Insertion Minimalist Grammars: Eliminating redundancies between merge and move

Tim Hunter*

Department of Linguistics
Yale University

Abstract. Minimalist Grammars (MGs) provide a setting for rigourous investigations of ideas that have been proposed at a more intuitive level in mainstream work in generative syntax. I address one such idea, namely the possibility that when an element appears to be "displaced", it might be usefully analysed not as having *merged* into one position and then *moved* to another position, but rather as simply having *merged* into one position, and then *merged again* into another. Intuitively, there appears to be some redundancy in a system where merge and move are unrelated primitive operations, because the structures that they build are of the same sort. I offer a careful illustration of how a MG variant based upon re-merging can eliminate these redundancies.

Stabler [16] presents Minimalist Grammars (MGs) as a precise formulation of the basic grammatical machinery introduced in [1]. MGs provide a setting for rigourous investigations of ideas that have been proposed at a more intuitive level in mainstream work in generative syntax. In this paper I address one such idea (suggested by, among others, [7, 2]), namely the possibility that when an element appears to be "displaced", it might be usefully analysed not as having *merged* into one position and then *moved* to another position, but rather as simply having *merged* into one position, and then *merged again* into another. Intuitively, there appears to be some redundancy in a system where merge and move are unrelated primitive operations, because the structures that they build are of the same sort, and the shift to a "move as re-merge" system offers the hope of eliminating this redundancy and therefore of a more parsimonious theory of grammar. In this paper I offer a careful illustration of how a MG variant based upon re-merging ("Insertion Minimalist Grammars", IMGs) can eliminate these redundancies. Since I also show that this variant is formally equivalent to the more standard version (both weakly and, in a meaningful sense, strongly), such qualitative issues as parsimony and perspicuity appear to be the most relevant criteria for deciding between the two formalisms.

I introduce standard MGs in section 1 and discuss the ways in which their definition appears to duplicate certain information in section 2. I then introduce the alternative IMG formalism, similar to the variant already presented in [17], in section 3, and discuss its consequences in section 4. The appendix demonstrates that IMGs are weakly equivalent to standard MGs.

* Thanks to Ed Stabler, Bob Frank, and anonymous reviewers for helpful advice.

1 (Standard) Minimalist Grammars

A Minimalist Grammar (MG) is a five-tuple $G = \langle \Sigma, Sel, Lic, Lex, \mathtt{c} \rangle$ where:

- Σ is a finite alphabet
- Sel ("selecting types") and Lic ("licensing types") are disjoint sets which together determine the set Syn ("syntactic features") as follows:

$$\begin{aligned} selectors &= \{\mathtt{=f} \mid \mathtt{f} \in Sel\} & licensors &= \{\mathtt{+f} \mid \mathtt{f} \in Lic\} \\ selectees &= \{\ \mathtt{f} \mid \mathtt{f} \in Sel\} & licensees &= \{\mathtt{-f} \mid \mathtt{f} \in Lic\} \\ Syn &= selectors \cup selectees \cup licensors \cup licencees \end{aligned}$$

- Lex ("the lexicon") is a finite subset of $\Sigma^* \times \{::\} \times (selectors \cup licensors)^* \times selectees \times licensees^*$
- $\mathtt{c} \in Sel$ is a designated type of completed expressions

Given an MG $G = \langle \Sigma, Sel, Lic, Lex, \mathtt{c} \rangle$, an expression is an element of

$$Expr = (\Sigma^* \times \{:, ::\} \times Syn^*) \times (\Sigma^* \times Syn^+)^*$$

and a lexical expression is an element of the set $Expr_{Lex} = Lex \times \{\epsilon\}$. For an expression of the form $\langle x, \langle y_1, y_2, \ldots, y_n \rangle \rangle$ I will often write $x, y_1 y_2 \ldots y_n$, and will sometimes call $y_1 y_2 \ldots y_n$ the "tail" of the expression; and for an expression of the form $\langle x, \epsilon \rangle$ (i.e. with an empty tail) I will often write just x.

$CL(G) \subseteq Expr$ is the closure of $Expr_{Lex}$ under the functions E-MRG and I-MRG, defined as follows. In these definitions: $s, t \in \Sigma^*$; $\mathtt{f} \in Sel \cup Lic$; $\alpha, \beta \in Syn^*$; $\gamma \in Syn^+$; $\cdot \in \{:, ::\}$; and $\phi, \psi \in (\Sigma^* \times Syn^+)^*$.

E-MRG = COMP-E-MRG ∪ SPEC-E-MRG ∪ NONFINAL-E-MRG

$$\frac{s :: \mathtt{=f}\alpha, \epsilon \qquad t \cdot \mathtt{f}, \psi}{st : \alpha, \psi}\ \text{COMP-E-MRG} \qquad \frac{s : \mathtt{=f}\alpha, \phi \qquad t \cdot \mathtt{f}, \psi}{ts : \alpha, \phi\psi}\ \text{SPEC-E-MRG}$$

$$\frac{s \cdot \mathtt{=f}\alpha, \phi \qquad t \cdot \mathtt{f}\gamma, \psi}{s : \alpha, \phi(t, \gamma)\psi}\ \text{NONFINAL-E-MRG}$$

I-MRG = SPEC-I-MRG ∪ NONFINAL-I-MRG

$$\frac{s : \mathtt{+f}\alpha, \phi(t, \mathtt{-f})\psi}{ts : \alpha, \phi\psi}\ \text{SPEC-I-MRG} \qquad \frac{s : \mathtt{+f}\alpha, \phi(t, \mathtt{-f}\gamma)\psi}{s : \alpha, \phi(t, \gamma)\psi}\ \text{NONFINAL-I-MRG}$$

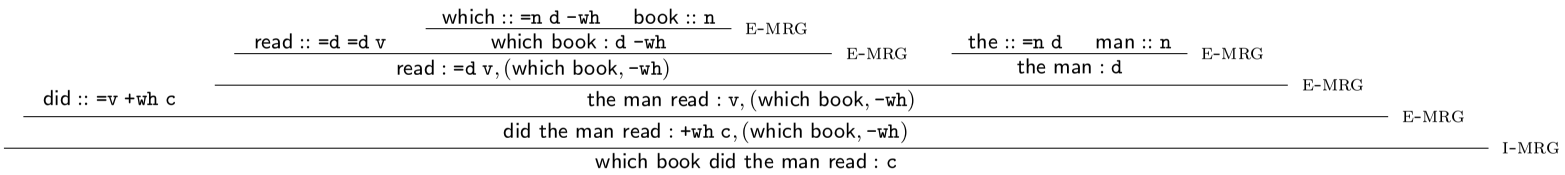
where we require, in all cases, that there is no other $(t', \mathtt{-f})$ or $(t', \mathtt{-f}\gamma')$ in ϕ or ψ

The language generated by G is $\mathcal{L}(G) = \{s \mid \langle s \cdot \mathtt{c}, \epsilon \rangle \in CL(G) \text{ for } \cdot \in \{:, ::\}\}$. The set of Minimalist Languages is defined as $ML = \{\mathcal{L}(G) \mid G \text{ is an MG}\}$.

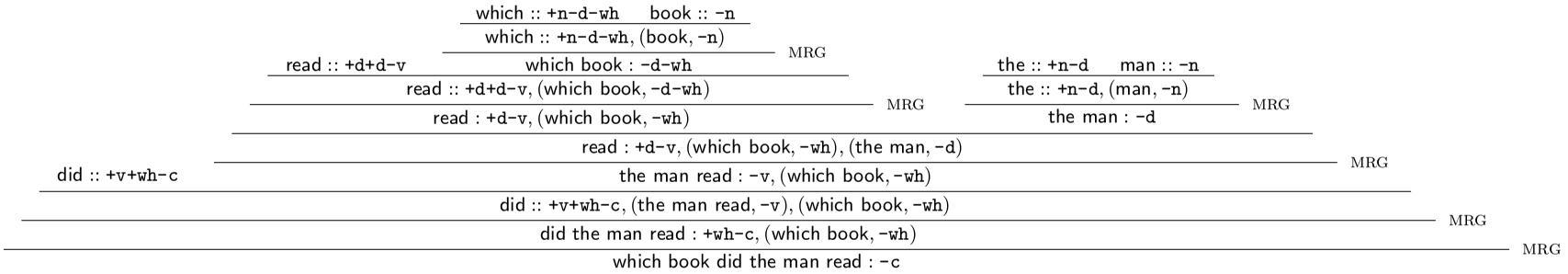
A derivation of a basic question which book did the man read can be derived from the lexical items in (1) as shown in Figure 1(a).

