All Movement Reprojects

Gregory M. Kobele

Abstract

In order to have a homomorphic mapping between the featural make-up of a phrase and the semantic type of its denotation, we must require that moved phrases always reproject.

That phrases are headed is a central tenet of linguistics. It is the head of a phrase that determines its category, its external distribution. The internal distribution of a phrase, the arrangement of phrases internal to it, is also commonly thought to be similarly determined by the head. The head specifies only certain combinations of dependents as acceptable. This assumption is visually most salient within (lexicalized) tree adjoining grammar (Frank 2002), whereby a lexical entry consists of a tree structure with a specific root node and with open positions for dependents and adjuncts. In categorial grammar as well, the syntactic category of a maximal expression, and the categorial identities of its dependents, are specified by the syntactic types assigned in the lexicon, although there exists the possibility of transferring arguments specified lexically as belonging to one head, to other, higher heads (via hypothetical reasoning), or of eliminating them altogether (via higher order selection). Regardless of details, the general picture is that each lexical item carries with it requests for dependents, as well as specifying the category of its ultimate projection.

In systems like categorial grammar or minimalist grammar, the maximal projections are constructed as part of a derivational process. Here dependents can be combined with the head one at a time up until the head is saturated, at which point it relinquishes control over the derivation, and can then itself be selected as a dependent. In both cases, the categories of the intermediate projections are determined by the category assigned to the head lexical item.

Hornstein & Uriagereka (2002) discuss *reprojection*, which allows a moved

Strict Cycling: A Festschrift for Gereon Müller, 313–324
Silke Fischer, Doreen Georgi, Fabian Heck, Johannes Hein, Anke Himmelreich, Andrew Murphy & Philipp Weisser (eds.)
STRICT CYCLING, Universität Leipzig 2024

expression to wrest control of the phrase from its erstwhile head.[1] Donati & Cecchetto (2011) make use of this mechanism to capture the conflict between the internal and external structure of free relative clauses. Internally, they are clauses, but externally they are nominal. According to Donati & Cecchetto, they are in fact (wh-)clauses, but the moved (nominal) wh-word has projected its category as the category of the phrase.

Reprojection allows the internal distribution of a phrase to be divorced from its external distribution. This is achieved by having one head be responsible for selecting arguments, and another for determining the category of the resulting phrase. When one head 'takes over' from another, it must be the most-recently merged element, if it is assumed that the head of a phrase Y must be the head of all phrases lying between it and Y.[2] If any element could reproject at any time, we might expect to find that the categorial status of phrases should be much more variable than it is. Donati & Cecchetto, observing that free relatives can only be headed by lexical wh-words, restrict reprojectability to moved heads.[3] They observe that this restriction fits naturally with Chomsky's formulations of how to determine which sister projects (Chomsky 2013).

One might of course be skeptical of the utility of labeling algorithms, in which case the mechanism of reprojection looks *ad hoc* and just one of many possible ways to derive free relative clauses.

I will here argue that semantically interpreted movement *always* reprojects. My argument is straightforward, and depends only on the existence of a mapping between syntactic category and semantic type, as is familiar in linguistics from categorial grammar. I will begin by presenting a proposal about how syntactic category and semantic type relate. This proposal will be incompatible with basic and obvious analyses, leading to a simple paradox of sorts. Deconstructing the paradox will lead to the needed formulation of syntactic reprojection.

[1] This is formally very different from reprojective *head movement*, as used for example by Georgi & Müller (2010), which there serves to allow for more flexibility between the c-command relations and linear order between post-head dependents.

[2] This is the axiom of *Succession* in Kornai & Pullum (1990).

[3] Heads seem to undergo a different array of processes than phrases. Nunes (2004) argues that only heads can be multiply spelled out, for example. I will not have anything to say about this.

1. A Formulae-as-types correspondence for minimalist grammars

I adopt the convention (Stabler 1997, Müller 2010) that there are two different kinds of features, one relevant for external **Merge**, and one relevant for operations (**Move**) within an already constructed structure.[4] For each feature type, there are positive and negative variants, which appear on the governor and dependent respectively. These are shown in table 1.

