**Abstract**

In this paper I use the formal framework of minimalist grammars to implement a version of the traditional approach to ellipsis as 'deletion under syntactic (derivational) identity', which, in conjunction with canonical analyses of voice phenomena, immediately allows for voice mismatches in verb phrase ellipsis, but not in sluicing. This approach to ellipsis is naturally implemented in a parser by means of threading a state encoding a set of possible antecedent derivation contexts through the derivation tree. Similarities between ellipsis and pronominal resolution are easily stated in these terms. In the context of this implementation, two approaches to ellipsis in the transformational community are naturally seen as equivalent descriptions at different levels: the LF-copying approach to ellipsis resolution is best seen as a description of the parser, whereas the phonological deletion approach a description of the underlying relation between form and meaning.

## 1 Introduction

In the transformational grammar community, analyses of ellipsis which involve reconstructing a syntactic structure have been proposed (Lees, 1960) and reproposed (Chung et al., 1995; Merchant, 2001; Kobele, 2009). The structure reconstructed stands in some, often syntactic, relation to some other syntactic structure, its antecedent. In conjunction with canonical transformational analyses of basic sentence structure, such as a 'phrasal' (as opposed to a 'lexical') approach to passive constructions (Jaeggli, 1986), this kind of approach to ellipsis is able to present a unified theory which neatly captures some differences between elliptical phenomena in the degree to which they are sensitive to syntactic properties of antecedents (Merchant, 2007, 2008; Tanaka, 2011).

Two mechanisms for dealing with ellipsis are prominent in today's transformational literature: deletion and copying. The first views ellipsis as a process of syntactically conditioned phonological deletion (Merchant, 2001). This approach must be complemented with an appropriate 'identity condition', which allows a phrase to be deleted just in case it is identical to some other phrase. In this approach, an ellipsis site might be assigned an arbitrarily complex syntactic structure. The second approach, more in line with perspectives in other approaches to grammar, views ellipsis as involving a process of LF-copying of an antecedent syntactic structure into a syntactically atomic empty category (Chung et al., 1995).[1]

In the context of the formal framework of minimalist grammars (Stabler, 1997), a mildly context-senstitive (Michaelis, 2001) formalization of the minimalist program (Chomsky, 1995), I use the mechanism of deletion to implement a modern

[1]It may seem that the LF-copying approach is to be preferred on the grounds that it does not require a seperate 'identity condition', and is thus more parsimonious. A more uniform perspective on the deletion and copying approaches, which renders them equally complex, is had when we view the copying approach as follows. First, we build a structure with an ellipsis site (a primitive formative). Next, we replace the ellipsis site with some complex structure, under the condition that this structure is identical to some other phrase. Thus, LF-copying is here seen as LF-insertion under identity.

version of the 'derivational identity' approach to ellipsis of Lees (1960). Borrowing ideas from Kobele (2009) and Lichte and Kallmeyer (2010) I take phonological deletion to be licensed by exact identity of derivation tree *contexts* (trees with holes at the leaves). The main advantages of this approach to deletion are (1) that it is naturally implemented in a parser (whose job it is to reconstruct derivation trees), and (2) that in conjunction with a compositional semantics (as in Kobele (2012c)) it allows derivation tree contexts to be replaced by their semantic interpretations. Furthermore, in the context of this formalization, the deletion and LF-copying approaches to ellipsis can be viewed as equivalent descriptions at different levels (in the sense of Marr (1982)): the LF-copying approach describes the algorithmic realization of the deletion approach in a parser. Thus, one of the main contributions of this paper is to demonstrate that the two main proposals regarding ellipsis in the transformational literature needn't be thought of as competitors, but can be viewed instead as equivalent descriptions of the same thing.

Moving from computation to algorithm, a natural way of implementing the licensing requirement on deletion (that, namely, an identical antecedent be present) involves passing a 'context' containing information about what antecedents are present. This can be managed by using monads (Wadler, 1992) (or continuations (Strachey and Wadsworth, 2000)) to control evaluation. In particular, different hypotheses about antecedent availability can be implemented by allowing the context information to flow in different directions (e.g. the state and the reverse state monads). Antecedent choice is made by means of a choice operator, which, as in a continuation-based treatment of pronouns (de Groote, 2006), may be made sensitive to discourse and other factors.

Thus, the formal approach to ellipsis in minimalism presented here clearly separates various empirical phenomena surrounding ellipsis – factors influencing the choice of antecedent go in the choice operator, restrictions on availability of antecedents are to be accounted for in the context passing mechanism, and which antecedents exist at all is the provenance of the syntactic analysis.

The paper is structured as follows. I begin by reviewing some of the main empirical motivations of this paper (§2). Next, I introduce minimalist grammars (§3), where I explain briefly the notion of derivation tree, and introduce an operation of deletion. Then I describe how to implement this approach in a parsing algorithm in §4. Section 5 is the conclusion.

