Late Merge as Lowering Movement in Minimalist Grammars

Thomas Graf

Department of Linguistics
Stony Brook University
mail@thomasgraf.net
http://thomasgraf.net

Abstract. Minimalist grammars can be specified in terms of their derivation tree languages and a mapping from derivations to derived trees, each of which is definable in monadic second-order logic (MSO). It has been shown that the linguistically motivated operation Late Merge can push either component past the threshold of MSO-definability. However, Late Merge as used in the syntactic literature can be elegantly recast in terms of Lowering movement within the framework of Movement-generalized Minimalist grammars. As the latter are MSO-definable, the linguistically relevant fragment of Late Merge is too.

Keywords: Minimalist grammars, countercyclic operations, late merge, lowering movement, monadic second-order logic

Introduction

Minimalist grammars (MGs; [24, 25]) were conceived as a formalization of the currently dominant framework in theoretical syntax [1], so it is hardly surprising that they have frequently been used as a testbed for the ideas and mechanisms entertained by linguists. In recent years, a lot of work has focused on countercyclic operations, in particular Late Merge [4, 10, 14]. Under a derivational conception of syntax, an operation is countercyclic iff any new material it introduces is not added to the top of the structure built so far but instead inserted at a lower position.

Countercyclic operations are intriguing from the perspective of a resource-sensitive formalism like MGs, where every lexical item has a finite number of features that must be checked. For one thing, feature checking — i.e. Minimalist resource consumption — is no longer linked to structure-building in a one-to-one fashion because the feature triggering a countercyclic operation may actually license the insertion of new structure at a very different position. More importantly, though, the material introduced by a countercyclic operation might itself contain features that need to be checked, so that a substructure that already had all of its features checked may suddenly be in need of feature checking again.

Both possibilities add a lot of complexity to the formalism. It is already known that Late Merge can increase weak generative capacity [14], and even

when Late Merge is sufficiently constrained to preserve strong generative capacity it requires a more powerful mapping from derivations to derived trees [4, 10, 18]. MGs can be specified in terms of their derivation tree languages and a mapping from derivation to derived tree languages [5, 8, 15]. For standard MGs, both components are definable in monadic second-order logic (MSO). Neither holds for MGs with unrestricted Late Merge [10].

While the formal underpinnings of unrestricted Late Merge are certainly interesting and provide deep insights into the inner workings of MGs, the loss of MSO-definability and the computational advantages that come with it should be avoided unless absolutely necessary. I argue in this paper that for the cases discussed so far in the linguistic literature, Late Merge can be easily reduced to the more standard operation of Lowering movement [6]. Since Lowering is MSO-definable, so must be Late Merge. In a certain sense, the reduction provides a normal form theorem for MGs — results on Lowering immediately carry over to Late Merge. For example, if the MG parser in [26, 27] is enriched with inference rules for Lowering, it will also work for grammars with Late Merge.

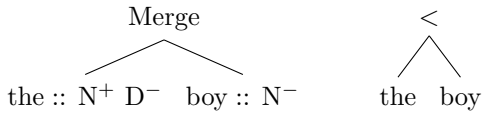
The paper is laid out as follows: Section 1.1 provides an intuitive introduction to MGs, which is subsequently expanded in Sec. 1.2 to Movement-generalized MGs, a weakly equivalent variant of MGs where grammars may have multiple types of movement, including Lowering. Section 1.3 then discusses Late Merge from a linguistic perspective. The reduction of Late Merge to Lowering is presented in Sec. 2. This is followed by technical observations on generative capacity and derivational complexity (3.1), limitations of the reduction (3.2), and the interaction of Late Merge and movement (3.3).

1 Preliminaries

1.1 Standard Minimalist grammars

MGs [24, 25] are a lexicalized grammar formalism where every lexical item has an ordered list of features that must be checked in this specific order by the operations Merge and Move, which in turn combine these lexical items into tree structures.

For instance, the DP *the boy* would be built by merging the lexical item *the* with *boy*. In order to license this application of Merge, *the* has a *selector feature* N^+ indicating that it selects a noun, and the noun *boy* has a matching *category feature* N^-. Moreover, since the entire phrase is a DP and *the* is the head of the DP, *the* also has a category feature D^-. The selector feature N^+ on *the* precedes the category feature D^- because *the* must select its argument before it can project a DP. The structure-building process can be depicted as a derivation tree, which closely mirrors the structure of the derived tree.



The main difference is the absence of features in the derived tree as well as the new label for the interior node, which points in the direction of the projecting head. This is always the head that had one of its positive polarity features (i.e. a feature with superscript +) checked by the operation in question.

