Gregory Kobele, Universität Leipzig

Computational Aspects of Cyclic Optimization

Project Description

This project investigates the computational underpinnings of cyclic optimization. Whereas the other projects aim to break new empirical ground, studying in depth if and how cyclic optimization can apply to certain empirical domains, this project attempts to understand the nature of cyclic optmization itself. The goals of this project are to determine the kinds of phenomena describable (or not) by cyclic optimization, to understand how cyclic optimization relates to other mechanisms for grammatical description, and to clarify the nature of the mechansims involved. The hope is that this will go hand in hand with the projects aiming to better understand the way cyclic optimization works in the context of natural language, with this project providing guidance for the proper implementation of theoretical ideas, and the other projects guiding this one with respect to the proper notion of cyclic optimization to formalise.

1. State of the art and preliminary work

The methods of theoretical linguistics are aimed at uncovering regularities in the mapping between linguistic form and meaning. Such regularities can be expressed in a multitude of ways, ranging from logical statements of well-formedness to generative procedures.

Given that we are studying properties of the brain, there is *a priori* reason to be skeptical that notational decisions are reflected in the actual brain-representation of the higher order linguistic regularities linguists study. We already recognize this fact, and do not attach any importance to notational decisions such as the length of edges in autosegmental representations, or the font used to write syntactic node labels, or even the names of syntactic node labels (deciding to shift node names one alphabet symbol down, AP becomes BP, CP becomes DP, DP becomes EP, . . . , PP becomes QP, VP becomes WP). Such things do not matter for the claims we intend our theories to be making, although we are forced to make some decision on these matters when presenting our theories. In these cases it is clear, but it can be difficult to distinguish between properties intrinsic to the phenomenon being investigated and accidental properties due to the notation or manner of its description.

Computational linguistics is able to help address this issue by proving (in-)equivalencies between different systems of notation. This allows us to identify the essential properties of phenomena as those which are preserved under notational changes. This perspective has been very successful, establishing (exceptionless) universal properties of language; that phonological patterns are (sub-)regular, and that syntactic ones are mildly context-sensitive [Heinz and Idsardi, 2013].

The discovery of bounds on the complexity of phonology and syntax allow us to categorize the fit to the typological data of linguistic grammar formalisms. If a syntactic grammar formalism goes beyond the mildly context-sensitive upper bound, we know that it generates not just things that we have never seen, but things that we have never seen that are more complex than anything we have ever seen. On the other hand, if our theory of grammar is too weak to generate mildly context-sensitive patterns, then we know that it cannot describe the full range of constructions in natural language.

In addition, these computationally real bounds on natural language provide guidance when developing theories which link the high level regularities in form and meaning discovered by linguists to its real-time implementation in performance systems, as in parsing.

When new theoretical frameworks emerge, it is important to determine what it means for a linguistic phenomenon to be describable in terms of a framework. In the context of the broader project, we can ask whether cyclic optimization is an essential property of natural language, or is it merely a notational device for describing a kind of pattern that could be described in other ways? If the latter, is there some class of patterns for which cyclic optimization provides a particularly succinct characterization? We thus in this project aim to directly address the hypothesis of the research unit that combining cyclicity and optimization achieves an added level of explanation not available to either alone.

Much work on optimization has been done in the computational linguistic world (beginning with Frank and Satta [1998], Karttunen [1998] and culminating with Riggle [2004]), which has shown in increasingly sophisticated and faithful ways how to view optimality theory as a simple regular relation between inputs and outputs. This demonstrates that aspects of optimization can actually be implemented without making reference to competitors at all. These works underscore the fact that (given certain formal conditions) the *effects* of optimization can be had without the *mechanism* of optimization. When these conditions are not met, having optimization is indeed more powerful than not having it - however, the patterns requiring the mechanism of optimization are unattested [Riggle, 2004, Heinz, 2018]; they include what phonologists have called *pathological* languages like "sour grapes" [Padgett, 1995] and "majority rule" [Bakovič, 2000].

As optimality theory is a framework for evaluating the interaction of ranked, violable constraints, it is in principle compatible with constraints of any formal power at all, including computationally completely intractable ones. If we want an explanatory theory (or one that could even in principle be realized in a finite computational device, such as the brain), we must therefore restrict the power of the constraints that can be used. In the simple syllable theory of Prince and Smolensky [2004], constraints have a very local character: don't add segments, don't delete segments, have onsets, and don't have codas. The 'local character' of these constraints can be operationalized by a formal implementation as weighted finite state transducers, mapping inputs to outputs paired with the number of constraint violations incurred. As noted by Ellison [1994], multiple individual finite state constraints can be ranked with respect to one another by transducer intersection, which combines multiple constraints into a single weighted transducer, whose weights are constraint violation vectors. Each path in this transducer corresponds to a candidate with its violation vector in a tableau. An optimal candidate corresponds to a path with a smallest violation vector. Finding such a candidate can be done via a standard shortest path algorithm. The crucial insight behind this approach is that: by using a representation (a finite state transducer) which compactly represents an infinite tableau, operations like optimization can be implemented (and efficiently).

On the syntactic side, Kepser and Mönnich [2006] show that, given similar conditions, optimality theory over trees can be implemented without making reference to competitors (i.e. as a finite state tree transduction). Graf [2013] has used this technique to demonstrate that transderivational constraints over minimalist grammar derivations are implementable without making reference to alternative derivations. As shown by [Graf, 2011, Kobele, 2011], arbitrary finite state filters on both derivations and derived trees can be encoded in the minimalist feature calculus. As a simple example, the effects of a complexity filter [Koopman, 2002] prohibiting a complex possessor in German (stating roughly that a SPEC-DP in German cannot contain a DP with an overt SPEC) can be achieved by modifying the categories of our grammar, splitting the monolithic and undifferentiated D feature into a D_e feature (representing a silent DP), a D_0 feature (representing an overt DP without an overt SPEC), a D_1 feature (representing an overt DP with a D_0 SPEC) and a D_* feature (representing an overt DP with a D_1 or higher SPEC). (We also need to differentiate categorially between overt and silent N and A heads.) These formal features are imbued with meaning by constraining their checking potential - a D_0 allows only a D_e to move to its specifier, a D_1 requires a D_0 to move to its specifier, and a D_* requires a D_1 or a D_* in its specifier.