## 2 Empirical Foundations

One of the biggest stumbling blocks to a unified theory of ellipsis (one which treats all elliptical phenomena as being the product of a single 'ellipsis' mechanism) is the fact that different 'sorts' of ellipsis have different properties (for more information see Kobele (2012a) and references therein). Most interesting to us here, as I will in fact be advocating for a theory involving exact syntactic identity, are the differences between *sluicing* (see Merchant, 2001, and references therein) and *verb phrase* *ellipsis (vpe)* (see Hardt, 1993, and references therein) with respect to the nature of the formal relation between antecedent and (supposed) ellipsis site. We focus our attention on the verbal category of voice (although Kim et al. (2011) works out a fragment in a related system allowing for mismatches along other dimensions), and in particular on whether antecedent and (supposed) ellipsis site may differ along this dimension.

## 2.1 VPE

Although it was initially thought that verb phrase ellipsis did not allow for voice mismatches between antecedent and ellipsis site (Sag, 1976), work culminating in Hardt (1993) made abundantly clear that at least *some*, corpus attested, examples of voice-mismatched vpe exist, and sound rather natural. Various psycholinguistic experiments (see Kim et al., 2011, and references therein) have further demonstrated that mismatched vpe examples are more acceptable than stereotypical ungrammatical sentences.

In such a situation, one can either decide to treat mismatching examples (of which at least some are unacceptable) as uniformly ungrammatical (Arregui et al., 2006), or to treat mismatching examples (of which at least some are acceptable) as uniformly grammatical (Kim et al., 2011). In either case, one ultimately needs to provide an account of why (adopting the first view) certain ungrammatical examples sound perfectly fine, or of why (adopting the second view) certain grammatical examples sound terrible.

Here (following Kim et al. (2011)) I treat voice-mismatched vpe as grammatical. There is no knock-down argument for this, as far as I am aware, but it seems more promising in terms of ultimately being able to explain both why people produce mismatching vpe sentences (they are grammatical; the other view must explain why people produce ungrammatical sentences), and why mismatch is not attested in sluicing (see below §2.2; the other view must explain why ungrammatical vpe sentences are acceptable and attested, but ungrammatical sluices of the 'same sort' are not).

## 2.2 Sluicing

As noted already in Merchant (2001) (see also Chung, 2006; Merchant, 2007; Tanaka, 2011), and in contrast to vpe, in (English) sluicing voice mismatches are uniformly unacceptable. To qualify this statement somewhat, there are no known sluices in English which are acceptable yet which involve voice mismatches between the antecedent and the ellipsis site.[2] As in the case of vpe, this empirical

[2]Martín González (2010) examines counterexamples to this claim in Spanish. He concludes that they all stem from underlying copular constructions, and thus that Spanish sluicing also prohibits voice mismatches. In more theory neutral terms, he observes that all acceptable examples of voice mismatched sluicing sentences in Spanish alternate with non-elliptical sentences where the sluice is replaced with a cleft, and that where this is not possible (e.g. with an active antecedent and a passive ellipsis site), the elliptical sentence is in fact unacceptable.

situation underdetermines the proper theoretical analysis; are voice mismatched sluices grammatical in English, but just hard to find? Or are they indeed ungrammatical? Because we have available to us (Merchant, 2007) a neat explanation of how voice mismatches in vpe can be grammatical, while being in sluicing ungrammatical, I choose tentatively (but following the authors cited above) to assume that the reason for the non-forthcomingness of acceptable voice-mismatched sluices is because there aren't any, and this because they are uniformly ungrammatical.

## 3 Minimalist Grammars

Minimalist grammars (Stabler, 1997) are a *mildly context-sensitive* grammar formalism (Michaelis, 2001). 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 impossible. The languages which are able to be described are all simple in a formally precise sense (Joshi, 1985), which makes it possible to build correct and efficient parsing algorithms for these grammar formalisms.[3]

A minimalist grammar has two structure building operations, binary **merge** and unary **move**, whose application to expressions is dependent on the syntactic categories of these expressions. 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**. I first describe categories, and then move on to a more detailed description of expressions, and the workings of the **merge** and **move** operations.

### 3.1 Categories

Categories are complex, as in categorial grammar, and are structured as lists of atomic *features*, which we will write as sequences $f_1 \cdots f_n$ and call feature bundles. 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, the heads of its two arguments must have matching accessible features. These features are eliminated in the derived structure which results from their merger. In the case of **move**, the head of its argument must have an accessible feature matching an accessible feature of the head of one of its subconstituents' $\Delta$. 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**,

[3]It remains, however, a programatic assumption that this sort of restrictiveness is desirable.

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 ⊖y, depending on whether the movement is overt (−y) or covert (⊖y).

## 3.2 Expressions

A lexical item is a syntactic atom. Intuitively, it represents an atomic pairing of form and meaning. Here, it consists of an index (a 'lexeme') 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 $\sigma$ is a lexeme, and $\delta$ is a feature bundle.

Complex expressions are written using the notation of Stabler (1997) for the 'bare phrase structure' trees of Chomsky (1995). 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 $\alpha$, and we say that $\alpha$ projects over $\beta$, while one of the form $[_> \alpha\ \beta]$ that its head is in $\beta$, and we say that $\beta$ projects over $\alpha$. Leaves are labeled with lexeme/feature bundle 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 we write $t$[H] to indicate this. (This means we can write lexical items $\langle\alpha, \delta\rangle$ as $\langle\alpha, \delta\rangle[\langle\alpha, \delta\rangle]$.)

