Supplementary information for the article "Binding in Finnish and the language-cognition interface" published in *Journal of Uralic Linguistics*

Pauli Brattico

2025

Abstract. This document provides technical supplementary information for the article "Binding in Finnish and the language-cognition interface" published in *Journal of Uralic Linguistics*.[1] It contains a detailed description of the computational methods and results.

[1] Original article: Brattico, P. (2025). Binding in Finnish and the language-cognition interface. *Journal of Uralic Linguistics 4(2)*.

1 Introduction

1.1 Binding

This study develops a hypothesis of binding and demonstrates that it suffices to calculate a dataset probing binding configurations in English and Finnish. The essence of the hypothesis is that much like the use of articles, demonstratives, word order, prosody and other devices, binding restricts the search space for possible denotations during assignment generation. The relevant restrictions are structure-dependent because they rely on a computational system of semantic working memory constructed on the basis of structural properties at the LF-interface (syntax-semantics interface). The hypothesis therefore assumes that binding takes place at the "outer edge" of language, between syntax and global semantics (or cognition more generally), and should be contrasted with hypotheses developing syntax-internal binding mechanisms. Because the hypothesis relies on comprehension, it was tested by a model in which grammatical derivations are generated on the basis of input expressions consumed incrementally from left to right, one morpheme at a time. It is assumed that real natural language comprehension proceeds in this way.[2]

1.2 Computational generative grammar methodology

A scientific theory or hypothesis is said to be justified only if it suffices to calculate or logically imply the data of interest. The other requirement is that it is also necessary, which is interpreted (not entirely unproblematically) to mean that the theory is the simplest imaginable alternative that satisfies the first condition. Together these two conditions exhaust what we mean by justification. Informal and informed argumentation is not the same thing as scientific justification (Brattico 2019: §3.2).

To satisfy these requirements the theory, dataset and the inferential relations between the two must be provided in an unambiguous (i.e. completely formal) form. The computational methodology adopted in the present study is meant to satisfy this criteria. In particular because the relationship between the theory and data is computationally complex due to the combinatorial nature of both, we have to use computational techniques to penetrate the complexity barrier. Python was used for the implementation in the present study. See Brattico 2019, 2021c, 2021b, 2024 for a general discussion of the computational generative grammar methodology.[3]

Justification must be distinguished from discovery. The empirical substance of the theory is contained in its independent empirical assumptions which have to be discovered and formulated and which will ultimately contain its empirical content. Justification in the above sense, on the other hand, is a technical tool for the verification of the ideas presented in the theory against observed reality. Because the empirical content of the theory almost always reaches beyond what is available in the immediate, observed reality – as the former unlike the latter is concerned with invisible and abstract properties and processes – we must separate the

[2] The same binding model could be described and formalized in a reverse system in which the same representations and properties are generated by the standard bottom-up derivation. This alternative can describe the same computational mappings, but in my view without providing an equally realistic picture of the human language faculty and its linguistic information processing.
[3] At the time of writing there are two video series explaining the methodology, available in the YouTube platform, https://www.youtube.com/channel/UCa0vhL8xC5aSH8uvZgF6PgA (in English) and https://www.youtube.com/playlist?list=PL35upitLda1fkGLFrdYEEy5P1LBq74dZ3 (in Finnish).

two. We can think of justification as a necessary requirement for any idea to be taken seriously, but then consider it a trivial requirement that has to be satisfied before serious discussion of the real substance can begin.[4]

Calculation of linguistic data requires that the model reproduces expressions and their relevant properties in the dataset, in the exact form the original expressions are provided in the dataset. Reproducing attested or grammatical expressions and their properties is not sufficient; we must show that the model reproduces all ambiguities and none of the ungrammatical expressions and/or unattested properties. The latter conditions require that we explore the whole derivational search space, thus every empirical implication of the grammatical model.[5] The derivational search space is explored by a recursive *derivational search function* (Brattico 2019, 2024). In the case of a standard bottom-up theory (not assumed here), the derivational search function works by trying out all possible derivations. In the case of a comprehension-oriented left-to-right model, in which syntactic representations are generated incrementally and in part cyclically by consuming words from left to right, the derivational search function must try all derivations compatible with the given input expression. This is implemented in the present study by exploring all possible attachment solutions for each incoming word.

