Appendix to the article "Predicate clefting and long head movement in Finnish"

to appear in *Linguistic Inquiry*

Final draft version

Pauli Brattico

IUSS University School for Advanced Studies

2021

Abstract. This document is a technical appendix to the original article "Predicate clefting and long head movement in Finnish" to appear in *Linguistic Inquiry*. After providing some further contextualization to the analysis it addresses details of the computational model that were omitted or only briefly mentioned in the main article.

Table of Contents

1 Introduction

1.1 Framework

This document constitutes an appendix to the original article “Predicate clefting and long head movement in Finnish.” The two documents approach the subject matter from complementary perspectives. Whereas the main article was concerned with the details of the empirical data, its interpretation in the light of the current minimalist theory, and used the computational framework mainly for the justification of the analysis, this document elucidates the computational model in some more detail and furthermore discusses in depth the raw output produced by the algorithm. I will also refer to the source code. Finally, the proposed approach is compared with what I consider to be a fairly standard cyclic grammatical model at present writing.

The main empirical problem addressed in this work can be introduced in the following way. We elicit linguistic data from human informants by having them judge properties of linguistic expressions, such as their grammaticality, acceptability, meaning and pragmatic felicity. Additional facts can be gathered by controlled psycho- and neurolinguistic experiments. Our goal is to provide an equation (formal algorithm, model) that captures these facts (Chomsky 1957, 1964, 1965). Here we are specifically interested in explaining such facts inasmuch as they concern the syntactic and semantic processing of morphologically complex phonological words, phenomena that fall mostly under the umbrella of “head movement” in the standard generative theory (e.g., Baker 1988; Matushansky 2006; Roberts 2010; Travis 1984). By saying that the model captures such facts we mean that it is sufficient to calculate the data; if the model is also necessary, then it satisfies an additional criterion of being also the simplest possible hypothesis.[1] Ultimately, though, we aim for a model that is both necessary and sufficient in capturing the widest possible range of facts.

The number of possible grammatical frameworks and models one could adopt as a starting point in cognitive science is very large, ranging from complex generative models to simple Pavlovian connectionist approaches. The standard method used in the more advances sciences for solving disputes of this type is to formalize the competing hypotheses, show that they calculate the facts, and then compare their accuracy, simplicity and generality. The choice of the starting point therefore reflects any author’s subjective guess on what the best approach might be, but justification relies on calculation. We proceed by using this method and expect all disagreement to be solved in the same way.

[1] Perhaps this requirement constitutes the core of “minimalism” in the sense of Chomsky 1995, but if so then minimalism would be nothing but the standard scientific method applied to generative theorizing. My reading of the thesis proposed by Chomsky in the source cited and others is that it goes in some respects beyond this characterization, but the matter is not easy to judge.

Due to the combinatorial nature of linguistic data, the logical distance between the theory and data is often quite vast. To overcome this problem, the hypothesis was written in a machine-readable formalism (Python programming language) so that we can examine its logical consequences efficiently and reliably by using a computer. To make this possible, the data was collected into a test corpus file and processed by a computer program embodying the hypothesis. We are interested in whether the output of the model agrees or disagrees with human behavior, both in relation to offline judgments and real-time reactions. The design is summarized in Figure 1.

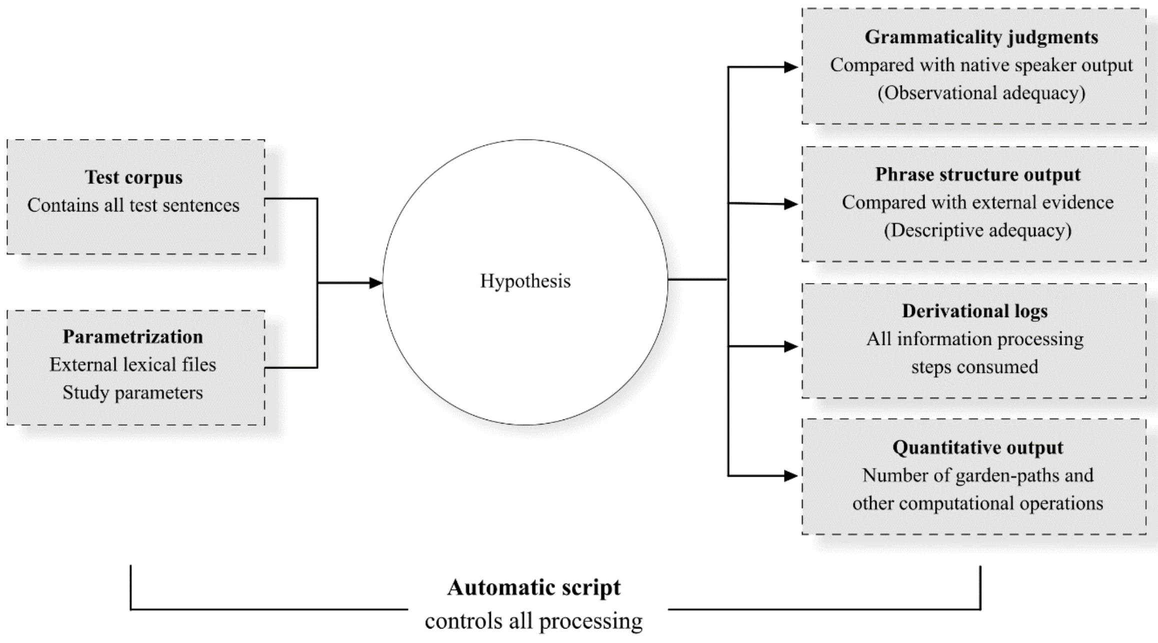


Figure 1. Design of the study. A test corpus containing sentences whose properties we are interested in together with the input parameters are used as the input to an algorithm embodying the hypothesis.

There are two in principle ways of bridging a formal linguistic hypothesis with a dataset: generation and recognition. In a generative theory, the axioms of the theory are employed to generate a set of expressions (phonological strings, syntactic analyses, semantic interpretations) from a set of lexical elements and other input parameters. The output set is compared with data from native speakers. The other method is recognition, in which the theory judges input expressions by linking them with grammaticality judgements, syntactic analyses and semantic interpretations. The two methods are mathematically equivalent, under very weak assumptions: it is easy to manufacture a recognition model from a generative algorithm, and vice versa. The recognition approach was adopted in this study. Even though a recognition algorithm can be and is sometimes seen as a performance hypothesis modelling real-time language comprehension, it incorporates competence inasmuch as it provides correct grammaticality judgments and replicates or otherwise correctly describes human semantic intuitions. In fact, once we develop an algorithm of this type it is often easy to separate its performance properties from those that are involved in competence calculations. Performance issues are discussed in Section 3.1.

1.2 Data

The dataset was discussed extensively in the main article and will not be repeated here. I will provide few examples for reference. An example of the empirical phenomena we are interested in in this study is provided in (1), where we pay special attention to the first word. The language under study is Finnish.[2]

(1)	Myy-dä-kö$_1$	Pekka	halus-i	___$_1$	koko	omaisuute-nsa?
	sell-A/INF-Q	Pekka.NOM	wanted-PAST.3SG		all	possessions-PX/3SG

‘Was it selling that Pekka wanted to do to all his possessions?’

The first word *myy-dä-kö* ‘sell-A/INF-Q’ contains a transitive verbal stem *myy-* ‘sell’, an infinitival suffix *-tA* (‘to’) and a yes/no interrogative particle *-kO* (glossed as Q). The word therefore carries considerable amount of grammatical information, all intuitively and instantly understood by native speakers. Consequently, human language seems to make a distinction between two ways of articulating grammatical information. One way is by squeezing the information inside one phonological word, here *myy-dä-kö* ‘sell-A/INF-Q’. Another is by means of separate words, so that the relevant grammatical information is dispersed into the sentence as independent phonological units. In English, for example, the complex meaning expressed by *myy-dä-kö* ‘sell-A/INF-Q’ requires at least three separate units, *whether* (yes/no interpretation), *to* (infinitival) and *sell* ‘transitive verb’. How to model this difference constitutes one of the questions that we address in this work. It is also an issue that is surrounded by an interesting controversy.

This example raises an additional problem, no less challenging. The canonical position of the complex infinitival verb is designated by ___$_1$ in the example (1), but in this sentence it appears at the beginning of the clause, which we consider a noncanonical position for an element of this type. The verb should be where the empty position is. The canonical position of the word is shown in (2).

(2)	Pekka	halusi	**myy-dä**	koko	omaisuutensa.
	Pekka	wanted	sell-A/INF	all	possessions

‘Pekka wanted to sell all his possessions.’

Thus, we must explain what the possible positions of the word are and why it occurs in the first position in the sentence (1).

[2] Symbol PX/3SG refers to the Finnish infinitival agreement suffix (Huhmarniemi and Brattico 2015). Finnish is a canonical SVO language with a relatively free word order associated with a degree of discourse configurationality, agglutinative morphology and the nominative-accusative case system, with the nominative grammatical subjects agreeing with finite elements in the clause. Grammaticalized articles are absent. Most of the example sentences cited in this document follow the canonical SVO template. For Finnish syntax in general, see Huhmarniemi (2012) and Manninen (2003).

To examine these problems, we gather further observations by presenting sentences of this type plus many variations to native speakers and ask them to judge whether the resulting sentences are possible or impossible and, if the former, what their meaning is. This investigation, when carried systematically, provides a wealth of observations that we add to our research agenda (and to the test corpus). For example, the investigation shows that the fact that the infinitival word has been dislocated to the noncanonical first position is related in some way to the present of the yes/no morpheme Q. The presence of this morpheme forces the word to dislocate (3). If the morpheme is not present, the word can remain in its canonical position (see (2) above).

(3)	*Pekka	halusi	myy-dä-**kö**	koko	omaisuutensa?
	Pekka	wanted	sell-A/INF-Q	all	possessions

The dislocation operation can furthermore target only one syntactic position in the clause: the word cannot occur in any other place expect in the first position. In addition, if the word gets dislocated without the yes/no morpheme or any other overt left peripheral feature, it will receive a special contrastive interpretation and normally requires also prosodic stress (4).

(4)	MYY-DÄ$_1$	Pekka	halusi	__$_1$	koko	omaisuutensa.
	sell-A/INF	Pekka	wanted		all	possessions

‘It was selling, not loaning, that Pekka wanted to do with his possessions.’

Therefore, the dislocation does constitute idle rearrangement that speakers could do or left undone without being aware of the changes such operations induce also to the meaning of the sentence. We must capture these semantic changes in our model.

These special interpretational effects (focusing, interrogativization) and the features that trigger them (stress, wh-features, Q) belong to a larger left peripheral C-system that involves also phrasal operator movement (Huhmarniemi 2012; Vilkuna 1989, 1995). The operation in (1) and phrasal movement use the same position, as shown in (5a). The two operations are also complementary (5b), reinforcing the conclusion that we are looking at two variations of the same mechanism.

(5)	a.	[Koko	omaisuute-nsa-**ko**]$_1$	Pekka	halusi	myy-dä	__$_1$?
		all	possessions-PX/3SG-Q	Pekka.NOM	wanted	sell-A/INF	

‘Was it all his possessions that Pekka wanted to sell?’

b.	*Koko	omaisuute-nsa-**ko**	myydä-**kö**	Pekka	halusi?
	all	possessions-PX/3SG-Q	sell-A/INF-Q	Pekka	wanted

Since the same features and the same syntactic position is involved in head movement and phrasal movement, our model should capture properties of both.

The dislocated element must correspond to a gap in the clause, and its grammatical properties must match with the properties of that gap. If the gap is filled in by an independent element (6a), or if the dislocated word is an infinitival (here the control VA-infinitival) that does not match with the grammatical position of the gap (b), the outcome is unacceptable.

(6)	a.	*Myy-dä-kö	Pekka	halusi	**osta-a**	koko	omaisuutensa?
		sell-A/INF-Q	Pekka	wanted	buy-A/INF	all	possessions
	b.	*Myy-**vän**-kö	Pekka	halusi	_	koko	omaisuutensa?
		see-VA/INF-Q	Pekka	wanted		all	possessions

We can therefore be reasonably certain that the dislocated word in (1) has been "moved" in some sense from its canonical position to the first position of the clause in order to create the special interpretation associated with its noncanonical position and the C-feature. This dependency is illustrated in (7).

(7)	Myy-dä-**kö**	Pekka	halus-i	__	koko	omaisuute-nsa?
	sell-A/INF-Q	Pekka.NOM	wanted-PAST.3SG		all	possessions-PX/3SG

This observation motivates in part the term "head movement" used frequently of the phenomena of this type in the generative literature. Complex phonological words are able to dislocate as one unit. This dependency is regulated by syntactic constraints. For example, it is not possible to extract and move a phonologically complex word out of a subject (8).

(8)	*Myy-dä-kö$_1$	[Peka-n	suunnitelma	__$_1$	auto]	epäonnistui.
	sell-A/INF-Q	Pekka-GEN	plan		car	failed

Intended: 'Was it Pekka's plan to sell (and not to borrow) the car that failed?'

We must capture all restrictions of this type in our model. Notice that the noncanonical position of the dislocated complex word is not the literal first position of the sentence that we could arrive at by "counting words." In the example (9)a, the dislocated word is in the fourth position, counting from the beginning.

(9)	a.	Pekka	mietti	että	**myy-dä-kö**	Merja	haluaa	_	omaisuutensa.
		1.	2.	3.	4.				
		Pekka	wondered	that	sell-A/INF-Q	Merja	want-3SG		possessions

'Pekka wondered if Merja wanted to sell her possessions.'

Instead, the notion of "first position" refers to a left peripheral position in the CP-cartography that has well-defined syntactic and semantic properties, as discussed in the primary literature (e.g., Huhmarniemi 2012b; Rizzi 1997; Ross 1967; Vilkuna 1989). We can characterize it intuitively as the most prominent syntactic projection of its own finite clause. It is not crosslinguistically surprising that the first/most prominent position of the finite clause represents discourse and clause typing features of this kind.

The phonologically complex word constitutes a literal complex word in the sense that its parts can occur inside other words as well. We can attach both the A-infinitival morpheme and the Q-morpheme to virtually any verbal stem. A model in which the complex word constitutes an independent atom that cannot be broken down into discrete grammatical constituents does not match with the grammatical system employed by the speakers and would be unacceptable.

The yes/no morpheme -kO used in the examples above constitutes one left peripheral suffix that is involved in the Finnish C-system and its CP-cartography. There are several others, and the suffixes can be stacked agglutinatively. For example, a word such as *KUKA-ko-han* 'who.WH.FOC-Q-FAM' combines an interrogative pronoun (and its *wh*-feature), contrastive focus via prosodic stress, a yes/no particle and a familiarity topic particle, all left peripheral C-features that contribute their own unique semantic effects to the sentence. I have counted more than twenty possible C-features and their combinations, but a detailed study of the complete repertoire is beyond the scope of the present work; twelve features and their combinations were addressed explicitly in this study and appear in the test corpus.

1.3 Syntactic background model

The syntactic background model of the present study captures human linguistic behavior by modelling the information processing steps that occur when informants or participants are presented with linguistic expressions such as (1-9) and are asked to elicit responses, such as grammaticality/acceptability judgments and semantic interpretations. The approach seemingly differs from the more traditional grammatical theories, which model linguistic competence as a disembodied repository of linguistic knowledge (Chomsky 1965), but the difference is in one sense inconsequential, as the recognition model produces grammaticality judgments and semantic intuitions much like any competence model. In addition, the model generates similar and in many cases identical syntactic representations than those generated by standard cyclic theories. The idea that at least some universal properties of human language(s) could emerge from the comprehension process is not new, however (see, e.g., Cann, Kempson, and Marten 2005; Chesi 2012; Hale 2014; Merchant 2019; Phillips 1996).

We begin from the assumption that the human brain utilizes at least three components when interpreting and judging sentences presented to them via the sensory systems: a lexico-morphological component that hosts the lexicon and is responsible for lexical retrieval and morphological decomposition; a syntactic component determining how the words are organized in relation to each other; and a semantic component that creates semantic interpretations. It seems impossible to assume anything less, given the type of facts reviewed in Section 1.2. The components are illustrated in Figure 2, with further comments and refinements below.[3]

[3] The terminology used in this document and in the main article for the most part matches with the terminology used in the Python source code. See Section 3.6.

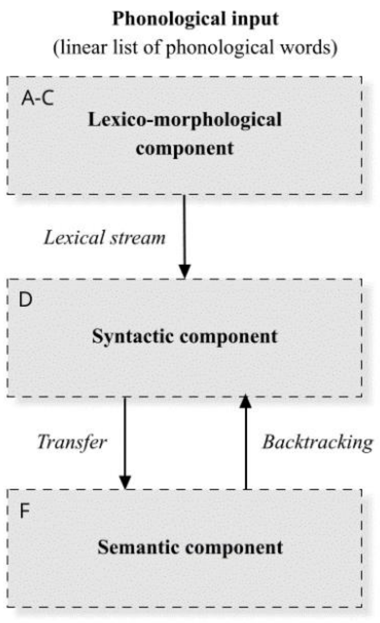


Figure 2. Components of the language faculty as assumed in the background theory.

The lexico-morphological component retrieves lexical information from the lexicon on the basis of the phonological words recognized in the input string. Each word is processed through the lexico-morphological and syntactic modules immediately after it arrives from the sensory systems, thus the model is incremental (Altmann and Kamide 1999; Altmann and Steedman 1988; Demberg, Keller, and Koller 2013; Keller 2010; Konieczny 2000; Marslen-Wilson 1973; Phillips 2003; Tanenhaus et al. 1995). The lexico-morphological module and the syntactic module are connected via a *lexical stream*, which sends lexical information ('words and their parts') from the former to the latter, in a linear event sequence. Taking sentence (1) as an example, this component is responsible for recognizing the internal parts of the complex word *myy-dä-kö* 'sell-A/INF-Q' and for forwarding this information in a well-defined sequence to the syntactic component.

How the lexico-syntactic interface and the lexicon works is a controversial empirical question that cannot be answered satisfactorily without hypothesizing something and then testing the consequences of that hypothesis. In the present work, lexical entries for morphologically complex words constitute linear sequences of derivational and inflectional elements that for the most part correspond directly to their surface decompositions (*myy-dä-kö* ~ sell#A/INF#Q). Lexical entries for the primitive lexical items (again derivational or inflectional elements) are sets of features (e.g., sell = {V, 'sell', /sell/, …}) without further structure. These assumptions provide what I consider to be a minimal starting point, in that it is difficult to imagine using anything less while engaging meaningfully with the data presented in Section 1.2. Linguistic features and the primitive lexical items assembled from them constitute the ultimate cognitive primitives of the theory. Lexical features are simple (neuronal) patterns that can be processed by elementary pattern matching principles. Two languages can differ in the way they assemble their lexicons, capturing one important source for language-specificity.

The syntactic module attempts to configure the incoming lexical items into a hierarchical representation. For example, the sentence *John saw the girl with a telescope* has two readings corresponding to two hierarchical arrangements of its lexical items, namely [*John saw* [$_{\text{DP}}$ *the girl with a telescope*] and [*John* [$_{\text{VP}}$ *saw* [*the girl*] *with a telescope*]]. Lexical items arriving to the syntactic component from the lexico-morphological module (Figure 2) are therefore attached to a partial phrase structure representation in the current syntactic working memory, which represents the syntactic interpretation developed by the algorithm on the basis of words it has seen up to that point.[4] This process is illustrated in Figure 3.

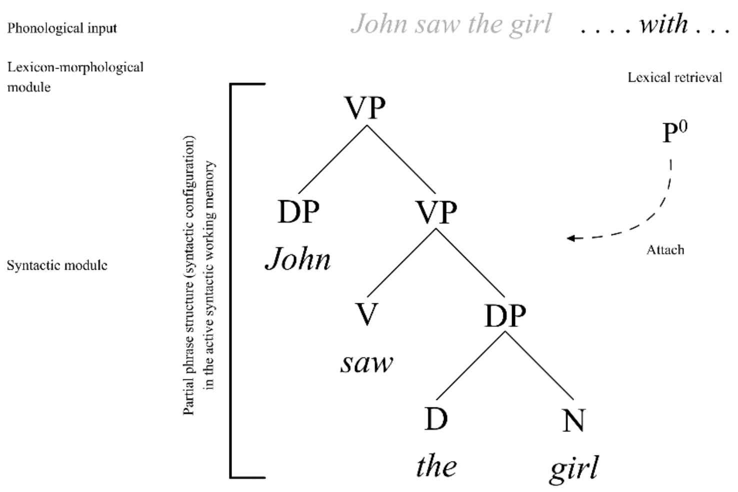

Phonological input
John saw the girl . . . . with . . .
Lexicon-morphological module
Lexical retrieval
Syntactic module
Partial phrase structure (syntactic configuration) in the active syntactic working memory
VP
DP
John
VP
V
saw
DP
D
the
N
girl
P0
Attach

Figure 3. Incremental attachment of incoming lexical items to the current partial phrase structure representation in the active syntactic working memory.