Despite optimization being well-studied in computational linguistics, next to nothing has been done in terms of cyclic optimization: cyclicity is itself a computationally very understudied mechanism [Dolatian, 2020, Sproat, 1992]. Intuitively, cyclicity involves repeatedly applying the rules of the (say, phonological) grammar at various points in a (separate, say, syntactic or morphological) structure building process. (Peters and Ritchie [1973] prove that cyclic rule application in transformational syntax is responsible for its overwhelming generative capacity. This conception of the cycle is unified with the structure building one in the next section.) Sproat (pg 107), in the context of morpho-phonology, asserts that "*cyclic rules are intimately intertwined with the process of word building.*" What makes this computationally problematic is that the grammatical output is the result not of applying the rules of the grammar to an input form, but of applying them over and over and over again. While it is understood how to apply a grammar once (or more generally some fixed finite number of times), with cyclicity one needn't be able to bound in advance the number of times a grammar might need to be applied to obtain a final output. The fundamental work of Kaplan and Kay on the formal power of phonological rule systems [Kaplan and Kay, 1994] explicitly restricts attention to *non-cyclic* rule application. They observe that individual contextual rewriting rules are regular relations. They note that while sequential and disjunctive combination of rule (blocks) preserves regularity, *cyclic* combination does not (and propose on these grounds to exclude it from consideration). While cyclic rule application was excluded from Kaplan and

Kay's study, the actual condition on the well-behavedness of phonological rules they adopted was more general; they required that no rule would be able to reapply to its own output. (Cyclicity would then be permitted, so long as the outputs of previous cycles were inviolable.) The earlier work by Johnson [1972] was the first to note that phonological rules as used in generative phonology were regular, and that iterative rule application did not, whereas simultaneous rule application did, preserve regularity. The crucial difference between simultaneous and iterative rule application schemes being, of course, that of rules reapplying to their own outputs. Yli-Jyrä [2019] notes that this is not an all-or-nothing property, and that certain instances of reapplication can in fact be dealt with in regular ways. He appeals to (old!) work in the theoretical computer science literature, observing that single tape turing machines working in linear time are regular. He shows how to leverage this result in the context of a formalism for inflecting and deriving word-forms (the *hunspell* formalism), which allows for (a limited form of) iterative rule application.

In an iterative setting, the grammar must reapply to its own output until some condition is met (usually: reaching a fixed point). What makes this fall short of cyclicity proper is that there is no structure creation being interleaved with (re)applying the grammar. This is exactly what is argued for by McCarthy [2010], in his 'serial' version of optimality theory, 'harmonic serialism'. Viewing a grammar as a relation (from inputs to outputs), reapplying the grammar to its own output can be modeled via the composition of this relation with itself. The n-fold application of a grammar to its outputs is then n-fold self-composition. In an iterative setting, the grammar might need to apply once to some inputs, twice to others, and so on, without an upper bound. This can be modeled by taking the (reflexive) transitive closure of the grammar, which is however known not to preserve regularity. The field of software verification (regular model checking) has encountered just this problem, and has explored conditions under which the reflexive transitive closure of regular relations remains regular [Nilsson, 2005]. Building on this literature, Hao [2017] has recently shown that (under certain assumptions) harmonic serialism over strings only allows for the description of regular relations. In order to make direct use of the theorems from the software verification literature, which require that input and output strings be of the same length, severe representational restrictions must be imposed, in particular that deletions and insertions are disallowed as such. Instead, each input is viewed as containing some number of 'empty' segments; insertion is construed as changing an empty segment to a non-empty one, and deletion is changing a non-empty segment to an empty one. (This purely formal move has parallels in the linguistic literature; the PARSE/FILL model of Prince and Smolensky [2004] has empty positions reserved for epenthesis, and deleted segments are still present in the output. Similarly, government phonology [Kaye et al., 1990] makes judicious use of empty positions in words.) Thus the relation computed by Hao's transitive closure of optimization steps is between words with a certain number of empty segments. The word *dog* would correspond to infinitely many distinct inputs (*dog*, *Edog*, *EEdog,. . . ,/dEog,*/dEEog/,. . . ; here *E* is an empty segment), each of which is mapped to its own output - these outputs needn't correspond to the same word, however! The awkwardness of empty segments is cleverly solved by Hao by a pre- and post-processing stage which non-deterministically inserts, and then deterministically deletes, these empty segments. The resulting non-deterministic transducer maps input strings (like *dog*) non-deterministically to padded input strings (like *EdEogEE*), which then get mapped non-deterministically to some number of optimization steps (perhaps 17, yielding *ddEEEdd*), and then get mapped deterministically to unpadded strings (like *dddd*). This does not yet have the fixed-point condition in place - given an input, we obtain all outputs which result from any number of optimization steps on some padded version of the input. We want however just those outputs which are the *fixed points* of optimization. Hao is able to show that in his setting, this can be represented as a finite state filter, the addition of which to the process preserves regularity. This is an *a priori* unexpected result, which is due to the simplicity of the constraints used: making no change is optimal whenever the input does not have any of a finite number of banned subsequences! The end result is a non-deterministic transducer mapping an input (*dog*) to the set of fixed points for each way of padding it out with empty segments. This single machine makes no use of optimization (iterated or otherwise), thereby demonstrating that even the effects of iterated optimization are achievable via more mundane means, in addition to proving that the input-output relations computed by Hao's version of harmonic serialism belong to a formally very restrictive class, thus (apparently correctly) predicting the typological absence of more computationally complex patterns.

While the formal study of iterativity (either in terms of optimization or rule application) is still in its infancy, there is as we have seen some promising prior work demonstrating *islands of tractability* of relevance to linguistics. Cyclicity is related to iterativity, in that the grammar is applied over and over again, but differs from it in that the reapplication of the grammar is determined by intermediate structure building steps.

Although it is natural to think of derivations as dynamic procedures, it has proven very useful to view the derivational process itself as a structure, typically a tree. This allows for the regularities in the derivational process to be investigated independently of the properties of the object the derivational process is constructing. This view is explicit in categorial grammar [Steedman, 2000], and is prominent in the tree-adjoining literature as well [Joshi and Schabes, 1997]. While less well-known in the linguistic literature on minimalism, this perspective is of fundamental importance here as well [Kobele et al., 2007]. A syntactic derivation proceeds by applying syntactic operations to syntactic expressions, which are themselves either lexical items, or previously derived objects. In the context of minimalist grammars, the syntactic operations are merge and move (or internal merge). A simplified derivation of the intransitive sentence *every sheep will bleat* might proceed as follows.

1. select *every* from the lexicon
2. select *sheep* from the lexicon
3. merge 1 and 2 (to form *every sheep*)
4. select *bleat* from the lexicon
5. merge 4 and 3 (to form the VP *[bleat [every sheep]]*)
6. select *will* from the lexicon
7. merge 6 and 5 (to form the TP *[will [bleat [every sheep]]]*)
8. move *every sheep* in 7

This characterization of a derivation as a list of numbered steps is reminiscient of a Hilbert style logic proof. In proof theory, many different representations of proofs have been studied, each of which foregrounds different aspects of the 'structure' of the proof. The derivation above can be presented as the structured object in figure 1 which represents the steps of the derivation above in a tree-like form that is congenial to linguists (and to computer scientists). In this structure, sisters are coarguments to

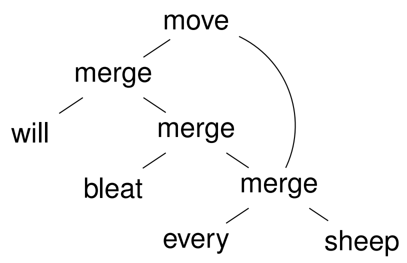


Figure 1: A derivation tree for *every sheep will bleat*

the operation their mothers are labeled with, and are thus derivationally independent - either one can be constructed before the other, or their construction can even be interleaved. Kobele et al. [2007] explicitly proves that the trees representing the convergent derivations in minimalist grammars are formally simpler than the trees they are the derivations of; the former are *regular*, while the latter are not. Using tools from automata theory, they show that in addition the map between the derivation trees and derived trees is regular as well, revealing a fundamental symmetry in human language - the set of phonotactically well-formed words of languages seem to be regular, and the maps between underlying forms and surface forms seem to be regular as well. This result was anticipated in Michaelis [2001], and the general idea of decomposing minimalist grammars into a regular set of underlying structures together with a uniform regular mapping realizing them as surface structures was worked out by Uwe Mönnich and his colleagues [Michaelis et al., 2001], though without the underlying structures corresponding to derivations.