1.3 Background theory

The hypothesis was built on the existing LPG model ("linear phase grammar," see Brattico 2019). LPG is based on the assumption, originally inspired by Colin Phillip's MIT dissertation (Phillips, 1996), that the native format of human linguistic processing, in connection with both comprehension and production, is the linear left-to-right cycle in which words (and more generally morphemes inside words) are produced in a temporally asymmetric linear order and which yields both the concrete spellout forms and semantically interpreted structural representations as outputs, in particular the two interface representations LF (Logical Form)

[4] Most of the linguistic literature, including literature produced within the framework of generative grammar, does not satisfy these minimal requirements and therefore puts forward tentative working hypotheses, conjectures or speculation, depending on how much weight one is willing to grant to unproven, and therefore preliminary, theoretical constructs.
[5] Or alternatively grammar, grammatical theory or grammatical hypothesis, with very little interesting differences between the various notions.

and PF (Phonological Form).[6] Comprehension differs from production in that the PF-interface representations are generated from the (potentially ambiguous) surface strings, ultimately constructed in some manner at the sensorimotoric interfaces, whereas in production the PF-interface representations are generated from linearly ordered sequences of disambiguated "items" which feed the sensorimotoric interface (SM) and the syntactic interfaces. The linear order is what is being generated – it's the primary generative engine of the brain – while the rest constitutes a secondary effect.

Notice that the LPG is aimed at capturing both competence and performance: the former because it is required to be observationally and descriptively adequate, the latter because it is supposed to mirror real language comprehension and production. Furthermore, LPG is not an engineering parser trying to classify bulks of text; it is a model of the human language faculty and as such developed to capture an observationally and descriptively adequate theory of the human language.

In a certain sense the LPG models language by reversing the standard bottom-up cycle: instead of generating PF-interface representations and sensorimotoric output from LF-representations or meanings, it generates the latter from the former. But this reversal is, in part, irrelevant: in many cases the exact same mappings could be generated by using the standard bottom-up cycle. For example, many assumptions made in the present study concerning binding such as assignment generation could be implemented without major modifications by using the more standard bottom-up model. In other words, the two models – the left-to-right cycle and the bottom-up model – are notational variants over a significant range of data. The qualification "over a significant range of data" is important though: the models themselves are not notational variants, rather there might be very few substantial differences between the two when examined in the context of narrow datasets or otherwise restricted empirical phenomena.[7]

[6] LF refers to a representational system which serves as an interface between syntax and semantics; PF to a system which connects syntax to postsyntactic modules and thus maps the output of syntax into sensorimotoric output. ”Postsyntactic” means information processing that does not involve syntactic representations or operations, but instead things like linear order, nonstructural representations such as (syntactically nonstructured) lexical items, prosodic and phonological features, and other lower level properties, with "nonsyntactic" being the unifying feature.

[7] The left-to-right model subsumes the bottom-up model in the sense that it relies on bottom-up operations such as basic Merge (Brattico 2019: §4.2, 7.1.8). There are, however, several axiomatic

Let us consider the model first by looking only at the mappings while ignoring the underlying left-to-right algorithm. At the computational level LPG posits cognitive representational systems and mappings between them, as summarized in (1) below. The levels are the *sensorimotoric level* (SM), which represents input expressions provided as linear strings of phonological material expressed by string literals in the current study[8]; the syntactic *PF-interface* ("PF"), which is mapped to SM via linearization and other postsyntactic or nonsyntactic operations; the *LF-interface* level ("LF") which is created on the basis of PF by applying cyclic and noncyclic transformations and which forms a syntax-semantics interface representation interacting with semantic interpretation[9]; the *semantic/pragmatic systems* ("narrow semantics", NS) which provides the input expressions with semantic attributes; and finally *global cognition* ("G") which contains representations of objects that also the extralinguistic cognitive systems such as thinking and planning can access and manipulate.