Let us assume that if the output of the syntactic component cannot be interpreted, the solution is rejected by the semantic component and the outcome is communicated back to the syntactic component, which reacts by trying to find alternative solutions by recursive backtracking. This creates garden path effects

[4] The underlying notion of phrase structure is an asymmetric binary branching bare phrase structure, very closely related to the one proposed in Chomsky 1995:224ff, 2008. Incoming lexical items are merged, in left-to-right order, to the right edge of the partial phrase structure. Call the operation Merge^{-1}. A model of this type was first proposed by Phillips (1996, 2003). Suppose the incoming lexical item is β; then Merge^{-1} targets some node α at the right edge of the existing phrase structure and substitutes it with $\gamma = [\alpha\ \beta]$, the latter created by standard cyclic Merge (hence Merge^{-1} = target + Merge + substitute). The label (lexical category label) of any phrase is the most prominent primitive lexical item in it. Suppose γ is a syntactic object created by Merge^{-1}, then the label of primitive γ is γ; otherwise suppose $\gamma = [\alpha\ \beta]$ and that if α is primitive, then it is the label; otherwise, if β is primitive it is the label; otherwise we apply the rule recursively to β. The notion of phrase structure used in this study is formalized in the module *phrase_structure.py*. Labelling algorithm is defined by function *head()*.

seen in the case of sentences such as *the horse raced past the barn fell*.[5] The same mechanism detects ambiguities: after finding an interpretation for *John saw* [*the girl with the telescope*] where the girl has the telescope, backtracking allows the syntactic component to find others (namely, one where John had it). Thus, there is a backward pointing arrow "backtracking" in the model in Figure 2 that goes from the semantic system to the syntactic component. When the model backtracks, the operation consumes more computational resources and time, predicting processing delays in the case of real human language comprehension.

We will assume that English-speakers have access to the same cognitive capacities and conceptual representations as Finnish-speakers, and vice versa. If this were not the case, then translation from one language into another would not be possible. The assumption seems to be an implicit or explicit part of virtually all cognitive theories of human reasoning and cognition, and in linguistics too it is commonly assumed that the conceptual systems and the interface between language and meaning "must be universal, in that any thought expressible in a human language is representable in it" (Chomsky 1995:18). The sensorimotoric properties of linguistic expressions do, however, vary depending on language. Some properties of (1-9) are not attested in English. The model (in fact, any language comprehension model that aims to predict semantic interpretations) must involve a component which sanitizes the input from language-specific sensorimotoric noise. That process is called *transfer* in the present model and takes place when a syntactic object is sent form the syntactic component to the semantic system for interpretation; see the arrow "transfer" in Figure 2. Properties of transfer are crucial for our analysis of (1-9) and will be elucidated later.

If the input cannot be mapped into any legitimate syntactic object, the model will judge the input ungrammatical. If the input is judged grammatical, then one or several syntactic analyses are generated which are linked with semantic interpretations, both which are checked for correctness by comparing them with native speaker intuitions. I will explain how this verification is performed later in this document. In addition, if we were interested in psycholinguistic realism, we would compare the performance properties of the model with those collected from native speakers in controlled experiments; this aspect will be elucidated briefly in Section 3.1.

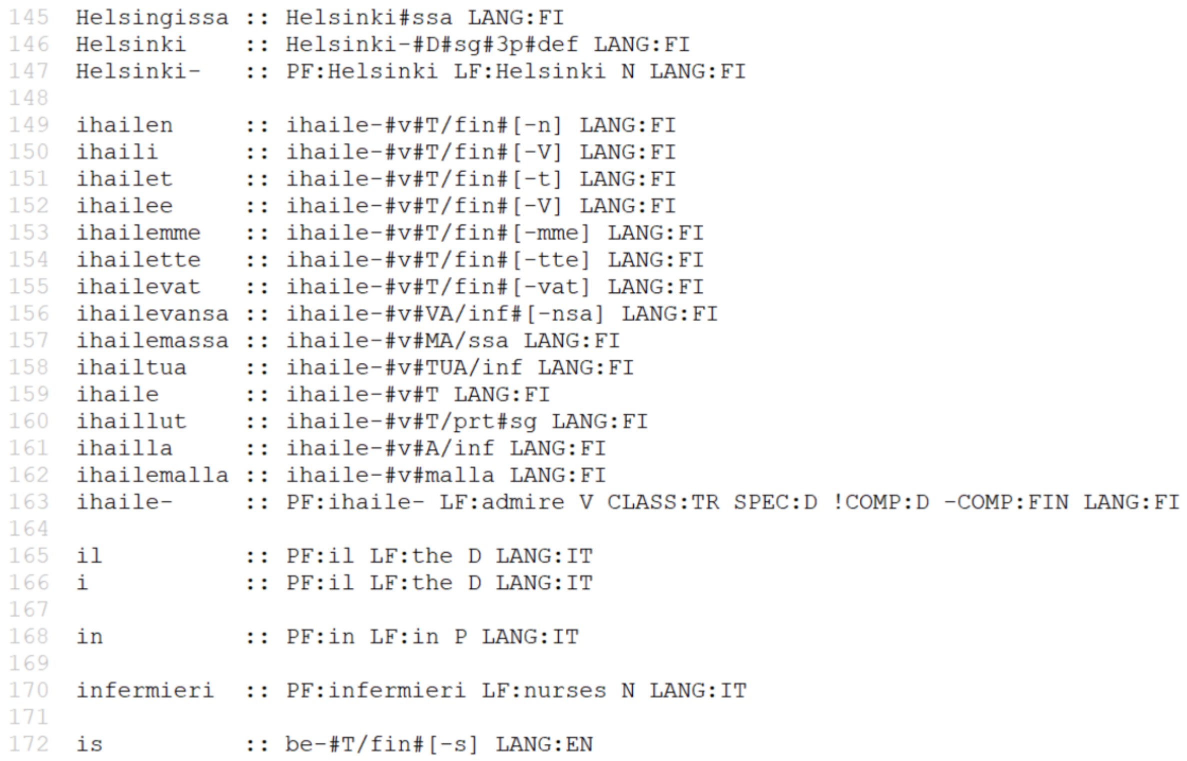
The model must be able to determine if the words in the input match with their grammatical contexts created in the syntactic component. The following variations of (1), in which we use the VA-infinitival instead of the A-infinitival, are ungrammatical.

(10) a.	*Myy-**vän**-kö	Pekka	halusi	__	koko	omaisuutensa?
	see-VA/INF-Q	Pekka	wanted		all	possessions

[5] The first pass parse [*the horse* [*raced* [*past the barn*]]] + *fell* will be rejected because the last word *fell* cannot be integrated into the existing structure. The model backtracks until a well-formed representation ('the horse, which raced past the barn, fell') is found.

b.	*Pekka	halusi	myy-**vän**	koko	omaisuutensa.
	Pekka	wanted	sell-VA/INF	all	possessions

Intuitively the words do not “match” with each other, either syntactically, semantically, or perhaps by a mixture of both criteria. We therefore assume that the primitive lexical items coming out of the lexico-morphological system contain lexical features which define what the admissible local grammatical contexts of the words are. For example, the verb *halusi* ‘wanted’ contains a lexical selection feature which allows it to select for a phrase headed by an A-infinitival (the corresponding English selection rule would be ‘want + to sell’). We can then determine that some phrases *cannot* occur in this position by using negative selection features or by assuming that only positively selected phrases are possible. For example, the finite verb ‘want’ cannot select for a control VA-infinitival (**Pekka halusi* ***myy-vän*** *omaisuutensa* ‘Pekka wanted sell-VA/INF possessions’). The lexicon is provided as an external parameter to the simulation, currently in three separate files which contain the primary language-specific lexicon, universal lexical redundancy rules, and a third file containing universal lexical elements or grammatical-functional morphemes. The lexicon is discussed further in Section 3.5. These three sources are synthesized into one lexicon at runtime, and separate lexicons are created for the speakers of each language. Figure 4 contains a screenshot from the file hosting the language-specific lexicon.



```
Helsingissa :: Helsinki#ssa LANG:FI
Helsinki    :: Helsinki-#D#sg#3p#def LANG:FI
Helsinki-   :: PF:Helsinki LF:Helsinki N LANG:FI

ihailen     :: ihaile-#v#T/fin#[-n] LANG:FI
ihaili      :: ihaile-#v#T/fin#[-V] LANG:FI
ihailet     :: ihaile-#v#T/fin#[-t] LANG:FI
ihailee     :: ihaile-#v#T/fin#[-V] LANG:FI
ihailemme   :: ihaile-#v#T/fin#[-mme] LANG:FI
ihailette   :: ihaile-#v#T/fin#[-tte] LANG:FI
ihailevat   :: ihaile-#v#T/fin#[-vat] LANG:FI
ihailevansa :: ihaile-#v#VA/inf#[-nsa] LANG:FI
ihailemassa :: ihaile-#v#MA/ssa LANG:FI
ihailtua    :: ihaile-#v#TUA/inf LANG:FI
ihaile      :: ihaile-#v#T LANG:FI
ihaillut    :: ihaile-#v#T/prt#sg LANG:FI
ihailla     :: ihaile-#v#A/inf LANG:FI
ihailemalla :: ihaile-#v#malla LANG:FI
ihaile-     :: PF:ihaile- LF:admire V CLASS:TR SPEC:D !COMP:D -COMP:FIN LANG:FI

il          :: PF:il LF:the D LANG:IT
i           :: PF:il LF:the D LANG:IT

in          :: PF:in LF:in P LANG:IT

infermieri  :: PF:infermieri LF:nurses N LANG:IT

is          :: be-#T/fin#[-s] LANG:EN
```

Figure 4. Screenshot from the file containing the language-specific lexicon. Phonological exponents matched from the input are on the left, morphological decompositions (if available) or feature sets of primitive lexical items on the right. The two must be separated by symbol “::”. Feature LANG:L determines the language of the item.

The Finnish agglutinative profile is modelled by assuming that the entries in this list can have complex internal structure. The entry for the verb *myy-dä-kö* 'sell-A/INF-Q' in the example (1), for example, contains four morphemes: the verb stem, a transitivizer morpheme v, the A-infinitival morpheme, and the yes/no particle. Each then points to another entry in the same lexicon which, if primitive, contains only a feature set and does not decompose further (see for example line 163 in Figure 4 above).[6]

The question of whether the lexical features that participate in the process of determining admissible grammatical contexts for words are semantic or syntactic (or most likely, both) has always remained unclear and subject to debate. In a fully computational model any feature that plays a computational role must have a formal shape which ultimately determines what it does in the system; there cannot be immaterial semantic features that have no formal properties. A semantic feature is therefore a feature that has both syntax and semantics, while a purely syntactic feature would be something that is completely invisible outside of the syntactic processing pathway. Both features exist in the model.

We can imagine at least three ways of interpreting a computational model of this type, borrowing an influential proposal from David Marr (1982). One is to consider it as a research tool that is employed to verify if some proposed hypothesis concerning grammatical competence is correct. This corresponds to Marr's notion of an "abstract computational theory," in which "the performance of the device [here the human brain] is characterized as a mapping from one kind of information to another" (p. 24), in the present case from sensory inputs into grammaticality judgments and semantic interpretations. Accordingly, the "grammar as a whole can thus be regarded as, ultimately, a device for pairing phonetically represented signals with semantic interpretations, this pairing being mediated through a system of abstract structures generated by the syntactic component" (Chomsky 1964: 9-10). This would represent a pure competence theory: description of what is possible in human language. Marr proposed however that we go further and consider also what kind of representations and algorithmic transformations the system uses in performing the mapping, and further how the algorithm is implemented in the human brain. These constitute the two other levels of description in Marr's theory, the algorithmic level and the implementation level. Using the former, we would consider that the components present in Figure 2 describe real information processing pathways that are activated in the human brain during language comprehension. Pursuing an implementation theory would require that all functions in the formal model be written in such a way that they would also constitute a model of the underlying physiological brain processes. My own interpretation of the background theory identifies it with Marr's first and second levels but not with the third: it is written as a literal model of human

[6] The lexico-morphological component must be able to perform morphological parsing. Morphological parsing has not been implemented at present writing, but the user can simulate it by providing morphological decompositions directly in the input. Thus, the model can read both *admires* and *admire#3p#sg*.

language comprehension that respects competence data, but remains agnostic concerning neuronal implementation, of which very little is known at present that could be presented as functional code.

2 Head reconstruction analysis

2.1 Overview

Let us consider next the head reconstruction analysis proposed in the main article. The background model elucidated in the previous section can already process phonologically complex words, in fact it was used in one published study (Brattico and Chesi 2020). The model attaches incoming lexical components automatically into a phrase structure in the order they appear in the input (11). Inflectional morphemes, such as the third person singular feature '3sg' in the example below, are inserted as features inside adjacent morphemes during lexical streaming.

(11)	a.	John+	admires +	Mary	(Input)
	b.	John	admire-#v#T#3sg	Mary	(Lexical decompositions)
	c.	John,	T, 3sg, v, V,	Mary	(Lexical stream)
	c.	[John	[T$_{[3SG]}$ [v [V	Mary]]]	(Attachment)

This model corresponds to a hypothesis in which complex phonological words are created and processed inside the lexico-morphological module (or what is called "phonology" in the standard theory) and are invisible to the syntactic and semantic components. This idea was discussed briefly in Chomsky 2001 and then developed in further work (Platzack 2013; Schoorlemmer and Temmerman 2012). The model is not implausible and produces useful results in a range of cases. It was noted immediately during the abovementioned study however that this assumption runs into problems with standard head movement constructions such as English aux-inversion. The calculated output in such cases is (12).

(12) Does +	John +	admire +	Mary?	
[T	[John	[v [V	Mary]]]	

The fronted auxiliary will discharge its internal lexical constituents to the highest position in the clause, matching its position in the left-to-right order. The problem is that the EPP-feature of T remains unchecked. These cases were handled by ad hoc rules,[7] but the approach is insufficient and quite likely incorrect due to the Finnish long head movement constructions (1), repeated here as (13).

[7] One solution is to reconstruct the postverbal subject upwards into SpecTP to check the EPP. This requires an anomalous operation that moves phrases from their canonical positions into formal EPP positions, in traditional terms. A second possibility is to relativize the EPP requirement, perhaps by nullifying it under some circumstances. Yet, the EPP principle seems to be operational in these verb-initial clauses: they do not become 'subjectless'.

(13)	**Myy-dä-kö**$_1$	Pekka	halus-i	__$_1$	koko	omaisuute-nsa?
	sell-A/INF-Q	Pekka.NOM	wanted-PAST.3SG		all	possessions-PX/3SG

‘Was it selling that Pekka wanted to do to all his possessions?’

Discharging the elements of the first word to the beginning of this clause produces complete gibberish, and it is also clear that some of these internal components “should” go into the position marked by the gap. This observation formed the background for the present study. Empirical investigation of (13) showed furthermore that the operation that dislocates the word to the first position of the clause follows syntactic constraints. This observation and the data cited in the main article suggested, then, that the syntactic component has access to some type of head reconstruction (head movement in the standard theory).

The sentence is, furthermore, interpreted so that the speaker has specifically targeted the lexical content of the complex predicate ‘to sell’ for a special left peripherical yes/no interpretation (‘is it selling…’), further pointing towards the possibility that the language faculty has access to a syntactic-semantic notion of ‘complex predicate’ that can be targeted for a cleft interpretation independently of its possible role within some larger phrase. Standard phrase structure objects created by merge or attachment presuppose, however, that the individual lexical nodes constitute separate phonological words, and that merging of two such units generates phrases. If the language faculty has access to a notion of complex predicate or complex head, then there must exist a syntactic configuration that does not behave like a syntactic phrase.

Before moving on with this proposal, it should perhaps be pointed out that this conclusion is going to be theoretically rather costly, requiring us to enrichen the standard phrase structure theory in some way. In addition, several possible solutions that can achieve the same outcome exist in the literature or could be easily imagined, some of which are more costly than others. Yet, it is hard to decide between them unless we run simulation tests. I therefore decided to adopt what could be considered an “engineering solution” in order to experiment with complex heads. This is the rationalization for using theory-neutral notation X(Y) for a complex head X that contains another head Y, leaving the implementation open for further experimentation and implementation.[8] The analysis and the output of the algorithm would remain the same independent of how X(Y) is actually implemented in the phrase structure geometry.

[8] The standard assumption at present writing seems to be a system in which complex heads are ordinary phrase structure objects specifically marked for being heads. Thus, Chomsky 1995:225 distinguishes terminal elements that are lexical feature bundles from zero-level categories X^0 that may also be phrasal, corresponding to complex heads in the present model. Thus, $[_{DP}$D N] would differ from $[_{N}$D N]. I found this proposal impossible to implement, leading to complications and empirically incorrect outcomes. It remains within the realm of empirical possibility, of course, and could be tested in some future study.

To add this option to the model, then, the notion of syntactic phrase was first defined so that a constituent K is considered a *syntactic phrase* if and only if it has exactly two daughter constituents and a *primitive nonphrasal constituent* otherwise. A syntactic phrase is an object that is in the domain of phrasal syntactic operations. A constituent that has no daughter constituents is, therefore, a primitive nonphrasal constituent, but it has a further property of being also a *terminal constituent*. This leaves room for a constituent that has only one daughter, which makes it a primitive nonterminal. These entities were used in this study to represent complex heads. They are designated by X(Y), Y being the sole daughter of X. The rule applies recursively, allowing the algorithm to wrap a linear sequence of heads inside another head.[9] Word-internal lexical items are attached to the structure in this way when they arrive from the lexico-morphological component to the syntactic component. The result is (14).

(14)	John	+	admires	+	Mary	(Input)
	John,		T, v, V,		Mary	(Lexical stream)
	[John		[T(v, V)^0		Mary]]	(Attachment)

Having admitted elements like T(v, V)^0 into the phrase structure, we can consider their reconstruction. Head reconstruction is executed during transfer (see Figure 2). The algorithm travels downstream on the right edge of the phrase structure in the active working memory, following labelling and selection, targets complex heads X(Y) and reconstructs Y downstream into a position [H [Y…]] in which it can be selected from above by H based on lexical features, adopting the first and hence most local solution available.[10] If no position is available, a last resort solution [X [Y…]] is adopted. X remains in situ, thus in the position where it was merged during left-to-right parsing or backtracking.[11] The algorithm then continues downwards. Thus, if Y is itself a complex head, it will be targeted later. All complex heads are reconstructed, in the manner shown in (15).[12] Notice that Q will remain in the left periphery,

[9] This notation makes room for primitive constituents that contain phrasal constituents. This was originally considered a potentially useful way of modelling clitics and incorporation. Perhaps it could be used to redefine the notion of specifier, so that [XP [H YP]] could be represented as [H(XP)^0 YP]. All this is speculation; there are no lexical or phrasal syntactic operations in the current implementation that have access to word-internal phrasal constituents.

[10] If X(Y) is the bottom right constituent, it is reconstructed as [X Y]. An additional edge case complication is presented by a situation in which the head that is needed for selection resides inside the bottom right node. This was solved by reconstructing the bottom right node before the reconstructing higher head is fitted into the structure.

[11] This means that the model cannot process null or idle head movement, in which (stated from the point of view of a standard cyclic theory) a head moves without merging with a higher head that has independent syntactic and/or semantic content.

[12] There is no axiom which prevents complex heads from occurring at the syntax-semantic interface. If a complex head X(Y) survives reconstruction, Y will behave at the syntax-semantics interface as if it were not present at all and the representation crashes because some selection rules involved with Y cannot be satisfied and/or because other reconstruction rules, namely those which depend on Y, fail or produce gibberish.

representing the fact that the sentence has interrogative force. It is also used for representing the propositional scope of the operation, as elucidated in Section 2.5.