In the sentence *the boy John likes*, the object DP *the boy* has been topicalized, which is analyzed as movement from the position where it was merged to a higher landing site. A simplified derived tree and its derivation are given in Fig. 1. Here

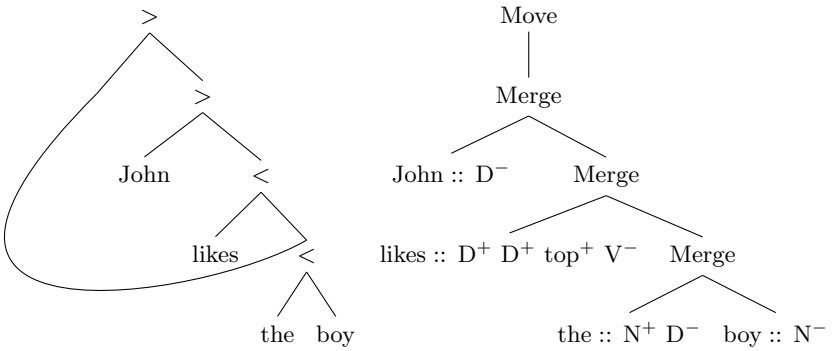


Fig. 1. Multi-dominance tree with movement and its corresponding derivation tree

the boy is assumed to move to a specifier of the verb *likes*. This is licensed by the *licensor feature* top^+ on *likes* and the *licensee feature* top^- on *the*. A few things are worth pointing out explicitly:

- Movement in the derived tree is indicated by adding branches. This yields a directed acyclic graph, which is called a *multi-dominance tree* in the syntactic literature. This format of representing movement is formally simpler than the standard trace-based mechanism.
- Once again the derivation tree is very similar to the derived tree. The latter can be obtained from the former as before by relabeling interior nodes and removing features, except that this time one must also add movement branches.
- Even though the licensee feature triggering topicalization is hosted by *the*, it isn't just the determiner that undergoes movement but the entire DP it projects.

Movement is subject to the Shortest Move Constraint (SMC), which blocks all configurations where LIs two LIs have the same licensee feature as their first unchecked feature. For example, if *John* in the derivation above actually had the feature specification D^- top^-, then the derivation would be aborted after *John* is merged because then both *John* and *the* would have the licensee feature top^- as their first unchecked feature.

The SMC grants MGs a variety of appealing properties. Their derivation tree languages are regular tree languages and thus definable in MSO [5, 15, 17]. The mapping from derivation trees to derived trees is also definable in MSO, which entails that MGs generate mildly context-sensitive string languages [6, 8, 15, 18, 19]. This result can be strengthened: MGs are weakly equivalent to MCFGs [9, 17].

1.2 Movement-Generalized Minimalist Grammars

Movement-generalized MGs (MGMGs) were introduced by Graf in [6] and allow for new types of movement instead of just upward movement of a phrase to a c-commanding position. Crucially, though, all movement types must be definable in MSO, which implies via a battery of previously known theorems that MGs and MGMGs are weakly (but not strongly) equivalent. A full definition of MGMGs is rather laborious and unnecessary for the purposes of this paper, so I restrict myself to an abridged version here; the interested reader is referred to [6].

MGMGs build on the insight that the MG feature calculus driving Move can be recast in tree-geometric terms. Basically, one has to ensure that every Move node is targeted by exactly one moving phrase, and every licensee feature on every LI is checked by exactly one Move operation. This amounts to regulating the distribution of Move nodes in the derivation.

The essential notion is that of *occurrences*. Given a lexical item l and node m in the derivation, m is the i-th occurrence of l iff it checks l's $(i+1)$-th negative polarity feature. Hence the *zero occurrence* of a lexical item l is the Merge node that checks its category feature and thus introduces it into the derivation. *Positive occurrences* (for which $i > 0$) are the Move nodes that check one of l's licensee features. The requirements of MG feature calculus with respect to Move can then be enforced as constraints on the distribution of occurrences. For every derivation tree t, node m of t, and lexical item l with licensee features $f_1^- \cdots f_n^-$, $n \geq 0$:

Move there exist distinct nodes $m_1, \ldots, m_n$ such that m_i (and no other node of t) is the i^{th} occurrence of l, $1 \leq i \leq n$.

SMC if m is a Move node, there is exactly one lexical item that m is an occurrence of.

But how does one determine whether a Move node is an occurrence for a lexical item? In the case of standard MGs, there is a simple algorithm that depends only on proper dominance.

occurrence A Move node m is the i-th occurrence of a lexical item l with licensee features $f_1^- \cdots f_n^-$ iff

- m is associated to the matching licensor feature f_i^+, and
- m properly dominates the $(i-1)$-th occurrence of l, and
- there is no Move node z such that m properly dominates z and z satisfies the previous two clauses.

Intuitively, the i-th occurrence of l is found by searching upwards from the $(i-1)$-th occurrence until one encounters a Move node that can check f_i^- on l.