(1) *Levels of representations and mappings*
   Sensorimotoric systems (SM)
   (Linearization and other nonsyntactic information processing)
   Phonological Form (PF)
   (Cyclic and noncyclic transformations)
   Logical Form (LF)
   (Semantic interpretation, narrow semantics)
   Global cognition (G)

All these levels play a role in binding. The SM-PF mapping analyses the input sentences, including various types of pronouns, and provides them with the first syntactic representations

properties of the bottom-up model which arguably receive elegant architectural explanations when looked from the perspective of the left-to-right model: locality of several operations such as A-chains can be deduced from the fact that they are performed cyclically when only local structure exists; CED-effects follow automatically from the left-to-right cycle which treats left constituents and adjuncts as phases (the working memory assumption of the present model, Brattico 2019: §4.4); transformations can be modelled as noise-tolerance operations; we can use performance data to constrain the model and in general formulate a model which uses the same computational operations of the UG for both comprehension and production.

[8] This approximation suffices for the present purposes since we are not trying to calculate and predict phonological or morphophonological properties.

[9] See Brattico 2019: §4.8.3 for the notions of "cyclic" and "noncyclic" in connection with the left-to-right assembly.

including lexical contents; PF-LF mapping handles reconstruction and abstracts away from surface variations such as displacement and the flexible word order property of Finnish; LF-NS mapping provides the semantic interpretation, including meanings related to the lexical features implied in the binding mechanism proposed in the main article; and finally the NS-G mapping creates denotations (or more generally, extensions) and assignments. The proposed binding mechanisms operate at the LF-NS-G interfaces, where the LF provides the notion of semantic working memory proposed in this study. These mappings constitute the language-cognition interface. Narrow semantics may be considered a "translation" between the two (Brattico 2019: §4.10).

The SM-PF mapping is potentially one-to-many, since it must capture ambiguities in the input expressions. PF-LF mapping is one-to-one. The syntax-cognition interface is one-to-many, since assignments generate semantic ambiguities. Semantic ambiguities are not provided in the form of semantic structures discovered by derivational search, but rather by means of sets of interpretations (here, sets of possible denotations, sets of possible assignments) linked with the LF-interface representations.

As already stated, at the algorithm level the model is based on the left-to-right assembly originally proposed by Phillips (1996), which reads linguistic information from the input expressions one morpheme at a time while creating incrementally a PF-interface representation (or a set of such representations due to the possibility of ambiguity). The PF representation is transformed into LF by cyclic and noncyclic transformations (Brattico 2019: §4.8). Because the operations are ordered in this way, the theory is particularly suitable for modelling language comprehension (Brattico & Chesi, 2020).[10] The sequence of computational steps from SM into various syntactic representations and ultimately to semantic attributes constitutes a *derivation*. Because the algorithm maps input expressions at SM into various syntactic and semantics attributes generated by the derivation, including grammaticality, it provides a characteristic function for a particular generative grammar, where the term "generative" should be understood in its original technical sense of denoting a grammar which defines a set of

[10] We ignore neuro- and psycholinguistic realism. It should be noted that the same mappings could be generated by an algorithm that implements the mappings in reverse order, by beginning from G and proceeding towards the SM.

expressions. The theory is justified if and only if the characteristic function is observationally, descriptively and explanatory adequate.[11] At the implementation level the model and the surrounding computational infrastructure is written in Python (Brattico, 2019).[12] For the Python implementation of binding discussed in this study, see Section 4 and Brattico (2019: §7.5.4).

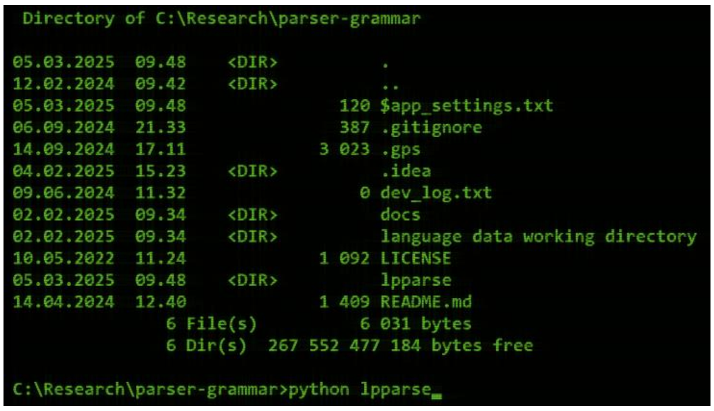
The whole approach makes certain assumptions about meaning, semantic interpretation and semantic representations, a controversial area in and itself (Brattico 2019: §4.10). There is no compositional language of though; rather, the output of the syntactic processing pipeline guides or affects the construction, maintenance and attentional framing of semantic objects, the latter which are internal representations of possible external objects and their properties and mutual relations, hence representations of persons instead of the real persons. There is no linguistic-like system with compositional truth-conditions. The whole operation is internal. If the speaker mentions a person or object, a corresponding semantic representation will be created in the mind of the addressee irrespective of whether such thing exists in the real world. Adding truth conditions into the system would require another medium, a computational representation of the 'real world', together with a mind-external mechanism or mechanisms for mediating between what is represented internally and externally. There must be psychological mechanisms for coordinating the two realities, but it is not clear to the present author if these mechanisms are part of language and linguistic processes, including part of the language-cognition interface: language seems to be able to describe purely imaginary realities as well as reality with no observable differences between the two modes.

