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
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 linguistic 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 ellipsis resolution is sensitive to purely syntactic properties of antecedents.2

---

1In this paper, the diacritic % will be used to indicate unacceptability, and * to indicate ungrammaticality. To say that a sentence is unacceptability 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.

2While 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 (Jaeger, 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)), 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 \textit{he} stood.
7. John stood under something, but I don’t know what \textit{he} stood under.

According to Merchant’s generalization, given that 7 is grammatical, we should (correctly) conclude that English allows preposition stranding under regular \textit{wh}-movement. Conversely, if the sentences above were from a language about which we know only that it allows for preposition stranding under regular \textit{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, \textit{sprouting} is the name of the transformation which inserts a trace (which can be then bound by the \textit{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

---

4Grammar 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 impossible. 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.

5A 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 merge(A, B) = {A, B}, and move(A) = 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.
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 organized into a feature bundle, which is simply a list of features. Lexical items are written using the notation $\langle w, \delta \rangle$, where $w$ is a lexeme, and $\delta$ 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 $\langle w, \delta \rangle$). 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 \textit{merge} and \textit{move}. Given the lexicon above, (all and only) the following additional expressions can be generated:

i. 

\text{merge}(⟨run, *d v⟩, ⟨carl, d k⟩):

\begin{center}
\begin{tikzpicture}
\node (run) {⟨run, v⟩} ;
\node (carl) {⟨carl, k⟩} ;
\node (merge) {\langle run, v \rangle \langle carl, k \rangle} ;
\path (run) -- node {$<$} (merge) ;
\path (carl) -- node {$<$} (merge) ;
\end{tikzpicture}
\end{center}

Because \textit{run}'s feature bundle begins with a \textit{*d}, and \textit{carl}'s begins with a matching \textit{d}, \textit{merge} can put them together as per the tree above. Note that both \textit{*d} and \textit{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 \textit{<}, which ‘points’ left, the head is in the left daughter subtree, which in this case is simply the leaf \textit{run}.

ii. 

\text{merge}(⟨will, *v *k t⟩, i): 

\begin{center}
\begin{tikzpicture}
\node (will) {⟨will, *k t⟩} ;
\node (run) {⟨run, t⟩} ;
\node (carl) {⟨carl, k⟩} ;
\node (merge) {\langle will, *k t \rangle \langle run, t \rangle} ;
\path (will) -- node {$<$} (merge) ;
\path (run) -- node {$<$} (merge) ;
\path (carl) -- node {$<$} (merge) ;
\end{tikzpicture}
\end{center}

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

iii. 

\text{move}(ii):

\begin{center}
\begin{tikzpicture}
\node (will) {⟨will, t⟩} ;
\node (run) {⟨run, t⟩} ;
\node (carl) {⟨carl, t⟩} ;
\node (move) {\langle will, t \rangle \langle run, t \rangle} ;
\path (will) -- node {$<$} (move) ;
\path (run) -- node {$<$} (move) ;
\path (carl) -- node {$<$} (move) ;
\end{tikzpicture}
\end{center}

Finally, \textit{move} attracts \textit{carl} to a projection of \textit{will}, driven by their respective \textit{k} and \textit{*k} features, which are then eliminated. This expression is a projection of \textit{will}, as can be seen by following from the root the path indicated by the arrowheads, which has just a single feature \textit{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.

\footnote{The root (labeled by \textgreater) points to the right, the right-hand daughter of the root (labeled by \textless) points to the left, the left-hand daughter of which is \textit{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 ($x, *y, \ldots$). 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. 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, and is written \[\text{type}(t) = \langle \alpha, A \rangle,\] where $\alpha$ 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 $\langle w, \delta \rangle$, \[\text{type}(\langle w, \delta \rangle) = \langle \delta, \emptyset \rangle.\] In the expressions derived above, \[\text{type}(i) = \langle v, \{k\} \rangle, \text{type}(ii) = \langle *k \ t, \{k\} \rangle,\] and \[\text{type}(iii) = \langle t, \emptyset \rangle.\]

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 $\langle w, \delta \rangle$ is a special case of a tree with only a single node). Each node in an expression is a projection of the unique leaf one arrives at by following the arrows down the tree. A
node is a maximal projection of a leaf \( \ell \) just in case it is a projection of \( \ell \) 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 \( \langle w, \delta \rangle[h] \).)