This ability to move between derivational and representational perspectives on structure allows us to eliminate the tension between *recapitulative* and *interactionist* perspectives on cyclicity. This will be taken up in 2.2.

1.1. Project-related publications

1.1.1. Articles published by outlets with scientific quality assurance, book publications, and works accepted for publication but not yet published

1. J. Heinz, G. M. Kobele, and J. Riggle. Evaluating the complexity of Optimality Theory. *Linguistic Inquiry*, 40(2):277–288, 2009.
2. G. M. Kobele. Minimalist tree languages are closed under intersection with recognizable tree languages. In S. Pogodalla and J.-P. Prost, editors, *LACL 2011*, volume 6736 of *Lecture Notes in Artificial Intelligence*, pages 129–144, 2011.
3. G. M. Kobele and S. Salvati. The IO and OI hierarchies revisited. *Information and Computation*, 243:205–221, 2015.
4. A. Clark, M. Kanazawa, G. M. Kobele, and R. Yoshinaka. Distributional learning of some nonlinear tree grammars. *Fundamentica Informaticae*, 146(4):339–377, 2016.
5. G. M. Kobele. Without remnant movement, MGs are context-free. In C. Ebert, G. Jäger, and J. Michaelis, editors, *MOL 10-11*, volume 6149 of *Lecture Notes in Computer Science*, pages 160–173. Springer, 2010.
6. G. M. Kobele and J. Michaelis. Disentangling notions of specifier impenetrability: Late adjunction, islands, and expressive power. In M. Kanazawa, A. Kornai, M. Kracht, and H. Seki, editors, *The Mathematics of Language*, volume 6878 of *Lecture Notes in Computer Science*, pages 126–142. Springer, 2011.
7. G. M. Kobele. Formalizing mirror theory. *Grammars*, 5(3):177–221, 2002.
8. G. M. Kobele. Idioms and extended transducers. In *Proceedings of the Eleventh International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+11)*, pages 153–161, 2012.
9. G. M. Kobele. Features moving madly: A formal perspective on feature percolation in the minimalist program. *Research on Language and Computation*, 3(4):391–410, 2005.

1.1.2. Other publications, both peer-reviewed and non-peer-reviewed

10. G. M. Kobele, C. Retoré, and S. Salvati. An automata theoretic approach to minimalism. In J. Rogers and S. Kepser, editors, *Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07*, 2007.

1.1.3. Patents

not applicable

2. Objectives and work programme

2.1. Anticipated total duration of the project

48 months

2.2. Objectives

In the early days of the transformational cycle in phonology [Chomsky and Halle, 1968], phonological grammars operated on bracketed strings from the inside out. These bracketed strings were (obtained from) the results of the syntactic structure building process. I will here present a reformulation of the transformational cycle in Chomsky and Halle [1968] in terms of trees. This will facilitate a deeper understanding of the cycle, that will be useful in order to frame the questions under investigation. Figure 3 shows the general format of cyclic phonological interpretation over trees, and figure 2 gives a side-by-side presentation of the bracketed string notation and tree notation. In figure 3, the phonological

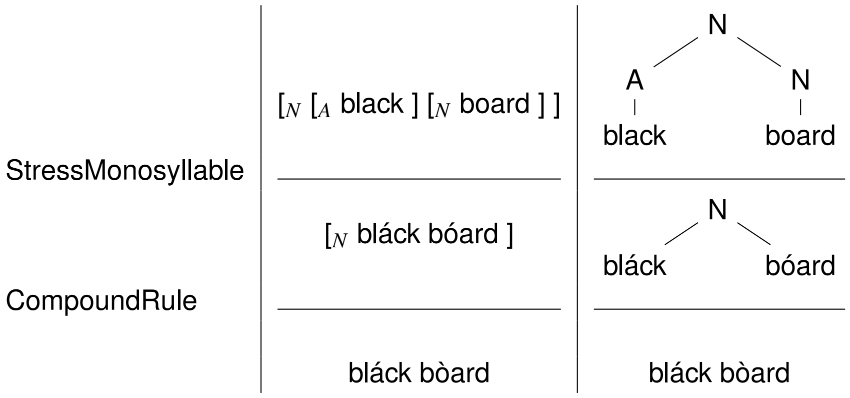


Figure 2: The transformational cycle in phonology: a comparison of notations

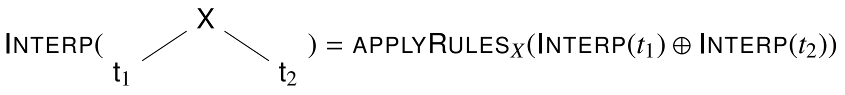


Figure 3: The transformational cycle in phonology: over trees

interpretation of a structure $[_X\ t_1\ t_2]$ involves first interpreting the immediate subparts, combining them into a single string (represented with the symbol $\oplus$), and then applying the ordered set of rules appropriate to the bracket type *X* to this string (representend with the symbol APPLYRULES$_X$). 'Bracket erasure' falls out from the recursive bottom up evaluation of phonological interpretation. (Brackets are used in the string representation to represent parent-child relations in the tree representation.) The scheme in figure 3 is a particular presentation of the formal notion of a homomorphic interpretation scheme, popularized by Montague [1974] for semantic interpretation. A homomorphic interpretation scheme maps a tree to a domain of interpretation in the way shown in figure 4. The transformational cycle can

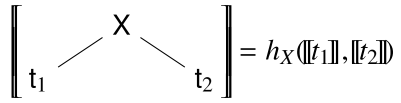


Figure 4: Homomorphic interpretation

be brought into the same format as homomorphic interpretation by setting the operation particular to a syntactic node *X* to be $h_X = \lambda x, y.$APPLYRULES$_X(x \oplus y)$. Similarly, any homomorphic interpretation scheme can be brought into the format of the transformational cycle by taking the generic combining operation to be the pairing operation: $m \oplus n = \langle m, n\rangle$. These formal translations violate the spirit of the transformational cycle, which is that the operation particular to the syntactic node *X* (APPLYRULES$_X$) should always be a *unary* operation, and the daughters of the syntactic node are combined in a fully generic way (i.e. a way not dependent on the identity of the node *X*) as an object of the same kind - strings should be combined to form strings, trees as trees, etc.