(15)	Myy-dä-kö	Pekka	halus-i	_		koko	omaisuute-nsa?
	Q(__)	Pekka.NOM	wanted-PAST.3SG	A/INF(__)	V	all	possessions-px/3sg

At the level of the source code the analysis is formalized in the module head_reconstruction.py. Once this mechanism is in place, it is possible to perform meaningful calculations by feeding the model with sentences that contain phonologically complex words.

Looked from an inverse point of view, the system appears to be compressing syntactic structures into words. Notice how the inverse of (15) literally removes both heads and phrasal positions from the structure. In (15), for example, the spellout structure will have 'all possessions' at the (wrong) complement position of 'want' due to grammatical compression, and a much simpler object is submitted to the sensorimotoric interface than would be the case if these operations were not done.

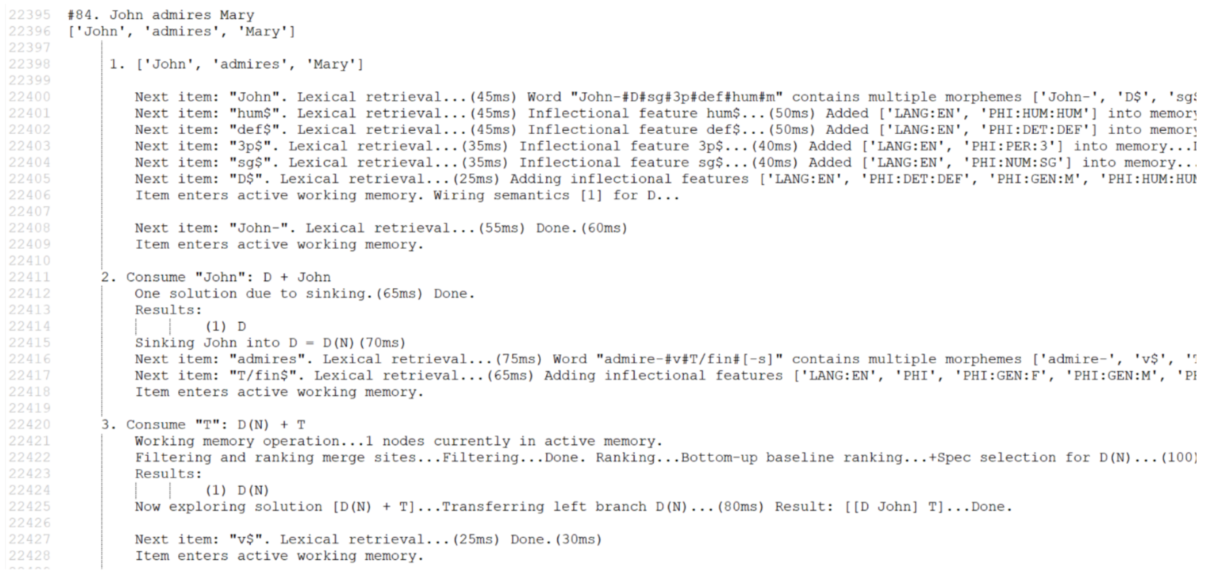
Some amount of skepticism, most of it justified in my view, exists in the literature against giving the UG access to complex heads. My own first attempt at using this model for grammatical analysis was based on the hypothesis that complex heads exist only inside the lexico-morphological system ("phonology"). The idea was to break complex words into their constituent parts inside the lexico-morphological component and then feed these parts into syntax in some order that would have resulted plausible phrase structure objects. But perhaps this is a misjudgment, or an accidental feature of focusing on relatively isolative Indo-European languages. Finnish is a polysynthetic and agglutinative language, and thus if we examined this language group more closely we could potentially find that the complex head/complex predicate strategy is as natural and widespread as the isolative one. Thus, perhaps complex heads do represent grammatical natural kinds in the same way as phrases do. There could be some cognitive or biological factor that makes sensorimotoric compression a natural disposition of the biological system. Still, for a substantial range of cases the output of head reconstruction is identical to the earlier model in which lexical decomposition was performed inside the lexico-morphological component. This makes these concerns still relevant.[13]

2.2 Simple sentences

We will begin the more detailed examination by looking at the derivation of simple sentences such as English transitive clause *John admires Mary* (sentence number #84 in the test corpus and output). To get a sense of how the model calculates the data, we will examine the detailed derivational log file that

[13] One possibility is that both principles are in operation: some words are mapped into complex heads while others are decomposed presyntactically. Why is it that long head movement only exists in the C-system? Are there examples in the empirical literature suggesting that the presyntactic model fails when processing heads in some other grammatical context?

contains a record of the linguistically relevant computational steps executed during the processing of the sentence. Figure 5 shows the initial part of the derivation (the file is available online).



```
22395 #84. John admires Mary
22396 ['John', 'admires', 'Mary']
22397
22398      1. ['John', 'admires', 'Mary']
22399
22400         Next item: "John". Lexical retrieval...(45ms) Word "John-#D#sg#3p#def#hum#m" contains multiple morphemes ['John-', 'D$', 'sg
22401         Next item: "hum$". Lexical retrieval...(45ms) Inflectional feature hum$...(50ms) Added ['LANG:EN', 'PHI:HUM:HUM'] into memor
22402         Next item: "def$". Lexical retrieval...(45ms) Inflectional feature def$...(50ms) Added ['LANG:EN', 'PHI:DET:DEF'] into memor
22403         Next item: "3p$". Lexical retrieval...(35ms) Inflectional feature 3p$...(40ms) Added ['LANG:EN', 'PHI:PER:3'] into memory...
22404         Next item: "sg$". Lexical retrieval...(35ms) Inflectional feature sg$...(40ms) Added ['LANG:EN', 'PHI:NUM:SG'] into memory..
22405         Next item: "D$". Lexical retrieval...(25ms) Adding inflectional features ['LANG:EN', 'PHI:DET:DEF', 'PHI:GEN:M', 'PHI:HUM:HU
22406         Item enters active working memory. Wiring semantics [1] for D...
22407
22408         Next item: "John-". Lexical retrieval...(55ms) Done.(60ms)
22409         Item enters active working memory.
22410
22411     2. Consume "John": D + John
22412         One solution due to sinking.(65ms) Done.
22413         Results:
22414                 (1) D
22415         Sinking John into D = D(N)(70ms)
22416         Next item: "admires". Lexical retrieval...(75ms) Word "admire-#v#T/fin#[-s]" contains multiple morphemes ['admire-', 'v$', '
22417         Next item: "T/fin$". Lexical retrieval...(65ms) Adding inflectional features ['LANG:EN', 'PHI', 'PHI:GEN:F', 'PHI:GEN:M', 'P
22418         Item enters active working memory.
22419
22420     3. Consume "T": D(N) + T
22421         Working memory operation...1 nodes currently in active memory.
22422         Filtering and ranking merge sites...Filtering...Done. Ranking...Bottom-up baseline ranking...+Spec selection for D(N)...(100
22423         Results:
22424                 (1) D(N)
22425         Now exploring solution [D(N) + T]...Transferring left branch D(N)...(80ms) Result: [[D John] T]...Done.
22426
22427         Next item: "v$". Lexical retrieval...(25ms) Done.(30ms)
22428         Item enters active working memory.
```

Figure 5. Screenshot from the initial part of the derivational log file containing the computations for sentence *John admires Mary*. See the main text for explanation.

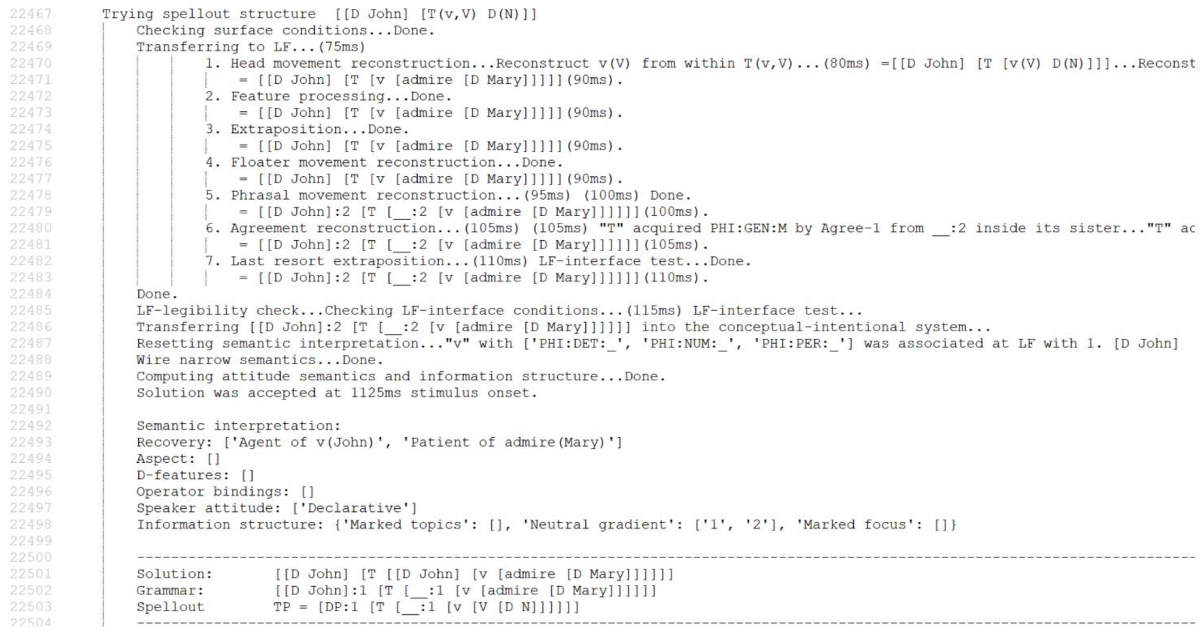
Each numbered step represents the processing of one lexical item. For example, *John* is decomposed into John#D (N#D) in the lexicon, followed by a list of inflectional features such as 'singular' or 'human' (lines 22400-22405) that we can ignore here but which are involved in the agreement system and in the creation of referential indexes and the semantic discourse inventory. The D-element arrives to syntax on line 22406, at which point it already contains all inflectional features inserted into it during lexical streaming (line 22405). It is followed by the noun stem (line 22411). At this point, the active working memory contains D, while N is waiting in the queue to be attached. Since D and N were inside the same phonological word, N will be attached inside D = D(N) (lines 22412-22415). The operation is called "sinking" at the level of the source code.[14] The rest follows the same formula: an item arrives, the syntactic module considers how to attach it, and the iteration continues until all words have been consumed and attached somewhere. After all words have been consumed, the syntactic component ends up with (16) (line 22467).

(16) John admires Mary
$[D(N)^0 \quad [T(v,V)^0 \quad D(N)^0]]$

[14] To recapitulate, phrasal Merge-1 targets α and β and yields [α β], whereas sinking yields [$_{\alpha}$ β]. The theoretical cost associated with the latter is obvious at the level of source code: we must write separate code for phrasal merge and sinking when both operations are possible.

This representation can be considered a first pass parse, a solution the parser considers the most plausible after reading all words once. It is a fairly transparent and simple syntactic representation for the input sentence. This transparency results from the new mechanism that creates complex heads, in that phonological words at the sensory level are mirrored by complex heads inside syntax. Solutions generated by the left-to-right attachment operation are called *spellout structures* because they are transparently related to sensory objects. Spellout structure are also highly ‘compressed’ structures under the present approach, which is necessitated by the rather austere amount of sensory information available during the first steps of the derivation.

There is another possible solution, γ = [$_{DP}$ [$_{VP}$ *John admires*] *Mary*], which the model rejects. All possible solutions are ranked in terms of their psycholinguistic plausibility, and the rankings are used during backtracking. The ranking is based on psycholinguistic heuristic principles that the user can activate or inactivate freely (and available in the module plausibility_metrics.py). All heuristic principles currently available were used in the simulations reported in this study. If we examine however what happened during the generation of γ, we find that the parser rejected this solution as impossible (line 22453), thus it was not inserted into the search tree. This is because [$_{DP}$[$_{VP}$ *John admires*] *Mary*] is so ill-formed that it can be rejected locally without further consideration. When a parsing path is closed off locally, we say that it is *filtered*. Filtering and ranking are the two mechanisms the parser uses to navigate through the labyrinth of possible derivations. Notice that filtered solutions cannot be targeted later by backtracking; they are closed off permanently. Once a solution has been generated, however, it will be sent off to the semantic component for evaluation via transfer. These operations are visible in the derivational log file and shown in Figure 6.



```
22467       Trying spellout structure  [[D John] [T(v,V) D(N)]]
22468       |   Checking surface conditions...Done.
22469       |   Transferring to LF...(75ms)
22470       |   |   |   1. Head movement reconstruction...Reconstruct v(V) from within T(v,V)...(80ms) =[[D John] [T [v(V) D(N)]]]...Reconst
22471       |   |   |   |   = [[D John] [T [v [admire [D Mary]]]]](90ms).
22472       |   |   |   2. Feature processing...Done.
22473       |   |   |   |   = [[D John] [T [v [admire [D Mary]]]]](90ms).
22474       |   |   |   3. Extraposition...Done.
22475       |   |   |   |   = [[D John] [T [v [admire [D Mary]]]]](90ms).
22476       |   |   |   4. Floater movement reconstruction...Done.
22477       |   |   |   |   = [[D John] [T [v [admire [D Mary]]]]](90ms).
22478       |   |   |   5. Phrasal movement reconstruction...(95ms) (100ms) Done.
22479       |   |   |   |   = [[D John]:2 [T [__:2 [v [admire [D Mary]]]]]](100ms).
22480       |   |   |   6. Agreement reconstruction...(105ms) (105ms) "T" acquired PHI:GEN:M by Agree-1 from __:2 inside its sister..."T" ac
22481       |   |   |   |   = [[D John]:2 [T [__:2 [v [admire [D Mary]]]]]](105ms).
22482       |   |   |   7. Last resort extraposition...(110ms) LF-interface test...Done.
22483       |   |   |   |   = [[D John]:2 [T [__:2 [v [admire [D Mary]]]]]](110ms).
22484       |   Done.
22485       |   LF-legibility check...Checking LF-interface conditions...(115ms) LF-interface test...
22486       |   Transferring [[D John]:2 [T [__:2 [v [admire [D Mary]]]]]] into the conceptual-intentional system...
22487       |   Resetting semantic interpretation..."v" with ['PHI:DET:_', 'PHI:NUM:_', 'PHI:PER:_'] was associated at LF with 1. [D John]
22488       |   Wire narrow semantics...Done.
22489       |   Computing attitude semantics and information structure...Done.
22490       |   Solution was accepted at 1125ms stimulus onset.
22491       |
22492       |   Semantic interpretation:
22493       |   Recovery: ['Agent of v(John)', 'Patient of admire(Mary)']
22494       |   Aspect: []
22495       |   D-features: []
22496       |   Operator bindings: []
22497       |   Speaker attitude: ['Declarative']
22498       |   Information structure: {'Marked topics': [], 'Neutral gradient': ['1', '2'], 'Marked focus': []}
22499       |
22500       |   --------------------------------------------------------------------------------------------------------------------
22501       |   Solution:       [[D John] [T [[D John] [v [admire [D Mary]]]]]]
22502       |   Grammar:        [[D John]:1 [T [__:1 [v [admire [D Mary]]]]]]
22503       |   Spellout        TP = [DP:1 [T [__:1 [v [V [D N]]]]]]
22504       |   --------------------------------------------------------------------------------------------------------------------
```

Figure 6. Screenshot from the derivational log file containing transfer. The candidate solution is transferred to the semantic component (syntax-semantic interface), where it is evaluated (and in this case, accepted). Transfer involves several operations, numbers from 1-7 on lines 22470-22483, including the head reconstruction algorithm proposed in this study (line 22470).

The result is that all complex head reconstruct (line 22471) and the grammatical subject is reconstructed to its canonical thematic position SpecvP (lines 22478-9).[15] The analysis returned by the model is (17), also shown in Figure 7. Notice that phrasal labels are invisible in the output; they are determined dynamically by the labeling algorithm (by the function head() in the module phrase_structure.py).

(17) [$_{TP}$ [$_{DP}$ D *John*]$_1$ [$_{TP}$ T [$_{vP}$ __$_1$ [$_{vP}$ v [$_{VP}$ *admire* [$_{DP}$ D *Mary*]]]]]]

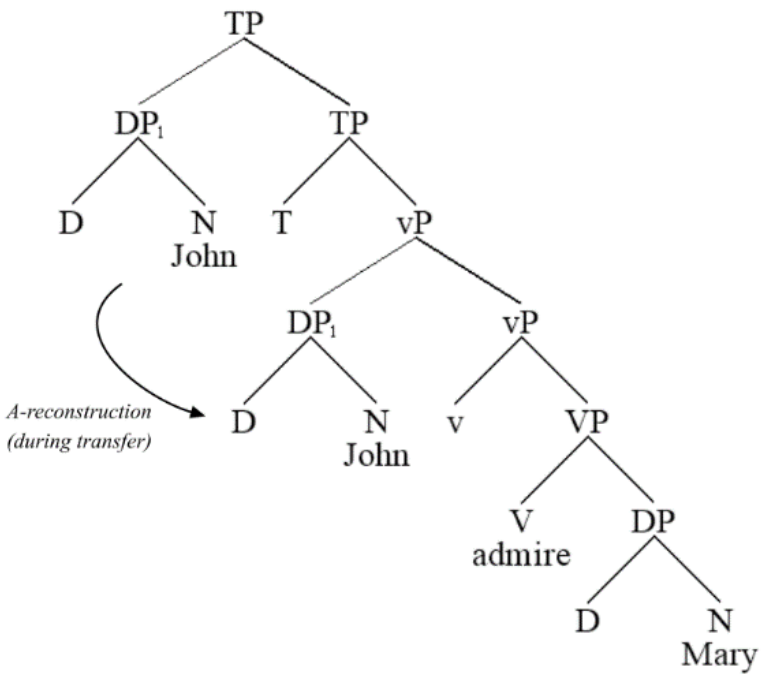


Figure 7. Output of the model for sentence *John admires Mary*.

Early experiments with this system revealed an interesting and unanticipated problem that concerns the way the parser processes complex heads. The matter is not directly relevant to the grammatical analysis but is relevant to the question of how a psycholinguistically realistic parser processes complex heads. Any solution [X(Y) + Z] branches into two possible derivations (a) [[X Y] Z] and (b) [X [Y Z]]. Nominals such as DP in general follow the former template (e.g., [[$_{DP}$ *John*] T...]) while verbal/predicate spreads the latter ([T [V...]]). The problem is that transfer is a one fell swoop cognitive reflex that cannot

[15] By A-reconstruction, which represents inverted A-movement. The correct algorithmic formal definition for A-movement has proven elusive. It is easy to formulate principles that work in connection with limited datasets but fail when tested with more representative samples. This difficulty stems in part from the fact that I have Finnish sentences in the larger datasets, but the topic of Finnish A-movement remains largely unexplored (no published studies exist, to my knowledge). The current algorithm is in the function A_reconstruct_() inside module phrasal_reconstruction.py.

branch. This required that some amount of parsing intelligence is added to head reconstruction so that a decision can be made. A left branch expansion (a) is created immediately upon [X(Y) + Z] by applying head reconstruction to X(Y), while the right edge expansion (b) is performed later when the same algorithm is applied to the hosting phrase structure object. The choice depends on whether the outcome generates something plausible. This mechanism explains why the grammatical subject argument *John* is reconstructed into [[D N]…] in (14) while the verbal spread creates a right branch.[16]

The model reacts automatically to all overt elements in the input. For example, in Finnish the sentential negation is marked by an auxiliary-like negative particle *e-* that occurs above the tense verb and exhibits full phi-agreement with a grammatical subject, if present (18). This sentence is number #8 in the simulation output.

(18)	Pekka	e-i	ihaile	Merja-a. (#8)
	Pekka.NOM	not-3SG	admire	Merja-PAR
	'Pekka does not admire Merja.'			

The output of the model is illustrated in Figure 8. Notice that the model will respond automatically to the presence of the negative word *e-* 'not' in the input by applying the same mechanisms as elucidated above: the word is retrieved from the lexicon and its components are forwarded to the syntactic module in the order they appear in the input and in the lexical decomposition (if any).

