Inverse Linking via Function Composition

Gregory M. Kobele

University of Chicago

Abstract

The phenomenon of Inverse Linking has proven challenging for theories of the syntax-semantics interface; a noun phrase within another behaves with respect to binding as though it were structurally independent. In this paper I show that, using an LF-movement style approach to the syntax-semantics interface, we can derive all and only the appropriate meanings for such constructions using no semantic operations other than function application and composition. The solution relies neither on a proliferation of lexical ambiguity nor on abandoning the idea that pronouns denote variables, but rather on a straightforward (and standard) reification of assignment functions, which allows us to define abstraction operators within our models.

1 Introduction

Inverse linking (see May and Bale (2005) for an historical overview) refers to the phenomenon of a quantificational noun phrase (QNP) embedded as the argument of a prepositional phrase attached to another QNP taking semantic scope over that noun phrase, as in reading b of example 1, the gross surface structure of which is as sketched in figure 1.

(1) At least two senators on every committee voted against the bill.

   a. At least two senators who are on every committee voted against the bill.

   b. For every committee, there are at least two senators on that committee who voted against the bill.

The challenge the simple existence of inversely linked readings poses for a theory of grammar is to account for how the embedded PP-internal QNP

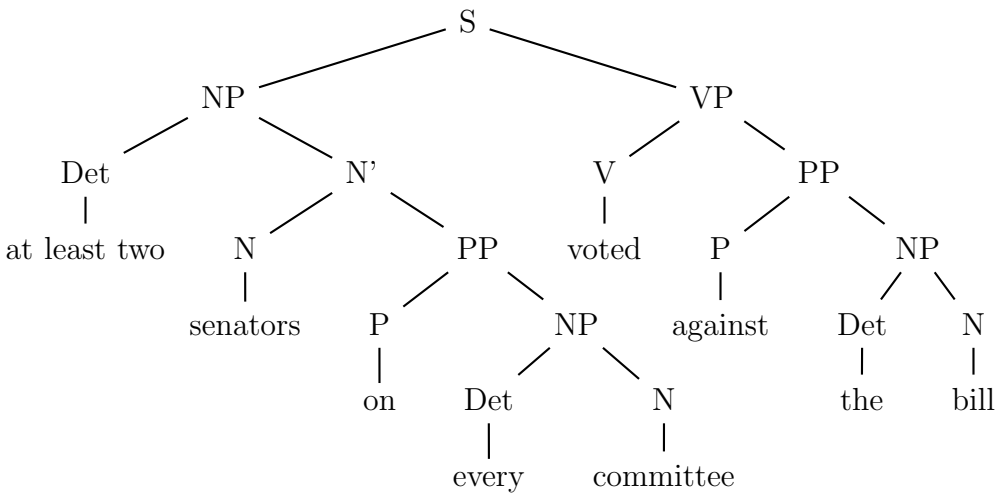


Figure 1: The gross surface structure of example 1

is able to semantically outscope the noun phrase which contains it. There are two major classes of analyses.[1] The first option (May, 1977; Sauerland, 2005), sketched in figure 2, is to interpret the embedded QNP as taking sentential scope (i.e. as scoping entirely outside of the QNP which contains it). The second option (May, 1985; Heim and Kratzer, 1998), as in figure 3, is to interpret the embedded QNP as forming a complex quantificational element with the QNP which contains it. There are two additional empirical constraints on a theory of inverse linking. First is the fact, called *Larson's generalization* in May and Bale (2005), that quantificational noun phrases external to the containing QNP cannot intervene semantically between the embedded and containing QNPs, as shown in example 2.

(2) Two politicians spy on someone from every city.

If the sentence in 2 had a reading in which *every city* took widest scope, and *someone* narrowest, with *two politicians* in between, then it should be true of a situation in which each city has different politicians ($\forall < 2$), but where no two politicians spy on the same individual ($2 < \exists$). However, it doesn't seem to be.

[1]Another alternative is offered by continuations (see, for example,Barker (2002)). Syntactic movement has at the very least intuitive relations to continuations (of the bounded variety (of which there are many)), but the analysis of quantification presented in Barker (2002) is not a natural fit with the syntactic analyses in the minimalist program, even if and when this latter is reanalyzed in terms of (string-)continuations. The same is true of analyses written in the context of syntactic frameworks, such as categorial grammar, in which type changing operations are more natural than in the minimalist framework used here.

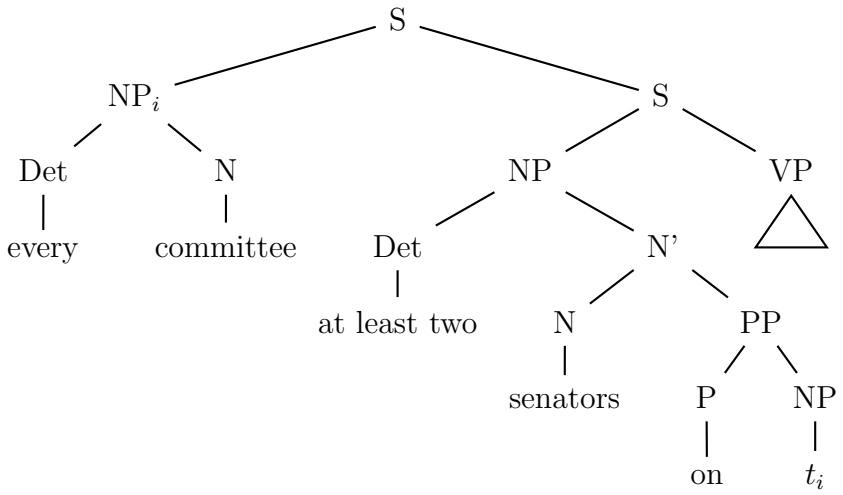


Figure 2: Explaining inverse linking: Sentential scope

The second empirical constraint on a theory of inverse linking comes from the fact that pronouns external to the containing QNP can be bound by the embedded QNP on the inversely linked reading, as in example 3.

(3) Some man from every city secretly despises it.

Example 3 can be true of a situation in which the city despised varies across individuals, i.e. Garrison from Lake Wobegon secretly despises Lake Wobegon, Garth from Waco secretly despises Waco, Gary from Wasilla secretly despises Wasilla, etc.

The sentential scope approach to inverse linking (as in figure 2) immediately accounts for the ability of embedded QNPs to bind variables external to their containing QNP, but requires additional stipulations to correctly rule out cases of scope-splitting. The complex quantifier approach to inverse linking (as in figure 3) on the other hand, while seeming as though it may provide a simple explanation of the scope-splitting prohibition, requires that some method of interpreting 'complex quantifiers' be specified before it is able to make predictions. What would seem to be needed is the following, where $\mathcal{Q}$ is the embedded QNP denotation, and $D(N(x))$ is the containing QNP denotation, where x is the interpretation of the trace of the embedded QNP.[2]

$$\lambda A_{et}.\mathcal{Q}(\lambda x_e.D(N(x))(A))$$

[2]Thus, the denotation of the NP in figure 3 should be the following.

$$\lambda A_{et}.\mathbf{every}(\mathbf{committee})(\lambda x_e.\mathbf{two}(\mathbf{senator} \wedge \mathbf{on}(x))(A))$$

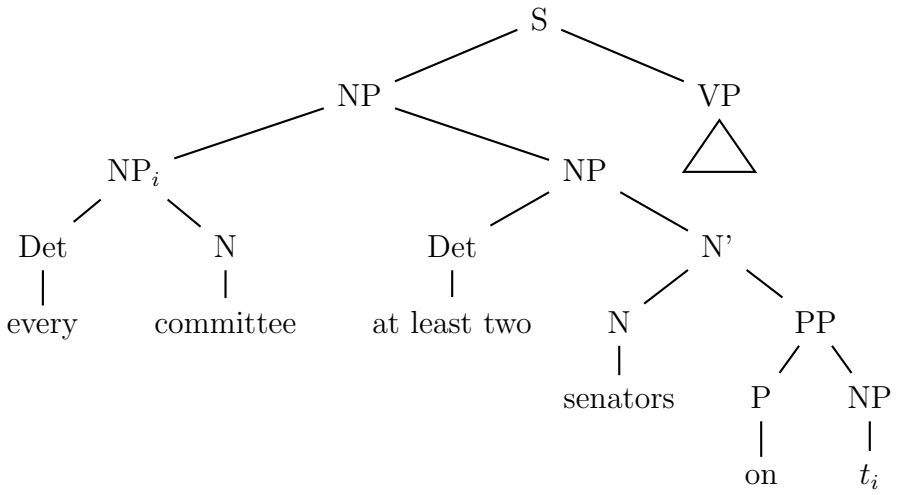


Figure 3: Explaining inverse linking: Complex quantifier formation

As can be verified by inspecting the term above, this denotation correctly rules out the scope splitting cases discussed above (see example 2). One problem with fleshing the complex quantifier approach to inverse linking out in this way is that it seems to require construction specific semantic machinery for its description: how does the meaning of the embedded QNP, $\mathcal{Q}$ combine with the meaning of the containing QNP, $D(N(x))$, so as to result in the (or something like the) meaning representation above? A more fundamental problem is that, although it allows for the embedded quantifier (denoting $\mathcal{Q}$) to take scope over the rest of the sentence (replacing the variable A), which we want in order to account for the binding facts, the representation here does *not* allow pronouns in the sentence to be bound by $\mathcal{Q}$, as the λ-calculus prohibits 'variable capture.'[3] This problem can be seen as undermining the position that pronouns should be interpreted as variables, and thus as an argument for an E-type treatment of pronouns (see Büring (2004) and §4.2).

In contrast to previous accounts of this phenomenon, which were able to maintain the assumption that pronoun binding is variable binding only by introducing purely syntactic distinctions into the semantics (see Larson (1985)

[3]Λ-terms provide an overly fine grained representation of functions, in the sense that infinitely many distinct terms can represent the same function. An example: $\lambda x.x$ is a different term than $\lambda y.y$, although they stand for the very same function. β-conversion is defined so as to make sure that, if α and α' are terms denoting the same function, then $\alpha(\beta)$ and $\alpha'(\beta)$ also denote the same object. For example, $\lambda x.\lambda y.x$ and $\lambda x.\lambda z.x$ both denote the function $\mathbf{K}(a)(b) = a$, and so $(\lambda x.\lambda y.x)(y)$ must denote the same function as $(\lambda x.\lambda z.x)(y)$, namely the function denoted by $\lambda x.y$. For this reason, variable capture is ruled out, as it, being sensitive to accidental properties of our representations, would disrupt the intended equivalences between terms.

and §4.1), the solution I will propose in §3 is that the mechanism underlying complex quantifier formation is simply function composition. Mixing meta- and object-language for the moment, the complex quantifier *a friend of every senator* will have the following form, where $\circ$ denotes function composition.

$$(\mathbf{every}(\mathbf{senator}) \circ \lambda y) \circ \mathbf{some}(\lambda x.\mathbf{friend}(x, y))$$

Applied to a predicate denotation, A, this will yield the following.

$$\begin{aligned}((&\mathbf{every}(\mathbf{senator}) \circ \lambda y) \circ \mathbf{some}(\lambda x.\mathbf{friend}(x, y)))(A) \\ &= (\mathbf{every}(\mathbf{senator}) \circ \lambda y)(\mathbf{some}(\lambda x.\mathbf{friend}(x, y))(A)) \\ &= \mathbf{every}(\mathbf{senator})(\lambda y.\mathbf{some}(\lambda x.\mathbf{friend}(x, y))(A))\end{aligned}$$

This provides a simple solution to the ability of inversely scoped quantifiers to bind into the scope argument of their containing QNP; at the point at which the scope argument is introduced, the variable binding of the inversely scoping QNP has not yet been performed.

Strictly speaking, however, the formula written above is nonsense: λ-abstraction is not a function in our model, and thus it cannot be 'composed' with something that is. This, however, is due to the fact that it has become common to think of assignment functions as parameterizing model-theoretic interpretation, rather than, equivalently, as part of the models themselves. In §2 I review how to reify assignment functions. This allows me to find a 'lambda abstraction' function in our models, which I use in §3 to make legitimate the solution outlined above. In contrast to the solution put forth in Sternefeld (1997), which also makes use of cylindrified models, mine does not treat the binding relation in inverse-linking constructions dynamically. I review previous approaches in §4. Section 5 is the conclusion.

2 The Status of Assignment Functions

It is standard in presentations of the semantics of first-order logic to treat formulae as having interpretations in the models only with respect to assignment functions. In other words, there is no single monolithic interpretation function $[\![\cdot]\!]_{\mathcal{M}}$, but instead a family of interpretation functions $([\![\cdot]\!]^g_{\mathcal{M}})_{g\in G}$ parameterized by assignments. An equivalent perspective reifies the family G of assignment functions, allowing there to be a single interpretation function $[\![\cdot]\!]_{\mathcal{M}}$, the relation of which to the family $([\![\cdot]\!]^g_{\mathcal{M}})_{g\in G}$ can be schematized as per the below.

$$[\![\phi]\!]_{\mathcal{M}} := \{g : [\![\phi]\!]^g_{\mathcal{M}} = \texttt{true}\}$$

In other words, this alternative perspective takes the meaning of a sentence ϕ to be the set of all assignment functions with respect to which ϕ is true on the standard perspective. Despite being equivalent from the perspective of entailment, moving the assignment functions into the model gives access to first rate model theoretic objects which behave like lambda abstraction, but which can be composed with, among other things, generalized quantifier denotations to yield semantic objects of the kind alluded to at the end of the previous section.

Our models are built from the following objects:

- E is the set of *entities*
- T is the set of *propositions*
- G is the set of *contexts of use*

Here, I will take T to be the set $\{0, 1\}$ of truth values, and G to be the set $E^{\mathbb{N}}$ of assignment functions, where $\mathbb{N} = \{0, 1, 2, \ldots\}$ is the set of natural numbers and B^A the set of functions with domain A and codomain B.[4] Given $g, h \in G$, I write g_i for $g(i)$, and $g \approx_i h$ is true just in case if g and h differ, then only in the value they take at i (i.e. for any j, if $g_j \neq h_j$ then $j = i$). The notation $g^{[i:=a]}$ denotes the assignment h, such that $h_i = a$, and $g \approx_i h$. I write $x \in y$ as an abbreviation for $y(x) = 1$.

Natural language expressions denote in domains built from these sets in the following straightforward way.

- $D_e := E^G$
- $D_t := T^G$
- $D_{\alpha\beta} := {D_\beta}^{D_\alpha}$

My use of G is thus similar to Montague's type s, as used in Montague (1970). Viewed from this perspective, my type e is Montague's type se (individual concepts), and my type t Montague's type st (propositions). The difference between my use of G and Montague's s is that the 'denotation domain' of type s was for Montague not just the set of assignment functions (as it is here), but rather the set of pairs of assignment functions and possible worlds.[5]

[4] To deal with intensionality, we can instead take $T = \{0, 1\}^W$, for W an arbitrary set (of *possible worlds*) (Keenan and Faltz, 1985). Furthermore, we can take $G = E^{\mathbb{N}} \times E^{\mathbb{N}}$ to deal with dynamic binding (Groenendijk and Stokhof, 1991).

[5] In Montague (1973), the 'denotation domain' of type s was understood as pairs of worlds and times, with assignment functions relegated to parameters on the interpretation function.

A model $\mathcal{M} = \langle E, I \rangle$ provides an interpretation function I mapping elements of individual constants to elements of E, and i-ary relation symbols to i-ary relations over E. A trace $\mathsf{t_i}$ denotes a function $[\![\mathsf{t_i}]\!]_{\mathcal{M}} \in D_e$ from G to E, such that $[\![\mathsf{t_i}]\!]_{\mathcal{M}}(g) = g_i$. I will call such functions *variables*, and write them $\mathbf{x_i}$ (so in particular $[\![\mathsf{t_i}]\!]_{\mathcal{M}} = \mathbf{x_i}$). We extend $[\![\cdot]\!]_{\mathcal{M}}$ to our other constants in the following way:

- for c an individual constant, $[\![c]\!]_{\mathcal{M}} \in D_e$

$$[\![c]\!]_{\mathcal{M}}(g) = I(c)$$

- for r^i an i-place predicate constant, $[\![r^i]\!]_{\mathcal{M}} \in D_{e^i t}$

$$g \in [\![r^i]\!]_{\mathcal{M}}(a_1) \ldots (a_i) \text{ iff } \langle a_1(g), \ldots, a_i(g) \rangle \in I(r^i)$$

While there are a great many functions from, say, D_e to D_t, the objects we are interested in denote in a very restricted subset of this space. This is captured in the present system by means of the interpretation function I, which assigns 'lowered' meanings to lexical items (so $I(\mathsf{laugh}) \in 2^E$), in terms of which the denotations of these items are defined (so $[\![\mathsf{laugh}]\!] \in D_{et} = [G \to E] \to G \to T$).

The primary difference between this system and the one presented in Montague (1974) is lies in the denotation of predicates, which in Montague's system are functions with domain $E^i \to G \to T$, and which are here functions with domain $\left(E^G\right)^i \to G \to T$.

The payoff for setting things up in this manner is that our models contain functions which act on sentence denotations to yield predicate denotations in the same way the lambda-operator can be prefixed to a sentence to yield a one-place predicate. For every $i \in \mathbb{N}$, we have a function $\lambda_i \in D_{tet}$, which we define as follows.[6] For any $H \in D_t$, and $a \in D_e$,

$$\lambda_i(H)(a) = \{g : g^{[i:=a(g)]} \in H\}$$

With these definitions, it can be proven that, for example,

$$\lambda_1([\![\mathsf{laugh}]\!]_{\mathcal{M}}(\mathbf{x_1}))(a) = [\![\mathsf{laugh}]\!]_{\mathcal{M}}(a)$$

[6]It is worth remarking in how far this defined lambda-abstraction operator faithfully models the behaviour of the lambda in the lambda-calculus. It is clear that they are not identical: the lambda-operator in the lambda calculus 'combines' with a variable of any type σ, and an expression of any type τ to form an expression of type $(\sigma\tau)$. This formal situation recalls the empirical restriction on quantification in natural language argued for by Landman (2005), according to which variables are only of type e.

$$\begin{aligned}\lambda_1([\![\mathsf{laugh}]\!]_{\mathcal{M}}(\mathbf{x_1}))(a) &= \{g : g^{[1:=a(g)]} \in [\![\mathsf{laugh}]\!]_{\mathcal{M}}(\mathbf{x_1})\}\} \\ &= \{g : \mathbf{x_1}(g^{[1:=a(g)]}) \in I(\mathsf{laugh})\} \\ &= \{g : (g^{[1:=a(g)]})_1 \in I(\mathsf{laugh})\} \\ &= \{g : a(g) \in I(\mathsf{laugh})\} \\ [\![\mathsf{laugh}]\!]_{\mathcal{M}}(a) &= \{g : g \in [\![\mathsf{laugh}]\!]_{\mathcal{M}}(a)\}\end{aligned}$$

3 Inverse linking via Function composition

Here I will illustrate the complex quantifier approach to inverse linking, as described in section §1, above.

I will adopt a Heim and Kratzer (1998)-style perspective on semantic interpretation, in the sense that the structures which serve as input to the denotation function $[\![\cdot]\!]_{\mathcal{M}}$ will be syntactic structures to which quantifier raising transformations have already applied. Aside from putting assignments into the model as in the previous section, a major difference in the system here is that indices will be represented on the moved expression (so NP_i is an NP which binds a trace t_i). Such an object will be interpreted as usual, except that its denotation will be composed with a binder for x_i, λ_i:

$$[\![XP_i]\!]_{\mathcal{M}} := [\![XP]\!]_{\mathcal{M}} \circ \lambda_i$$

I adopt a type-driven approach to semantic combination (Klein and Sag, 1985), with basic combinatory operators not only forward and backward function application, but also forward and backward function composition.[7] For α a binary branching node with daughters β and γ, I write $[\![\alpha]\!]_{\mathcal{M}} = \textsc{combine}([\![\beta]\!]_{\mathcal{M}}, [\![\gamma]\!]_{\mathcal{M}})$, where COMBINE is a catch-all for whichever of the above named combinatory operators fits the bill. As mentioned above, a movement subscript contributes a semantic binder; so if the mother is α_i, and the daughters are β and γ, $[\![\alpha_i]\!]_{\mathcal{M}} = \textsc{combine}([\![\beta]\!]_{\mathcal{M}}, [\![\gamma]\!]_{\mathcal{M}}) \circ \lambda_i$.

3.1 Inverse Scope

We can calculate the meaning of the inversely linked reading in 4 below (with gross syntactic structure as in figure 4) in the following manner.

[7] Adding function composition as an available semantic mode of combination means that object quantifier phrases are semantically combinable with predicates in situ. Allowing object DPs to combine with their verbs via function composition is not what we want, as then [John [loves [every girl]]] would translate into (**every**(**girl**) ∘ **love**)(**john**), which means that every girl loves John! Thus, by allowing function composition as a legitimate mode of semantic combination, I am forced to the position that quantifier raising is not 'type driven', but is rather syntactic feature driven.

(4) Some relative of every lawyer despises him.

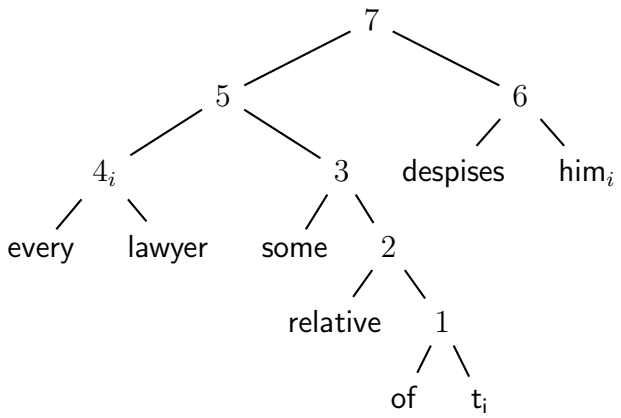


Figure 4: The LF-structure for the inverse scope reading of example 4

The lexical items (the leaves of figure 4) denote as described below.

$$[\![\mathsf{some}]\!]_{\mathcal{M}} = \mathbf{some} \in D_{(et)(et)t}$$
$$[\![\mathsf{every}]\!]_{\mathcal{M}} = \mathbf{every} \in D_{(et)(et)t}$$
$$[\![\mathsf{of}]\!]_{\mathcal{M}} = \mathbf{of} \in D_{ee}$$
$$[\![\mathsf{t_i}]\!]_{\mathcal{M}} = [\![\mathsf{him_i}]\!]_{\mathcal{M}} = \mathbf{x_i} \in D_e$$
$$[\![\mathsf{lawyer}]\!]_{\mathcal{M}} = \mathbf{lawyer} \in D_{et}$$
$$[\![\mathsf{relative}]\!]_{\mathcal{M}} = \mathbf{relative} \in D_{eet}$$
$$[\![\mathsf{despises}]\!]_{\mathcal{M}} = \mathbf{despise} \in D_{eet}$$

The denotations of the internal nodes (which are numbered in the figure), are given below.