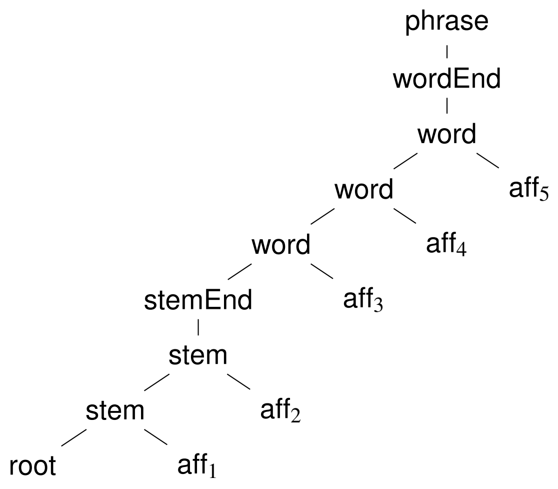
Given this intended restriction, semantic interpretation (a là Heim and Kratzer [1998]) should be brought into the rubic of the transformational cycle in the following way: the generic operations combining the interpretation of sisters are the type driven composition rules of function application and predicate modification, and the node-specific operation is the identity function. Settings in which pragmatic aspects of the utterance (such as scalar implicatures) are to be computed recursively in the syntactic structure (as proposed by Chierchia [2006]) can be accomodated by assigining this operation of pragmatic computation to the function APPLYRULES$_X$.

The current presentation of the transformational cycle is intended to illustrate that the cycle and homomorphic (compositional) interpretation are closely related: *the cycle is fundamentally a perspective on interface interpretation*. There are two foundational questions which emanate from this:

1. the nature of cyclic optimization

2. the proper formulation of cycle in syntax

Cyclic optimization is just cyclic interpretation where the operation APPLYRULES$_X$ involves optimization. Consider a Kiparsky style version of Stratal OT [Kiparsky, 2000], with three OT grammars corresponding to the stem, word, and phrase level. Here we have a single structure building operation, namely, adding an affix. After adding a stem level affix, we apply the stem level grammar, after applying *all* word level affixes we apply the word level grammar, and finally after constructing the final utterance the phrase level grammar applies. In accord with the terminology in the main proposal, we might say that Stratal OT is strictly iterative at the stem level, but works in batch mode at the word level. Shifting to a more representational perspective, a word we might wish to interpret phonologically will have the structure shown on the left in figure 5. In this example, there are two stem level affixes (numbered 1 and



phrase → wordEnd
wordEnd → word
word → word aff
word → stemEnd aff
stemEnd → stem
stem → stem aff
stem → root aff

Figure 5: The structure of words implicit in Stratal OT

2), and three word level affixes (three through five). The tree has three unary branches, marking the shift between levels. The operation APPLYRULES$_{stem}$ is the stem level OT grammar. This implements the idea that the stem level grammar applies after the addition of each stem level affix. As nothing special happens at the end of the stem domain, the operation APPLYRULES$_{stemEnd}$ is the identity function. This situation is reversed in the word domain. Here, nothing happens as each word level affix is added, and so the operation APPLYRULES$_{word}$ is the identity function. The action in the word domain comes after the word level affixes have been added. To implement this, we have created the node *wordEnd*, and interpret the operation APPLYRULES$_{wordEnd}$ as the word level grammar. Finally, the operation APPLYRULES$_{phrase}$ is the phrase level grammar. We see that various degrees of internal iterativity are reflected in whether and which nodes are interpreted with the identity function. The fact that Stratal OT is *layered*, i.e. that the stem level is beneath the word level which is in turn beneath the phrase level, is reconstructed representationally as a (regular) constraint on the possible input structures; they must be the parse trees of the context free grammar on the right in figure 5. Layering is thus seen to refer to whether the structures underlying possible inputs are restricted in some way (i.e. whether they have a non-trivial underlying grammar) or not.

The formal power of cyclic optimization in this sense and cyclic (ordered rule) interpretation coincide to the extent that optimality theory and ordered rewrite rules can be approximated with finite state means

[Frank and Satta, 1998, Kaplan and Kay, 1994]; (finite state) cyclic interpretation, whether optimizing or not, involves interpreting trees homomorphically where internal nodes are interpreted as finite state transductions. This is actually a *two-step* process, whereby an input tree is mapped homomorphically to an output tree, and then this output tree is evaluated to a string, by interpreting its internal nodes as finite state transductions. Nevertheless, even restricting the operations to finite state transductions does not mean that the formal power of cyclic interpretation is so restricted. There are two boundary cases that illustrate this point. Let us ignore the finite state transductions applying at each node (by restricting them to compute the identity function). Then cyclic interpretation reduces to tree-to-string homomorphisms, which are known to map the regular tree languages to the context-free string languages [Comon et al., 2002]. Even relaxing this onerous restriction on finite state transductions just a little bit, allowing them to compute strictly 3-local transductions [Chandlee and Heinz, 2018], enables simulating turing machine computations: we introduce a special set of alphabet symbols $q_0, q_1, \ldots$ which represent the position of the machine head and its internal state. Leftward (resp. rightward) moves of the machine on the tape are effected by SL-2 transductions of the form $aq_i \mapsto q_ja$ (resp. $q_ia \mapsto aq_j$). (To allow the machine access to its infinite tape, we can add a new symbol, *B*, representing a blank square, assume that the end of the string is marked with the special symbol #. Then moving rightward past the (current) end of the string is effected with a 2-SL transduction $q_i\# \mapsto Bq_j\#$.) Rewritings are similarly SL-2 transductions of the form $q_ia \mapsto q_jb$. (Deletions are the special case where *b* is empty.) This allows us to interpret unary branching trees as descriptions of turing machine computations. The successful turing machine computations will be those in which the head is at the beginning of the word, in a final state. This can be detected by a 2-(O)SL finite state transduction, which erases the head symbol if it is indeed in a final state at the beginning of the word, and leaves everything else untouched, or erases everything, if the head symbol is either not at the beginning of the word or it is not in a final state. We can thus use cyclic interpretation to simulate arbitrary turing machine computations, even with a regular, unary branching tree language as input. (Peters and Ritchie [1973] show the same is true in the more specialized setting of aspects-style transformational syntax.) As turing machines have universal computing power, any grammar formalism that is equivalent to them in power is unable to discriminate between language-like patterns and non-language-like ones.