Table 1: Features

	Merge	Move
Positive	$\mathtt{x}^{\bullet}$	$\mathtt{x}^{+}$
Negative	$\mathtt{x}^{\circ}$	$\mathtt{x}^{-}$

Features represent dependencies that lexical entries must enter into, with the polarity of the feature indicating the direction of the dependency. Feature bundles are sequences of features. Just as in Müller (2010), grammatical operations target the first feature in a bundle. I will make a number of assumptions about feature bundles. First, I will assume that all positive features precede all negative features in a feature bundle. This reflects my assumption that the maximal projection of a head must be fully constructed before it can be selected by another. Second, I will assume that there is exactly one negative merge feature in a bundle. This reflects my assumption that a phrase can only be externally merged once. Finally, I will assume that the negative merge feature is the first negative feature in the bundle. This reflects my assumption that an expression must be externally merged before it can be internally merged.

A simple example of a lexicon is given in table 2.

Table 2: A simple lexicon.

every : : $\mathtt{n}^{\bullet}\mathtt{d}^{\circ}.\mathtt{k}^{-}$	will : : $\mathtt{v}^{\bullet}.\mathtt{k}^{+}.\mathtt{s}^{\circ}$
child : : $\mathtt{n}^{\circ}$	laugh : : $\mathtt{d}^{\bullet}.\mathtt{v}^{\circ}$

This lexicon allows for the generation of the single sentence *every child will laugh*, by first merging *every* and *child*, then merging *laugh* and *every child*,

[4] I differ here from Müller (2010) in taking **Agree** to supervene on **Merge**, and not to be a syntactic operation in its own right (Ermolaeva & Kobele 2022).

then merging *will* and *laugh every child*, and then finally moving *every child* to the specifier of *will*. This derivation can be depicted in terms of the tree in figure 1.

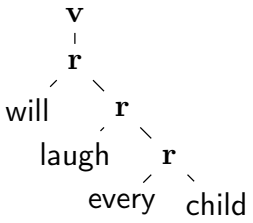

Figure 1: A derivation tree. **v** and **r** abbreviate Move and Merge respectively.

Intuitively, the meaning of this sentence is that every child will, at some future time, laugh. This can be rendered in higher order logic as $\textsc{every}(\textsc{child})(\lambda x.\textsc{will}(\textsc{laugh}\ x))$. We would like to assign each of the four constants used in this term to one of the lexical items in table 2, in the obvious way given in table 3.[5]

Table 3: An interpreted lexicon

EVERY : $(et)(et)t$	$\rightsquigarrow$ every : : $\mathtt{n}^\bullet.\mathtt{d}^\circ.\mathtt{k}^-$
CHILD : et	$\rightsquigarrow$ child : : $\mathtt{n}^\circ$
WILL : tt	$\rightsquigarrow$ will : : $\mathtt{v}^\bullet.\mathtt{k}^+.\mathtt{s}^\circ$
LAUGH : et	$\rightsquigarrow$ laugh : : $\mathtt{d}^\bullet.\mathtt{v}^\circ$

The question I would like to address is to what extent the semantic type of a lexical item's meaning is predictable from its syntactic feature bundle. As a simple observation, the only syntactic argument of *every* must have feature $\mathtt{n}^\circ$, and the first semantic argument of *every* should have type *et*. The lexical item *child*, which has (just) feature $\mathtt{n}^\circ$ has exactly this semantic type. This makes at least initially plausible the idea that positive merge features correspond to semantic arguments.

[5] I here make use of basic types *e* (entities) and *t* (truth values). The type $\alpha \to \beta$, representing the type of functions with inputs of type α and outputs of type β, will be for convenience sometimes represented with simply juxtaposition ($\alpha\beta$). The type forming operator $\to$ is right associative, and parentheses will be left out with this in mind: *abc* abbreviates $a(bc)$, not $(ab)c$.

There are two logically possible semantic configurations for (external) merger. In one, a scope taking element is merged, which must later move to a scope position, leaving behind only a trace. Thus, the selector must semantically select for an expression with the type of the trace. In the other, the merged dependent is interpreted *in situ*. In this case, either expression could in principle take the other as its argument. However, a uniform characterization of the map between features and types is obtained if the selector always takes the selectee as its argument.

These considerations motivate the (partially specified) mapping between feature bundles and types given in table 4. The last line in table 4 expresses

Table 4: A preliminary feature bundle to type correspondence (just Merge features)

$\mathtt{ty}(\mathtt{x}^{\bullet}.\alpha) \mapsto \tau(\mathtt{x}) \to \mathtt{ty}(\alpha)$
$\mathtt{ty}(\mathtt{x}^{\circ}) \mapsto \tau(\mathtt{x})$
$\mathtt{ty}(\mathtt{x}^{\circ}.\alpha) \mapsto (\tau(\mathtt{x}) \to \_) \to \_$

the case when the merged element is a scope taker, and thus has the type of a generalized quantifier $(A \to B) \to C$. The type A is the type of the 'trace' the quantifier leaves behind. The blanks indicate that we are not yet sure what type to put in for the B and C.