## 3.3 Operations

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.

$$\mathbf{merge}(t_1[\langle\alpha, =\mathrm{x}\delta\rangle], t_2[\langle\beta, \mathrm{x}\gamma\rangle]) = \begin{array}{c} < \\ \diagup \qquad \diagdown \\ t_1[\langle\alpha, \delta\rangle] \quad t_2[\langle\beta, \gamma\rangle] \end{array}$$

$$\mathbf{merge}(t_1[\langle\alpha, \mathrm{x}=\delta\rangle], t_2[\langle\beta, \mathrm{x}\gamma\rangle]) = \begin{array}{c} > \\ \diagup \qquad \diagdown \\ t_2[\langle\beta, \gamma\rangle] \quad t_1[\langle\alpha, \delta\rangle] \end{array}$$

If the selecting tree is both a lexical item and an affix (which we 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.

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

The operation **move** applies to a single tree $t[\langle\alpha, +\text{y}\delta\rangle]$ only if there is *exactly one* leaf $\ell$ in $t$ with matching first feature $-\text{y}$ or $\ominus\text{y}$. This is 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 have 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 $-\text{y}$, and covertly, in the case of $\ominus\text{y}$), 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 $\ell_t^M$ denote the maximal projection of $\ell$ in $t$, for any leaf $\ell$.

$$\textbf{move}(t[\langle\alpha, +\text{y}\delta]) = \begin{array}{ccc} & > & \\ \diagup & & \diagdown \\ t'[\langle\beta, \gamma\rangle] & & t[\langle\alpha, \delta\rangle]\{t' \mapsto \langle\epsilon, \epsilon\rangle\} \end{array}$$

(where $t' = \langle\beta, -\text{y}\gamma\rangle_t^M$)

$$\textbf{move}(t[\langle\alpha, +\text{y}\delta]) = \begin{array}{ccc} & > & \\ \diagup & & \diagdown \\ \langle\epsilon, \gamma\rangle & & t[\langle\alpha, \delta\rangle]\{t' \mapsto t'[\langle\beta, \epsilon\rangle]\} \end{array}$$

(where $t' = \langle\beta, \ominus\text{y}\gamma\rangle_t^M$)

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

## 3.4 Derivations

A *derivation tree* is a (complete) description of how to construct an expression. (Derivation trees are presented here in the style of Kobele (2012b), which imposes useful restrictions on deletability.) A derivation tree is a labeled ordered tree with nodes labeled with lexical items, subject to the condition that the number of daughters a node with label $\ell$ has is the same as the number of positive **selection** features $\ell$ has. (The first daughter represents the first expression $\ell$ was merged with, the second daughter the second, etc.) The derivation trees which are *well-formed*–those which actually represent 'convergent' derivations—can be charactized directly (i.e. they form a regular set (Kobele et al., 2007)), and are the objects of primary concern in parsing (Harkema, 2001) and semantic interpretation (Kobele, 2012c).

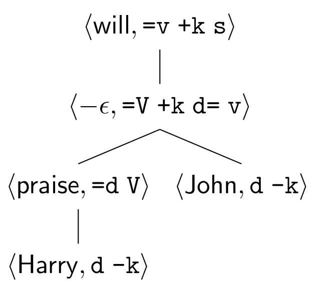

Figure 1: A derivation of the sentence “John will praise Harry”

Figure 1 presents the derivation tree for a transitive sentence given a (simplified, but fairly standard) minimalist analysis.[4] Note that each node in the tree has exactly as many daughters as it has positive selection features. For example, *will* has one positive selection feature (=v), and one daughter, whereas the ‘little-v’ head $\langle -\epsilon, \mathtt{=V\ +k\ d=\ v}\rangle$ has two positive selection features (=V and d=), and two daughters.

To determine which derived object is denoted by a given derivation tree is computationally very simple (Hale and Stabler, 2005; Kobele et al., 2007), although admittedly complex to describe intuitively. Given a derivation tree (such as in figure 1) with root $\sigma$ with features $=\mathrm{x}_1{+}\vec{\mathrm{y}}_1{=}\mathrm{x}_2 \cdots {=}\mathrm{x}_n{+}\vec{\mathrm{y}}_n\mathrm{x}{-}\vec{\mathrm{y}}$ and daughters $t_1, \ldots, t_n$, the derived expression is obtained by merging $\sigma$ with the expression denoted by $t_1$, then applying the move operation $|{+}\vec{\mathrm{y}}_1|$ times (as many times as $\sigma$ has positive licensing features between its first two positive selection features $=\mathrm{x}_1$ and $=\mathrm{x}_2$), then merging the result with $t_2$, then applying the move operation to that $|{+}\vec{\mathrm{y}}_2|$ times, etc. Essentially, the immediate dominance relation mirrors a **merge** operation, and the left-to-right order of daughters the derivational order of these **merge** operations. The **move** operation is not explicitly represented, but is uniquely determined by the features of the root.

