Unfortunately the pdf and code for the poster were lost!

A Hidden Consensus: Computational Invariants of Minimalist Syntax

***Keywords*: computational linguistics, Minimalist Grammars, derivation trees, features, movement, constraints**

Overview A common sentiment among linguists is that the Minimalist literature features a dazzling array of competing proposals that seem to share little common ground in their technical assumptions. While differences certainly do exist between accounts, a computationally informed perspective reveals a set of invariants that unify Minimalist analyses and differentiate them from competing proposals such as HPSG and LFG. These invariants have been studied extensively by the Minimalist grammar community, but have mostly gone unnoticed by syntacticians. Taking these findings as my vantage point, I argue that Minimalist syntax can be understood as a theory of derivation trees and corresponding mappings to interface representations, with the restriction that both derivations and mappings are computable with a finite amount of working memory. Each Minimalist analysis occupies a specific point within this class, depending on which ancillary assumptions it adopts, and we can measure the distance between two proposals in this space according to certain formal metrics of equivalence.

Minimalist grammars Minimalist grammars (MGs) were introduced by Stabler (1997) as a bare bones formalization of pre-Agree Minimalist syntax. Standard MGs only use the binary operations Merge and (phrasal) Move, both of which must be triggered by features of opposite polarity on the heads of their two arguments. It is also assumed that the features of a given head must be checked in a specific order, similar to the model in Müller (2010). An MG might derive, say, *car the red* by first merging *car* as the argument of *red*, which in turn is selected by *the*. After that, *car* moves to the specifier of *the* to check some feature *f* and thus produces the desired output order. This can be written more succinctly as the term $move\big(merge(\text{the} :: A^+f^+D^-, merge(\text{red} :: N^+A^-, \text{car} :: N^-f^-)), \text{car} :: N^-f^-\big)$, which is more easily represented as a graph (see Fig .1). These graphs are called *derivation trees* in the MG literature.

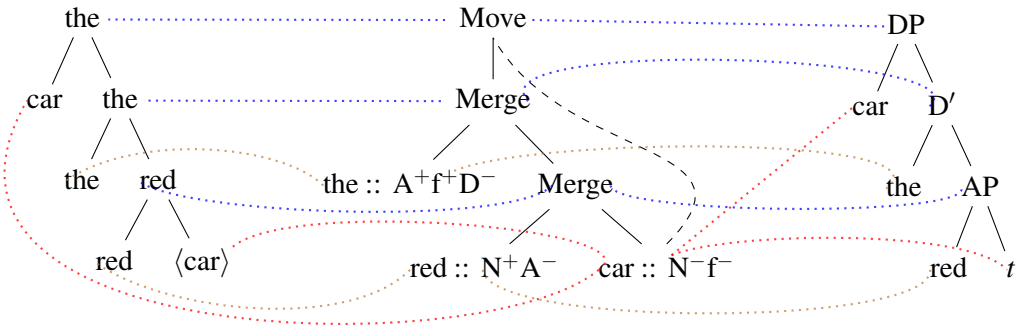

Figure 1: Minimalist derivation tree and mappings to two kinds of phrase structure trees

Every MG is fully characterized by its set of well-formed derivation trees (Michaelis 2001; Kobele et al. 2007). If one knows which derivations are licit given a specific MG, one also knows which output structures it licenses — phrase structures trees, LF, PF, string yield, and so on. Furthermore, the set of well-formed derivation trees can be computed with a finite amount of working memory as long as there is a finite bound on how many phrases are allowed to move to a given target position. In this case it also holds that the mapping from the derivation tree to the corresponding output structure can be computed with a finite amount of working memory. This means that MGs consist of two highly restricted components: a set of well-formed derivation trees, and a translation of these derivations into their output structures, both of which require only a finitely bounded amount of working memory.

Three Measures of Equivalence A lot of MG research is concerned with proving that two MG variants are equivalent in the sense that one can be translated into the other without changing the set of output strings (*w[eak]-equivalence*), output structures (*s[trong]-equivalence*), or possibly even the shape of the derivation trees (*d[erivational]-equivalence*). These measures can be aligned along a continuum from E-language to I-language such that w-equivalence is closest to E-language, d-equivalence closest to I-language, and s-equivalence lies between the two. Note also that s-equivalence always implies w-equivalence, and d-equivalence implies s-equivalence if the mapping from derivations to output structures is held constant across grammars. This allows us to establish a similarity ranking between Minimalist analyses depending on their degree of equivalence.