We can use this feature-type mapping schema to align the feature bundles and types of our simple lexicon, at least for all words but *will*, which has a positive move feature in it.

Table 5: Aligning actual and calculated types

every	$(e \to t) \to (e \to t) \to t$
$\mathtt{ty}(\mathtt{n}^{\bullet}.\mathtt{d}^{\circ}.\mathtt{k}^{-})$	$\tau(\mathtt{n}) \to (\tau(\mathtt{d}) \to \_) \to \_$
child	$e \to t$
$\mathtt{ty}(\mathtt{n}^{\circ})$	$\tau(\mathtt{n})$
laugh	$e \to t$
$\mathtt{ty}(\mathtt{d}^{\bullet}.\mathtt{v}^{\circ})$	$\tau(\mathtt{d}) \to \tau(\mathtt{v})$

It is easy to visually identify the desired feature to type map that will

associate the types assigned to the lexical items with their feature bundles:

$$\tau = \begin{cases} \mathtt{v} \mapsto t \\ \mathtt{d} \mapsto e \\ \mathtt{n} \mapsto e \to t \end{cases}$$

We can see in addition that, at least for *every*, the 'empty boxes' should both be realized as the propositional type t.

Turning our attention to the remaining lexical item *will*, we note that, were we simply to ignore the positive movement feature $\mathtt{k}^+$, this feature-type map would correctly assign the desired type $t \to t$ to it so long as the feature $\mathtt{s}^\circ$ were mapped to the type t. We present our initial feature bundle to type correspondence in table 6.

Table 6: A preliminary feature bundle to type correspondence

$$\begin{array}{|c|}\hline \mathtt{ty}(\mathtt{x}^\bullet.\alpha) \mapsto \tau(\mathtt{x}) \to \mathtt{ty}(\alpha) \\ \mathtt{ty}(\mathtt{x}^+.\alpha) \mapsto \mathtt{ty}(\alpha) \\ \mathtt{ty}(\mathtt{x}^\circ) \mapsto \tau(\mathtt{x}) \\ \mathtt{ty}(\mathtt{x}^\circ.\alpha) \mapsto (\tau(\mathtt{x}) \to t) \to t \\ \hline \end{array}$$

2. A relative paradox

Consider now adding the lexical items in table 7 to our original lexicon in table 2. These new items are designed to allow for relative clauses according to a raising analysis.

Table 7: Lexical items for the raising analysis of relative clauses.

$$\begin{array}{|cc|}\hline \mathsf{D} :: \mathtt{n}^\bullet.\mathtt{d}^\circ.\mathtt{k}^-.\mathtt{rel}^- & \mathsf{that} :: \mathtt{s}^\bullet.\mathtt{rel}^+.\mathtt{n}^\circ \\ \hline \end{array}$$

Adding the abstract *D* head to a common noun creates a relative DP *D child* waiting to move to check its $\mathtt{rel}^-$ feature. The lexical item *that* selects a sentence, and then triggers movement of a relative DP to its specifier, at which point it becomes a common noun. The NP *child that will laugh* can be derived as in figure 2.

The intuitive meaning of this noun phrase can be represented in higher order logic as the term $\lambda x.\textsc{child}\ x \wedge \textsc{will}(\textsc{laugh}\ x)$. There is only one new

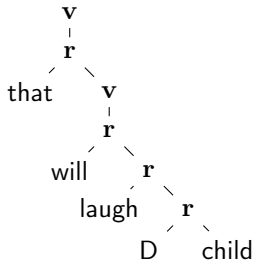

Figure 2: A derivation tree for the NP *child that will laugh*.

constant in this term, the coordinator $\wedge$, but two new lexical entries. Kobele (2006) suggests interpreting the abstract *D* head as coordination, and the relative *that* as the identity function, as shown in table 8. The type of the identity function is polymorphic, and will be seen shortly to be problematic.