1. $[\![1]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{of}]\!]_{\mathcal{M}}, [\![\mathsf{t_i}]\!]_{\mathcal{M}})$. As the type of $[\![\mathsf{of}]\!]_{\mathcal{M}}$ is ee, and the type of $[\![\mathsf{t_i}]\!]_{\mathcal{M}}$ is e, we apply the first argument to the second and obtain a value of type e:
$$[\![\mathsf{of}]\!]_{\mathcal{M}}([\![\mathsf{t_i}]\!]_{\mathcal{M}}) = \mathbf{of}(\mathbf{x_i})$$

2. $[\![2]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{relative}]\!]_{\mathcal{M}}, [\![1]\!]_{\mathcal{M}})$. As the type of $[\![\mathsf{relative}]\!]_{\mathcal{M}}$ is eet, and the type of $[\![1]\!]_{\mathcal{M}}$ is e, we again apply the first argument to the second and obtain a value of type et:
$$[\![\mathsf{relative}]\!]_{\mathcal{M}}([\![1]\!]_{\mathcal{M}}) = \mathbf{relative}(\mathbf{of}(\mathbf{x_i}))$$

3. $[\![3]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{some}]\!]_{\mathcal{M}}, [\![2]\!]_{\mathcal{M}})$. As $[\![\mathsf{some}]\!]_{\mathcal{M}}$ is of type $(et)(et)t$, and the type of $[\![2]\!]_{\mathcal{M}}$ is et, we apply once more the first argument to the second and obtain a value of type $(et)t$:
$$[\![\mathsf{some}]\!]_{\mathcal{M}}([\![2]\!]_{\mathcal{M}}) = \mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i})))$$

4. $[\![4_i]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{every}]\!]_{\mathcal{M}}, [\![\mathsf{lawyer}]\!]_{\mathcal{M}}) \circ \lambda_i$. As the type of $[\![\mathsf{every}]\!]_{\mathcal{M}}$ is $(et)(et)t$, and the type of $[\![\mathsf{lawyer}]\!]_{\mathcal{M}}$ is et, we apply yet again the first argument to the second and obtain a value of type $(et)t$, which composes with λ_i of type tet to yield a value of type tt:

$$[\![\mathsf{every}]\!]_{\mathcal{M}}([\![\mathsf{lawyer}]\!]_{\mathcal{M}}) \circ \lambda_i = \mathbf{every}(\mathbf{lawyer}) \circ \lambda_i$$

5. $[\![5]\!]_{\mathcal{M}} = \textsc{combine}([\![4_i]\!]_{\mathcal{M}}, [\![3]\!]_{\mathcal{M}})$. As the type of $[\![4_i]\!]_{\mathcal{M}}$ is tt, and the type of $[\![3]\!]_{\mathcal{M}}$ is $(et)t$, we compose the first argument with the second and obtain a value of type $(et)t$:

$$(\mathbf{every}(\mathbf{lawyer}) \circ \lambda_i) \circ (\mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i}))))$$

6. $[\![6]\!]_{\mathcal{M}} = \textsc{combine}([\![\mathsf{despises}]\!]_{\mathcal{M}}, [\![\mathsf{him}_i]\!]_{\mathcal{M}})$, as the type of $[\![\mathsf{despises}]\!]_{\mathcal{M}}$ is eet, and the type of $[\![\mathsf{him}_i]\!]_{\mathcal{M}}$ is e, we apply the first argument to the second and obtain a value of type et:

$$[\![\mathsf{despises}]\!]_{\mathcal{M}}([\![\mathsf{him}_i]\!]_{\mathcal{M}}) = \mathbf{despise}(\mathbf{x_i})$$

7. $[\![7]\!]_{\mathcal{M}} = \textsc{combine}([\![5]\!]_{\mathcal{M}}, [\![6]\!]_{\mathcal{M}})$. As the type of $[\![5]\!]_{\mathcal{M}}$ is $(et)t$, and the type of $[\![6]\!]_{\mathcal{M}}$ is et, we apply one last time the first argument to the second and obtain a value of type t:

$$((\mathbf{every}(\mathbf{lawyer}) \circ \lambda_i) \circ (\mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i})))))(\mathbf{despise}(\mathbf{x_i}))$$

By the definition of function composition, this set of assignments is identical to the below:

$$\mathbf{every}(\mathbf{lawyer})(\lambda_i(\mathbf{some}(\mathbf{relative}(\mathbf{of}(\mathbf{x_i})))(\mathbf{despise}(\mathbf{x_i}))))$$

Although not much more will be said about the denotations of common nouns and verbs, the quantifiers and prepositions are intended to be logical constants in our models, and we can calculate on that basis a more refined description of the denotation of this reading.

The denotations of our constants are given in figure 5. As **of** is here taken to denote the identity function, the set of assignments derived above is identical to the below.

$$\mathbf{every}(\mathbf{lawyer})(\lambda_i(\mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i}))))$$

Applying the definition of the function **some** in the above, we obtain the following.

$$\{g : \forall a.\ g \in \mathbf{lawyer}(a) \rightarrow g \in \lambda_i(\mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i})))(a)\}$$

$$\mathbf{some}(A)(B) := \{g : \text{for some } a \in D_e \; g \in A(a) \text{ and } g \in B(a)\}$$

$$\mathbf{of}(a) = a \qquad\qquad \mathbf{x_i}(g) = g_i$$

$$\mathbf{lawyer}(a) = \{g : a(g) \in \texttt{lawyer}\}$$

$$\mathbf{relative}(a)(b) = \{g : \langle a(g), b(g)\rangle \in \texttt{relative}\}$$

$$\mathbf{despise}(a)(b) = \{g : \langle a(g), b(g)\rangle \in \texttt{despise}\}$$

$$\mathbf{every}(A)(B) := \{g : \text{for every } a \in D_e \text{ if } g \in A(a) \text{ then } g \in B(a)\}$$

Figure 5: The denotations of the constants

Unpacking the definition of λ_i from section §2, we arrive at the below.

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow g \in \{h :\; h^{[i:=a(g)]} \in \mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i}))\}\}$$

Rewriting $g \in \{h : \Phi(h)\}$ as $\Phi(g)$ nets us the following.

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow g^{[i:=a(g)]} \in \mathbf{some}(\mathbf{relative}(\mathbf{x_i}))(\mathbf{despise}(\mathbf{x_i}))\}$$

By the definition of **some**, this set is identical to the one below.

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow g^{[i:=a(g)]} \in \{h : \exists b.\; h \in \mathbf{relative}(\mathbf{x_i})(b) \wedge h \in \mathbf{despise}(\mathbf{x_i})(b)\}\}$$

Again, we rewrite $g \in \{h : \Phi(h)\}$ as $\Phi(g)$:

$$\{g : \forall a.\; g \in \mathbf{lawyer}(a) \rightarrow \exists b.\; g^{[i:=a(g)]} \in \mathbf{relative}(\mathbf{x_i})(b) \wedge g^{[i:=a(g)]} \in \mathbf{despise}(\mathbf{x_i})(b)\}$$

Finally, unpacking the definitions of the non-logical constants, we arrive at the description of this set below.

$$\{g : \forall a.\; a(g) \in \texttt{lawyer} \rightarrow \exists b.\; \langle b(g^{[i:=a(g)]}), a(g)\rangle \in \texttt{relative} \wedge \langle b(g^{[i:=a(g)]}), a(g)\rangle \in \texttt{despise}\}$$

I claim that this is the right meaning for the sentence. In particular, that this set is equivalent to the more familiar description below, which we see

immediately to describe G if every lawyer has a relative who despises him, and $\emptyset$ otherwise:

$$\{g : \forall\alpha \in E.\ \alpha \in \texttt{lawyer} \rightarrow \exists\beta \in E.\ \langle\beta, \alpha\rangle \in \texttt{relative} \wedge \langle\beta, \alpha\rangle \in \texttt{despise}\}$$

We can confirm this in the following manner. First, although quantification in the set we derived is over $D_e = [G \rightarrow E]$, the objects quantified over are always applied to assignment functions, resulting in elements of E. In particular, even though we universally quantify over a, we are in fact looking at all possible images of g under a, which is simply E (D_e contains among others functions $\mathbf{e}$ for every $e \in E$ such that $\mathbf{e}(g) = e$ for all g). Similarly for the existential quantification over b; even though b might be 'crazy', at the end of the day, we are only viewing its image in the set of entities E. Thus, if there is some individual $\beta \in E$ such that β is a relative of α, then, regardless of the assignment function g, if we look at enough $b \in D_e$, we are guaranteed to find one such that $b(g) = \beta$. Similarly, if every individual $\alpha \in E$ is such that if he is a lawyer, then he has a relative who despises him, then regardless of which $a \in D_e$ we look at, and which $g \in G$ we apply a to, $a(g)$ is always going to be such an α. The situation here is parallel to quantifying over intentions, but always talking about their extensions at a particular world.

3.2 Direct Scope

Using function composition, we are also able to render the direct scope reading of example 4 without modifying the denotations assigned to our lexical items, simply by locating the landing site of QR beneath the determiner some, as in figure 6. The denotation of which we calculate in the above man-

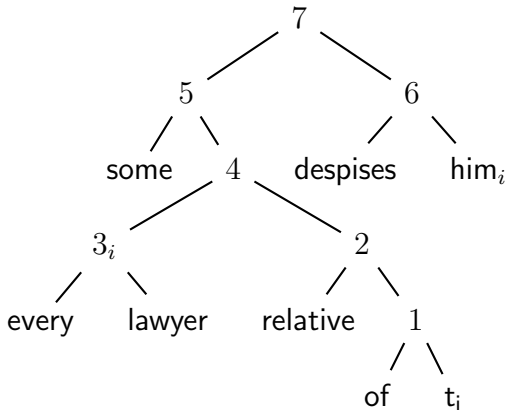


Figure 6: The LF-structure for the direct scope reading of example 4

ner to be $\mathbf{some}((\mathbf{every}(\mathbf{lawyer}) \circ \lambda_i) \circ (\mathbf{relative}(\mathbf{x_i})))(\mathbf{despise}(\mathbf{x_i}))$, which cashes out to

$$\begin{aligned}\{g : \exists b.\ (\forall a.\ g \in \mathbf{lawyer}(a) \rightarrow \\ g^{[i:=a(g)]} \in (\mathbf{relative}(\mathbf{x_i})(b))) \\ \wedge g \in \mathbf{despise}(\mathbf{x_i})(b)\}\end{aligned}$$

As before, we can express this in more familiar terms: this reading is true with respect to an assignment g just in case there is an individual b of whom it is true that he despises g_i, and that for every lawyer a, b is a relative of $(g^{[i:=a]})_i = a$. Note that the QP every lawyer is *not* able to bind pronouns in the scope argument of the QP some relative of every lawyer in this reading.

4 A Comparison with Other Approaches

In what follows I present two alternative analyses of the phenomena of inverse linking, and, in so far as possible, compare them to the one presented herein. The first is the analysis presented by Larson (1985), which provides a mechanism for complex quantifier formation, and makes the standard assumption that pronouns denote variables. Next, I consider the analysis of Büring (2001), who abandons the idea that pronouns denote variables.

4.1 Larson (1985)

Larson (1985) presents a cooper-storage account of inverse-linking (see also Keller (1988)). Modifying Cooper's (1983) original proposal, Larson allows stored expressions to themselves contain stored expressions, et cetera; the denotation of syntactic structures are pairs of expressions, and lists of denotations of syntactic structures (the base case is when the lists are empty). For example, one denotation of the NP *some relative of every lawyer* is as shown below:

$$\langle \mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i}))), \langle \mathsf{every}(\mathsf{lawyer}), \mathsf{x_i} \rangle \rangle$$

When this NP combines with the VP *despises him*, its entire denotation is stored:

$$\langle \mathsf{despises}(\mathsf{x_i})(\mathsf{x_k}), \langle \mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i}))), \langle \mathsf{every}(\mathsf{lawyer}), \mathsf{x_i} \rangle, \mathsf{x_k} \rangle \rangle$$

Having introduced a new 'type' of stored object, Larson is at liberty to define how it is retrieved from the store. In order to capture Larson's generalization, he requires that upon retrieval of a complex expression from

the store, its pieces be applied to the main meaning, in order (from least to most deeply embedded). Continuing with our example above, when $\langle \mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i}))), \langle \mathsf{every}(\mathsf{lawyer}), \mathsf{x_i} \rangle, \mathsf{x_k} \rangle$ is removed from the store, both quantifiers are applied sequentially to the main meaning, starting with the least embedded one:

$$\langle \mathsf{every}(\mathsf{lawyer})(\lambda \mathsf{x_i}.\mathsf{some}(\mathsf{relative}(\mathsf{of}(\mathsf{x_i})))(\lambda \mathsf{x_k}.\mathsf{despises}(\mathsf{x_i})(\mathsf{x_k}))) \rangle$$

The main difference between Larson's account and the one presented herein lies in the nature of the elements on the store (viewing the present account from the perspective of cooper-storage, as in Kobele (2006)). In the account here, stored elements are uniformly model-theoretic objects (functions of type $(et)t$), whereas Larson is forced to view stored elements as (at best) heterogeneous triples consisting of model-theoretic objects, a list of stored elements, and syntactic objects (variable names). Whereas Larson is forced to stipulate his eponymous generalization (by means of his definition of retrieval of complex stored elements), in the present system it is a necessary consequence of the fact that complex quantifiers are formed by function composition.

4.2 Büring (2001)

Büring (2001) (see also Büring (2004)) adopts the complex quantifier perspective on inverse linking, and thus assigns an LF structure to inverse linking sentences similar to that in figure 3. To allow the embedded QNP to combine with the container NP, he introduces a 'composition' combinator, which applies to $\mathcal{Q}$ of type $(et)t$, and $\lambda x.D(N(x))$ of type $e(et)t$ to give $\lambda A_{et}.\mathcal{Q}(\lambda y.D(N(y))(A))$ of type $(et)t$. As per the discussion in footnote 3, free variables in the scope argument of this complex quantifier are unbindable. To overcome this problem, Büring rejects the view of pronouns as denoting variables, and adopts an E-type view of pronouns (see Elbourne (2002) and references therein), according to which the denotation of a pronoun *it* is $\textsc{the}(P)$, where P is a contextually determined property. Thus, according to Büring, the following two sentences have identical LFs:

(5) Some man from every city secretly despises it

(6) Some man from every city secretly despises the city

The semantic co-variance between the pronoun and the embededded quantifier is mediated by virtue of the fact that quantifiers (as shown in figure 7) introduce a universal quantification over minimal situations q satisfying their restrictors (in the case of 5, q would contain a man, a city, and the man

$$[\![Q_\sigma]\!]^g(A)(B) :=$$
$$\{t : \mathcal{Q}(\lambda x.cp(g(\sigma))(t) \in A(x))(\lambda x.\forall q \in min(A(x)).q \in B(x))\}$$

$cp(s)(t)$ denotes the counterpart situation to s in the world of situation t

Figure 7: Quantifiers in an E-type semantics

being from that city). The pronoun (covert definite description of the form the_τ city) in the scope argument then picks out the unique city in each such minimal situation. This is achieved by modifying the assignment function which parameterizes the interpretation function on the quantifier's scope argument to interpret the situation parameter on the definite determiner (τ) as the situation q which was introduced in the semantic scope of the quantifier every, and which is a minimal satisfier of *x is a city that some man is from.*[8]

5 Conclusion

We have seen that making assignment functions first class denizens of our models allows for straightforward implementation of obvious ideas about the interpretation of inverse linking constructions. The ideas outlined above in section §3 show how complex quantifier formation can be done using the functions λ_i and function composition. Of independent interest is that these very same functions and operations allow us to treat syntactic indices on moving expressions in the way syntacticians are used to, assigning a denotation directly to a DP with index i ($[\![DP]\!]_\mathcal{M} \circ \lambda_i$), instead of having to re-arrange structures so as to introduce the indicies in separate syntactic positions (as done by Heim and Kratzer (1998)).

It might be claimed that the incorporation of assignment functions into our models is too high a price to pay, as it makes the denotations of things ugly. Important to keep in mind when evaluating this claim is that the assignment functions have been there all along, as meta-linguistic parameters on an uncountably infinite family of interpretation functions. In contrast to the E-type account of Büring (2004), which is also "nerve-wrackingly

[8]Also necessary is a 'downward closure' operator, $\leq$, which applies to a set of situations s to give back the set of all subsituations of all situation in s ($\leq(s) := \{s' : s' \leq s\}$). This allows, for example, a sentence like *no man's mother despises [the man]* to be understood as satisfying a situation s just in case the set of men's mothers in s is disjoint from the set of individuals for whom every minimal situation q of them being a man's mother can be extended to ($\leq$) a situation of them despising the unique man in q.

complex," the pronouns-as-variables approach explored here can be presented at a high-enough level so as to allow us to do semantics as we are used to, without needing to delve into the details of the semantic objects manipulated.

The debate about whether pronouns should be interpreted as variables or as definite descriptions (or something else) is a complex one, with sophisticated analyses having been developed on all sides. What I hope to have shown here is that, if our semantic meta-language allows for variable capture, an elegant account of inverse linking constructions simply falls out if we treat pronouns as variables.

References

Barker, C. (2002). Continuations and the nature of quantification. *Natural Language Semantics 10*, 211–242.

Büring, D. (2001). A situation semantics for binding out of DP. In R. Hastings, B. Jackson, and Z. Zvolenski (Eds.), *Proceedings from Semantics and Linguistic Theory XI*, Ithaca, pp. 56–75. CLC.

Büring, D. (2004). Crossover situations. *Natural Language Semantics 12*(1), 23–62.

Cooper, R. (1983). *Quantification and Syntactic Theory*. Dordrecht: D. Reidel.

Elbourne, P. (2002). *Situations and Individuals*. Ph. D. thesis, Massachusetts Institute of Technology.

Groenendijk, J. and M. Stokhof (1991). Dynamic predicate logic. *Linguistics and Philosophy 14*, 39–100.

Heim, I. and A. Kratzer (1998). *Semantics in Generative Grammar*. Blackwell Publishers.

Keenan, E. L. and L. M. Faltz (1985). *Boolean Semantics for Natural Language*. Dordrecht: D. Reidel.

Keller, W. R. (1988). Nested cooper storage: The proper treatment of quantification in ordinary noun phrases. In U. Reyle and C. Rohrer (Eds.), *Natural Language Parsing and Linguistic Theories*, Number 35 in Studies in Linguistics and Philosophy, pp. 432–447. Dordrecht: D. Reidel.

Klein, E. and I. A. Sag (1985). Type-driven translation. *Linguistics and Philosophy 8*, 163–201.

Kobele, G. M. (2006). *Generating Copies: An investigation into structural identity in language and grammar*. Ph. D. thesis, University of California, Los Angeles.

Landman, M. (2005). *Variables in Natural Language*. Ph. D. thesis, University of Massachusetts, Amherst.

Larson, R. K. (1985). Quantifying into NP. unpublished ms.

May, R. (1977). *The Grammar of Quantification*. Ph. D. thesis, MIT, Cambridge, Massachusetts.

May, R. (1985). *Logical Form: Its Structure and Derivation*. Cambridge, Massachusetts: MIT Press.

May, R. and A. Bale (2005). Inverse linking. In M. Everaert and H. van Riemsdijk (Eds.), *The Blackwell Companion to Syntax*, Volume 2, Chapter 36, pp. 639–667. Oxford: Blackwell.

Montague, R. (1970). Universal grammar. *Theoria 36*(3), 373–398.

Montague, R. (1973). The proper treatment of quantification in ordinary english. In J. Hintikka, J. Moravcsik, and P. Suppes (Eds.), *Approaches to Natural Language*, pp. 221–242. Dordrecht: D. Reidel.

Montague, R. (1974). English as a formal language. In *Formal Philosophy: Selected Papers of Richard Montague*, Chapter 6, pp. 188–221. New Haven: Yale University Press. edited and with an introduction by R. H. Thomason.

Sauerland, U. (2005). DP is not a scope island. *Linguistic Inquiry 36*(2), 303–314.

Sternefeld, W. (1997). The semantics of reconstruction and connectivity. *Arbeitspapiere des SFB 340 97*, 1–58. Tübingen.

Importing Montagovian Dynamics into Minimalism

Gregory M. Kobele

University of Chicago
kobele@uchicago.edu

Abstract. Minimalist analyses typically treat quantifier scope interactions as being due to movement, thereby bringing constraints thereupon into the purview of the grammar. Here we adapt De Groote's continuation-based presentation of dynamic semantics to minimalist grammars. This allows for a simple and simply typed compositional interpretation scheme for minimalism.

1 Introduction

Minimalist grammars [33] provide a mildly context sensitive perspective on mainstream chomskyian linguistic theory. Although semantic interpretation in chomskyian linguistics is traditionally viewed as operating on derived structures [17], this was faithfully reformulated in terms of a compositional semantics over derivation trees in [23]. There, in keeping with the treatment of pronouns as denoting variables, the standard semantic domains (of individuals E and of propositions T) were paramaterized with the set G of assignment functions, and a function $\diagdown : \left[E^G \to T^G \to E^G \to T^G\right]$ which behaves in a manner similar to lambda abstraction was defined. Around the same time, a continuation-based reinterpretation of dynamic semantics using the simply typed lambda calculus was presented [8]. Instead of being variables, pronouns are treated there as (lifted) choice functions over *contexts*, which parameterize the type o of propositions.[1]

In this paper, we adopt the choice function treatment of pronouns [8], and reformulate the non-canonical semantics of minimalist grammars [23] in these terms. This allows for a simply typed and variable free (§3.3) presentation of minimalist semantics, within which constraints on quantifier scoping are most naturally formulated syntactically. This constrasts with a previous semantic interpretation scheme for (a logical reconstruction of) minimalist grammars [2, 27], which, using the lambda-mu calculus [29] to represent the meanings of sentences, treated scope taking as a consequence of different reduction orders. (And which, as a consequence, was not able to account for the various seemingly syntactic constraints on scope.)

The main 'data' to be covered include the following sentences.

[1] A greater similarity with [23] emerges if we do not lift the pronouns of [8], but instead treat them as denoting functions from contexts to individuals. Then both propositions *and* individuals must be parameterized by contexts.

1. Every boy believed that every man believed that he smiled.
2. Some man believed that every woman smiled.
3. Some man believed every woman to have smiled.

The interest in these sentences is as follows. Sentence 1 has a reading in which the pronoun *he* is bound by the matrix subject *every boy*. This can be 'straightforwardly' dealt with if *he* denotes a variable, as long as this variable has a different name than the one bound by the embedded subject. If we identify variable names with movement features (as we will here), this cannot be done. Sentences 2 and 3 differ in the scope taking behavior of the quantified noun phrase *every woman*. In 2, this QNP must scope under the matrix subject, while in 3 either scope order is possible.

The remainder of this paper is structured as follows. Section §2 introduces the minimalist grammar formalism. Section 3 presents a semantics for this formalism in terms of the simply typed lambda calculus. In §4, a grammar fragment is presented which allows for quantifiers to scope out of arbitrarily many non-finite clauses, but not past a tense clause boundary, as is the received wisdom in the linguistic literature [21]. Section 5 is the conclusion.

2 Minimalist Grammars

Minimalist grammars make use of two syntactic structure building operations; binary **merge** and unary **move**. **Merge** acts on its two arguments by combining them together into a single tree. The operation **move** rearranges the pieces of its single and syntactically complex argument. The generating functions **merge** and **move** are not defined on all objects in their domain. Whether a generating function is defined on a particular object in its domain (a pair of expressions in the case of **merge**, or a single expression in the case of **move**) is determined solely by the syntactic categories of these objects. In minimalist grammars, syntactic categories take the form of 'feature bundles', which are simply finite sequences of features. The currently accessible feature is the feature at the beginning (leftmost) position of the list, which allows for some features being available for checking only after others have been checked. In order for **merge** to apply to arguments Γ and Δ, the heads of both expressions must have matching first features in their respective feature bundles. These features are eliminated in the derived structure which results from their merger. In the case of **move**, the head of its argument Γ must have a feature matching a feature of the head of one of its subconstituents' Δ. In the result, both features are eliminated. Each feature type has an attractor and an attractee variant (i.e. each feature is either positive or negative), and for two features to match, one must be positive and the other negative. The kinds of features relevant for the **merge** and **move** operations are standardly taken for convenience to be different. For **merge**, the attractee feature is a simple categorial feature, written `x`. There are two kinds of attractor features, `=x` and `x=`, depending on whether the selected expression is to be merged on the right (`=x`) or on the left (`x=`). For the **move** operation, there is

a single attractor feature, written +y, and two attractee features, -y and $\ominus$y, depending on whether the movement is overt (-y) or covert ($\ominus$y).

