HEADEDNESS IN A CATEGORILESS MEANING-DEPENDENT GRAMMAR*

PAUL ELBOURNE
University of Oxford

1 Introduction

This article will introduce and begin to evaluate a variant of the grammar that I proposed in Elbourne 2024. That grammar, while being broadly Minimalist in terms of outlook and technical details, differed from previous Minimalist grammars in abolishing both syntactic categories and headedness (along with related notions like projection). The variant to be introduced in the current paper will continue to eschew syntactic categories but will reintroduce headedness. The whole project can conveniently be labelled *Meaning-Dependent Grammar*; in the particular variant explored here, the aim is to use semantic categories and principles to take over the work formerly done by syntactic categories, thus rendering the latter otiose.

I see one major reason for being interested, in this framework, in reintroducing headedness. That is that the system thus created is arguably more clearly free from overgeneration than the system in Elbourne 2024, in that at any stage of a derivation the choice of constituent to be added next is more narrowly constrained than it is in the earlier system. Relatedly, there seem to be some data concerning the subcategorization properties of English verbs that present a problem to the older system that can be solved by means of this innovation. I discuss these considerations at greater length in Section 3. But first, to put the discussion on a concrete footing, here is a grammar of the kind I am currently contemplating.

2 A New Grammar

This section lays out a Meaning-Dependent Grammar based closely on the one found in Elbourne 2024; the presentation is largely an abbreviation of what is found in that article, which should be consulted for further details on many points. The two most significant changes from Elbourne 2024 are the addition of headedness and a slightly more sophisticated treatment of tense. As before, the

*It is a great pleasure to contribute an article to a volume dedicated to Danny Fox, whose perceptive comments greatly improved my PhD thesis. I thank Danny for his mentorship and for his dedication to the MIT Linguistics section. I am very grateful to Moshe Bar-Lev for comments on a draft of this article.

grammar is based closely on Stabler's Minimalist Grammars syntactic framework (Stabler, 1997, 2011, 2013, Elbourne, 2016).

2.1 Syntax

The grammar will be a *directional minimalist grammar* (i.e. one that allows linear order to be read off from trees and taken into account in the formulation of Merge; Stabler 2011:635).

2.1.1 Features

There are three kinds of features: semantic, phonological, and syntactic. The following abbreviations will be adopted: (donkey) is the semantic value of the word *donkey*; '/donkey/' represents the phonological features; 'donkey' alone, without quotation marks, summarizes the semantic and phonological features; a double colon, '::', separates the different kinds of features of one word in lexical entries and trees.

There are three kinds of syntactic features:

(1) *Selector features*. A feature E_L indicates that a constituent needs to combine with another constituent to its left via External Merge. Likewise, with the obvious change, for E_R. Selector features in this particular system are annotated with semantic types: $E_{R,\langle eit,eitit\rangle}$, for example, is a feature that requires something whose head is of type $\langle eit,\langle eit,it\rangle\rangle$ (the type of determiners in this system—'i' indicates time intervals).

(2) *Probe features*. Features written with various forms of the letter *i* are triggers of movement (Internal Merge). The only probe features used in the current article will be weak features, which produce covert movement, which is to say movement of only the semantic and syntactic features of the constituent concerned. These features will be written 'i'. Only items whose head is of type $\langle eit,\langle eit,it\rangle\rangle$ (i.e. determiner phrases) will be moved in the current system, so no indication of type is necessary. (This could obviously be changed in future.)

(3) Features interpreted by syncategorematic rules, restricted in the current system to those that characterize λ-operators and traces.

Selector features and probe features are descendants of the features variously called c-selectional features (Adger, 2003:96), edge features (Chomsky, 2008:139), selector features (Stabler, 2011, Elbourne, 2016), and trigger features (Collins and Stabler, 2016:62) by other Minimalist authors.

Goal features are not used in the current system. Selector features and probe features cannot be interpreted by the semantics and will sometimes be lumped together and referred to as *uninterpretable*. Features interpreted by syncategorematic rules are *interpretable*, of course.

2.1.2 Sentences and Derivations

The role of a generative grammar is seen as being to generate sentences, where a *sentence* is defined as a structure of type $\langle i,t\rangle$ that does not contain any uninterpretable features.