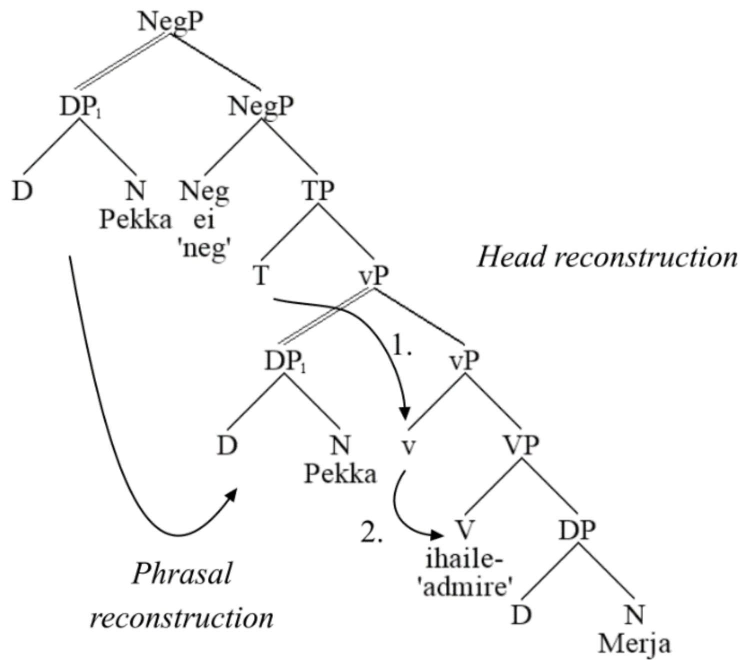
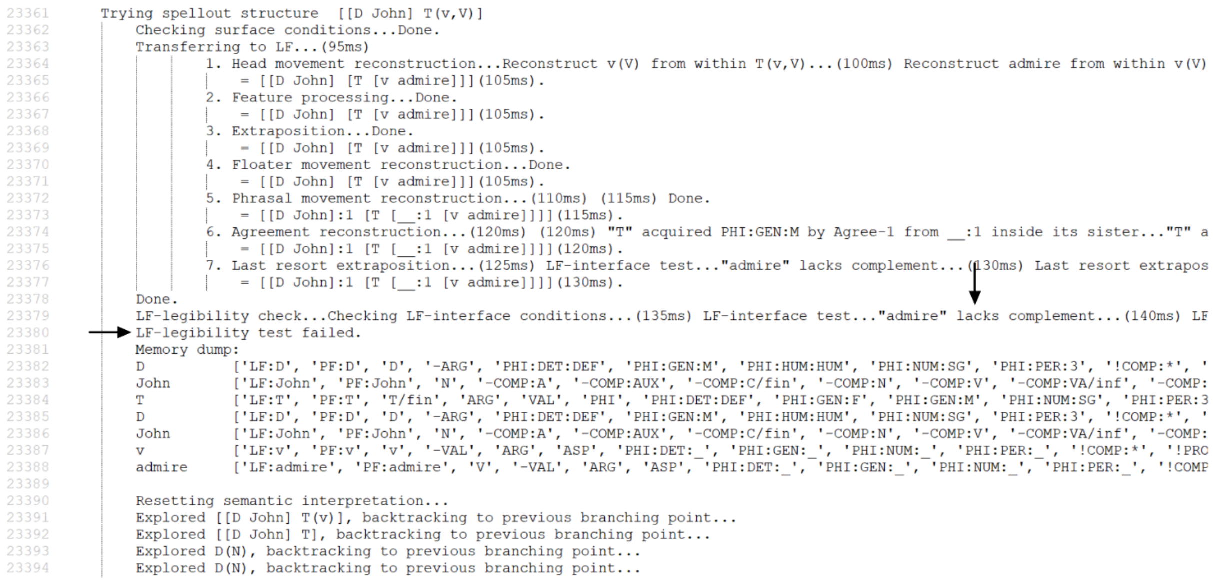


Figure 8. Output for Finnish negative clause.

[16] Notice in particular that it was not assumed that head reconstruction must be applied to the top node of the phrase structure in the current working memory. The operation is applied to α if and only if α is transferred. In the case of *John* = D(N), it is applied to D(N) which creates [$_{DP}$ D(_$_1$) N$_1$]

The algorithm generates the negative head *e-i* 'not.3sg' into the correct position on the basis of its occurrence and position in the surface string. The grammatical subject is reconstructed directly into SpecvP, while the finite verb is reconstructed into the verbal spread T-v-V. Auxiliaries, both in Finnish and English, are treated in the same way.

Ungrammatical inputs do not produce output solutions, but they leave computational steps into the derivational log file that reports the reasons for rejection. For example, sentence #88 = *John admires* = [*John* [T [v V]]] is rejected at the syntax-semantics interface because the transitive verb V lacks a mandatory DP complement (line 23379 in the log file) and no other solutions are found. See Figure 9.



```
23361       Trying spellout structure  [[D John] T(v,V)]
23362       |   Checking surface conditions...Done.
23363       |   Transferring to LF...(95ms)
23364       |   |   |   1. Head movement reconstruction...Reconstruct v(V) from within T(v,V)...(100ms) Reconstruct admire from within v(V)
23365       |   |   |   |   = [[D John] [T [v admire]]](105ms).
23366       |   |   |   2. Feature processing...Done.
23367       |   |   |   |   = [[D John] [T [v admire]]](105ms).
23368       |   |   |   3. Extraposition...Done.
23369       |   |   |   |   = [[D John] [T [v admire]]](105ms).
23370       |   |   |   4. Floater movement reconstruction...Done.
23371       |   |   |   |   = [[D John] [T [v admire]]](105ms).
23372       |   |   |   5. Phrasal movement reconstruction...(110ms) (115ms) Done.
23373       |   |   |   |   = [[D John]:1 [T [__:1 [v admire]]]](115ms).
23374       |   |   |   6. Agreement reconstruction...(120ms) (120ms) "T" acquired PHI:GEN:M by Agree-1 from __:1 inside its sister..."T" a
23375       |   |   |   |   = [[D John]:1 [T [__:1 [v admire]]]](120ms).
23376       |   |   |   7. Last resort extraposition...(125ms) LF-interface test..."admire" lacks complement...(130ms) Last resort extrapos
23377       |   |   |   |   = [[D John]:1 [T [__:1 [v admire]]]](130ms).
23378       |   Done.
23379       |   LF-legibility check...Checking LF-interface conditions...(135ms) LF-interface test..."admire" lacks complement...(140ms) LF
23380     ──▶   LF-legibility test failed.
23381       |   Memory dump:
23382       |   D          ['LF:D', 'PF:D', 'D', '-ARG', 'PHI:DET:DEF', 'PHI:GEN:M', 'PHI:HUM:HUM', 'PHI:NUM:SG', 'PHI:PER:3', '!COMP:*', '
23383       |   John       ['LF:John', 'PF:John', 'N', '-COMP:A', '-COMP:AUX', '-COMP:C/fin', '-COMP:N', '-COMP:V', '-COMP:VA/inf', '-COMP:
23384       |   T          ['LF:T', 'PF:T', 'T/fin', 'ARG', 'VAL', 'PHI', 'PHI:DET:DEF', 'PHI:GEN:F', 'PHI:GEN:M', 'PHI:NUM:SG', 'PHI:PER:3
23385       |   D          ['LF:D', 'PF:D', 'D', '-ARG', 'PHI:DET:DEF', 'PHI:GEN:M', 'PHI:HUM:HUM', 'PHI:NUM:SG', 'PHI:PER:3', '!COMP:*', '
23386       |   John       ['LF:John', 'PF:John', 'N', '-COMP:A', '-COMP:AUX', '-COMP:C/fin', '-COMP:N', '-COMP:V', '-COMP:VA/inf', '-COMP:
23387       |   v          ['LF:v', 'PF:v', 'v', '-VAL', 'ARG', 'ASP', 'PHI:DET:_', 'PHI:GEN:_', 'PHI:NUM:_', 'PHI:PER:_', '!COMP:*', '!PRO
23388       |   admire     ['LF:admire', 'PF:admire', 'V', '-VAL', 'ARG', 'ASP', 'PHI:DET:_', 'PHI:GEN:_', 'PHI:NUM:_', 'PHI:PER:_', '!COMP
23389       |
23390       |   Resetting semantic interpretation...
23391       |   Explored [[D John] T(v)], backtracking to previous branching point...
23392       |   Explored [[D John] T], backtracking to previous branching point...
23393       |   Explored D(N), backtracking to previous branching point...
23394       |   Explored D(N), backtracking to previous branching point...
```

Figure 9. A screenshot from the derivational log file containing part of the derivation for sentence **John admires*.

If we provided in the lexicon that *admires* is either an intransitive verb or ambiguous between transitive and intransitive readings, then a solution would be found.

2.3 CP cartography mechanics

In most of the data examined in this work, the predicate moves to the left peripheral CP-domain while carrying a grammatical feature (or several) expressing one of the special interpretations associated with these operations. In the example (19), this involves yes/no interrogativization.

(19)	Myy-dä-**kö**	Pekka	halus-i	_	koko	omaisuute-nsa?
	sell-A/INF-**Q**	Pekka.NOM	wanted-PAST.3SG		all	possessions-px/3sg

'Was it selling that Pekka wanted to do to all his possessions?'

We examine how the analysis calculates sentences of this type. I will focus on syntax and discuss semantics in a separate subsection 2.5. We begin by considering sentence (20), #102 in the output.

(20)	IHAILE-E	Pekka	Merja-a!
	admire-3SG.FOC	Pekka.NOM	Merja-PAR
	’Pekka DOES admire Merja!’ or		
	’Pekka ADMIRES Merja!’		

The fronted finite verb has been stressed prosodically to emphasize its contrastive focus reading (‘Pekka admires, not hates, Merja’ or ‘Pekka DOES admire Merja’). Prosodic information is transmitted to the syntactic component by prosodic features, which are (due to the lack of real prosody in the text files) attached to the input string as inflectional features. This is a simplification, of course. The calculated output for this input is shown in Figure 10.

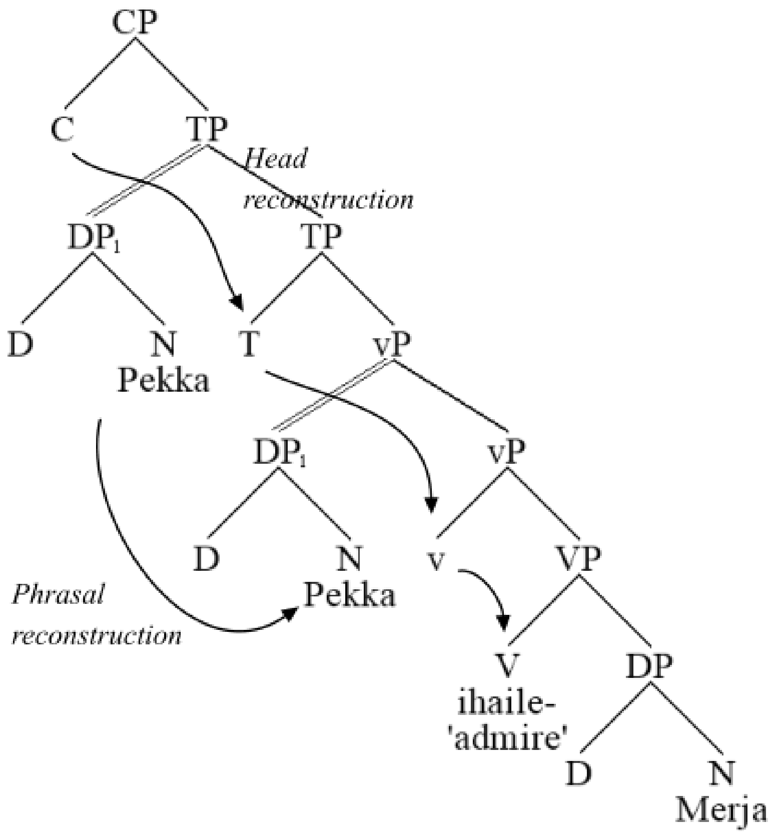


Figure 10. Calculated output for a simple T-to-C reconstruction

The head reconstruction algorithm leaves the highest head in situ and reconstructs the rest. The first step in the derivation is therefore (21)a-b.

(21) a.	Ihailee-foc	Pekka		Merja-a!
	admires-FOC	Pekka.NOM		Merja-PAR
b.	C_{foc}(__)	[Pekka	T(v, V)	Merja-a]

This will eventually result in the phrase structure shown in Figure 10, in which the clause is headed by a contrastive focus C-element. Crucially, these mechanisms correctly reject sentences in which the complex head with a C-feature occurs in a wrong position, generating the C-head into a position in which it cannot be selected (22).

(22) a.	*Pekka	ihaile-e-**ko**	Merja-a?
	Pekka	admire-3SG-Q	Merja-PAR
b.	*Pekka	Merja-a	ihaile-e-**ko**?
	Pekka	Merja-PAR	admire-3SG-Q

It would be easy to develop an algorithm that reconstructs C-elements from the wrong positions shown in (22) into the correct positions in (21). Yet, this would not match with the empirical facts. It is the lack of an operation of this type that under the present hypothesis explains why wrongly positioned words create ungrammatical outputs. Indeed, it is interesting to ask why the hearer could not in fact "repair" the strings in (22).

It is presupposed in all of the above that the prosodic focus feature [foc] is not an inflectional element but an independent morpheme C_{foc}. It can be defined in the lexicon either as a feature, head, or both (if we wanted to make it ambiguous). What remains a problem, though, is the fact that what looks to be the exact same feature can occur in two other contexts: as attached to a predicate in situ, creating an in situ prosodic focus construction (23a), and inside a pied-piping phrase that is moved to SpecCP (23b).

(23) a.	Pekka	IHAILEE	Merja-a.	
	Pekka	admires.3SG.FOC	Merja-PAR	
	'Pekka does admire Merja.'			
b.	[Merja-n	SISKOA]	Pekka	ihaile-e.
	Merja-GEN	sister.FOC	Pekka.NOM	admire-3SG
	'It is Merja's SISTER, not mother, that Pekka admires.'			

Assuming that [foc] is mapped to C would create a clause-internal CP-layer for (23a) and an extra CP-layer inside the pied-piped phrase for (b), both which would get rejected at the syntax-semantic interface. There is no empirical evidence for such layers either. In addition, positing them would create empirical and theoretical problems, as discussed in the main article. An additional complication is that due to its agglutinative nature, it is possible to attach several features to one head or phrase. They do not generate separate C-heads but are all accumulated inside the same head. Taken together these data show that these morphemes are inflectional C-features that can be collected inside some head, yet the head reconstruction derivation assumed that [foc] correspond to a head. Consider however again (23b). The surface string contains a tensed verb preceded by two arguments and no overt C-head. The model must generate a phonologically null C-head to the structure on the basis of the criterial C-feature inside the moved phrase (Brattico and Chesi 2020). A similar mechanism was created for the purposes of the present study. The mechanism operates inside the lexico-morphological component and extrapolates a phonologically null C-head to the lexical stream when needed (i.e. [foc] →C#foc). This is visible in the derivational log file, for example in line 25301, where the C/fin morpheme is inserted to the lexical stream and then appears in syntax as a separate morpheme. All C-features, if there indeed are many, can then be inserted inside this head as features, from where they enter into semantic computations.

These mechanisms were tested in connection with negation, modal auxiliary, auxiliary verbs and with finite verbs that take infinitival complements. In such cases, the fronted element (negation, auxiliary, modal) is reconstructed to its correct gap position based on selection, while the C-element remains in situ and hosts the C-features. The mechanism was also tested with noncanonical word orders (Group 2.1.7 in the test corpus), an intervening PP argument (Group 2.1.8) and with pro-drop constructions (Group 2.1.9). These data too were calculated correctly.

2.4 Long head reconstruction

The Finnish long head movement pattern, in which a nonfinite verb moves over a finite verb (1), is perhaps not the norm, although long head movement constructions have been found from various languages as discussed in the main article (Lema and Rivero 1990; Rivero 1991; Roberts 1993, 2010). In English, head movement is local. Moreover, in Finnish the nonlocal pattern is restricted to cases which involve C-features; all other instances are local. Finally, there are examples in Finnish in which also head movement to C is local. Pure auxiliaries, for example, cannot move nonlocally (24).

(24) a.	*Ollut$_1$-ko	Pekka	e-i	ole	__$_1$	tänään	koulussa?
	been-Q	Pekka.NOM	not-3SG	be		today	school

The distinction between local and nonlocal head movement seems to be associated with whether the moved element has enough lexical content to form a legitimate target for C-features. When an element such as a pure auxiliary moves, the interpretation does not target the specific lexical context of the auxiliary (if it has any) but the (truth of the) whole proposition (25).

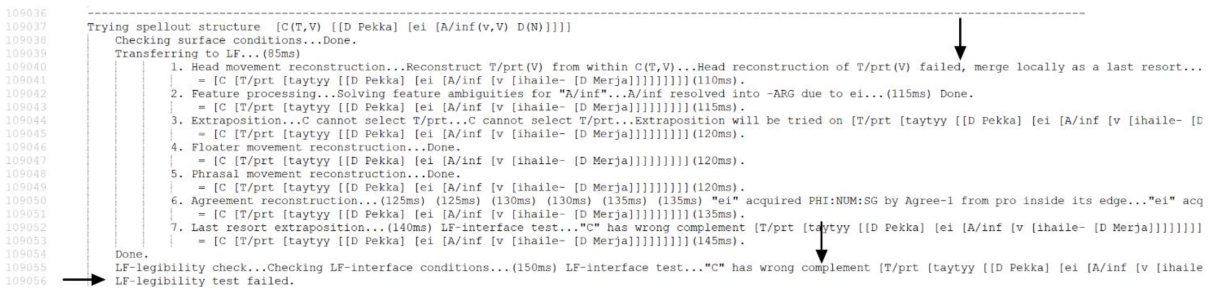
(25)	On$_1$	Pekka	__$_1$	ollut	tänään	koulussa!
	be	Pekka		been	today	in.school

‘Pekka HAS been today in school.’

What is contrasted is the truth of the whole proposition against some prior claim in which the opposite is claimed. To capture this pattern, it was assumed that head reconstruction cannot skip heads unless it is triggered by a special C*-feature, referring to the type of left peripheral features associated with both long head and phrasal movement in Finnish. Strict locality (in the sense of Travis 1984) applies to regular verbal spreads (T-v-V) as well as to auxiliary reconstruction (25).[17] This is captured at the level

[17] Since it is unclear to me at present writing what the semantic effect of the local T-to-C movement is (perhaps verum focus, at least in some cases), local T-to-C construction is derived by creating a regular finite C-head (C/fin) without special C*-features. In English, fronting an auxiliary creates an interrogative sentence, but the interrogativization targets the whole proposition (thus, what is created is

of code by using feature intervention. If a C*-feature is not present, then all functional heads cause intervention for head reconstruction, which becomes as local as possible, discharging everything into local head spreads.[18] This mechanism explains the lack of long head movement in cases that do not exhibit C*-features, both in English and Finnish. In such cases, the algorithm will attempt local head movement, but the resulting configuration is rejected at the syntax-semantics interface. This process is shown in Figure 11, which is a screenshot from a derivational log file for sentence #440 showing what happens when long head movement would be required for an element that does not have a C*-feature.



```
109036       ----------------------------------------------------------------------------------------------------------------------------
109037       Trying spellout structure  [C(T,V) [[D Pekka] [ei [A/inf(v,V) D(N)]]]]
109038           Checking surface conditions...Done.
109039           Transferring to LF...(85ms)
109040                   1. Head movement reconstruction...Reconstruct T/prt(V) from within C(T,V)...Head reconstruction of T/prt(V) failed, merge locally as a last resort...
109041                      = [C [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]]](110ms).
109042                   2. Feature processing...Solving feature ambiguities for "A/inf"...A/inf resolved into -ARG due to ei...(115ms) Done.
109043                      = [C [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]]](115ms).
109044                   3. Extraposition...C cannot select T/prt...C cannot select T/prt...Extraposition will be tried on [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D
109045                      = [C [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]]](120ms).
109046                   4. Floater movement reconstruction...Done.
109047                      = [C [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]]](120ms).
109048                   5. Phrasal movement reconstruction...Done.
109049                      = [C [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]]](120ms).
109050                   6. Agreement reconstruction...(125ms) (125ms) (130ms) (130ms) (135ms) (135ms) "ei" acquired PHI:NUM:SG by Agree-1 from pro inside its edge..."ei" acq
109051                      = [C [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]]](135ms).
109052                   7. Last resort extraposition...(140ms) LF-interface test..."C" has wrong complement [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]
109053                      = [C [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile- [D Merja]]]]]]]]](145ms).
109054           Done.
109055           LF-legibility check...Checking LF-interface conditions...(150ms) LF-interface test..."C" has wrong complement [T/prt [taytyy [[D Pekka] [ei [A/inf [v [ihaile
109056           LF-legibility test failed.
```

Figure 11. Failed head reconstruction. The gap is nonlocal, but head reconstruction cannot reach it (line 109030); hence the last resort option is selected. This configuration fails at the syntax-semantic interface (line 109056)

Provided that the reconstruction is triggered by a C*-feature in the complex head, long head movement becomes available. The head reconstruction algorithm travels downwards on the projectional spine of the phrase structure, following labelling and selection and searching for the first suitable position for the reconstructed object. The process is called *minimal search*, suggesting a similarity to the minimal search system proposed in Chomsky 2008 but interpreted as a top-down reconstruction. Minimal search is defined in the phrase_structure.py module, function minimal_search(). The essence of the search algorithm is (26).[19] Suppose α is targeted for search, then

(26) *Minimal search*

if α is primitive, terminate search; otherwise search X in $\{_{XP}$ X Y$\}$, but if X is primitive, search Y.

finite C plus some type of interrogative feature). Some languages seem to exhibit a purely formal root level fronting. This requires a separate study that includes such languages in the dataset.

