximally simple notion of tiers. There is only a single tier, and whether a segment is projected depends only on whether it is part of the tier alphabet. If one *a* projects then all *a*s have to project. One cannot distinguish between different instances of *a* based on their position, their local context, or what else will be projected on the tier. The classes ITSL, OTSL, and IOTSL in (8) are extensions of TSL that use more elaborate tier projection mechanisms. It is striking though that these classes are rarely encountered—most phonological phenomena stay within the realm of TSL.

If there is a general bias towards simple phenomena that are at most TSL, the fact that the first-last harmony from (7) is unattested is less surprising because it is not a TSL phenomenon. In first-last harmony, the left string in (13) would be well-formed, whereas the right one would be ill-formed. But since both have exactly the same sibilant tier, there is no way a TSL language could contain one but not the other.

(13) **Identical sibilant tiers for distinct strings**

Tier:	⋊	s		ʃ		⋉	*Tier:*	⋊	s		ʃ ⋉
		|		|					|	|
Word:		s	a	ʃ	a		*Word:*	*	s	a	ʃ

While there are few attested cases where the very simple tier projection mechanisms of TSL is insufficient, the limitation to just one tier is more severe. For one thing, if two distinct phenomena are each TSL, that does not mean that a language that has both is also TSL. The two phenomena may require wildly different tiers. The multi-tier-based strictly local languages (MTSL) are exactly those that are the intersection of two or more TSL languages, which is a formal way of saying that the language is TSL with multiple tiers. While MTSL is a much larger class than TSL, it also points towards phonology being remarkably restricted. Aksënova and Deshmukh (2018) observe that if a harmony system requires multiple tiers, the tier alphabets are either completely disjoint or stand in a subset relation. We do not find phenomena that require incomparable tier alphabets, e.g. $T_1 = \{a, b\}$ and $T_2 = \{b, c\}$, where the tier alphabets have some overlap yet neither subsumes the other. But if we consider all logically conceivable MTSL languages, the vast majority of them (often over 95%) cannot be described without such incomparable tiers. Forbidding incomparable tiers drastically narrows down the space of relevant MTSL languages, and it is striking that natural language seems to obey this requirement.

The reason for this may lie in learnability. There are several formal learnability results for classes of subregular languages (Heinz et al. 2012; Jardine and McMullin 2017). Due to the high level of mathematical abstraction, these findings do not provide a concrete model for language acquisition, but they nonetheless tell us something about the relative difficulty of the learning task and what kind of generalizations are needed to identify the target language from a reasonably small data sample. In the case at hand, McMullin et al. (2019) present a learning algorithm for MTSL, but crucially it only works reliably if the target language does not require incomparable tier alphabets. A bias towards simple learning naturally weeds out these MTSL languages, leaving us with the attested types of MTSL languages where tier alphabets are either disjoint or stand in a subset relation.

This is a nice example of one particular way the subregular approach combines formal grammar and theoretical linguistics. Just like Shieber (1985) presented a complexity result for syntax that is largely independent of any syntactic assumptions, the subregular perspective allows us to measure the complexity of various phonological phenomena without committing to a specific formalism like SPE or OT. However, the definition of the formal classes does take inspiration from linguistic proposals, such as autosegmental phonology, and it is guided by the desire to account for a wide range of phenomena unearthed by linguists. By analyzing phenomena from a subregular perspective, we arrive at a rough complexity landscape that can be combined with learnability theorems to provide specific, often novel predictions about what kinds of patterns should be expected to exist.

3.4 String subregularity beyond phonotactics

The discussion in the preceding section has centered around phonology as a generator of well-formed surface strings. In linguistic terminology, this equates phonology with phonotactics. But the reach of subregular linguistics extends far beyond that, and in many ways the findings get even more interesting.

Subregular transductions (Chandlee 2014; Chandlee and Heinz 2018; Heinz and Lai 2013; Mohri 1996, 1997, 2000; Schützenberger 1977) provide a formal model of phonology as a system that maps underlying representations to surface forms. Even with this shift in perspective, phonology is still remarkably simple. For example, the class of *input strictly local* functions (ISL) is very limited. ISL is to string transductions as SL is to string languages. It provides a formal model of maximally local processes, processes that can be described by an SPE rule of the form $a \rightarrow b \mid \alpha \_ \beta$ where α and β are n-grams.[10] Like SL, ISL covers a lot of empirical ground, but additional classes are needed to accommodate processes that can apply across great distances. In particular suprasegmental phenomena seem to require more power (Jardine 2016), which is in line with the common belief that suprasegmental phonology is qualitatively different from segmental phonology.

All work so far suggests that the limited nature of phonology in terms of phonotactics is mirrored by its weakness in terms of mappings from underlying representations to surface forms. Once again this allows for typological predictions. Attested phenomena like locally bounded metathesis and progressive harmony are much simpler than unattested ones like Sour Grapes and Majority Rules (Finley 2008), both of which look very natural from the perspective of OT.

But the subregular perspective is not limited to phonology, it has also been applied to morphology. Again we find that TSL seems to play a central role for morphotactics (Aksënova et al. 2016), and that ISL captures many rewriting processes in morphology (Chandlee 2017). And similar to the earlier discussion about whether some vowel harmony patterns are SL or TSL, we find cases where linguistic considerations give rise to different complexity results. Moradi et al. (2019) consider the case of the inflection affixes *wum* and *fum* in Noon (Niger Congo), which micromorphology (Stump 2017) reanalyzes as combinations of the affixes *w-*, *f-*, and *-um*. Even though the distribution of *wum* and *fum* is TSL, reanalyzing them as *w-um* and *f-um* requires the more powerful class ITSL, where the tier projection can also take a symbol's local context into account. This creates an interesting

10 The connection between transductions and specific types of SPE rewrite rules does not mean that a transduction-based view of phonology is tantamount to an SPE-based view of phonology. Transductions are a more abstract concept and can also be related to other formalisms like OT (Frank and Satta 1998; Jäger 2002; Karttunen 1998).

coupling of formal grammar and theoretical linguistics: arguments against ITSL become arguments against micromorphology, and arguments for micromorphology become arguments for ITSL.

It is curious that not only do both phonology and morphology happen to be subregular, they also cluster largely around the same classes, in particular TSL. Graf (2019) even argues that TSL plays a role in morphosemantics and might explain the limited range of meanings for monomorphemic quantifiers across languages.[11] Based on this surprising degree of parallelism, I propose the following hypothesis:

(14) **Cognitive parallelism hypothesis**
Distinct language modules have the same complexity.

This is actually a meta-hypothesis as it leaves open just what the relevant level of complexity is. Since larger classes subsume weaker ones, picking a large class makes the hypothesis more robust, but also less precise. At this point, it is a fairly safe bet that both phonotactics and morphotactics are star-free (but see fn. 4). A less robust but more precise hypothesis is that distinct language modules are "almost TSL": most phenomena fit in TSL when viewed as string constraints, and only a few attested outliers require natural extensions of TSL.

If one is to seriously pursue this hypothesis, though, one must first address the elephant in the room, which is syntax. As discussed in Section 2.1, syntax is at least mildly context-sensitive. Therefore it goes far beyond the boundary of regular string languages. But lack of regularity entails lack of subregularity. Given what we already know about syntax, it is impossible that syntax is star-free or TSL. This is a serious problem, but it can be resolved in a manner that makes the cognitive parallelism hypothesis even stronger.

4 From strings to trees: subregular syntax

When viewed as a generator of string languages, syntax is at least mildly context-sensitive. But linguists have insisted for a long time that syntax is about trees, not strings. If we follow this credo and look at the complexity of tree languages instead of string languages, a subregular view of syntax may be more feasible. In fact, there

11 If one adopts the semantic automata approach (van Benthem 1986), every quantifier can be associated with a string language over the alphabet 0 and 1. For example, *every* corresponds to the set of all strings that contain no 0, *no* is the set of all strings that only contain 0, and *some* is the set of all strings that contain at least one 1. All of these *quantifier languages* are TSL. On the other hand, the string languages of quantifiers like *an even number* or *a third of* are not TSL, and there seems to be no language that expresses these quantifiers with a single monomorphemic word.

is overwhelming evidence that syntax is subregular in this sense (Section 4.1), and its most fundamental aspects may even be TSL (Section 4.2). TSL not only captures the basics of movement, it also produces island effects "for free" (Section 4.3).