While it turns out that cyclic optimization has the same formal properties as cyclic interpretation in the finite state setting, the question of how to conceptualize the cycle in syntax is not as straightforward - the previous discussion revealed that cyclic interpretation pertains fundamentally to *interfaces*, whereas the syntax is a *generative* module: syntax constructs structures, while cyclic interpretation is about deconstructing structures. The cycle in syntax was formulated at a time when syntax *was* an interpretative system: deep structures were given as inputs, and were interpreted (using transformations) as surface structures.[1] Unlike the case in phonology or semantics, here the inputs and outputs of cyclic interpretation were objects of the same kind: trees. This somewhat obfuscates the interpretative character of transformational syntax, which is depicted in figure 6. Historically, the diverse transformations

$$\textsc{Interp}\left(\begin{array}{ccc} & X & \\ \diagup & & \diagdown \\ t_1 & & t_2\end{array}\right) = \textsc{applyRules}_X\left(\begin{array}{ccc} & \bullet & \\ \diagup & & \diagdown \\ \textsc{Interp}(t_1) & & \textsc{Interp}(t_2)\end{array}\right)$$

where

$$\textsc{applyRules}_X = \text{change } \bullet \text{ to } X \text{ and } \begin{cases}\text{apply rules} & \text{if } X \text{ is cyclic}\\ \text{nothing else} & \text{otherwise}\end{cases}$$

Figure 6: The transformational cycle in syntax

of transformational grammar were replaced by the single operation of movement, and the bottom-up interpretative cycle operating over fully formed input trees was reformulated to be applied incrementally during the construction process of these trees (which can be rationally reconstructed using a technique called *deforestation* [Wadler, 1990]). Interleaving the transformational cycle (at nodes *C* and *D* instead of the now defunct *S* and *NP*) with a tree construction process based on external merge, we end up with something very much like the system of *On phases* [Chomsky, 2008], with feature transfer occuring once *C* (or *D*) is merged so as to provide the conditioning contexts for movement (internal merge). The

[1]This is true regardless of whether the locus of *semantic* interpretation is deep structure (as in generative semantics) or a transformationally obtained interpretation thereof (as in mainstream transformational syntax).

original formulation of minimalism [Chomsky, 1995] emerges from a transformational grammar where every node is cyclic - which means that movement operations (i.e. transformations) are (able to be) performed after each (external) merge step. Cyclic *optimization* in the syntax would be instantiated by, for example, using optimality theory at each phasal node [Müller, 2000, Fanselow and Čavar, 2001] to select the optimal way of applying movements.

This is quite different from the approach in Heck and Müller [2013], where what is being optimized is the choice of operation to perform at each derivational step. Here structure building competes with transformations (agreement and movement), with the operation allowed to be performed being the one whose application results in the most optimal structure. In apparent contrast to the cyclic interpretation approaches above, in this context the 'deep structure' input (or its dynamic unfolding as a construction process) seems to be lacking, as this construction process itself is the target of the optimization. There is an input here however: it takes the form of a workspace (or rather, a lexical subarry), which is at its core a specification of which lexical items should be externally merged with which others. We are thus given as input a 'merge-tree', corresponding again to deep structure, and must determine where in this tree movements and agreement may apply. As an example, the workspace for the sentence *every sheep will bleat*, divided up into sub-arrays at phasal nodes *C*, *v*, and *D* can be represented as follows:

$$\underbrace{\{_C\ will,\ \overbrace{\{_v\ bleat,\ \underbrace{\{_d\ every,\ sheep\}}_{D-array}\}}^{v-array}\}}_{C-array}$$

The three sub-arrays include the D-array containing *every* and *sheep*, the v-array containing *bleat* and the D-array, and the C-array containing *will* and the v-array. When constructing a derivation according to a workspace like the above, one must essentially work from the inside out, applying the grammatical operations to the objects contained in the sub-arrays, until they have been exhausted. Once they have been exhausted, the containing brackets are erased, the derivation continues with the next sub-array. This workspace structure can be given as a tree, as shown in figure 7.

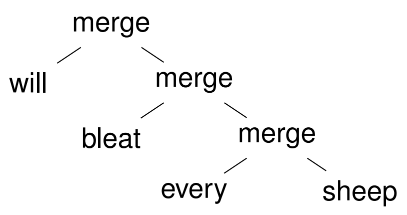


Figure 7: A lexical subarray for "every sheep will bleat", shown as a tree

Turning now to iterative optimization, as exemplified by harmonic serialism (over strings), there are two difficulties in recasting it in terms of cyclic interpretation. First, and foremost is the fact that there appears to be nothing to interpret, cyclicly or otherwise: the input to a harmonically serialising grammar is a string (there is no structure building taking place). Secondly, there is no fixed stopping point determined by the input: when cyclically interpreting a structure, you stop once the structure is interpreted. But when harmonically serializing a string, you stop once you reach a fixed point (if ever). The first point is dealt with by assuming that there is in fact (invisible) structure in the input. This structure is trivial, i.e. non-branching, and each node in the structure corresponds to one step of optimization. Thus iterativity is just a simple kind of cyclicity (where the structure is unary branching), but where the underlying structure is not given, and must be inferred. The indeterminacy of the underlying structure relates to the unknown number of cycles of optimization that should be undergone - this is the representational analogue of Hao's transitive closure computation. The filtering step, whereby only those outputs are kept which are fixed points, can be achieved by the addition of a single root node, which is interpreted as Hao's filter. Figure 8 shows some of the underlying structures for *dog* which harmonic serialism must cyclically interpret. The nodes labeled `cyc` are each interpreted as a single optimization cycle, and the node labeled `fix` is interpreted as a filter which accepts only those forms which are optimal.

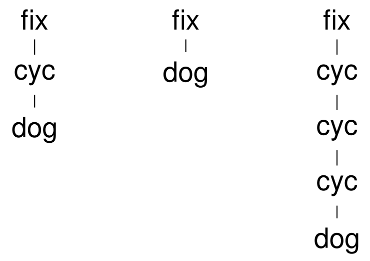


Figure 8: Some underlying structures to be harmonically optimized

2.3. Work programme including proposed research methods

The primary goal of this project is to understand the mechanism of cyclic optimization from a computational perspective. The fundamental computational problem here lies in the *cyclic* aspect thereof. We have two distinct settings for *cyclic optimization*:

1. the number of cycles is unknown in advance: there is a generative process (Harmonic serialism), which optimises until a fixed point is reached, if ever
2. the number of cycles is known in advance: there is an interpretative process which compositionally interprets a structure

Both of these settings will be targets of study.

2.3.1. Harmonic serialism

Harmonic Serialism over Strings (HS$_S$) The fundamental difficulty posed by harmonic serialism is the iteration of optimization, ending only if a fixed point is reached. Hao [2017] has ingeniously imported a technique from software verification which allows him to show that arbitrarily many cycles of optimization can be simulated by a single finite state transducer. Because the requirement that inputs and outputs be of equal length is not linguistically motivated or plausible, Hao's results need to be improved upon. One immediate goal of the project is to do just this. It seems plausible that there be an upper bound on the size increase that a string-based harmonic serialist grammar can compute: after a certain point, if epenthesis is an optimal output in a cycle, the outputs of the grammar will never converge. Establishing this will be a first step in bringing Hao's result in line with linguistic assumptions; we will be able to uniformly map an input word to a single string with a particular, maximal, number of silent segments. This approach to bounding the number of possible cycles in the length of the input was advocated for by Peters and Ritchie [1973], which demonstrated that different kinds of bounds on possible cycles restricted the generative capacity of transformational syntax.