Graf notes that new movement types can be created by replacing proper dominance by a different relation over trees. For example, the inverse of proper dominance gives rise to downward movement since now the i-th occurrence is dominated by the $(i-1)$-th occurrence rather than the other way round. Since the constraints **Move** and **SMC** make no assumptions as to which nodes may count as occurrences, they restrict this kind of movement in the same fashion as standard movement. Thus new types of movement can be introduced without altering the spirit of movement regarding the MG feature calculus.

As long as the relation connecting occurrences is MSO-definable, **Move** and **SMC** will be, too, which implies that MGMGs still have regular — that is, MSO-definable — derivation tree languages. The mapping from derivations to derived trees is also built over the notion of occurrences, so it need not be altered either. Graf allows for every feature to specify whether the moving phrase should be linearized to the left or to the right of the landing site, but this is a technically innocent change without negative repercussions. The weak generative capacity of MGMGs therefore does not exceed that of standard MGs, yet they provide a simple mechanism for adding new movement types that can generate more complex tree languages.

Building on the MGMG system, I will use two types of movement in this paper. *Raising* is the standard kind of MG movement. That is to say, the phrase targets a c-commanding position in the derived tree, which corresponds to using proper dominance for determining occurrences. *Lowering* is the inverse of raising: a phrase moves to a c-commanded position, and occurrences are determined by the inverse of proper dominance. Raising and Lowering will be indicated by the prefixed subscripts $\uparrow$ and $\downarrow$, respectively. Moreover, the prefixed superscripts λ and ρ indicate whether a mover is linearized to the left or the right. So ${}^{\rho}_{\downarrow}f^-$, for instance, is a licensee feature triggering rightward Lowering. A (linguistically implausible) example is given in Fig. 2.

1.3 Late Merge

In the Minimalist literature, *Late Merge* refers to cases where a tree s is merged with a tree t at some point after t has already been merged with another tree u. The best known instance of this is in Lebeaux's analysis of Principle C exceptions [16].

(1) a. * He$_i$ believed the argument that John$_i$ made.
b. Which argument that John$_i$ made did he$_i$ believe?

In the ungrammatical (1a) the R-expression *John* is c-commanded by the co-indexed DP *he*, which constitutes a Principle C violation. This part is expected; the puzzle is why (1b) is grammatical. Since *which argument that John$_i$ made* started out as the object of *believe*, *John* was c-commanded by a co-indexed DP at some point during this derivation, too, so a Principle C violation should

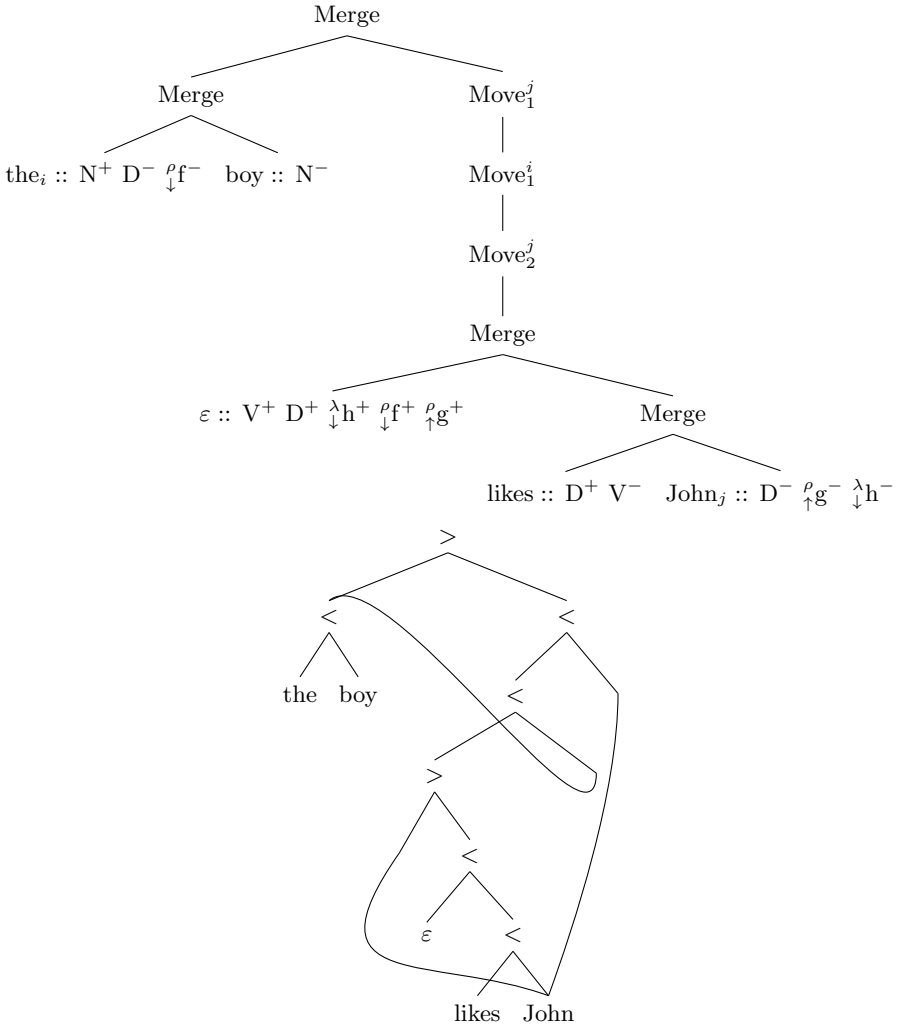


Fig. 2. An implausible derivation of *John likes the boy* with Raising and Lowering; occurrences are distinguished by superscripts i and j and numbered with subscripts