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 \( *x \) or \( \Box x \), 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 \( t_2 \) 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.

\[
\text{merge}(\langle \alpha, *x \delta \rangle, t_2[[\beta, x \gamma]]) = \begin{cases} \langle t_1[[\alpha, \delta]], \beta \rangle & \text{(} t_1 \text{ is a lexical item)} \\ \langle t_2[[\beta, \gamma]], t_1[[\alpha, \delta]] \rangle & \text{(} t_1 \text{ is not a lexical item)} \end{cases}
\]

The \( \Box x \) 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.

\[
\text{merge}(\langle -\alpha, \Box x \delta \rangle, t_2[[\beta, x \gamma]]) = \langle \beta-\alpha, \delta \rangle \quad \langle \beta, \gamma \rangle \quad \text{t}_2[[\beta, \gamma]]
\]

2.1.2. Move

The operation move applies to a single tree \( t[[\alpha, \bullet y \delta]] \) (where \( \bullet y \) is either \( *y \) or \( \Box y \)) only if there is exactly one leaf \( \ell \) in \( t \) with matching first feature \( y \). 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 \( \ell \) to a newly created specifier position in \( t \) ( overtly, in the case of \( *y \), and covertly, in the case of \( \Box y \)), and deletes both licensing features. To make this precise, let
denote the result of replacing all subtrees \( t_1 \) in \( t \) with \( t_2 \), for any tree \( t \), and let \( t^M_\ell \) denote the maximal projection of \( \ell \) in \( t \), for any leaf \( \ell \).

\[
\text{move}(t[(\alpha, \cdot)\delta]) = t'[(\beta, \gamma)] \xrightarrow{\{t' \mapsto (\epsilon, \epsilon)\}} (\text{where } t' = (\beta, \gamma)_t^M)
\]

\[
\text{move}(t[(\alpha, @)\delta]) = (\epsilon, \gamma) \xrightarrow{\{t[\alpha, \delta]\}t' \mapsto t'[\beta, \epsilon]\}} (\text{where } t' = (\beta, \gamma)_t^M)
\]

An expression is \textit{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 \( \gamma \) feature, then if at any point a tree with multiple \( \gamma \) heads is derived, then the \textit{move} operation can never check these \( \gamma \) 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 \( \gamma \) feature for every feature type \( \gamma \). This means that in the type of an expression \( t \), \( \text{type}(t) = (\alpha, A) \), 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 \( \gamma \), it is the only feature bundle which does; thus \( A \) can also be thought of as a \textit{partial function} from feature types \( \gamma \) 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 lexical items in figure 4 (adapted from Kobele (2006)). A transitive verb, here \textit{praise},

\[
\left[ (\text{will, \textit{*v \textit{*k t)}} \quad \langle -\epsilon, @\text{agrO \textit{*d v} \rangle} \quad \langle -\epsilon, @\text{V \textit{*k agrO} \rangle} \quad \langle \text{praise, \textit{*d V} \rangle} \right]
\]

Figure 3: Basic clause-structure

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 \( \langle -\epsilon, \oplus V \odot k \text{ agrO} \rangle \), 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 \( \langle -\epsilon, \oplus \text{agrO} \odot *d v \rangle \), 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 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 (\( \oplus k \)) and the latter is overt (\( \ast k \)). 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.

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 \( \epsilon \) 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).
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 move 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. 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 type(t) = 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, spellOut(α) 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 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 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 $\text{merge}$ and $\text{move}$). Grammatical operations are functions over expressions. The operation $\text{merge}$ is a binary operation (i.e. it takes two arguments), $\text{move}$ 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 constrains the possible definitions of $e$: it must be something that $Oda$ can be merged with, giving rise to something which $\text{will}$ 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 $v'$. 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 distinguished (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 $\tau$, there is an ellipsis operation $e_\tau$ at that type. In these terms, the ellipsis operation underlying the sentence $Oda \text{ will, too}$ is $e_\tau$. 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

```
move
l
merge

(will, *v *k t)
merge

\(e\)

(oda, d k)
```

Figure 10: The structure of an elliptical sentence
the object derived by $e$ as the same as the object derived by its antecedent, modulo pronounced material; loosely speaking, $\text{spellOut}(e_v') = \text{delete(antecedent}(e_v'))$.$^{14}$

This idea, though simple, misses the obvious (but important) point that, no matter what the antecedent, $\text{spellOut}(e_v')$ 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 stipulated, 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 $\gamma$, and no phonological content, and (ii) it contains $n$ silent moving subparts, with features $\beta_1$ through $\beta_n$. For the special case of $e_v'$, we have that $\text{spellOut}(e_v') = \langle \epsilon, \ast d v \rangle$ (recall that $v'$ is an abbreviation for the syntactic type $\langle \ast d v, \emptyset \rangle$).