A lexical item is an atomic pairing of form and meaning, along with the syntactic information necessary to specify the distribution of these elements in more complex expressions. We write lexical items using the notation $\langle \sigma, \delta \rangle$, where σ is a lexeme, and δ is a feature bundle.

Complex expressions are written using the notation of [33] for the 'bare phrase structure' trees of [5]. These trees are essentially X-bar trees without phrase and category information represented at internal nodes. Instead, internal nodes are labeled with 'arrows' > and <, which point to the head of their phrase. A tree of the form $[< \alpha\ \beta]$ indicates that the head is to be found in the subtree α, and we say that α projects over β, while one of the form $[> \alpha\ \beta]$ that its head is in β, and we say that β projects over α. Leaves are labeled with lexeme/feature pairs (and so a lexical item $\langle \alpha, \delta \rangle$ is a special case of a tree with only a single node). The head of a tree t is the leaf one arrives at from the root by following the arrows at the internal nodes. If t is a bare phrase structure tree with head H, then I will write t[H] to indicate this. (This means we can write lexical items $\langle \alpha, \delta \rangle$ as $\langle \alpha, \delta \rangle[\langle \alpha, \delta \rangle]$.) The **merge** operation is defined on a pair of trees t_1, t_2 if and only if the head of t_1 has a feature bundle which begins with either =x or x=, and the head of t_2 has a feature bundle beginning with the matching x feature. The bare phrase structure tree which results from the merger of t_1 and t_2 has t_1 projecting over t_2, which is attached either to the right of t_1 (if the first feature of the head was =x) or to the left of t_1 (if the first feature of the head was x=). In either case, both selection features are checked in the result.

$$\textbf{merge}(t_1[\langle \alpha, \texttt{=x}\delta \rangle], t_2[\langle \beta, \texttt{x}\gamma \rangle]) = \begin{array}{ccc} & < & \\ t_1[\langle \alpha, \delta \rangle] & & t_2[\langle \beta, \gamma \rangle] \end{array}$$

$$\textbf{merge}(t_1[\langle \alpha, \texttt{x=}\delta \rangle], t_2[\langle \beta, \texttt{x}\gamma \rangle]) = \begin{array}{ccc} & > & \\ t_2[\langle \beta, \gamma \rangle] & & t_1[\langle \alpha, \delta \rangle] \end{array}$$

If the selecting tree is both a lexical item and an affix (which I notate by means of a hyphen preceding/following the lexeme in the case of a suffix/prefix), then head movement is triggered from the head of the selected tree to the head of the selecting tree.

$$\textbf{merge}(\langle \text{-}\alpha, \texttt{=x}\delta \rangle, t_2[\langle \beta, \texttt{x}\gamma \rangle]) = \begin{array}{ccc} & < & \\ \langle \beta\text{-}\alpha, \delta \rangle & & t_2[\langle \epsilon, \gamma \rangle] \end{array}$$

The operation **move** applies to a single tree $t[\langle \alpha, \texttt{+y}\delta \rangle]$ only if there is *exactly one* leaf ℓ in t with matching first feature -y or $\ominus$y.[2] This is a radical version of the shortest move constraint [5], and will be called the SMC – it requires that an

[2] Other constraints have been explored in [12].

expression move to the first possible landing site. If there is competition for that landing site, the derivation crashes (because the losing expression will have to make a longer movement than absolutely necessary). If it applies, **move** moves the maximal projection of ℓ to a newly created specifier position in t (overtly, in the case of $\texttt{-y}$, and covertly, in the case of $\ominus\texttt{y}$), and deletes both licensing features. To make this precise, let $t\{t_1 \mapsto t_2\}$ denote the result of replacing all subtrees t_1 in t with t_2, for any tree t, and let ℓ_t^M denote the maximal projection of ℓ in t, for any leaf ℓ.

$$\mathbf{move}(t[\langle \alpha, \texttt{+y}\delta\rangle]) = \begin{array}{c} > \\ t'[\langle \beta, \gamma\rangle] \quad t[\langle \alpha, \delta\rangle]\{t' \mapsto \langle \epsilon, \epsilon\rangle\} \end{array} \quad (\text{where } t' = \langle \beta, \texttt{-y}\gamma\rangle_t^M)$$

$$\mathbf{move}(t[\langle \alpha, \texttt{+y}\delta\rangle]) = \begin{array}{c} > \\ \langle \epsilon, \gamma\rangle \quad t[\langle \alpha, \delta\rangle]\{t' \mapsto t'[\langle \beta, \epsilon\rangle]\} \end{array} \quad (\text{where } t' = \langle \beta, \ominus\texttt{y}\gamma\rangle_t^M)$$

2.1 The Shortest Move Constraint

Minimalist grammars with the shortest move constraint were proven [28] to be weakly equivalent to multiple context free grammars [31]. The proof that minimalist languages are contained in the MCFLs proceeds by constructing an equivalent MCFG whose derivation trees are identical to those of the minimalist grammar (modulo a projection). Each component of a derived tuple of strings in the target MCFG corresponds to either a moving subexpression in the MG or to the fixed head and non-moving material around it. The shortest move constraint ensures that there is a finite upper bound on the number of possible moving subexpressions, and thus on the dimension of the target MCFG.

The derived trees of Minimalist Grammars, while not corresponding to so natural a class due to non-logical restrictions (such as the distribution of traces), are contained [26] in the tree languages derivable by multiple regular tree grammars, which are MCFGs where the derived tuples contain trees instead of strings [10].

These observations motivate the idea that, at least in the context of the SMC, the natural data structure for the objects derived by minimalist grammars are tuples, where all positions but the first are indexed by feature types. An alternative presentation of this data structure is as a *store*, in the sense of [6], which is a pair of an object and a finite map from feature types to objects. In the case of syntax, the objects are trees. In the case of semantics, they will turn out to be simply typed lambda terms, as explained next.

2.2 Derivations

A derivation tree is an element of the term language over the ranked alphabet $A_0 \cup A_1 \cup A_2$, where $A_0 = Lex$ is the set of nullary symbols, $A_1 = \{\mathbf{move}\}$

is the set of unary symbols, and $A_2 = \{\textbf{merge}\}$ the set of binary symbols. As a consequence of the translation of minimalist grammars into multiple context free grammars [28, 16], and as described in [26], the set of derivation trees in a minimalist grammar of an expression with unchecked feature string γ at the root and no features anywhere else is regular.

For reasons of space (and because the derivation tree is more informative than any single derived tree), we will present only derivation trees for the expressions in this paper.

3 Minimalist Semantics

Here we present a rule-by-rule semantic interpretation scheme for minimalist grammars. The denotation of a syntactic object t is a pair of a simply typed lambda term and a quantifier store. A quantifier store is a partial function from feature types to simply typed terms. The idea is that a syntactic object t with a moving subexpression $t' = \langle \beta, \texttt{-y}\gamma \rangle_t^M$ has a quantifier store $\mathcal{Q}$ such that $\mathcal{Q}(y)$ is the stored meaning of t'.

We use lower case greek letters ($\alpha, \beta, \ldots$) to stand for denotations of syntactic objects (pairs of simply typed lambda terms and quantifier stores), and the individual components of these denotations will be referred to with the corresponding roman letters in lowercase for the lambda term component, and in uppercase for the store component.[3] Thus, $\alpha = \langle a, A \rangle$.

We treat lexical items as being paired with the empty store.

Quantifier stores With $\emptyset$ we denote the empty quantifier store, such that for all feature types F, $\emptyset(f)$ is undefined. If two quantifier stores $\mathcal{Q}_1, \mathcal{Q}_2$ have disjoint domains,[4] then $\mathcal{Q}_1 \vee \mathcal{Q}_2$ denotes the store such that:

$$\mathcal{Q}_1 \vee \mathcal{Q}_2(f) := \begin{cases} \mathcal{Q}_1(f) & \text{if defined} \\ \mathcal{Q}_2(f) & \text{otherwise} \end{cases}$$

Let $\mathcal{Q}$ be a quantifier store, and F, G feature types. Then $\mathcal{Q}/f$ is the quantifier store just like $\mathcal{Q}$ except that it is undefined on F, $\mathcal{Q}[f := \alpha]$ is the store just like $\mathcal{Q}$ except that it maps F to α, and $\mathcal{Q}_{f \leftarrow g}$ is the store just like $\mathcal{Q}/g$ except that it maps F to whatever $\mathcal{Q}$ mapped G to.

Variable naming Our approach to variable naming is from [32], and is based on the observation that, in the context of the SMC, the type of the next feature of a moving expression uniquely identifies it. We will thus need no free individual variables (those of type e) other than these, and thus we subscript them with feature types.

[3] Lower case greek letters also have been used to stand for feature sequences, lexemes, etc. This overloading is hoped to be clear from context.

[4] If this is not the case, then the syntactic expressions they correspond to cannot be syntactically merged, as this would result in a violation of the SMC.

3.1 Merge

There are two possible semantic reflexes of syntactic merger. The first case is function application from one argument to the other (in a type driven manner). In the second case one of the arguments is a moving expression, and the other denotes a function from individuals. Here we insert it into the store indexed by the next licensee feature type it will move to check. The other argument is applied to a variable of the same name as the index under which the moving argument was stored. We denote these two semantic operations **mergeApp** and **mergeStore**.[5]

$$\textbf{mergeApp}(\alpha, \beta) = \begin{cases} \langle a(b),\ A \cup B \rangle \\ \qquad \text{or} \\ \langle b(a),\ B \cup A \rangle \end{cases}$$

$$\textbf{mergeStore}(\alpha, \beta) = \langle a(x_f),\ A \cup B[f := b] \rangle$$
(where β's next feature is `-f`)

3.2 Move

There are multiple possible semantic reflexes of syntactic movement as well.[6] The first, **moveEmpty**, is used when the moving expression has previously taken scope. The second, **moveLater**, will be used when the moving expression is going to take scope in a later position. The third, **moveNow**, is used when the moving expression is going to take scope in this position. Here, the appropriate variable is abstracted over, and the resulting predicate is given as argument to the stored expression. In the below, we assume `-f` to be the feature checked by this instance of **move**.

$$\textbf{moveEmpty}(\alpha) = \langle a, A \rangle$$
where A is undefined at f

$$\textbf{moveLater}(\alpha) = \langle (\lambda x_f.a)(x_g), A_{g \leftarrow f} \rangle$$
for `-g` the next feature to check

$$\textbf{moveNow}(\alpha) = \langle A(f)(\lambda x_f.a),\ A/f \rangle$$

[5] Note that the condition on **mergeStore** refers to the features of the moving expression. This information is present in the categories used in the MCFG translation of a minimalist grammar, and is a finite state bottom up relabeling of the standard minimalist derivation tree. Thus while not a homomorphic interpretation of minimalist derivations, it is a homomorphic interpretation of a finite state relabeling of minimalist derivations, or a transductive interpretation of minimalist derivations.

[6] [24] argues for the inclusion of function composition, in order to account for inverse linking constructions, which are beyond the scope of this paper.

3.3 Going Variable Free

As observed in [23], remnant movement wreaks havoc with the interpretation of movement dependencies presented here. The problem is that we can end up with a term in which variables remain free, with the lambda which was supposed to have bound them to their right. The reason this problem exists is that the semantics presented here treats each moving expression independently of all others, while the existence of remnant movement forces us to 'coordinate' the interpretations of two moving expressions where one contains the base position of the other. There are various strategies for resolving this difficulty [23], however, the one which suggests itself in the present simply typed context is nice because it simply eliminates free variables (named after features or not).

The basic idea is straightforward: an expression $\alpha = \langle a, A\rangle$ 'abbreviates' an expression $\alpha' = \langle \lambda x_{f_1}, \ldots, x_{f_n}.a, A\rangle$, where the domain of A is exactly the set of features $f_1, \ldots, f_n$. In other words, we take the 'main denotation' of an expression to have the variables expressions in the store may have introduced in it already and always bound. For any store A, define $\mathsf{var}(A)$ to be a fixed enumeration of the domain of A (viewed as variables), and $\Lambda(A).\ \phi$ to be a prefix of lambda abstractions over the domain of A viewed as variables. Then the **mergeApp** rule can be given as follows:

$$\mathbf{mergeApp}(\alpha, \beta) = \begin{cases} \Lambda(A \cup B).\ a(\mathsf{var}(A))(b(\mathsf{var}(B))),\ A \cup B\rangle \\ \qquad\qquad \text{or} \\ \langle \Lambda(B \cup A).b(\mathsf{var}(B))(a(\mathsf{var}(A))),\ B \cup A\rangle \end{cases}$$

The rules **moveEmpty** and **moveLater** are also simply recast in these terms.

$$\mathbf{moveEmpty}(\alpha) = \langle \Lambda(A).a(\mathsf{var}(A)), A\rangle$$
$$\mathbf{moveLater}(\alpha) = \langle \Lambda(A_{j\leftarrow i}).(\lambda x_i.a(\mathsf{var}(A)))(x_j), A_{j\leftarrow i}\rangle$$

Not everything is so straightforward, unfortunately. What are we to do with the **mergeStore** rule, when the expression whose denotation is to be stored itself contains moving pieces? An example is verb phrase topicalization (in a sentence like *"Use the force, Luke will"*). In a typical minimalist analysis of this construction, the lexical item *will* merges with the verb phrase t_k *use the force*, which itself contains a moving expression, the subject $Luke_k$. Thus, under our 'variable free' view, we have a VP with main denotation of type *et* (because its store contains the denotation of *Luke*), but *will* is of type *tt*. There is a natural resolution to this (self-inflicted) problem, which implements the transformational observation that 'remnant movement obligatorilly reconstructs' [4]. We split the **mergeStore** rule into **mergeStoreMB**,[7] which simply stores expressions with empty stores, and **mergeStoreHO**,[8] which stores the denotation of a merging

[7] For '**M**onadic **B**ranching' [22].
[8] For '**H**igher **O**rder'.

expression with moving pieces, and inserts a higher order variable.

$$\textbf{mergeStoreMB}(\alpha, \langle b, \emptyset \rangle) = \langle \Lambda(X).\ a(\mathsf{var}(A))(x_i),\ X \rangle \quad (\text{for } X = A[i := b], \text{ and } \mathtt{ty}(b) = (\mathtt{ty}(x_i) \to \gamma) \to \delta.)$$

$$\textbf{mergeStoreHO}(\alpha, \beta) = \langle \Lambda(X).\ a(\mathsf{var}(A))(x_i(\mathsf{var}(B))),\ X \rangle \quad (\text{for } X = A \cup B[i := b], \text{ and } \mathtt{ty}(x_i) = \mathtt{ty}(b).)$$

The quantifier store holds both generalized quantifiers, as before, as well as relations like $\lambda x.x$ *uses the force*. This latter type of expression seems to be used as an argument of some operator (such as a focus operator). Accordingly, we adjust the **moveNow** rule to allow a type driven retrieval scheme:

$$\textbf{moveNowFunc}(\alpha) = \langle \Lambda(X).\ A(i)(\lambda x_i.a(\mathsf{var}(A)),\ X \rangle \qquad (\text{for } X = A/i)$$

$$\textbf{moveNowArg}(\alpha) = \langle \Lambda(X).\ \lambda x_i.a(\mathsf{var}(A))(A(i)),\ X \rangle \qquad (\text{for } X = A/i)$$

This is not a particularly interesting semantic treatment of remnant movement, as it simply implements obligatory 'semantic reconstruction' [7]. To properly implement linguistic analyses, it seems that some lexical items need access to the stores of their arguments. This would allow us to assign the following interpretation to a topicalization lexical item, which checks the topic feature of a moving expression, and returns a bipartite structure of the form $\mathtt{top}(\phi)(\psi)$, which asserts that ψ is the topic of ϕ: $[\![\langle \epsilon, \texttt{=t +top c} \rangle]\!](\alpha) = \Lambda(A).\mathtt{top}(\lambda x_{top}.a(v(A)))(x_{top})$. This and alternatives need to be worked out further.

3.4 Determining Quantifier Scope

It is natural to view the minimalist grammar operations as pairs of syntactic and semantic functions. Thus, for example, we have both $\langle \mathtt{merge}, \textbf{mergeApp} \rangle$ and $\mathtt{merge}, \textbf{mergeStore} \rangle$. The semantic interpretation rules implement a reconstruction theory of quantifier scope [19], according to which the positions in which a quantified noun phrase can take scope are exactly those through which it has moved.

A shortcoming of this proposal as it now stands is that the yield language of the well formed derivation trees is ambiguous (in contrast to the uninterpreted minimalist grammar system [15]). Because minimalist grammar derivation tree languages are closed under intersection with regular sets [14, 25], any regular strategy for determining which of the possible positions a quantified noun phrase should be interpreted in can be used to give control over scope taking back to the lexicon. A simple such strategy is to assign to a licensee feature which may be used at **moveNow** a particular diacritic. A moving expression is then required to take scope at the highest (derivational) position permitted in which it checks its features, or in its merged position, should no other possibility obtain. We will adopt this proposal here, writing a licensee feature of type F which requires scope taking with a hat ($\mathtt{-\hat{f}}$ or $\ominus\hat{\mathtt{f}}$), and one which does not without ($\mathtt{-f}$ or $\ominus\mathtt{f}$). This move restores the functionality of the relation between sequences of lexical

items and well-formed derivations, at the cost of increasing the size of the lexicon (by an additive factor).

3.5 Lexical Interpretations

A model is given by a set of atomic individuals E, a set of propositional interpretations T, and an interpretation function $\mathcal{I}$ assigning to each lexical item a model theoretic object in a set built over E and T. We assume throughout that $\mathcal{I}$ assigns to lexical items denotations of the 'standard' type; common nouns denote functions from individuals to propositions, n-ary verbs functions from n individuals to propositions, determiners relations between common noun denotations, etc.

We call the type of atomic individuals e, and that of propositions as o. Following [8], let the type of a (discourse) context be γ, and a sentence (to be revised shortly) to evaluate to a proposition only in a context (i.e. to be a function from contexts to propositions). Contexts will serve here to provide the input to pronoun resolution algorithms. For simplicity, we will treat them as lists of individuals (with [8], but see [3] for a more sophisticated treatment). The operation of updating a context $c \in \gamma$ with an individual a is written $a :: c$. Pronoun resolution algorithms *select* individuals from contexts, and are generically written $\mathtt{sel}$.[9] We will assume that the individual selected from the context is actually present in the context ($\mathtt{sel}(\gamma) \in \gamma$), leaving aside the question of empty contexts, and more precise conditions on the identity of the selected individual.

To deal with dynamic phenomena in his system, [8] lifts the type of a sentence once again to be a function from contexts (of type γ) and discourse continuations (functions of type γo) to propositions; an expression of type $\gamma(\gamma o)o$, which we will abbreviate as t. The idea is that a sentence is interpreted as a function from both its left context c and its right context ϕ to propositions.

With the exception of verb denotations, and expressions (such as relative pronouns) which are analysed there as taking verb denotations as arguments, we are able to simply take over the denotations assigned to lexical items from [8].

The style of analysis popular in the minimalist syntactic literature (and followed here), makes less straightforward a homomorphic relation between syntactic and semantic types. For example, the subject of a sentence is typically selected by a 'functional head', i.e. a lexical item other than the verb.

Relations We first look at the denotations of n-ary relation denoting lexical items, such as nouns and verbs. While nouns are treated here just as in [8], we treat verbs as on par with nouns, not, as does [8], as functions which take generalized quantifier denotations as arguments. This is because, in the minimalist grammar system, scope is dealt with by movement, not by modifying the verbal denotation (as is standard in categorial approaches to scope [18]).

[9] [9] gives pronoun resolution algorithms an additional property argument, and proposes that they select an individual with that property from the context.

A common noun, such as *monkey*, which is interpreted in the model as a function **monkey** of type eo mapping entities to true just in case they are monkeys, denotes a function of type et:

$$[\![monkey]\!](x)(c)(\phi) := \mathbf{monkey}(x) \wedge \phi(c)$$

Similarly, a transitive verb, such as *eat*, interpreted as a function **eat** : eeo mapping pairs of entities to true just in case the second ate the first, denotes a function of type eet:

$$[\![eat]\!](x)(y)(c)(\phi) := \mathbf{eat}(x)(y) \wedge \phi(c)$$

In general, given a function $f : \underbrace{e \cdots e}_{n \text{ times}} o$, we lift it to $\textsc{lift}(f) : \underbrace{e \cdots e}_{n \text{ times}} t$:

$$\textsc{lift}(f)(x_1) \cdots (x_n)(c)(\phi) := f(x_1) \cdots (x_n) \wedge \phi(c)$$

The conjunction and conjunct $\phi(c)$ common to all such predicates cashes out the empirical observation that propositions in a discourse combine conjunctively [30].

We adopt the following convention regarding arguments of type o (propositions): they are lifted to arguments of type t, and are passed as arguments the left and right context parameters of the lifted lexical item. As an example, take **believe** to be interpreted as a function of type oeo; a relation between a proposition (the belief) and an individual (the believer).

$$\textsc{lift}(\mathbf{believe})(S)(x)(c)(\phi) := \mathbf{believe}(S(c)(\phi))(x) \wedge \phi(c)$$

This illustrates a difficulty with the dynamic aspects of the system; it is not obvious how to allow a context to pick up referents contained in a different branch (here the propositional argument to **believe**) than which the continuation is (here in a position 'c-commanding' this argument). Should the propositional argument to **believe** contain a proper name, for example, this should be able to be found in the context of the remainder of the sentence. We do not dwell further on this difficulty here (though see footnote 11).

Noun phrases Traditional noun phrases (which are here, in line with [1], called 'determiner phrases', or DPs) such as *Mary*, *he*, or *every monkey*, denote functions $g : (et)t$. Proper names are interpreted as generalized quantifiers of type $(eo)o$, which are then lifted to the higher type via the operation GQ:

$$\textsc{gq}(G)(P)(c)(\phi) = G(\lambda x_e.\ P(x)(c)(\lambda d.\phi(x::d)))$$

Note that the individual 'referred to' by the generalized quantifier is incorporated into the context $(x::c)$ of the continuation of the sentence (ϕ). This permits future pronouns to pick up this individual as a possible referent. As an example, $\mathbf{Mary} = \lambda P_{eo}.P(m)$. And so $\textsc{gq}(\mathbf{Mary}) = \lambda c, \phi.P(m)(c)(\lambda d.\phi(m::d))$.

As mentioned above, we interpret pronouns, not as variables, but as noun phrase denotations involving pronoun resolution algorithms: $\texttt{sel} : \gamma e$.

$$[\![pro]\!](P)(c)(\phi) := P(\texttt{sel}(c))(c)(\phi(c))$$

Determiners The system presented in [8] is limited to quantifiers of type $\langle 1 \rangle$.[10,11] Accordingly, we restrict ourselves to the standard universal and existential quantifiers $\forall, \exists : (eo)o$. Determiners *every* and *some* are of type $(et)(et)t$, and denote the following functions.

$$[\![every]\!](P)(Q)(c)(\phi) := \forall(\lambda x.\neg P(x)(c)(\lambda d.\neg Q(x)(x :: d)(\lambda d.\top)) \wedge \phi(c))$$
$$[\![some]\!](P)(Q)(c)(\phi) := \exists(\lambda x.P(x)(c)(\lambda d.Q(x)(x :: d)(\phi)))$$

As explained by De Groote, the negations inside of the lambda term representing the denotation of *every* make the conjunctive meanings of the properties P and Q equivalent to the desired implication. As an example, take $P = [\![man]\!] = \lambda x, c, \phi.\mathbf{man}(x) \wedge \phi(c)$ and take $Q = [\![smile]\!] = \lambda x, c, \phi.\mathbf{smile}(x) \wedge \phi(c)$. Then $[\![every]\!]([\![man]\!])([\![smile]\!])(c)(\phi)$ β-reduces to the following lambda term, taking $A \rightarrow B$ as an abbreviation for $\neg(A \wedge \neg B)$.

$$\forall(\lambda x.\mathbf{man}(x) \rightarrow \mathbf{smile}(x) \wedge \top) \wedge \phi(c)$$

Finally, $\top$ stands for the always true proposition. As it always occurs as part of a conjunction, we simply and systematically replace $A \wedge \top$ everywhere with the equivalent A.