## 3.5 Deletion

Our deletion operation targets arbitrary *connected* subparts of the derivation tree.[5] Intuitively, we want to be able to ‘draw a circle’ around a connected subpart, which indicates that this subpart is elided, and under identity with some other subpart elsewhere in the discourse. We implement this intuition by introducing two new operations, **delete**, and **elide**. **Delete** is a lexical operation (i.e. it applies to lexical

[4]The lexical items used are exactly those at the nodes of the derivation tree.

[5]This is related to the notion of *catenae* from the dependency grammar literature. As argued by (Osborne et al., 2013), canonical minimalist analyses of eliptical constructions plausibly allow the material elided to be conceived of in such a manner.

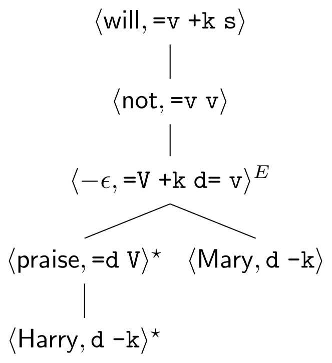

Figure 2: A derivation of the elliptical sentence “Mary will not ~~praise Harry~~”

items, not to arbitrary expressions), and we will write $\ell^\star$ instead of the more cumbersome $\mathsf{delete}(\ell)$, for $\ell$ a lexical item. The interpretation of the operation **delete** on a lexical item is simply to delete its phonological exponent. This ensures that even discontinuous non-deleted material is treated normally by the grammar. The operation **elide** delimits a stretch of deleted elements as a single eliptical unit, and can be applied only to a derivation tree whose root is deleted. It has no other effect. We write $\ell^E(t_1, \ldots, t_n)$ for the more cumbersome $\mathsf{elide}(\ell^\star(t_1, \ldots, t_n))$.[6]

A derivation tree with a deleted node can be well-formed only if this node is dominated either by another deleted node or by a node labeled elide. A derivation tree with a node labeled elide can be well-formed only if the expanse of deleted nodes ultimately licensed by this elide node is identical (modulo deletion) to some other part of this derivation tree. (Cross-sentential ellipsis must here be dealt with by taking discourse and sentence grammar to be identical, as argued for in Webber (2004).) This provides us with a direct description of the form-meaning pairs licensed by the grammar.[7]

As a concrete example, consider the derivation in figure 2. In this sentence, the elided material is the portion of derivation consisting of ‘little-v’, the verb *praise*, and its object *Harry* (all three are marked for deletion), but excluding the subject *Mary*. Note that the (local) restrictions on **delete** and **elide** are respected in this

[6]This notation is inspired by the notion of an ‘E’ feature driving ellipsis (Merchant, 2001). Note that this is not an actual feature in the present system.

[7]The ability to represent ellipsis in the grammar comes at a computational cost – the identity condition on ellipsis (that a derivation tree with a node labeled elide is well-formed only if the expanse of deleted nodes licensed by it is identical (modulo deletion) to some other part of the derivation tree) is not representable as a regular constraint, and thus the set of well-formed derivation trees in this system is no longer regular.

This is not a result of the present system, but rather of the inherent complexity of ellipsis, which must manifest itself in any system dealing with elliptical phenomena. Perhaps one useful aspect of this paper is making this complexity explicit by putting it all into the grammar.

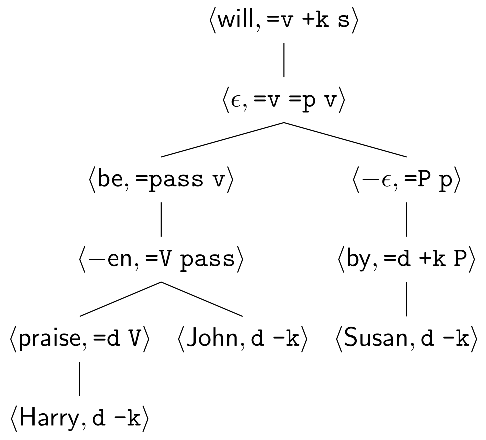

Figure 3: The derivation of "Harry will be praised by Susan"

derivation. In order for this derivation to satisfy the identity requirement on ellipsis, it must be part of a larger derivation which contains a non-elided identical subpart (such as coördinated with the derivation in figure 1).

## 3.6 Examples

Here I present some examples to illustrate both the minimalist grammar ellipsis system and the linguistic analysis. (For more details, the interested reader may consult Kim et al. (2011), which uses a related system – the analysis is identical.)

Let us examine the following sentences:

(1) Harry will be praised by Susan.

(2) Someone will be praised by Susan, but I do not know whom Susan will praise.[8]

Example 1 is presented in figure 3. Comparing the derivation trees in figure 3 and in figure 2, one sees that they share a common subpart, consisting of the verb *praise* and its argument *Harry* (but not the voice head immediately dominating *praise*). This common subpart (I'll call it the 'VP') suffices to license VP-deletion despite the mismatch in voice. (Kim et al. (2011) attempt to link the smaller identical subpart (VP versus vP) to the lower acceptability ratings assigned to sentences containing mismatched as opposed to matched vpe.)