obtain as before. Lebeaux contends that the violation can be avoided because the relative clause *that John made* does not have to merge with *which argument* until the latter has moved out of the c-command domain of *he*. Lebeaux's derivation can be roughly sketched as follows:

$$\text{did he}_i \text{ believe [which argument]} \tag{1}$$

$$\text{[which argument] did he}_i \text{ believe} \tag{2}$$

$$\text{[which argument [that John}_i \text{ made]] did he}_i \text{ believe} \tag{3}$$

No derivation resembling the one above is compatible with standard MGs. Combining the relative clause with *which argument* must involve a feature on *which* or *argument*. But after *which argument* undergoes movement, all its features have been discharged, it has become completely opaque as far as the formalism is concerned. Any implementation of Late Merge thus has to modify the way resources are handled in MGs, and the modification has to work for a variety of cases.

Lebeaux's account is just one of many where Late Merge is used to factor some subtree out of a domain where its presence would cause problems. Ochi [22] posits that VP-adjuncts undergo Late Merge because they would otherwise act as an intervener between a verb and the inflectional head T, which would incorrectly predict under his proposal that *John quickly left* is always realized as *John did quickly leave*. Nissenbaum [21] also has to assume that VP-adjuncts are merged late in order to avoid a type conflict which can only arise before the subject undergoes movement. And Stepanov [28] proposes that the phrase *to Mary* in *John$_i$ seems to Mary [t_i to be smart]* must be merged countercyclically in order to avoid a subjacency violation.

In all the cases above, only adjuncts undergo Late Merge, which is why the operation is also called *Late Adjoin* or *Late Adjunction*. However, there are accounts that extend Late Merge to arguments, too [29]. For instance, the LF-derivation for *every spy sneezed* could proceed along the following lines:

$$[\text{every}]_i \text{ sneezed} \tag{1}$$

$$[\text{every}]_i\ t_i \text{ sneezed} \tag{2}$$

$$[\text{every spy}]\ t_i \text{ sneezed} \tag{3}$$

Late Merger of the NP *spy* in this toy example serves little linguistic purpose. The real motivation for Late Merger of arguments stems from Principle C exceptions that are similar to the ones noted by Lebeaux for adjuncts, except that they now arise with NP-arguments of quantified DPs.

(2) a. * Which argument that John$_i$ is a genius seems to him$_i$ to be true?

b. Every argument that John$_i$ is a genius seems to him$_i$ to be true

Takahashi and Hulsey [29] argue that (2b) is well-formed because the presence of quantifier allows for the argument to be late-merged. The derivation proceeds

roughly as follows:

$$[\text{every}]_i \text{ true} \quad (1)$$

$$[\text{every}]_i \text{ seems to him}_i\ [t_i \text{ to be } [t_i \text{ true}]] \quad (2)$$

$$[[\text{every}]\ [\text{argument that John}_i \text{ is a genius }]] \text{ seems to him}_i\ [t_i \text{ to be } [t_i \text{ true}]] \quad (3)$$

Since this example involves a lot of structure and intermediate movement steps, the toy example *every spy sneezed* will be used in the next section for the sake of simplicity.

Even though we omitted many technical details, it should be clear that none of the cases above require something like Late Merge. In each case, it would be sufficient to modify the mechanism that conflicts with the subtree, rather than have the subtree be merged in a countercyclic fashion. Principle C could ignore adjuncts of a moving DP, Ochi's account of English inflection could stipulate that adjuncts do not count as interveners, and so on. Readers familiar with model-theoretic approaches such as the formalization of Government-and-Binding theory in terms of MSO [23] will also realize that making such modifications is easier than adding a completely new operation and restricting its application domain. So from a purely formal perspective, there is no compelling argument for Late Merge. Both weak and strong generative capacity of the standard formalisms are already sufficient for the phenomena above, and bringing in Late Merge in a controlled fashion such that the computational properties of the formalism are preserved just means extra work.