Everything Else All other lexical items are interpreted as the identity function of the appropriate type. While some (such as auxiliaries) should be assigned a more sophistiated denotation in a more sophisticated fragment, others (such as most of the 'functional' lexical items) play no obvious semantic role, and are there purely to express syntactic generalizations.

4 A Fragment

There are two main constructions addressed in this section. First (in §4.1), we show the necessity of de Groote's discourse contexts (or something like them) for the variable naming scheme adopted here. Then (in §4.2), as advertised in the abstract, we illustrate the 'tensed-clause boundedness' of quantifier raising.

We draw lexical items mostly unchanged from [23] (see figure 1). The fragment derives the following sentences, with the indicated scope relations.

4. Some man believed that every woman smiled. $(\exists > \forall)$
5. Some man believed every woman to have smiled. $(\exists > \forall,\ \forall > \exists)$

Although the number of lexical items (especially in the verbal domain) may look at first blush imposing, there are two things to bear in mind. First, most of them are (intended to be) 'closed class' items, meaning that they needn't ever

[10] A quantifier of type $\langle n_1, \ldots, n_k \rangle$ takes k predicates of arities $n_1, \ldots, n_k$ respectively, and returns a truth value.

[11] The extension to quantifiers of type $\langle n \rangle$ for any n is straightforward, but how it should be extended to handle binary quantifiers (of type $\langle 1, 1 \rangle$) is non-obvious.

NAME	FEATURES	PRONUNCIATION	MEANING
that	=s t	"that"	id
-ed	=p +k +q s	"-ed"	id
to	=p t	"to"	id
have	=en p	"have"	id
-en	=v en	"-en"	id
Asp	=v p	"ϵ"	id
Qv	=v +q v	"ϵ"	id
v	=V =d v	"ϵ"	id
AgrO	=V +k V	"ϵ"	id
smile	=d v	"smile"	LIFT(**smile**)
praise	=d V	"praise"	LIFT(**praise**)
believe	=t V	"believe"	LIFT(**believe**)

(a) verbal elements

NAME	FEATURES	PRONUNCIATION	MEANING
dD	=D d -k $\ominus$q	"ϵ"	id
dQ	=D d -k $\ominus\hat{\text{q}}$	"ϵ"	id
every	=n D	"every"	$[\![every]\!]$
some	=n D	"some"	$[\![some]\!]$
man	n	"man"	LIFT(**man**)
woman	n	"woman"	LIFT(**woman**)
pro	D	"he"	$[\![pro]\!]$
j	D	"John"	GQ(**John**)
b	D	"Bill"	GQ(**Bill**)

(b) nominal elements

Fig. 1: Lexical items

$$
\begin{aligned}
R(a)(b) &= \mathbf{merge}(a,b) & \qquad \mathtt{iv}(s)(v) &= R(v)(s)\\
V(a) &= \mathbf{move}(a) & \qquad \mathtt{to}(v) &= R(\mathsf{to})(R(\mathsf{have})(R(\text{-}\mathsf{en})(v)))\\
ObjK(v) &= V(R(\mathsf{AgrO})(v)) & \qquad \mathtt{pst}(v) &= V(V(R(\text{-}\mathsf{ed})(R(\mathsf{Asp})(v))))\\
ObjQ(v) &= V(R(\mathsf{Qv})(v)) & \qquad \mathtt{c} &= R(\mathsf{that})\\
Sub(s)(v) &= R(R(\mathsf{v})(v))(s) & \qquad \mathtt{d} &= R(\mathsf{dD})\\
\mathtt{tv}(s)(v)(o) &= ObjQ(Sub(s)(ObjK(R(v)(o)))) & \qquad \mathtt{q} &= R(\mathsf{dQ})
\end{aligned}
$$

Fig. 2: Abbreviations

be added to as more novel words are encountered. Second, these closed class items in fact participate in a very regular way in derivations. We introduce some abbreviations (figure 2), so as to be able to concisely describe derivations of sentences.[12]

The sentence in 6 has the two (semantically equivalent) derivations represented in 7, and interpretation as in 8.

6. Some man smiled.
7. $\mathtt{pst}(\mathtt{iv}(f(\mathsf{some})(\mathsf{man}))(\mathsf{smile}))$, for $f \in \{D, Q\}$
8. $\lambda c, \phi.\ \exists(\lambda x.\ \mathbf{man}(x) \wedge \mathbf{smile}(x) \wedge \phi(c))$

The transitive sentence in 9 has the derivation in 10, which corresponds to the object wide scope reading in 11, and the derivation in 12 which corresponds to the subject wide scope reading in 13.

9. Some woman praised every man.
10. $\mathtt{pst}(\mathtt{tv}(\mathtt{d}(\mathbf{some})(\mathbf{woman}))(\mathbf{praise})(\mathtt{q}(\mathbf{every})(\mathbf{man})))$
11. $\lambda c, \phi.\forall(\lambda x.\mathbf{man}(x) \rightarrow \exists(\lambda y.\mathbf{woman}(y) \wedge \mathbf{praise}(x)(y))) \wedge \phi(c)$
12. $\mathtt{pst}(\mathtt{tv}(\mathtt{q}(\mathbf{some})(\mathbf{woman}))(\mathbf{praise})(\mathtt{q}(\mathbf{every})(\mathbf{man})))$
13. $\lambda c, \phi.\exists(\lambda y.\mathbf{woman}(y) \wedge \forall(\lambda x.\mathbf{man}(x) \rightarrow \mathbf{praise}(x)(y)) \wedge \phi(y :: c))$

[12] The 'abbreviations' in figure 2 are λ-terms over derivation trees. The naming of bound variables is purely mnemonic, as they are all of atomic type.

4.1 Pronouns are not variables

The idea that pronouns are to be rendered as variables in some logical language is widespread in the semantic literature (though see [11, 20] for some alternatives). However in the present system, where possible binders (DPs) bind variables according to the reason for their movement (of which there are only finitely many), sentences like the below prove an insurmountable challenge to this naïve idea.

14. Every boy believed that every man believed that he smiled.

In sentence 14, the pronoun can be bound either by *every boy* or by *every man*. However, if it 'denotes' a variable, it must be one of x_k or x_q, and both of these are bound by the closer DP, *every man*, leaving no possibility for the reading in which every boy smiles.[13] Here we see that treating pronouns as (involving) pronoun resolution algorithms provides a simple way to approach a resolution to this problem.[14]

Sentence 14 has its equivalent derivations in 15, with meaning representation in 16.

15. $\mathtt{pst}(\mathtt{iv}\ (f(\mathsf{every})(\mathsf{boy}))$
$(R(\mathsf{believe})(\mathtt{c}(\mathtt{pst}(\mathtt{iv}\ (g(\mathsf{every})(\mathsf{man}))$
$(R(\mathsf{believe})(\mathtt{pst}(\mathtt{iv}(h(\mathsf{pro})(\mathsf{smile}))))))))))$
for $f, g, h \in \{\mathtt{d}, \mathtt{q}\}$
16. $\lambda c, \phi.\forall(\lambda x.\ \mathbf{boy}(x) \rightarrow$
$\mathbf{believe}(\forall(\lambda y.\ \mathbf{man}(y) \rightarrow$
$\mathbf{believe}(\mathbf{smile}(\mathtt{sel}(y :: x :: c)))(y))(x)) \wedge \phi(c)$

Crucially, in every context, the input to the pronoun resolution algorithm includes the (bound) variables x and y, the choice of which would result in the bound reading of the pronoun for *every boy* and *every man* respectively.

4.2 The tensed-clause boundedness of QR

Now we present derivations for sentences 4 and 5 (repeated below as 24 and 17). We begin with 17, which has two surface scope readings which correspond to *de re* (19) and *de dicto* (21) beliefs, and an inverse scope reading (in 23).

17. Some man believed every woman to have smiled.
18. $\mathtt{pst}(\mathtt{tv}(f(\mathsf{some})(\mathsf{man}))(\mathsf{believe})(\mathtt{to}(\mathtt{iv}(\mathtt{d}(\mathsf{every})(\mathsf{woman}))(\mathsf{smile}))))$,
where $f \in \{\mathtt{d}, \mathtt{q}\}$
19. $\lambda c, \phi.\exists(\lambda x.\mathbf{man} \wedge \mathbf{believe}(\forall(\lambda y.\mathbf{woman}(y) \rightarrow \mathbf{smile}(y)))(x) \wedge \phi(x :: c))$
20. $\mathtt{pst}(\mathtt{tv}(\mathtt{q}(\mathsf{some})(\mathsf{man}))(\mathsf{believe})(\mathtt{to}(\mathtt{iv}(\mathtt{q}(\mathsf{every})(\mathsf{woman}))(\mathsf{smile}))))$

[13] In response to this problem, [23] abandons the idea of [32] that variables are named after movement dependencies, and with it the simply typed lambda calculus.

[14] It is not in itself an answer to this problem, as it introduces an unanalyzed 'unknown' in the form of a pronoun resolution algorithm.

21. $\lambda c, \phi.\exists(\lambda x.\mathbf{man} \wedge \forall(\lambda y.\mathbf{woman}(y) \rightarrow \mathbf{believe}(\mathbf{smile}(y))(x)) \wedge \phi(x :: c))$
22. $\mathtt{pst}(\mathtt{tv}(\mathtt{d}(\mathsf{some})(\mathsf{man}))(\mathsf{believe})(\mathtt{to}(\mathtt{iv}(\mathtt{q}(\mathsf{every})(\mathsf{woman}))(\mathsf{smile}))))$
23. $\lambda c, \phi.\forall(\lambda y.\mathbf{woman} \rightarrow \exists(\lambda x.\mathbf{man}(x) \wedge \mathbf{believe}(\mathbf{smile}(y))(x))) \wedge \phi(c)$

Now we turn to 24. Here the inverse scope reading is not available for the simple reason that a tensed clause (one with the lexical item -ed) forces a Q feature to be checked. Thus the embedded DP *every woman* does not enter into a movement relationship with anything after the matrix subject is present.

24. Some man believed that every woman smiled.
25. $\mathtt{pst}(\mathtt{iv}(f(\mathsf{some})(\mathsf{man}))(R(\mathsf{believe})(\mathtt{c}(\mathtt{pst}(\mathtt{iv}(g(\mathsf{every})(\mathsf{woman}))(\mathsf{smile}))))))$, for all $f, g \in \{\mathtt{q}, \mathtt{d}\}$
26. $\lambda c, \phi.\exists(\lambda x.\mathbf{man} \wedge \mathbf{believe}(\forall(\lambda y.\mathbf{woman}(y) \rightarrow \mathbf{smile}(y)))(x) \wedge \phi(x :: c))$

The tensed-clause boundedness of quantifier scope is a fragile and analysis dependent property; a simple 'splitting' of the -ed lexical item into one of the form $\langle -\mathsf{ed}, \mathtt{=p\ +k\ s} \rangle$ and another of the form $\langle \epsilon, \mathtt{=s\ +q\ s} \rangle$ suddenly makes the inverse scope reading in sentence 24 derivable. There are two things to say at this point. First, the 'actual' generalization about scope taking inherent in (this version of) the minimalist grammar framework is that an expression may take scope over only those others dominated by a node in the derivation tree with which it enters into a feature checking relationship. This happens to coincide in our fragment with tensed clauses. Second, the intuitive generalizations this fragment is making are (1) that scope can be checked at the VP and at the S level, and (2) that scope must be checked as soon as possible. If this latter condition is formulated as a transderivational constraint [13], and applied to derivations in the alternative fragment (in which the -ed lexeme is 'split' as per the above remarks), the present fragment can be viewed as the result of 'compiling out' the effect of the constraint on the alternative fragment.

5 Conclusion

We have shown how De Groote's simply typed account of dynamic phenomena can be used in a minimalist grammar. This allows us to maintain Stabler's ideas about variables in movement dependencies, as well as to use nothing but the simply typed lambda calculus to deliver the same meanings as the more standard 'LF' interpretative accounts of semantics in the chomskyian tradition (modulo pronouns). What is doing the work here is the rejection of the pronouns-as-variables view, in favor of a pronouns-as-functions-from-contexts view. This latter seems to be a novel perspective on the the pronouns-as-definite-description view [11] (especially in the light of [9]).

In order to preserve the functional relation between derivations and meanings, we have incorporated information into the feature system (whether or not a licensee feature has a hat diacritic). However, this strategy seems non-ideal, as it results in cases of spurious ambiguity (the worst offender in this paper was example 1, with six equivalent derivations).

Finally, we have noted that there are open questions regarding the best way of incorporating sentential complement embedding verbs and type $\langle 1, 1 \rangle$ quantifiers into the dynamism-via-continuation framework used here [8].

References

1. Abney, S.P.: The English Noun Phrase in its Sentential Aspect. Ph.D. thesis, Massachusetts Institute of Technology (1987)
2. Amblard, M.: Calculs de représentations sémantiques et syntaxe générative: les grammaires minimalistes catégorielles. Ph.D. thesis, Université Bordeaux I (2007)
3. Asher, N., Pogodalla, S.: A montagovian treatment of modal subordination. In: Li, N., Lutz, D. (eds.) Semantics and Linguistic Theory (SALT) 20. pp. 387–405. eLanguage (2011)
4. Barss, A.: Chains and Anaphoric Dependence: On Reconstruction and its Implications. Ph.D. thesis, Massachusetts Institute of Technology (1986)
5. Chomsky, N.: The Minimalist Program. MIT Press, Cambridge, Massachusetts (1995)
6. Cooper, R.: Quantification and Syntactic Theory. D. Reidel, Dordrecht (1983)
7. Cresti, D.: Extraction and reconstruction. Natural Language Semantics 3, 79–122 (1995)
8. de Groote, P.: Towards a montagovian account of dynamics. In: Gibson, M., Howell, J. (eds.) Proceedings of SALT 16. pp. 1–16 (2006)
9. De Groote, P., Lebedeva, E.: Presupposition accommodation as exception handling. In: Fernandez, R., Katagiri, Y., Komatani, K., Lemon, O., Nakano, M. (eds.) The 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue - SIGDIAL 2010. pp. 71–74. Association for Computational Linguistics, Tokyo, Japan (2010)
10. Engelfriet, J.: Context-free graph grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3: Beyond Words, chap. 3, pp. 125–213. Springer Verlag (1997)
11. Evans, G.: Pronouns. Linguistic Inquiry 11(2), 337–362 (1980)
12. Gärtner, H.M., Michaelis, J.: Some remarks on locality conditions and minimalist grammars. In: Sauerland, U., Gärtner, H.M. (eds.) Interfaces + Recursion = Language?, Studies in Generative Grammar, vol. 89, pp. 161–195. Mouton de Gruyter, Berlin (2007)
13. Graf, T.: A tree transducer model of reference-set computation. UCLA Working Papers in Linguistics 15, Article 4 (2010)
14. Graf, T.: Closure properties of minimalist derivation tree languages. In: Pogodalla, S., Prost, J.P. (eds.) LACL 2011. Lecture Notes in Artificial Intelligence, vol. 6736, pp. 96–111 (2011)
15. Hale, J.T., Stabler, E.P.: Strict deterministic aspects of minimalist grammars. In: Blache, P., Stabler, E.P., Busquets, J., Moot, R. (eds.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 3492, chap. 11, pp. 162–176. Springer (2005)
16. Harkema, H.: Parsing Minimalist Languages. Ph.D. thesis, University of California, Los Angeles (2001)
17. Heim, I., Kratzer, A.: Semantics in Generative Grammar. Blackwell Publishers (1998)

18. Hendriks, H.: Studied Flexibility: Categories and types in syntax and semantics. Ph.D. thesis, Universitaet van Amsterdam (1993)
19. Hornstein, N.: Movement and chains. Syntax 1(2), 99–127 (1998)
20. Jacobson, P.: Towards a variable-free semantics. Linguistics and Philosophy 22(2), 117–184 (1999)
21. Johnson, K.: How far will quantifiers go? In: Martin, R., Michaels, D., Uriagereka, J. (eds.) Step by Step: Essays on Minimalist Syntax in Honor of Howard Lasnik, chap. 5, pp. 187–210. MIT Press, Cambridge, Massachusetts (2000)
22. Kanazawa, M., Michaelis, J., Salvati, S., Yoshinaka, R.: Well-nestedness properly subsumes strict derivational minimalism. In: Pogodalla, S., Prost, J.P. (eds.) Logical Aspects of Computational Linguistics, LACL 2011. Lecture Notes In Computer Science, vol. 6736, pp. 112–128. Springer, Berlin (2011)
23. Kobele, G.M.: Generating Copies: An investigation into structural identity in language and grammar. Ph.D. thesis, University of California, Los Angeles (2006)
24. Kobele, G.M.: Inverse linking via function composition. Natural Language Semantics 18(2), 183–196 (2010)
25. Kobele, G.M.: Minimalist tree languages are closed under intersection with recognizable tree languages. In: Pogodalla, S., Prost, J.P. (eds.) LACL 2011. Lecture Notes in Artificial Intelligence, vol. 6736, pp. 129–144 (2011)
26. Kobele, G.M., Retoré, C., Salvati, S.: An automata theoretic approach to minimalism. In: Rogers, J., Kepser, S. (eds.) Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07. Dublin (2007)
27. LeComte, A.: Semantics in minimalist-categorial grammars. In: de Groote, P. (ed.) FG 2008. pp. 41–59. CSLI Press (2008)
28. Michaelis, J.: On Formal Properties of Minimalist Grammars. Ph.D. thesis, Universität Potsdam (2001)
29. Parigot, M.: $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In: Voronkov, A. (ed.) Logic Programming and Automated Reasoning. Lecture Notes in Computer Science, vol. 624, pp. 190–201. Springer-Verlag, Berlin Heidelberg (1992)
30. Pietrowski, P.M.: Events and Semantic Architecture. Oxford University Press (2005)
31. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science 88, 191–229 (1991)
32. Stabler, E.P.: Computing quantifier scope. In: Szabolcsi, A. (ed.) Ways of Scope Taking, chap. 5, pp. 155–182. Kluwer, Boston (1997)
33. Stabler, E.P.: Derivational minimalism. In: Retoré, C. (ed.) Logical Aspects of Computational Linguistics, Lecture Notes in Computer Science, vol. 1328, pp. 68–95. Springer-Verlag, Berlin (1997)

LF-Copying without LF

Gregory M. Kobele

Linguistics Department & Computation Institute
University of Chicago

Abstract

A copying approach to ellipsis is presented, whereby the locus of copying is not a level of derived syntactic structure (LF), but rather the derivation itself. The ban on preposition stranding in sprouting follows without further stipulation, and other, seemingly structure sensitive, empirical generalizations about elliptical constructions, including the preposition stranding generalization, follow naturally as well. Destructive operations which 'repair' non-identical antecedents are recast in terms of exact identity of derivations with parameters. In the context of a compositional semantic interpretation scheme, the derivational copying approach to ellipsis presented here is revealed to be a particular instance of a proform theory, thus showing that the distinctions between, and arguments about, syntactic and semantic theories of ellipsis need to be revisited.

Key words: Minimalist grammars, ellipsis, preposition stranding, sprouting

1. Introduction

As Merchant (2001) puts it, "nowhere does [the] sound-meaning correspondance break down so spectacularly as in ellipsis". In a discourse context as in 1, we interpret an elliptical sentence like 1a as meaning the same thing as 1b.

1. Someone praised Oskar.
   (a) I wonder who.
   (b) I wonder who praised him.

Elliptical sentences are dependent on the surrounding mostly (Hankamer and Sag, 1976) linguistic context for their interpretation; in the context of *Oskar criticized someone*, an utterance of 1a would mean something quite different. From the perspective of the listener, the central problem posed by ellipsis is that of inferring the intended meaning from the context, which is called ellipsis resolution. A primary task of ellipsis research is to characterize the nature of this inference, with a major current research area focussed on the question of what information about the context is relevant to this inference; does

Email address: kobele@uchicago.edu (Gregory M. Kobele)

the listener's inference procedure make reference to semantic properties of the context, to syntactic ones, to yet something else, or to combinations of these?

One difficulty posed by ellipsis is the fact that the folk classification of lingusitic constructions do not seem to provide a fine-grained enough description of conditions to answer this question. Sentence 2 (from Hardt (1993)) shows that a syntactic difference (mismatching voice features) between antecedent and ellipsis site does not block the listener's inference, whereas sentence 3, despite having a similar form, is much less acceptable.[1]

2. This information could have been released, but Gorbachev chose not to ~~release this information~~.

3. %Information was leaked, but Snowden didn't ~~leak information~~.

At the moment, the conditions under which various information becomes relevant to the listener's inference have yet to coalesce into a clear picture, although information and discourse structure seem to play an important role (Rooth, 1992; Kehler, 2002; Kertz, 2010).

In light of this, one strategy is to focus on first explaining general tendencies in the data. For example, whereas in verb phrase ellipsis (VPE; sentences 2 and 3) voice mismatches are at least sometimes quite acceptable, the same does not appear to be true in the case of sluicing (Merchant, 2013) irrespective of information or discourse structural properties (SanPietro et al., 2012).

4. %Someone murdered Joe, but we don't know by whom ~~he was murdered~~.

Following this strategy, the problem of accounting for sentences like 2, 3, and 4 can be divided into the two problems of (i) accounting for the fact that voice mismatches are *never* good in sluicing, but are *sometimes* good in VPE, and (ii) accounting for the varying acceptability of voice mismatches in VPE. This strategy will be adopted in this paper, where a unified account of problems like (i) will be provided; Kim et al. (2011) show how an analysis of problem (i) of the sort to be presented in this paper can serve as the foundation for an analysis of problem (ii).

The following generalizations provide some of the most influential arguments for the proposition that elipsis resolution is sensitive to purely syntactic properties of antecedents.[2]

[1] In this paper, the diacritic % will be used to indicate unacceptability, and * to indicate ungrammaticality. To say that a sentence is unacceptable is to make an empirical claim (about either formal or informal experiments). To say that a sentence is ungrammatical is to claim that it has a particular theoretical property, that of not being generated by the grammar. The link between the formal notion of ungrammaticality and the empirical one of unacceptability is still unclear (see Clark et al. (2013) for recent advances), but a rough idea is that ungrammatical sentences should be, *ceteris paribus*, less acceptable than grammatical ones.

[2] While *case-matching* (Ross, 1969) is often taken as another argument for the role of syntactic identity in ellipsis, it has become evident that the case matching effects can be modeled without any reference to syntactic identity (Jäger, 2005; Barker, 2013). These works, from the categorial grammar tradition, enrich the syntactic types of expressions to include information about case. The approach adopted here can be thought of as an extension (across grammar frameworks) of the same basic idea. This is a viable strategy whenever there are only finitely many distinct cases; the way to challenge this, therefore, is to investigate sluicing in languages with *Suffixaufnahme* (Plank, 1995), where the number of cases has (potentially) no upper bound.

I. the differential acceptability of voice mismatches across ellipsis types (Merchant, 2013)

II. the preposition stranding generalization (Merchant, 2001)

III. the ban on preposition stranding in sprouting (Chung et al., 1995)

These generalizations are reviewed in the subsections to follow.