It is easy to see that the passive example 1 has no subpart in common with an active sentence which includes both the main verb and the finite auxiliary. This

[8]This is an extremely marked sentence of English. It is included here for illustrative purposes, as it allows me to ignore complications associated with sprouting (Chung et al., 1995) and pied-piping.

simple fact blocks voice mismatches in sluicing, understood as ellipsis including the finite auxiliary and the main verb (and usually one or more arguments of the verb, as well). Example 2 does not permit ellipsis of the part *Susan will praise*; the present analysis is straightforwardly able to account for this.

## 4 Computation

I adopt a 'levels' approach to understanding complex information processing systems (a seminal work in this area is Marr, 1982). Our abilities to use language can be viewed in this context as systems for transforming sounds to meanings and vice versa. To fully understand such a system, we need describe it at (at least) three different levels. The first (what Marr calls the computational) level is a specification of the transformation effected (a description of which sounds are associated with which meanings). The second (what Marr calls the algorithmic) level is a description of an algorithm which realizes this specification.[9] The third (Marr's implementational) level is a description of how the algorithm is realized in the physical medium (our brains). The levels approach offers a natural perspective on the relation between grammar and parser; the grammar is a specification of the parser, which is the algorithm computing the form-meaning relation described by the grammar.

The above account of ellipsis is stated at Marr's computational level, which describes what is being computed but not how. The most natural way of implementing the recognition of ellipsis in this context is to separate the *detection* of ellipsis sites from their *resolution*, at least logically (Kobele, 2012a) (although this separation can and should be 'parallelized' on-line). This allows perfectly standard minimalist grammar parsing algorithms (Harkema, 2001) to be used to construct parse trees with unresolved ellipsis sites, which are written in fraktur as $\mathfrak{E}$ (upper case 'E').[10] Note that because we allow for deletion of contexts, not just subtrees, unresolved ellipsis sites take the form of relation symbols with rank arbitrary (but bound by a function of the length of the sentence), as in figure 4, which represents the derivation tree in figure 2 with its ellipsis site unresolved.

A theory neutral way of stating an ellipsis resolution algorithm is the following. We are given a type of a *context* and a *selection function* `sel` which determines

[9]Peacocke (1986) suggests that linguistic theory is actually at a mid-point between levels one and two (what he appropriately calls level 1.5), where not only the sound-meaning relation is described, but also the major data structures (Marr's representations) used in its computation.

[10]This is actually a consequence of the fact (mentioned in footnote 7) that the distributional restrictions on **delete** and **elide** operations are, modulo the identity condition, regular, together with the fact that the maximum rank of an ellipsis symbol is bounded by the number of words in a sentence. (This bound requires us to take empty lexical items into account – a better approach would be to somehow determine an *a priori* upper bound.) This also raises an interesting question about the proper place for ellipsis in a grammatical theory, one which is borne upon by data such as vehicle change (Fiengo and May, 1994) and split antecedence (Hardt, 1993). Here, my goal is to examine the relation between deletion and copying theories of ellipsis, and I will not consider this question further.

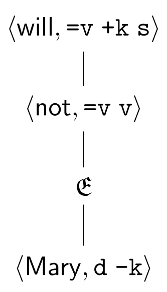

Figure 4: A derivation tree with unresolved ellipsis site for "Mary will not"

the best resolution to a particular ellipsis site given a context. We traverse a tree with unresolved ellipsis sites and use `sel` to resolve any ellipsis sites encountered based on the current context.

Two things are of particular interest about this setup. First, this is the same basic framework as the one developed by (de Groote, 2006) for pronoun resolution, and, as noted there, the selection function `sel` can be parameterized by arbitrary contextual information. Thus, the parallels between ellipsis resolution and pronoun resolution (cf. (Hardt, 1993)) are hereby partially explained by treating them in terms of the same (or a similar) mechanism. Second, taking the goal of parsing to be the recovery not of a parse tree but of a semantic interpretation, the context can simply record semantic denotations (paired with syntactic categories) instead of derivation tree contexts. This follows from the existence of a variable free interpretation scheme for minimalist grammars (Kobele, 2012c), which can assign semantic interpretations to arbitrary derivation tree contexts. Thus, although this is an implementation of a syntactic identity theory of ellipsis, we are free to 'leave syntax behind', and work at the level of meanings.

## 4.1 Resolving Ellipsis

As noted above, I concieve the processing of minimalist grammars with ellipsis (as described in §3) in two steps. In the first step, we ignore the difficulty of resolving ellipsis, and focus on finding structures for sentences which may contain special ellipsis symbols (𝔈). The second step, which I describe briefly here, involves 'fleshing out' hypothesized ellipsis sites (𝔈) with possible antecedents, which then permits a standard computation of meaning (in the sense of compositional semantics). (Note that this step can be viewed as iterated (*OI* (Engelfriet and Schmidt, 1977)) second-order language substitution.[11]) Note that this step is necessary for

[11] 'Iterated' because an 𝔈 may be replaced with a structure which itself contains an 𝔈 (a possibility which would allow a natural account of certain data (Tomioka, 2008)), and 'OI' because different 𝔈s may be replaced by different structures. This is second-order substitution because we are substituting

resolving the membership problem ('is this string generated by the grammar'), as a string could be accepted by the first step of the parser (i.e. the parser says 'if you can find an antecedent for this ellipsis site, the answer is yes'), yet there might be no possible antecedent (i.e. there is no derivation which actually gives rise to that string).

