Formalizing Feature Inheritance

Anonymous ACL submission

1 Introduction

Viewing context-free base rules as structure building operations (a rule $S \rightarrow NP\ VP$ builds an S out of a NP and a VP), the transformational cycle in syntax was a principle that governed the interleaving of transformational operations with context-free structure building operations. In particular, (cyclic) transformational rules were applied only once certain categories (always S, often NP, sometimes PP) of expressions were built. In early minimalism, the transformational rule of movement was interleaved with the structure building operation of merge, however, movement could in principle apply at any time, regardless of the categorial status of its input. Chomsky (2008) has suggested a mechanism of *feature inheritance*, which in effect delays transformations until a particular category is reached. Thus, minimalism with feature inheritance seems to be a return to the original conception of the syntactic cycle.

In this paper we provide a formalization of the mechanism of feature inheritance in the context of minimalist grammars (MGs), itself a formalization of Chomsky's (1995) Minimalist program. The weak generative capacity and worst-case parsing complexity of *feature inheritance* is then compared to that of vanilla MGs.

2 Feature Inheritance

Assume the general clausal architecture of minimalism, where the complementizer head selects the tense head which in turn selects the little-v head which selects the big-V head. A large body of work assumes a shared property between little-v and C; these two heads are said to define locality domains in the syntax (called *phases*). A basic goal expressed by Chomsky (2008) to reduce the stipulations needed in the theory. As little-v and C share one non-trivial property already, determining whether more of their properties can be identified would potentially reduce the number of independent stipulations needed to describe the lexicon. Feature Inheritance (FI) is introduced in (Chomsky, 2008) as a way of reconciling a number of related observations with theoretical assumptions, and is made use of by little-v and C, which increases their similarity a great deal.

A main theoretical motivation for FI is to give a larger role to phases. Phases are said to coincide with the portion of the syntactic structure that the interfaces can refer to. In other words, they are the units that semantic and phonological interpretation are defined over. Chomsky suggests that both interfaces refer to the same units of syntactic structure. In addition, he suggests that syntactic operations (like movement and agreement) are not distributed throughout phases, but are rather deferred until the last head in the phase (little-v or C). This desideratum is problematic from the perspective of orthodox analytical assumptions, as the T head is generally considered to trigger movement of and agreement with the surface subject.

One relevant observation is that only finite T heads trigger movement and agreement. A second observation is that the distribution of finite vs nonfinite T is related to the choice of C: for example, the declarative complementizer `that` selects for finite T, whereas `for` selects for non-finite T. Chomsky's resolution to the problem is to shift the finite-nonfinite distinction over to C, making T into an underspecified tense head. Then it is C which selects for a generic T head, and it must be C which is responsible for triggering movement and agreement **on T**. FI is the mechanism by which movement triggered by a higher head targets the projection of a lower head, which allows for the idea that movement and agreement is deferred until phase heads are introduced to be realized.

C (and little-v) also permit generic movement to their edges, for example, to break long distance movement into phase-sized chunks. Thus C can

trigger movement multiple times, both to its edge, as well as to the edge of the head immediately below it. However, the movements that C now triggers are typically thought to be of two fundamentally different kinds: the movement to T is A-movement, and that targeting C is A-bar-movement. These kinds of movements have importantly different properties (pronouns can be bound after moving over them with A-movement, but not with A-bar-movement, for example), and Chomsky (1995) has proposed that movement steps between the highest A-bar position and the lowest base-merge position of expressions be invisible to various well-formedness conditions. Making the A and A-bar movements which C triggers happen simultaneously (as opposed to serially) structures the movement dependencies entered into by DPs as trees (ordered by derivational order), rather than sequences. This then eliminates the need to postulate an independent operation which deletes intermediate elements in a sequence of movement dependencies – these are no longer on a single branch of the tree.

Feature Inheritance thus paves the way for 1. phase heads to be the locus of movement and agreement triggers, and 2. a novel approach to the distinction between A and A-bar movements.

3 Formal background

We couch our formalization of feature inheritance in the formal framework of minimalist grammars (Stabler, 1997, 2011), an extensible and well-understood grammar formalism capable of transparently representing minimalist analyses. Minimalist grammars are a lexicalized grammar formalism, like categorial grammars, with universal grammatical rules and complex lexical entries. The categories of lexical entries take the form of lists of features, written α or β, called *feature bundles*, where a list is a data structure where only the first element is directly accessible. Removing ('checking') the first element of a nonempty list α results in the remainder of the list α' (so $\alpha = a.\alpha'$). Features have one of two polarities (positive and negative), and come in different kinds, represented as different names ($k,\ wh,\ q,\ d,\ \ldots$). Two features `+x` and `-y` of opposite polarity *match* iff they are of the same kind (i.e. $x = y$).