(1)

which :: =n d -wh	the :: =n d	read :: =d =d v
book :: n	man :: n	did :: =v +wh c



(a) A simple MG derivation, discussed in section 1



(b) A simple IMG derivation, discussed in section 3 (INSERT steps are unlabeled)

Fig. 1. Derivations of which book did the man read in each of the two formalisms considered

The E-MRG operation "checks" (or deletes) a selector feature (eg. `=n`) and a corresponding selectee feature (eg. `n`), and combines the two expressions bearing these features, in a manner roughly analogous to slash-elimation or function-application rules in categorial grammars. The distinctive notion of movement is implemented by the I-MRG operation which checks licensor (eg. `+wh`) and licensee (eg. `-wh`) features. The features on the lexical item which :: `=n d -wh` dictate that after being selected as a "`d`-type" element, an application of I-MRG will move (a phrase headed by) which will move to another position in order to check its `-wh` feature. Accordingly, the application of E-MRG that combines read with which book is, more specifically, an application of NONFINAL-E-MRG, and places (which book, `-wh`) in the "tail" of the resulting expression. Intuitively the tail of an expression records those parts of the expression that have been merged into an initial position but will move on to a different position at some point in the future of the derivation; this idea also bears some ressemblance to the "Cooper Storage" approach to quantification [4], where the "stored" things are not quantifier meanings but strings (see also [13]).

2 Redundancies in MGs

There are certain patterns which are shared by the definitions of both E-MRG and I-MRG. Each has the ability to create specifiers (phrases that are linearised to the left of the head that projects over them) and the ability to create empty positions. It is natural to ask what reasons there may be for duplicating in this way the stipulations of how these two things should be achieved.

2.1 Two ways to create specifiers

Consider the definitions of SPEC-E-MRG and SPEC-I-MRG. Each has the effect of concatenating a string t, contributed by a phrase that is not projecting, on the left of a string s, contributed by a phrase that is projecting.[1] That the two are stated separately prompts the question of whether this similarity is a mere accident and the two results "could have been" different: say, ts in one case and st in the other; or sst in one case and $ttss$ in the other; or ts in one case and $t^\frown\mathsf{foo}^\frown s$ in the other, where foo is a phonological reflex (not of just *being in* a specifier position, but rather) of specifically being *moved to* a specifier position. While it is obviously an empirical question whether any phenomena of this sort will be discovered, it is more typically assumed that the relevant notion is that of specifier, simpliciter; and that SPEC-E-MRG and SPEC-I-MRG construct the same string for a single reason, namely that *specifiers* are linearised uniformly.

[1] "Projecting" can be defined roughly as "contributing features to the non-tail part of the resulting expression". In the cases under discussion, s is contributed by a phrase that projects because the features α that were associated with s in the input expressions are carried over to the non-tail part of the expression $s : \alpha, \phi(t, \gamma)\psi$ that is generated.

More generally, it is standard to assume that the structural positions created by movement are not different from those created by (external) merge. This is the straightforward analog of the "structure-preserving constraint" [5] from pre-minimalist theories, according to which the target sites of movements must be positions that the base component "could have generated". Consider, for example, the positions of he and she in the two sentences in (2). They appear to be the *same* position (namely, subject position), as evidenced by the fact that they both participate in verbal agreement and both show nominative Case.

(2) a. He was arrested
b. She was happy

While he in (2a) has (on standard assumptions) been moved into this position by a passivisation transformation, she is base-generated (or externally merged) there in (2b).[2] The movement of he is therefore structure-preserving in the sense of [5], since the target position is the very same one that she is generated in *by the base* in (2b); in terms of the MG system as defined above, the position into which he would be placed by SPEC-I-MRG is the very same one that she is placed into by SPEC-E-MRG. But if all movements are structure-preserving in this way, as is generally assumed, then it is clear that SPEC-E-MRG and SPEC-I-MRG will necessarily produce output that is equivalent in significant respects, and so defining them as independent operations will entail unnecessary redundancy.

The redundancy described here, as a result of having two distinct ways to build the one kind of structural position, will be multiplied if it turns out to be necessary to permit *additional kinds* of structural positions. [8] present an extension of the MG formalism that has exactly this form. They supplement the system with a binary operation roughly analogous to E-MRG, called ADJOIN, which establishes adjunction configurations; these have certain properties (not relevant for present purposes) distinct from those of normal, non-adjunction configurations, to which standard MGs restrict attention. In keeping with structure preservation, however, there are also cases where elements are *moved into* adjunction configurations, just as was discussed in the case of subject positions above. Therefore [8] are forced to also add a unary operation which (i) differs from their new ADJOIN operation in precisely the way that I-MRG already differs from E-MRG, and (ii) differs from I-MRG in precisely the way that ADJOIN already differs from E-MRG, thus *restating* the distinctive properties of adjunction. The four operations they end up with, rather than each being distinct primitives, look quite clearly to be the result of two independent underlying distinctions: first, the distinction between attaching as an adjunct and attaching as a non-adjunct, and second, the distinction between attaching and re-attaching. The reformulated version of MGs I discuss below reflects this underlying pattern.

[2] For ease of exposition I abstract away from the fact that it is now more common to assume that subjects originate in a predicate-internal position even in active sentences. For a comparison parallel to that presented in (2) that is more consistent with modern assumptions, consider the thematic "wanter" position in John wants to win and John wants Mary to win in the context of the movement theory of control [9]: John will be moved there in the former but not in the latter.

2.2 Two ways to create vacated positions

Having observed similarities between the definitions of SPEC-E-MRG and SPEC-I-MRG, consider now those of NONFINAL-E-MRG and NONFINAL-I-MRG. Each results in an expression with s as the non-tail string and with t as a part of the tail, and each differs from its specifier-creating counterpart precisely in requiring that there be additional features associated with t besides the one being checked (recall that $\gamma \neq \epsilon$). Intuitively these two operations both create "vacated positions" or "trace positions", positions which t occupies only temporarily and where t therefore does not contribute to the eventual string yield. Again, we can ask whether the similarities in the two definitions are a mere accident; the alternative is that the similarities are there because there really is a uniform notion of a vacated position, in which case we would prefer a formalism where the properties of such positions (eg. contributing nothing to string yield) were stated only once.

Evidence suggests that it is oversimplifying to assume that *all* nonfinal positions are phonetically null, and one might therefore question the claim that NONFINAL-E-MRG and NONFINAL-I-MRG share an underlying mechanism, on the grounds that the similarities are artificially exaggerated in the definitions above. But while the question of how a sequence of positions linked by movement are phonetically realised remains very much open, in order to undermine the claim that a generalisation is being missed one would need evidence that the correct account will need to distinguish between *base* vacated positions (created by NONFINAL-E-MRG) and *intermediate* vacated positions (created by NONFINAL-I-MRG). This does not seem to be the case: wh-phrases, for example, can surface as such in "final" (3a), intermediate (3b) or base positions (3c); and resumptive pronouns can appear in intermediate (4a) or base positions (4b).

(3) a. **Who**$_i$ did John meet e_i?
b. Was$_i$ denkst du **wen**$_i$ Fritz e_i eingeladen hat?
c. Who e_i bought **what**$_i$?

(4) a. the book which$_i$ I wonder whether **it**$_i$ was read t_i
b. the book which$_i$ I wonder whether John read **it**$_i$

This range of data suggests that while the assumption that all nonfinal positions are phonetically null will have to be modified in some way, there is little reason to believe that a distinction between base vacated positions and intermediate vacated positions will play an important role in the eventual explanation. Put differently, there is little reason to believe that these two kinds of positions should not be treated with the same derivational mechanisms.[3] The definitions of NONFINAL-E-MRG and NONFINAL-I-MRG given above therefore constitute another instance of unnecessary redundancy.

[3] The data does suggest that the *kind of feature checked* (eg. wh feature or Case feature) in a particular position may be relevant to determining the phonetic realisation of that position, but this is different from saying that whether the feature checking occurred via E-MRG or I-MRG is relevant.