4.1 Syntax and first-order logic

There cannot be a subregular view of syntax over trees unless syntax can be shown to be at most regular over trees. That is to say, if one considers the set of all well-formed syntactic trees of any given natural language, is this set guaranteed to form a regular tree language? For the longest time, the answer to that was believed to be negative. The string yield of a regular tree language is a context-free string language, so the mildly context-sensitive string language of Swiss German cannot be the string yield of some regular tree language. But this claim hinges on the assumption that the tree language is a set of phrase structure trees, the string yield of which is obtained by reading the leaf nodes from left to right. If one uses an abstract tree format instead that relies on a more complex string yield function, then regular tree languages can produce string languages that go beyond context-free (Koller and Kuhlmann 2011; Mönnich 1999; Morawietz 2003). For instance, Minimalist grammars (MGs; Stabler 1997, 2011) are a formalization of Minimalist syntax that can generate very complex string languages (Harkema 2001; Michaelis 2001). They even cover PMCFLs (Kobele 2006), which lie outside the mildly context-sensitive region. But for each MG, one can look at its set of well-formed derivation trees, and this set is regular (Kobele et al. 2007; Michaelis 2001). In fact, it can be defined in first-order logic where $\lhd$ and $\lhd^+$ are interpreted as the mother-of relation and proper dominance, respectively (Graf 2012a). Recall from Section 3.2 that first-order logic with $\lhd$ and $\lhd^+$ over strings defines the subregular class of star-free languages, so this is evidence that syntax is subregular over trees.

The subregular nature of MGs is not limited to the basic version defined in Stabler (1997), it also holds if one adds head movement (Stabler 2003), sidewards movement (Stabler 2006), copy movement (Kobele 2006), across-the-board movement (Kobele 2008; Torr and Stabler 2016), clustering (Gärtner and Michaelis 2010), phases (Stabler 2011), Late Merge (Graf 2014a), multiple implementations of adjunction (Fowlie 2013; Frey and Gärtner 2002; Graf 2014b; Hunter 2011), case theory (Laszakovits 2018), Distributed Morphology (Kobele 2011), morphosyntactic agreement (Ermolaeva 2018), and a large number of constraints (Gärtner and Michaelis 2007; Graf 2011; Kobele 2011; Michaelis 2004, 2009), including even transderivational ones (Graf 2013). Virtually all operations and constraints that Minimalists have proposed can be defined in terms of first-order logic over trees,

making it very unlikely that syntax contains phenomena that absolutely cannot be described in first-order logic.[12]

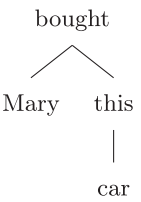
But just like the star-free languages still allow for too many unnatural patterns, first-order logic can express all kinds of unnatural conditions. For any attested constraint, first-order logic can also enforce its symmetric opposite, e.g. that licensees must c-command their licensors. Attested constraints can be combined via disjunction or implication, most likely yielding highly unnatural patterns. Once again a tighter characterization would be preferable.

4.2 The TSL nature of movement

For phonology, SL and TSL provide a good balance between restrictiveness and empirical coverage, so by the cognitive parallelism hypothesis we should expect to find them in syntax, too. This is indeed the case, although the details depend a great deal on what kind of tree representation one chooses. I will use a format here that combines linguistic intuitiveness with subregular simplicity.

Suppose that we represent syntactic structures in terms of dependency trees. The sentence *Mary bought this car* would have the dependency tree below.

(15)



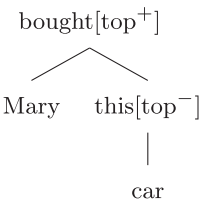
Clearly the string is not obtained by reading the leafs from left to right, as this would only yield *Mary car*. Instead, the string yield function has to understand how

12 The only challenge to this claim seems to come from phenomena at the syntax-semantics interface. For example, the literature contains some constraints like Rule I (Heim 1998; Reinhart 1996) where one must determine if two sentences have identical meaning. In general, this is undecidable, but it may be the case that the specific cases where Rule I is invoked allow for a simpler solution in terms of first-order logic. A related problem arises with ellipsis, which is sometimes analyzed as deletion under semantic identity, but perhaps alternative accounts such as Kobele (2015) could prove helpful here. Principle B has also been argued to display a high degree of computational complexity (Ristad 1993), but Graf and Abner (2012) contend that if one carefully distinguishes between syntactic binding and discourse binding, Principle B is suspended in exactly those cases that would make it undefinable in first-order logic. Finally, Principle C is also problematic, but it is unclear whether it should be considered a syntactic principle or a purely semantic one.

arguments are linearized with respect to heads, i.e. whether the subtree below a node represents a string that goes to the left or the right of that node. It does this by paying attention to the order of daughters: the left daughter goes to the left of the mother, the right daughter to the right.

Now suppose that we are not interested in *Mary bought this car* but in the topicalized version *this car, Mary bought*. Taking a hint from Minimalist syntax, we may view this as displacement of *this car* to a position to the left of *bought*. Let us also follow Minimalism in assuming that this displacement is triggered by some kind of feature checking, perhaps between a negative topicalization feature top^- on the head of the moving phrase and a positive top^+ on the head that provides the landing site. Then we can make a minor change to the tree in (15) so that it now encodes this different string.

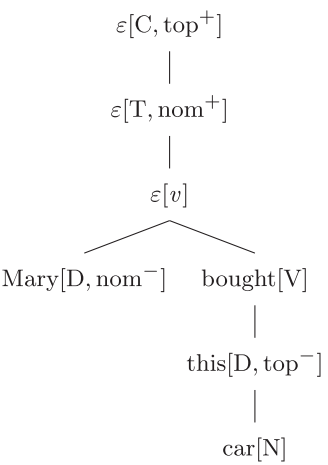
(16)



Due to the presence of the top-features, the string yield function now linearizes *this car* to the left of *bought* and its other arguments, yielding the desired *this car, Mary bought*.

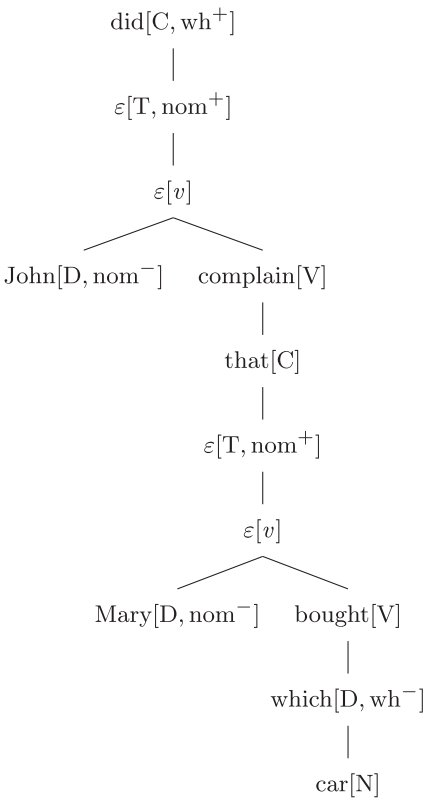
We could also add the commonly posited heads *v*, T and C (assumed here to be empty, which is denoted by *ε*). Then the object actually undergoes topicalization to Spec,CP and the subject moves from Spec,*v*P to Spec,TP (subject movement will be encoded by the feature nom, but this term is just a mnemonic without any relation to actual case checking). And of course we may add more features, e.g. category features.

(17)



A slightly more complex case is shown below, which yields the question *which car did John complain that Mary bought*.

