Commonality in Disparity: The Computational View of Syntax and Phonology

Keywords: syntax, phonology, computational linguistics, structural complexity, constraints, locality

Thomas Graf & Jeffrey Heinz

Overview Heinz and Idsardi (2013) draw attention to a computational difference between syntax and phonology established by earlier research: phonology only requires regular computations over strings (Johnson 1972; Kaplan and Kay 1994), whereas syntax involves non-regular computations over strings (Chomsky 1956; Shieber 1985). In the present work, the computational difference between these two domains is studied from a perspective which recognizes two dimensions of formal grammars: the nature of the model (the data structures) and the power of the computations that manipulate these structures. We argue that the differences between syntax and phonology can be recast entirely in terms of the differences in the data structures rather than the power of the computations. Both phonological and syntactic dependencies turn out to be local given a suitable domain of relativization, which we formalize in terms of tiers.

Tier-Local Phonology Many dependencies in phonology have an upper bound on the number of nodes they may cross. Such local dependencies can be regulated via n-grams, i.e. substrings that consist of n segments. If, say, underlying /nb/ is pronounced [mb], then this can be captured by the surface constraint *nb or the corresponding bigram *nb*. Dependencies that are not locally bounded do not fit this pattern — no matter what threshold n one picks, it will be exceeded by some long-distance dependencies. Heinz et al. (2011) argue that long-distance dependencies are nonetheless local in a more relaxed sense: they are *tier-based strictly local*. Suppose phonological structures also include tiers, which contain only certain parts of the pronounced string. Given a sibilant tier T, for example, a ban against s following ʃ is enforced by the T-bigram ʃs. Tiers thus render non-local dependencies local by increasing the complexity of the data structure.

Minimalist Grammars Every proof that syntactic dependencies are tier-local must build on a rigorous model of syntax. In an effort to stay close to the syntactic consensus, we pick Minimalist grammars (MGs; Stabler 1997) as a formalization of Minimalist syntax. MGs assemble trees from feature-annotated lexical items (LIs) via Merge and Move, which are triggered by the LIs' features. As phrase structure trees are the result of applying Merge and Move operations in a specific order, they can be equated with their *derivation trees*. A derivation tree looks almost exactly like the bare phrase structure tree that is built from it, except that I) interior nodes are labeled by the operation taking place, and II) phrases undergoing movement remain *in situ* (cf. Fig. 1). Note that Move nodes are unary branching because Move is a deterministic operation in MGs, so there is no need to explicitly indicate which phrase is moving (empirically mandated non-determinism is modeled as a non-deterministic choice between different LIs).

Building on the insight that every MG can be identified with its set of well-formed derivation trees, Graf (2012) shows that these derivations are fully characterized by a few tree-geometric constraints. We only focus on the much more complex constraints for Move here, and we assume for the sake of simplicity that every phrase moves at most once (which does not decrease the power of the formalism). Given an LI l that needs to undergo movement in order to check its feature f, the *occurrence of l* is the lowest Move node that dominates l and can check f. A derivation contains no illicit instances of Move iff it obeys the following two constraints:

Move1 If an LI has a feature that triggers its movement, then it also has a Move occurrence.
Move2 Every Move node is an occurrence of exactly one LI.

Tier-Local Syntax When viewed from the perspective of constraints on derivation trees, Merge dependencies are easily shown to be local thanks to the local nature of subcategorization. Move dependencies, on the other hand, are not locally bounded because there is no absolute limit on how many nodes a moving phrase may cross. They do incorporate a relative notion of locality,