3 Insertion Minimalist Grammars

The central idea, following [17], is to define a single MRG operation which performs all feature-checking (intuitively, all structure-building). Since all feature-checking is performed by a single operation, we also eliminate the distinction between selector/selectee (`=f`/`f`) feature pairs and licensor/licensee (`+f`/`-f`) feature pairs.[4] I present here a bare bones implementation of these ideas, minimally different from standard MGs, leaving aside other issues addressed in [17] such as covert movement, copy movement, sidewards movement and persistent features.

An Insertion Minimalist Grammar is a four-tuple $G = \langle \Sigma, Lic, Lex, \mathtt{c} \rangle$ where:

- Σ is a finite alphabet
- Lic ("licensing types") is a set which determines the set Syn ("syntactic features") as follows:

$$\begin{aligned} licensors &= \{\mathtt{+f} \mid \mathtt{f} \in Lic\} \\ licensees &= \{\mathtt{-f} \mid \mathtt{f} \in Lic\} \\ Syn &= licensors \cup licencees \end{aligned}$$

- Lex ("the lexicon") is a finite subset of $\Sigma^* \times \{::\} \times (licensors^* \times licensees^+)$
- $\mathtt{c} \in Lic$ is a designated type of completed expressions

Given an IMG $G = \langle \Sigma, Lic, Lex, \mathtt{c} \rangle$, an expression is an element of the set

$$Expr = (\Sigma^* \times \{:, ::\} \times Syn^*) \times (\Sigma^* \times Syn^+)^*$$

and a lexical expression is an element of the set $Expr_{Lex} = Lex \times \{\epsilon\}$. For an expression of the form $\langle x, \langle y_1, y_2, \ldots, y_n \rangle \rangle$ I will often write $x, y_1 y_2 \ldots y_n$, and will sometimes call $y_1 y_2 \ldots y_n$ the "tail" of the expression; and for an expression of the form $\langle x, \epsilon \rangle$ (i.e. with an empty tail) I will often write just x.

$CL(G) \subseteq Expr$ is the closure of $Expr_{Lex}$ under the functions INSERT and MRG, defined as follows. In these definitions: $s, t \in \Sigma^*$; $\mathtt{f} \in Lic$; $\alpha, \beta \in Syn^*$; $\gamma \in Syn^+$; $\cdot \in \{:, ::\}$; and $\phi, \psi \in (\Sigma^* \times Syn^+)^*$.

$$\frac{s \cdot_1 \mathtt{+f}\alpha, \phi \qquad t \cdot_2 \mathtt{-f}\beta, \psi}{s \cdot_1 \mathtt{+f}\alpha, \phi(t, \mathtt{-f}\beta)\psi} \text{ INSERT}$$

MRG = COMP-MERGE ∪ SPEC-MERGE ∪ NONFINAL-MERGE

$$\frac{s :: \mathtt{+f}\alpha, \phi(t, \mathtt{-f})\psi}{st : \alpha, \phi\psi} \text{ COMP-MERGE} \qquad \frac{s : \mathtt{+f}\alpha, \phi(t, \mathtt{-f})\psi}{ts : \alpha, \phi\psi} \text{ SPEC-MERGE}$$

$$\frac{s \cdot \mathtt{+f}\alpha, \phi(t, \mathtt{-f}\gamma)\psi}{s : \alpha, \phi(t, \gamma)\psi} \text{ NONFINAL-MERGE}$$

where we require, in all cases, that there is no other $(t', \mathtt{-f})$ or $(t', \mathtt{-f}\gamma')$ in ϕ or ψ

[4] Emonds [6, p.53, n.14] notes that the assumption that merge and move check the same features is closely related to his earlier notion of structure preservation.

The language generated by G is $\mathcal{L}(G) = \{s \mid \langle s \cdot \texttt{-c}, \epsilon\rangle \in CL(G) \text{ for } \cdot \in \{:, ::\}\}$. The set of Insertion Minimalist Languages is defined as $IML = \{\mathcal{L}(G) \mid G \text{ is an IMG}\}$.

The IMG derivation of which book did the man read, analogous to the MG derivation presented earlier in Figure 1(a), is given in Figure 1(b). The relevant lexical items are given in (5); compare with (1). I have left INSERT steps unlabelled in Figure 1(b). This saves space but also facilitates comparison with the earlier MG derivation.

(5)

which :: +n-d-wh	book :: -n	the :: +n-d
man :: -n	read :: +d+d-v	did :: +v+wh-c

The basic difference between the IMG derivation in Figure 1(b) and the corresponding MG derivation in Figure 1(a) is that what was previously achieved by E-MRG is now achieved by applying INSERT and then applying (generic) MRG to the result. For example, instead of forming the man from lexical items via a single application of E-MRG, first INSERT produces (without checking any features) an expression which contains (man, -n) as part of its tail, and then MRG performs the feature-checking and concatenation. Crucially, this MRG step "does not know" whether (man, -n) came to be in the tail of the expression via an immediately preceding application of INSERT (as in fact it has in this example), or whether it is there as a "moving" element that has been "left there" after checking features in a non-final position. Thus the operation that "re-merges" which book into its final position at the end of the derivation is no different from, for example, that which "first merges" the man into its one and only position.

4 Comparison of IMGs and MGs

4.1 Elimination of redundancies

Observations in section 2 led us to the intuition that in the original MG formalism, SPEC-E-MRG and SPEC-I-MRG "did the same thing", in some sense — roughly, concatenate a string t, associated with only one unchecked feature, on the left of a string s — the only difference being that while SPEC-E-MRG draws t from (the non-tail part of) a separate expression, SPEC-I-MRG draws t from the tail of the same expression that already contains s. Similarly, NONFINAL-E-MRG and NONFINAL-I-MRG differed only in where they look for the element t which, in the "nonfinal" case, remains detached but has a feature checked.

The IMG formalism generalises to the case of drawing t from the tail of the expression that already contains s, i.e. essentially the case dealt with by I-MRG in MGs. The task of transforming a situation requiring E-MRG into one requiring (I-)MRG is taken up by INSERT, which puts a sought-after but hitherto uninvolved element into the tail of an expression, "within view" of the seeking element. To the extent that we can maintain the usual intuition of what a derived tree is, INSERT puts things into the derivation (or into a "workspace") without putting them anywhere in the derived tree, making them available for MRG to put them into the derived tree — as many times as necessary. Thus

what seemed to be the one thing done by both SPEC-E-MRG and SPEC-I-MRG, namely creation of specifiers, is naturally seen as such in IMG the definition of SPEC-MERGE. The connection between the condition of t being associated with only one remaining `-f` feature and the concatenation with (non-lexical) s to form ts is stated only there. Similarly, what seemed to be the one thing done by both NONFINAL-E-MRG and NONFINAL-I-MRG is now stated just once in the IMG definition of NONFINAL-MRG.

Note that on this view, "external merge" is the more complex operation which has "internal merge" as a part. More precisely, the derivative notion "external merge" is a name for a certain combination of INSERT and MRG operations, whereas internal merge is implemented by an application of MRG alone. When the possibility of unifying the two operations is discussed in the linguistics literature, however, this relationship is reversed: the idea is usually that internal merge might be analysed as copy plus external merge ([3, 9], among many others).

4.2 Other consequences of adopting IMGs

There have been suggestions that the way adjuncts differ from non-adjuncts might be usefully analysed as being not just *differently* attached, but in some sense *less* attached, or less tightly attached. More specifically, it has been proposed that adjunction structures differ in being unlabelled [10], or that "An adjunct is simply activated on a derivational phase, without connecting to the phrase-marker . . . adjuncts can be activated by simply 'being there' " [14, pp.254–255]. The intuition is appealing but needs sharpening to be pursued rigourously. Decoupling the work of introducing new elements into the derivation from the work of building the phrase-marker, as IMGs do (delegating these two responsibilities to INSERT and MRG respectively), makes room for one approach to formalising this intuition: [11] presents a variant of IMGs where adjuncts are *only* inserted and never merged.[5] There is no correlate of this idea in standard MGs because in that system an element is introduced into the derivation precisely when it first checks features; there is no "middle ground" status corresponding to being present in the derivation without being part of the phrase-marker.