(18)



Note that instead of strings, one could also obtain phrase structure trees from these dependency trees. The dependency trees contain all the information required for that, it just happens to be encoded in a less familiar manner.

How complex, then, is syntax if we view it through the lens of dependency trees? The answer depends on what aspect of syntax one focuses on. Let us start with movement. I have not said much about what kind of movement configurations should be licit. Recent work on subregular syntax adopts a system that generalizes a specific version of MGs (Graf et al. 2016):

(19) **A simple movement system**

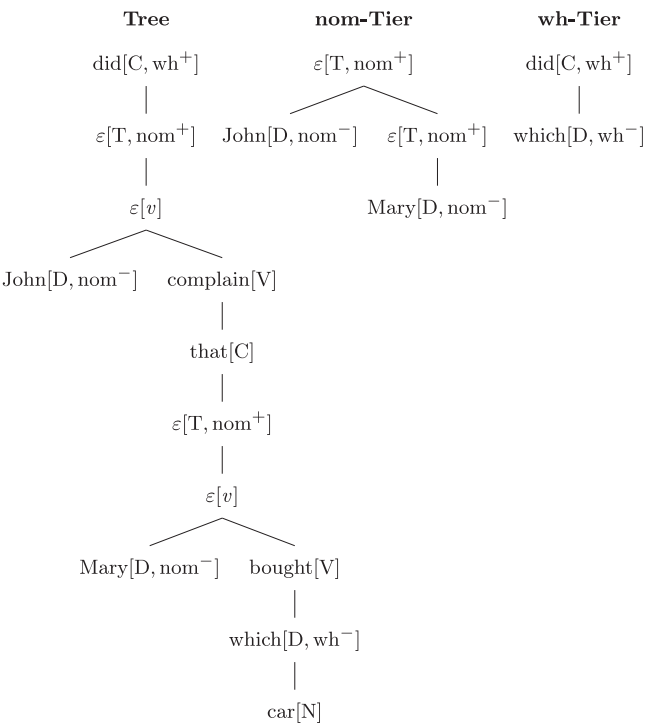
a. Every lexical item has 0 or more negative movement features (in contrast to standard MGs, these features are unordered with respect to each other).

b. If *l* is a lexical item with negative movement feature f^-, then its f-match is the lowest lexical item that properly dominates *l* and carries f^+.
c. Every lexical item carrying f^- must have exactly one f-match.
d. Every lexical item carrying f^+ is an f-match for exactly one lexical item.

These conditions ensure that both positive and negative features get checked and that no head with an f^- can move across a landing site with f^+. Both requirements are met by our example tree in (17). First, *this* has exactly one top-match, which is the C-head, and *Mary* has the T-head as its nom-match. This takes care of (19c). In addition, there is no other top-match for the C-head and no other nom-match for the T-head, so (19d) is also satisfied.

The constraints in (19) are easy to verify by hand, and they are also very simple from a subregular perspective. It turns out that they can be described by a tree analogue of TSL. Suppose that for every pair of movement features f^+ and f^- we construct a tier that contains only those nodes in the tree that carry one of these features. As in the string case, we mask out all other nodes, but we keep the relative order between the remaining nodes (for strings, this was given by precedence, in the trees it is given by dominance).

(20)



Once we have these tiers, the constraints (19c) and (19d) can be rephrased as conditions on what string of daughters a given node in a tier may have.

(21) **Constraints on f-movement tiers**

a. Every node carrying f^- must be the daughter of a node that carries f^+.
b. Every node carrying f^+ has exactly one node among its daughters that carries f^-.

It should be easy to see that these constraints are maximally local as they only involve looking at a node and either its mother or all its daughters on the tier. We are hence dealing with a constraint that is strictly local over tree tiers, or simply *tree-TSL*.

Let us flesh out this intuition in formal terms by lifting TSL-2 from strings to trees (extending TSL-3 and higher to trees requires too many definitions to merit discussion here; for the same reason, I omit defining the order between siblings on a tier).

(22) **Definition of TSL-2 over trees**

Given some alphabet Σ, a *tree tier alphabet* is some fixed subset T of Σ. For every tree t over Σ and node i in t, $T(i)$ is true iff i has a label in T. We define the *tier mother-of* relation $\lhd_T$ as follows ($\lhd^+$ denotes proper dominance):

$$x \lhd_T y \Leftrightarrow T(x) \wedge T(y) \wedge x \lhd^+ y \wedge \neg \exists z [T(z) \wedge x \lhd^+ z \wedge z \lhd^+ y]$$

The *T-tier projected from t* consists of all nodes s with a label in T, ordered by $\lhd_T$. If a node i has no mother on the T-tier, then it is considered a $\lhd_T$-daughter of the root marker $\rtimes$. If i has no daughter on the T-tier, then it is considered a $\lhd_T$-mother of the bottom marker $\ltimes$.
An SL-2 function associates $\rtimes$ and every symbol in Σ with a (possibly empty) set of licit daughter strings. A tree language L is *tier-based strictly 2-local* (TSL-2) iff there is a tier alphabet T and an SL-2 function γ such that a tree t is a member of L iff it holds for every node i of T (including $\rtimes$) that the daughter string of i is a member of $\gamma(i)$.

This definition is a faithful generalization of TSL-2 over strings: strings can be regarded as unary trees, and the class of TSL-2 tree languages is exactly the class of unary tree sets that encode TSL-2 string languages. Even the wording sticks very close to (11), with the only difference being the switch from n-grams to SL-functions to accommodate that tree tiers allow for a node to have multiple successors.

Using the more formal format of (22), the constraints in (21) correspond to a specific SL-2 function.

(23) **SL-2 function for f-tiers**

We use L^- to denote any node that carries f^-, l^- for any node that lacks f^- (including the special leaf marker $\ltimes$), and l^{-*} for 0 or more iterations of l^-. Then the SL-2 function γ distinguishes two cases:

a. If node i carries f^+, then $\gamma(i) = \bar{l}^* L^- \bar{l}^*$ ("exactly one mover under each landing site").[13]

b. Otherwise, $\gamma(i) = \bar{l}^*$ ("movers appear only under landing sites").

Let us check this against the nom-tier in (20). First we check the special node $\rtimes$, which forms the hidden root of the tier. Its daughter string consists of $\epsilon[\mathrm{T}, \mathrm{nom}^+]$. Since $\rtimes$ does not carry f^+, its daughter string must not contain any movers, which is indeed the case here as the T-head only carries nom^+, not nom^-. As for the T-head itself, its daughter string must follow the pattern $\bar{l}^* L^- \bar{l}^*$. This is the case. Its daughter string is John $[\mathrm{D}, \mathrm{nom}^-]$ $\epsilon[\mathrm{T}, \mathrm{nom}^+]$, which contains exactly one node with a negative movement feature. Without the nom^- feature on *John*, this would have been an ill-formed daughter string, rendering the whole tier and hence the whole dependency tree illicit. The reader may verify on their own that the remaining two nodes also have well-formed daughter strings, so that the whole nom-tier is well-formed.

With this system, it is also very easy to handle selection. A verb like, say, *introduce* may take a DP as its complement (i.e. right daughter) and another DP as its specifier (i.e. left daughter). But it could also take a PP as its complement instead. Hence the set of licit daughter strings for *introduce* contains D D and D P, where D and P are shorthand for any lexical item with the corresponding category feature. The same simple strategy can be used with any lexical item that takes at least one argument. No tier is needed, so selection is SL-2 over dependency trees. This supports the cognitive parallelism hypothesis: the classes SL and TSL that play a major role for phonotactics and morphotactics also make an appearance in syntax, where selection is SL-2 and each type of movement is TSL-2.