[18] A functional head is defined by feature [!COMP:*] = 'must have some complement' which in this case functions as the intervening feature. The feature intervention approach was adopted in order to leave room for long head movement constructions that do not follow the A-bar profile exhibited by Finnish. Nevertheless, the approach, although based on a fairly popular approach going back to Rizzi 1990, still feels ad hoc when examined at the level of actual code.

[19] The intervention condition is currently provided in the head reconstruction algorithm, but a deeper explanation would include it into minimal search itself. It would then generalize to all reconstruction operations that use this algorithm.

The algorithm dodges left phrases (specifiers, subjects) and right adjuncts and enters complements when X is primitive. This captures CED effects in the dataset. All reconstruction uses minimal search. It is interesting to observe in this connection that both left branches and right adjuncts are transferred independently to the semantic component as phases, hence this could ultimately provide a deeper explanation for (26).[20] However, no such unification was pursued in this study. Figure 12 shows the calculated results for a long-distance head movement sentence (27).

(27)	IHAILLA$_1$	Pekka	sanoo	että	Pekka	haluaa	__$_1$	Merja-a. (#448)
	admire.A/INF.FOC	Pekka	says	that	Pekka	wants		Merja-PAR

[20] We could perhaps reduce the definition into ‘search what is currently in the active working memory’, assuming that left branches and right adjuncts have been transferred independently (as they are).

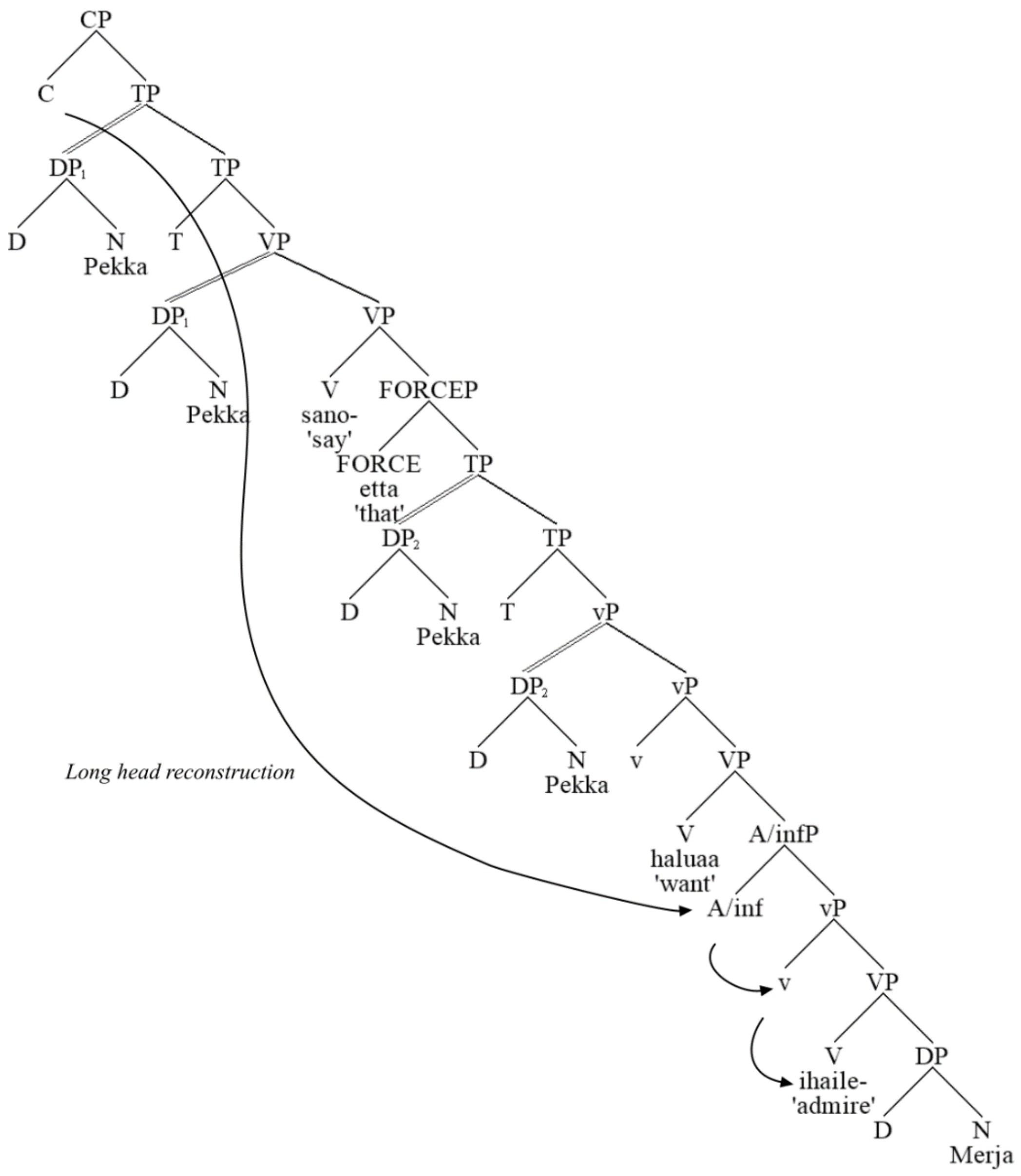


Figure 12. Output of the model for sentence (27)

The model reconstructs A/inf(V) from within the fronted head, performs (in principle unlimited) search and finds the position indicated by the arrow, where the A-infinitival can be selected. All long head movement constructions are derived in the same way.

In the main article some considerations were presented suggesting that the reconstruction dependencies are those of the A-bar system. The C*-features associated with the fronted head are also involved in Finnish phrasal A-bar movement.[21] As explained in the next section, both cases are handled by the same system. This idea was to my knowledge first explored by Roberts (1993, 2010), where it was proposed that the A-bar system can target also (verbal) heads, not only phrases (see also

[21] With few exceptions: imperatives are only possible in connection with verbal head movement, whereas wh-features cannot be attached to verbal heads.

Matushansky 2006). Suppose we consider regular phrasal A-bar movement that involves pied-piping, such as (28).

(28)	[Peka-n-ko	auto]	hajosi?
	Pekka-GEN-Q	car	broke

’Was it Pekka’s (and not Merja’s) car that broken down?’

Once the pied-piped phrase is reconstructed, the syntax-semantics interface sees an operator-scope dependency presented in (29).

(29)	C(ko). . .	T [[Peka-n-ko auto] broke]			
	C(Q). . .	T	[[Pekka-GEN-Q	car	broke

There is a dependency between C(Q) and a word *Pekka*. Head movement constructions exhibit the same dependency between C(Q) and some verbal head such as the A-infinitival. Therefore, if the mechanism can target proper names or others heads such as demonstratives, why not a verbal head? The only difference between these two cases appears to be that the former involves pied-piping as an *additional mechanism*. This led Roberts to the insight that the problem is to explain, not why there is head movement, but why phrasal movement triggers additional pied-piping. Although many problems remain, this to me seems to be the correct or at the very least useful way of looking at these data.

2.5 Semantics and the operator-variable system

Before addressing the semantic interpretation of the head movement constructions specifically, I will provide a brief description of the overall general semantic architecture assumed at the present writing to make navigating the source code easier.

When a phrase structure object is transferred to the semantic component, it first arrives at the syntax-semantic interface (LF interface) that mediates most of the communication between syntax and semantics.[22] Syntax-semantics interface objects are phrase structure objects arriving from transfer. Most of the phrase structures illustrated in this document and in the main article are syntax-semantic interface objects in this sense. All intermediate objects are available in the derivational logs, however. Finally, the syntax-semantics objects are not transformed or manipulated in any way; they are only interpreted.

Information at the syntax-semantic interface is interpreted by a larger system called *narrow semantics* (module narrow_semantics.py) which translates information arriving from the syntactic pathway into a form understood by the conceptual-intentional systems (global cognition). Narrow semantics consists of several subsystems reacting to features in the input. These extensions are shown in Figure 11. OV represents the subsystem responding to C*-features, discussed below in more detail.

[22] There is an additional pragmatic interface that reacts to noncanonical word orders, but this interface is not relevant to the issue at hand and does not affect syntactic processing in any way.

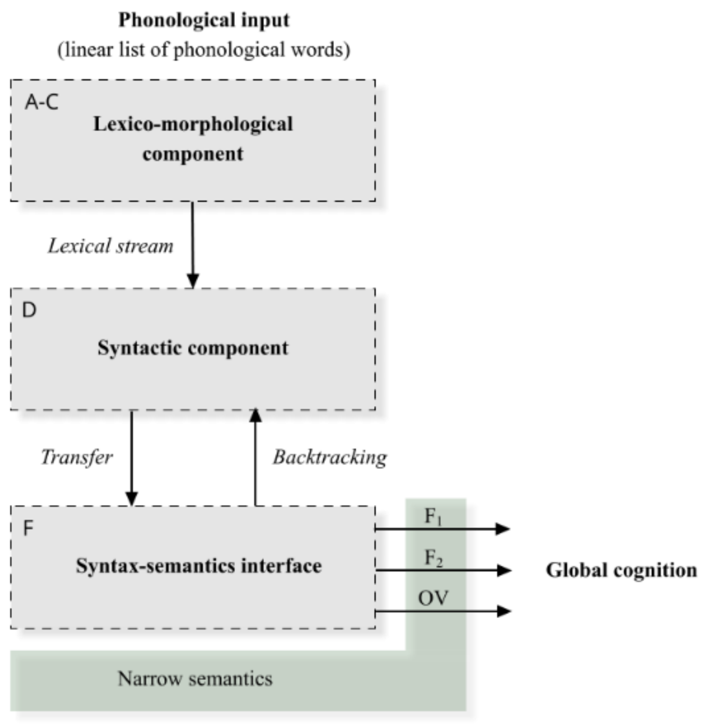


Figure 13. Internal structure of the semantic component.

The various subsystems existing inside narrow semantics (e.g., F_1, F_2, OV) respond to particular feature types. One of these systems, called the operator-variable module, responds to C*-features.[23] It handles both phrasal operator constructions and head operator constructions. An operator is interpreted by linking it with a propositional scope. Propositional scope is determined in one of two ways. If the input string contained an overt scope marker, then the scope will be marked by the finite head that contains the same operator feature, copied from the overt scope marker during transfer/reconstruction, as explained above. If the scope is not marked overtly, then it is assumed to be ambiguous and any c-commanding finite head can be selected. Thus, in the case of a regular interrogative (30a), the reconstructed operator *who* is linked with the propositional scope as indicated by the original surface position of the same operator.

(30) a. Who + does + John + admire? (Input)

b. [who$_1$ [C [John [T [admire who$_1$]]]]] (Syntax-semantics interface object)

FIN OP:WH

OP:WH

(Scope)

[23] At the level of code C*-features are denoted by [OP:F], with F replacing the feature (e.g., WH, Q) and OP diverging the processing to the operator-variable module.

This is interpreted so that the scope of the question is the material covered by the scope dependency, shown by the line in (30). The operator feature that appears at C is copied during transfer, as explained in Section 2.3. If the operator were inside an embedded clause and the scope in the main clause, a wide scope reading results (31). The scope dependency covers more material that it does in (30).

(31) Who$_1$ did John claim that Mary admires __$_1$?

The same system interprets Finnish long head movement constructions, in agreement with Roberts' idea that long head movement and long phrasal movement are based on the same basic system. Thus, (32) is interpreted so that the fronted A-infinitival constitutes a yes/no operator associated with a predicate with a scope spanning the whole clause. The long head movement tells the algorithm what the scope is, as the C-head remains in situ while the rest reconstructs. This matches with the semantic interpretation elicited from native speakers. The speaker is asking if it's true that Pekka wanted to sell (and not, say, borrow) his possessions.

(32)	Myy-dä-kö	Pekka	halus-i	_		omaisuute-nsa?
	sell-A/INF-Q	Pekka.NOM	wanted-PAST.3SG			possessions-px/3sg
	C			A/inf	V	
	[OP:Q][FIN]			[OP:Q]		

Figure 14 shows the outcome for a structurally similar sentence (33) that can be found from the test materials (sentence number #266).

(33)	Ihailla-ko	Pekka	haluaa	_	Merja-a?
	admire.A/INF-Q	Pekka	wants		Merja-PAR

'Does Pekka want to admire (and not hate) Merja?'

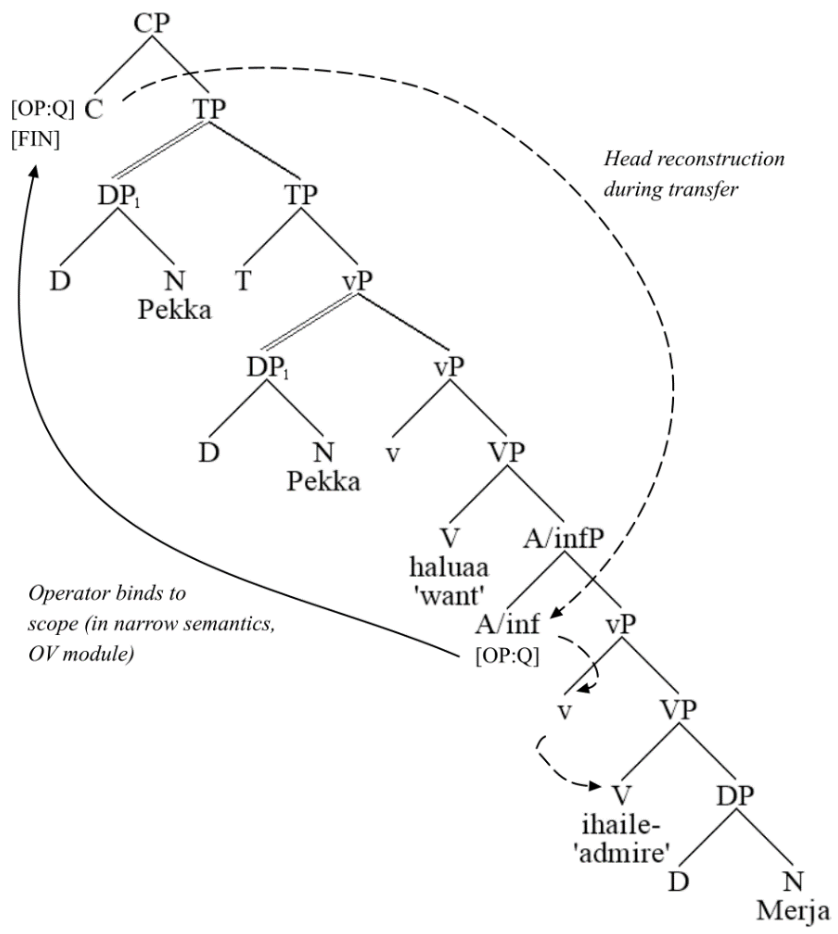
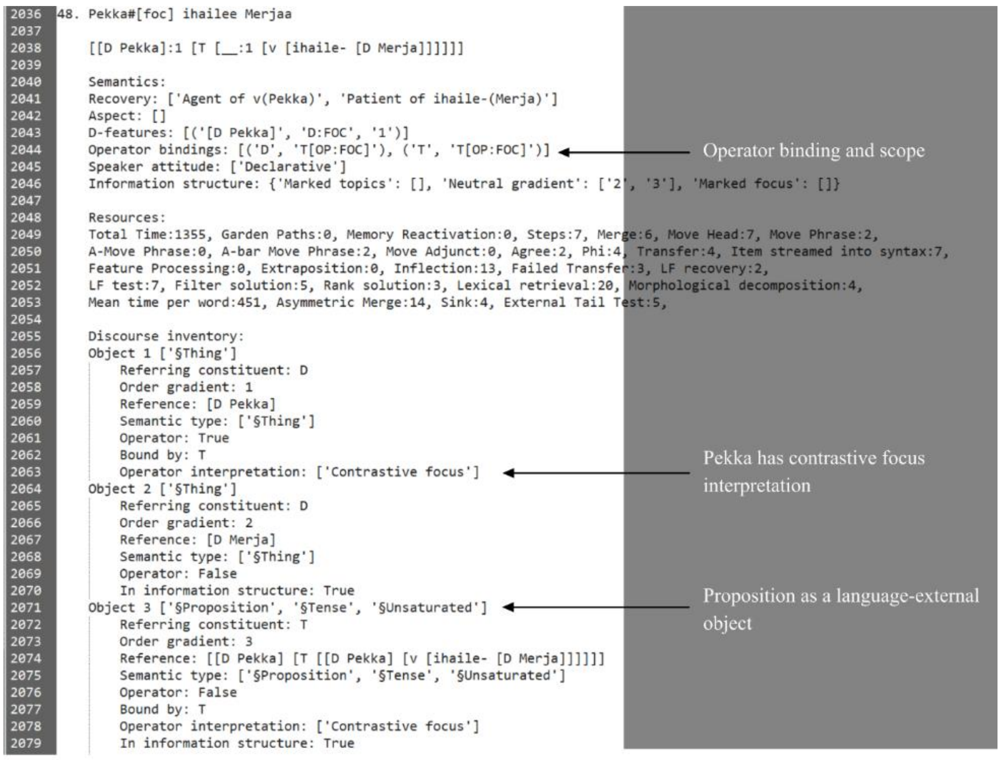


Figure 14. Head reconstruction and operator binding for a simple long head movement construction in Finnish.