Finally, there is a potential disadvantage that comes with the shift from MGs to IMGs. Collapsing selectee features with licensee features has the effect of eliminating the usual notion of "syntactic category". A lexical item such as `who`, for example, will have two features of the same kind, `-d` and `-wh`. This means that we can no longer say (as we could in the MG system, where `who` would bear the features `d` and `-wh` of different kinds) that `d` is in any sense *the*

[5] This naturally requires further decoupling feature-checking (corresponding to building the phrase-marker) from string concatenation, both of which are achieved by MRG in the IMG system presented here. Without this additional step it would be impossible for an element that is only inserted to contribute to the eventual yield. The basic analysis of adjuncts in [11] seems unlikely to affect the generative capacity of the IMG formalism, although the proof in the appendix of this paper does not address this.

category of this element. One response to this apparent problem would be to modify the MRG function such that the first licensee (`-f`) feature checked by a particular element is recorded and remains with the element, in a way that characterises it for the remainder of the derivation; this would mimic the MGs' stipulation that each lexical feature sequence contain exactly one selectee feature in a specific position. Such a stipulation on the structure of the lexicon would, of course, leave IMGs' distinctive (and, I have argued, beneficial) derivational machinery unchanged.

5 Conclusion

I have argued that there are significant unnecessary redundancies in the standard MG formalism, and presented a reformulated version, IMGs, which eliminate these. The central idea is to reduce merge and move to a single operation, in accord with the intuition that the tree structure built by merge steps does not differ in kind from that built by move steps. This reformulated version produces the same string languages as the original formalism, and derivation structures that can be put in a transparent one-to-one correspondence with those of the original. Comparisons of the virtues of the two formalisms can therefore focus on the extent to which IMGs bring gains in theoretical parsimony and perspicuity.

References

1. Chomsky, N.: The Minimalist Program. MIT Press, Cambridge, MA (1995)
2. Chomsky, N.: Beyond explanatory adequacy. In: Belletti, A. (ed.) Structures and Beyond. Oxford University Press, Oxford (2004)
3. Collins, C.: Local Economy. MIT Press, Cambridge, MA (1997)
4. Cooper, R.: Quantification and Syntactic Theory. Reidel, Dordrecht (1983)
5. Emonds, J.: Root and Structure-Preserving Transformations. Ph.D. thesis, MIT (1970)
6. Emonds, J.: Adjectival passives. In: Everaert, M., van Riemsdijk, H. (eds.) The Blackwell Companion to Syntax, pp. 16–60. Wiley-Blackwell, Malden, MA (2005)
7. Epstein, S.D., Groat, E., Kawashima, R., Kitahara, H.: A Derivational Approach to Syntactic Relations. Oxford University Press, Oxford (1998)
8. Frey, W., Gärtner, H.M.: On the treatment of scrambling and adjunction in minimalist grammars. In: Jäger, G., Monachesi, P., Penn, G., Wintner, S. (eds.) Proceedings of Formal Grammar 2002. pp. 41–52 (2002)
9. Hornstein, N.: Move! A minimalist theory of construal. Blackwell, Oxford (2001)
10. Hornstein, N., Nunes, J.: Adjunction, labeling, and bare phrase structure. Biolinguistics 2(1), 57–86 (2008)
11. Hunter, T.: Relating Movement and Adjunction in Syntax and Semantics. Ph.D. thesis, University of Maryland (2010)
12. Kallmeyer, L.: Parsing Beyond Context-Free Grammars. Springer-Verlag, Berlin Heidelberg (2010)
13. Kobele, G.M.: Inverse linking via function composition. Natural Language Semantics 18(2), 183–196 (2010)

14. Lasnik, H., Uriagereka, J.: A Course in Minimalist Syntax. Blackwell, Malden, MA (2005)
15. Michaelis, J.: Derivational minimalism is mildly context-sensitive. In: Moortgat, M. (ed.) Logical Aspects of Computational Linguistics, LNCS, vol. 2014, pp. 179–198. Springer, Berlin Heidelberg (2001)
16. Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) Logical Aspects of Computational Linguistics. LNCS, vol. 1328, pp. 68–95. Springer, Berlin Heidelberg (1997)
17. Stabler, E.P.: Sidewards without copying. In: Wintner, S. (ed.) Proceedings of FG-2006: The 11th Conference on Formal Grammar. pp. 157–170. CSLI Publications, Stanford, CA (2006)

A Appendix: The generative capacity of MGs and IMGs

This appendix provides a proof that MGs and IMGs are weakly equivalent, i.e. that $ML = IML$. I first show that for every IMG there is a weakly equivalent multiple context-free grammar (MCFG), and hence $IML \subseteq MCFL$, in subsection A.1. I then show that for every MG there is a weakly equivalent IMG, and hence $ML \subseteq IML$, in subsection A.2; the construction clearly establishes a one-to-one correspondence between the two systems' derivation trees, in accord with the intuitions discussed in section 4. In combination with the fact that $ML = MCFL$ [15], these results entail that $IML = ML = MCFL$.

A.1 Showing that $IML \subseteq MCFL$

Given an IMG $G = \langle \Sigma, Lic, Lex, \mathtt{c} \rangle$, we will construct an MCFG $G' = \langle N, T, F, P, S \rangle$ (taking definition from [12, pp.110-111]) such that $\mathcal{L}(G) = \mathcal{L}(G')$. The set of terminals of the MCFG, T, is Σ from the IMG. The given IMG determines a set of categories $Cat = \{::, :\} \times (Syn^*)^*$. For an IMG expression $e = s \cdot \alpha_0, (t_1, \alpha_1)(t_2, \alpha_2) \ldots (t_n, \alpha_n)$, the category of e is $\mathrm{Cat}(e) = \langle \cdot, \alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$.

The set of nonterminals of the MCFG $N = N' \cup \{S\}$, where N' is a certain finite subset of Cat. Specifically, a category $C = \langle \cdot, \alpha_0, \alpha_1, \ldots, \alpha_n \rangle \in N'$ iff the following are true: (i) no two of the α_i share the same first feature, and (ii) for every i such that $0 \leq i \leq n$, $|\alpha_i| \leq k$, where k is the length of the longest feature-sequence in Lex.

The start symbol of the MCFG is S. The set F contains the function $f_{\mathrm{id}} : \Sigma^* \to \Sigma^*$, $f_{\mathrm{id}}(s) = s$, and the set P contains the two productions $S \to f_{\mathrm{id}}[\langle :, \mathtt{-c} \rangle]$ and $S \to f_{\mathrm{id}}[\langle ::, \mathtt{-c} \rangle]$, in accordance with the fact that a complete expression may be either derived or lexical.

The MRG operation is encoded in the MCFG as follows. For every category $C = \langle \cdot, \mathtt{+f}\alpha_0, \alpha_1, \ldots, \alpha_n \rangle \in N$:

- if there is a unique α_i, $1 \leq i \leq n$, such that $\alpha_i = \mathtt{-f}\alpha_i'$ and $\alpha_i' = \epsilon$, then:
  - F contains a function $f_C : (\Sigma^*)^{n+1} \to (\Sigma^*)^n$, such that

$$f_C(\langle s_0, s_1, \ldots, s_n \rangle) = \begin{cases} \langle s_0 s_i, s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n \rangle & \text{if } \cdot = :: \\ \langle s_i s_0, s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n \rangle & \text{if } \cdot = : \end{cases}$$

  - P contains a production $C' \to f_C[C]$, where $C' = \langle :, \alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n \rangle$

– if there is a unique α_i, $1 \leq i \leq n$, such that $\alpha_i = \texttt{-f}\alpha_i'$ and $\alpha_i' \neq \epsilon$, then:
  - F contains a function $f_C : (\Sigma^*)^{n+1} \to (\Sigma^*)^{n+1}$, such that

$$f_C(\langle s_0, s_1, \ldots, s_n \rangle) = \langle s_0, s_1, \ldots, s_n \rangle$$

  - P contains a production $C' \to f_C[C]$, where $C' = \langle :, \alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_i', \alpha_{i+1}, \ldots, \alpha_n \rangle$

The INSERT operation is encoded in the MCFG as follows. For every two categories $C_1 = \langle \cdot_1, \texttt{+f}\alpha_0, \alpha_1, \ldots, \alpha_n \rangle \in N$ and $C_2 = \langle \cdot_2, \texttt{-f}\beta_0, \beta_1, \ldots, \beta_m \rangle \in N$:

– F contains a function $f_{C_1,C_2} : (\Sigma^*)^{n+1} \times (\Sigma^*)^{m+1} \to (\Sigma^*)^{n+m+2}$, such that

$$f_{C_1,C_2}(\langle s_0, s_1, \ldots, s_n \rangle, \langle t_0, t_1, \ldots, t_m \rangle) = \langle s_0, s_1, \ldots, s_n, t_0, t_1, \ldots, t_m \rangle$$