4.3 Deriving the existence of islands

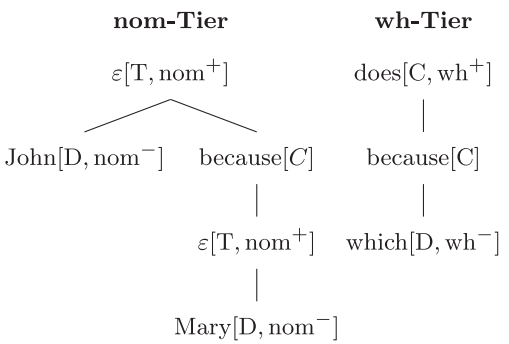
The TSL-nature of movement also addresses a long-standing puzzle of generative syntax, namely why movement should be subject to island constraints. For most approaches, including Minimalism, there is no intrinsic reason why languages should display island effects. Island constraints have to be stipulated, and the formalism would actually be simpler if they did not exist. The subregular perspective switches this dynamic: it would be surprising if languages did not display any island effects.

---

13 The string language $\bar{l}^* L^- \bar{l}^*$ already made an appearance in Section 3.3 as $\sigma^* \acute{\sigma} \sigma^*$. This was the formlization of culminativity, which requires every word to have exactly one primary stress. Not only, then, do phonology and syntax both involve TSL to a significant extent, the very same formal constraints appear in vastly different domains.

Take the case of an adjunct island violation, as in **Which car does John complain because Mary bought*. This sentence has almost the same dependency tree as the one in (20). Without further assumptions, it will also have the same tiers and should thus be licit. But remember that tiers are constructed based on a tier alphabet T. We said that for each movement type f, T contains all and only those lexical items that carry f^+ or f^-. There is no reason, though, why this should be the case. Suppose that T also contains because[C]. Then the tiers for our illicit example sentence look slightly different.

(24)



The nom-tier is still well-formed according to the SL-2 function in (23)—since *because* does not carry a positive movement feature, its daughter string must match l^{-*}. The tier for *wh*-movement, on the other hand, is illicit. This is because the daughter string of does[C, wh^+] consists only of because[C], which does not match $l^{-*}L^{-}l^{-*}$. Projecting *because* has broken up the local licensing relation between the mover and its landing site.[14]

14 The reader may wonder how this account can be reconciled with the common assumption that movers undergo successive-cyclic movement. In *which car did John complain that Mary bought*, the wh-mover *which car* should first move to the embedded Spec,CP provided by *that*. But since movement in this system is triggered by positive and negative features, *that* must carry wh^+, which means that it would be present on the wh-tier. However, this would leave the C-head *did*, which also carries wh^+, without any wh^- among its tier daughters, so that the wh-tier would be ill-formed. Paradoxically, then, island effects should arise with any C-head, not just *because*.

Suppose instead that intermediate wh-movement does not involve any extra features and is triggered by the wh^+ feature on the head that furnishes the final landing site. A mover traveling from its base position to some Spec,XP must stop in every Spec,CP along the way, but it is not "lured" to those positions by a specific feature. This reinterpretation is fully compatible with how successive-cyclic movement is invoked in the literature. It also avoids some undesirable complications, e.g. the split between Q-features and wh-features in Chomsky (1995) to explain why declarative embedded clauses would have a feature that attracts wh-phrases. But if one wants to stay as close as possible to current Minimalism, one could also say that these intermediate positions contain a special edge feature that marks them as landing sites but never triggers tier projection.

As long as one can reliably identify the heads of adjuncts, then, adjunct islands are the result of a system that projects all adjunct heads onto all movement tiers. If one annotates adjuncts with a dedicated feature as in Frey and Gärtner (2002), this is a standard TSL tier projection. Without such a feature, the tier projection needs to be able to look at the node's local context (mother, siblings, daughters) to determine if it is an argument or an adjunct. This would be a tree-analogue of the string class ITSL, which is also needed for phonotactics.

The same logic can be extended to other island constraints, but again they may require additional features or an ITSL-style tier projection. The Complex NP constraint, for instance, is tantamount to projecting all nouns that select a C-head. Either the tree representation explicitly lists a node's subcategorization frame, or we have to inspect its daughters.

Depending on one's assumptions about feature annotations, then, there are two alternative stories as to why the existence of island effects is not that puzzling: they are either an unsurprising by-product of the TSL-nature of movement, or they are a syntactic counterpart to ITSL phenomena in phonotactics. This view also explains why certain logically conceivable types of islands never arise. For instance, one could imagine a gang-up adjunct island effect where movers can escape from a single adjunct but not from two, in which case *Which car does John complain because Mary bought* would be well-formed but *Which car does John complain because Sue was annoyed because Mary bought* would not be. This kind of pattern is neither TSL-2 nor ITSL-2, which sets it apart from attested island constraints.[15]

That said, the TSL account of islands is far from exhaustive. We have gotten rid of the puzzle why island effects exist, only to be faced with the puzzle why island effects are so systematic. Any given language could decide to project only *because* and *rumor*, or to project the C-head of an adjunct clause only if it starts with a vowel, or to enforce island constraints only on some random subset of all tiers. Nor is there a good reason as to why we do not find languages with argument islands instead of adjunct islands, or anti-complex NP islands. While these issues could be put aside as a matter of substantive universals, there are also island constraints

15 Gang-up effects do seem to arise with parasitic gaps, which can be licensed across one island but not two. However, parasitic gaps cannot be handled with TSL to begin with, although they are still subregular and should fit into a specific class proposed in Graf and De Santo (2019) as an upper bound on syntactic dependencies (the class of dependencies that are recognizable by a sensing tree automaton). Crucially, this class allows for gang-up effects. The fact that parasitic gaps are not TSL thus suggests that the mechanisms behind them differ fundamentally from movement, and hence we would expect them to display certain behaviors like gang-up effects that do not arise with movement.

that are not TSL at all. This includes the Specifier Island Constraint, which is ITSL but not TSL, and the Coordinate Structure Constraint (which isn't even ITSL).

Interestingly, cases where the TSL perspective over trees fails may fall out from a different subregular perspective that focuses on dominance and c-command (cf. Frank and Vijay-Shanker 2001). This perspective was developed to handle phenomena like NPI-licensing and binding (Graf and Shafiei 2019; Shafiei and Graf 2020). The approach bears some resemblance to functional uncertainty in LFG (Kaplan and Zaenen 1988) as well as recent proposals under the banner of *one-dimensional syntax* or *precedence syntax* (Brody 2019): For every node in the tree, one looks at the node's set of c-commanders (or, alternatively, its set of ancestors) and orders them from structurally lowest to highest. The result is a string that lists the node's c-commanders in the order they are encountered as one moves from said node to the root of the tree. The string is then checked against specific well-formedness constraints, which turn out to be subregular.

Consider the binding of pronominals, reduced to the distributional problem whether the sentence contains any suitable antecedents for each pronominal. Then Principle A corresponds to the constraint that a reflexive's string of c-commanders must contain a ϕ-compatible DP somewhere to the left of the first T-head, which ensures the presence of a suitable c-commanding antecedent within the smallest TP containing the reflexive. This is the case in the well-formed *[C John$_i$ T$_{fin}$ [seems [t$_i$ to t$_i$ adore himself]]]*, where the string of c-commanders for *himself* is *John adore to seems T$_{fin}$ C*, with the ϕ-compatible *John* preceding the first finite T-head.[16] Now compare this to the pronominal *aapaṇ* in Marathi, which also has to be syntactically bound but must not be locally bound (Kiparsky 2002). Hence a sentence containing *aapaṇ* may be well-formed only if the string of c-commanders for *aapaṇ* has a compatible DP after the first finite T-head (rather than before). Going back to our discussion of TSL phonotactics in Section 3.3, we can compare the subregular complexity of these two string constraints. Principle A is TSL: after projecting all ϕ-compatible DPs and all finite T-heads, the resulting tier must not start with T. The non-local binding of *aapaṇ*, on the other hand, requires the more powerful class OTSL that we briefly encountered in the hierarchy in (8). TSL is insufficient because it cannot distinguish a well-formed tier, which contains at least one ϕ-compatible DP but this DP may occur anywhere in the tier as long as it is not to the left of the first T, from an ill-formed tier that contains only T-heads and no ϕ-compatible DPs at all. Even so, OTSL is still subregular and thus binding—when construed as a distributional constraint—reduces to subregular conditions