though, as the definition of occurrence refers to the **closest** dominating Move node that can check the relevant feature. It is this restriction that makes Move local.
First, we generalize the notion of tiers from strings to trees. A string can be viewed as a 1-dimensional object that is obtained by ordering 0-dimensional objects, i.e. nodes, via precedence. A tree, in turn, is the result of ordering 1-dimensional objects — the strings of siblings — via dominance. A *tree-tier*, then, is a 2-dimensional tier: string tiers are ordered via dominance. In the case of MGs, this idea is implemented as follows. Given a derivation tree, every LI with movement feature f and every Move node that can check f is projected onto an f-tier. Note that the projection step preserves the dominance relations between these nodes. Within the f-tier, every string of siblings also projects a lexical tier, which lists only LIs (cf. Fig. 1). The Move configurations banned by **Move1** and **Move2** are expressed by four templates, where the daughter string references the lexical tier. This establishes that Move dependencies, just like long-distance dependencies in phonology, are local over an enriched data structure with tiers.

Move1		**Move2**	
"LIs need a mother"	"mother must be Move"	"at least one occurrence"	"at most one occurrence"
\$	LI	Move	Move
|	|	|	|
LI	LI	\$—\$	LI—LI

Conclusion Taking as our vantage point the linguistically plausible assumption that phonology operates over strings and syntax over trees, we unearthed a surprising similarity between the two: non-local dependencies between nodes/segments are local within some suitable relativization domain, formalized via tiers. This is not just a simple coding trick such that every non-local dependency is local over a suitable choice of tiers — tier-based formalisms are provably weaker than what can be done with some (empirically unattested) regular and supra-regular computations. A linguistically informed choice of data structure thus highlights profound parallels in the computational complexity of phonological and syntactic dependencies.

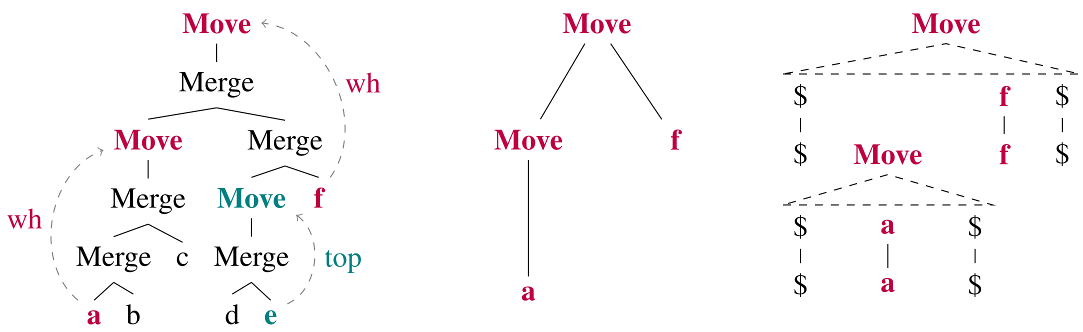

Figure 1: Left: derivation tree with expository Move arrows; Middle: nodes on wh-movement tier ordered by dominance; Right: wh-movement tier with lexical tiers above each sequence of siblings

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory* 2:113–124. ● **Graf**, Thomas. 2012. Locality and the complexity of minimalist derivation tree languages. In *Formal Grammar 2010/2011*, *LNCS* 7395, 208–227. ● **Heinz**, Jeffrey, and William Idsardi. 2013. What complexity differences reveal about domains in language. *Topics in Cognitive Science* 5:111–131. ● **Heinz**, Jeffrey, Chetan Rawal, and Herbert G. Tanner. 2011. Tier-based strictly local constraints in phonology. In *Proceedings of 49th Annual Meeting of the ACL*, 58–64. ● **Johnson**, C. Douglas. 1972. *Formal aspects of phonological description*. The Hague: Mouton. ● **Kaplan**, Ronald M., and Martin Kay. 1994. Regular models of phonological rule systems. *Computational Linguistics* 20:331–378. ● **Shieber**, Stuart M. 1985. Evidence against the context-freeness of natural language. *Linguistics and Philosophy* 8:333–345. ● **Stabler**, Edward P. 1997. Derivational minimalism. In *Logical aspects of computational linguistics*, *LNCS* 1328, 68–95.