The grammar is set up in such a way that any object produced by it in the course of a derivation will contain at most one node that bears uninterpretable syntactic features. This node will be called the *driver*, since the features on it determine the direction of the computation.

2.1.3 Projection and Headedness

When two constituents combine, one of them will always *immediately project* over the other. This is a primitive relation. The more general relation of *projection*, simpliciter, is the reflexive and transitive closure of the relation of immediate projection (Stabler, 1997).

A *head* is defined as follows (Stabler, 1997).

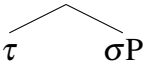
(4) A node x is a head of a node y iff either
   a. y is a leaf and $x = y$, or
   b. there is a node z with the following properties:
      i. y is the parent of z;
      ii. z projects over any daughters of y; and
      iii. x is a head of z.

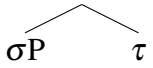
The *maximal projection* of a head x is a particular member of the set of nodes whose head is x, namely the one that no member of this set dominates.

2.1.4 Rules

The following abbreviations will be used in the presentation of rules. $\tau[\alpha]$ denotes a tree whose driver has a sequence of uninterpretable syntactic features whose first element is α. Given a structure $\tau[\alpha]$, τ denotes the result of erasing feature α. Given a tree τ_1 with subtree τ_2, $\tau_1\{\tau_2 \mapsto \tau_3\}$ is the result of replacing τ_2 with τ_3 in τ_1. ε indicates an empty sequence. Given a tree τ, $|\tau|$ indicates the syntactic features of τ. σ is used as a variable over semantic types. σP is the maximal projection of a head of type σ. The grammar will include the following rules.

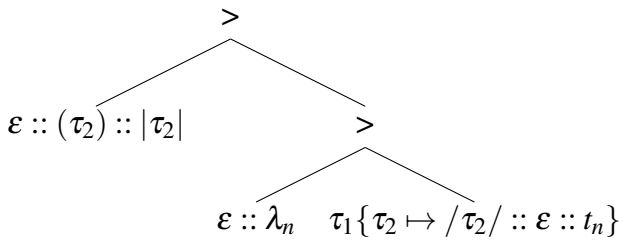
(5) *External Merge.* Since we are operating with a directional minimalist grammar, there are two cases.

a. $merge(\tau[\mathrm{E}_{\mathrm{R},\sigma}], \sigma\mathrm{P}) =$ 

b. $merge(\tau[\mathrm{E}_{\mathrm{L},\sigma}], \sigma\mathrm{P}) =$ 

The < and > indicate the projection relation: τ projects over σP in both cases. This notation has the consequence that we can find the head of any node simply by following the arrows downwards until we reach a leaf. Note that the E feature on τ has been deleted.

(6) *Internal Merge* applies as follows to a tree $\tau_1[\mathrm{i}]$ containing a subtree τ_2 whose head is of type $\langle \mathrm{eit}, \langle \mathrm{eit}, \mathrm{it} \rangle \rangle$:



This is covert movement, since we are dealing with a weak (i) probe feature. The semantic and syntactic features of the relevant determiner phrase raise, leaving behind the phonological features and a trace. The i feature in τ_1 is deleted. Other kinds of movement will not be shown here.

The following principle will constrain External Merge (Collins and Stabler, 2016:64):

(7) *Argument Interpretability*
The constituent selected for by a selector feature must not contain any uninterpretable syntactic features.

Covert movement will be constrained by Fox's (2000:23) Scope Economy constraint:

(8) *Scope Economy*
Covert scope-shifting operations that are not forced for type considerations must have a semantic effect.

2.1.5 Lexical entries

Entries in the list in (9) have the following form: 'phonological features :: semantic features :: syntactic features'. Optional syntactic features are given in angle brackets: $\langle\alpha\rangle$. The others are compulsory. In subscripted types, s is the type of events and i is the type of time intervals. The semantic types of the words are given on the right for convenience.