It is instructive to consider a naïve 'two step' approach to the ellipsis resolution problem, whereby we first traverse the derivation tree to flesh out our context, and then next replace ellipsis sites with appropriate antecedents as given by the context.[12] We first need to be a little more explicit about what, exactly, a context is. An ellipsis site $\mathfrak{E}$ occurs in a derivation as a daughter to some node $c$, and with immediate subtrees $t_1$ through $t_n$. Any legitimate fleshing out of this ellipsis site must therefore be by some object $C[x_1, \ldots, x_n]$ with $n$ 'holes' (for subtrees $t_1$ through $t_n$), and moreover the hole $x_i$ for subtree $t_i$ must be able to be filled by something with the featural make-up of subtree $t_i$, for each $1 \leq i \leq n$, and the result of plugging subtrees $t_1, \ldots, t_n$ into holes $x_1, \ldots, x_n$ must be something with the appropriate featural make-up to occur in the original derivation tree as the daughter of node $c$. We can represent such an object as a typed lambda term $\lambda x_1^{T_1}, \ldots, x_n^{T_n}.C(x_1^{T_1}) \cdots (x_n^{T_n})$ of type $T_1 \rightarrow \cdots \rightarrow T_n \rightarrow T$, where the types reflect the featural make-up of an object.[13] A context should then be a map from types to sets of terms of that type. I define the spine $\mathtt{sp}(t)$ of a term $t$ to be the set of second order $T$ (excluding the identity function) such that there are some $t_1, \ldots, t_{n_T}$ such that $T(t_1) \cdots (t_{n_T}) = t$. Then the set $[\![t]\!]$ of all antecedents in a given derivation tree $t$ (viewed as a lambda term) is simply the set $\mathtt{sp}(t)$, if $t$ is a constant, and $[\![M]\!] \cup [\![N]\!] \cup \mathtt{sp}(MN)$, if $t = MN$.

A first option is to simply set our context to be the function from a type to the set of terms of that type in $[\![t]\!]$. This has the consequence that there are no (logical) constraints on accessibility for antecedents – an ellipsis site can take as antecedent something on its right, something above it, something beneath it, etc. Any *empirical* constraints would then need to be implemented in terms of restrictions on the

not (only) trees but contexts. Viewing this as substitution actually requires us to use different versions of $\mathfrak{E}$ depending on the categories of its arguments and the category it 'produces'. (Bringing us into a many-sorted algebra, or, more generally, a typed lambda calculus.)

[12] I will make use of the simply typed lambda calculus in this section (Hindley and Seldin, 2008). A type is either atomic $a$, or an implicaiton $\alpha\beta$ for types $\alpha$ and $\beta$. A lambda term of type $\alpha$ is either a constant $c^\alpha$ (from some set of constants), or a variable $x^\alpha$, or an application $(MN)$ of one term $M$ of type $\beta\alpha$ to another $N$ of type $\beta$, or, if $\alpha = \beta\zeta$ an abstraction $\lambda x^\beta.M$ of variable $x^\beta$ of type $\beta$ in term $M$ of type $\zeta$. I make use of the standard notions of $\alpha, \beta, \eta$ reduction and equivalence, and consider terms in $\eta$ long form (those where every argument position is saturated with a variable). A term is *second order* if every variable which occurs in it is of atomic type. It is *first order* if no variable occurs in it. A first order term can be thought of as a tree, and a second order term can be thought of as a function from trees to trees. I am concerned only with terms which are *closed* (which means that every variable occuring in them is bound) and *linear* (which means that every variable occuring in them occurs exactly once as an argument). Finally, I write $M^\alpha$ to indicate that $M$ is a term of type $\alpha$.

[13] More precisely, the 'types' here are the (finitely many) categories of the MCFG (Seki et al., 1991) obtained by translating a MG in the manner of Michaelis (2001).

antecedent selection function `sel`.

Another option is to take the set of antecedents which are available to a particular ellipsis site to be a function of that ellipsis site's position in the derivation tree.[14] This is really a family of options, as there are a great many ways of indexing antecedent availability to position. The advantage of this is that it removes some of the burden that the previous option puts on the selection function, but of course it can only be empirically adequate if there are indeed hard restrictions on antecedent availability. The most interesting way of indexing antecedent availability to position is one which can be linked to a particular traversal of the derivation tree (Gerdemann, 1994); this can then be implemented by incrementally updating the context during a traversal, and allowing an ellipsis site to take as antecedent only those terms in the context when the ellipsis site is encountered. This would seem to lend itself naturally to an incremental parsing strategy, which resolves ellipsis sites in an online manner.