16 In this approach, it is assumed that heads are structurally more prominent c-commanders than their specifiers. This is why our example string of c-commanders orders *John* before *adore*, indicating that the latter is structurally more prominent than the former.

on strings of c-commanders. In this very specific sense, syntax is subregular even over strings, assuming one uses the right kind of strings.[17]

As can be seen from these few examples, subregular syntax is still in flux, with several approaches homing in on different notions of subregularity over specific types of syntactic structure. Tree tiers have proven very useful for movement, whereas c-command strings and ancestor strings are better suited to syntactic licensing conditions. The boundaries between the two are still blurry, though, and subject to ongoing exploration. Crucially, this exploration is driven by empirical considerations and familiar linguistic concerns. Familiar phenomena are given new analyses that unify them with seemingly unrelated constructions. Graf (2022), for example, shows that the TSL-view of movement not only predicts the existence of island effects, it also explains the existence of multiple wh-movement (cf. Graf and Kostyszyn 2021), wh-agreement, the *that*-trace effect, and the anti-*that*-trace effect because all of these are the result of projecting specific tiers that are restricted via SL-functions. The computational space carved out by TSL movement already contains all these movement-related phenomena. A lot remains to be done, of course, but the existing work shows how subregularity refines the standard notions of complexity in formal grammar and thus makes them a highly sensitive tool for linguistic analysis.

5 Conclusion

I have argued that subregular linguistics offers a unique synthesis of theoretical linguistics and formal grammar. A lot has been accomplished in the last 10 years, much more than could be fit into a single paper. Yet there is still much more to learn. Since subregular linguistics is built on rigorous mathematical ideas, formal grammarians can contribute by growing our understanding of subregular languages and transductions (i.e. rewriting mechanisms). In particular the area of subregular tree transductions needs a lot more work before it can be insightfully applied to movement. At the same time, subregular linguistics engages very directly with empirical data. So far this has mostly been limited to reanalyzing existing data from the literature, but it need not stop there. Subregular concepts like TSL are simple enough that theoretical linguists can incorporate them directly

17 This string-based view of syntax contains echoes of the findings in Kornai (1985) that syntax is subregular over strings if one abandons the competence-performance distinction and assumes a finite cut-off point for syntactic constructions. The findings of Graf and Shafiei (2019) and Shafiei and Graf (2020) show that syntax exhibits string-based subregularity even with the competence-performance distinction as long as the strings encode c-command relations or dominance relations instead of linear precedence.

into their daily work. Linguistic theory has long been a useful guide in what constitutes interesting problems to work on, and subregular linguistics can play a similar role. A phenomenon that seems unremarkable could actually provide key insights into subregular complexity, and the other way round.

Acknowledgement: I thank the editors for their very detailed comments and their tremendous patience during the revision process.
Research funding: The work reported in this paper was supported by the National Science Foundation under Grant No. BCS-1845344.

Appendix: String languages in the (refined) Chomsky hierarchy

For the interested reader, I include a brief overview of the classes listed in (1) here. The presentation is deliberately informal and emphasizes general intuitions rather than precise definitions.

REG (regular). In mathematical terms, a string language is regular iff it can be recognized by a finite-state automaton. There are many additional characterizations, a.o. definability in monadic second-order logic with successor, having a Myhill-Nerode relation of finite index, or being a projection of a strictly 2-local string language. Each one of these characterizations has unique advantages—automata provide a way of distinguishing well-formed from ill-formed strings, the Myhill-Nerode characterization makes it easy to show that a given language is not regular, monadic second-order logic offers a very succinct, constraint-based description, and so on. Thanks to the large number of equivalent perspectives, there are many different intuitions for what it means to be regular, but the following is perhaps the most accessible: each string in a regular string language can be correctly built from left-to-right while only memorizing a finitely bounded amount of information about the string built so far. For example, the regular language $(aa)^*$ only contains strings over *a* whose length is even, e.g. *aa* or *aaaa*, but not *aaa*. While building such a string, one does not need to store the exact number of *a*s already built, it suffices to keep track of whether the length of the string built so far is odd or even, and that is a finite amount of information that does not scale with the actual length of the string.

CFL (context-free). A string language is context-free iff it can be generated by a context-free grammar (the mathematical counterpart to the familiar phrase structure grammars). Context-free string languages can exhibit an unbounded number of nested dependencies. The palindrome language, for instance, contains strings that read the same from left to right as from right to left, including *aa*, *abba*,

aaaa, *ababa*, *abbba*, *abbcbba*, and so on. The string *abbab*, on the other hand, is not part of the palindrome language because the first symbol does not match the last one (nor does the second symbol match the last but one). Each string of the palindrome language consists of multiple nested dependencies: in a string of length n, the i-th symbol must match the $(n - i + 1)$-th symbol. Hence in abbba, which has length 5, we have a dependency between the two *b*s in positions 2 and $5 - 2 + 1 = 4$, and this dependency in turn is nested inside another dependency between the two *a*s in positions 1 and $5 - 1 + 1 = 5$.

Again there are many equivalent characterizations of the context-free languages, but on an intuitive level the central idea behind context-freeness is the ability to take an assembled object, split it in two pieces, and then wrap those pieces around some other object of bounded size. Hence abbba can be understood as taking aa and breaking it in two pieces a and a, which are then wrapped around bb to yield abba. Then abba is again split into ab and ba and wrapped around b to yield abbba. This view of context-freeness is useful because it can be generalized to yield the mildly context-sensitive classes TAL and MCFL.

TAL (tree-adjoining languages). TALs were originally defined as the string languages that can be generated by Tree Adjoining Grammars, but once again many equivalent definitions have been found, for instance in terms of embedded push-down automata (Vijay-Shanker 1987). One may think of TALs as a generalization of CFLs where a string can be broken into three pieces instead of just two. This makes it possible to generate not only unbounded nested dependencies, but also unbounded crossing dependencies. As a simple example, consider a language where the only available symbols are *a*, *b*, and *c*, and every string exhibits a limited kind of unbounded reduplication: if a string contains a *c*, then the part before the first *c* in the string must be the output of reduplication. This means that a well-formed string containing *c* must be of the form uucv, where *u* and *v* are arbitrary strings over *a*, *b*, and *c*. Hence abbabbcb would be well-formed, but abaacb would be ill-formed because abaa does not consist of two identical halves. This kind of reduplication can establish unbounded crossing dependencies because we have two copies, abb and abb, and the *i*-th symbol of the first copy must match the *i*-th symbol of the second copy. The bigger the copies, the larger the number of crossing dependencies between them.

A well-formed string like abbabbcb is easy to build as long as we have the ability to break assembled strings into three separate pieces: We start with aacb and split it into the three pieces *a*, *a*, and *cb*. We then take those three pieces and wrap them around *b* and *b* to yield ababcb. This is once more broken up into three pieces—*ab*, *ab*, and *cb*—which are again wrapped around *b* and *b* to yield the desired abbabbcb.

MCFL (multiple context-free languages). The class MCFL generalizes TAL in two ways. First, there is no longer a universal upper bound on the number of pieces

one may have to juggle when building a string. For one language, one may need the ability to split strings into four pieces, for another language the number might be twelve, or a million. Second, pieces can be shuffled around before reassembly, so that the second piece may end up following the fourth piece. This means that MCFLs may exhibit much more complicated crossing dependencies than TALs. However, MCFLs are still fairly similar to TALs, which is why both are considered to be part of the mildly context-sensitive region.