Crucially, though, weak and strong generation of natural languages is but one of many duties of a grammar formalism. The formalism also has to be capable of stating structural generalizations. This isn't just a matter of scientific methodology — the fewer generalizations the formalism makes, the greater the number of stipulations it requires, which implies a bigger grammar size. Small grammars show better parsing performance and are easier to maintain; so even if one only cares about the engineering aspects of a formalism, a mechanism that unifies a range of empirical phenomena is worth studying regardless of whether it is vacuous with respect to generative capacity.

2 Late Merge as Lowering

2.1 Late Merger of Arguments

A maximally faithful implementation of how linguists think about Late Merge in sentences like *every spy sneezed* would arguably involve an MG derivation like the one in Fig. 3. This derivation is ill-formed in three respects: *every* cannot be merged directly with *sneezed* given its feature make-up, *spy* cannot be merged with the empty C-head, and no Merge operation combines *every* and *spy*.

However, if one uses MGMGs as defined in [6], then the minimally different derivation in Fig. 4 is well-formed and produces the intended string *every spy sneezed*. The selector feature N^+ on *every* has been replaced by a rightward lowering licensor feature $\overset{\rho}{\downarrow}1^+$, and a corresponding Move node has been added

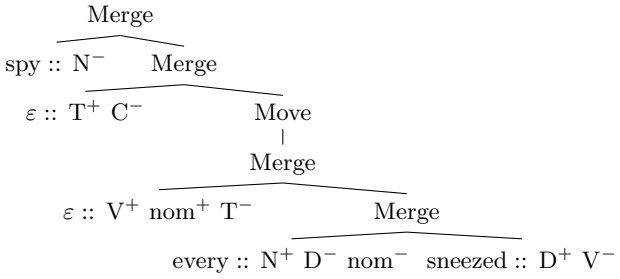


Fig. 3. A faithful candidate for a Late Merge derivation

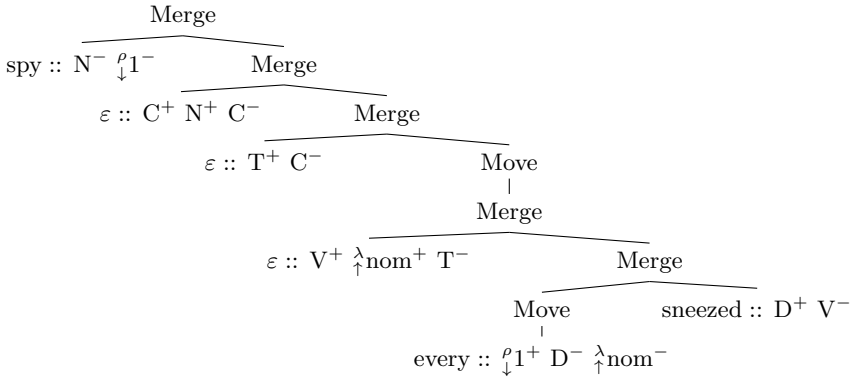


Fig. 4. A Lowering derivation for Late Merge

in the derivation. The matching licensee feature $\overset{\rho}{\downarrow}1^-$ occurs on *spy*. Hence *spy* will lower down to *every*, where it is linearized to its right. The other minor change is the presence of a second empty C-head, with the first C-head as its complement and *spy* as a specifier. This head only serves as a means for introducing *spy* into the derivation — there are many viable alternatives, for instance adjunction of *spy* to the first C-head using one of the many MG implementations of adjunction [2, 3]. The important point is that the countercyclic behavior of Late Merge is emulated by countercyclic movement, i.e. Lowering.

However, Lowering does not produce the intended structure under the standard MSO-transduction from derivation to multi-dominance trees given in [6, 8], as is evidenced by Fig. 5. Since movement is invariably linked to the introduction of movement branches in the standard transduction, Lowering gives rise to a structure where *spy* originates in a high position. This is at odds with linguists' intuitions, according to which *spy* has not been part of the structure at all before being merged in its surface position. An easy solution is at hand, though: a return to the original MG conception of movement in which one does not keep track of where a mover originated from. It is a simple task to modify

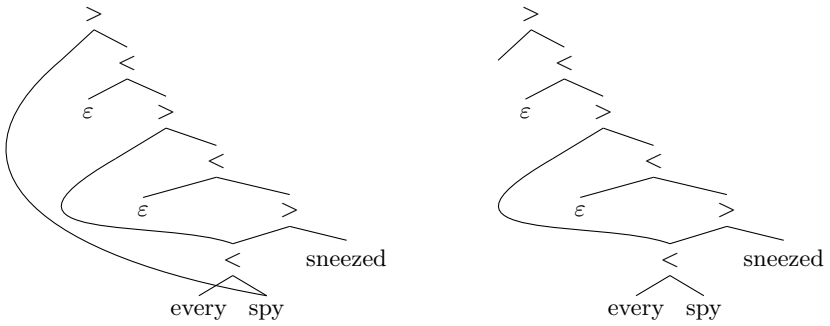


Fig. 5. Lowering with the standard transduction yields the wrong structure (left), but the original MG treatment of movement is adequate (right)

the MSO transduction such that Lowering does not introduce branches or leave behind traces. As a matter of fact, once could even delete the empty head that introduced the late-merged phrase into the derivation via a simple modification of the transduction defined in Sec. B.4.6 of [8].