1.1. Voice (mis)matches

Given that some voice mismatches in VPE are very acceptable, it is natural to treat them uniformly as derivable syntactically (i.e. grammatical). Similarly, as (to date) no voice mismatches in sluicing are acceptable, it is natural to treat them uniformly as syntactically underivable (i.e. ungrammatical). Merchant (2013) (see also Tanaka (2011)) observes that the grammaticality difference in voice mismatches across ellipsis types can be accounted for if one adopts a phrasal approach to the passive (Bach, 1980; Keenan, 1980), and formulates the listener's inference in terms of simply identifying the ellipsis site with a syntactic antecedent. The availability of voice mismatches in VPE then follows from the fact that the antecedent is unspecified for voice (it is beneath the position at which voice is determined), whereas the unavailability of such in sluicing (analyzed as TP ellipsis) from the fact that the antecedent is specified for voice.

An overlooked prediction of this sort of account is that voice mismatches should be a sort of 'root phenomenon'; it is not that VPE should allow voice mismatches across the board, but rather that VPE should only allow voice mismatches *in the clause in which ellipsis takes place*. In particular, voice mismatches should be impossible if embedded within an elided structure (to use a PF-deletion metaphor). Sentence 5 tests this prediction.

5. %This information seems to have been released, but Gorbachev doesn't ~~seem to have released this information~~.

The unacceptability of this sentence is unexpected under a theory which allows voice mismatches in VPE across the board (as for example does that of Hardt (1993)),[3] but is exactly what one would expect under a unified timing-based theory of ellipsis (*à la* Merchant (2013)).

1.2. The preposition stranding generalization

Merchant (2001) observes that not all languages allow for preposition stranding in sluicing, and that moreover there is a strong correlation between whether a language allows preposition stranding at all, and whether it allows preposition stranding in sluicing. He advances the following generalization, based on a preliminary sample of 24 languages from three different families:

Merchant's generalization:

> *A language L will allow preposition stranding under sluicing iff L allows preposition stranding under regular* wh*-movement.*

[3] Kobele (2012b) provides additional examples with mismatch along the dimension of raising.

For illustration, consider the following English sentences.

6. John stood under something, but I don't know under what ~~he stood~~.

7. John stood under something, but I don't know what ~~he stood under~~.

According to Merchant's generalization, given that 7 is grammatical, we should (correctly) conclude that English allows preposition stranding under regular *wh*-movement.

Conversely, if the sentences above were from a language about which we know only that it allows for preposition stranding under regular *wh*-movement, on the basis of Merchant's generalization we should predict 7 to be grammatical.

1.3. The ban on preposition stranding in sprouting

The phenomenon of sprouting encompasses sentences like the below:

8. John ate, but I don't know what.

9. John ate pancakes, but I don't know why.

10. Pancakes were eaten, but I don't know by whom.

What unifies the sprouting sentences 8–10 is that the relation between antecedent and (syntactically fleshed out) ellipsis site is one of suppressed and realized optional argument. In the LF-copying theory of Chung et al. (1995), according to which a derived syntactic antecedent is selected, and then manipulated via a set of transformations before being inserted in the ellipsis site, *sprouting* is the name of the transformation which inserts a trace (which can be then bound by the *wh*-phrase) in an appropriate place in the antecedent structure.

Chung et al. (1995) observe that, in constrast to the usual case in (English) sluicing, in sprouting contexts preposition stranding is prohibited. This is illustrated by the sentences below.

11. John stood near something, but I don't know near what.

12. John stood near something, but I don't know what.

13. John stood, but I don't know near what.

14. *John stood, but I don't know what.

1.4. Plan of the paper

One of the interesting aspects of decompositional analyses in syntax is that they make available the possibility that the appropriate notion of inference can be characterized as exact identity (of a very abstract part), without the need for operations which alter structure; 'inference' reduces to simple copying. This paper is a working out of how this might look in a range of cases. The basic idea is that the copied structure is the derivation itself. This forces one to the perspective that the shape of the copy is not a tree, as is commonly assumed, but rather a context (a tree with holes). The account is presented in terms of the formal framework of minimalist grammars (MGs; (Stabler, 1997)), which is a well-understood and extensible grammar formalism (Stabler, 2011) capable of directly

implementing minimalist-style analyses. After the basic minimalist grammar system is presented in §2, it is extended in §3 to allow for 'LF-copying' (in quotes because there is no LF involved). This section presents the details of the ellipsis mechanism in the context of a running example of VPE. Section 4 presents a fragment of English, and demonstrates how this simple set-up allows for an account of the varying acceptability of voice mismatches, the ban on preposition stranding in sprouting, and the preposition stranding generalization of Merchant (2001). Finally §5 discusses phenomena, such as antecedent containment and island effects, which did not make it into the paper, reflects upon the debate about whether there is syntactic structure in ellipsis sites in light of the derivational copying theory presented herein, and concludes.

2. Minimalist Grammars

Minimalist grammars are a *mildly context-sensitive* grammar formalism (Harkema, 2001; Michaelis, 2001), inspired by Chomsky (1995).[4] MGs provide a framework in which minimalist-style analyses and proposed mechanisms can be directly implemented, and theoretical proposals formally evaluated. Since their introduction in Stabler (1997), many variants of the 'barebones' MG formalism have been proposed and investigated (an overview is in Stabler (2011)). What follows is a fairly canonical version, without regard to many potential points of linguistic controversy. The results in this paper are largely independent of this particular version.

A minimalist grammar has a fixed set of structure building operations, which are taken here to be just binary **merge** and unary **move**, whose application to expressions is dependent on the syntactic categories of these expressions.[5] The language of a particular minimalist grammar consists of those expressions which can be built up from lexical items by finitely many applications of the operations **merge** and **move**. The formalism is introduced by means of an example.

In order to describe sentences with ellipsis, one must be able to describe those without ellipsis, at the very least so as to be able to provide a discourse context for ellipsis resolution. Consider the simple intransitive sentence, *Carl will run*, analyzed in the familiar way sketched in figure 1. This structure records that the sentence has a derivation which proceeds as follows: (i) merge the lexical item for *run* together with the one for *Carl*, (ii) then merge the result together with the lexical item for *will* (iii) finally move *Carl* to the specifier position of *will*. A number of questions arise, among which are included: 1. why can't *Carl* and *will* be merged first? 2. why does *Carl* move and not

[4] Grammar formalisms belonging to this class (such as tree adjoining grammars, combinatory categorial grammars, and multiple context-free grammars) are unable to describe an infinite number of recursively enumerable languages, and are thus restrictive in the sense of ruling out *a priori* a large number of computationally possible languages as linguistically *im*possible. The languages which can be described are all simple in a formally precise sense (Joshi, 1985), which makes it possible to, among other things, build correct and efficient parsing algorithms for these grammar formalisms.

[5] A currently influential idea is that the structure building operations should be reduced to a single one, with **move** being a special case of **merge**, which itself is the operation of set formation. A particularly simple way of doing this in the present system takes $\mathbf{merge}(A, B) = \{A, B\}$, and $\mathbf{move}(A) = \mathbf{merge}(A, A) = \{A\}$. These two operations are kept separate in this paper because nothing said here depends on them being unified in any particular way. Readers whom this discomforts may read **external-merge** and **internal-merge** for **merge** and **move**, respectively.

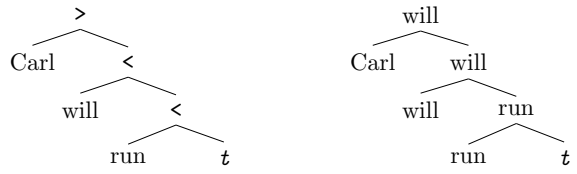


Figure 1: The basic structure of an intransitive sentence. Left: a label-free representation. Internal nodes record only which daughter they are a projection of by 'pointing' in the direction of their heads; '>' points to the right, and '<' to the left. Right: the bare-phrase structure equivalent.

$$\langle \mathsf{carl}, \mathtt{d\ k} \rangle \quad \langle \mathsf{run}, \mathtt{*d\ v} \rangle$$
$$\langle \mathsf{will}, \mathtt{*v\ *k\ t} \rangle$$

Figure 2: Lexical items for figure 1

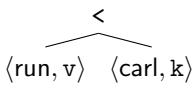
run? The common answer to the first question is that there is some sort of selection at work here; *Carl* has a certain property (being a DP), and *will* just is not looking for something with that property. The common answer to the second is similar: *Carl* has some property (needing case), and *will* is looking for something with that property. This sort of information will be represented here in terms of *features*; a feature `x` indicates that an expression has a certain property, and a feature `*x` indicates that an expression is looking for another with that property. The features had by a lexical item are orgainized into a feature bundle, which is simply a list of features. Lexical items are written using the notation $\langle \mathsf{w}, \delta \rangle$, where w is a lexeme, and δ is a feature bundle. Lexical items will, when context permits, be referred to by their lexeme (and so 'w' may be used to refer to the lexical item $\langle \mathsf{w}, \delta \rangle$). In the lexicon below, the lexical item carl has the feature bundle `d k`, which can be understood intuitively as saying that it is a DP (`d`) which needs case (`k`).[6] The lexical item run has the feature bundle `*d v`, which indicates that it must select a DP (`*d`) and will then be a vP (`v`). Finally, the feature bundle of the lexical item will indicates that it must select a vP (`*v`), assign case to something (`*k`), and will then be a TP (`t`). It is important to distinguish between a *grammar formalism*, which defines a space of possible analyses, and a *grammatical analysis* of some phenomenon, written in that formalism. In lexicalized grammar formalisms, such as minimalist grammars, an analysis is given by presenting a lexicon. Throughout this paper, lexical items (which constitute particular analyses) are put in boxes as in figure 2.

The expressions *generated by* a lexicon are those which can be built up from lexical

[6] An influential idea in modern minimalist syntax is that at least some of the information represented here in terms of lists of syntactic features might in fact be derivable from something more basic, in particular morphological feature matrices. For example, that a DP should move to check its syntactic case feature, here encoded by a `k` feature, might ultimately be derivable from the fact that in its morphological feature matrix the CASE attribute is unvalued. This is an interesting reductionist idea, but is largely orthogonal to the concerns of this paper. In particular, as the fragment to be described in this paper abstracts away from inflectional distinctions, morphological feature matrices of lexical entries are suppressed entirely. Someone who favours this view may understand these syntactic feature bundles as emergent properties of the morphological feature matrices associated with individual lexical items.

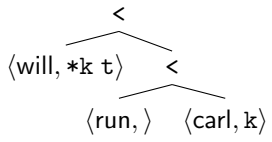
items by a finite number of applications of **merge** and **move**. Given the lexicon above, (all and only) the following additional expressions can be generated:

i. **merge**($\langle$run, *d v$\rangle$, $\langle$carl, d k$\rangle$):



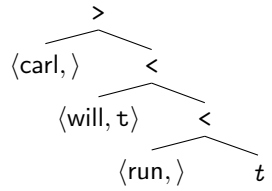
Because run's feature bundle begins with a *d, and carl's begins with a matching d, **merge** can put them together as per the tree above. Note that both *d and d features are eliminated in the resulting expression, the head of which can be found by walking down from the root always toward the direction pointed at by the node label; as here the root is labeled with <, which 'points' left, the head is in the left daughter subtree, which in this case is simply the leaf run.

ii. **merge**($\langle$will, *v *k t$\rangle$, i):



Merge applies to the indicated expressions because the one has a feature *v, and the head of the other has the feature v. Again, both features are eliminated in the result, the head of which is will. Note also that all of the features of run have been checked; the leaf run and the subtree headed by it are now syntactically inert.

iii. **move**(ii):



Finally, **move** attracts carl to a projection of will, driven by their respective k and *k features, which are then eliminated. This expression is a projection of will, as can be seen by following from the root the path indicated by the arrowheads,[7] which has just a single feature t representing the fact that it is a TP (and can be selected as such). All features of all other lexical items used to build this expression have been checked.

[7]The root (labeled by >) points to the right, the right-hand daughter of the root (labeled by <) points to the left, the left-hand daughter of which is will.

Categories, which are here called feature bundles, are complex, as in categorial grammar, and are structured as lists of atomic features, themselves with various diacritics (x, *y, ...). The currently accessible feature is the feature at the beginning (leftmost) position of the list, which allows for some features being available for checking only after others have been checked. Although there are many different notations, the basic idea is familiar (Adger, 2003; Müller, 2010). The present system differs formally from these primarily in that both features (x and *x) triggering an operation are checked. Adger and Müller also make a distinction among the features *x according to whether they (in the case of move/internal merge) trigger overt displacement. To keep novelty to a minimum, this distinction will be used here as well; *x will continue to be used for overt movement, and ⊛x will be used for covert movement.[8] In the case of merge/external merge, ⊛x will be used for merger with head movement. The operations **merge** and **move** will be discussed in more detail in §2.1. The features *x and ⊛x will be called the *attractor* and x the *attractee* variants of the feature type x. This is *not* the same as the interpretable/uninterpretable distinction, which serves both to allow asymmetric feature checking (interpretable features are not necessarily checked), and to describe which derivations are well-formed at the interfaces (those without uninterpretable features). As set out here, all features are uninterpretable from the checking perspective (what Stabler (2011) calls *persistent* features may be thought of as interpretable ones in this sense). From the perspective of interface well-formedness, all features must be checked except for a single attractee feature of the head of the expression, which is to be understood as the category of the expression.

As discussed in the next section, the internal tree geometric structure of expressions is not relevant in determining whether **merge** or **move** can apply. Instead, all that matters are (i) the features of the head, and (ii) the features (if any) of the other heads in the tree. This information provides all the information which is relevant to determining whether the structure building operations can apply. Because the more familiar term 'category' is typically associated with only the properties of the head of an endocentric expression, this information will instead be called the *type* of an expression,[9] and is written $\mathsf{type}(t) = \langle \alpha, A \rangle$, where α is the feature bundle of the head of t, and A contains the feature bundles of the other heads in t. Note that for any lexical item $\langle \mathsf{w}, \delta \rangle$, $\mathsf{type}(\langle \mathsf{w}, \delta \rangle) = \langle \delta, \emptyset \rangle$. In the expressions derived above, $\mathsf{type}(i) = \langle \mathtt{v}, \{\mathtt{k}\} \rangle$, $\mathsf{type}(ii) = \langle \mathtt{*k\ t}, \{\mathtt{k}\} \rangle$, and $\mathsf{type}(iii) = \langle \mathtt{t}, \emptyset \rangle$.

2.1. Operations

Derived expressions are binary branching trees, whose internal nodes are labeled with < or > and where leaves are labeled with lexeme/feature bundle pairs (and so a lexical item $\langle \mathsf{w}, \delta \rangle$ is a special case of a tree with only a single node).[10] Each node in an expression is a projection of the unique leaf one arrives at by following the arrows down the tree. A

[8] Equivalently, one might think of ⊛x as the basic selection feature, and *x as a ⊛x feature with an EPP diacritic.

[9] When it is important to distinguish this notion from the semantic one, it will be called the *syntactic* type of an expression.

[10] The symbol 't', representing a trace, needn't be taken as a primitive; it can be defined as the pair $\langle \epsilon, \rangle$, where the first component, ϵ, is the empty word (something without any phonetic material), and the second component is the empty feature bundle.

node is a *maximal projection* of a leaf ℓ just in case it is a projection of ℓ and its parent (if it has one) is a projection of a different leaf. In terms of paths, the maximal projection of a leaf is obtained by walking up from the leaf until the arrow at a node no longer points down to it. If t is a bare phrase structure tree with head H, then we write t[H] to indicate this. (This means that the lexical item $\langle w, \delta\rangle$ can be written as $\langle w, \delta\rangle[\langle w, \delta\rangle]$.) The notation t[H′] indicates the tree like t[H] but with the head H replaced by H′.

2.1.1. Merge

The **merge** operation is defined on a pair of trees t_1, t_2 if and only if the head of t_1 has a feature bundle beginning with $*\mathtt{x}$ or $\circledast\mathtt{x}$, and the head of t_2 has a feature bundle beginning with the matching $\mathtt{x}$ feature. The bare phrase structure tree which results from the merger of t_1 and t_2 always has t_1 projecting over t_2, in other words, the head of the result is always the head of the expression with the attractor feature. In case t_1 is a lexical item, t_2 is linearized to its right (a complement), and otherwise t_2 is to its left (a specifier). In either case, both selection features are checked in the result.

$$\mathbf{merge}(t_1[\langle \alpha, *\mathtt{x}\delta\rangle], t_2[\langle \beta, \mathtt{x}\gamma\rangle]) = \quad \begin{array}{c} < \\ t_1[\langle \alpha, \delta\rangle] \quad t_2[\langle \beta, \gamma\rangle] \end{array} \qquad (t_1 \text{ is a lexical item})$$

$$\mathbf{merge}(t_1[\langle \alpha, *\mathtt{x}\delta\rangle], t_2[\langle \beta, \mathtt{x}\gamma\rangle]) = \quad \begin{array}{c} > \\ t_2[\langle \beta, \gamma\rangle] \quad t_1[\langle \alpha, \delta\rangle] \end{array} \qquad (t_1 \text{ is not a lexical item})$$

The $\circledast\mathtt{x}$ feature triggers head movement, which, following tradition (Baker, 1988), is permitted only from a complement (i.e. first merged) position. Head movement is here analyzed as a phonological reflex of a particular kind of merger (Stabler, 1997), and not the result of a special movement step. More sophisticated treatments of head movement-like phenomena, such as in mirror theory (Brody, 2000), are straightforwardly implementable (Kobele, 2002). What is important is that head movement divorces the surface position of a head from its maximal projection, without recourse to movement/internal merge. A hyphen preceding/following a lexeme indicates whether it is a suffix/prefix.

$$\mathbf{merge}(\langle \text{-}\alpha, \circledast\mathtt{x}\delta\rangle, t_2[\langle \beta, \mathtt{x}\gamma\rangle]) = \quad \begin{array}{c} < \\ \langle \beta\text{-}\alpha, \delta\rangle \quad t_2[\langle \epsilon, \gamma\rangle] \end{array}$$

2.1.2. Move

The operation **move** applies to a single tree $t[\langle \alpha, \bullet y\delta\rangle]$ (where $\bullet y$ is either $*\mathtt{y}$ or $\circledast\mathtt{y}$) only if there is *exactly one* leaf ℓ in t with matching first feature $\mathtt{y}$. This is at least conceptually related to (although formally quite different from) the shortest move constraint (Chomsky, 1995), and is called the SMC (Stabler, 1997) – it requires that an expression move to the first possible landing site. If there is competition for that landing site, the derivation crashes (because the losing expression will end up having to make a longer movement than absolutely necessary). If it applies, **move** moves the maximal projection of ℓ to a newly created specifier position in t (overtly, in the case of $*\mathtt{y}$, and covertly, in the case of $\circledast\mathtt{y}$), and deletes both licensing features. To make this precise, let

$t\{t_1 \mapsto t_2\}$ denote the result of replacing all subtrees t_1 in t with t_2, for any tree t, and let ℓ_t^M denote the maximal projection of ℓ in t, for any leaf ℓ.

$$\textbf{move}(t[\langle \alpha, *\mathtt{y}\delta\rangle]) = \begin{array}{c} > \\ t'[\langle \beta, \gamma\rangle] \quad t[\langle \alpha, \delta\rangle]\{t' \mapsto \langle \epsilon, \epsilon\rangle\} \end{array} \qquad (\text{where } t' = \langle \beta, \mathtt{y}\gamma\rangle_t^M)$$

$$\textbf{move}(t[\langle \alpha, \circledast\mathtt{y}\delta\rangle]) = \begin{array}{c} > \\ \langle \epsilon, \gamma\rangle \quad t[\langle \alpha, \delta\rangle]\{t' \mapsto t'[\langle \beta, \epsilon\rangle]\} \end{array} \qquad (\text{where } t' = \langle \beta, \mathtt{y}\gamma\rangle_t^M)$$

An expression is *complete* just in case it has exactly one attractee feature at its head – this feature can be thought of as its 'category' in the traditional sense.

The SMC is a substantive restriction on MGs (Salvati, 2011), which guarantees their computational efficiency. Note that, since movement can only apply if there is at most one subtree with the relevant y feature, then if at any point a tree with multiple y heads is derived, then the **move** operation can never check these y features, and thus this tree can never be part of a derivation of a complete expression (Michaelis, 2001). Thus, any tree which is either itself complete, or is produced during the derivation of a complete expression, has at most one subtree hosting a y feature for every feature type Y. This means that in the type of an expression t, $\mathsf{type}(t) = \langle \alpha, A\rangle$, the component A consisting of the feature bundles of all leaves (other than the head of t) with unchecked features, can be construed as a simple set of feature bundles. Moreover, because of the SMC, if a feature bundle in A begins with a feature y, it is the only feature bundle which does; thus A can also be thought of as a *partial function* from feature types Y to the unique feature bundle beginning with a feature of that type, if there is one.

2.2. Analytical Background

The basic clause structure assumed here is as in figure 3 (Koopman and Sportiche, 1991; Koizumi, 1995). Sentences with this clause structure can be derived using the

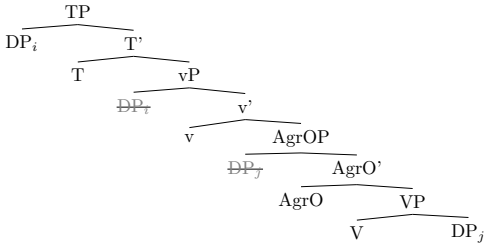


Figure 3: Basic clause-structure

lexical items in figure 4 (adapted from Kobele (2006)). A transitive verb, here praise,

$\langle \mathsf{will}, *\mathtt{v}\ *\mathtt{k}\ \mathtt{t}\rangle$	$\langle -\epsilon, \circledast\mathtt{agr0}\ *\mathtt{d}\ \mathtt{v}\rangle$	$\langle -\epsilon, \circledast\mathtt{V}\ \circledast\mathtt{k}\ \mathtt{agr0}\rangle$	$\langle \mathsf{praise}, *\mathtt{d}\ \mathtt{V}\rangle$

Figure 4: Lexical items for figure 3

is first merged with a DP (its logical object) to form a VP. This VP then is merged with $\langle -\epsilon, \circledast \texttt{V}\ \circledast \texttt{k}\ \texttt{agrO} \rangle$, triggering head movement of the V head to this higher AgrO head.[11] Next, the DP is covertly moved to the specifier of AgrOP to check its case (k) feature. This AgrOP is then merged with $\langle -\epsilon, \circledast \texttt{agrO}\ \texttt{*d}\ \texttt{v} \rangle$, again triggering head movement of the complex AgrO head. Next a DP (the logical subject) is merged into the specifier of vP. This vP is then merged with will, and then the subject DP is overtly moved to the

$\texttt{TP} \sim \langle \texttt{t}, \emptyset \rangle$	$\texttt{AgrOP} \sim \langle \texttt{agrO}, \emptyset \rangle$
$\texttt{T'} \sim \langle \texttt{*k}\ \texttt{t}, \{\texttt{k}\} \rangle$	$\texttt{AgrO'} \sim \langle \circledast \texttt{k}\ \texttt{agrO}, \{\texttt{k}\} \rangle$
$\texttt{vP} \sim \langle \texttt{v}, \{\texttt{k}\} \rangle$	$\texttt{VP} \sim \langle \texttt{V}, \{\texttt{k}\} \rangle$
$\texttt{v'} \sim \langle \texttt{*d}\ \texttt{v}, \emptyset \rangle$	$\texttt{DP} \sim \langle \texttt{d}\ \texttt{k}, \emptyset \rangle$

Figure 5: Categories and Types

specifier of TP to check case. It is important to emphasize that object case checking is in this analysis *different* from subject case checking, in that the former is *covert* ($\circledast$k) and the latter is *overt* (*k). That object case checking is covert is motivated by the upcoming generalization 1 in section 3.

Given these lexical items, each internal node in the tree in figure 3 is associated with a particular type, as depicted in figure 5.

2.3. Derivations

A *derivation tree* is a (complete) description of how to construct an expression. Formally, a minimalist derivation tree has leaves labeled with lexical items and internal nodes labeled with either **merge** or **move**. As an example, the derivation tree corresponding to the structure derived in iii in the previous subsection (the sentence *Carl will run*) is given in figure 6.