Table 8: Interpretations for the relative clause lexical items

$\lambda x.x : \alpha\alpha$	$\rightsquigarrow$ that $::$ $\mathtt{s}^\bullet.\mathtt{rel}^+.\mathtt{n}^\circ$
$\lambda P,Q,x.P\,x \wedge Q\,x : (et)(et)et$	$\rightsquigarrow$ D $::$ $\mathtt{n}^\bullet.\mathtt{d}^\circ.\mathtt{k}^-.\mathtt{rel}^-$

The types associated with these terms are aligned with their calculated types, given by the scheme in table 6, as shown in table 9. The lexical entry

Table 9: Aligning actual and calculated types for relative clause lexical items.

that	$\alpha \to \alpha$
$\mathtt{ty}(\mathtt{s}^\bullet.\mathtt{rel}^+.\mathtt{n}^\circ)$	$\tau(\mathtt{s}) \to \tau(\mathtt{n})$
D	$(e \to t) \to (e \to t) \to (e \to t)$
$\mathtt{ty}(\mathtt{n}^\bullet.\mathtt{d}^\circ.\mathtt{k}^-.\mathtt{rel}^-)$	$\tau(\mathtt{n}) \to (\tau(\mathtt{d}) \to t) \to t$

for *that* diverges from what we would expect based on our previous analytic assumptions. In particular, for *that* to be well-typed, $\tau(\mathtt{s})$ must be identical to $\tau(\mathtt{n})$. But of course, $\tau(\mathtt{s}) = t$ whereas $\tau(\mathtt{n}) = e \to t$. We cannot plausibly treat common nouns as having type t, nor can we reasonably treat propositions as being of type $e \to t$. If we were to attempt to construct a term of type $\tau(\mathtt{s}) \to \tau(\mathtt{n}) = t \to e \to t$, we would find that the only possibility would be a variant of $\mathbb{K} = \lambda\phi,x.\phi$, which does not result in a plausible meaning. As *that*

merges with a sentence (and thus combines with a proposition as its input), it must be lexically assigned the type $t \to t$, which would indeed be what we obtained if its syntactic feature bundle were $\mathtt{s}^{\bullet}.\mathtt{rel}^{+}.\mathtt{s}^{\circ}$, but then we would not be able to assign the desired meaning above to the derivation in 2, as the meaning is of type $e \to t$, but the object derived would have feature $\mathtt{s}^{\circ}$. This then is the paradox of relative clauses.

The type we have calculated for D is also problematic. In particular, we have calculated that the quantifier should scope over a property ($e \to t$), and return a proposition (t), but the desired type scopes over a property and returns a property. It appears we must adjust our type calculation for scope taking expressions. The return value should depend on the feature driving movement, in this case `rel`, as shown in table 10.

Table 10: Updating the feature bundle to type correspondence

$\mathtt{ty}(\mathtt{x}^{\circ}.\alpha) \mapsto (\tau(\mathtt{x}) \to t) \to \mathtt{ty}(\alpha)$
$\mathtt{ty}(\mathtt{x}^{-}) \mapsto \tau(\mathtt{x})$
$\mathtt{ty}(\mathtt{x}^{-}.\alpha) \mapsto \mathtt{ty}(\alpha)$

We observe that the desired types for the scope takers in our grammar emerge if we assign types to atomic features as follows:

$$\tau = \left\{ \begin{array}{ll} & \vdots \\ \mathtt{k} & \mapsto t \\ \mathtt{rel} & \mapsto e \to t \end{array} \right.$$

3. Reprojective resolutions

The problem is that the mover necessarily projects semantically: the mover has type $(A \to B) \to C$, and the expression out from inside of which it moves has type B. The type C is unknown by the lexically specified categorial feature of the head, which dictates at most the type B. To make the syntactic type correctly reflect the semantic type, the syntactic type must be controlled by the mover as well. This means that the category feature of the expression (potentially)

changes after each movement. We write $\mathtt{x}^{\mathtt{y}}$ for the type determined by $\mathtt{y}^{-}$ and $\mathtt{x}^{\circ}$.[6] Table 11 provides a table of some values for this operation.

Table 11: Reprojecting types

inputs	output	type
$\mathtt{s}^{\mathtt{rel}}$	$\mathtt{n}^{\circ}$	*et*
$\mathtt{s}^{\mathtt{k}}$	$\mathtt{s}^{\circ}$	*t*

The problematic lexical feature bundle we assigned to the relative C head, $\mathtt{s}^{\bullet}.\mathtt{rel}^{+}.\mathtt{n}^{\circ}$, is revealed to be $\mathtt{s}^{\bullet}.\mathtt{rel}^{+}.\mathtt{s}^{\mathtt{rel}}$. Thus the lexical entry for the relative C head is simply $\mathsf{C}_{Rel} :: \mathtt{s}^{\bullet}.\mathtt{rel}^{+}.\mathtt{s}^{\circ}$, of semantic type $\tau(\mathtt{s}) \to \tau(\mathtt{s})$ with meaning $\lambda x.x$. Our mistake lay in trying to assign the type x^{y} to a lexical item—the type x^{y} indicates that control has been passed to the mover. Interestingly enough, the natural lexical entry for the +Q C head would be $\mathsf{C}_{Q} :: \mathtt{s}^{\bullet}.\mathtt{wh}^{+}.\mathtt{s}^{\circ}$ as well, also with meaning $\lambda p.p$. A natural generalization has it that there is but one such head, whose combinatory future depends on which feature it ends up attracting to its specifier.