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 $\text{merge}$ node immediately above oskar, we would have an ellipsis node of type $\langle V, \{k\} \rangle$, abbreviated as $\text{VP}$. In this case the sentence $Oda$ will, too would

\[
\text{spellOut}(e_{\langle \gamma, \{\beta_1, \ldots, \beta_n\}\rangle}) = \langle \epsilon, \beta_1 \rangle = \langle \epsilon, \beta_2 \rangle = \cdots = \langle \epsilon, \beta_n \rangle = \langle \epsilon, \gamma \rangle
\]

Figure 11: The spellout of ellipsis operations

$^{14}$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,*

\[
\begin{array}{c}
\text{move} \\
\text{merge}
\end{array}
\begin{array}{c}
\langle \text{will, } \ast v \ast k \ t \rangle \\
\langle \text{oda, } d k \rangle
\end{array}
\begin{array}{c}
\text{merge} \\
\langle \epsilon, \ast \text{agro} \ast d \ v \rangle
\end{array}
\begin{array}{c}
\text{move} \\
\text{merge}
\end{array}
\begin{array}{c}
\langle \epsilon, \ast v \ast k \ \text{agro} \rangle
\end{array}
\begin{array}{c}
\epsilon\text{VP}
\end{array}
\begin{array}{c}
\text{spellOut}(\epsilon\text{VP}) = (\epsilon, k)(\epsilon, v)
\end{array}
\]

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. While there are some basic structural differences between e′ and eVP, such as that only the latter represents the ellipsis of a maximal projection, a more fundamental difference is that eVP 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. 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 ellipsis operation shown in figure 11 permits, using a deletion metaphor, movement of expressions which have been deleted (those with feature bundles βi). 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.

---

15Kim et al. (2011) suggest that both options be allowed.
16This 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 determining 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 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 \( \diamond x \)) 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 \((\diamond y)\).

This stipulation accounts for the ungrammaticality of the derivation in 13 because the elliptical sub-piece \( (\epsilon, k) \) has its \( k \) feature checked by the \( *k \) feature of \textit{will}. An equivalent way of putting this views \( *k \) as the combination of \( \diamond k \) 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

\(\text{move} \quad l \quad \text{merge} \quad (\text{will, } *v \quad *k \quad t) \quad \text{merge} \quad (\text{be, } *\text{pass v}) \quad \text{merge} \quad (\text{praise, } *d \quad V) \quad \epsilon \text{VP} \quad > \quad (\epsilon, ) \quad \diamond \quad (\text{will, t}) \quad < \quad (\text{be, )} \quad \diamond \quad (\text{praise, } *d \quad V) \quad \epsilon \text{VP} \quad > \quad (\epsilon, ) \quad \diamond \quad (\text{oskar, d k}) \)

*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), $\alpha$-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 (□) represents a missing piece, which, in its occurrence 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

\[
\text{merge} \quad \langle \text{praise}, \ast d \ V \rangle \quad \square
\]

Figure 15: A unary derivation tree context
to unary contexts (contexts with just one hole) in this paper.\footnote{A more general solution (Kobele, 2012b) involves treating trees as $\lambda$-terms, and contexts as $\lambda$-abstracts. For example, the tree in figure 14 can be represented as the first-order $\lambda$-term move(merge(will))(merge(be)(merge(en)(merge(praise)(carl)))�), and the context in figure 15 as the second-order $\lambda$-term $\lambda x.\text{merge(praise)}(x)$. Restricting attention to contexts amounts to, in the more general setting of the $\lambda$-calculus, limiting our attention to terms of order at most two.} 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 $c' \rightarrow c$. More concisely, if $C$ is a context, and $i$ is an input such that type$(C[i])$ is defined, then $C$ has type $i \rightarrow \text{type}(C[i])$. The context in 15 has type $(d \ k, \emptyset) \rightarrow (V, \{k\})$, or, in abbreviated form, $\text{DP} \rightarrow \text{VP}$; it is a procedure which, given a DP, constructs a VP.\footnote{This is not yet entirely satisfactory. Consider again our context which is a VP missing a DP. One type it should have is $(d \ k, \emptyset) \rightarrow (V, \{k\})$, i.e. something which, if you give it a DP will result in a VP. But it also has the type $(d \ k \ \text{wh}, \emptyset) \rightarrow (V, \{k \ \text{wh}\})$, 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 $\alpha$, it has the category $(d \ alpha, \emptyset) \rightarrow (V, \{alpha\})$. 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 $\forall \alpha.(d \ alpha, \emptyset) \rightarrow (V, \{alpha\})$. 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 context replaces to obtain the tree on the right of figure 14. It represents the ellipsis site. It differs from e_{DP} in that e_{DP→VP} is a unary operation, whereas e_{eP} is a nullary operation. The natural generalization is to say that e_{\tau} is a grammatical operation, which takes a number of arguments appropriate to its type \tau. 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 e_{DP→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 DP→vP = \langle d k, \emptyset \rangle → \langle v, \{ 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 e_{f} for every grammatically permissible way f of linearly ordering the moving pieces in expressions which are arguments of e_{f}. 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 \tau, there are only finitely many possible f, whereas there are (generally) infinitely many possible deleted structures.
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 delimitied 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

21Another 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.