PMCFL (parallel multiple context-free languages). PMCFLs can be regarded as MCFLs with recursive copying. Consider, for instance, the English *schm-X* construction, where a single noun like *rules* may be partially copied to yield *rules schmules*. If this operation could apply recursively, then one could feed it its own output *rules schmules* and obtain *[rules schmules] [schmules schmules]*, and from that one could build *[rules schmules schmules schmules] [schmules schmules schmules schmules]*. Now if one considers only the set of strings that this operation can build from *rules*, one gets the PMCFL *rules schmules*$^{2n-1}$ ($n \geq 0$), i.e. *rules* followed by $2^0 - 1 = 1 - 1 = 0$ instances of *schmules*, or $2^1 - 1 = 2 - 1 = 1$ instance, or $2^2 - 1 = 4 - 1 = 3$ instances, or $2^3 - 1 = 8 - 1 = 7$, or $2^4 - 1 = 16 - 1 = 15$, and so on. Notice how the length of strings grows exponentially, which is a common property of PMCFLs. An MCFL must have an upper bound k such that if one looks at some string s of length n, the shortest string that is longer than s has at most length $n + k$. PMCFLs may lack this *constant growth property*.

CSL (context-sensitive) and RE (recursively enumerable). CSL and RE are such powerful classes that it is hard to give them intuitive characterizations. RE is the class of all computable string languages. In linguistic terms, one may think of RE as the class of string languages that can be generated by unrestricted SPE-style rewrite rules of the form $\alpha \rightarrow \beta \mid \phi \_ \psi$ (as discussed in Section 3.1, this does not mean that SPE as used by linguists generates RE languages). Here α, β, ϕ and ψ are arbitrary finite sequences of symbols. Essentially, then, anything can be rewritten as anything, conditioned by arbitrary finite contexts. CSL is the special case of RE where β never contains fewer symbols than α, so the output of a rewrite rule cannot be shorter than its input. Both classes allow for dependencies very much unlike what we find in natural language, e.g. that the length of a well-formed string must be a prime number.

References

Aksënova, Alëna & Sanket Deshmukh. 2018. Formal restrictions on multiple tiers. *SCiL* 1. 64–73.

Aksënova, Alëna, Thomas Graf & Sedigheh Moradi. 2016. Morphotactics as tier-based strictly local dependencies. *SIGMORPHON* 14. 121–130. Available at: https://www.aclweb.org/anthology/W/W16/W16-2019.pdf.

Backofen, Rolf, James Rogers & K. Vijay-Shanker. 1995. A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language and Information* 4. 5–39.
Becker, Michael. 2019. *Egyptian Arabic stress is local*. Stony Brook University Ms.
van Benthem, Johan. 1986. Semantic automata. In *Essays in logical semantics*, 151–176. Dordrecht: Reidel.
Blackburn, Patrick. 1993. Modal logic and attribute value structures. In Maarten de Rijke (ed.), *Diamonds and defaults*, 19–65. Norwell, MA: Kluwer.
Bošković, Željko. 2002. On multiple wh-fronting. *Linguistic Inquiry* 33. 351–383.
Brody, Michael. 2019. Some biolinguistic remarks. *Acta Linguistica Academica* 66. 335–348.
Carden, Guy. 1983. The non-finite = state-ness of the word formation component. *Linguistic Inquiry* 14. 537–541.
Chandlee, Jane. 2014. *Strictly local phonological processes*. University of Delaware Doctoral Dissertation. Available at: http://udspace.udel.edu/handle/19716/13374.
Chandlee, Jane. 2017. Computational locality in morphological maps. *Morphology* 27. 599–641.
Chandlee, Jane & Jeffrey Heinz. 2018. Strict locality and phonological maps. *Linguistic Inquiry* 49. 23–60.
Chomsky, Noam. 1956. Three models for the description of language. *IEEE Transactions on Information Theory* 2. 113–124.
Chomsky, Noam. 1957. *Syntactic structures*. The Hague: Mouton.
Chomsky, Noam. 1959. On certain formal properties of grammars. *Information and Control* 2. 137–167.
Chomsky, Noam. 1965. *Aspects of the theory of syntax*. Cambridge, MA: MIT Press.
Chomsky, Noam. 1995. Categories and transformations. In *The Minimalist program*, 219–394. Cambridge, MA: MIT Press.
De Santo, Aniello & Thomas Graf. 2019. Structure sensitive tier projection: Applications and formal properties. In Raffaella Bernardi, Gregory Kobele & Sylvain Pogodalla (eds.), *Formal grammar*, 35–50. Heidelberg: Springer.
Dolatian, Hossep & Jeffrey Heinz. 2020. Computing and classifying reduplication with 2-way finite-state transducers. *Journal of Language Modelling* 8. 179–250.
Ermolaeva, Marina. 2018. Morphological agreement in Minimalist grammars. In Annie Foret, Reinhard Muskens & Sylvain Pogodalla (eds.), *Formal grammar*, 20–36. Heidelberg: Springer.
Finley, Sara. 2008. *Formal and cognitive restrictions on vowel harmony*. Johns Hopkins University Doctoral Dissertation.
Fowlie, Meaghan. 2013. Order and optionality: Minimalist grammars with adjunction. *MoL* 13. 12–20.
Frank, Robert & Giorgio Satta. 1998. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics* 24. 307–315.
Frank, Robert & K. Vijay-Shanker. 2001. Primitive c-command. *Syntax* 4. 164–204.
Frey, Werner & Hans-Martin Gärtner. 2002. On the treatment of scrambling and adjunction in Minimalist grammars. In Gerhard Jäger, Paola Monachesi, Gerald Penn & Shuly Wintner (eds.), *Proceedings of the conference on formal grammar*, 41–52.
Gärtner, Hans-Martin & Jens Michaelis. 2007. Some remarks on locality conditions and Minimalist grammars. In Uli Sauerland & Hans-Martin Gärtner (eds.), *Interfaces + recursion = language?*, 161–196. Berlin: Mouton de Gruyter.

Gärtner, Hans-Martin & Jens Michaelis. 2010. On the treatment of multiple-wh-interrogatives in Minimalist grammars. In Thomas Hanneforth & Gisbert Fanselow (eds.), *Language and logos*, 339–366. Berlin: Akademie Verlag.

Gazdar, Gerald, Ewan Klein, Geoffrey K. Pullum & Ivan A. Sag. 1985. *Generalized phrase structure grammar*. Oxford: Blackwell.

Gazdar, Gerald & Geoffrey K. Pullum. 1982. *Generalized phrase structure grammar: A theoretical synopsis*. Cognitive Science Research Paper 007, University of Sussex.

Goldsmith, John. 1976. *Autosegmental phonology*. MIT Doctoral Dissertation.

Graf, Thomas. 2010. Comparing incomparable frameworks: A model theoretic approach to phonology. *University of Pennsylvania Working Papers in Linguistics* 16: Article 10. Available at: http://repository.upenn.edu/pwpl/vol16/iss1/10.

Graf, Thomas. 2011. Closure properties of Minimalist derivation tree languages. In Sylvain Pogodalla & Jean-Philippe Prost (eds.), *LACL 2011*, 96–111. Heidelberg: Springer.

Graf, Thomas. 2012a. Locality and the complexity of Minimalist derivation tree languages. In Philippe de Groote & Mark-Jan Nederhof (eds.), *Formal grammar*, 208–227. Heidelberg: Springer.

Graf, Thomas. 2012b. Movement-generalized minimalist grammars. In Denis Béchet & Alexander J. Dikovsky (eds.), *LACL 2012*, 58–73. Heidelberg: Springer.

Graf, Thomas. 2013. *Local and transderivational constraints in syntax and semantics*. UCLA Doctoral Dissertation. Available at: http://thomasgraf.net/doc/papers/Graf13Thesis.pdf.

Graf, Thomas. 2014a. Late merge as lowering movement in Minimalist grammars. In Nicholas Asher & Sergei Soloviev (eds.), *LACL 2014*, 107–121. Heidelberg: Springer.

Graf, Thomas. 2014b. Models of adjunction in Minimalist grammars. In Glynn Morrill, Reinhard Muskens, Rainer Osswald & Frank Richter (eds.), *Formal grammar*, 52–68. Heidelberg: Springer.