## 4.2 The competence hypothesis

One question which may arise at this point is the precise relation between the grammar of ellipsis (using deletion under derivational identity) and its processing (using 'LF-copying'). My claim is that this relation satisfies even the *strong competence hypothesis* of Bresnan (1982). I assume there is no doubt that the parsers (Harkema, 2001) for minimalist grammars *without* ellipsis satisfy this constraint (for otherwise MGs with ellipsis fail to satisfy strong competence for uninteresting reasons). The question, then, is whether LF-copying is reasonably thought of as an algorithmic implementation of deletion under derivational identity. As the question of when two algorithms are the same is still unresolved (Blass et al., 2009), this question can only here be answered by appeal to intuitions. The main work which needs to be done in parsing minimalist grammars with ellipsis is, once an ellipsis site has been postulated, to find an appropriate antecedent and verify that the content of the ellipsis site is identical to this antecedent (this is a side condition on the application of the operation **elide**). There are two natural ways of proceeding. One is the 'generate-and-test' method, according to which a possible fleshing out of the ellipsis site is created, and checked against the antecedent. The other option is to, making use of the constraint on the nature of the legitimate fleshings out of the ellipsis site imposed by the grammar, use the antecedent to compute an appropriate one such. This option can be seen as a *guided* version of the generate-and-test method, in a manner similar to how top-down parsing can be seen as a(n input-) guided version of a generate-and-test parsing algorithm. In fact, the copying of the antecedent is similar to *memoization* (Johnson, 1995); as we have already gone through the steps of constructing the antecedent, we can simply re-use them *en masse*.[15] Finally, the argument for strong competence comes down to this: if

[14]Position in the surface structure can be computed on the basis of position in the derivation tree.

[15]Indeed, the procedures alluded to in §4.1 for computing the possible antecedents can be thought of as constructing memo-tables of type-indexed subterms.

we do not balk at calling memoized or input-guided generate-and-test algorithms realizations of a grammar, why should we hesitate here?[16]

## 5 Conclusions

I have presented a formalization of some common themes in the minimalist literature regarding ellipsis, where there is a debate between proponents of a LF-copying approach to ellipsis and those of a PF-deletion approach to the same. I have shown that, under the assumption that the identity condition governing ellipsis is formulable in terms of identity of derivations, the differences between the two approaches disappear. Keeping syntactic identity at the level of the derivation (instead of the derived tree), allows for some flexibility regarding suface antecedent-ellipsis mismatches. One such which has been worked out is the differential acceptability of voice mismatches in VP ellipsis and sluicing (following Merchant, 2007). Other well-known surface mismatches (such as 'vehicle change' (Fiengo and May, 1994), or split antecedence (Hardt, 1993)) do not appear to have a natural syntactic characterization, but do nevertheless seem amenable to treatment at the algorithmic level if we 'deforest' the trees and use instead semantic terms (vehicle change by means of 'copying' the pronoun selection function instead of the pronoun and split antecedence by allowing the ellipsis selection function to choose simultaneously multiple antecedents, and then combine them semantically using either pointwise conjunction or some other operator). This move, however, takes us away from a syntactic identity theory, and is left to future work. It is hoped that this formal presentation will serve to make clear the commitments, prospects, and difficulties faced by a deletion under identity theory of ellipsis.

## References

Arregui, Ana, Clifton, Jr., Charles, Frazier, Lyn and Moulton, Keir. 2006. Processing elided verb phrases with flawed antecedents: The recycling hypothesis. *Journal of Memory and Language* 55, 232–246.

Blass, Andreas, Dershowitz, Nachum and Gurevich, Yuri. 2009. When are two algorithms the same? *Bulletin of Symbolic Logic* 15(2), 145–168.

Bresnan, Joan (ed.). 1982. *The Mental Representation of Grammatical Relations*. Cambridge, Massachusetts: MIT Press.

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

[16]Although the discussion here is somewhat divorced from the literature, it pertains, I believe, just as well to the discussion between Martin and McElree (2008) and Frazier and Clifton Jr. (2001).

Chung, Sandra. 2006. Sluicing and the lexicon: The point of no return. In Rebecca T. Corver and Yuni Kim (eds.), *BLS 31: General Session on Parasession on Prosodic Variation and Change*, pages 73–91, Berkeley Linguistics Society, Berkeley, CA.

Chung, Sandra, Ladusaw, William A. and McCloskey, James. 1995. Sluicing and Logical Form. *Natural Language Semantics* 3(3), 239–282.

de Groote, Philippe. 2006. Towards a Montagovian Account of Dynamics. In Masayuki Gibson and Jonathan Howell (eds.), *Proceedings of SALT 16*, pages 1–16.

Engelfriet, Joost and Schmidt, Erik Meineche. 1977. IO and OI. I. *Journal of Computer and System Sciences* 15(3), 328–353.

Fiengo, Robert and May, Robert. 1994. *Indices and Identity*. Cambridge, Massachusetts: MIT Press.

Frazier, Lyn and Clifton Jr., Charles. 2001. Parsing Coordinates and Ellipsis: Copy $\alpha$. *Syntax* 4(1), 1–22.

Gerdemann, Dale. 1994. Parsing as Tree Traversal. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING)*, volume 1, pages 396–400.