Harmonic Serialism over Structures (HS$_T$) As many proposals about the representation of phonological elements view them as structured objects (e.g. autosegmental theory, government phonology, etc), extending Hao's work so as to apply over trees (and more generally graphs) is a natural further step. However, it is non-obvious just what reasonable minimal changes should be allowed for generating candidates, and therefore also the appropriate notion of 'sameness of structure' which in the domain of strings was realized as length. The problem here is not technical, but rather empirical; there have not been enough linguistic studies using harmonic serialism over trees to abduce this information from.

2.3.2. Cyclic Interpretation

Generative Capacity (GC) As sketched previously, the generative capacity of cyclic interpretation is unrestricted, as even with very restricted operations we can simulate turing machine computations. The construction given in that section, while formally correct, is linguistically unsatisfying: it relies on a number of linguistically questionable moves, such as extending our vocabulary with abstract symbols, repeatedly altering the outputs of previous cycles, having one syntax node per turing machine rule, and performing arbitrary, non-phonologically motivated symbol manipulations. Additionally, while the transductions implementing the turing machine transitions are formally quite simple, it is not immediately clear how to implement them in terms of OT grammars (i.e. optimization). The question here is to determine the actual expressivity of cyclic interpretation *as it is used by linguists*, as well as whether there is a way to reduce the expressivity of cyclic interpretation so as to obtain a better empirical fit with the typological data. Some of these linguistically unsatisfying moves can be resolved in a linguistically satisfying way without affecting the argument: instead of having one syntactic node per turing machine rule, we can compose the turing machine rules into a single finite state machine, and apply this machine at each node. This will no longer be SL-2, however. Instead of introducing new vocabulary symbols, we can use our original vocabulary to encode arbitrary information. (So *aaa* might be interpreted as state q_7, and *aab* as the phonological letter *c*.) However this will no longer be SL-2 either, and the problem of the linguistic unnaturalness of the manipulations will be exacerbated. On the other hand, some of the linguistically unsatisfying moves seem more fundamental. The strict cycle condition limits the manipulation of material created during a previous cycle. This would completely block the construction in the previous section, which relies on being able to modify previous cycle material *ad libitum*. Formalizing and studying the effects of the strict cycle condition (and its linguistically attested variants) is a major goal of this task. Similarly, naturality (of a.o. phonological rules) is a major focus of linguistic theory, and its imposition would seem to block the turing machine simulation. Another goal is to better understand the role of naturalness in limiting the generative capacity of cyclic interpretation.

On Phases (OP) In the previous section, Chomsky's 'On Phases' system [Chomsky, 2008] was argued to be the natural modern incarnation of the transformational cycle in syntax. The crucial mechanism of the 'On Phases' system is feature transmission, which allows a feature of a higher head to be checked counter-cyclically by a lower head. The goal of this sub-project is to implement and study feature transmission in minimalist grammars [Stabler, 1997], a formal framework for the minimalist program. Although counter-cyclicity is formally not very well understood (Kobele and Michaelis [2011] is one of the only studies investigating the impact of counter-cyclicity on the generative capacity of minimalist grammars), the kind of counter-cyclicity involved in feature transmission is very local (one might analogize it to strict locality in phonology), and seems unlikely to change the generative capacity in a substantive way. I suspect that the formalism of 'On phases' and the original Stabler formalism will be intertranslatable into one another, in the sense that there will be a direct translation mapping derivation trees of the one into those of the other.

Local Optimization (LO) The system of Heck & Müller [Heck and Müller, 2013] instantiates cyclic optimization over derivational steps. As sketched in the previous section, it is a form of cyclic interpretation (via optimization) where the object cyclically interpreted is something akin to a workspace.[2] The output of cyclic interpretation is a derivation tree, which can be interpreted (homomorphically) as a derived tree in the usual way. One goal of this sub-project is to formalize this system as just described. It seems initialy plausible, based on the constraints proposed in Heck and Müller [2013], that the information about the derived structure relevant to evaluating constraint violations is already present in what Kobele [2015] calls 'syntactic type': a fixed, finite set of information about primarily unchecked features that is efficiently computable on the basis of the derivation tree. For example, $\bullet k\ t; k$ is the syntactic type of a finite T', which contains an expression which has not yet been assigned case (k). This, if it holds up under more detailed linguistic investigation, would render unnecessary information about the numeration – in other words, this would allow us to reformulate Heck & Müller's systems in a purely algebraic way. The

[2]This is a familiar conceptual difficulty encountered more generally in OT approaches to syntax: what is the input? As one usually doesn't want to compare sentences expressing fundamentally different meanings, the input is taken to specify at a minimum the basic predicate argument relations that the sentences to be compared should express. In (non-generative semantic) transformational grammar, this was the level of DS, which in minimalism is obtained by erasing the movement nodes from the derivation tree. The lexical sub-array data structure may sometimes also deliver this information, so long as there are sub-arrays between all arguments of the verb.

numeration plays a role in Heck & Müller's system when evaluating `(external) merge`, as one needs to make reference to which expressions are available to be merged. Without access to a numeration, it would seem that there would be infinitely many candidates for `merge`. However, if syntactic type is all that is necessary in order to perform constraint evaluation, then we just need one `merge` operatation type for each possible syntactic type (of which there are finitely many [Michaelis, 2001]). Another role of the numeration for Heck & Müller is to enforce a ban on counter-bleeding; stated without numerations it becomes: if merger of an expression at some earlier point would have been an optimal choice (i.e. it would have bled other operations), but was not performed, than this expression cannot be merged later on. This counter-bleeding can be achieved without numerations by percolating a filter up the derivation tree, banning merging an expression of a syntactic type which would have won, had it been merged earlier. A second goal of this sub-project is to determine to what extent Heck & Müller's system can be reformulated in this way, and more generally, what sort of properties this reformulation is contingent on.

Parsing (P) The goal of interest in this sub-project is to better understand the nature of cyclic optimization from the perspective of language use, in particular, as regards to how parsing according to a cyclicly optimizing grammar might be achieved. In addition, the lower-level vocabulary of parsing makes available many more subtle distinctions between expressions on the basis of which observed differences in linguistic behaviour can be explained. For example, in the now classic approach of Chomsky and Miller [Miller and Chomsky, 1963], it is observed that the transient memory load of a parser (qua push-down automaton) as it processes center vs right embedded sentences corresponds to increased processing difficulty (see Resnik [1992] for a more systematic version of this idea).

There is often a tension in the parsing literature (in computer science) between the succinctness of the grammar formalism, and the complexity of parsing. The efficiency of parsing is often increased by doing as much work as possible in advance of any input. This involves putting the grammatically relevant information into as efficiently accessible a form as possible. The more succinct a grammar formalism is, however, the more work needs to be done to unpack the information it contains. Returning to our topic, even if it turns out that cyclicly optimizing grammars are efficiently parsable, this might only be with respect to a different representation. Accordingly, we will investigate both parsing cyclicly optimizing grammars directly, and via alternative representations. We will also study whether cyclicly optimizing grammars provide a good fit to behavioural data via complexity metrics.

2.3.3. Timeline

The first PhD student will focus on formalizing and comparing the 'on phases' and the 'Heck & Müller' extensions of minimalism, in the context of minimalist grammars [Stabler, 1997]. The overall goal of the dissertation is to understand cyclic interpretation in modern syntax, and to evaluate these (and perhaps alternatives) with respect to one another in terms of empirical coverage, generative capacity, parsing complexity, and on complexity metrics linking grammars to empirically determined processing difficulties.