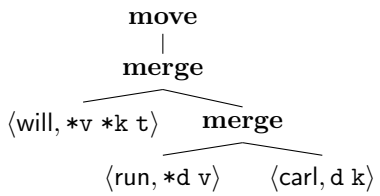


Figure 6: A derivation tree

Even though, formally speaking, a derivation is simply a tree-like object, which is connected to another tree-like object, the derived structure, in a regular way, it is sometimes helpful to think of derivations procedurally, as instructions for constructing a derived structure.[12] A derivation tree is then a description of a *process*. A subtree thereof represents a subprocess, describing how to construct one of the ingredients to be used. There

[11] The symbol ϵ indicates that the lexical item has no phonetic content.

[12] Thinking about derivations procedurally can also be extremely misleading. The process perspective on derivations is orthogonal to the process of parsing; even though in algebraic formalisms like minimalist grammars the most natural procedural perspective on derivations is bottom-up, it is straightforward to construct correct parsers which recognize these 'bottom-up' derivations in a top-down left-to-right manner (Harkema, 2001; Stabler, 2013).

$\langle -\mathsf{en}, \circledast \mathtt{V}\ \mathtt{pass}\rangle$	$\langle \mathsf{seem}, *\mathtt{c}\ \mathtt{v}\rangle$
$\langle \mathsf{be}, *\mathtt{pass}\ \mathtt{v}\rangle$	$\langle \mathsf{to}, *\mathtt{v}\ \mathtt{c}\rangle$

Figure 7: Passive and Raising

are as many subtrees of a tree as there are nodes; each node determines a subtree whose root is that node. For example, the **merge** node immediately dominating run and carl is the root of a subtree which describes the process of merging run and carl, which results in the derived tree in i.

Derivation trees, viewed as recipes for constructing expressions, can be used to represent the expression obtained by following the recipe. A derivation tree which consists of a single node labeled with a lexical item represents that lexical item. A tree t with root labeled **move** and with single daughter t' represents the result of applying the **move** operation to the expression t' represents, and a tree t with root labeled **merge** and with daughters t_1 and t_2 represents the result of applying the **merge** operation to the expressions represented by t_1 and t_2. A derivation tree which represents an expression is called *convergent*.[13] To convergent derivation trees are associated the types of the expressions they represent. In other words, if t is a derivation tree which represents e, then $\mathsf{type}(t) = \mathsf{type}(e)$. A non-convergent derivation tree has no type.

Derivation trees will figure prominently in the remainder of this paper. This will result in an unfortunate use/mention ambiguity; the expression **merge**(run, carl) might either denote the result of merging the lexical item run with the lexical item carl, or the *derivation tree* which describes this process. When it becomes important to disambiguate these two, $\mathsf{spellOut}(\alpha)$ will denote the result of carrying out the process described by the derivation tree α.

2.4. The analysis continued

To derive passive sentences, the two lexical items on the left in figure 7 are used. These implement an analysis of passives whereby a head (-en) which does not assign case merges with a VP, and then another head (be) merges with the result, forming a vP. This is just a recasting of Jaeggli (1986) in more modern terms. An example derivation, and the resulting derived structure, of a passive sentence is given in figure 8. Of course, one doesn't say "*praise-en*" but rather "*praised*." Head movement arranges lexical formatives

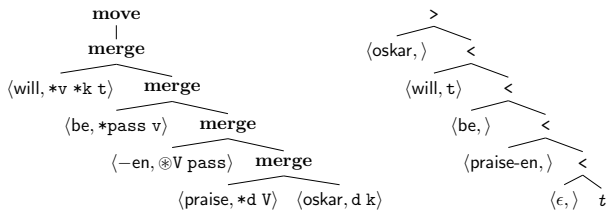


Figure 8: A derivation tree (left) for a passive (right).

[13] As an example, the derivation tree **move**(**move**(**move**(carl))) is not convergent.

so as to have stems adjacent to their affixes. The need for a (post-syntactic) theory of morphology is not thereby eliminated. In the present simplified setting, a transductive theory like that of Beesley and Karttunen (2003) easily maps complex heads like *praise-en* to the desired *praised*. More involved theories of morphology, such as Distributed Morphology (Halle and Marantz, 1993) or Paradigm Function Morphology (Stump, 2001), and even head movement itself, can be viewed as special cases of transductive theories (Kobele, 2012c).

Extending the fragment to derive raising-to-subject sentences, the two lexical items on the right in figure 7 implement a literal raising to subject analysis.

3. LF-copying, derivationally

According to both LF-copying (Chung et al., 1995) and proform (Hardt, 1993) approaches to ellipsis, ellipsis sites are syntactically atomic; they do not contain unpronounced structure. The differences between the two can be usefully thought of in terms of what antecedents are, and how ellipsis sites are resolved. In the LF-copying approach, antecedents are syntactic objects, and ellipsis sites are resolved by replacing them with their antecedent in the syntax, whereas in the proform approach, antecedents are semantic objects, and ellipsis sites are resolved by replacing them with their antecedent in the semantics. In the derivational copying approach introduced here, ellipsis sites are resolved by replacing them with their antecedents semantically, but antecedents are delimited syntactically.

The theory advanced herein takes the derivation to be the relevant level of structure. To get an intuition for how this could work, the LF-copying approach is recast in these terms. Consider the discourse "Carl will praise Oskar. Oda will, too." Clearly, the sentence "Carl will praise Oskar" is providing an appropriate antecedent for the subsequent elliptical sentence. The structures of the unelided versions of these sentences share a common subderivation, colored in in figure 9. Because these are derivation trees, not

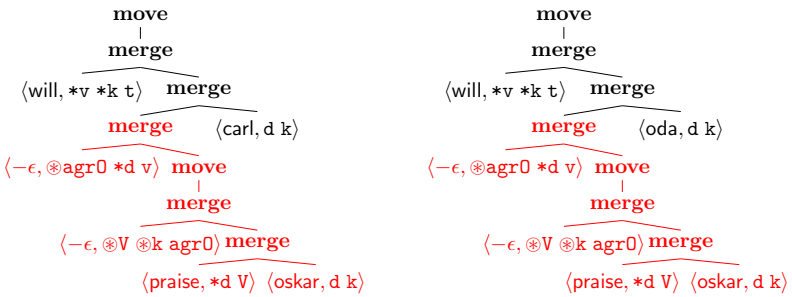


Figure 9: Shared structure

derived trees, antecedents can be viewed procedurally; an antecedent is a sequence of derivational steps which have already been performed.

Note that not just any part of a derivation tree provides an appropriate antecedent for an ellipsis site. First and foremost, it must have the correct syntactic type. (Work by Yoshida (2010) challenges even this basic assumption; his analysis is incompatible with

this setup.) This assumption is common to nearly all syntactic approaches to ellipsis, and can be thought of as a refinement of the basic semantic constraint that the meanings of the elliptical pieces must fit appropriately into the meanings of the surroundings. One important difference between the current approach and these others is that syntactic types provide a much more refined notion of syntactic categories.

If the tree on the right of figure 9 is viewed, in accord with a copying theory, as the result of replacing an ellipsis site with the antecedent context (the colored part of the tree on the left), the tree in figure 10 could be thought of as underlying the sentence "Oda will." The bold **e** in figure 10 is what the antecedent context (the colored subtree

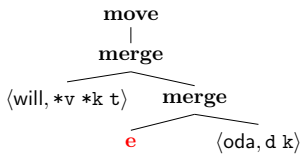


Figure 10: The structure of an elliptical sentence

on the left in figure 9) replaces to obtain the tree on the right of figure 9. It represents the ellipsis site. Figure 10 will be the representation adopted in this paper for elliptical sentences. It remains to understand what it means.

Since **e** occurs as a node in a derivation tree, it must have a status similar to the other nodes; i.e. it must be a grammatical operation (like **merge** and **move**). Grammatical operations are functions over expressions. The operation **merge** is a binary operation (i.e. it takes two arguments), **move** is unary (i.e. it takes one argument), and, as seen here, **e** is nullary (i.e. it takes no arguments). To define **e**, one must specify how it maps inputs to outputs. The derivation tree in figure 10 constraints the possible definitions of **e**: it must be something that *Oda* can be merged with, giving rise to something which *will* can merge with. In other words, it must play exactly the same role in the derivation in 10 that the colored subderivation in figure 9 would; it must give an expression with a feature bundle of the form `*d v`. This amounts to saying that **e** has type $\langle$`*d v`$, \emptyset\rangle$, which is abbreviated as `v'`, just as does the antecedent subtree. Looking ahead to §3.4, the operation **e** is *parameterized* with a type, in this case `v'`. This permits ellipsis operations at different types to be distingushed (Merchant, 2004), which may prove useful in an account of why ellipsis constructions differ across languages. For instance, German does not have a VP-ellipsis construction, but it does have sluicing and gapping constructions. Accordingly, given a syntactic type τ, there is an ellipsis operation $\mathbf{e}_\tau$ at that type. In these terms, the ellipsis operation underlying the sentence *Oda will, too* is $\mathbf{e}_{\texttt{v'}}$. Thus is resolved half of the question about the nature of the operation **e**. Next is the question of what effect **e** has on the derived structure of an expression.

3.1. Resolution

In the LF-copy theory, the ellipsis site is simply replaced, after being pronounced, by its antecedent. A first approximation has it that any appropriately categorized object in the discourse can antecede an ellipsis site. This is only a first approximation; it is well-known that the actual contextually possible antecedents are a fraction of the logically possible ones. Ported over to the derivational setting, the basic idea would be to treat

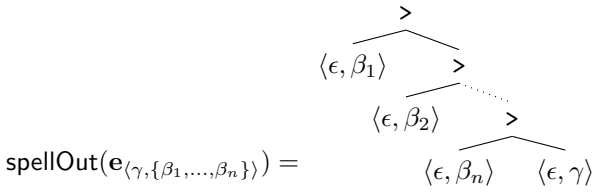


Figure 11: The spellout of ellipsis operations

the object derived by **e** as the same as the object derived by its antecedent, modulo pronounced material; loosely speaking, $\mathsf{spellOut}(\mathbf{e}_{\texttt{v'}}) = \mathsf{delete}(\mathsf{antecedent}(\mathbf{e}_{\texttt{v'}}))$.[14]

This idea, though simple, misses the obvious (but important) point that, no matter what the antecedent, $\mathsf{spellOut}(\mathbf{e}_{\texttt{v'}})$ is always pronounced the same. (This also violates the spirit of a syntactic version of the principle of compositionality.) In other words, the *form* of an elliptical sentence is completely independent of its antecedent, and only the *meaning* thereof is not. In the LF-copying theory, this fact is reduced to rule-ordering; pronunciation happens before ellipsis resolution, which happens before interpretation. The fact that ellipsis sites are pronounced the way they are (as nothing, rather than as something) is a brute stipulation. Here, the phonetic form of the expression generated by a nullary ellipsis operation must still be stiplulated, but there is no need for stipulations about when ellipsis resolution occurs. There are any number of possible derived structures which could reasonably be associated with ellipsis sites; the one shown in figure 11 will be used here. This derived structure encodes the information that (i) the head of this expression has features γ, and no phonological content, and (ii) it contains n silent moving subparts, with features β_1 through β_n. For the special case of $\mathbf{e}_{\texttt{v'}}$, we have that $\mathsf{spellOut}(\mathbf{e}_{\texttt{v'}}) = \langle \epsilon, \texttt{*d v} \rangle$ (recall that `v'` is an abbreviation for the syntactic type $\langle \texttt{*d v}, \emptyset \rangle$).

Part of the interest in this account of ellipsis resolution is that all that needs to be known about the antecedent is its syntactic type, not its internal structure. The internal syntactic structure of an expression is the glue that connects its pronounced form to its meaning. As the pronounced form of ellipsis is completely independent of any antecedent, there is no need to reconstruct a syntactic structure in the ellipsis site. Instead, an ellipsis site can be thought of as denoting a proform, which is anaphoric on the meaning of some derivational structure of the appropriate type. This observation makes a connection between the derivational copying account of ellipsis and a proform account of ellipsis, with the former being a special kind of the latter.

3.2. Ellipsis operations of different types

Revisiting figure 9, observe that the colored subderivations are not the only parts shared between the two trees; indeed, they are the maximal shared subderivations. Choosing instead the **merge** node immediately above *oskar*, we would have an ellipsis node of type $\langle \texttt{V}, \{\texttt{k}\} \rangle$, abbreviated as `VP`. In this case the sentence *Oda will, too* would

[14] This suggests a relation between deletion and copying theories of ellipsis. In Kobele (2012a) it is argued that copying theories should be thought of as descriptions of the algorithm implementing deletion theories.

have the derivation on the left in figure 12, with the structure on the right the corresponding derived object. (The elements colored in in the structure on the right are from the ellipsis site.) Given that both figure 10 and figure 12 are pronounced as *Oda will*,

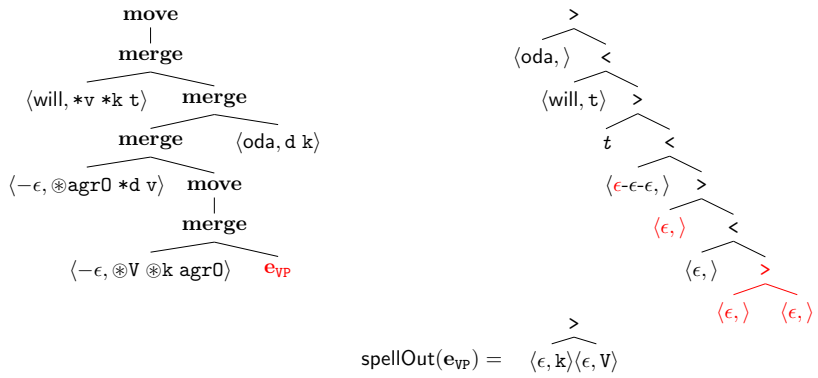


Figure 12: Another potential structure for VPE

and are interpreted identically in the context of the sentence "*Carl will praise Oskar*," one might wonder how to choose between them.[15] While there are some basic structural differences between $\mathbf{e}_{\texttt{v'}}$ and $\mathbf{e}_{\texttt{VP}}$, such as that only the latter represents the ellipsis of a maximal projection, a more fundamental difference is that $\mathbf{e}_{\texttt{VP}}$ allows for passive-active mismatched verb phrase ellipsis as in 2. Consider the discourse "*Oskar will be praised. Oda will.*," where the first sentence has a structure as in figure 8, and the second sentence means that Oda will praise Oskar.[16] This sentence provides no antecedent for an ellipsis operation of type `v'`, but does one for one of type `VP`. Thus the derivation in figure 12 can serve as the structure of "*Oda will, too*" not only when it is anteceded by an active, but also when it is anteceded by a passive.

3.3. Restrictions on movement out of ellipsis sites

The general form of the result of an ellipisis operation shown in figure 11 permits, using a deletion metaphor, movement of expressions which have been deleted (those with feature bundles β_i). This should not be allowed. Consider 15 below, where the intended reading is indicated with crossing-out.

15. Oskar should be praised. *~~Oskar~~ won't be ~~praised~~.

[15] Kim et al. (2011) suggest that both options be allowed.

[16] This discourse (and indeed most of the very short example discourses presented in this paper) is not a very natural one. A complete theory of ellipsis must account for this fact. One basic strategy (adopted here) is to say that this discourse is well-formed syntactically and semantically, and to then appeal to some other factors to account for its deviance. A natural way to account for 'unexpected unacceptability' is by appeal to properties of language use. Kim et al. (2011) explores how performance factors could be added to a system like the one presented here. Another strategy is to say that this discourse is ill-formed syntactically or semantically, and to then appeal to some other factors to account for the acceptability of superficially similar ones as in example 2. I do not know of a way of determinining *a priori* which strategy is going to prove more fruitful in any given case.

This is derivable with the current grammar fragment, where the structure of the elliptical sentence is as in figure 13. An important difference between the desirable derivation in

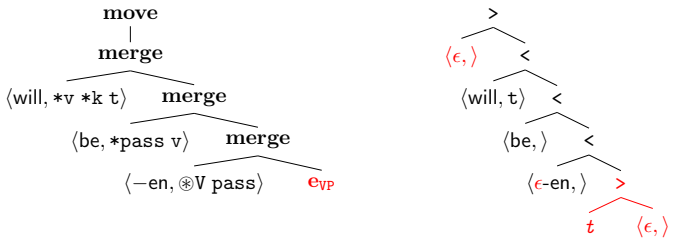


Figure 13: The structure of the elliptical sentence in 15

figure 11 and the undesirable one in figure 13 is that the source and target positions of the movement in 11 are not separated by any overt elements, while those in 13 are. In other words, the good movement is string vacuous, while the bad one is not. As it is more natural to state this sort of restriction in terms of covert movement (i.e. triggered by a feature of the form ⊛x) than in terms of string vacuous movement, I propose the following stronger generalization.[17]

Restriction 1. *All movement features (*y*) generated inside of an ellipsis site must be checked covertly (⊛*y*).*

This stipulation accounts for the ungrammaticality of the derivation in 13 because the elliptical sub-piece $\langle \epsilon, \mathtt{k} \rangle$ has its k feature checked by the *k feature of *will*. An equivalent way of putting this views *k as the combination of ⊛k with an EPP-feature, and then restriction 1 is a prohibition against elliptical sub-pieces being used to satisfy EPP-features.

3.4. Antecedents of a higher type

Consider the discourse "Carl will be praised. Oskar will be, too." Comparing the derivations of the unelided versions of these sentences in figure 14, they share no non-trivial subtrees, but would if their different DPs could be ignored. This situation is familiar

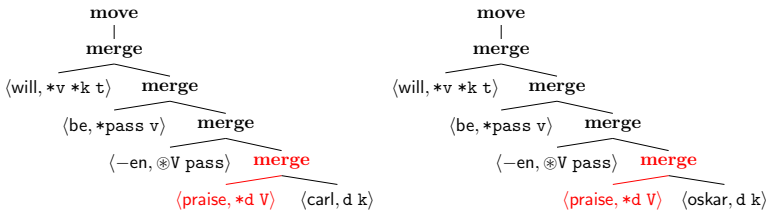


Figure 14: Shared structure

[17] Note that covert movement is by definition string vacuous, but that not all string vacuous movement need be covert.

from theories of ellipsis which make use of derived structure. There the offending DPs are moved out, and an operation (*trace conversion* in Fox (2002), α-conversion in Sag (1976)) makes what is left identical. Because of the present focus on derivation trees (and not derived trees), operations which modify internal structure are not available. Instead, the effects of derived-structure analyses must be restated in derivational terms. These analyses have the common effect of ignoring parts of a subtree; accordingly, we need to be able to talk about trees with missing parts, like the colored in portions of the trees in 14.

3.4.1. Tree Contexts

Parts of trees like these colored in almost-constituents are known in the computer science literature (Comon et al., 2002) as *(tree) contexts*. The context in figure 15 occurs in the derivation tree in figure 14, where the empty box ($\Box$) represents a missing piece, which, in its occurrance in figure 14, is filled by *carl*. Viewed procedurally, it describes a function which takes an argument and merges *praise* with it. We restrict our attention

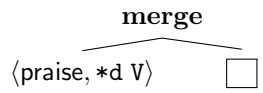


Figure 15: A unary derivation tree context

to *unary* contexts (contexts with just one hole) in this paper.[18] Given a context C and a tree t, $C[t]$ denotes the tree obtained by putting t into the hole in C. Note that tree contexts, viewed as procedures, are not defined on all arguments. In the case of figure 15, only inputs i which can be merged with *praise* (i.e. whose heads have first feature d) are legitimate arguments. Types describe the behaviour of derivations. As a context is just a subtree with a piece missing, if you put the missing piece back in, you should get a subtree, which has a type in the usual way, call it c. But the missing piece (also a subtree) has a type too, call it c'. Then the type of a context can be thought of as a *function* $c' \to c$. More concisely, if C is a context, and i is an input such that $\mathsf{type}(C[i])$ is defined, then C has type $\mathsf{type}(i) \to \mathsf{type}(C[i])$. The context in 15 has type $\langle \mathtt{d\ k}, \emptyset\rangle \to \langle \mathtt{V}, \{\mathtt{k}\}\rangle$, or, in abbreviated form, DP → VP; it is a procedure which, given a DP, constructs a VP.[19]

[18] A more general solution (Kobele, 2012b) involves treating trees as λ-terms, and contexts as λ-abstracts. For example, the tree in figure 14 can be represented as the first-order λ-term **move**(**merge**(will)(**merge**(be)(**merge**(-en)(**merge**(praise)(carl))))), and the context in figure 15 as the second-order λ-term $\lambda x.$**merge**(praise)(x). Restricting attention to *contexts* amounts to, in the more general setting of the λ-calculus, limiting our attention to terms of order at most two.

[19] This is not yet entirely satisfactory. Consider again our context which is a VP missing a DP. One type it should have is $\langle \mathtt{d\ k}, \emptyset\rangle \to \langle \mathtt{V}, \{\mathtt{k}\}\rangle$, i.e. something which, if you give it a DP will result in a VP. But it also has the type $\langle \mathtt{d\ k\ wh}, \emptyset\rangle \to \langle \mathtt{V}, \{\mathtt{k\ wh}\}\rangle$, i.e. it is something which, if you give it a [+wh] DP will result in a VP which contains a *wh*-word. Thus, this context has at least two categories as we have defined them. In fact, it is easy to see that it has infinitely many categories; for any α, it has the category $\langle \mathtt{d}\ \alpha, \emptyset\rangle \to \langle \mathtt{V}, \{\alpha\}\rangle$. However, all of these categories are related in a natural sense. A natural way to express this relation *in the type system* is to quantify over feature bundles; we might assign it the (single) type $\forall\alpha.\langle \mathtt{d}\ \alpha, \emptyset\rangle \to \langle \mathtt{V}, \{\alpha\}\rangle$. Technically, because there are only a finite number of useful categories in Minimalist Grammars with the shortest move constraint (Michaelis, 2001), we do not have to use such a powerful type theory. We could use intersection types, and express the type of the ellipsis

3.4.2. Passive-Passive VPE revisited

Returning to the motivating example of figure 14, we see that antecedents must be *derivational contexts* – recall that a subtree is a special case thereof, and so this is a strict generalization of the previous perspective on ellipsis. Viewing the tree on the right of figure 14 as the result of replacing an ellipsis site with the antecedent context (the colored in part of the tree on the left), the tree in figure 16 can be thought of as underlying the sentence "Oskar will." The bold **e** in figure 16 is what the antecedent

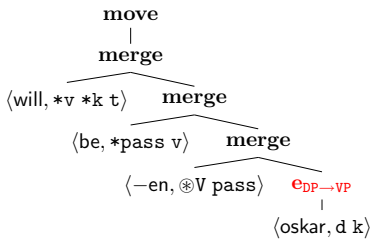


Figure 16: The derivation of the elliptical sentence "*Oskar will be.*"

context replaces to obtain the tree on the right of figure 14. It represents the ellipsis site. It differs from $\mathbf{e}_{\mathtt{vP}}$ in that $\mathbf{e}_{\mathtt{DP}\rightarrow\mathtt{VP}}$ is a unary operation, whereas $\mathbf{e}_{\mathtt{vP}}$ is a nullary operation. The natural generalization is to say that $\mathbf{e}_{\tau}$ is a grammatical operation, which takes a number of arguments appropriate to its type τ. Note that this kind of view is forced by a derivational presentation of a copying theory of ellipsis; if movement chains are created via movement, then ellipsis sites must be able to contain expressions which have moved out of them. The derivational solution to this puzzle is to treat ellipsis sites as operations which apply to the expressions that, on a deletion approach, one would say were generated within them but not deleted.