This information is present in all output from the algorithm, and for all grammatical input sentences. Figure 15 shows the same information in a text format for sentence *Pekka#foc ihailee Merjaa* ‘PEKKA admires Merja’, where the grammatical subject has been focussed contrastively.



```
2036 48. Pekka#[foc] ihailee Merjaa
2037
2038     [[D Pekka]:1 [T [__:1 [v [ihaile- [D Merja]]]]]]
2039
2040     Semantics:
2041     Recovery: ['Agent of v(Pekka)', 'Patient of ihaile-(Merja)']
2042     Aspect: []
2043     D-features: [('[D Pekka]', 'D:FOC', '1')]
2044     Operator bindings: [('D', 'T[OP:FOC]'), ('T', 'T[OP:FOC]')]
2045     Speaker attitude: ['Declarative']
2046     Information structure: {'Marked topics': [], 'Neutral gradient': ['2', '3'], 'Marked focus': []}
2047
2048     Resources:
2049     Total Time:1355, Garden Paths:0, Memory Reactivation:0, Steps:7, Merge:6, Move Head:7, Move Phrase:2,
2050     A-Move Phrase:0, A-bar Move Phrase:2, Move Adjunct:0, Agree:2, Phi:4, Transfer:4, Item streamed into syntax:7,
2051     Feature Processing:0, Extraposition:0, Inflection:13, Failed Transfer:3, LF recovery:2,
2052     LF test:7, Filter solution:5, Rank solution:3, Lexical retrieval:20, Morphological decomposition:4,
2053     Mean time per word:451, Asymmetric Merge:14, Sink:4, External Tail Test:5,
2054
2055     Discourse inventory:
2056     Object 1 ['§Thing']
2057         Referring constituent: D
2058         Order gradient: 1
2059         Reference: [D Pekka]
2060         Semantic type: ['§Thing']
2061         Operator: True
2062         Bound by: T
2063         Operator interpretation: ['Contrastive focus']
2064     Object 2 ['§Thing']
2065         Referring constituent: D
2066         Order gradient: 2
2067         Reference: [D Merja]
2068         Semantic type: ['§Thing']
2069         Operator: False
2070         In information structure: True
2071     Object 3 ['§Proposition', '§Tense', '§Unsaturated']
2072         Referring constituent: T
2073         Order gradient: 3
2074         Reference: [[D Pekka] [T [[D Pekka] [v [ihaile- [D Merja]]]]]]
2075         Semantic type: ['§Proposition', '§Tense', '§Unsaturated']
2076         Operator: False
2077         Bound by: T
2078         Operator interpretation: ['Contrastive focus']
2079         In information structure: True
```

Operator binding and scope
Pekka has contrastive focus interpretation
Proposition as a language-external object

Figure 15. Result summary for sentence *Pekka#foc ihailee Merjaa* ‘PEKKA admires Merja’.

The field “semantics” in Figure 15 provides a summary of the semantic interpretation, while the field “operator bindings” shows that the D-operator (head of DP = *Pekka*) was bound by finite T with feature [OP:FOC].[24] The lower part of the results summary contains a list of language-external objects that were created during processing. The proposition is projected into the global discourse directory when it is identified by the operator it contains.

This approach to semantic interpretation was inspired by Chomsky 2008, where the author proposes a “duality of interpretation” approach in which two distinct semantic systems interpret different aspects of the incoming syntactic structure. The idea was implemented by assuming that the second system responsible for operators and their propositional scopes reacts to specific feature types in the input. We can think of the linguistic structure as grammaticalizing conceptual realms that are part of the extralinguistic global cognition. I do not restrict the number of semantic processing pathways into two, however. For example, noncanonical word orders are interpreted by a separate pragmatic pathway to represent discourse-configurational topic-focus structures (see line 2406 in Figure 15).

[24] Finite T will also “bind itself,” as a consequence of the assumptions just made. This could be ruled out by blocking reflexive binding, yet the matter is more complex because T can function as an operator (T-to-C movement with an operator feature at T). Another reason why this mechanism was left into the model was because it could be useful in the explanation of the genesis of verum focus interpretation that targets the whole proposition. Perhaps in such cases the operator and the scope marker are the same element?

Notice that the algorithm that calculates any operator's scope relies crucially on finiteness. The scope is determined by finiteness + a copy of the operator feature itself. This links grammatical finiteness with the semantic notion of proposition.

3 Additional topics

3.1 Performance

We can collect detailed performance metrics of the operation of the algorithm. This makes it possible to compare computational costs and the psycholinguistic plausibility of alternative hypotheses and proposed computational mechanisms. We could also compare different solutions in terms of their economy properties if such were considered relevant.[25] Performance data concerning the simulations run in the present study are summarized in Table 1 and explained further below.

Table 1.
Performance metrics (mean processing times, number of operations) for the present dataset.

	Group						
	1	2	3	4	5	6	All
Means per relevant unit							
Total PPT/sentence (ms)	1878.9	1791.6	2048.2	2163.8	3511.0	4860.0	2042.8
PPT/word (ms)	447.0	493.6	497.6	432.8	505.1	1185.0	495.1
N of garden paths/sentence	0.2	0.0	0.1	0.0	0.0	2.4	0.1
N of Merge-1/sentence	8.4	8.4	9.6	10.6	14.7	18.0	9.4
N of Move head/sentence	18.2	15.9	17.2	21.9	49.3	35.4	19.1
N of Move phrase/sentence	3.0	3.1	3.1	2.2	15.1	9.4	3.5
N of Agree/sentence	6.6	7.3	8.6	14.3	24.5	4.4	9.0
N of Asymmetric Merge/sentence	55.8	66.0	80.7	74.4	228.7	82.4	75.1

Groups: 1 = baseline tests, 2 = Local T-to-C movement, 3 = LHM, 4 = LHM over two V-elements, 5 = Super (long distance) LHM, 6 = VP fronting.

The columns in Table 1 represent the experimental groups of the test corpus (1 = baseline tests, 2 = Local T-to-C movement, 3 = LHM, 4 = LHM over two V-elements, 5 = Super (long distance), LHM, 6 = VP fronting; 7-10 contained ungrammatical sentences and are ignored). The user can use whatever groups and grouping relevant for the research agenda or hypothesis, here I use the groups that were readily available in the test corpus. The rows show summaries of the performance metric calculated over all sentences in each group. The first row represents total PPT per sentence, the second the mean PPT per word. PPT refers to *predicted processing time*. Predicted processing times are calculated in the following way. Each independent syntactic operation projected by the model (e.g., Merge, Move) is associated with a predicted cognitive cost (in milliseconds), corresponding to an estimated neuronal

[25] Complexity metrics have played a theoretical and/or empirical role in the minimalist program (Chomsky 2000:104-105, 110-111), where the putative empirical evidence supporting the use of such notions in linguistic theorizing has resulted in them being used also as theoretical constraints or goals.

processing cost in the human brain of that operation, which are summed up during real-time processing to yield total predicted processing times for each input sentence. The PPTs shown in Table 2 are computed in this way. For example, the current model predicts that the average comprehension speed for sentences in groups 1-5 is approximately 500ms per word, whereas in group 6 it is much longer. These results are obtained when most elementary operations projected by the model are associated with the PPT of 5ms. They should ultimately reflect real physiological properties of the human brain, although currently they are only used in a relative sense to compare the computational costs of various algorithmic language comprehension models.[26] Notice that these numbers include not only operations that were done overtly and recorded into the log files, but also all covert operations that were used when for example evaluating candidate solutions.

Below the two PPT estimates is a row recording the mean number of garden paths (row "N of garden paths/sentence" in Table 1). The algorithm generates garden paths in connection with 26 test sentences in this test corpus, almost all which occur in connection with VP-fronting (Group 6 in the test corpus, #479-485) that are (rightly or wrongly) hard for the model. In all other cases the first pass parse corresponds to an acceptable (and correct) interpretation. Thus, there is very little garden pathing. The model uses several cognitive heuristic parsing principles to achieve this efficiency.[27] The researcher can activate parsing heuristics freely when configuring the simulation. In the main article it was noted that the grammaticality and semanticality judgments that involved VP-fronted were unclear and difficult to do. This could be because they are associated with an extra processing cost, influencing acceptability.

Raw performance metrics are provided in the output file LHM_corpus_resources_FINAL.txt, which contains a comma-delimited text file listing each input sentence followed by all performance metrics associated with the processing of that sentence, as recorded during processing. This file can be loaded into an external program such as Excel or SPSS for analysis.

3.2 Transfer and transformations

Transfer is executed when a candidate solution, either a whole sentence or some part of it, is submitted to the syntax-semantic interface for evaluation. Head reconstruction is part of transfer. Here I will describe some of the more general features of this operation.

[26] Merge^{-1} ("inverse Merge") in Table 1 refers to an operation in which a primitive or phrasal constituent is attached to the right edge of an existing partial phrase structure during parsing, including when garden pathing (backtracking). It does not include more primitive Merge operations that are executed during reconstruction, for example. Asymmetric Merge includes all elementary bottom-up Merge operations in which two constituents are combined together anywhere in the system. This operation occurs in connection with Merge-1 as well as with any phrasal or head reconstruction, thus the number is higher. Asymmetric Merge and the number of garden paths provides the best overall estimation of the computational cost of any proposed solution.

[27] If these principles are knocked out, efficiency drops dramatically. In one simulation the processing of one sentence with one embedded that-clause (e.g., *John said that Mary admires him*) consumed one hour from a standard desktop computer and generated 134795 garden paths; in the present study the number of garden paths for sentences like this was zero.

One way to see why a mechanism of this type must exist is to consider the Finnish free word order profile. In a basic ditransitive clause virtually all logically possible word order permutations are also grammatical, yet the attested word forms themselves are under strict grammatical control.[28] The A-infinitival, for example, can be selected only by certain kinds of words, independent of where they happen to be in the input that can sometimes resemble 'word salad'. Because word order is relatively free, these constrains must operate independent of the surface positions of any of these elements. Furthermore, the restrictions are closely matched or identical with conditions on semantic interpretation. Thus, the intuitive reason why the Finnish A-infinitival can occur only in specific grammatical environments has to do with the way it is interpreted as denoting an 'object event' of some other predicate ('wants + to sell' but not 'to see wants'). If we assume that the conceptual system is universal, then there has to be a mechanism that neutralizes the attested surface variations and produces of canonical structures that the universal semantic system can interpret. Head reconstruction follows the same logic: it analyses contents of complex phonological words so that the selectional restrictions between its internal constituent lexical items can be evaluated and interpreted by principles that do not presuppose that the language is agglutinative.

I have found that the best match between empirical data and the model can be attained if we adopt a broadly generative framework and assume that what have traditionally been described as transformations in that theory are interpreted as a mirror image of transfer. This explains why the model produces what look like inverted transformational chains. This assumption is empirical. I do not know anything else in the current cognitive science literature that comes even close to capturing the complex properties of Finnish long head reconstruction. It is, however, in principle easy to imagine and develop more elegant alternatives and then demonstrate that they are superior by embedding them into some algorithm and calculating the correct empirical properties.

3.3 A note on phonological content at C and the V2 property

The test corpus includes examples in which C*-features are attached to a wrong head or the host head is not in a fronted position. An earlier model overgeneralized examples such as (34), judging them grammatical.

(34)	*Pekka	ihaile-e-ko	Merja-a? (Wrongly judged grammatical)
	Pekka.NOM	admire-T/FIN.3SG-Q	Merja-PAR

These sentences are ungrammatical because it is not possible to fill in SpecCP and C at the same time. Capturing this restriction turned out to be nontrivial, because we must relate the availability of the SpecCP position in some way to the phonological content at C, yet phonological content was up to this point unavailable inside narrow syntax. It has never been needed for the derivation of any other data.

[28] Case suffixes in the case of arguments, specific verb forms in the case of predicates.

The problem is possibly more general, however, in that the model was also accepting (35) where the subject appears above the high complementizer *että* ‘that’ that no phrase can fill under any circumstances in Finnish.

(35)	*Jari	sanoi	[$_{CP}$Pekka$_1$	että	__$_1$	[ihaile-e	Merja-a.]]]
	Jari	said	Pekka.NOM	that		admire-T/FIN.3SG	Merja-PAR

In order to connect the availability of SpecCP to the phonological content at C, a mechanism was created which embeds a negative specifier selection feature to the C-head when it is created from the phonological string. The operation is performed inside the presyntactic lexico-morphological component that has access to phonological material. The same feature is part of the lexical content of the high complementizer *että* ‘that’. The solution is not entirely satisfactory, but perhaps not completely implausible either since this phenomenon is in some ways reminiscent of the V2 profile, thus possibly not an isolated anomaly.

3.4 Standard cyclic theory and the analysis: some comparisons

Before trying to compare the present analysis with standard cyclic theories, it is important to point out once more that most or all standard theories, whether they have been developed within the generative perspective or in some other formalism, are typically interpreted as analyses of grammatical competence, thus they are essentially descriptions of what is possible and impossible in a human language. They therefore take a neutral or agnostic stance on how the described limitations are brought into the actual generation or recognition of linguistic forms by native speakers when they use language for communication or for some other purpose. A cyclic, bottom-up derivation such as that proposed in Roberts (2010), for example, does not describe the actual causal sequences that occur when speakers generate utterances. Thus, Chomsky points out that

> it is perhaps worth while to reiterate that a generative grammar is not a model for a speaker or a hearer. It attempts to characterize in the most neutral possible terms the knowledge of the language that provides the basis for actual use of language by a speaker hearer. When we speak of a grammar as generating a sentence with a certain structural description, we mean simply that the grammar assigns this structural description to the sentence. When we say that a sentence has a certain derivation with respect to a particular generative grammar, we say nothing about how the speaker or hearer might proceed, in some practical or efficient way, to construct such a derivation. These questions belong to the theory of language use - the theory of performance. (Chomsky 1965: 9).

A pure competence theory provides the “boundary conditions” that a more detailed information processing theory must obey. In the same way, although I interpret the present analysis in terms of the two levels in Marr’s system, we could retreat from this position and claim that the description specifies only a characteristic function for the set of grammatical expressions. Once we do this, it is possible to compare the two approaches.

For example, what in the cyclic bottom-up theories are described as case assignment must be described as checking in the inverse model, since all agreement and case features are already present in the input. What we must do instead is to "verify them for correctness." Thus, valuation and assignment translate into checking. The two approaches are not notational variants, however. Case suffixes, for example, reflect the grammatical contexts of the canonical positions of nominal arguments and can be used for reconstruction purposes. In general, morphological word forms will likely play a major role in the computations, as hierarchical structure is absent from the sensory input and must be inferred from something, most likely at least in part from the overt morphological forms. Thus, what in the cyclic bottom-up theories are sometimes described as idle morphosyntactic by-products of the generation – "virus" features that need elimination[29] – become essential ingredients from which the recognition algorithm can infer the hierarchical structure and other properties. The same reasoning applies to linear order. Whereas in the cyclic generative approaches linear order can be considered a relatively unimportant ancillary phenomenon, a recognition model cannot afford to ignore it.

The same reasoning can perhaps be applied to the question of what might ultimately motivate displacement itself, a problem that has been animating research for six decades or more. In the present dataset the dislocated predicate signals the *scope* of the operator it carries. The scope, as elicited from native speakers, always corresponds to the material that falls under the dependency (36).

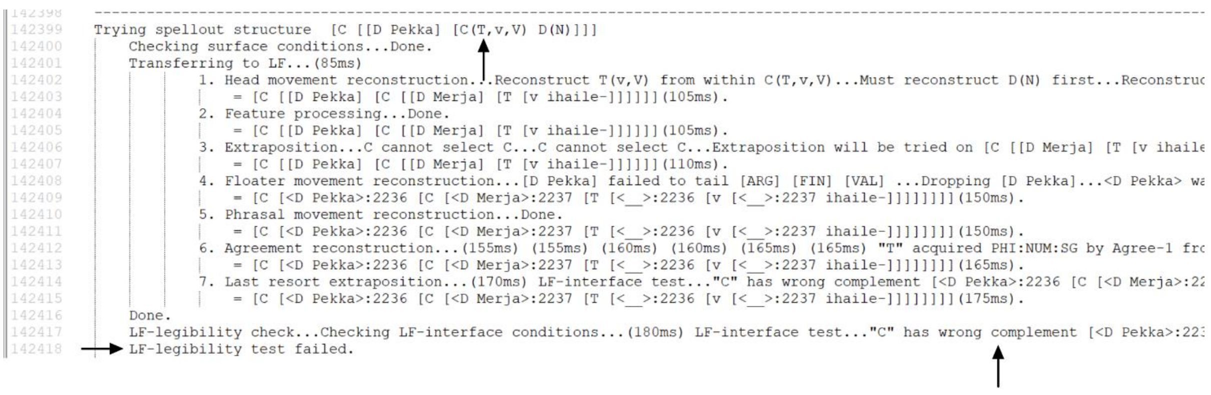
(36)	Myy-dä-kö	Pekka	aikoi	$\_\__1$	kaiken	omaisuutensa?
	sell-A/INF-Q	Pekka	planned		all	possessions

In the cyclic generative model, on the other hand, scope is part of the intended interpretation and comes essentially for free. It is where the speaker intends it. Thus, the operation copying criterial features to a local head functions as the scope marker for the reconstruction operation and therefore communicates the propositional context of the operator through the sensorimotoric interface.[30] The same reasoning can perhaps be applied to other forms of displacement. Thus, we assume that they trigger semantically interpretable reactions by transfer. Of course, these speculations would be useless unless it can be shown that they provide correct predictions also in the case of ungrammatical input sentences. In some range

[29] Thus, the actual overt morphosyntactic features can be viewed as "redundant" in having "no semantic interpretation," thus "they must be deleted before they reach the semantic interface for the derivation to converge" (Chomsky 2008: 154).

[30] It is presupposed here that (i) the properties of any head H can be reconstructed from its own overt content and from the material in its specifier(s) and that (ii) noncanonical phrases are reconstructed to their thematic canonical positions by transfer. Condition (i) copies criterial operator features to the finite C-head, while condition (ii) repairs noncanonical orders. As a consequence, the input to the semantic module interpreting operator-variable constructions generates the correct entities and dependencies to the global discourse inventory.

of core cases they do. For example, if the complex word containing the operator feature occurs in situ, an ill-formed phrase structure object that hosts a C-element in the middle of the clause would arrive to the syntax-semantic interface in such position, leading into a crash. An example is provided by sentence #558 in the test corpus, and a screenshot of the relevant derivational log file is below[31]:



```
142398  ------------------------------------------------------------------------------------------------------------------
142399  Trying spellout structure  [C [[D Pekka] [C(T,v,V) D(N)]]]
142400      Checking surface conditions...Done.
142401      Transferring to LF...(85ms)
142402              1. Head movement reconstruction...Reconstruct T(v,V) from within C(T,v,V)...Must reconstruct D(N) first...Reconstruc
142403                 = [C [[D Pekka] [C [[D Merja] [T [v ihaile-]]]]]](105ms).
142404              2. Feature processing...Done.
142405                 = [C [[D Pekka] [C [[D Merja] [T [v ihaile-]]]]]](105ms).
142406              3. Extraposition...C cannot select C...C cannot select C...Extraposition will be tried on [C [[D Merja] [T [v ihaile
142407                 = [C [[D Pekka] [C [[D Merja] [T [v ihaile-]]]]]](110ms).
142408              4. Floater movement reconstruction...[D Pekka] failed to tail [ARG] [FIN] [VAL] ...Dropping [D Pekka]...<D Pekka> wa
142409                 = [C [<D Pekka>:2236 [C [<D Merja>:2237 [T [<__>:2236 [v [<__>:2237 ihaile-]]]]]]]](150ms).
142410              5. Phrasal movement reconstruction...Done.
142411                 = [C [<D Pekka>:2236 [C [<D Merja>:2237 [T [<__>:2236 [v [<__>:2237 ihaile-]]]]]]]](150ms).
142412              6. Agreement reconstruction...(155ms) (155ms) (160ms) (160ms) (165ms) (165ms) "T" acquired PHI:NUM:SG by Agree-1 fro
142413                 = [C [<D Pekka>:2236 [C [<D Merja>:2237 [T [<__>:2236 [v [<__>:2237 ihaile-]]]]]]]](165ms).
142414              7. Last resort extraposition...(170ms) LF-interface test..."C" has wrong complement [<D Pekka>:2236 [C [<D Merja>:22
142415                 = [C [<D Pekka>:2236 [C [<D Merja>:2237 [T [<__>:2236 [v [<__>:2237 ihaile-]]]]]]]](175ms).
142416      Done.
142417      LF-legibility check...Checking LF-interface conditions...(180ms) LF-interface test..."C" has wrong complement [<D Pekka>:223
142418      LF-legibility test failed.
```

Figure 21. A derivation that crashes due to a sentence-internal CP-element. Notice that extra CP-layer on line 142399.

Linear order plays a key role in any explanation of this type; whether simple and elegant large-scale model can be constructed on these grounds remains of course to be seen.