(9)

/Fido/ :: $\lambda f_{\langle e,it\rangle}.\lambda t.f(o)(t)$ :: ε	$\langle$eit,it$\rangle$
/someone/ :: $\lambda f_{\langle e,it\rangle}.\lambda t.\exists x(\text{person}(x)(t)\ \&\ f(x)(t))$ :: ε	$\langle$eit,it$\rangle$
/everyone/ :: $\lambda f_{\langle e,it\rangle}.\lambda t.\forall x(\text{person}(x)(t) \rightarrow f(x)(t))$ :: ε	$\langle$eit,it$\rangle$
/every/ :: $\lambda f_{\langle e,it\rangle}.\lambda g_{\langle e,it\rangle}.\lambda t.\forall x(f(x)(t) \rightarrow g(x)(t))$ :: $E_{R,\langle e,it\rangle}$	$\langle$eit, $\langle$eit,it$\rangle\rangle$
/some/ :: $\lambda f_{\langle e,it\rangle}.\lambda g_{\langle e,it\rangle}.\lambda t.\exists x(f(x)(t)\ \&\ g(x)(t))$ :: $E_{R,\langle e,it\rangle}$	$\langle$eit, $\langle$eit,it$\rangle\rangle$
/donkey/ :: $\lambda x.\lambda t.\text{donkey}(x)(t)$ :: ε	$\langle$e,it$\rangle$
/woman/ :: $\lambda x.\lambda t.\text{woman}(x)(t)$ :: ε	$\langle$e,it$\rangle$
/cute/ :: $\lambda f_{\langle e,it\rangle}.\lambda x.\lambda t.f(x)(t)\ \&\ \text{cute}(x)(t)$ :: $\langle E_{R,\langle e,it\rangle}\rangle$	$\langle$eit,eit$\rangle$
/dance/ :: $\lambda R_{\langle i,it\rangle}.\lambda e.\lambda t.\text{dance}(e)\ \&\ R(\text{CUL}(e))(t)$ :: $E_{R,\langle i,it\rangle}$	$\langle$iit,sit$\rangle$
/inspect/ :: $\lambda R_{\langle i,it\rangle}.\lambda x.\lambda e.\lambda t.\text{inspection}(e)\ \&\ \text{Theme}(e)(x)\ \&\ R(\text{CUL}(e))(t)$ :: $E_{R,\langle i,it\rangle}\ E_{R,\langle eit,eitit\rangle}$	$\langle$iit,$\langle$e,sit$\rangle\rangle$
/ed/ :: $<$ :: ε	$\langle$i,it$\rangle$
/beautifully/ :: $\lambda F_{\langle s,it\rangle}.\lambda e.\lambda t.F(e)(t)\ \&\ \text{beautiful}(e)$ :: $E_{R,\langle iit,*\rangle}/E_{L,\langle iit,*\rangle}$	$\langle$sit,sit$\rangle$
ε :: $\lambda F_{\langle s,it\rangle}.\lambda x.\lambda t.\exists e(F(e)(t)\ \&\ \text{Agent}(e)(x))$ :: $E_{R,\langle iit,*\rangle}\ E_{L,\langle eit,eitit\rangle}\ \langle i\rangle\ \langle i\rangle$	$\langle$sit,eit$\rangle$
/not/ :: $\lambda p_{\langle i,t\rangle}.\lambda t.\neg p(t)$ :: ε	$\langle$it,it$\rangle$

Little v (the penultimate entry) is shown in simplified form: an optional raft of features designed to raise the subject from a vP-internal position in the presence of negation has been omitted.[1]

The symbol '$<$' in the lexical entry of the past tense morpheme designates a function of type $\langle$i,it$\rangle$ such that $<(t)(t')$ iff t occurred before t'. For any event e, 'CUL(e)' is the culmination of e, which is to say the last temporal point at which e obtains.

A subscripted type in a denotation ('$\lambda f_{\langle e,it\rangle}$'...) means that we are dealing with a function that takes something of that type as an argument in the semantics; but a subscripted type on an E_R or E_L feature indicates a word or constituent that takes in the syntax something whose head is of that type. The type annotation '$\langle$iit,*$\rangle$' is a regular expression that picks out functions that map

[1] I will continue to use traditional syntactic category labels for convenience sometimes, as I just did; but such things have no official place in the theory.

functions of type ⟨i,it⟩ to something or other; it picks out all and only the verbs in this system. (Verbs are things that combine with tense.)

As long as lexical items of different traditional syntactic categories are given different semantic types, as they are in the above listing, a system like the current one will be able to mimic the use of syntactic categories and projections thereof by its use of semantic types and projections thereof.

2.2 Semantics

The semantics of this fragment will be as in Heim and Kratzer 1998: I take it that their rules of Functional Application, Predicate Abstraction, Lexical Terminals, and Traces and Pronouns will be known to readers of this article. Nothing else will be needed.