Now $\mathbf{e}_{\mathtt{DP}\rightarrow\mathtt{VP}}$ has been determined to be a unary grammatical operation, the next question to be asked is what effect it has on the derived structure of an expression. As before, there are many possible (and equally good) answers to this question; as we are here restricting our attention to unary contexts, the special case in figure 17 will suffice.[20] We may revisit our conclusions about the structure of active-active VPE reached above in light of the richer system of ellipsis operations now available to us. Looking back at figure 14, a context of type $\mathtt{DP} \rightarrow \mathtt{vP} = \langle \mathtt{d\ k}, \emptyset\rangle \rightarrow \langle \mathtt{v}, \{\mathtt{k}\}\rangle$ is also shared between the two structures (this is the context which includes all of the colored in subtree, plus the

site in terms of the coordination of all of the simple types we would normally want to assign to it. However, this would not allow us to express the similarities between these types, which we can do in a modern type theory (Martin-Löf, 1984; Luo, 1994); we need to (among others) be able to quantify over feature bundles and test whether two feature bundles are both non-empty.

[20] We need one variant of each $\mathbf{e}^{f}_{\tau}$ for every grammatically permissible way f of linearly ordering the moving pieces in expressions which are arguments of $\mathbf{e}^{f}_{\tau}$. As expanded upon by Kobele (2012a), there is a close relationship between PF-deletion theories and LF-copying theories. The parameter f can be thought of as expressing how the deleted material manipulates the non-deleted material; in other words, f describes what would have happened if you had merged and moved as described by the antecedent, and then deleted all of the formatives which are part of the antecedent. An important difference is that, for each type τ, there are only finitely many possible f, whereas there are (generally) infinitely many possible deleted structures.

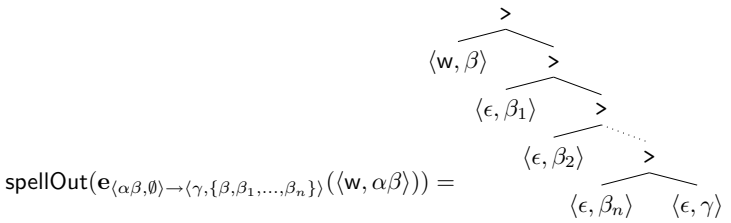


Figure 17: The spellout of unary ellipsis operations

parent of the root; the missing piece is the sister of the root of the colored in subtree, the logical subject of the clause). Note that restriction 1 blocks the otherwise possible context of identical type which is missing the logical object instead of the logical subject.

A final note about the structure in figure 16 is in order. The derived tree has as its yield the string "*Oskar will be -en*," and not the string "*Oskar will be.*" In line with Lasnik (1981), the stranded affix *-en* can be assumed to be deleted by some post-syntactic rule.[21] Not all affixes stranded in this way behave alike (Potsdam, 1997); the English tense affixes *-s* and *-ed* surface as *does* and *did*. I have nothing insightful to say about this; in the present simple treatment of morphology, a rule of *do*-insertion ordered before the stranded affix deletion rule will work.

3.4.3. Contexts and overgeneration

Although allowing contexts to antecede ellipsis sites is the derivational version of modifying derived structure, there is a justifiable worry about overgeneration. Indeed, there are many more (unary) contexts than subtrees of a given structure, and, although*some* of these contexts are needed, we certainly do not want them all. This problem is not unique to the present theory however (although it is worse here): even restricting attention to subtrees, there are far more subtrees than possible ellipsis sites. There are two standard mutually compatible approaches to this sort of problem. The first approach to this problem is to restrict the distribution of ellipsis. I have already appealed to this in one form, by suggesting that each language pick ellipsis operations of particular (possibly different) syntactic types. In addition one can further restrict the distribution of ellipsis sites by requiring that they have some sort of contextual property (such as being head governed (Lobeck, 1995), or 'maximal' in some sense (Merchant, 2008)). The second approach to this problem is to focus instead on what makes a good antecedent. It is well known that information structure is a significant factor in making a good antecedent (Kertz, 2010). Additionally, especially when antecedents are delimited in terms of their syntactic properties, it is possible to make reference to these syntactic properties when talking about what makes an antecedent good. Two possibilities are the following. First, one can impose a restriction on the distribution of 'holes' in a potential antecedent to the effect that a subtree can be left out of an antecedent (i.e. can be part of the hole) just in case it has some important property, such as being focussed, having a particular

[21] Another alternative is to treat the head *-en* not as an independent affix but rather as an empty head which influences a morphological feature (verb-form) of its (in this case silent) complement. I view this as equivalent in all relevant respects.

syntactic feature, or not being c-commanded by another expression of the same syntactic type. Second, as mentioned in Kobele (2012b), parsers incrementally construct derivational contexts during a parse. This suggests relating the possible antecedent contexts to those which are constructed during the parsing process (Lavelli and Stock, 1990).

A theory of the mechanisms of ellipsis (such as the one presented here) is a necessary part of an account of ellipsis. As shown in the next section, the derivational copying theory is able to give a simple and unified account of some otherwise unexplained data. Before celebrating, we should of course remember that it relies on as yet unspecified theories of the distribution of ellipsis sites, and of what makes for a good antecedent (a property shared by all of its competitors). Still, one advantage of the present theory is that it is formally explicit, and therefore its predictions can be teased out with pencil, paper, and sufficient time and interest.

4. Case studies

The previous sections have introduced the minimalist grammar formalism (§2), and the extension to it of a derivational ellipsis mechanism (§3). Although this formal framework was presented in tandem with a particular analysis of English syntax, it is important to note that the formal framework is not tied to any particular analysis, although it does, as a restrictive grammar formalism, make substantive claims about what kinds of patterns one can find realized in natural language. The fundamental claim of this paper is the following:

Claim 1. *Apparently structure-sensitive properties of ellipsis are reducible to syntactic types.*

Moreover, analyses in the minimalist grammar framework end up assigning to expressions the right kinds of syntactic types. To provide support for claim 1, natural and independently motivated analyses of English clause and preposition structure and *wh*-movement will now be presented, and it will be demonstrated that these account for the relevant elliptical data as well, in conjunction with the derivational-copying theory of ellipsis from §3.[22]

Before beginning, a cautionary note is in order. Many factors influence the acceptability of elliptical sentences (Kehler, 2002; Kertz, 2010). The particular sentences derived here are typically not very acceptabile. Indeed, they were chosen primarily for their short length, so as to admit derivation trees which fit legibly on the page. They do however exemplify sentence types which have tokens of high acceptability, and, in line with the discussion in §1, the current project is to account for generalizations I–III, with the hope being that such an account will provide the scaffolding on which to hang a more complete account of the data.

[22] A reviewer notes that the analyses presented herein do not do justice to the richness of the respective phenomena. Implicit in what follows of course is the hope that these simplified syntactic analyses fit so neatly with the derivational copying theory of ellipsis not because of some accident, but instead because the framework, the theory of ellipsis, and the analysis are on the right track, and that therefore more sophisticated analyses of clauses, prepositions, and *wh*-movement will continue to support claim 1.

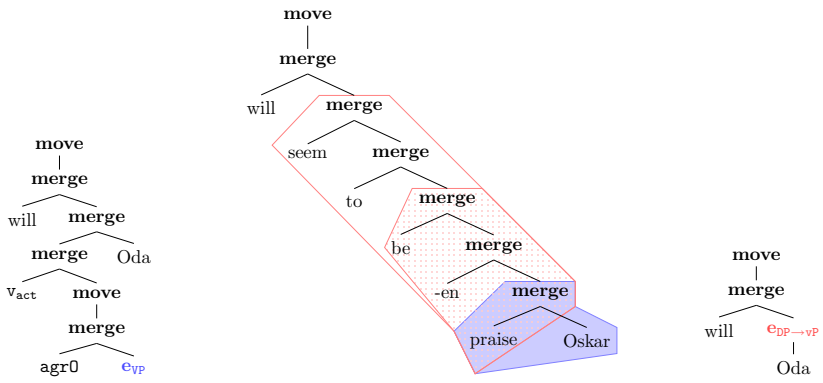
4.1. Voice (mis)matches

Verb phrase ellipsis has been the running example in the previous section, and we have seen that active-passive and passive-active mismatches are derivable in the present theory of ellipsis. The types compatible with the given analysis are as in figure 18. Not

Antecedent	Ellipsis	Type(s)
active	active	`v'`, `VP`, `DP` → `vP`
active	passive	`DP` → `VP`
passive	active	`VP`, `DP` → `VP`
passive	passive	`DP` → `VP`, `DP` → `PassP`

Figure 18: Types for VPE

all of these types have been discussed, nor have possible relations between them been explored. Kim et al. (2011) suggest, with terminology adapted to the slightly different theory, that the attested acceptability gradients between each of the four conditions in figure 18 are predictable from the height of the result category of the respective ellipsis sites; and thus the mismatched VPE conditions (the common highest result category is `VP`) are worse than the matched conditions (with highest result categories `vP` and `PassP` respectively), and that the active-active condition is better than the passive-passive one. Here it is shown that this 'voice insensitivity' which has been derived for VPE does not

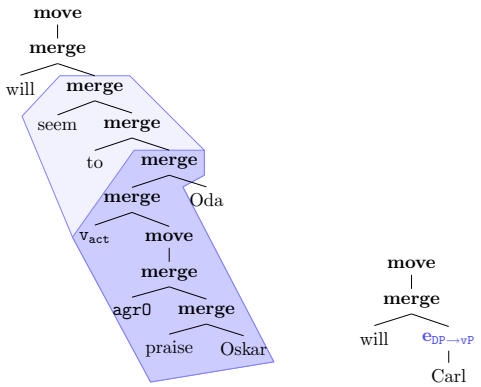


(a) VPE (b) Antecedent (c) vPE

Figure 19: Oskar will seem to be praised. *Oda will ~~seem to praise Oskar~~

extend to sluicing, nor does it apply to non-'root' VPE contexts, as per the discussion in §1.1. Figure 19 presents the structure for a VPE sentence like 5, with mismatched voice features internal to the antecedent and the desired resolution of the ellipsis site. The ellipsis sites and their possible antecedents are color-coördinated in the figure; the blue ellipsis site of type `VP` in subfigure 19a must take as antecedent something of type `VP`; the only possible such being outlined in blue in subfigure 19b. Similarly, the two possible antecedents of type `DP` → `vP` in subfigure 19b are outlined in red. Inspection of the figure reveals that the undesired non-'root' voice mismatch, which would mean *Oda will ~~seem to praise Oskar~~*, is not possible with these structures. That it is not possible with *any*

structure can be determined as follows. Because oda is overt (i.e. pronounced), oda must either be outside of the ellipsis site altogether (as in the structure in 19a), or it must be an argument to it (as in the structure in 19c). If it is an argument to the ellipsis site, the ellipsis site must take as antecedent a context missing a DP; the only possible such are those missing oskar, and so there is no way for this to give rise to the undesired reading. If oda is external to the ellipsis site, it must be merged with a structure containing the ellipsis site. As the antecedent derivation in 19b has no subtrees with an unchecked *d feature containing oskar, oda must be merged with a lexical item containing one such; an active voice head. But then this precludes the ellipsis site from having type vP, and thus it cannot contain seem. Figure 20 presents possible structures for the mismatching



(a) Antecedent (b) vPE

Figure 20: Oda will seem to praise Oskar. *Carl will ~~seem to be praised~~.

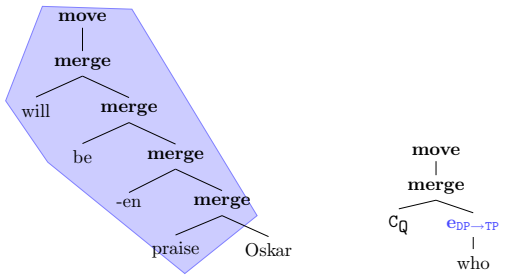
active-passive counterpart of figure 19. The reading which must be blocked is where the sentence is understood as *Carl will ~~seem to be praised~~*. The only way to obtain such a reading would have carl be an argument to the ellipsis site, and have it take an antecedent missing the logical object oskar. But the only such antecedents contain the logical subject oda, and thus by restriction 1 they are disallowed (as the silent ~~*Oda*~~ would need to move overtly to SPEC-TP for case). Instead, the only possible antecedents are where the logical subject oda is missing, shown in blue, and mean that *Carl will ~~praise Oskar~~* and that *Carl will ~~seem to praise Oskar~~*, which are appropriate for discourses of this general type. Examples with agentive *by*-phrases will be dealt with in §4.2 (see figure 24), after the introduction of PPs and adjunction.

$\langle \epsilon, \texttt{*t *wh c} \rangle$
$\langle \textsf{who}, \texttt{d k wh} \rangle$

Figure 21: Lexical Entries for *wh*-movement

As sluicing involves *wh*-words, an analysis of sluicing must also contain an analysis of sentences with *wh*-words. Merchant (2001) argues that the distribution of *wh*-words in elliptical contexts parallels the distribution of the same in non-elliptical contexts, and thus that there should be but a single statement governing the distribution of *wh*-words

across a language. Here it will be assumed that *wh*-words have the licensee feature `wh`, and that a silent CP provides an appropriate landing site. This analysis is a simple version of the canonical analysis of *wh*-movement in the generative tradition. More sophisticated variations on this theme could be investigated as well. These lexical items are in figure 21. A sluiced clause, following Merchant (2001), is analyzed as an actual CP with an ellipsis site. In the present analysis, a particularity of sluicing, as opposed to, say, VPE, is that the ellipsis site *always* has a higher order type – it is always of type $c \rightarrow c'$ – which means that it is a gap which contains some pronounced material (typically the *wh*-phrase which ends up moving to SpecCP). It is clear that it must be something of the rough form $c \rightarrow \langle \mathtt{t}, \{\mathtt{wh}\}\rangle$, as the result is a TP which contains a *wh*-moving phrase. But c could be any category which introduces a *wh*-movement feature; a proper *wh*-DP, or a *wh*-PP, or even some category which only contains a *wh*-mover.



(a) Antecedent (b) Sluice

Figure 22: Oskar will be praised, *but I don't know who ~~will praise Oskar~~

Figure 22 shows a sluicing construction, and its (only possible) antecedent. Note that this is the surface form that a passive-active voice mismatched sluice would take, which is here assumed in general to be unacceptable. Instead of being ruled out as ungrammatical however, the present analysis considers the sentence to be syntactically well-formed. Yet the reading is not the mismatching voice one of "I do not know who will praise Oskar", but rather the matching voice but nonsensical one of "I do not know who will be praised." I therefore attribute the unacceptability of this sort of sluice to semantic incongruity.[23] In support of this, note that the structurally identical sentence in 16 is well-formed.

16. Someone will be praised, but I don't know who ~~will be praised~~.

According to Chung et al. (1995), the ellipsis site in 16 must be fleshed out using the entire antecedent TP *Someone will be praised*. However, this entire TP will not 'fit' in the ellipsis site, as the *wh*-word *who* needs to bind a trace. They propose a new operation called *merger*, whereby a DP like *someone* is turned into a trace. The basic phenomenon that merger is intended to describe is that a constituent with a (usually indefinite) DP

[23] As pointed out by Tue Trinh, the unacceptability of this sentence is not obviously related to Moore's paradox, as changing *I* to *Oda* does not seem to improve matters. It is worth pointing out that changing *who* to *who else* does seem to make the sentence acceptable, which, depending on one's assumptions about the syntactic representation of information structure, suggests that the original sentence should in fact be generated by the grammar.

can serve as an antecedent for an ellipsis site which, intuitively, doesn't contain that DP. Of course, if the antecedent is not a constituent, but rather a context which excludes that DP, then no difficulties arise. In the derivational copying analysis, there is no separate phenomenon of merger; the offending DP is simply not part of the antecedent.

4.2. Sprouting

The phenomenon of sprouting encompasses sentences like 8–10, repeated below as 17–19:

17. Carl ate, but I don't know what.

18. Carl ate pancakes, but I don't know why.

19. Pancakes were eaten, but I don't know by whom.

In each of the three sentences above, it is not immediately apparent what sort of antecedent context is present which would have type $c_{wh} \rightarrow \langle \mathtt{t}, \mathtt{wh} \rangle$ (for c_{wh} some *wh*-feature containing category). In the LF-copying theory of Chung et al. (1995), according to which a derived syntactic antecedent is selected, and then manipulated via a set of transformations before being inserted in the ellipsis site, *sprouting* is the name of the transformation which inserts a trace (which can be then bound by the *wh*-phrase) in an appropriate place in the antecedent structure. What unifies the sprouting sentences 17–19 is that the relation between antecedent and (syntactically fleshed out) ellipsis site is one of suppressed and realized optional argument. According to the logic of the theory proposed here, we need to find an analysis of the above kind of structures which allows a context of type $c_{wh} \rightarrow \langle \mathtt{t}, \mathtt{wh} \rangle$ (for c_{wh} some *wh*-feature containing category) to be found.

This section focusses on the cases of sprouting involving adjuncts (adjunct *wh*-phrases, and *wh*-PPs, which are analyzed in the same way); whether the facts surrounding null objects (as with the intransitive usage of *eat* in 17) allow the same analysis to be maintained is unclear. The basic idea is that implicit arguments need to be syntactically represented. In the transformational literature this is often cast in terms of *pro* or PRO. In the present system, this amounts to having a special formative (which will be called *pro* and written as $\emptyset$), and to require that phrases which we normally think of as *allowing* adjunction actually *require* it. If this adjunct is *pro*, then it gives the appearance of optionality.[24] Adjunction has been implemented in many (subtly different) ways in minimalist grammars (Frey and Gärtner, 2002; Hunter, 2010; Fowlie, 2014). For reasons of space, I simply help myself to without explicitly defining an operation **adjoin**, and assume that it applies obligatorily wherever it can. The desired effects of the operation

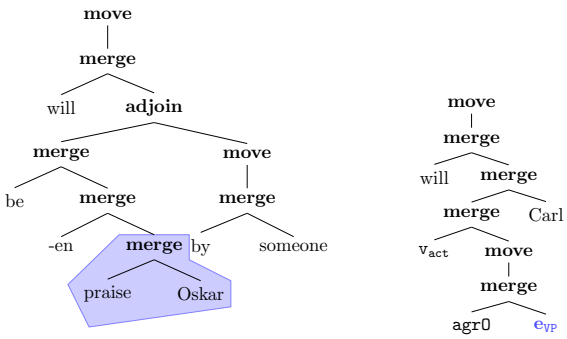
[24]It might be objected that this is unnecessarily complex, and *ad hoc*. However, the same idea is present even in the Tree Adjoining Grammar (TAG) formalism (Joshi, 1987). A natural perspective to take on adjunction nodes X in a TAG tree is that they are higher order non-terminals N_X. (This is made explicit in context-free tree grammar (Kepser and Rogers, 2011) or abstract categorial grammar (de Groote and Pogodalla, 2004) presentations of TAGs.) An adjunct XP of the form t is adjoined at an adjunction site by means of the rewrite rule $N_{XP}(x) \rightarrow N_{XP}(xp(x,t))$. Note that this rule preserves the 'adjoinable' status of the XP by introducing another non-terminal node N_{XP}. In order to have a well-formed tree, all adjunction sites must be 'closed off'. This is achieved via the rule $N_{XP}(x) \rightarrow x$. The important thing to note is that choosing not to adjoin something requires doing something (rewriting using the rule $N_{XP}(x) \rightarrow x$).

adjoin can be simulated with just **merge**, if we (i) assume that categories which must be adjoined to are of the form $\texttt{c}^*$ instead of $\texttt{c}$, and (ii) make use of the lexical items $\langle \epsilon, \texttt{*c}^* \ \texttt{*adj c} \rangle$ and $\langle \epsilon, \texttt{*x adj} \rangle$, for every category $\texttt{x}$ which can function as an adjunct. Concretely, it will be assumed that vP is an obligatory adjunction site. For simplicity, the

$\langle \texttt{by}, \texttt{*d} \circledast \texttt{k P} \rangle$

Figure 23: Lexical Entries for PPs

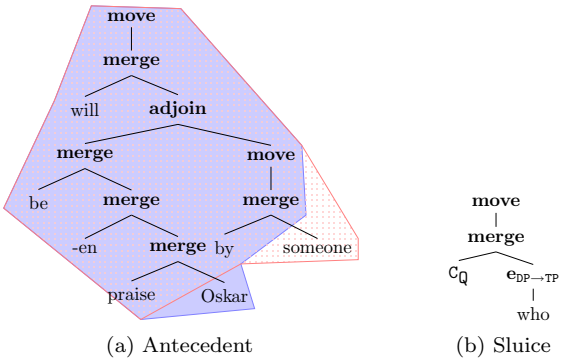
adjunct type investigated will be a PP; PPs will be assumed to covertly check the case of their objects. A P will then select a DP complement and assign it case. This lexical item is in figure 23. This allows the structure in figure 24 to be assigned to a passive sentence with an overt agent. With these structures in place, we can consider figures 25 and 26



(a) Antecedent (b) VPE

Figure 24: Oskar will be praised by someone, *but Oda won't ~~praise Oskar~~

in light of the discussion in §1.1. In figure 25, the most salient reading is the merger of who and someone, which is given by the blue context. A (for this sentence) less salient reading is the merger of who and oskar, given by the red context. The reading that would

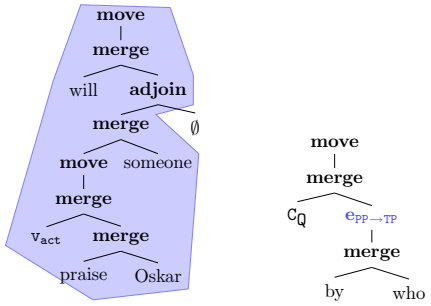


(a) Antecedent (b) Sluice

Figure 25: Oskar will be praised by someone, *but I don't know who ~~will praise Oskar~~

come from a mismatching voice antecedent, *who ~~someone praised~~*, is synonymous with

the reading given by the red context (*who ~~was praised by someone~~*). Still, it is clear that there is no active antecedent. In figure 26, matters are much clearer. We wish to verify that the elliptical sentence cannot mean that I do not know who praised Oskar. Observe that while there is a possible antecedent context, the PP *by whom* cannot be interpreted as an agentive *by*-phrase (as the subject, someone, is present in that context), but must rather, if at all, be interpreted as a locational PP. Implicit here is the assumption that the syntactic structure of passives with agentive *by*-phrases and passives without agentive *by*-phrases but with locative *by*-phrases is identical.[25]



(a) Antecedent (b) Sluice

Figure 26: Someone praised Oskar, *but I don't know by whom ~~Oskar was praised~~

4.3. No preposition stranding in Sprouting

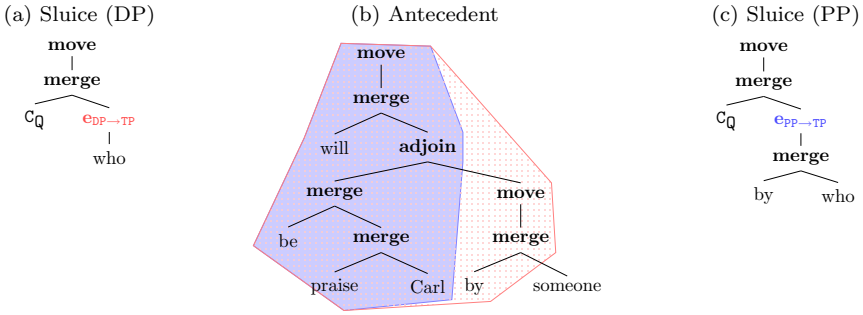
Chung et al. (1995) observe that, in constrast to the usual case in sluicing, in sprouting contexts preposition stranding is prohibited. This is illustrated by the sentences below.

20. Carl stood near something, but I don't know near what.
21. Carl stood near something, but I don't know what.
22. Carl stood, but I don't know near what.
23. *Carl stood, but I don't know what.