A syntactic expression is either a pair $\langle w, \alpha\rangle$ consisting of a string of phonemes w and a feature bundle α (written $\mathtt{w}:\alpha$), or a term $\bullet(t_1, t_2)$, where t_1 and t_2 are syntactic expressions, and $\bullet$ is either < or >. The *head* of a syntactic expression t is t itself, if a pair, and the head of t_H if $t = \bullet(t_1, t_2)$, where $t_H = t_1$ if $\bullet = <$, and $t_H = t_2$ if $\bullet = >$.

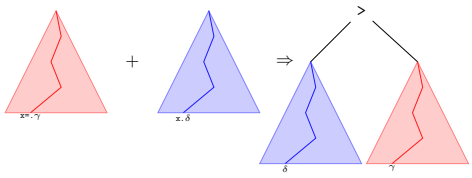

Figure 1: Merge of a specifier

Given a syntactic expression t, the result of checking the first feature of its head is written t'. When t is a term, it represents a tree, and the internal nodes 'point' in the direction of the head. A trace is a pair of the empty string and the empty feature bundle, written `t`.

There are two syntactic operations, **Merge** and **Move**. **Merge** is binary, and **Move** unary. They are both restricted in their application by the feature bundles present in their arguments. The head of the first argument of both operations must be a positive feature. **Merge** applies to two expressions t and s just in case the heads of both have matching first features. **Move** applies to its single argument just in case this argument contains a unique leaf whose first feature matches the first feature of the head.

The output of **Merge** depends on whether its first argument is a leaf or a complex term. If a leaf ℓ, then $\textbf{Merge}(l, s) = <(\ell', s')$, and if a proper term t, $\textbf{Merge}(t, s) = >(t', s')$, as is depicted in figure 1.

Move replaces a subterm of the input with a trace, and so we need a notation which simplifies referring to subterms. We define *maximal projection contexts* $C[x]$ to be either a variable x, or a structure of one of the two forms: $>(C[x], t)$ or $<(t, C[x])$. A maximal projection context $C[x]$ is a term where x occurs without any arrows pointing to it, and replacing the variable x with a term s is written $C[s]$. **Move** applies to t iff $t = C[s]$, where s is a term whose head begins with a negative feature which matches that of t. $\textbf{Move}(C[s]) = >(s', C[\mathtt{t}]')$, as is depicted in figure 2.

Both operations have the effect of removing features from feature bundles one at a time, and features in feature bundles are checked one at a time from left to right.

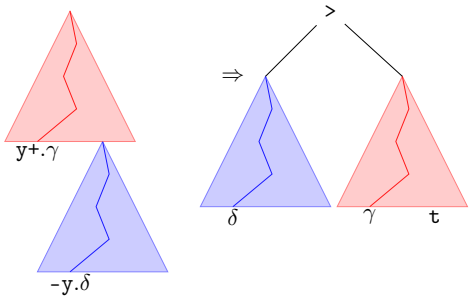

Figure 2: Movement leaves a trace

4 Implementing Feature Inheritance

Feature inheritance diverges from minimalist grammars as they have been defined above in two ways. First, two features can be checked at the same time. Second, movement can target not the top of an expression, but rather some node embedded inside it. To deal with the first difference, we allow feature bundles to contain not just individual features, but also pairs of features. Given a pair of features $\langle \mathsf{+x}, \mathsf{+y} \rangle$, it is intended that they be checked during the same derivational step. To deal with the second difference, we allow positive features to take a diacritic (written: $\mathsf{+x}^{\downarrow}$) indicating that they should target the sister node to the head. We can augment the **Move** operation so that it can deal with these new feature types. For example, given a term t the first feature of the head of which begins with $\mathsf{+x}^{\downarrow}$, whose complement $C[s]$ contains a unique term s with matching first feature, write $t = D[C[s]]$. Then $\textbf{Move}(D[C[s]]) = D[\mathsf{>}(s', C[t])]'$.