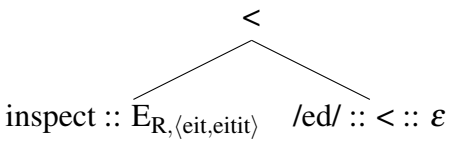
2.3 An Example

As an illustration of the workings of this system, I will show how it derives the structure and meaning of the inverse scope reading of the following sentence:

(10) Some woman inspected every donkey.

Starting with the verb *inspect*, we see from its $E_{R,\langle i,it\rangle}$ feature in (9) that it has to merge with something whose head is of type ⟨i,it⟩. There is only one constituent of this kind, /ed/. So we obtain the following:

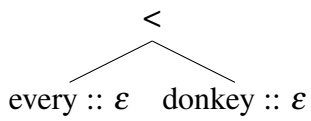
(11)



The driver is *inspect*. Its uninterpretable feature requires merger with something whose head is of type ⟨eit,eitit⟩. We could attempt to merge in a bare determiner, but these have their own uninterpretable features, so that would violate (7) (Argument Interpretability).

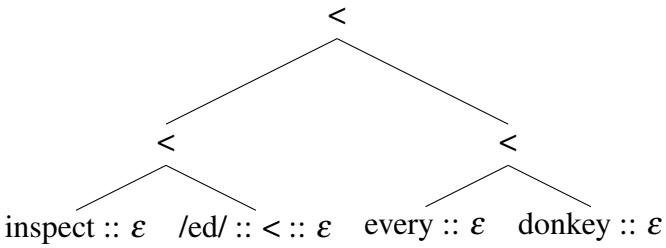
We select *every* to be merged in. In order to remove its uninterpretable feature, we have to combine it with something whose head is of type ⟨e,it⟩. This could in principle be a noun modified by an adjective or in some other way, but for simplicity we will just choose a bare noun, *donkey*. It combines with *every* as follows:

(12)



Now we have a structure whose head is of type ⟨eit,eitit⟩ with which *inspected* can combine:

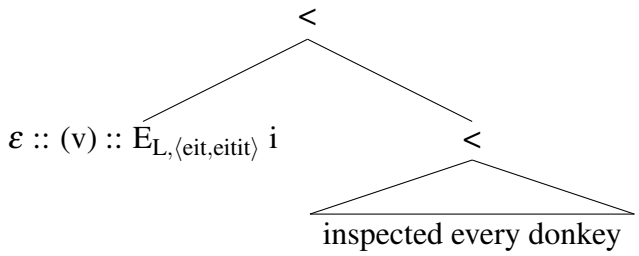
(13)



This structure has no more uninterpretable features to guide the derivation, but it has a head whose type falls under the ⟨iit,*⟩ pattern noted earlier. So we have to combine it with little v, since nothing

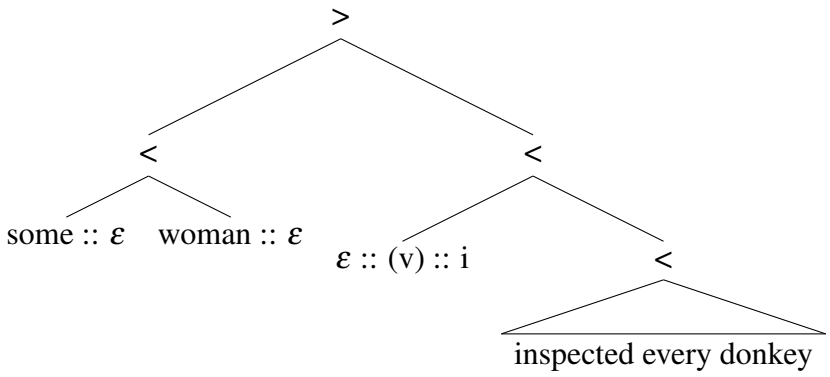
else in (9) takes things of that type (except adverbs, but we fill ignore those for now). We include only one of little v's optional i features:

(14)