The second PhD student will focus on harmonic serialism (the generative perspective on cyclic optimization), first over strings, then over tree-like structures. After the formal development, parsing will be investigated. Finally, these tools will be brought to bear on the particular analyses developed in project 7.1.

The PI will contribute to research on all topics, and will be primarily concerned with the general architecture of cyclic interpretation.

	PhD Student 1	PhD Student 2	Principal Investigator
2021	LO	HS_S	LO, HS_S, GC
2022	LO	HS_T	LO, HS_T, GC
2023	OP	P(HS)	OP, P, GC
2024	P(LO,OP)	HS	P, GC

2.4. Data handling

The source code for all programs to be written will be made publicly available.

2.5. Other information : Cooperation within the research unit

This project investigates the nature of cyclic optimization; its relation to other formalisms, which features of cyclic optimization influence its generative capacity, etc. Accordingly, the present project contributes to all other projects utilizing cyclic optimization in being able to inform theoretical decisions based on alternative equivalent representations and on which theoretical proposals are more or less restrictive. On the other hand, all projects utilizing cyclic optimization contribute directly to this one in that they offer linguistically motivated proposals to guide formalization and analysis. Many projects push up against the limits of cyclicity: the cycle as described here is a particular kind of homomorphic interpretation, and homomorphic interpretation requires all information to be local: the interpretation of a complex is a function of its mode of combination together with the interpretations of its immediate parts. In some cases, however, seemingly non-local information must be taken into account in determining the results of a particular cycle. The standard 'trick' to circumvent non-locality (in the computer science literature) is to enrich the representations so as to encode the long-distance information locally. In this case, the important questions (from a computational perspective) revolve around the nature and amount of information which must be maintained and passed around.

Mor🚲Mor A fundamental goal of one of the graduate students working on the present project is the formal development of harmonic serialism from simple strings to richer structures, or in other words to develop a formal foundation for harmonic serialism over structures richer than strings. The **Mor🚲Mor** project promises a case study in exactly this, and there will be close cooperation between these two projects.

Mor🚲Phon Recent computational investigation into the nature of tonal phenomena [Jardine, 2016] suggests that these are different in kind than other phonological phenomena (e.g. unbounded tone plateauing and the like are quite common in the tonal domain, but vanishingly infrequent in other phonological domains, like stress or harmony). This might suggest, following the logic in Heinz and Idsardi [2013], that tone may be a linguistic domain in its own right. Regardless of the ultimate validity of this suggestion, we would like our theories of linguistic phenomena to explain their typological variability, both within and *across* domains. An important interim goal is to be able to *characterize* the range of possible variation within a theory. This might take the form of constraints on representations (perhaps an autosegmental-like representation for tone allows for a reduction of complexity of the needed operations to derive unbounded tone plateauing vs a comparable vowel harmony process over sequences of phonemes), or of constraints on possible operations. This project on morphological strata of tone investigates both rich representational questions, as well as questions of how cyclic optimization should be structured so as to best account for the data.

Sem🚲Phon The project on semantic and phonological correlates of affix order utilizes stratal OT, thus providing an empirical case study into the nature of strata and cyclic optimization which is useful in a particular domain (which is relevant for identifying possible substantive constraints on cyclic systems which rein in their generative capacity while still being able to describe the data). Perhaps more interesting, it investigates cases of non-locality (discontinuous dependencies), which would seem to pose a problem for the cycle. The important question here is to determine the nature and limits of discontinuity, and thus of the information which must be percolated throughout the cycle.

Syn🚲Syn The project on syntactic repairs has at least two points of contact with the present proposal. First, one approach to repair to be investigated will be the 'local optimization' approach to minimalism which is the subject of one of the graduate student sub-projects. Second, instances of non-local information transmission occur (at least in the form of resumptive pronominalization), the solutions to which are directly relevant to the question of how much information must be percolated through the cycle. In addition, though less directly related to the present project's goals, it may prove useful to couch some of the investigations into repairs in the minimalist program in the context of a formalization thereof, which is both a research interest of mine and is as well an object of study in the afore mentioned graduate project.

Syn🚲Phon One point of particular interest about the project on prosodic dislocation is that it is at its outset confronted with difficulties regarding non-locality for cyclicity. This manifests itself in particular in that the *B* conjunct of a conjunction *A & B* might be a very large expression, which has been subject to perhaps multiple cycles, and yet its edge may remain permeable for the conjunctive marker. Another interesting aspect of this project it that it involves what we might call a *cyclic pipeline*: syntax is interpreted (cyclicly) as a prosodic structure, which itself contains pieces (words comprised of morphemes) which must be interpreted cyclicly as phonological sequences. The question thus is made very salient: how are multiple cyclic interpretive processes to interact? One possibility is that these are in a straightforward feeding relationship, whereby syntax is cyclically interpreting words as phonological sequences in a bottom up way, and larger constituents containing these are then cyclically interpreted as prosodic structures (which themselves might be cyclicly interpreted as articulatory scores). Another is that the constraints of each cyclic system are combined into a single system. This permits bidirectional informational interactions between the domains, but runs into the problem of opacity - the representations some constraints are stated over (the prosodic structure, say) is no longer present in the output.
Syn🚲Sem This project is focussed on non-local interactions over perhaps multiple cycles, in a more semantic domain. Of particular interest is the distinction between 'top-down' and 'bottom-up' effects, which at least in its nomenclature suggests that an attempt to move the purview of the top-down effects into the performance systems (where top-down parsing is sometimes called 'predictive parsing') might be of some value. Both top-down and bottom-up effects are as described in the project non-local, and thus pose challenges for cyclicity. They are conjectured to behave quite differently from one another, which is the beginning of a taxonomy of non-local effects.
Syn🚲Mor The project on morphosyntactic number plans to compare harmonic serialist and stratal OT analysis to one another in the domain of under- and overexponence. Moreover, this is done with the backdrop of a broadly minimalist syntax. This project thus makes use of the formal theories the graduate students assigned to this project will be developing, which provides for useful checks and balances on the formal side.

2.6. Descriptions of proposed investigations involving experiments on humans or human materials

not applicable

2.7. Information on scientific and financial involvement of international cooperation partners

not applicable

3. Bibliography

E. Bakovič. *Harmony, dominance and control*. PhD thesis, Rutgers, 2000.

J. Chandlee and J. Heinz. Strict locality and phonological maps. *Linguistic Inquiry*, 49(1):23–60, 2018.

G. Chierchia. Broaden your views: Implicatures of domain widening and the "logicality" of language. *Linguistic Inquiry*, 37(4):535–590, 2006.

N. Chomsky. *The Minimalist Program*. MIT Press, Cambridge, Massachusetts, 1995.

N. Chomsky. On phases. In R. Freidin, C. P. Otero, and M. L. Zubizarreta, editors, *Foundational Issues in Linguistic Theory*, pages 133–166. MIT Press, Cambridge, Massachusetts, 2008.