– P contains a production $C' \to f_{C_1,C_2}[C_1, C_2]$, where $C' = \langle \cdot_1, \texttt{+f}\alpha_0, \alpha_1, \ldots, \alpha_n, \texttt{-f}\beta_0, \beta_1, \ldots, \beta_m \rangle$

The IMG lexicon is encoded in the MCFG as follows. For every lexical item $s :: \alpha \in Lex$:

– F contains a function $f_s : (\Sigma^*)^0 \to (\Sigma^*)^1$ such that $f_s(\langle\rangle) = \langle s \rangle$
– P contains a production $C \to f_s[]$ where $C = \langle ::, \alpha \rangle$

Lemma 1. *If the MCFG G' can derive that the tuple of strings $\langle s_0, \ldots, s_n \rangle$ has category $\langle \cdot, \alpha_0, \ldots, \alpha_n \rangle$, then the IMG G can derive the expression $s_0 \cdot \alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)$.*

Proof. – Base case. Any leaf of an MCFG derivation tree corresponds to a nullary rule of the MCFG. The only nullary rules we have included in G' are of the form $C \to f_s[]$ where $f_s(\langle\rangle) = \langle s \rangle$ and $C = \langle ::, \alpha \rangle$ for some $s :: \alpha$ in the lexicon of G. Therefore if there is a length-one G' derivation which derives $\langle s_0, \ldots, s_n \rangle$ with category C, then $n = 0$ and C is $\langle ::, \alpha \rangle$ for some $s_0 :: \alpha \in Lex$, and there is a length-one G derivation of the expression $s_0 :: \alpha$.
– Induction step, unary case. Any unary-branching step in a G' derivation tree is licensed by some production $C' \to f_C[C]$ and a subtree deriving that $\langle s_0, \ldots, s_n \rangle$ has category $C = \langle \cdot, \alpha, \alpha_1, \ldots, \alpha_n \rangle$. By the induction assumption, there is a G derivation tree deriving the expression $\langle s_0 \cdot \alpha, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle$. Since the production $C' \to f_C[C]$ is present, we are in one of the following two cases:
  - $\alpha = \texttt{+f}\alpha_0$ and there is a unique α_i such that $\alpha_i = \texttt{-f}\alpha_i'$, and $\alpha_i' = \epsilon$. Let $s' = s_0 s_i$ if $\cdot = ::$ and $s' = s_i s_0$ otherwise. Then

$$f_C(\langle s_0, s_1, \ldots, s_n \rangle) = \langle s', s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n \rangle$$

and $C' = \langle \cdot, \alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n \rangle$. But also

$$\textsc{mrg}(\langle s_0 \cdot \alpha, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle) = \langle s' : \alpha_0, (s_1, \alpha_1), \ldots, (s_{i-1}, \alpha_{i-1}), (s_{i+1}, \alpha_{i+1}), \ldots, (s_n, \alpha_n) \rangle$$

and therefore there is a G derivation of

$$\langle s' : \alpha_0, (s_1, \alpha_1), \ldots, (s_{i-1}, \alpha_{i-1}), (s_{i+1}, \alpha_{i+1}), \ldots, (s_n, \alpha_n) \rangle$$

as required.

- $\alpha = \texttt{+f}\alpha_0$ and there is a unique α_i such that $\alpha_i = \texttt{-f}\alpha_i'$, and $\alpha_i' \neq \epsilon$. Then

$$f_C(\langle s_0, s_1, \ldots, s_n \rangle) = \langle s_0, s_1, \ldots, s_n \rangle$$

and $C' = \langle :, \alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_i', \alpha_{i+1}, \ldots, \alpha_n \rangle$. But also

$$\textsc{mrg}(\langle s_0 \cdot \alpha, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle) = \langle s_0 : \alpha_0, (s_1, \alpha_1), \ldots, (s_i, \alpha_i'), \ldots, (s_n, \alpha_n) \rangle$$

and therefore there is a G derivation of

$$\langle s_0 : \alpha_0, (s_1, \alpha_1), \ldots, (s_i, \alpha_i'), \ldots, (s_n, \alpha_n) \rangle$$

as required.

– Induction step, binary case. Any binary-branching step in a G' derivation tree is licensed by some production $C' \rightarrow f_{C_1,C_2}[C_1, C_2]$ and two subtrees, one deriving that $\langle s_0, \ldots, s_n \rangle$ has category $C_1 = \langle \cdot_1, \alpha_0, \ldots, \alpha_n \rangle$ and one deriving that $\langle t_0, \ldots, t_m \rangle$ has category $C_2 = \langle \cdot_2, \beta_0, \ldots, \beta_m \rangle$. By the induction assumption, there are G derivation trees deriving the expressions $\langle s_0 \cdot_1 \alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle$ and $\langle t_0 \cdot_2 \beta_0, (t_1, \beta_1), \ldots, (t_m, \beta_m) \rangle$. Since the production $C' \rightarrow f_{C_1,C_2}[C_1, C_2]$ is present, there must be some $\texttt{f}$ such that $\alpha_0 = \texttt{+f}\alpha_0'$ and $\beta_0 = \texttt{-f}\beta_0'$. Now

$$f_{C_1,C_2}(\langle s_0, s_1, \ldots, s_n \rangle, \langle t_0, t_1, \ldots, t_m \rangle) = \langle s_0, s_1, \ldots, s_n, t_0, t_1, \ldots, t_m \rangle$$

and $C' = \langle \cdot_1, \alpha_0, \alpha_1, \ldots, \alpha_n, \beta_0, \beta_1, \ldots, \beta_m \rangle$. But also

$$\textsc{insert}(\langle s_0 \cdot_1 \alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle, \langle t_0 \cdot_2 \beta_0, (t_1, \beta_1), \ldots, (t_m, \beta_m) \rangle) = \langle s_0 \cdot_1 \alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n), (t_0, \beta_0), (t_1, \beta_1), \ldots, (t_m, \beta_m) \rangle$$

and therefore there is a G derivation of

$$\langle s_0 \cdot_1 \alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n), (t_0, \beta_0), (t_1, \beta_1), \ldots, (t_m, \beta_m) \rangle$$

as required.

Therefore, if G' can derive that a one-tuple $\langle s \rangle$ has category $\langle :, \texttt{-c} \rangle$ then G can derive the expression $s : \texttt{-c}$, and if G' can derive that a one-tuple $\langle s \rangle$ has category $\langle ::, \texttt{-c} \rangle$ then G can derive the expression $s :: \texttt{-c}$. Also $s \in \mathcal{L}(G')$ iff G' can derive that $\langle s \rangle$ either has category $\langle :, \texttt{-c} \rangle$ or has category $\langle ::, \texttt{-c} \rangle$. Thus if $s \in \mathcal{L}(G')$ then $s \in \mathcal{L}(G)$, and $\mathcal{L}(G') \subseteq \mathcal{L}(G)$.

Lemma 2. *If the IMG G can derive the expression $\langle s_0 \cdot \alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)\rangle$, then the MCFG G' can derive that the tuple of strings $\langle s_0, \ldots, s_n\rangle$ has category $\langle \cdot, \alpha_0, \ldots, \alpha_n\rangle$.*

Proof. – Base case. A length-one G derivation is simply a lexical item $s :: \alpha$. For each such lexical item, G' has a production $C \to f_s[]$ where $C = \langle ::, \alpha\rangle$ and $f_s(\langle\rangle) = s$. Therefore there is a length-one G' derivation of $\langle s\rangle$ with category $\langle ::, \alpha\rangle$, as required.