These operations allow us to write lexical items with the desired behaviour; Chomsky's C head would have feature bundle $\mathsf{+T}.\langle \mathsf{+k}^{\downarrow}, \mathsf{+wh} \rangle.\mathsf{-C}$, indicating that it merges with a TP, then simultaneously triggers k-movement to TP and wh-movement to itself, and then is itself a CP. However, we are also able to combine pairs and downward diacritics in undesired ways. For example, we could just as easily write the intended $\langle \mathsf{+k}, \mathsf{+wh}^{\downarrow} \rangle$ as $\langle \mathsf{+k}, \mathsf{+wh}^{\downarrow} \rangle$ or $\langle \mathsf{+k}^{\downarrow}, \mathsf{+wh}^{\downarrow} \rangle$ or as $\langle \mathsf{+k}, \mathsf{+wh} \rangle$. The intent of Chomsky's proposal seems best captured by the following well-formedness constraint on lexical feature bundles. A lexical feature bundle containing a feature pair $\langle f, g \rangle$ is well-formed if all of the following obtain:

1. There is exactly one feature pair in the bundle
2. The feature pair is in the second position in the bundle, immediately following a positive feature
3. The first feature in the pair has a downward diacritic
4. No other feature in the bundle has a downward diacritic

These constraints are motivated by the following intuitions about Chomsky's proposal.

1. Feature inheritance happens just once
2. Feature inheritance happens as soon as the complement is merged
3. Feature inheritance involves transfering features to the head of the complement
4. The only way to transfer features to the head of the complement is via feature inheritance

Taken together, these constraints on feature bundles allow for an alternative implementation of feature inheritance. As feature inheritance targets the first merged argument of the head, and takes place immediately after this argument is merged, it is simple to deal with feature inheritance during this **Merge** step, where the top of the second argument is still accessible. Let ℓ be a lexical item whose feature bundle begins with the following two features: $\mathsf{+x}$ and $\langle \mathsf{+y}^{\downarrow}, \mathsf{+z} \rangle$. There are two cases to consider, depending on whether one mover matches both features in the pair, or whether they are matched by different movers. For the first case, let $C[s]$ be a term with first feature $\mathsf{-x}$, and where the first two features of s are $\mathsf{-y}$ and $\mathsf{-z}$. Then $\textbf{Merge}(\ell, C[s]) = \mathsf{>}(s'', \mathsf{<}(\ell'', \mathsf{>}(\mathsf{t}, C[\mathsf{t}]')))$. In the other case, let $C[x, y]$ be a maximal projection context with *two* variables, and let $C[r, s]$ be a term whose first feature is $\mathsf{-x}$, and where the first features of r and s are $\mathsf{-y}$ and $\mathsf{-z}$ respectively. Then $\textbf{Merge}(\ell, C[r, s]) = \mathsf{>}(s', \mathsf{<}(\ell'', \mathsf{>}(r', C[\mathsf{t}, \mathsf{t}]')))$.

It only matters that the movement steps be simultaneous if the same mover is targeted in both cases. We thus require that if a single mover can satisfy both features, it must. This is to avoid a situation where two different movers are targeted simultaneously, and then the first one is targeted by a different positive feature, circumventing the intent of simultaneity.

5 Complexity Analysis

Michaelis (2001) proves the equivalence between minimalist grammars and multiple context-free

grammars, providing a scaffolding for future demonstrations that extensions do not increase generative capacity. To establish such an equivalence, we need to present the modified operations in inference rule format, stated over finite sequences of strings paired with feature bundles. Our revised implemention of feature inheritance only modifies the **Merge** rule (by adding to it two new cases), and so we present just these in inference rule format (see Stabler and Keenan (2003) for the others). In inference rule notation, to each term corresponds a sequence of string-feature bundle pairs. Each pair beyond the first corresponds to a maximal proper subterm whose head begins with negative features. The first pair corresponds to the term minus these moving pieces.

The inference rule **MrgFI1a** describes the situation where there is a single mover, for whom this is the last movement step, and therefore is pronounced in its highest position.

The inference rule **MrgFI1b** describes the situation where the single mover has features left over, and thus continues moving.