N. Chomsky and M. Halle. *The Sound Pattern of English*. MIT Press, Cambridge, Massachusetts, 1968.

H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available at `http://www.grappa.univ-lille3.fr/tata`, 2002.

H. Dolatian. *Computational locality of cyclic phonology in Armenian*. PhD thesis, Stony Brook University, 2020.

T. M. Ellison. Phonological derivation in optimality theory. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, volume 2, pages 1007–1013, 1994.

G. Fanselow and D. Čavar. Remarks on the economy of pronunciation. In G. Müller and W. Sternefeld, editors, *Competition in Syntax*, volume 49 of *Studies in Generative Grammar*, pages 107–150. Mouton de Gruyter, Berlin, 2001.

R. Frank and G. Satta. Optimality theory and the generative complexity of constraint violability. *Computational Linguistics*, 24(2):307–315, 1998.

T. Graf. Closure properties of minimalist derivation tree languages. In S. Pogodalla and J.-P. Prost, editors, *LACL 2011*, volume 6736 of *Lecture Notes in Artificial Intelligence*, pages 96–111, 2011.

T. Graf. *Local and transderivational constraints in syntax and semantics*. PhD thesis, UCLA, 2013.

Y. Hao. Harmonic serialism and finite-state optimality theory. In F. Drewes, editor, *Proceedings of the 13th International Conference on Finite State Methods and Natural Language Processing*, pages 20–29, 2017.

F. Heck and G. Müller. Extremely local optimization. In H. Broekhuis and R. Vogel, editors, *Linguistic Derivations and Filtering*, pages 135–166. Equinox, Sheffield, 2013.

I. Heim and A. Kratzer. *Semantics in Generative Grammar*. Blackwell Publishers, 1998.

J. Heinz. The computational nature of phonological generalizations. In L. Hyman and F. Plank, editors, *Phonological Typology*, volume 23 of *Phonology and Phonetics*, pages 126–195. De Gruyter, 2018.

J. Heinz and W. Idsardi. What complexity differences reveal about domains in language. *Topics in Cognitive Science*, 5(1):111–131, 2013.

A. Jardine. Computationally, tone is different. *Phonology*, 33:247–283, 2016.

C. D. Johnson. *Formal Aspects of Phonological Description*. Mouton de Gruyter, 1972.

A. K. Joshi and Y. Schabes. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 2, pages 69–124. Springer, 1997.

R. M. Kaplan and M. Kay. Regular models of phonological rule systems. *Computational Linguistics*, 20 (3):331–378, 1994.

L. Karttunen. The proper treatment of optimality in computational phonology. In L. Karttunen and K. Oflazer, editors, *Proceedings of the International Workshop on Finite State Methods in Natural Language Processing*, pages 1–12, 1998.

J. Kaye, J. Lowenstamm, and J.-R. Vernaud. Constituent structure and government in phonology. *Phonology*, 7:193–231, 1990.

S. Kepser and U. Mönnich. Closure properties of linear context-free tree languages with an application to optimality theory. *Theoretical Computer Science*, 354:82–97, 2006.

P. Kiparsky. Opacity and cyclicity. *The Linguistic Review*, 17(2-4):351–365, 2000.

G. M. Kobele. Minimalist tree languages are closed under intersection with recognizable tree languages. In S. Pogodalla and J.-P. Prost, editors, *LACL 2011*, volume 6736 of *Lecture Notes in Artificial Intelligence*, pages 129–144, 2011.

G. M. Kobele. LF-copying without LF. *Lingua*, 166, part B:236–259, 2015.

G. M. Kobele and J. Michaelis. Disentangling notions of specifier impenetrability: Late adjunction, islands, and expressive power. In M. Kanazawa, A. Kornai, M. Kracht, and H. Seki, editors, *The Mathematics of Language*, volume 6878 of *Lecture Notes in Computer Science*, pages 126–142. Springer, 2011.

G. M. Kobele, C. Retoré, and S. Salvati. An automata theoretic approach to minimalism. In J. Rogers and S. Kepser, editors, *Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07*, Dublin, 2007.

H. Koopman. Derivations and complexity filters. In A. Alexiadou, E. Anagnostopoulou, S. Barbiers, and H.-M. Gärtner, editors, *Dimensions of Movement: From features to remnants*, number 48 in Linguistik Aktuell/Linguistics Today, chapter 8, pages 151–188. John Benjamins, 2002.

J. McCarthy. An introduction to harmonic serialism. *Language and Linguistics Compass*, 4(10):1001–1018, 2010.

J. Michaelis. *On Formal Properties of Minimalist Grammars*. PhD thesis, Universität Potsdam, 2001.

J. Michaelis, U. Mönnich, and F. Morawietz. On minimalist attribute grammars and macro tree transducers. In C. Rohrer, A. Rossdeutscher, and H. Kamp, editors, *Linguistic Form and its Computation*, pages 51–91. CSLI Publications, 2001.

G. A. Miller and N. Chomsky. Finitary models of language users. In R. D. Luce, R. R. Bush, and E. Galanter, editors, *Handbook of mathematical psychology*, chapter 13, pages 419–491. John Wiley, New York, 1963.

R. Montague. English as a formal language. In *Formal Philosophy: Selected Papers of Richard Montague*, chapter 6, pages 188–221. Yale University Press, New Haven, 1974. edited and with an introduction by R. H. Thomason.

G. Müller. Das pronominaladverb als reparaturphänomen. *Linguistische Berichte*, 182:139–178, 2000.

M. Nilsson. *Regular Model Checking*. PhD thesis, Uppsala University, 2005.

J. Padgett. Partial class behavior and nasal place assimilation. In K. Suzuki and D. Elzinga, editors, *Proceedings of the 1995 Southwestern Workshop on Optimality Theory*, pages 145–183, Tuscon, 1995. Dept of Linguistics, University of Arizona.

P. S. Peters and R. W. Ritchie. On the generative power of transformational grammar. *Information Sciences*, 6:49–83, 1973.

A. Prince and P. Smolensky. *Optimality Theory: Constraint interaction in generative grammar*. Basil Blackwell, Oxford, 2004.

P. Resnik. Left-corner parsing and psychological plausibility. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, Nantes, France, 1992.

J. A. Riggle. *Generation, Recognition, and Learning in Finite State Optimality Theory*. PhD thesis, UCLA, 2004.

R. Sproat. *Morphology and Computation*. MIT Press, Cambridge, Massachusetts, 1992.

E. P. Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer-Verlag, Berlin, 1997.

M. Steedman. *The Syntactic Process*. MIT Press, 2000.

P. Wadler. Deforestation: Transforming programs to eliminate trees. *Theoretical Computer Science*, 73: 231–248, 1990.

A. Yli-Jyrä. Forgotten islands of regularity in phonology. In B. Gyuris, K. Mády, and G. Recski, editors, *K+K=120: Papers dedicated to László Kálmán and András Kornai on the occasion of their 60th birthdays*, pages 541–559. MTA Research Institute for Linguistics, Budapest, 2019.