– Induction step, unary case. A unary-branching step in a G derivation tree is licensed by applying MRG to some expression $\langle s_0 \cdot \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)\rangle$. By the induction assumption, there is a G' derivation of $\langle s_0, s_1, \ldots, s_n\rangle$ with category $C = \langle \cdot, \texttt{+f}\alpha_0, \alpha_1, \ldots, \alpha_n\rangle$. Since MRG is applicable, we are in one of the following two cases:

- There is a unique $\alpha_i, 1 \leq i \leq n$, such that $\alpha_i = \texttt{-f}\alpha_i'$ and $\alpha_i' = \epsilon$. Let $s' = s_0 s_i$ if $\cdot = ::$ and $s' = s_i s_0$ otherwise. Then

$$\begin{aligned}\textsc{mrg}(\langle s_0 \cdot \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)\rangle) = \\ \langle s' : \alpha_0, (s_1, \alpha_1), \ldots, (s_{i-1}, \alpha_{i-1}), (s_{i+1}, \alpha_{i+1}), \ldots (s_n, \alpha_n)\rangle\end{aligned}$$

  But also G' contains a production $C' \to f_C[C]$, where

$$\begin{aligned}f_C(\langle s_0, s_1, \ldots, s_n\rangle) = \langle s', s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n\rangle \\ C' = \langle :, \alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n\rangle\end{aligned}$$

  and therefore there is a G' derivation of $\langle s', s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n\rangle$ with category $\langle :, \alpha_0, \alpha_1, \ldots, \alpha_{i-1}, \alpha_{i+1}, \ldots, \alpha_n\rangle$, as required.
- There is a unique $\alpha_i, 1 \leq i \leq n$, such that $\alpha_i = \texttt{-f}\alpha_i'$ and $\alpha_i' \neq \epsilon$. Then

$$\begin{aligned}\textsc{mrg}(\langle s_0 \cdot \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)\rangle) = \\ \langle s_0 : \alpha_0, (s_1, \alpha_1), \ldots, (s_i, \alpha_i'), \ldots, (s_n, \alpha_n)\rangle\end{aligned}$$

  But also G' contains a production $C' \to f_C[C]$, where

$$\begin{aligned}f_C(\langle s_0, s_1, \ldots, s_n\rangle) = \langle s_0, s_1, \ldots, s_n\rangle \\ C' = \langle :, \alpha_0, \alpha_1, \ldots, \alpha_i', \ldots, \alpha_n\rangle\end{aligned}$$

  and therefore there is a G' derivation of $\langle s_0, s_1, \ldots, s_n\rangle$ with category $\langle :, \alpha_0, \alpha_1, \ldots, \alpha_i', \ldots, \alpha_n\rangle$, as required.

– Induction step, binary case. A binary-branching step in a G derivation tree is licensed by applying INSERT to two expressions $\langle s_0 \cdot_1 \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)\rangle$ and $\langle t_0 \cdot_2 \texttt{-f}\beta_0, (t_1, \beta_1), \ldots, (t_m, \beta_m)\rangle$. By the induction assumption, there are G' derivation trees deriving that $\langle s_0, s_1, \ldots, s_n\rangle$ has category $C_1 = \langle \cdot_1, \texttt{+f}\alpha_0, \ldots, \alpha_n\rangle$ and that $\langle t_0, t_1, \ldots, t_m\rangle$ has category $C_2 = \langle \cdot_2, \texttt{-f}\beta_0, \ldots, \beta_m\rangle$. Now

$$\begin{aligned}\textsc{insert}(\langle s_0 \cdot_1 \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)\rangle, \langle t_0 \cdot_2 \texttt{-f}\beta_0, (t_1, \beta_1), \ldots, (t_m, \beta_m)\rangle) \\ = \langle s_0 \cdot_1 \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n), (t_0, \texttt{-f}\beta_0), \ldots, (t_m, \beta_m)\rangle\end{aligned}$$

But also G' contains a production $C' \to f_{C_1,C_2}[C_1, C_2]$, where

$$f_{C_1,C_2}(\langle s_0, s_1, \ldots, s_n\rangle, \langle t_0, t_1, \ldots, t_m\rangle) = \langle s_0, s_1, \ldots, s_n, t_0, t_1, \ldots, t_m\rangle$$
$$C' = \langle \cdot_1, \texttt{+f}\alpha_0, \alpha_1, \ldots, \alpha_n, \texttt{-f}\beta_0, \beta_1, \ldots, \beta_m\rangle$$

and therefore there is a G' derivation of $\langle s_0, s_1, \ldots, s_n, t_0, t_1, \ldots, t_m\rangle$ with category $\langle \cdot_1, \texttt{+f}\alpha_0, \alpha_1, \ldots, \alpha_n, \texttt{-f}\beta_0, \beta_1, \ldots, \beta_m\rangle$, as required.

Therefore, if G can derive an expression $s : \texttt{-c}$ then G' can derive that the one-tuple $\langle s\rangle$ has category $\langle :, \texttt{-c}\rangle$, and if G can derive an expression $s :: \texttt{-c}$ then G' can derive that the one-tuple $\langle s\rangle$ has category $\langle ::, \texttt{-c}\rangle$. Also $s \in \mathcal{L}(G')$ iff G' can derive that $\langle s\rangle$ either has category $\langle :, \texttt{-c}\rangle$ or has category $\langle ::, \texttt{-c}\rangle$. Thus if $s \in \mathcal{L}(G)$ then $s \in \mathcal{L}(G')$, and $\mathcal{L}(G) \subseteq \mathcal{L}(G')$.

A.2 Showing that *ML* $\subseteq$ *IML*

Given an MG $G = \langle \Sigma, Sel, Lic, Lex, \texttt{c}\rangle$, we will construct an IMG $G' = \langle \Sigma, Lic', Lex', \hat{\texttt{c}}\rangle$ such that $\mathcal{L}(G) = \mathcal{L}(G')$. The set of licensing types of the IMG is $Lic' = Lic \cup \{\hat{\texttt{f}} \mid \texttt{f} \in Sel\}$ (assuming that each $\hat{\texttt{f}}$ is a new symbol not in Lic). This determines the set of syntactic features as usual, $Syn' = \{\texttt{-f} \mid \texttt{f} \in Lic'\} \cup \{\texttt{+f} \mid \texttt{f} \in Lic'\}$. The lexicon of the IMG is $Lex' = \{s{::}\boldsymbol{T}(\alpha) \mid s{::}\alpha \in Lex\}$, where $\boldsymbol{T} : Syn \to Syn'$ is defined as follows (and is lifted to apply to sequences as necessary):

$$\boldsymbol{T}(\texttt{f}) = \texttt{-}\hat{\texttt{f}} \quad \boldsymbol{T}(\texttt{-f}) = \texttt{-f} \quad \boldsymbol{T}(\texttt{=f}) = \texttt{+}\hat{\texttt{f}} \quad \boldsymbol{T}(\texttt{+f}) = \texttt{+f}$$

For any MG expression E, $\boldsymbol{T}(E)$ is the IMG expression that results from replacing every feature f in E with $\boldsymbol{T}(f)$; likewise for other structures, eg. $\boldsymbol{T}(\phi)$ where $\phi \in (\Sigma^* \times Syn^+)^*$. Since we assume that $\{\hat{\texttt{f}} \mid \texttt{f} \in Sel\} \cap Lic = \varnothing$, the inverse of $\boldsymbol{T}$ is also defined. Let $\boldsymbol{U}$ be this function, eg. $\boldsymbol{U}(\texttt{-}\hat{\texttt{f}}) = \texttt{f}$, $\boldsymbol{U}(\texttt{+}\hat{\texttt{f}}) = \texttt{=f}$.

Lemma 3. *If there is a G' derivation of e' that does not end with an application of* INSERT*, then there is a G derivation of $e = \boldsymbol{U}(e')$.*

Proof. – Base case. A length-one G' derivation consisting of nothing but a lexical item does not end with an application of INSERT. Since any lexical expression of G' is $s :: \boldsymbol{T}(\alpha)$ for some $s :: \alpha \in Lex$, there is a G derivation of $\boldsymbol{U}(s :: \boldsymbol{T}(\alpha)) = s :: \alpha$.