The reprojection operation itself is straightforwardly implementable in the syntax. A concrete implementation (using the 'chain notation' of Stabler & Keenan (2003)) is shown in figure 3.

$$\frac{\mathtt{x}^{+}.\alpha.\mathtt{z}^{\circ}.\beta;\Phi;\mathtt{x}^{-};\Psi}{\alpha.\mathtt{z}^{\mathtt{x}}.\beta;\Phi;\Psi}\ \text{Move}$$

Figure 3: Movement with reprojection

In this rule we see that the moved item is replacing the category of the head to whose specifier it moves, but leaving all of the other features the same. Thus this kind of reprojection is not about 'wresting control' of the derivation from another expression—the identity of the head remains the same after movement—but rather just changing the categorial identity of this expression.

[6] While the movement feature completely determines the semantic type of the resulting expression, the syntactic feature which maps to that type is underdetermined by the movement feature. For example, the `rel` feature dictates that the result of movement should be of type $e \to t$, but that this type should be expressed by the category $\mathtt{n}^{\circ}$ is not obviously derivable in a principled way.

4. Conclusion

I have shown that a homomorphism between syntactic features and semantic type can be imposed upon minimalist grammars. Doing so requires of us that we recognize that semantically interpreted movement always reprojects (though this may at times be vacuous). Having an explicit relation between syntax and semantics allows us to use information about one to constrain our development of the other. Current minimalism has moved away from an explicit characterization of feature bundles, preferring instead to enforce combinatorial constraints more holistically, at interfaces.[7] While the categorial information about lexical items must still be encoded somehow at the interfaces, the tight connection between syntax and semantics becomes muddled and therefore difficult for the linguist to take advantage of, if features are left implicit.

References

Chomsky, Noam. 2013. Problems of projection. *Lingua* 130. 33–49.

Donati, Caterina & Carlo Cecchetto. 2011. Relabeling heads: A unified account for relativization structures. *Linguistic Inquiry* 42(4). 519–560.

Ermolaeva, Marina & Gregory M. Kobele. 2022. Agree as information transmission over dependencies. *Syntax* 25. 466–507.

Frank, Robert. 2002. *Phrase structure composition and syntactic dependencies*, vol. 38 Current Studies in Linguistics. Cambridge, MA: MIT Press.

Georgi, Doreen & Gereon Müller. 2010. Noun-phrase structure by reprojection. *Syntax* 13(1). 1–36.

Graf, Thomas. 2017. A computational guide to the dichotomy of features and constraints. *Glossa* 2. 1–36.

Hornstein, Norbert & Juan Uriagereka. 2002. Reprojections. In Samuel David Epstein & T. Daniel Seely (eds.), *Derivation and explanation in the minimalist program*, chap. 5, 106–132. Oxford: Blackwell.

Kobele, Gregory M. 2006. *Generating copies: An investigation into structural identity in language and grammar*: University of California, Los Angeles dissertation.

Kobele, Gregory M. 2014. Meeting the boojum. *Theoretical Linguistics* 40(1-2). 165–173.

[7] Kobele (2014: section 3.1) notes that the formal properties of minimalist grammars ensure that the effects of features can be pushed off to the interfaces, and the filtering effects of the interfaces can be moved into the features. See Graf (2017) for a more extended discussion.

Kornai, András & Geoffrey K. Pullum. 1990. The X-bar theory of phrase structure. *Language* 66. 24–50.

Müller, Gereon. 2010. On deriving CED effects from the PIC. *Linguistic Inquiry* 41(1). 35–82.

Nunes, Jairo. 2004. *Linearization of chains and sideward movement*. Cambridge, Massachusetts: MIT Press.

Stabler, Edward P. 1997. Derivational minimalism. In Christian Retoré (ed.), *Logical aspects of computational linguistics*, vol. 1328 Lecture Notes in Computer Science, 68–95. Berlin: Springer-Verlag.

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