Selected Invariants The list of discovered invariants is enormous, even at the level of d-equivalence. For instance, standard MGs bifurcate the set of features into those that only trigger Merge, and those that only trigger Move. But this distinction can be done away with without affecting the shape of derivation trees. On the other hand, grammars with ordered features are s-equivalent to features with unordered features, but not d-equivalent because the feature order must be decomposed into a hierarchy of functional heads which are erased during the mapping to output structures (Graf 2013:106f). Finally, grammars with privative, binary, or attribute-valued feature systems (cf. Adger 2010) are d-equivalent, but none of them are even w-equivalent to grammars with recursive feature structures in the vein of HPSG.

A much more surprising d-equivalence holds between MGs with various types of constraints. As shown in Graf (2013), MGs with subcategorization constraints, MGs with local constraints (e.g. restricted to phase), MGs with non-local constraints, and MGs with transderivational constraints are all d-equivalent. Since subcategorization constraints can be easily replaced by features triggering Merge, even the dichotomy of features and constraints turns out to be immaterial with respect to d-equivalence. And since Agree can be recast as a constraint, this also implies that the addition of Agree preserves d-equivalence.

Things are more involved with respect to movement. Graf (2012) shows that the restriction to finite working memory mappings allows for much more movement types and includes in fact all that have been proposed in the literature, in particular head movement and sidewards movement. The addition of these new movement types does not increase weak generative capacity, but it may allow for new tree structures and derivations, so that s-equivalence and d-equivalence do not hold.

Every MG can also be translated into a w-equivalent one where every phrase moves at most once. In fact, these grammars are almost d-equivalent, the only difference being the absence of intermediate landing sites. These can be inserted by the output mapping, so that both s- and d-equivalence holds.

Finally, grammars are trivially d-equivalent if they only differ in the mapping to output structures. Hence it does not matter whether the grammar builds multi-dominance trees, trees with (possibly pronounced) copies, trees with traces, bare-phrase structure sets, sets without labels, the derivation always stays the same (nor is there a noteworthy difference in working memory usage). Since the derivation is the central locus of information, this entails that all these different variants have the same complexity and power.

Conclusion Although Minimalist syntax comes in many different flavors, it has become increasingly clear over the last 15 years that they fit comfortably within the bounds of MGs, understood as a combination of two specific types of objects: a set of well-formed derivations that can be computed with a finite amount of working memory, and a finite working memory translation from derivations to output structures. This basic characterization unifies all Minimalist analysis and differentiates them from HPSG and LFG, among others, which require unbounded memory due to their complex feature structures.

There is still a lot of variability within the class, but we can measure the distance between two proposals by their degree of equivalence, with d-equivalence being the strongest notion to be studied so far. It turns out that many contentious issues in the literature, such as the choice between features and constraints, are irrelevant with respect to d-equivalence, whereas for instance the choice of movement type has a noticeable effect even on what tree structures can be generated (s-equivalene) but not on the output strings (w-equivalence). Although d-equivalence may not be the perfect metric as it still ignores certain factors like the size of the lexicon, it is a lot more adequate than s-equivalence and w-equivalence by virtue of reflecting the grammars' *modus operandi*. Therefore d-equivalence should be taken more seriously than the lesser two degrees of equivalence and can indeed provide helpful clues as to which technical differences are more important than others.

Adger, David. 2010. A minimalist theory of feature structure. In *Features: Perspectives on a key notion in linguistics*, ed. Anna Kibort and Greville G. Corbett, 185–218. Oxford: Oxford University Press. • **Graf**, Thomas. 2012. Movement-generalized minimalist grammars. In *LACL 2012*, ed. Denis Béchet and Alexander J. Dikovsky, volume 7351 of *Lecture Notes in Computer Science*, 58–73. • — 2013. *Local and transderivational constraints in syntax and semantics*. Doctoral Dissertation, UCLA. • **Kobele**, Gregory M., Christian Retoré, and Sylvain Salvati. 2007. An automata-theoretic approach to minimalism. In *Model Theoretic Syntax at 10*, ed. James Rogers and Stephan Kepser, 71–80. • **Michaelis**, Jens. 2001. Transforming linear context-free rewriting systems into minimalist grammars. *Lecture Notes in Artificial Intelligence* 2099:228–244. • **Müller**, Gereon. 2010. On deriving CED effects from the PIC. *Linguistic Inquiry* 41:35–82. • **Stabler**, Edward P. 1997. Derivational minimalism. In *Logical aspects of computational linguistics*, ed. Christian Retoré, volume 1328 of *Lecture Notes in Computer Science*, 68–95. Berlin: Springer.