– Induction step, unary case. A unary-branching step in a G' derivation of e' is licensed by applying MRG to some expression $e'_p = \langle s_0{\cdot}\texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n)\rangle$. Since MRG is applicable to e'_p, there is a unique α_i, $1 \le i \le n$, such that $\alpha_i = \texttt{-f}\alpha'_i$, and

$$e' = \text{MRG}(e'_p) =$$
$$\begin{cases} \langle s_0 s_i : \alpha_0, (s_1, \alpha_1), \ldots, (s_{i-1}, \alpha_{i-1}), (s_{i+1}, \alpha_{i+1}), \ldots, (s_n, \alpha_n)\rangle & \text{if } \alpha'_i = \epsilon \text{ and } \cdot = :: \\ \langle s_i s_0 : \alpha_0, (s_1, \alpha_1), \ldots, (s_{i-1}, \alpha_{i-1}), (s_{i+1}, \alpha_{i+1}), \ldots, (s_n, \alpha_n)\rangle & \text{if } \alpha'_i = \epsilon \text{ and } \cdot = : \\ \langle s_0 : \alpha_0, (s_1, \alpha_1), \ldots, (s_i, \alpha'_i), \ldots, (s_n, \alpha_n)\rangle & \text{if } \alpha'_i \neq \epsilon \end{cases}$$

We are in one of the following cases:

- The G' derivation of e'_p ends with an application of INSERT. Suppose that $e'_p = \text{INSERT}(e'_q, e'_r)$. Then e'_q must be of the form $\langle s_0 \cdot \texttt{+f}\alpha_0, \phi_q\rangle$, and e'_r must be of the form $\langle s_i \cdot_2 \texttt{-f}\alpha'_i, \phi_r\rangle$. Clearly e'_r is of a form that cannot be produced by INSERT, so by the induction assumption there is a G derivation of $\boldsymbol{U}(e'_r)$. Also, e'_q cannot have been produced by INSERT, for if it were, ϕ_q would already contain a feature sequence beginning with `-f` and so there would be *two* such feature sequences in e'_p, preventing application of MRG to e'_p, contrary to our assumptions; therefore there is also a G derivation of $\boldsymbol{U}(e'_q)$.
  Every feature sequence in the lexicon Lex of G contains a selectee feature which precedes all licensee features; so every feature sequence in the lexicon Lex' of G' contains a feature of the form $\texttt{-}\hat{\texttt{g}}$, $\texttt{g} \in Sel$, which precedes all features of the form `-h`, $\texttt{h} \in Lic$. Since none of these features can be checked before becoming part of the tail of some expression, the feature sequence $\texttt{-f}\alpha'_i$ in e'_r must begin with a feature of the form $\texttt{-}\hat{\texttt{g}}$, $\texttt{g} \in Sel$, so $\boldsymbol{U}(\texttt{-f}) = \texttt{g}$ and $\boldsymbol{U}(\texttt{+f}) = \texttt{=g}$. Therefore
  $$\boldsymbol{U}(e'_q) = \langle s_0 \cdot \texttt{=g}\boldsymbol{U}(\alpha_0), \boldsymbol{U}(\phi_q)\rangle$$
  $$\boldsymbol{U}(e'_r) = \langle s_i \cdot_2 \texttt{g}\boldsymbol{U}(\alpha'_i), \boldsymbol{U}(\phi_r)\rangle$$
  $$\text{E-MRG}(\boldsymbol{U}(e'_q), \boldsymbol{U}(e'_r)) = \begin{cases} \langle s_0 s_i : \boldsymbol{U}(\alpha_0), \boldsymbol{U}(\phi_q\phi_r)\rangle & \text{if } \alpha'_i = \epsilon \text{ and } \cdot = :: \\ \langle s_i s_0 : \boldsymbol{U}(\alpha_0), \boldsymbol{U}(\phi_q\phi_r)\rangle & \text{if } \alpha'_i = \epsilon \text{ and } \cdot = : \\ \langle s_0 : \boldsymbol{U}(\alpha_0), \boldsymbol{U}(\phi_q(s_i, \alpha'_i)\phi_r)\rangle & \text{if } \alpha'_i \neq \epsilon \end{cases}$$
  Since $e'_p = \text{INSERT}(e'_q, e'_r)$, we know that $(s_1, \alpha_1), \ldots, (s_n, \alpha_n) = \phi_q(s_i, \texttt{-f}\alpha'_i)\phi_r$, and therefore
  $$\text{E-MRG}(\boldsymbol{U}(e'_q), \boldsymbol{U}(e'_r)) = \boldsymbol{U}(\text{MRG}(e'_p)) = \boldsymbol{U}(e')$$
  so there is a G derivation of $\boldsymbol{U}(e')$, as required.
- The G' derivation of e'_p ends with an application of MRG. None of the feature sequences $\alpha_1, \ldots, \alpha_n$ in the tail of e'_p can begin with a feature of the form $\texttt{-}\hat{\texttt{g}}$, $\texttt{g} \in Sel$, since this is possible only in expressions produced by INSERT. Therefore the feature `-f` in $\alpha_i = \texttt{-f}\alpha'_i$ is of the form `-h`, $\texttt{h} \in Lic$, and so $\boldsymbol{U}(\texttt{-f}) = \texttt{-h}$ and $\boldsymbol{U}(\texttt{+f}) = \texttt{+h}$. Therefore by the induction assumption there is a G derivation of
  $$\boldsymbol{U}(e'_p) = \langle s_0 \cdot \texttt{+h}\boldsymbol{U}(\alpha_0), (s_1, \boldsymbol{U}(\alpha_1)), \ldots, (s_n, \boldsymbol{U}(\alpha_n))\rangle$$
  Now $\alpha_i = \texttt{-f}\alpha'_i = \texttt{-h}\alpha'_i$, and there is no j, $j \neq i$, such that $\boldsymbol{U}(\alpha_j)$ begins with `-h`. And since $\texttt{h} \in Lic$, $\boldsymbol{U}(e'_p)$ is not a lexical expression and so $\cdot = :$. Therefore
  $$\begin{aligned} &\text{I-MRG}(\boldsymbol{U}(e'_p)) \\ &= \begin{cases} \langle s_i s_0 : \boldsymbol{U}(\alpha_0), \boldsymbol{U}((s_1, \alpha_1), \ldots, (s_{i-1}, \alpha_{i-1}), (s_{i+1}, \alpha_{i+1}), \ldots, (s_n, \alpha_n))\rangle & \text{if } \alpha'_i = \epsilon \\ \langle s_0 : \boldsymbol{U}(\alpha_0), \boldsymbol{U}((s_1, \alpha_1), \ldots, (s_i, \alpha'_i), \ldots, (s_n, \alpha_n))\rangle & \text{if } \alpha'_i \neq \epsilon \end{cases} \\ &= \boldsymbol{U}(\text{MRG}(e'_p)) = \boldsymbol{U}(e') \end{aligned}$$
  so there is a G derivation of $\boldsymbol{U}(e')$, as required.

– Induction step, binary case. A binary-branching step in a G' derivation of e' is licensed by applying INSERT to two expressions, so the condition is vacuously satisfied.

If $s \in \mathcal{L}(G')$ then $\langle s \cdot \texttt{-}\hat{\texttt{c}}, \epsilon \rangle \in CL(G')$, and since this expression is clearly not produced by INSERT, $\langle s \cdot \texttt{c}, \epsilon \rangle \in CL(G)$ and $s \in \mathcal{L}(G)$. Thus $\mathcal{L}(G') \subseteq \mathcal{L}(G)$.

We now turn to showing that $\mathcal{L}(G) \subseteq \mathcal{L}(G')$.

Let us say that an expression $\langle s \cdot \alpha, \phi \rangle \in CL(G)$ is *relevant* if and only if ϕ has the form $(s_1, \texttt{-f}_1\alpha_1), (s_2, \texttt{-f}_2\alpha_2), \ldots, (s_n, \texttt{-f}_n\alpha_n)$ for $n \geq 0$, i.e. iff every feature sequence in ϕ begins with a feature of the form $\texttt{-f}$.

Proposition 1. *Every G derivation that contains an irrelevant expression, concludes with an irrelevant expression. (Intuitively: irrelevance is "maintained" by all derivational operations.)*

Note that $\mathcal{L}(G)$ is defined such that it depends only on the relevant expressions in $CL(G)$ (in particular those where $n = 0$ and so $\phi = \epsilon$).

Lemma 4. *If the MG G can derive a relevant expression e, then the IMG G' can derive the expression $e' = \boldsymbol{T}(e)$.*

Proof. – Base case. A length-one G derivation is simply a lexical item $s :: \alpha \in Lex$. For each such lexical item, G' has a lexical item $s :: \boldsymbol{T}(\alpha) \in Lex'$. This constitutes a G' derivation of $\boldsymbol{T}(s :: \alpha)$.