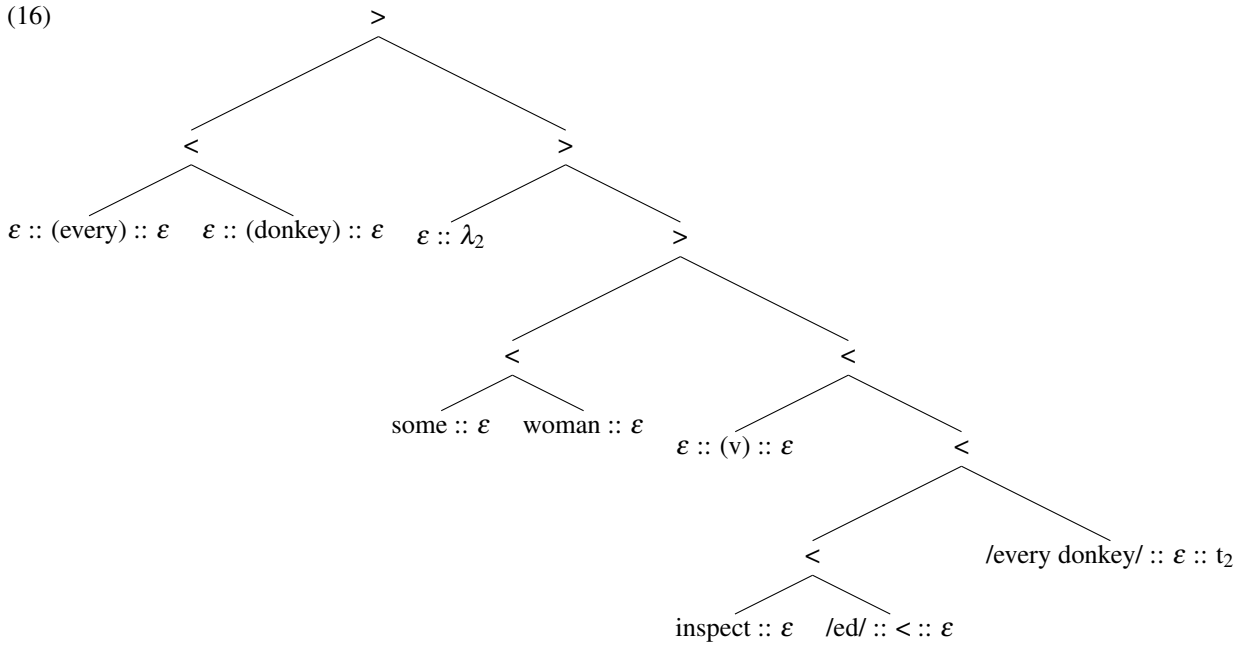
Little v is now the driver. Its first syntactic feature tells us that we have to merge in to its left something whose head is of type ⟨eit,eitit⟩. This is similar to the requirement of *inspected* that we saw earlier. We form a determiner phrase and merge it in:

(15)



There is just one uninterpretable syntactic feature left, the i feature on little v, which mandates covert movement of something whose head is of type ⟨eit,eitit⟩. There are two such structures in the tree. Which one is to be moved? The answer follows from (8) (Scope Economy): moving the subject would be a trivial, short-distance movement with no semantic effect. So we have to move the object. This creates the final structure for our example:

(16)



All the phonological and semantic features are laid out in appropriate positions for interpretation by their respective interfaces. A straightforward calculation shows that the denotation of the whole tree is as follows:

(17) $\lambda t.\forall x(\text{donkey}(x)(t) \rightarrow \exists y(\text{woman}(y)(t)\ \&\ \exists e(\text{inspection}(e)\ \&\ \text{Agent}(e)(y)\ \&\ \text{Theme}(e)(x)\ \&\ <(\text{CUL}(e))(t))))$

These truth conditions are intuitively adequate.

3 The Risk of Overgeneration

As mentioned in Section 1, one possible advantage of the system proposed here is that it is arguably more clearly free from overgeneration than the system in Elbourne 2024, in that at any stage of a derivation the choice of what constituent to add next is very narrowly constrained. I take it that the preceding example has illustrated this property for the current system. In the system in Elbourne 2024, since there is no headedness (and no syntactic categories, of course), we have to rely to a large extent on the overall semantic type of constituents to give trees the right shape: little v does not take something whose head fits the ⟨iit,*⟩ template but something of overall type ⟨s,it⟩, and so on. A possible problem with that system is that in many cases it is not clear how we can demonstrate that nothing of the requisite overall type can be constructed that we do not want to occupy the relevant position.