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

Graf, Thomas. 2018. Locality domains and phonological c-command over strings. *NELS* 48. 257–270. Available at: http://ling.auf.net/lingbuzz/004080.

Graf, Thomas. 2019. A subregular bound on the complexity of lexical quantifiers. In Julian J. Schlöder, Dean McHugh & Floris Roelofsen (eds.), *Proceedings of the 22nd Amsterdam colloquium*, 455–464.

Graf, Thomas. 2022. Typological implications of tier-based strictly local movement. *SCiL* 5. 184–193.

Graf, Thomas & Natasha Abner. 2012. Is syntactic binding rational? In *Proceedings of the 11th international workshop on tree adjoining grammars and related formalisms (TAG+11)*, 189–197. Available at: http://thomasgraf.net/doc/papers/GrafAbner12TAG.pdf.

Graf, Thomas, Alëna Aksënova & Aniello De Santo. 2016. A single movement normal form for Minimalist grammars. In Annie Foret, Glyn Morrill, Reinhard Muskens, Rainer Osswald & Sylvain Pogodalla (eds.), *Formal grammar*, 200–215. Heidelberg: Springer.

Graf, Thomas & Aniello De Santo. 2019. Sensing tree automata as a model of syntactic dependencies. *MoL* 16. 12–26. Available at: https://www.aclweb.org/anthology/W19-5702.

Graf, Thomas & Kalina Kostyszyn. 2021. Multiple wh-movement is not special: The subregular complexity of persistent features in Minimalist grammars. *SCiL* 4. 275–285.

Graf, Thomas & Connor Mayer. 2018. Sanskrit n-retroflexion is input-output tier-based strictly local. *SIGMORPHON* 15. 151–160.

Graf, Thomas & Nazila Shafiei. 2019. C-command dependencies as TSL string constraints. *SCiL* 2. 205–215.

Halle, Morris. 1973. Prolegomena to a theory of word-formation. *Linguistic Inquiry* 4. 451–464.
Harkema, Henk. 2001. A characterization of Minimalist languages. In Philippe de Groote, Glyn Morrill & Christian Retoré (eds.), *Logical aspects of computational linguistics (LACL'01)*, 193–211. Heidelberg: Springer.
Hayes, Bruce. 1995. *Metrical stress theory*. Chicago: Chicago University Press.
Heim, Irene. 1998. Anaphora and semantic interpretation: A reinterpretation of Reinhart's approach. *MIT Working Papers in Linguistics* 25. 205–246.
Heinz, Jeffrey. 2009. On the role of locality in learning stress patterns. *Phonology* 26. 303–351.
Heinz, Jeffrey. 2010. Learning long-distance phonotactics. *Linguistic Inquiry* 41. 623–661.
Heinz, Jeffrey. 2014. Culminativity times harmony equals unbounded stress. In Harry van der Hulst (ed.), *Word stress: Theoretical and typological issues*, 255–275. Cambridge, UK: Cambridge University Press.
Heinz, Jeffrey, Kasprzik Anna & Timo Kötzing. 2012. Learning in the limit with lattice-structured hypothesis spaces. *Theoretical Computer Science* 457. 111–127.
Heinz, Jeffrey & Regine Lai. 2013. Vowel harmony and subsequentiality. *MoL* 13. 52–63. Available at: http://www.aclweb.org/anthology/W13-3006.
Heinz, Jeffrey, Chetan Rawal & Herbert G. Tanner. 2011. Tier-based strictly local constraints in phonology. *ACL* 49. 58–64. Available at: http://www.aclweb.org/anthology/P11-2011.
Hunter, Tim. 2011. Insertion Minimalist grammars: Eliminating redundancies between merge and move. In Makoto Kanazawa, András Kornai, Marcus Kracht & Hiroyuki Seki (eds.), *The mathematics of language: 12th Biennial Conference*, 90–107. Heidelberg: Springer.
Hunter, Tim & Robert Frank. 2021. Comparing methods of tree-construction across mildly context-sensitive formalism. *SCiL* 4. 355–358.
Huybregts, Riny. 1984. The weak adequacy of context-free phrase structure grammar. In Ger J. de Haan, Mieke Trommelen & Wim Zonneveld (eds.), *Van periferie naar kern*, 81–99. Dordrecht: Foris.
Jäger, Gerhard. 2002. Gradient constraints in finite state OT: The unidirectional and the bidirectional case. In Ingrid Kaufmann & Barbara Stiebels (eds.), *More than words. A festschrift for Dieter Wunderlich*, 299–325. Berlin: Akademie Verlag.
Jardine, Adam. 2016. Computationally, tone is different. *Phonology* 33. 247–283.
Jardine, Adam & Kevin McMullin. 2017. Efficient learning of tier-based strictly k-local languages. In *Proceedings of language and automata theory and applications*, 64–76. Berlin: Springer.
Joshi, Aravind. 1985. Tree-adjoining grammars: How much context sensitivity is required to provide reasonable structural descriptions? In David Dowty, Lauri Karttunen & Arnold Zwicky (eds.), *Natural language parsing*, 206–250. Cambridge: Cambridge University Press.
Joshi, Aravind, K. Vijay-Shanker & David Weir. 1991. The convergence of mildly context-sensitive grammar formalisms. In Peter Sells, Stuart M. Shieber & Thomas Wasow (eds.), *Foundational issues in natural language processing*, 31–81. Cambridge, MA: MIT Press.
Kalin, Laura. 2017. Licensing and differential object marking: The view from Neo-Aramaic. *Syntax* 21. 112–159.
Kanazawa, Makoto & Sylvain Salvati. 2012. MIX is not a tree-adjoining language. *ACL* 50. 666–674.
Kaplan, Ronald M. & Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20. 331–378.
Kaplan, Ronald M. & Annie Zaenen. 1988. Long-distance dependencies, constituent structure, and functional uncertainty. In Mark Baltin & Anthony Kroch (eds.), *Alternative conceptions of phrase structure*. Chicago, IL: Chicago University Press.

Karttunen, Lauri. 1998. The proper treatment of optimality in computational phonology. In *Proceedings of the international workshop on finite state methods in natural language processing*, 1–12. Stroudsburg, PA: Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/W98-1301.

Karttunen, Lauri & Kenneth R. Beesley. 2005. Twenty-five years of finite-state morphology. In Antti Arppe, Lauri Carlson, Krister Lindén, Jussi Piitulainen, Mickael Suominen, Martti Vainio, Hanna Westerlund & Anssi Yli-Jyrä (eds.), *Inquiries into words, constraints and context. Festschrift for Kimmo Koskenniemi on his 60th birthday*, 71–83. Stanford, CA: CSLI.

Karttunen, Lauri, Ronald M. Kaplan & Annie Zaenen. 1992. Two-level morphology with composition. In *COLING'92*, 141–148. Availble at: http://www.aclweb.org/anthology/C92-1025.

Kasper, Robert, Bernd Kiefer, Klaus Netter & K. Vijay-Shanker. 1995. Compilation of HPSG to TAG. *ACL* 33. 92–99.

Kiparsky, Paul. 2002. Disjoint reference and the typology of pronouns. In Ingrid Kaufmann & Barbara Stiebels (eds.), *More than words*, 179–226. Berlin: Akademie Verlag.

Kobele, Gregory M. 2006. *Generating copies: An investigation into structural identity in language and grammar*. UCLA Doctoral Dissertation. Available at: https://home.uni-leipzig.de/gkobele/files/unpub/Kobele06GeneratingCopies.pdf.

Kobele, Gregory M. 2008. Across-the-board extraction and Minimalist grammars. In *Proceedings of the ninth international workshop on tree adjoining grammars and related frameworks*, 113–128. ACL.

Kobele, Gregory M. 2010. Without remnant movement, MGs are context-free. *MOL* 10–11. 160–173. Heidelberg: Springer.

Kobele, Gregory M. 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In Sylvain Pogodalla & Jean-Philippe Prost (eds.), *LACL 2011*, 129–144. Heidelberg: Springer.