The standard cyclic theory derives expressions by applying an iterative operation Merge, which takes two elements α, β as input and yields a new element {α, β} as output (Chomsky 2008: 138). The assumption that Merge creates sets requires an extra axiom which restricts the size to two. Implementing this requirement in Python looked suspicious to me. It implies that sets have to be counted every time merge is attempted. The counting operation looks completely ad hoc: a set is by its nature an entity that should not be limited in that way, which raises the question if merge really generates sets. Second, and more importantly, sets as such do not have left-right order that is essential when working with the sensory input. We cannot afford to lose linear order when interpolating syntactic structures from the sensory input. These two problems were solved in this study by assuming that Merge creates asymmetric binary-branching structures [α, β] with the left and right constituents being defined axiomatically. This assumption simplifies also labelling and derives automatically the limitation that linguistic constituents must have exactly two daughters. It provides axiomatic and simple definitions for the notions of left and right edge. An additional difference between the cyclic theory and the present

[31] I found one edge case in which a sentence of this type (e.g., #579) crashed because the algorithm was unable to discharge all internal components of the wrongly positioned complex predicate. This happens because the model reconstructs a complex participle verb into a left branch [[T(v, V) D] N] out of a wrongly positioned C and thus never reaches the complex head again (recall that head reconstruction uses minimal search). It is hard to say if this is correct or incorrect since the original input sentence is ungrammatical. It requires further scrutiny. This situation occurs when there is a wrongly positioned complex predicate in the middle of the sentence, T is not finite, the verb is transitive, and the direct object follows the complex predicate in the input.

model is that a recognition model creates phrase structures from the sensory input, hence we require a more complex operation (at least when working with syntactic objects that reside at the syntax-lexicon interface) which targets nodes α in the existing phrase structure and substitutes them with $[\alpha, \beta]$, β being the incoming lexical item. The target + Merge + substitute was called Merge^{-1}. Instead of expanding the structure at the top, it is expanded at the right edge. Yet, right edge expansion presupposes cyclic Merge (called *asymmetric Merge* in the implementation), which is therefore part of the present model as well.

It should be noted in this connection that the right edge expansion does not force one to adopt the comprehension perspective. It is possible that language is produced or at the very least can be produced by using the same mechanism, thus by creating sentences in the same order as they are uttered. It is also possible, at least in theory, that the motoric planning stage is executed by creating spellout structures and not syntactic-semantics interface objects. While speculation, this point is relevant in the sense that we do not need to associate any generative model with any concrete causal information processing model. We can consider the present model, as well as any cyclic system, as a way of specifying only the logic of recursive composition without taking a stance of what it means from the point of view of comprehension or production.

Ultimately the matter comes to the question of what kind of phenomenon language is. Chomsky argues that "the primary contribution to the structure of [faculty of language] may be optimization of mapping to the C-I interface. . . while the mapping to the [sensorimotoric] interface is an ancillary process" that does not enter in the any "principled explanation" (Chomsky 2008: 136). This is almost the exact opposite of what I have claimed here. Transfer exists to process sensory objects (sensorimotoric interface objects) into a format that can be understood by the syntax-semantics interface, and therefore we would conclude that they are both equally important components in any "principled explanation." It is clear that much of the universal limitations of language are derived in the present model from the concrete sensorimotoric properties (linear order, overt morphology, displacement). For example, the fact that a wrongly positioned complex predicate with a C*-feature leads into ungrammaticality is explained as a consequence of the very narrowly construed and thus limited operation that generates syntactic structures and semantic interpretation from the linear order itself. The mechanism is so unintelligent that it insists on positioning the element into the middle of the clause as directly cued by its position in the surface string. In the same way, the system refuses to consider hypotheses in which the highest morpheme of a complex word would not belong to the exact position in which it appears in the surface string.

3.5 Lexicon, crosslinguistic variation and parameters

The issue of modelling crosslinguistic variation turned out to be nontrivial both from an empirical and implementation point of view.

The lexicon is created at runtime by using three sources, all which exist as independent text files. One file contains a list of universal morphemes, such as tense ('past, present'), interrogativization ('who, what'), transitivity ('admires Mary'), conjunction ('and') and many others that are assumed to be grammaticalized in many or perhaps in all languages.[32] Another lexical file contains language-specific lexical items, mostly open-class content words. Finally, there is a third file that contains the lexical redundancy rules. A lexical redundancy rule can be thought of as specifying a cluster of default features. For example, it is specified in the lexical redundancy rule list that any preposition can or must have a DP-complement. Each lexical redundancy rule is defined by using a formula $f_1 \ldots f_n \rightarrow g_1 \ldots g_n$, in which the presence of features $f_1 \ldots f_n$ in a lexical item triggers the presence of features $g_1 \ldots g_n$. If a lexical redundancy rule is violated in the language-specific lexicon, the latter wins; hence lexical redundancy rules specify default behavior that can be overridden by the language specific properties. When a lexicon or a lexical element is created at runtime, its properties are first retrieved from the language specific lexicon and then processed through lexical redundancy rules.

This architecture raises the question, however, of how to define language specific features for functional elements. English finite T, for example, must contain a feature requiring strict EPP behavior, but this feature should not be present if the language has a consistent noncanonical VSO profile. In Finnish, finite verbs exhibit EPP behavior but the system is based on definiteness or topicness, not grammatical subjecthood.

In some of the early implementations this problem was addressed by using a third lexical layer or computational prism that applied parametric rules to lexical elements when they were retrieved from the lexicon. A parametric rule was a rule that attributed behaviours to functional lexical elements during lexical retrieval on the basis of the input language. The whole component looked completely ad hoc, however, and it was impossible to me to think that crosslinguistic parameters were based on an explicit lexical transformation algorithm. Thus, the system was replaced by a mechanism, used in the present study as well, in which lexical redundancy rules could be specified in such a way that they applied only to selected languages. For example, the English EPP rule could be stated as a lexical redundancy rule 'LANG:EN, T, FIN $\rightarrow$ EPP', which would generate an EPP feature to all finite T elements in English during lexical retrieval.

While this system can be used to *describe* crosslinguistic variation, it fails to capture the fact that many language-specific lexical redundancy rules apply in clusters – their basic parametric quality. In Finnish, for example, there is a generalization according to which any phrase that is pied-piped to the scope position at SpecCP must have the operator feature at its edge (Huhmarniemi 2012). This means

[32] It does not matter if there are languages that do not use some of these items, or if there are languages that use items that are not in this list; the reason they are contained in a separate file is to make finding and editing them easier. Also, by putting an item into this list the researcher is tentatively claiming that it should be found from several or all languages, in some form. Most or in fact all of these elements are functional heads and inflectional features.

that all functional items that head pied-piped phrases in Finnish must project the required syntactic specifier position, a behavior that is absent in English. Unless we structure the lexicon in such a way that these properties are honoured, either Finnish or English operator constructions will be derived wrongly. While it is indeed possible to describe such behavior in individual lexical redundancy rules, the underlying assumption would then be that they could be distributed also randomly throughout the functional lexicon, which does not seem to be the case. In other words, any lexical rule is as good as any other – they are all stipulated formulas that could be written into the lexical redundancy file – thus there is no natural way to capture properties that cluster together or exhibit some form of internal logic or a general pattern. In this sense the third layer architecture was not entirely incorrect; it forced the processing into a narrowly defined pathway, even though this forcing looked mysterious.

The problem remains unsolved. It is connected to a large problem: what ultimately determines the possible forms of the lexical redundancy rules? Arbitrary lexical rules are unrealistic also from the point of view of language acquisition. It is not possible that the child generates arbitrary redundancy rules into an empty storage medium of some kind as a function of his or her linguistic experience; the system must be constrained by something. The principles and parameters framework used grammatical parameters to achieve the required constraints, but what else expect some type of ad hoc code could force the grammar into some parametric space? A brute ad hoc parametrization of the grammatical rules or algorithms themselves would not solve this problem since the parametric properties must be reflected also in the lexicon.

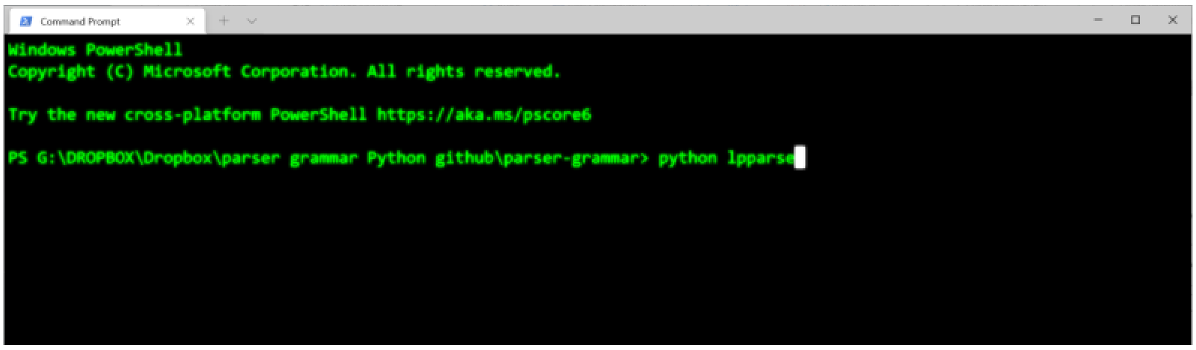
These questions raise another problem: how to implement the notion of 'language of a sentence' and direct the processing of the sentence to a correct model/lexicon. This is currently addressed by creating separate models for speakers of all languages present in the lexicon, and then by selecting the model based on the language of the words that appear in each input sentence. There is a separate language guesser module that accomplishes this. Thus, an input sentence that contains mostly English words will be directed for a parser model that uses the English functional lexicon, hence to a parser that we can think of "modelling an English speaker." This means that even if the input sentence would contain some words from Finnish, it would still get processed by using the English functional lexicon/parametrization; it is not possible to shift the language in the middle of the sentence.

3.6 Source code

The implementation was done in Python programming language (version 3.x) without external libraries with the exception of the pyglet library that is used in drawing the phrase structure trees. Thus, if the user wants the algorithm to provide phrase structure trees (shown in this document), this library must be installed in the local machine alongside Python itself. The version of the program that was used to simulate the data in this study is 9.0. Python can be installed by following the instructions at https://www.python.org/.

The source code is stored at https://github.com/pajubrat/parser-grammar and can be installed from there into a local machine by following regular installation instructions, also provided in the software documentation (Brattico 2019) that is available in the /docs subfolder. Standard installation (not cloning) means that the user copies the material from the above address to a local machine, into some local directory that the user can select freely. That directory then becomes the root directory of the program. The source code that the user downloads from the repository consists of the latest version of the program, also called the master branch. This is typically but not always a functional version of the code and should do something meaningful when downloaded. The version that was used in this study is stored in the branch *long-head-movement-(Study-4d)*. Branches *4e*, *f* and so on will be used if improvements or corrections are required. Currently these branches do not exist.

The source code, when downloaded to the local machine, can be executed by writing "python lpparse" into the command prompt that is in the selected root directory in the local machine. The user must have Python installed and configured on the local machine.

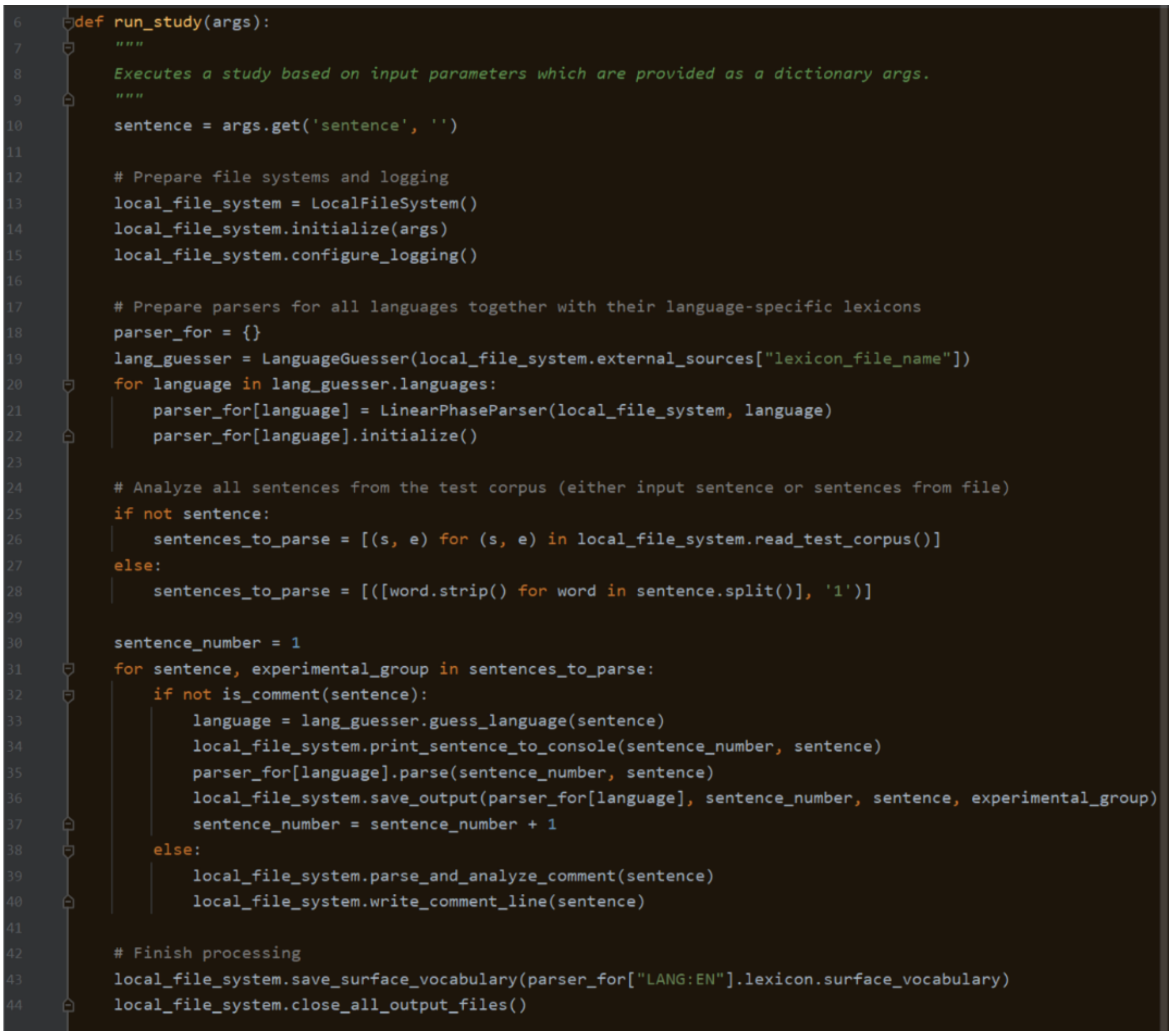


```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS G:\DROPBOX\Dropbox\parser grammar Python github\parser-grammar> python lpparse
```

Figure 16. Execution of the program. The program package must be installed into some directory in the local machine, freely selected and/or created by the user. Thus, the path shown above only corresponds to the location where it is in one of my own local machines.

This will execute the program package. The source code consists of ordinary text files that are read by a Python interpreter, which translates them into concrete actions performed by the computer (calculating results, reading and writing files). Hence the user can examine the program with any text editor, such as Windows notepad. The source code can be found from the folder /lpparse and is contained in the text files named xxxx.py. The.py extension refers to a Python source code module. For example, to examine how head reconstruction works, the user can open the module head_reconstruction.py with Windows notepad and find the actual code.

When then user writes the above command into the command prompt, the program will attempt to read the __main__.py module from the folder /lpparse, which then executes the master script called main.py.[33] This module in turn has a function run_study(), which runs one whole study. The code for run_study() is provided in Figure 17.



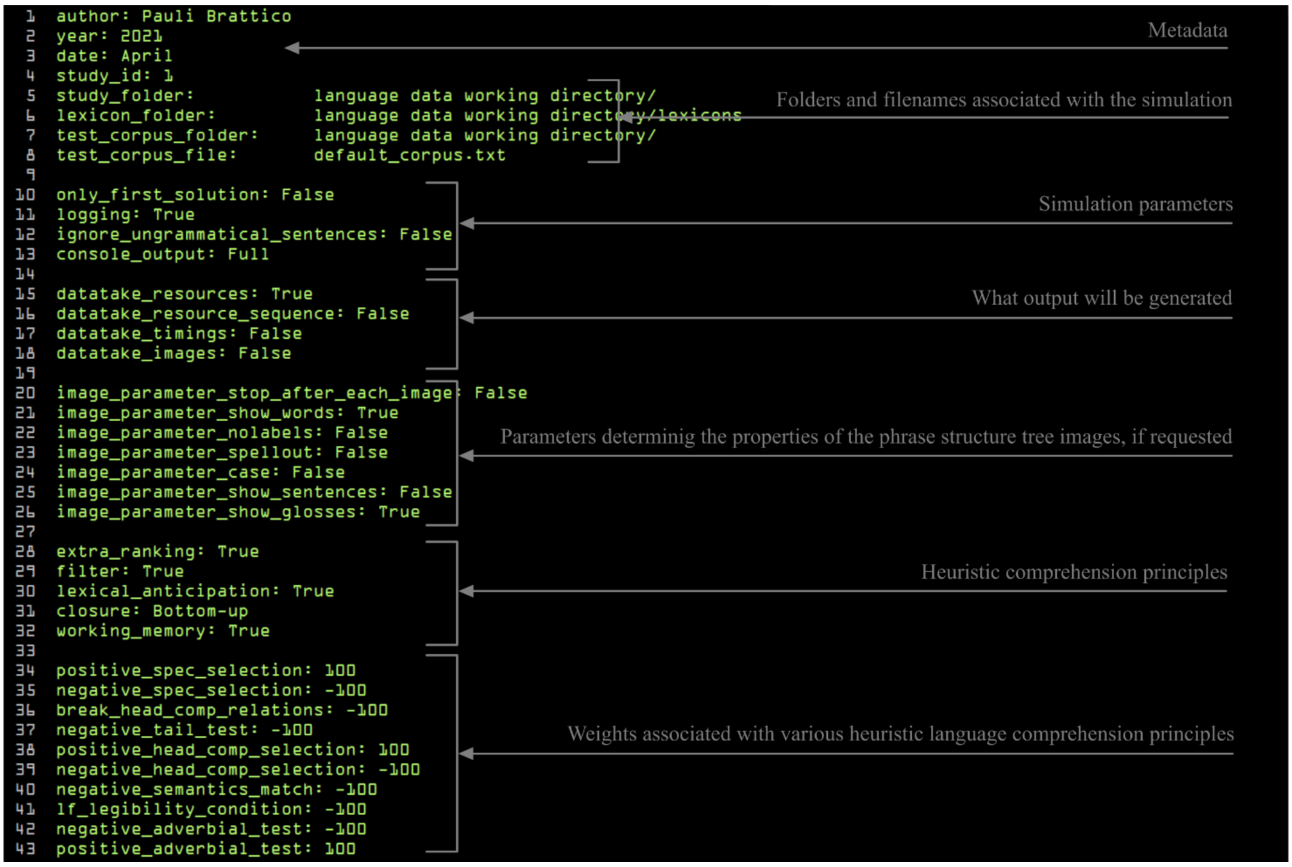
```python
6  def run_study(args):
7      """
8      Executes a study based on input parameters which are provided as a dictionary args.
9      """
10     sentence = args.get('sentence', '')
11
12     # Prepare file systems and logging
13     local_file_system = LocalFileSystem()
14     local_file_system.initialize(args)
15     local_file_system.configure_logging()
16
17     # Prepare parsers for all languages together with their language-specific lexicons
18     parser_for = {}
19     lang_guesser = LanguageGuesser(local_file_system.external_sources["lexicon_file_name"])
20     for language in lang_guesser.languages:
21         parser_for[language] = LinearPhaseParser(local_file_system, language)
22         parser_for[language].initialize()
23
24     # Analyze all sentences from the test corpus (either input sentence or sentences from file)
25     if not sentence:
26         sentences_to_parse = [(s, e) for (s, e) in local_file_system.read_test_corpus()]
27     else:
28         sentences_to_parse = [([word.strip() for word in sentence.split()], '1')]
29
30     sentence_number = 1
31     for sentence, experimental_group in sentences_to_parse:
32         if not is_comment(sentence):
33             language = lang_guesser.guess_language(sentence)
34             local_file_system.print_sentence_to_console(sentence_number, sentence)
35             parser_for[language].parse(sentence_number, sentence)
36             local_file_system.save_output(parser_for[language], sentence_number, sentence, experimental_group)
37             sentence_number = sentence_number + 1
38         else:
39             local_file_system.parse_and_analyze_comment(sentence)
40             local_file_system.write_comment_line(sentence)
41
42     # Finish processing
43     local_file_system.save_surface_vocabulary(parser_for["LANG:EN"].lexicon.surface_vocabulary)
44     local_file_system.close_all_output_files()
```

Figure 17. The code for the main.py in the current master branch. The logic is explained in the main text.