Figure 17: The spellout of unary ellipsis operations
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 acceptable. 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.

22A 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 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 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-coordinated 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 \( DP \rightarrow vP \) in subfigure 19b are outlined in red. Inspection of the figure reveals that the undesired non-‘root’ voice mismatch, which would mean \( Oda \) will *\( \text{seem to praise Oskar} \)*, is not possible with these structures. That it is not possible with any

<table>
<thead>
<tr>
<th>Antecedent</th>
<th>Ellipsis</th>
<th>Type(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>active</td>
<td>active</td>
<td>( v' ), ( VP ), ( DP \rightarrow vP )</td>
</tr>
<tr>
<td>active</td>
<td>passive</td>
<td>( DP \rightarrow VP )</td>
</tr>
<tr>
<td>passive</td>
<td>active</td>
<td>( VP ), ( DP \rightarrow VP )</td>
</tr>
<tr>
<td>passive</td>
<td>passive</td>
<td>( DP \rightarrow VP ), ( DP \rightarrow PassP )</td>
</tr>
</tbody>
</table>

Figure 18: Types for VPE

Figure 19: Oskar will seem to be praised. *Oda will *\( \text{seem to praise Oskar} \)*
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 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.

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

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 \textit{wh}-words have the licensee feature \textit{wh}, and that a silent \textit{CP} provides an appropriate landing site. This analysis is a simple version of the canonical analysis of \textit{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 \textit{CP} with an ellipsis site. In the present analysis, a particularity of sluicing, as opposed to, say, VPE, is that the ellipsis site \textit{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 \textit{wh}-phrase which ends up moving to Spec\CP). It is clear that it must be something of the rough form \( c \rightarrow \langle t, \{ \textit{wh} \} \rangle \), as the result is a TP which contains a \textit{wh}-moving phrase. But \( c \) could be any category which introduces a \textit{wh}-movement feature; a proper \textit{wh}-DP, or a \textit{wh}-PP, or even some category which only contains a \textit{wh}-mover.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure22.png}
\caption{Oskar will be praised, *but I don’t know who \textit{will praise Oskar}}
\end{figure}

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 \textit{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.\footnote{As pointed out by Tue Trinh, the unacceptability of this sentence is not obviously related to Moore’s paradox, as changing \textit{I} to \textit{Oda} does not seem to improve matters. It is worth pointing out that changing \textit{who} to \textit{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.}

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 \textit{will be praised}.

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

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 adjjoin, and assume that it applies obligatorily wherever it can. The desired effects of the operation

---

24It 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 $c^*$ instead of $c$, and (ii) make use of the lexical items $(\epsilon, *c^* adj c)$ and $(\epsilon, *x adj)$, for every category $x$ which can function as an adjunct. Concretely, it will be assumed that vP is an obligatory adjunction site. For simplicity, the

\[ \langle \text{by, } *d \oplus \text{k P} \rangle \]

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

![Figure 24](image1.png)

(a) Antecedent

![Figure 25](image2.png)

(a) Antecedent

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

![Figure 25](image3.png)

(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 will praise Oskar**, 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.\(^{25}\)

![Diagram](image)

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 contrast 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.
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 27: Carl was praised by someone, but I don’t know (by) who(m)

(a) Sluice (DP)

(b) Antecedent

(c) Sluice (PP)

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

(a) Sluice (DP)

(b) Antecedent

(c) Sluice (PP)

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).

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

\( \langle \epsilon, *p \bowtie wh \ p \wh \rangle \)

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 have already been 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 \( \langle t, \{wh\} \rangle \) 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 straightforwardly. 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-externally) and an ellipsis operation of type \( \text{PP} \rightarrow \text{TP} \). 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 \( e_{\text{PP}} \). Only \( \text{GALA} \) did \( e_{\text{PP}} \rightarrow \text{VP} \) 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 □ of type DP → 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 derivations 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 \( \text{type}(i) \rightarrow \text{type}(C[i]) \), 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 elliptical 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