The inference rule **MrgFI2a** describes the situation where there are two movers, for both of which this is the last movement step, and therefore are pronounced in their highest positions. In the result, we see that the phonetic part o of the tucking-in mover is sandwiched between the head m selecting the complement, and the pronunciation n of this complement. The inference rule **MrgFI2b** describes the situation where there are two movers, but there is continuing movement. Pursuant to the discussion above, we only allow targeting two independent movers if the first one does not have multiple features, and thus the only possible continuing mover is the second. As noted by Stanojević (2019), parsers derived from the above inference rule notation can have their worst-case time complexity read directly off of the rules themselves. Representing each string as a *span*, a pair of integer variables indicating what portion of the input string that string should cover, the number of distinct variables in the antecedents of a rule polynomially bounds its contribution to worst case complexity. This summation and the associated computational complexity is indicated next to the names of each of the rules above. We see that the rule **MrgFI1b** contributes the most to the worst case time complexity of the new rules. To put this in perspective, the worst case time complexity of minimalist grammars *without* feature inheritance is also $\mathcal{O}(n^{2k+3})$ (Fowlie and Koller, 2017; Stanojević, 2019). Thus minimalist grammars with feature inheritance have the same worst case time complexity as vanilla MGs.

6 Conclusion

We have presented a formalization of Chomsky's ((2008)) mechanism of feature inheritance, which has played an important role in minimalist syntactic theory over the intervening nearly two decades. It is formally innocuous: it increases neither the weak generative capacity nor the worst case time complexity of the MG formalism.

References

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

Noam Chomsky. 2008. On phases. In Robert Freidin, Carlos P. Otero, and Maria Luisa Zubizarreta, editors, *Foundational Issues in Linguistic Theory*, pages 133–166. MIT Press, Cambridge, Massachusetts.

Meaghan Fowlie and Alexander Koller. 2017. Parsing minimalist languages with interpreted regular tree grammars. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 11–20, Umeå, Sweden. Association for Computational Linguistics.

Jens Michaelis. 2001. *On Formal Properties of Minimalist Grammars*. Ph.D. thesis, Universität Potsdam.

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

Edward P. Stabler. 2011. Computational perspectives on minimalism. In Cedric Boeckx, editor, *The Oxford Handbook of Linguistic Minimalism*, Oxford Handbooks in Linguistics, chapter 27, pages 616–641. Oxford University Press, New York.

Edward P. Stabler and Edward L. Keenan. 2003. Structural similarity within and among languages. *Theoretical Computer Science*, 293:345–363.

Miloš Stanojević. 2019. On the computational complexity of head movement and affix hopping. In *Formal Grammar*, pages 101–116, Berlin, Heidelberg. Springer Berlin Heidelberg.

$$\frac{\langle m, \mathsf{+x}.(\mathsf{+y}^{\downarrow}, \mathsf{+z}).\alpha\rangle \qquad \langle n, \mathsf{-x}\rangle, \vec{\phi}, \langle o, \mathsf{-y}.\mathsf{-z}\rangle, \vec{\psi}}{\langle omn, \alpha\rangle, \vec{\phi}, \vec{\psi}}\ \text{MrgFI1a} \qquad \mathcal{O}(n^{2k+2})$$

$$\frac{\langle m, \mathsf{+x}.(\mathsf{+y}^{\downarrow}, \mathsf{+z}).\alpha\rangle \qquad \langle n, \mathsf{-x}\rangle, \vec{\phi}, \langle o, \mathsf{-y}.\mathsf{-z}.\beta\rangle, \vec{\psi}}{\langle mn, \alpha\rangle, \vec{\phi}, \langle o, \beta\rangle, \vec{\psi}}\ \text{MrgFI1b} \qquad \mathcal{O}(n^{2k+3})$$

$$\frac{\langle m, \mathsf{+x}.(\mathsf{+y}^{\downarrow}, \mathsf{+z}).\alpha\rangle \qquad \langle n, \mathsf{-x}\rangle, \vec{\phi}, \langle o, \mathsf{-y}\rangle, \vec{\psi}, \langle p, \mathsf{-z}\rangle, \vec{\chi}}{\langle pmon, \alpha\rangle, \vec{\phi}, \vec{\psi}, \vec{\chi}}\ \text{MrgFI2a} \qquad \mathcal{O}(n^{2k+1})$$

$$\frac{\langle m, \mathsf{+x}.(\mathsf{+y}^{\downarrow}, \mathsf{+z}).\alpha\rangle \qquad \langle n, \mathsf{-x}\rangle, \vec{\phi}, \langle o, \mathsf{-y}\rangle, \vec{\psi}, \langle p, \mathsf{-z}.\beta\rangle, \vec{\chi}}{\langle mon, \alpha\rangle, \vec{\phi}, \vec{\psi}, \langle p, \beta\rangle, \vec{\chi}}\ \text{MrgFI2b} \quad \mathcal{O}(n^{2k+2})$$