Lines (13-15) initialize the simulation. The most important function of these operations is to read the configuration file that sets a variety of internal parameters for the simulation. The configurational file is currently named config_study.txt and is located in the root directory (missing files and parameters are substituted by default behavior, ignored here).[34] Figure 18 shows a screenshot of this file at present writing together with explanations of the most important parameters.

[33] The module __main__.py parses command line arguments. It can be used to divert processing to special purpose programs or functions, such as macro scripts that run several studies.

[34] Any parameter can also be given as a command line argument, which are convenient when writing scripts that execute several studies.


Metadata
Folders and filenames associated with the simulation
Simulation parameters
What output will be generated
Parameters determinig the properties of the phrase structure tree images, if requested
Heuristic comprehension principles
Weights associated with various heuristic language comprehension principles

```
 1  author: Pauli Brattico
 2  year: 2021
 3  date: April
 4  study_id: 1
 5  study_folder:          language data working directory/
 6  lexicon_folder:        language data working directory/lexicons
 7  test_corpus_folder:    language data working directory/
 8  test_corpus_file:      default_corpus.txt
 9
10  only_first_solution: False
11  logging: True
12  ignore_ungrammatical_sentences: False
13  console_output: Full
14
15  datatake_resources: True
16  datatake_resource_sequence: False
17  datatake_timings: False
18  datatake_images: False
19
20  image_parameter_stop_after_each_image: False
21  image_parameter_show_words: True
22  image_parameter_nolabels: False
23  image_parameter_spellout: False
24  image_parameter_case: False
25  image_parameter_show_sentences: False
26  image_parameter_show_glosses: True
27
28  extra_ranking: True
29  filter: True
30  lexical_anticipation: True
31  closure: Bottom-up
32  working_memory: True
33
34  positive_spec_selection: 100
35  negative_spec_selection: -100
36  break_head_comp_relations: -100
37  negative_tail_test: -100
38  positive_head_comp_selection: 100
39  negative_head_comp_selection: -100
40  negative_semantics_match: -100
41  lf_legibility_condition: -100
42  negative_adverbial_test: -100
43  positive_adverbial_test: 100
```

Figure 18. Screenshot from the configuration file determining the input parameters.

The most important parameters are the file and folder names (lines 5-8) which the program uses to read all inputs (test corpus and lexicons) and where it produces the output (parameter study_folder). Each individual study should use its own folder that contains all the required materials, perhaps with the exclusion of the lexicons that I currently load form a master lexicon folder and which remains mostly the same from one study to the next.[35]

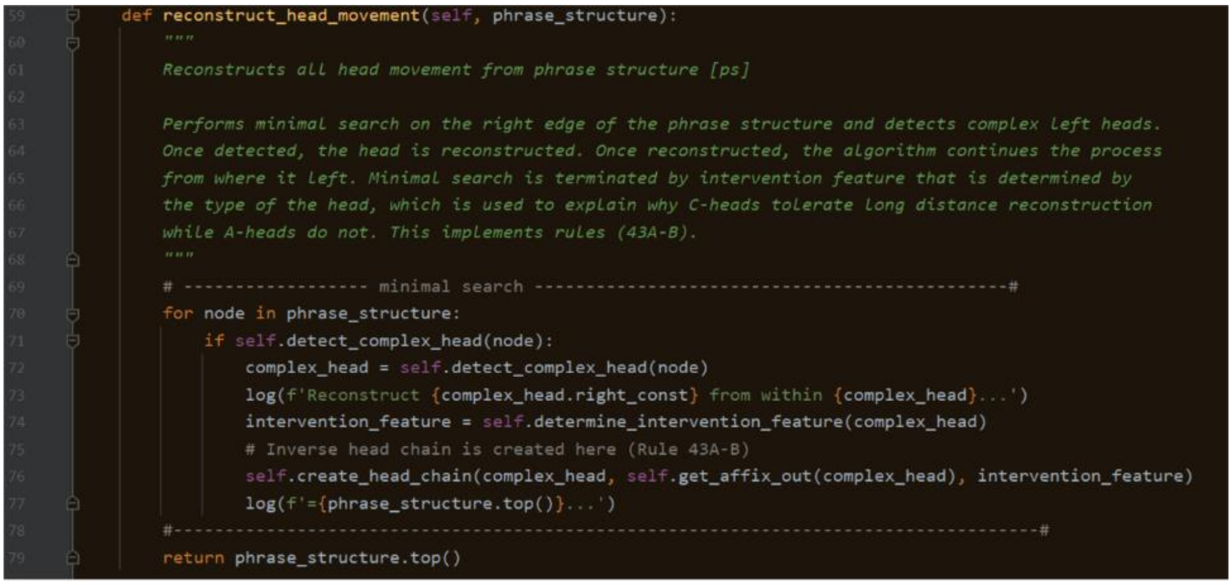
Next the function prepares a "brain model" for the speaker of each language present in the lexicon (lines 17-22) and initializes them. These models are matched with the languages of the test sentences at runtime. Finnish sentences will be processed by a Finnish speaker, English sentences by English speaker, and so on. Bilingual speakers are assumed to shift from one model to another based on the input. Separate models are required for each language because the properties of functional items potentially differ from one language to the next. The program then reads the input sentences from the test corpus (lines 24-28), as defined in the configuration file (Figure 19), and sends them, one by one, to the parser model based on the language (lines 30-40).

A parser model is defined in the class LinearPhaseParser, which uses parse() for parsing the sentence it receives. This function is called for each input sentence (line 35). This module and its

[35] The lexicon used in connection with any particular study should still be stored together with the rest of the files.

functions define what could be characterised as an "executive control" for the parsing process, thus step-by-step instructions on what to do with each input and in which order. All submodules of the system (lexico-morphological module, transfer, syntax-semantic interface and others) are instantiated inside this class from their own modules, so that the LinearPhaseParser class can now be thought of as a general model of the speaker-hearer (of each language).[36] It will also perform many auxiliary tasks, such as recording performance metrics. The parsing function performs necessary initialization and then calls a recursive parsing function (parse_new_item()) which consumes words from the input and sends them through the pipeline as defined in Figure 2. The phrase structure argument of this function represents the current syntactic analysis in current working memory. Once all words have been consumed, the function attempts to complete processing (function complete_processing()) which tries to transfer the result into the syntax-semantics interface, creates a semantic interpretation if successful and performs several logging functions. Control is then handed back to the main loop which backtracks (exits from the recursion in a well-defined order).

Head reconstruction is defined in its own module head_reconstructions.py and is part of the transfer (transfer.py). The core content of the analysis is contained in two functions. One is reconstruct_head_movement(), which performs minimal search, detects complex heads and reconstructs them. It is shown in Figure 19.

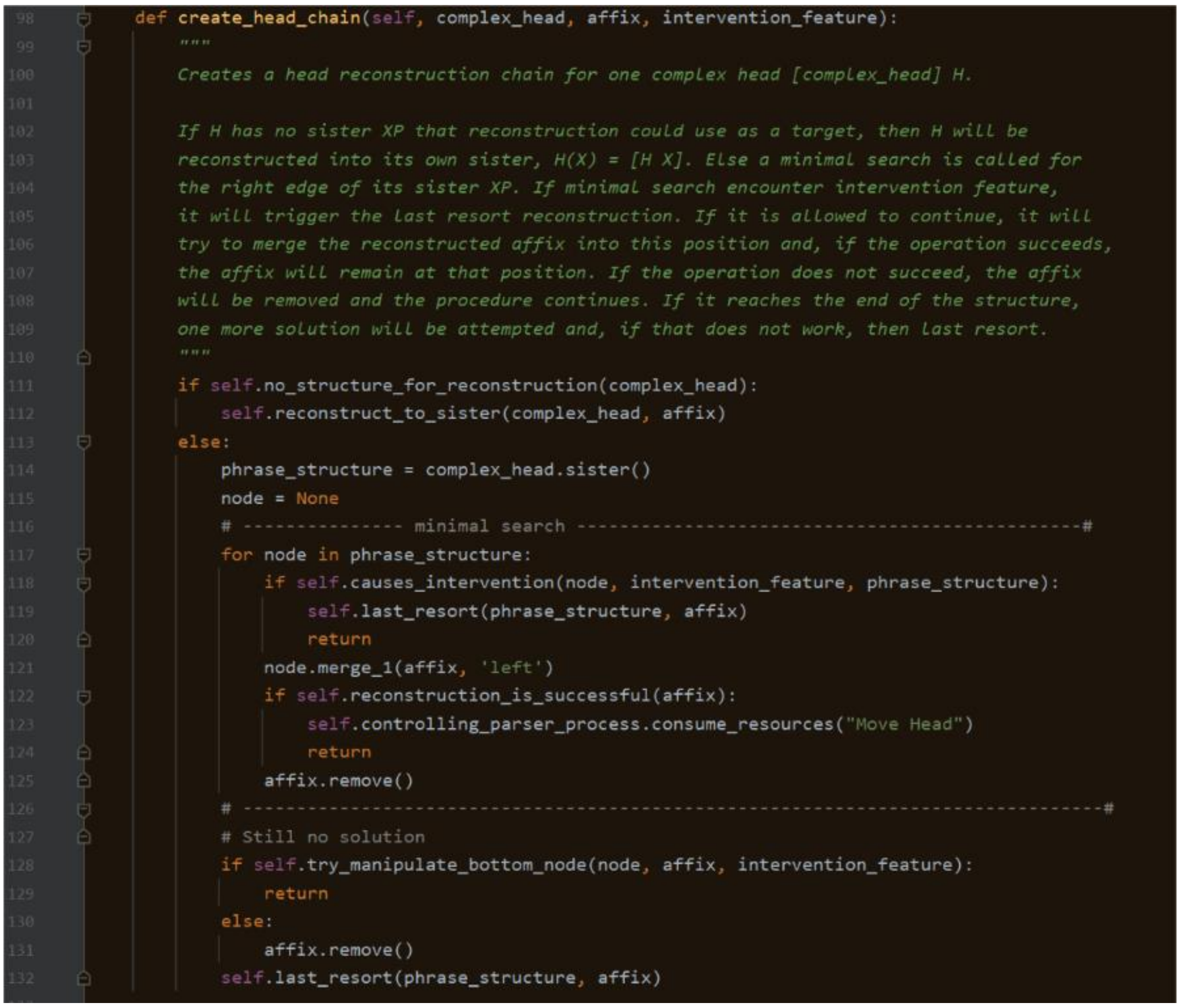


```python
    def reconstruct_head_movement(self, phrase_structure):
        """
        Reconstructs all head movement from phrase structure [ps]

        Performs minimal search on the right edge of the phrase structure and detects complex left heads.
        Once detected, the head is reconstructed. Once reconstructed, the algorithm continues the process
        from where it left. Minimal search is terminated by intervention feature that is determined by
        the type of the head, which is used to explain why C-heads tolerate long distance reconstruction
        while A-heads do not. This implements rules (43A-B).
        """
        # ----------------- minimal search -------------------------------------------#
        for node in phrase_structure:
            if self.detect_complex_head(node):
                complex_head = self.detect_complex_head(node)
                log(f'Reconstruct {complex_head.right_const} from within {complex_head}...')
                intervention_feature = self.determine_intervention_feature(complex_head)
                # Inverse head chain is created here (Rule 43A-B)
                self.create_head_chain(complex_head, self.get_affix_out(complex_head), intervention_feature)
                log(f'={phrase_structure.top()}...')
        #-----------------------------------------------------------------------------#
        return phrase_structure.top()
```

Figure 19. Implementation of minimal search.

[36] An alternative is an architecture in which the parser module is embedded inside some still more large executive system. This could be used if the output of the syntactic pathway were used for some other purpose, such as reasoning or communication.

The algorithm implements minimal search (line 70) by enumerating the phrase structure on its right edge by following labelling, and then detects complex heads X(Y) (line 71). Line 74 defines the intervention feature (notice the stipulative character of this component) and line 76 triggers head chain creation which reconstructs the head X(Y) → [X [...Y...]]. Head reconstruction is defined in function create_head_chain() and is provided in Figure 20.



```
98      def create_head_chain(self, complex_head, affix, intervention_feature):
99          """
100         Creates a head reconstruction chain for one complex head [complex_head] H.
101
102         If H has no sister XP that reconstruction could use as a target, then H will be
103         reconstructed into its own sister, H(X) = [H X]. Else a minimal search is called for
104         the right edge of its sister XP. If minimal search encounter intervention feature,
105         it will trigger the last resort reconstruction. If it is allowed to continue, it will
106         try to merge the reconstructed affix into this position and, if the operation succeeds,
107         the affix will remain at that position. If the operation does not succeed, the affix
108         will be removed and the procedure continues. If it reaches the end of the structure,
109         one more solution will be attempted and, if that does not work, then last resort.
110         """
111         if self.no_structure_for_reconstruction(complex_head):
112             self.reconstruct_to_sister(complex_head, affix)
113         else:
114             phrase_structure = complex_head.sister()
115             node = None
116             # -------------- minimal search ---------------------------------------------#
117             for node in phrase_structure:
118                 if self.causes_intervention(node, intervention_feature, phrase_structure):
119                     self.last_resort(phrase_structure, affix)
120                     return
121                 node.merge_1(affix, 'left')
122                 if self.reconstruction_is_successful(affix):
123                     self.controlling_parser_process.consume_resources("Move Head")
124                     return
125                 affix.remove()
126             # ---------------------------------------------------------------------------#
127             # Still no solution
128             if self.try_manipulate_bottom_node(node, affix, intervention_feature):
129                 return
130             else:
131                 affix.remove()
132             self.last_resort(phrase_structure, affix)
```

Figure 20. Implementation of the head reconstruction algorithm.

Lines 111-112 implement the special edge case in which the complex head is itself the bottom right constituent, in which case there is no structure that could be targeted; it targets the sister (X(Y) → [X Y]). We perform minimal search (lines 116-126) and examine each possible position for the head (line 121) and evaluate if the position satisfies the conditions for gap (line 122). If it does, the head remains in that position and the operation terminates (line 124); if not, search continues. Intervention is calculated at line 118, which, if it happens, causes the algorithm to implement the last resort solution X(Y) → [X [Y...]] defined in its own function. Lines 126-131 handle an edge case in which the gap position exists inside the bottom right node, for example, if Y must be selected by Z but Z is still inside the bottom right node, W(Z).

References

Altmann, Gerry and Yuki Kamide. 1999. "Incremental Interpretation at Verbs: Restricting the Domain of Subsequent Reference." *Cognition* 73:247–64.

Altmann, Gerry and Mark Steedman. 1988. "Interaction with Context during Human Sentence Processing." *Cognition* 30(3):191–238.

Baker, Mark. 1988. Incorporation. A Theory of Grammatical Function Changing. Chicago: University of Chicago Press.

Brattico, Pauli. 2019. Computational Implementation of a Linear Phase Parser. Framework and Technical Documentation (Version 9.0). Pavia.

Brattico, Pauli. 2020. "Finnish Word Order: Does Comprehension Matter?" *Nordic Journal of Linguistics* 44(1): 38–70.

Brattico, Pauli and Cristiano Chesi. 2020. "A Top-down, Parser-Friendly Approach to Operator Movement and Pied-Piping." *Lingua* 233:102760.

Brattico, Pauli, Saara Huhmarniemi, Jukka Purma, and Anne Vainikka. 2013. "The Structure of Finnish CP and Feature Inheritance." *Finno-Ugric Languages and Linguistics* 2(2):66–109.

Cann, Ronnie, Ruth Kempson, and Lutz Marten. 2005. *The Dynamics of Language: An Introduction (Syntax and Semantics, Volume 35)*. Amsterdam: Elsevier Academic Press.

Chesi, Cristiano. 2012. Competence and Computation: Toward a Processing Friendly Minimalist Grammar. Padova: Unipress.

Chomsky, Noam. 1957. *Syntactic Structures*. The Hague: Mouton.

Chomsky, Noam. 1964. Current Issues in Linguistic Theory. Mouton.

Chomsky, Noam. 1965. *Aspects of the Theory of Syntax*. Cambridge, MA.: MIT Press.

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

Chomsky, Noam. 2001. “Derivation by Phase.” Pp. 1–37 in *Ken Hale: A Life in Language*, edited by M. Kenstowicz. Cambridge, MA.: MIT Press.

Chomsky, Noam. 2008. “On Phases.” Pp. 133–66 in *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, edited by C. Otero, R. Freidin, and M.-L. Zubizarreta. Cambridge, MA.: MIT Press.

Demberg, Vera, Frank Keller, and Alexander Koller. 2013. “Incremental, Predictive Parsing with Psycholinguistically Motivated Tree-Adjoining Grammar.” *Computational Linguistics* 39(4):1025–66.

Hale, John. 2014. *Automaton Theories of Human Sentence Comprehension*. Stanford, CA: CSLI Publications.

Huhmarniemi, Saara. 2012. *Finnish A´-Movement: Edges and Islands*. University of Helsinki, Helsinki.

Huhmarniemi, Saara and Pauli Brattico. 2015. “The Finnish Possessive Suffix.” *Finno-Ugric Languages and Linguistics* 4(1–2):2–41.

Keller, Frank. 2010. “Cognitively Plausible Models of Human Language Processing.” Pp. 60–67 in Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics: Short Papers.

Konieczny, Lars. 2000. “Locality and Parsing Complexity.” *Journal of Psycholinguistic Research* 29(6):627–45.

Lema, José and María-Luisa Rivero. 1990. “Long Head Movement: ECP vs. HMC.” *Proceedings of NELS* 20:333–47.

Manninen, Satu. 2003. *Small Phrase Layers: A Study of Finnish Manner Adverbials*. Amsterdam/Philadelphia: John Benjamins.

Marr, David. 1982. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information.* Cambridge, MA.: MIT Press.

Marslen-Wilson, William D. 1973. “Linguistic Structure and Speech Shadowing at Very Short Latencies.” *Nature* 244:522–33.

Matushansky, Ora. 2006. “Head Movement in Linguistic Theory.” *Linguistic Inquiry* 37:69–109.

Merchant, Douglas. 2019. “Idioms at the Interface(s): Towards a Psycholinguistically Grounded Model of Sentence Generation.” University of Georgia.

Phillips, Colin. 1996. “Order and Structure.” Cambridge, MA.

Phillips, Colin. 2003. “Linear Order and Constituency.” *Linguistic Inquiry* 34:37–90.

Platzack, Christer. 2013. “Head Movement as a Phonological Operation.” Pp. 21–43 in *Diagnosing syntax*, edited by L.-S. Lisa Cheng and N. Corver. Oxford: Oxford University Press.

Rivero, María-Luisa. 1991. “Long Head Movement and Negation: Serbo-Croatian vs. Slovak and Czech.” *The Linguistic Review* 8:319–51.

Rizzi, Luigi. 1990. *Relativized Minimality*. MIT Press.

Rizzi, Luigi. 1997. “The Fine Structure of the Left Periphery.” Pp. 289–330 in *Elements of Grammar*, edited by L. Haegemann. Amsterdam: Kluwer.

Roberts, Ian. 1993. Verbs and Diachronic Syntax. A Comparative History of English and French. Amsterdam: Kluwer Academic Press.

Roberts, Ian. 2010. Agreement and Head Movement: Clitics, Incorporation, and Defective Goals. Cambridge, MA.: MIT Press.

Ross, John Robert. 1967. “Constraints on Variables in Syntax.” MIT, Cambridge, MA.

Schoorlemmer, Erik and Tanja Temmerman. 2012. “Head Movement as a PF-Phenomenon: Evidence from Identity under Ellipsis.” Pp. 232–40 in *West Coast Conference on Formal Linguistics (WCCFL) 29*.

Tanenhaus, M. K., M. J. Spivey-Knowlton, K. M. Eberhard, and J. C. Sedivy. 1995. “Integration of Visual and Linguistic Informationin Spoken Language Comprehension.” *Science* 268(5217):1632–34.

Travis, Lisa. 1984. “Parameters and Effects of Word Order Variation.” Dissertation, MIT.

Vilkuna, Maria. 1989. Free Word Order in Finnish: Its Syntax and Discourse Functions. Helsinki: Finnish Literature Society.

Vilkuna, Maria. 1995. “Discourse Configurationality in Finnish.” Pp. 244–68 in *Discourse Configurational Languages*, edited by K. É. Kiss. Oxford: Oxford University Press.