I do not know of many attested cases that actually present problems along these lines. But there are one or two to be found in the nominal realm. In Elbourne 2024, nouns were of type ⟨e,t⟩ and adjectives were of type ⟨et,et⟩. We also need to look at determiner phrases whose determiner is *a*, which I will call *a*-phrases. Now adjectives and *a*-phrases partially overlap in their distribution:

(18) a. The guard is alert.
b. The guard is a good marksman.

This implies, in the system of Elbourne 2024, that these two kinds of phrases should have the same semantic type (at least sometimes). But this system then has difficulty explaining how one kind but not the other kind is available after certain verbs. Let us take *get* in the sense of 'make become':

(19) a. The sergeant got the guard alert.
b. *The sergeant got the guard a good marksman.

Since distribution is dictated by overall semantic type in my earlier system and there is evidence that *a good marksman*, at least sometimes, is of the same type as *alert*, it is mysterious why one of these phrases but not the other can be taken as a predicate complement by *get* (in the relevant sense).

The system in the current article, however, will have no difficulty in distinguishing between adjectives and *a*-phrases when the need arises. It is plausible that *a*-phrases acting as predicates have *a* as their head, with a special meaning that will take a noun meaning and map it to something of the same type as adjectives. Adjectives and *a*-phrases will be of the same overall type, which helps to explain how they can both sit comfortably after the copula in (18). In this use, then, *a* will be of type ⟨eit,eiteit⟩, another unique type. So adjectives and adjective phrases will be items whose head is of type ⟨eit,eit⟩ and *a*-phrases (in their predicative use) will be items whose head is of type

⟨eit,eiteit⟩; and *get*, in the relevant sense, subcategorizes for one of these but not the other. In this respect, then, the system in the current article has an advantage over the system in Elbourne 2024.

4 Conclusion

So should we alter the system in Elbourne 2024 along the lines suggested here? I am tentatively inclined to do so, but the judgement is not straightforward. The current system has the advantages just outlined. But it adds an important new primitive relation, projection, to the system in Elbourne 2024 and adds a lot of information (about the headedness of arguments) to the lexical entries in that earlier paper.

It might be possible to deal with cases like *get* by means of a localized system of headedness. In other words, it might be possible to devise E features that would be borne by relevant lexical items that would select complements on the basis of their heads without instituting a pervasive system of headedness, as is done in the current paper. The calculations of headedness, or something like it, would be local, without headedness being represented in every part of the tree. But it remains to be seen exactly what such a system would look like and whether it would truly be more economical than a pervasive system of headedness.

In the meantime, we should also note that the possibility of mimicking syntactic categories and their projections by means of semantic types and their projections makes the current project of Meaning-Dependent Grammar more similar in structure to previous varieties of Minimalist syntax. I hope that this might be found reassuring.

References

Adger, David. 2003. *Core syntax*. Oxford: Oxford University Press.

Chomsky, Noam. 2008. On phases. In *Foundational issues in linguistic theory*, ed. Robert Freidin, Carlos P. Otero, and Maria Luisa Zubizarreta, 133–166. Cambridge, MA: MIT Press.

Collins, Chris, and Edward Stabler. 2016. A formalization of minimalist syntax. *Syntax* 19:43–78.

Elbourne, Paul. 2016. Incomplete descriptions and indistinguishable participants. *Natural Language Semantics* 24:1–43.

Elbourne, Paul. 2024. A program for eliminating syntactic categories. *Linguistic Inquiry* Online Early. https://doi.org/10.1162/ling_a_00540.

Fox, Danny. 2000. *Economy and semantic interpretation*. Cambridge, MA: MIT Press.

Heim, Irene, and Angelika Kratzer. 1998. *Semantics in generative grammar*. Oxford: Blackwell.

Stabler, Edward. 1997. Derivational minimalism. In *Logical aspects of computational linguistics: First International Conference, LACL '96, Nancy, France, September 1996, selected papers*, ed. Christian Retoré, 68–95. Berlin: Springer.

Stabler, Edward. 2011. Computational perspectives on minimalism. In *Oxford handbook of linguistic minimalism*, ed. Cedric Boeckx, 617–642. Oxford: Oxford University Press.

Stabler, Edward. 2013. Two models of minimalist, incremental syntactic analysis. *Topics in Cognitive Science* 5:611–633.