The ban on preposition stranding in sprouting follows directly from the characterization of ellipsis antecedents in terms of syntactic contexts. Preposition stranding can only obtain when the preposition itself is part of the antecedent. In cases of sprouting, there is no preposition in the antecedent, and thus the ellipsis site cannot contain a preposition (i.e. it must surface). From the perspective of the theory advanced here, this is a very general and very straightforward prediction. The categories of overt expressions in an ellipsis site must match with the type of the available antecedent contexts. A context that wants a PP cannot be used as the antecedent of an ellipsis site that has a DP argument. The reason that, in non-sprouting cases, one can either have a PP or a DP is that the antecedent contains a PP, and so there are available two possible antecedent

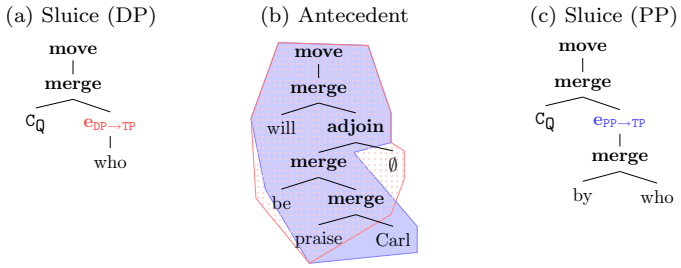
[25]The theory in Kobele (2012c) explains how this might work.

Figure 27: Carl was praised by someone, but I don't know (by) who(m)



contexts – one that contains the P, and one that doesn't. This is illustrated in figure 27. Here the red and blue ellipsis sites have the same types as the red and blue contexts. (Note that there is an additional possible context for the red ellipsis site, namely □ *was praised by someone*; see the discussion surrounding figure 22.) The reason that the same freedom does not exist in sprouting contexts is because the antecedent does not contain a (full) PP (but only a *pro*). There is thus no antecedent context containing a P-head. This is illustrated in figure 28. The only possible interpretation of the ellipsis site in red is via the context □ *was praised* (see the discussion of figure 22).

Figure 28: Carl was praised, but I don't know (by) who(m)



4.3.1. The preposition stranding generalization

Merchant (2001) observes that not all languages allow for preposition stranding in sluicing, and that there is a strong correlation between whether a language allows preposition stranding at all, and whether it allows preposition stranding in sluicing. A natural move, made by Merchant, is to attempt to explain this tendency by lifting it to an exceptionless universal, analysing away exceptions in some (hopefully) principled manner. The present theory of sluicing can derive this exceptionless generalization, as long as optional pied-piping is resolved internal to the PP by some sort of derivational process (that is, whether a PP will be pied-piped or not is determined before the PP is merged

into the larger structure).[26]

One way to implement this is along the lines of Cable (2010), according to (my reading of) which a functional projection of P agrees with the *wh*-feature of its complement DP, checking it, and which has its own *wh*-feature. In the present system, this can be implemented with the lexical item in figure 29. This functional head first selects a PP

$$\langle \epsilon, \texttt{*p} \circledast \texttt{wh p wh} \rangle$$

Figure 29: Lexical Entry for Pied-Piping

(`*p`), then checks a *wh*-feature covertly (⊛`wh`), and is then selectable as a PP (`p`) and has a *wh*-feature (`wh`).

Now examine the following English sentences.

24. Carl stood under something, but I don't know under what.

25. Carl stood under something, but I don't know what under.

26. Carl stood under something, but I don't know what.

Sentence 25 is a *swiping* construction (Merchant, 2002). Sentences 24 and 26 we have already seen, although we must revisit their structure in light of our focus on pied-piping. What will be crucial for this analysis is that the PP in the antecedent, *under something*, is *not* a pied-piping structure. That is, whatever process a PP undergoes to be the target of later *wh*-movement (in the present analysis, it is the merger of the lexical item in figure 29), *under something* did *not* undergo that. We now continue with an analysis of 24 and 25. Here, the relevant antecedent context *excludes* the PP, and so we can *either* put a pied-piping PP into the ellipsis site which takes this context as antecedent and derive 24, *or* put a non-pied-piping PP into the ellipsis site and derive 25. For 25, the structure $\mathsf{e}(\mathsf{under\ what})$ has type $\langle \texttt{t}, \{\texttt{wh}\} \rangle$, except that the moving element is the *wh*-word what, and not the entire PP *under what*. Once a *wh*-C is introduced, and the *wh*-word what is moved to its specifier, the desired word order is obtained. This analysis of 25 predicts that swiping should be possible in sprouting sentences, which seems to be the case (sentence 27).

27. Carl built a tower, but I don't know what with.

Now consider what could happen were English a language with obligatory pied-piping. In this case, there would be only one derivation of the PP *under what*, which forces under to pied-pipe to Spec-CP. Sentence 24 is still possible. But the swiped sentence 25 is no longer derivable for the simple reason that there is no non-pied-piping version of *under what*. What about sentence 26? In order to make any sort of determination about this case we need to improve our understanding of just how obligatory pied-piping works. We could implement obligatory pied-piping in the present analysis by making use of derivational constraints (Graf, 2011; Kobele, 2011). In this case, the type of the ellipsis

[26] What this is meant to exclude is the 'feature percolation' implementation of Kobele (2005), which makes no derivational distinction between pied-piping and non-pied-piping contexts.

site $c \rightarrow c'$ would indicate that the input argument of type c must not be something that would normally require pied-piping.

It is worth reiterating that the account of this half of Merchant's generalization, from obligatory pied-piping to lack of preposition stranding in sluicing, is highly dependent upon a particular array of (non-necessary) assumptions about obligatory pied-piping. The other half, from optional pied-piping to preposition strandability in sluicing, follows more straight-forwardly. Considering the space of analytical possibilities, this leads us as linguists to reserve more subjective degrees of belief to the possibility that a language might exist which has obligatory pied-piping but allows preposition stranding in sluicing, than to the possibility that a language might have optional pied-piping yet prohibit preposition stranding in sluicing.

An apparently correct prediction of the present theory is that swiping should never be possible in a language with obligatory pied-piping. Another prediction is that swiping should be grammatically possible whenever a language has optional pied-piping (which is determined PP-internally) and an ellipsis operation of type $\texttt{PP} \rightarrow \texttt{TP}$. This is counter-exemplified by Frisian, Icelandic, and Swedish, which have optional pied-piping, and an appropriately typed ellipsis operation, but which do not allow swiping (in fact, only Danish, English, and Norwegian allow for swiping). As also noted by Merchant (2002), swiping is (mostly) limited to monomorphemic *wh*-phrases, which is also not predicted by this approach. Although the proper analysis of swiping is orthogonal to the account of Merchant's generalization, it is suggestive that swiping emerges so naturally from the mechanisms already in place. In order to rein in the above-described overgeneration of the present theory, one might postulate that there is an additional, perhaps prosodic, constraint which must be satisfied in order for swiping to obtain.

5. Conclusion

In the preceding sections, a general theory of ellipsis was introduced, and an analysis of certain English constructions in the context of this theory was given. It was shown that, within a certain range of possible analyses of particular constructions, only a restricted class of elliptical sentences would be derivable, thereby accounting for certain typological generalizations. The theory presented herein is strongly related to the theory of Kobele (2009), and embodies the same ideas as that of Barker (2013). All three theories take derivational structure as basic, but Kobele (2009) and Barker (2013) restrict the kinds of antecedents to be derivational constituents. This is possible because these latter two theories have extremely flexible notions of derivations; Kobele (2009) uses a version of late-merger, and Barker (2013) has access to hypothetical reasoning — these operations allow for the constituentification of what are for the theory in this paper contexts. This means, however, that in these other two theories there is a great deal of seemingly spurious ambiguity, which needs to be resolved to determine whether a particular expression can be an antecedent for an ellipsis site. This would seem to preclude the same expression serving simultaneously as an antecedent for different types of elliptical constructions, as sketched schematically in 28.

28. Antecedent. VPE. Gapping.

29. This story should have been made public. Most magazines chose not to $\mathbf{e}_{\texttt{VP}}$. Only Gala did $\mathbf{e}_{\texttt{DP}\rightarrow\texttt{VP}}$ the particularly juicy bits.

Discourse 29 instantiates the schema in 28. It sounds to my ear a bit strained, but the predicted grammaticality of this sort of discourse is a formal difference between the present, derivational context-based, theory of ellipsis, and the other two, derivational constituent-based, ones.

Many influential topics remain to be considered. Two will be briefly mentioned here. (1) Antecedent contained deletion, as in a sentence like *Sebastian examined every patient Wolf did*, poses a problem if the antecedent contains the ellipsis site. In the present theory, the antecedent can be chosen to be the context *examine* $\Box$ of type $\texttt{DP} \rightarrow \texttt{VP}$, which simply excludes the DP *every patient Wolf did* containing the ellipsis site. A moment's reflection will reveal this to be 'the same idea' as moving the DP containing the ellipsis site out of the desired antecedent, but recast in a theory which treats the derivation as the only relevant level of syntactic structure, and thus prohibits destructive modifications of already built syntactic structure. (2) Since Merchant (2001), much work has been devoted to explaining why island effects do not always appear in elliptical sentences, especially under the assumption that the ellipsis site houses unpronounced syntactic structure. A main idea of this line of work has been that ellipsis repairs (some) islands. This idea requires a particular perspective on (reparable) islands: they must be syntactically derivable. An island is, under this view, a filter on representations, which ignores, for whatever reason, anything which has been elided. This idea can be imported into the derivational theory of ellipsis unchanged. It may however appear that the derivational theory is unable to enforce the non-reparable island constraints. This is not the case. The minimalist grammar formalism incorporates a hard constraint on movement, the SMC, which makes certain derivations non-convergent. The type assignment system presented herein does not assign types to contexts which would violate the SMC: a context has types $\texttt{type}(i) \rightarrow \texttt{type}(C[i])$, which is undefined at a given i if $C[i]$ violates the SMC. This is not an arbitrary decision, but is necessary in order to maintain a correct link between the type of a context and the meaning associated with it (Kobele, 2012d). Thus, islands whose source is a hard grammatical constraint can and must be represented by the syntactic type of an antecedent. An island may be irreparable without necessarily being reducible to a hard grammatical constraint, however. Two reductionist approaches to islands reduce them to some property of i) the semantic representation or value of an expression (Szabolcsi and Zwarts, 1992), or ii) the human sentence processor (Kluender, 1992). Island effects of the first type would also not be reparable in the present system (or presumably in any other). Those of the second type, however, are more interesting to speculate about in the context of the derivational theory here; more than speculation would require a precise theory of parsing elliptical sentences, which has not been presented in this paper. A predictive parser for the derivational theory of ellipsis would, upon predicting an ellipsis site, retrieve from the discourse context an antecedent of the appropriate syntactic type, and interpret the ellipsis site using the meaning of the antecedent. There is therefore no reason to expect that islands based on the difficulty of parsing the non-elliptical version of a sentence would appear in an elliptical sentence.

More generally, the present work can be seen as addressing a more refined version of the debate about the syntactic representation of ellipsis sites. Instead of a boolean 'is there structure or not,' here the focus has been on *how much* structural information is needed to account for the relevant linguistic phenomena. One aim of this paper has been to show that information about syntactic type is already sufficient. Another point of dispute in the literature is whether the relation between ellipsis site and antecedent is

syntactic or semantic. The present theory has it that the meaning of an ellipsis site is the same as the meaning of its antecedent, and thus it is on the semantic side of this divide. On the other hand, possible antecedents must have a certain syntactic property (having the same syntactic type) in order to be eligible for antecedence. Thus antecedents in the derivational theory are *semantic objects* which are *characterized syntactically*. This is in contrast to theories like that of Dalrymple et al. (1991) or Hardt (1993), in which an antecedent is a piece of a larger semantic representation, whose connection to syntax has been long since lost. From the derivational perspective, the syntax sensitivity of ellipitical processes comes not from reconstructing a syntactic structure, but rather from a strong syntactic filter on semantic antecedents. The fundamental claim of the derivational perspective is that there is only a fixed finite amount of syntactic information (here encoded as a syntactic type) to which elliptical processes need refer.

Acknowledgments

I am grateful for the criticism of an anonymous reviewer, to Jason Merchant for comments on an earlier draft, to John Hale and Tim Hunter for discussion, as well as to audiences at the ELLIPSIS conference in Vigo, HPSG 19, and the Identity in Ellipsis conference in Leiden, who have seen various incarnations of the ideas contained herein. I am also grateful to Marcus Kracht, Uwe Mönnich, and Ed Stabler, for their prescient suggestions at a much earlier stage of this work. As always, the responsibility for the shortcomings of this paper lies with me.

References

References

Adger, D., 2003. Core Syntax. Oxford University Press.

Bach, E. W., 1980. In defense of passive. Linguistics and Philosophy 2, 297–341.

Baker, M., 1988. Incorporation: a theory of grammatical function changing. MIT Press, Cambridge, Massachusetts.

Barker, C., 2013. Scopability and sluicing. Linguistics and Philosophy 36 (3), 187–223.

Beesley, K. R., Karttunen, L., 2003. Finite State Morphology. CSLI Publications.

Brody, M., 2000. Mirror theory: Syntactic representation in perfect syntax. Linguistic Inquiry 31 (1), 29–56.

Cable, S., 2010. The Grammar of Q. Oxford University Press.

Chomsky, N., 1995. The Minimalist Program. MIT Press, Cambridge, Massachusetts.

Chung, S., Ladusaw, W. A., McCloskey, J., 1995. Sluicing and logical form. Natural Language Semantics 3 (3), 239–282.

Clark, A., Giorgolo, G., Lappin, S., 2013. Statistical representation of grammaticality judgements: the limits of n-gram models. In: Proceedings of the Fourth Annual Workshop on Cognitive Modeling and Computational Linguistics (CMCL). Association for Computational Linguistics, Sofia, Bulgaria, pp. 28–36.

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Tison, S., Tommasi, M., 2002. Tree automata techniques and applications, available at `http://www.grappa.univ-lille3.fr/tata`.

Dalrymple, M., Shieber, S. M., Pereira, F. C. N., 1991. Ellipsis and higher-order unification. Linguistics and Philosophy 14 (4), 399–452.

de Groote, P., Pogodalla, S., 2004. On the expressive power of Abstract Categorial Grammars: Representing Context-Free formalisms. Journal of Logic, Language and Information 13 (4), 421–438.

Fowlie, M., 2014. Adjuncts and minimalist grammars. In: Morrill, G., Muskens, R., Osswald, R., Richter, F. (Eds.), Formal Grammar. Vol. 8612 of Lecture Notes in Computer Science. Springer, pp. 34–51.

Fox, D., 2002. Antecedent-contained deletion and the copy theory of movement. Linguistic Inquiry 33 (1), 63–96.
Frey, W., Gärtner, H.-M., 2002. Scrambling and adjunction in minimalist grammars. In: Jäger, G., Monachesi, P., Penn, G., Wintner, S. (Eds.), Proceedings of Formal Grammar 2002. pp. 41–52.
Graf, T., 2011. Closure properties of minimalist derivation tree languages. In: Pogodalla, S., Prost, J.-P. (Eds.), LACL 2011. Vol. 6736 of Lecture Notes in Artificial Intelligence. pp. 96–111.
Halle, M., Marantz, A., 1993. Distributed morphology and the pieces of inflection. In: Hale, K., Keyser, S. J. (Eds.), The View from Building 20. MIT Press, Cambridge, Massachusetts, pp. 111–176.
Hankamer, J., Sag, I. A., 1976. Deep and surface anaphora. Linguistic Inquiry 7 (3), 391–428.
Hardt, D., 1993. Verb phrase ellipsis: Form, meaning, and processing. Ph.D. thesis, University of Pennsylvania.
Harkema, H., 2001. Parsing minimalist languages. Ph.D. thesis, University of California, Los Angeles.
Hunter, T., 2010. Relating movement and adjunction in syntax and semantics. Ph.D. thesis, University of Maryland.
Jaeggli, O. A., 1986. Passive. Linguistic Inquiry 17 (4), 587–622.
Jäger, G., 2005. Anaphora and Type Logical Grammar. Vol. 24 of Trends in Logic. Springer, Dordrecht, The Netherlands.
Joshi, A. K., 1985. Tree adjoining grammars: How much context-sensitivity is required to provide adequate structural descriptions. In: Dowty, D., Karttunen, L., Zwicky, A. (Eds.), Natural Language Processing: Theoretical, Computational and Psychological Perspectives. Cambridge University Press, NY, pp. 206–250.
Joshi, A. K., 1987. An introduction to tree adjoining grammars. In: Manaster-Ramer, A. (Ed.), Mathematics of Language. John Benjamins, Amsterdam, pp. 87–115.
Keenan, E., 1980. Passive is phrasal not (sentential or lexical). In: Hoekstra, T., van der Hulst, H., Moortgat, M. (Eds.), Lexical Grammar. Vol. 3 of Publications in Language Sciences. Foris Publications, Dordrecht, Ch. 7, pp. 181–214.
Kehler, A., 2002. Coherence, Reference, and the Theory of Grammar. No. 104 in CSLI Lecture Notes. CSLI Publications, Stanford.
Kepser, S., Rogers, J., 2011. The equivalence of Tree Adjoining Grammars and monadic linear context-free tree grammars. Journal of Logic, Language and Information 20, 361–384.
Kertz, L., 2010. Ellipsis reconsidered. Ph.D. thesis, University of California, San Diego.
Kim, C. S., Kobele, G. M., Runner, J. T., Hale, J. T., 2011. The acceptability cline in VP ellipsis. Syntax 14 (4), 318–354.
Kluender, R., 1992. Deriving island constraints from principles of predication. In: Goodluck, H., Rochemont, M. (Eds.), Island constraints: theory, acquisition, and processing. Vol. 15 of Studies in Theoretical Psycholinguistics. Kluwer Academic Publishers, Dordrecht, NL, Ch. 8, pp. 223–258.
Kobele, G. M., 2002. Formalizing mirror theory. Grammars 5 (3), 177–221.
Kobele, G. M., 2005. Features moving madly: A formal perspective on feature percolation in the minimalist program. Research on Language and Computation 3 (4), 391–410.
Kobele, G. M., 2006. Generating copies: An investigation into structural identity in language and grammar. Ph.D. thesis, University of California, Los Angeles.
Kobele, G. M., 2009. Syntactic identity in survive minimalism: Ellipsis and the derivational identity hypothesis. In: Putnam, M. T. (Ed.), Towards a purely derivational syntax: Survive-minimalism. John Benjamins, pp. 195–230.
Kobele, G. M., 2011. Minimalist tree languages are closed under intersection with recognizable tree languages. In: Pogodalla, S., Prost, J.-P. (Eds.), LACL 2011. Vol. 6736 of Lecture Notes in Artificial Intelligence. pp. 129–144.
Kobele, G. M., 2012a. Eliding the derivation: A minimalist formalization of ellipsis. In: Müller, S. (Ed.), Proceedings of the 19th International Conference on Head-Driven Phrase Structure Grammar, Chungnam National University Daejeon. CSLI Publications, Stanford, pp. 307–324.
Kobele, G. M., 2012b. Ellipsis: computation of. WIREs Cognitive Science 3 (3), 411–418.
Kobele, G. M., 2012c. Idioms and extended transducers. In: Proceedings of the Eleventh International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+11), *Paris*. pp. 153–161.
Kobele, G. M., 2012d. Importing montagovian dynamics into minimalism. In: Béchet, D., Dikovsky, A. (Eds.), Logical Aspects of Computational Linguistics. Vol. 7351 of Lecture Notes in Computer Science. Springer, Berlin, pp. 103–118.
Koizumi, M., 1995. Phrase structure in minimalist syntax. Ph.D. thesis, Massachusetts Institute of Technology.
Koopman, H., Sportiche, D., 1991. The position of subjects. Lingua 85, 211–258.

Lasnik, H., 1981. Restricting the theory of transformations: A case study. In: Hornstein, N., Lightfoot, D. (Eds.), Explanations in Linguistics: The logical problem of language acquisition. Longmans, London, pp. 152–173.
Lavelli, A., Stock, O., 1990. When something is missing: ellipsis, coordination and the chart. In: Proceedings of COLING-90. pp. 184–189.
Lobeck, A., 1995. Ellipsis: Functional Heads, Licensing, and Identification. Oxford University Press, New York.
Luo, Z., 1994. Computation and Reasoning: A Type Theory for Computer Science. Oxford University Press.
Martin-Löf, P., 1984. Intuitionistic Type Theory. Bibliopolis, Napels.
Merchant, J., 2001. The Syntax of Silence: Sluicing, Islands, and the Theory of Ellipsis. Vol. 1 of Oxford Studies in Theoretical Linguistics. Oxford University Press, New York.
Merchant, J., 2002. Swiping in Germanic. In: Zwart, C. J.-W., Abraham, W. (Eds.), Studies in Germanic Syntax. John Benjamins, Amsterdam, pp. 289–315.
Merchant, J., 2004. Fragments and ellipsis. Linguistics and Philosophy 27 (6), 661–738.
Merchant, J., 2008. Variable island repair under ellipsis. In: Johnson, K. (Ed.), Topics in Ellipsis. Oxford University Press, Oxford, Ch. 6, pp. 132–153.
Merchant, J., 2013. Voice and ellipsis. Linguistic Inquiry 44 (1), 77–108.
Michaelis, J., 2001. On formal properties of minimalist grammars. Ph.D. thesis, Universität Potsdam.
Müller, G., 2010. On deriving CED effects from the PIC. Linguistic Inquiry 41 (1), 35–82.
Plank, F. (Ed.), 1995. Double Case: Agreement by Suffixaufnahme. Oxford University Press.
Potsdam, E., 1997. English verbal morphology and VP ellipsis. In: Proceedings of the 27th Meeting of the North East Linguistic Society. Amherst, Massachusetts, pp. 353–368.
Rooth, M., 1992. Ellipsis redundancy and reduction redundancy. In: Berman, S., Hestvik, A. (Eds.), Proceedings of the Stuttgart Ellipsis Workshop. No. 29 in Arbeitspapiere des SFB 340.
Ross, J. R., 1969. Guess who? In: Binnick, R., Davison, A., Green, G., Morgan, J. (Eds.), Papers from the Fifth Regional Meeting of the Chicago Linguistics Society (CLS). Vol. 5. Chicago, pp. 252–286.
Sag, I. A., 1976. Deletion and logical form. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.
Salvati, S., 2011. Minimalist grammars in the light of logic. In: Pogodalla, S., Quatrini, M., Retoré, C. (Eds.), Logic and Grammar: Essays Dedicated to Alain Lecomte on the Occasion of his 60th Birthday. Vol. 6700 of Lecture Notes in Artificial Intelligence. Springer, pp. 81–117.
SanPietro, S., Xiang, M., Merchant, J., 2012. Accounting for voice mismatch in ellipsis. In: Arnett, N., Bennett, R. (Eds.), Proceedings of the 30th West Coast Conference on Formal Linguistics. Cascadilla Proceedings Project, Somerville, MA, pp. 303–312.
Stabler, E. P., 1997. Derivational minimalism. In: Retoré, C. (Ed.), Logical Aspects of Computational Linguistics. Vol. 1328 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, pp. 68–95.
Stabler, E. P., 2011. Computational perspectives on minimalism. In: Boeckx, C. (Ed.), The Oxford Handbook of Linguistic Minimalism. Oxford Handbooks in Linguistics. Oxford University Press, New York, Ch. 27, pp. 616–641.
Stabler, E. P., 2013. Two models of minimalist, incremental syntactic analysis. Topics in Cognitive Science 5 (3), 611–633.
Stump, G. T., 2001. Inflectional Morphology: A Theory of Paradigm Structure. Cambridge University Press.
Szabolcsi, A., Zwarts, F., 1992. Weak islands and an algebraic semantics for scope taking. Natural Language Semantics 1, 235–284.
Tanaka, H., 2011. Voice mismatch and syntactic identity. Linguistic Inquiry 42 (3), 470–490.
Yoshida, M., 2010. Antecedent contained sluicing. Linguistic Inquiry 41 (2), 348–356.