Hale, John T. and Stabler, Edward P. 2005. Strict deterministic aspects of minimalist grammars. In Philippe Blache, Edward P. Stabler, Joan Busquets and Richard Moot (eds.), *Logical Aspects of Computational Linguistics*, volume 3492 of *Lecture Notes in Computer Science*, Chapter 11, pages 162–176, Springer.

Hardt, Daniel. 1993. *Verb Phrase Ellipsis: Form, Meaning, and Processing*. Ph. D.thesis, University of Pennsylvania.

Harkema, Henk. 2001. *Parsing Minimalist Languages*. Ph. D.thesis, University of California, Los Angeles.

Hindley, J. Roger and Seldin, Jonathan P. 2008. *Lambda-calculus and combinators: an introduction*. Cambridge University Press.

Jaeggli, Osvaldo A. 1986. Passive. *Linguistic Inquiry* 17(4), 587–622.

Johnson, Mark. 1995. Memoization in top-down parsing. *Computational Linguistics* 21(3), 405–417.

Joshi, Aravind K. 1985. How much context-sensitivity is necessary for characterizing structural descriptions. In David Dowty, Lauri Karttunen and Arnold Zwicky (eds.), *Natural Language Processing: Theoretical, Computational and Psychological Perspectives*, pages 206–250, NY: Cambridge University Press.

Kim, Christina S., Kobele, Gregory M., Runner, Jeffery T. and Hale, John T. 2011. The acceptability cline in VP ellipsis. *Syntax* 14(4), 318–354.

Kobele, Gregory M. 2009. Syntactic Identity in Survive Minimalism: Ellipsis and the Derivational Identity Hypothesis. In Michael T. Putnam (ed.), *Towards a purely derivational syntax: Survive-minimalism*, John Benjamins.

Kobele, Gregory M. 2012a. Ellipsis: computation of. *WIREs Cognitive Science* 3(3), 411–418.

Kobele, Gregory M. 2012b. Idioms and extended transducers. In *Proceedings of the Eleventh International Workshop on Tree Adjoining Grammars and Related Frameworks (TAG+11),* Paris.

Kobele, Gregory M. 2012c. Importing Montagovian Dynamics into Minimalism. In Denis Béchet and Alexandre Dikovsky (eds.), *Logical Aspects of Computational Linguistics*, volume 7351 of *Lecture Notes in Computer Science*, pages 103–118, Berlin: Springer.

Kobele, Gregory M., Retoré, Christian and Salvati, Sylvain. 2007. An automata theoretic approach to minimalism. In James Rogers and Stephan Kepser (eds.), *Proceedings of the Workshop Model-Theoretic Syntax at 10; ESSLLI '07*, Dublin.

Lees, Robert Benjamin. 1960. *The grammar of English nominalizations*. The Hague: Mouton.

Lichte, Timm and Kallmeyer, Laura. 2010. Gapping through TAG Derivation Trees. In *Proceedings of the 10th Conference on Tree Adjoining Grammar and Related Frameworks*, New Haven, CT.

Marr, David. 1982. *Vision*. New York: W. H. Freeman and Company.

Martin, Andrea E. and McElree, Brian. 2008. A content-addressable pointer mechanism underlies comprehension of verb-phrase ellipsis. *Journal of Memory and Language* 58, 879–906.

Martín González, Javier. 2010. Voice mismatches in English and Spanish sluicing. *Iberia* 2(2), 23–44.

Merchant, Jason. 2001. *The Syntax of Silence: Sluicing, Islands, and the Theory of Ellipsis*, volume 1 of *Oxford Studies in Theoretical Linguistics*. New York: Oxford University Press.

Merchant, Jason. 2007. Voice and ellipsis, ms., University of Chicago.

Merchant, Jason. 2008. An asymmetry in voice mismatches in VP-ellipsis and pseudogapping. *Linguistic Inquiry* 39(1), 169–179.

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

Osborne, Timothy, Putnam, Michael and Groß, Thomas. 2013. Catenae: Introducing a novel unit of syntactic analysis. *Syntax* .

Peacocke, Christopher. 1986. Explanation in Computational Psychology: Language, Perception, and Level 1.5. *Mind & Language* 1(2), 101–123.

Sag, Ivan A. 1976. *Deletion and Logical Form*. Ph. D.thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts.

Seki, Hiroyuki, Matsumura, Takashi, Fujii, Mamoru and Kasami, Tadao. 1991. On multiple context-free grammars. *Theoretical Computer Science* 88, 191–229.

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

Strachey, Christopher and Wadsworth, Christopher P. 2000. Continuations: A mathematical semantics for handling full jumps. *Higher-Order and Symbolic Computation* 13, 135–152.

Tanaka, Hidekazu. 2011. Voice mismatch and syntactic identity. *Linguistic Inquiry* 42(3), 470–490.

Tomioka, Satoshi. 2008. A step-by-step guide to ellipsis resolution. In Kyle Johnson (ed.), *Topics in Ellipsis*, Chapter 9, pages 210–228, Cambridge University Press.

Wadler, Philip. 1992. Comprehending Monads. *Mathematical Structures in Computer Science* 2, 461–493.

Webber, Bonnie. 2004. D-LTAG: extending lexicalized TAG to discourse. *Cognitive Science* 28, 751–779.