– Induction step, unary case. A unary-branching step in a G derivation tree is licensed by applying I-MRG to some expression $e = \langle s_0 : \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle$. By the induction assumption, there is a G' derivation of $\boldsymbol{T}(e) = \langle s_0 : \texttt{+f}\boldsymbol{T}(\alpha_0), (s_1, \boldsymbol{T}(\alpha_1)), \ldots, (s_n, \boldsymbol{T}(\alpha_n)) \rangle$. Since I-MRG is applicable, there is a unique $\alpha_i, 1 \leq i \leq n$, such that $\alpha_i = \texttt{-f}\alpha_i'$. We are in one of the following two cases:

- $\alpha_i' = \epsilon$, and so

$$\begin{aligned} \text{I-MRG}(e) &= \text{I-MRG}(\langle s_0 : \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle) \\ &= \langle s_i s_0 : \alpha_0, (s_1, \alpha_1), \ldots, (s_{i-1}, \alpha_{i-1}), (s_{i+1}, \alpha_{i+1}), \ldots, (s_n, \alpha_n) \rangle \end{aligned}$$

  But also $\boldsymbol{T}(\alpha_i) = \texttt{-f}$, and there is no α_j, $j \neq i$, such that $\boldsymbol{T}(\alpha_j)$ begins with $\texttt{-f}$, so

$$\begin{aligned} \text{MRG}(\boldsymbol{T}(e)) &= \text{MRG}(\langle s_0 : \texttt{+f}\boldsymbol{T}(\alpha_0), (s_1, \boldsymbol{T}(\alpha_1)), \ldots, (s_n, \boldsymbol{T}(\alpha_n)) \rangle) \\ &= \boldsymbol{T}(\text{I-MRG}(e)) \end{aligned}$$

  and therefore there is a G' derivation of $\boldsymbol{T}(\text{I-MRG}(e))$, as required.

- $\alpha_i' \neq \epsilon$, and so

$$\begin{aligned} \text{I-MRG}(e) &= \text{I-MRG}(\langle s_0 : \texttt{+f}\alpha_0, (s_1, \alpha_1), \ldots, (s_n, \alpha_n) \rangle) \\ &= \langle s_0 : \alpha_0, (s_1, \alpha_1), \ldots, (s_i, \alpha_i'), \ldots, (s_n, \alpha_n) \rangle \end{aligned}$$

But also $\boldsymbol{T}(\alpha_i) = \texttt{-f}\boldsymbol{T}(\alpha'_i)$, and there is no α_j, $j \neq i$, such that $\boldsymbol{T}(\alpha_j)$ begins with `-f`, so

$$\begin{aligned}\textsc{mrg}(\boldsymbol{T}(e)) &= \textsc{mrg}(\langle s_0 : \texttt{+f}\boldsymbol{T}(\alpha_0), (s_1, \boldsymbol{T}(\alpha_1)), \ldots, (s_n, \boldsymbol{T}(\alpha_n))\rangle)\\ &= \langle s_0 : \boldsymbol{T}(\alpha_0), (s_1, \boldsymbol{T}(\alpha_1)), \ldots, (s_i, \boldsymbol{T}(\alpha'_i)), \ldots, (s_n, \boldsymbol{T}(\alpha_n))\rangle\\ &= \boldsymbol{T}(\textsc{i-mrg}(e))\end{aligned}$$

and therefore there is a G' derivation of $\boldsymbol{T}(\textsc{i-mrg}(e))$, as required.

– Induction step, binary case A binary-branching step in a G derivation tree is licensed by applying E-MRG to two expressions $e_1 = \langle s \cdot_1 \texttt{=f}\alpha, \phi\rangle$ and $e_2 = \langle t \cdot_2 \texttt{f}\beta, \psi\rangle$. By the induction assumption, there are G' derivations of $\boldsymbol{T}(e_1)$ and $\boldsymbol{T}(e_2)$. And since e_1 and e_2 are both relevant, no feature sequence in ϕ or ψ begins with a feature of the form `f`, and so no feature sequence in $\boldsymbol{T}(\phi)$ or $\boldsymbol{T}(\psi)$ begins with a feature of the form $\texttt{-}\hat{\texttt{f}}$. We are in one of the following three cases:

- $e_1 = \langle s :: \texttt{=f}\alpha, \epsilon\rangle$ and $e_2 = \langle t \cdot_2 \texttt{f}, \psi\rangle$, so $\textsc{mrg}(e_1, e_2) = \langle st : \alpha, \psi\rangle$. But also $\boldsymbol{T}(e_1) = \langle s :: \texttt{+}\hat{\texttt{f}}\boldsymbol{T}(\alpha), \epsilon\rangle$ and $\boldsymbol{T}(e_2) = \langle t \cdot_2 \texttt{-}\hat{\texttt{f}}, \boldsymbol{T}(\psi)\rangle$, and so

$$\begin{aligned}\textsc{ins}(\boldsymbol{T}(e_1), \boldsymbol{T}(e_2)) &= \langle s :: \texttt{+}\hat{\texttt{f}}\boldsymbol{T}(\alpha), (t, \texttt{-}\hat{\texttt{f}}), \boldsymbol{T}(\psi)\rangle\\ \textsc{mrg}(\textsc{ins}(\boldsymbol{T}(e_1), \boldsymbol{T}(e_2))) &= \langle st : \boldsymbol{T}(\alpha), \boldsymbol{T}(\psi)\rangle\\ &= \boldsymbol{T}(\textsc{e-mrg}(e_1, e_2))\end{aligned}$$

- $e_1 = \langle s : \texttt{=f}\alpha, \phi\rangle$ and $e_2 = \langle t \cdot_2 \texttt{f}, \psi\rangle$, so $\textsc{mrg}(e_1, e_2) = \langle ts : \alpha, \phi\psi\rangle$. But also $\boldsymbol{T}(e_1) = \langle s : \texttt{+}\hat{\texttt{f}}\boldsymbol{T}(\alpha), \boldsymbol{T}(\phi)\rangle$ and $\boldsymbol{T}(e_2) = \langle t \cdot_2 \texttt{-}\hat{\texttt{f}}, \boldsymbol{T}(\psi)\rangle$, and so

$$\begin{aligned}\textsc{ins}(\boldsymbol{T}(e_1), \boldsymbol{T}(e_2)) &= \langle s : \texttt{+}\hat{\texttt{f}}\boldsymbol{T}(\alpha), \boldsymbol{T}(\phi), (t, \texttt{-}\hat{\texttt{f}}), \boldsymbol{T}(\psi)\rangle\\ \textsc{mrg}(\textsc{ins}(\boldsymbol{T}(e_1), \boldsymbol{T}(e_2))) &= \langle ts : \boldsymbol{T}(\alpha), \boldsymbol{T}(\phi), \boldsymbol{T}(\psi)\rangle\\ &= \boldsymbol{T}(\textsc{e-mrg}(e_1, e_2))\end{aligned}$$

- $e_1 = \langle s \cdot \texttt{=f}\alpha, \phi\rangle$ and $e_2 = \langle t \cdot \texttt{f}\beta, \psi\rangle$, so $\textsc{mrg}(e_1, e_2) = \langle s : \alpha, \phi(t, \beta)\psi\rangle$. But also $\boldsymbol{T}(e_1) = \langle s \cdot \texttt{+}\hat{\texttt{f}}\boldsymbol{T}(\alpha), \boldsymbol{T}(\phi)\rangle$ and $\boldsymbol{T}(e_2) = \langle t \cdot \texttt{-}\hat{\texttt{f}}\boldsymbol{T}(\beta), \boldsymbol{T}(\psi)\rangle$, and so

$$\begin{aligned}\textsc{ins}(\boldsymbol{T}(e_1), \boldsymbol{T}(e_2)) &= \langle s \cdot \texttt{+}\hat{\texttt{f}}\boldsymbol{T}(\alpha), \boldsymbol{T}(\phi), (t, \texttt{-}\hat{\texttt{f}}\boldsymbol{T}(\beta)), \boldsymbol{T}(\psi)\rangle\\ \textsc{mrg}(\textsc{ins}(\boldsymbol{T}(e_1), \boldsymbol{T}(e_2))) &= \langle s : \boldsymbol{T}(\alpha), \boldsymbol{T}(\phi), (t, \boldsymbol{T}(\beta)), \boldsymbol{T}(\psi)\rangle\\ &= \boldsymbol{T}(\textsc{e-mrg}(e_1, e_2))\end{aligned}$$

If $s \in \mathcal{L}(G)$ then $\langle s \cdot \texttt{c}, \epsilon\rangle \in CL(G)$, and so by this lemma $\langle s \cdot \texttt{-}\hat{\texttt{c}}, \epsilon\rangle \in CL(G')$ and $s \in \mathcal{L}(G')$. Thus $\mathcal{L}(G) \subseteq \mathcal{L}(G')$.

Theorem 1. *ML ⊆ IML ⊆ MCFL, which, given also that ML = MCFL, entails that IML = ML.*