Another point where the Lowering implementation differs slightly from linguistic intuition is that *spy* lowers before *every* moves into subject position. This is due to an explicit ban against licensee features preceeding licensor features in [6], wherefore *every* cannot move on its own to check one of its licensee features before furnishing a landing site for *spy* via one of its licensor features. One could lift this ban for the relevant cases here, but this isn't needed from an MG perspective. Since the derived tree structures have ancillary status at best in MGs and most work is done directly over derivations [8, 12, 13, 15], what matters

is that *spy* starts out at a higher position in the derivation. Its location in the derived tree is of interest only for computing the output string of the derivation.

2.2 Late Merger of Adjuncts

The strategy of emulating Late Merge via Lowering can be extended to adjuncts without much trouble. If only one adjunct is being late-merged, the corresponding derivation perfectly matches the one outlined in the previous section.

In principle, however, a phrase p may have an unbounded number of adjuncts, so one would expect that an unbounded number of adjuncts can be late-merged with p. But due to the SMC, the number of moving elements at any given step of the derivation is finitely bounded, so it seems that only a bounded number of adjuncts can be late-merged with p.

This issue has some linguistic relevance for if one follows Ochi and Nissenbaum [21, 22], then all VP adjuncts — of which there can be unboundedly many — are introduced by Late Merge. Crucially, though, VP adjuncts fall into two major categories determined by whether they are linearized to the left or to the right of the phrase they adjoin to. Suppose, then, that there are two empty heads h_l and h_r with Lowering licensee features ${}^{\lambda}_{\downarrow}1^-$ and ${}^{\rho}_{\downarrow}1^+$, respectively. Left adjuncts adjoin to h_l, whereas right adjuncts adjoin to h_r (this can be enforced with an MSO-constraint). Then h_l and h_r both lower to positions furnished by the phrase the adjuncts are supposed to late-merge with; an example with Frey and Gärnter adjunction [3] is given in Fig. 6.

The kind of "piggybacking" proposal advocated here captures the spirit of Late Merge while producing the right surface string. The derived tree it produces deviates from standard assumptions in that the adjuncts no longer c-command the phrase they are late-merged with. However, there are few cases where c-command matters for adjuncts. If for some reason adjunct a must c-command the phrase it adjoins to, a minor variant of Lowering may be added to the grammar that allows the adjunct to move to the landing site on its own even though the former does not c-command the latter. Alternatively, one could also switch the adjunction feature of a to a category feature and have the empty head select a directly. In this case lowering of a would proceed exactly as for late-merged arguments.

3 Technical Remarks

3.1 Generative Capacity and Derivational Complexity

The reduction of Late Merge to Lowering immediately entails the definability of the former in MSO and, by extension, that Late Merge does not increase the weak generative capacity of standard MGs. Their strong generative capacity is not increased, either, even though MGs with raising and lowering movement are more powerful than standard MGs in this respect (a corollary of [7]).

Generative capacity is preserved because Lowering movement by itself only generates regular tree languages, which is due to the impossibility of *remnant*

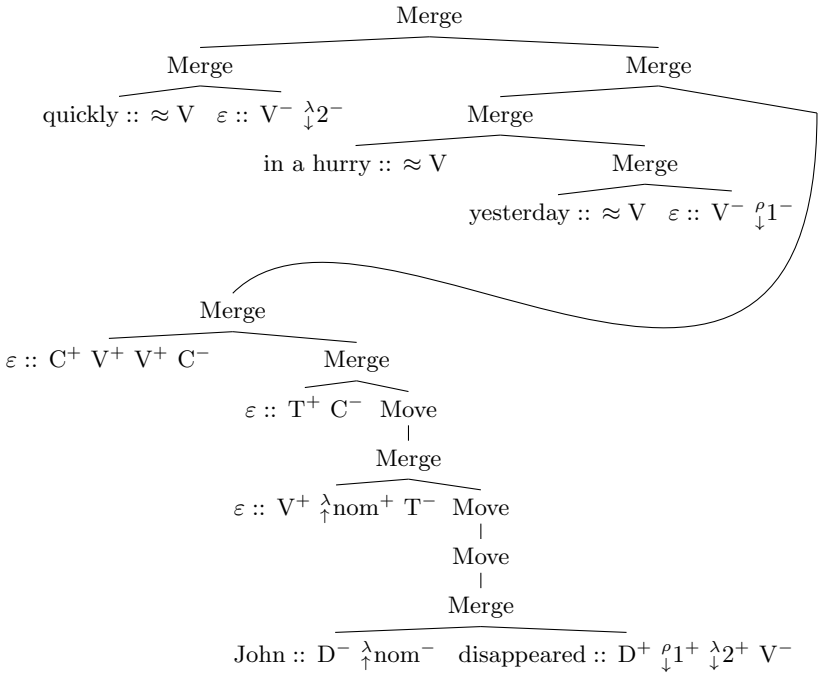


Fig. 6. Derivation (left) and derived tree (right) for Late Merge of multiple adjuncts