Kobele, Gregory M. 2015. LF-copying without LF. *Lingua* 166. 236–259.

Kobele, Gregory M., Christian Retoré & Salvati. Sylvain. 2007. An automata-theoretic approach to Minimalism. In James Rogers & Stephan Kepser (eds.), *Model theoretic syntax at 10*, 71–80.

Koller, Alexander & Marco Kuhlmann. 2011. A generalized view on parsing and translation. In *Proceedings of the 12th international conference on parsing technologies*, 2–13. Stroudsburg, PA: Association for Computational Linguistics. Available at: http://www.aclweb.org/anthology/W11-2902.

Kornai, Andras. 1985. Natural language and the Chomsky hierarchy. In *Proceedings of the EACL 1985*, 1–7.

Koskenniemi, Kimmo. 1983. *Two-level morphology: A general computational model for word-form recognition and production. Publication 11*. Helsinki: University of Helsinki, Department of General Linguistics.

Langendoen, D. Terence. 1981. The generative capacity of word-formation components. *Linguistic Inquiry* 12. 320–322.

Laszakovits, Sabine. 2018. Case theory in Minimalist grammars. In Annie Foret, Greg Kobele & Sylvain Pogodalla (eds.), *Formal Grammar*, 37–61. Heidelberg: Springer.

McMullin, Kevin, Alëna Aksënova & Aniello De Santo. 2019. Learning phonotactic restrictions on multiple tiers. *SCiL* 2. 377–378.

McNaughton, Robert. 1974. Algebraic decision procedures for local testability. *Mathematical Systems Theory* 8. 60–76.

McNaughton, Robert & Seymour Papert. 1971. *Counter-free automata*. Cambridge, MA: MIT Press.

Michaelis, Jens. 2001. Transforming linear context-free rewriting systems into Minimalist grammars. In *LACL'01: Proceedings of the 4th international conference on logical aspects of computational linguistics*, 228–244.
Michaelis, Jens. 2004. Observations on strict derivational minimalism. *Electronic Notes in Theoretical Computer Science* 53. 192–209.
Michaelis, Jens. 2009. An additional observation on strict derivational minimalism. In James Rogers (ed.), *FG-MOL 2005*, 101–111. Heidelberg: Springer.
Michaelis, Jens & Marcus Kracht. 1997. Semilinearity as a syntactic invariant. In Christian Retoré (ed.), *Logical aspects of computational linguistics*, 329–345. Heidelberg: Springer.
Mohri, Mehryar. 1996. On some applications of finite-state automata theory to natural language processing. *Journal of Natural Language Engineering* 2. 1–20.
Mohri, Mehryar. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics* 23. 269–311.
Mohri, Mehryar. 2000. Minimization algorithms for sequential transducers. *Theoretical Computer Science* 234. 177–201.
Mönnich, Uwe. 1999. On cloning context-freeness. In Hans-Peter Kolb & Uwe Mönnich (eds.), *Mathematics of syntactic structure*, 195–231. Berlin: Walter de Gruyter.
Moradi, Sedigheh, Alëna Aksënova & Thomas Graf. 2019. The computational cost of generalizations: An example from micromorphology. *SCiL* 2. 367–368.
Morawietz, Frank. 2003. *Two-step approaches to natural language formalisms*. Berlin: Walter de Gruyter.
Nevins, Andrew. 2011. Multiple agree with clitics: Person complementarity vs. omnivorous number. *Natural Language and Linguistic Theory* 28. 939–971.
Peters, Stanley & Robert W. Ritchie. 1971. On restricting the base component of transformational grammars. *Information and Control* 18. 483–501.
Peters, Stanley & Robert W. Ritchie. 1973. On the generative power of transformational grammars. *Information Sciences* 6. 49–83.
Pin, Jean-Eric. 1997. Syntactic semigroups. In *Handbook of language theory*, vol. 1, 679–764. Berlin: Springer.
Radzinski, Daniel. 1991. Chinese number names, tree adjoining languages, and mild context sensitivity. *Computational Linguistics* 17. 277–300.
Reinhart, Tanya. 1996. Interface economy: Focus and markedness. In Chris Wilder, Hans-Martin Gärtner & Manfred Bierwisch (eds.), *The role of economy principles in linguistic theory*, 146–169. Berlin: Akademie Verlag.
Ristad, Eric Sven. 1993. *The language complexity game*. Cambridge, MA: MIT Press.
Roark, Brian & Richard Sproat. 2007. *Computational approaches to morphology and syntax.* Oxford: Oxford University Press.
Rogers, James. 1998. *A descriptive approach to language-theoretic complexity*. Stanford, CA: CSLI.
Rogers, James. 2003. Syntactic structures as multi-dimensional trees. *Research on Language and Computation* 1. 265–305.
Rogers, James, Jeffrey Heinz, Gil Bailey, Edlefsen Matt, Molly Vischer, David Wellcome & Sean Wibel. 2010. On languages piecewise testable in the strict sense. In Christan Ebert, Gerhard Jäger & Jens Michaelis (eds.), *The mathematics of language*, 255–265. Heidelberg: Springer.
Rogers, James & Geoffrey K. Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information* 20. 329–342.

Ruiz, José, Salvador España & Pedro García. 1998. Locally threshold testable languages in strict sense: Application to the inference problem. In Vasant Honavar & Giora Slutzki (eds.), *Grammatical inference: 4th international colloquium, ICGI-98 Ames, Iowa, USA, July 12–14, 1998*, 150–161. Berlin: Springer.

Salvati, Sylvain. 2011. *MIX is a 2-MCFL and the word problem in is captured by the IO and the OI hierarchies*. Technical report, France: INRIA Bordeaux.

Schützenberger, Marcel-Paul. 1965. On finite monoids having only trivial subgroups. *Information and Control* 8. 190–194.

Schützenberger, Marcel-Paul. 1977. Sur une variante des fonctions séquentielles. *Theoretical Computer Science* 4. 47–57.

Seki, Hiroyuki, Takashi Matsumura, Mamoru Fujii & Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88. 191–229.

Shafiei, Nazila & Thomas Graf. 2020. The subregular complexity of syntactic islands. *SCiL* 3. 272–281.

Shieber, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8. 333–345.

Simon, Imre. 1975. Piecewise testable events. In Helmut Brakhage (ed.), *Automata theory and formal languages 2nd GI Conference*, 214–222. Berlin: Springer.

Stabler, Edward P. 1992. *The logical approach to syntax: Foundations, specifications and implementations of theories of government and binding*. Cambridge, MA: MIT Press.

Stabler, Edward P. 1997. Derivational minimalism. In Christian Retoré (ed.), *Logical aspects of computational linguistics*, 68–95. Berlin: Springer.

Stabler, Edward P. 2003. Comparing 3 perspectives on head movement. *UCLA Working Papers in Linguistics* 10. 178–198.

Stabler, Edward P. 2006. Sidewards without copying. In Gerald Penn, Giorgio Satta & Shuly Wintner (eds.), *Formal grammar*, 133–146. Stanford, CA: CSLI.

Stabler, Edward P. 2011. Computational perspectives on Minimalism. In Cedric Boeckx (ed.), *Oxford handbook of linguistic Minimalism*, 617–643. Oxford: Oxford University Press.

Stump, Gregory. 2017. Rule conflation in an inferential realizational theory of morphotactics. *Acta Linguistica Academica* 64. 79–124.

Torr, John & Edward P. Stabler. 2016. Coordination in Minimalist grammars: Excorporation and across the board (head) movement. In *Proceedings of the 12th international workshop on tree Adjoining grammars and related formalisms (TAG+12)*, 1–17. Düsseldorf, Germany. Available at: https://www.aclweb.org/anthology/W16-3301.

Vijay-Shanker, K. 1987. *A study of tree adjoining grammars*. University of Pennsylvania Doctoral Dissertation.

Williams, Edwin. 2003. *Representation theory*. Cambridge, MA: MIT Press.