lowering (cf. [11]). Remnant movement refers to cases where a phrase X moves out of a containing phrase Y, which subsequently undergoes movement, too. This can never happen with Lowering because if X is contained in Y and undergoes Lowering, it is still contained in Y. Only if Lowering interacts with the standard raising movement is it possible to create such patterns. But no such interaction is required for Late Merge, and hence the dependencies it creates in the derived tree are all regular (in a certain sense, Late Merge corresponds to the simplest kind of Lowering a grammar may possess). With respect to generative capacity, then, Late Merge adds nothing to the standard formalism.

The complexity of the mapping from derivations to derived trees shows an increase, though, as was already noted in [4]. For standard MGs, this mapping is a *direction preserving* MSO transduction [18, 19]. That is to say, if x immediately dominates y in the derived tree, then x properly dominates y in the derivation tree. For grammars with Lowering instead of Raising, the opposite holds instead: if x immediately dominates y in the derivation tree, then x properly dominates y in the derived tree (this is different from *inverse direction preserving* MSO transductions, for which it holds that if x immediately dominates y in the derived tree, then y properly dominates x in the derivation tree [18, 20]). The addition of Lowering clearly entails that the mapping to derived trees is no longer direction preserving.

3.2 Formal Limitations and Linguistic Adequacy

It is important to keep in mind that only a subpart of Late Merge is emulated by Lowering. A completely unrestricted version of Late Merge is capable of increasing the weak generative capacity of MGs [14]. The strategy provided here inherits the major limitation of movement in MGs: only a finitely bounded number of items can undergo Late Merge at the same time.

Yet the limitation to a bounded number of Late Mergers does not seem to be problematic for linguistic applications. Late Merge is not delayed indefinitely but usually takes place within the smallest domain that properly subsumes the application domain of the constraint that the late-merged material has to escape, usually a CP. Given this assumption the only scenarios where one could possibly have an unbounded number of Late Merge operations involve adjuncts or recursion inside an argument, e.g. a DP whose NP selects a DP. But we have already seen that adjuncts can be limited to two instances of Late Merge, and there are no cases in the literature where, say, every NP of a recursive DP has some late-merged adjunct such that all of them originate outside the DP. As a matter of fact, there aren't even any cases of Late Merger with a late-merged phrase. Overall, then, linguists use only a fraction of the power a maximally general version of Late Merge could provide, and this fragment is easily recast in terms of Lowering.

3.3 Movement of Late-Merged Phrases

An unrestricted version of Late Merge increases the generative capacity of MGs as it makes it possible to introduce new licensee features into the derivation without violating the SMC. For example, if both a and b have a licensee feature f^- and they both are merged before the first f^+ occurs, the derivation is aborted due to the SMC. But if a is late-merged after f^- on b has been checked, then no SMC violation obtains and a can undergo movement once another f^+ has been introduced. Among other things, this grants MGs the power to generate tree languages with context-free path languages.

I have not said anything about the interaction of Late Merge with movement, for two reasons: I) the syntactic literature discusses no such cases, and II) once Late Merge is reduced to Lowering, movement of a late-merged phrase is regulated by the constraints MGMGs impose on Move [cf. 6]. In particular, the SMC holds for all movement types in MGMGs, so any configuration like the one above will always be blocked.

Conclusion

Emulating Late Merge via Lowering movement reduces an operation that is still not particularly well understood to the familiar MG mechanism of movement. Lowering is not powerful enough to handle everything that Late Merge is in principle capable of, but it is more than sufficient for the cases discussed in

the syntactic literature. So even though a lot of work remains to be done on Late Merge from a technical perspective, the reduction to Lowering shows that Late Merge as used by syntacticians is easily accommodated by MGs. Therefore a linguistically adequate treatment of Late Merge can be reconciled with MSO-definability in a straightforward fashion. Moreover, as we learn more about Lowering and its effects on MGs, our grasp of the linguistically relevant fragment of Late Merge will improve, too.

Bibliography

[1] Chomsky, N.: The Minimalist Program. MIT Press, Cambridge, Mass. (1995)

[2] Fowlie, M.: Order and optionality: Minimalist grammars with adjunction. In: Proceedings of MOL 2013 (2013), to appear

[3] Frey, W., Gärtner, H.M.: On the treatment of scrambling and adjunction in minimalist grammars. In: Proceedings of the Conference on Formal Grammar (FGTrento). pp. 41–52. Trento (2002)

[4] Gärtner, H.M., Michaelis, J.: A note on countercyclicity and minimalist grammars. In: Penn, G. (ed.) Proceedings of Formal Grammar 2003. pp. 95–109. CSLI-Online, Stanford (2008)

[5] Graf, T.: Locality and the complexity of minimalist derivation tree languages. In: de Groot, P., Nederhof, M.J. (eds.) Formal Grammar 2010/2011. Lecture Notes in Computer Science, vol. 7395, pp. 208–227. Springer, Heidelberg (2012)

[6] Graf, T.: Movement-generalized minimalist grammars. In: Béchet, D., Dikovsky, A.J. (eds.) LACL 2012. Lecture Notes in Computer Science, vol. 7351, pp. 58–73 (2012)

[7] Graf, T.: Tree adjunction as minimalist lowering. In: Proceedings of the 11^{th} International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11). pp. 19–27 (2012)

[8] Graf, T.: Local and Transderivational Constraints in Syntax and Semantics. Ph.D. thesis, UCLA (2013)

[9] Harkema, H.: A characterization of minimalist languages. In: de Groote, P., Morrill, G., Retoré, C. (eds.) Logical Aspects of Computational Linguistics (LACL'01), Lecture Notes in Artificial Intelligence, vol. 2099, pp. 193–211. Springer, Berlin (2001)

[10] Kobele, G.M.: On late adjunction in minimalist grammars (2010), slides for a talk given at MCFG+ 2010

[11] Kobele, G.M.: Without remnant movement, MGs are context-free. In: Ebert, C., Jäger, G., Michaelis, J. (eds.) MOL 10/11. Lecture Notes in Computer Science, vol. 6149, pp. 160–173 (2010)

[12] Kobele, G.M.: Idioms and extended transducers. In: Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11). pp. 153–161. Paris, France (September 2012), http://www.aclweb.org/anthology-new/W/W12/W12-4618

[13] Kobele, G.M.: Importing montagovian dynamics into minimalism. In: Béchet, D., Dikovsky, A. (eds.) LACL 2012. Lecture Notes in Computer Science, vol. 7351, pp. 103–118 (2012)
[14] Kobele, G.M., Michaelis, J.: Disentangling notions of specifier impenetrability. In: Kanazawa, M., Kornia, A., Kracht, M., Seki, H. (eds.) The Mathematics of Language. Lecture Notes in Artificial Intelligence, vol. 6878, pp. 126–142 (2011)
[15] Kobele, G.M., Retoré, C., Salvati, S.: An automata-theoretic approach to minimalism. In: Rogers, J., Kepser, S. (eds.) Model Theoretic Syntax at 10. pp. 71–80 (2007)
[16] Lebeaux, D.: Language Acquisition and the Form of the Grammar. Ph.D. thesis, University of Massachusetts, Amherst (1988)
[17] Michaelis, J.: Transforming linear context-free rewriting systems into minimalist grammars. Lecture Notes in Artificial Intelligence 2099, 228–244 (2001)
[18] Mönnich, U.: Grammar morphisms (2006), ms. University of Tübingen
[19] Mönnich, U.: Minimalist syntax, multiple regular tree grammars and direction preserving tree transductions. In: Rogers, J., Kepser, S. (eds.) Model Theoretic Syntax at 10. pp. 83–87 (2007)
[20] Mönnich, U.: A logical characterization of extended TAGs. In: Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11). pp. 37–45. Paris, France (September 2012), http://www.aclweb.org/anthology-new/W/W12/W12-4605
[21] Nissenbaum, J.: Covert movement and parasitic gaps. In: Proceedings of North Eastern Linguistic Society. vol. 30 (2000)
[22] Ochi, M.: Multiple spell-out and PF adjacency. In: Proceedings of the North Eastern Linguistic Society. vol. 29 (1999)
[23] Rogers, J.: A Descriptive Approach to Language-Theoretic Complexity. CSLI, Stanford (1998)
[24] Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 1328, pp. 68–95. Springer, Berlin (1997)
[25] Stabler, E.P.: Computational perspectives on minimalism. In: Boeckx, C. (ed.) Oxford Handbook of Linguistic Minimalism, pp. 617–643. Oxford University Press, Oxford (2011)
[26] Stabler, E.P.: Top-down recognizers for MCFGs and MGs. In: Proceedings of the 2011 Workshop on Cognitive Modeling and Computational Linguistics (2011), to appear
[27] Stabler, E.P.: Bayesian, minimalist, incremental syntactic analysis. Topics in Cognitive Science 5, 611–633 (2012)
[28] Stepanov, A.: Late adjunction and minimalist phrase structure. Syntax 4, 94–125 (2001)
[29] Takahashi, S., Hulsey, S.: Wholesale Late Merger: Beyond the A/$\overline{\text{A}}$ distinction. Linguistc Inquiry 40, 387–426 (2009)