[11] To recapitulate: the model is observationally adequate iff it is both sufficient and necessary to calculate (replicate, imply) the grammaticality judgments in the dataset; it is descriptively adequate iff it is both sufficient and necessary to calculate all other linguistic properties of interest (e.g., ambiguities, semantic interpretations, syntactic dependencies) in the dataset; it is explanatorily adequate iff it satisfies the two previous conditions for a multilingual dataset.
[12] The relevant software documentation can be acquired from `https://github.com/pajubrat/parser-grammar/blob/master/docs/Old%20documentation/Computational%20implementation%20of%20a%20linear%20phase%20parser%20(version%2020.1).pdf` which describes the version (20.1) used in the present study. This version is likely to be outdated soon and should only be used for replication.

2 Methods

2.1 Replication

The study can be replicated by downloading the source code from the source code repository and by running the main script.[13] The main script can be executed by writing `python lpparse` to the command prompt in the local installation directory:



```
 Directory of C:\Research\parser-grammar

05.03.2025  09.48    <DIR>          .
12.02.2024  09.42    <DIR>          ..
05.03.2025  09.48               120 $app_settings.txt
06.09.2024  21.33               387 .gitignore
14.09.2024  17.11             3 023 .gps
04.02.2025  15.23    <DIR>          .idea
09.06.2024  11.32                 0 dev_log.txt
02.02.2025  09.34    <DIR>          docs
02.02.2025  09.34    <DIR>          language data working directory
10.05.2022  11.24             1 092 LICENSE
05.03.2025  09.48    <DIR>          lpparse
14.04.2024  12.40             1 409 README.md
               6 File(s)          6 031 bytes
               6 Dir(s)  267 552 477 184 bytes free

C:\Research\parser-grammar>python lpparse_
```

Figure. The study can be replicated by writing `python lpparse` in the installation directory.

Here the local installation directory is `C:\Research\prase-grammar`. The key directories of the installation package are `/docs`, which contains documentation and other documents (article preprints, supplementary documents, including the present document), `/lpparse`, which contains the source code, `/language data working directory` which contains language data (lexicons, datasets, outputs), including the input and output data of the present study. For more detailed instructions on how to replicate the study see Brattico (2019: §2).

2.2 Design and procedure

The master script reads the dataset file and processes each sentence independently (taking conversations into account). It tries to find a derivation for each sentence, and if one or several

[13] The version of the source code that was used to calculate the data and which must be used for replication purposes is contained in the branch `Binding-(Study-10)` in the parser-grammar project (`https://github.com/pajubrat/parser-grammar`) and which can be accessed directly from `https://github.com/pajubrat/parser-grammar/tree/Binding-(Study-10)`. The latest version of model should not be used for replication of the present study.

is found, the expression is judged grammatical; if not, ungrammatical. A model that judges an expression grammatical if and only if it is judged grammatical by native speakers is said to be *observationally adequate*. The number of errors in grammaticality judgments is also reported after the script has finished. If the model finds several syntactic analyses for the input expression, the expressions is judged *ambiguous*. Grammatical sentences are provided with a syntactic analysis (or several) and semantic interpretation (or several). If the analyses match with native speaker intuition and are not implausible from the point of view of linguistic theory, the model satisfies the condition of *descriptive adequacy*.

The script creates several raw output data files when it is executed. The most important are the *derivational log file*, which contains a record of all calculation steps executed during the derivations[14]; the *results file*, which contains summaries of the results such as syntactic analyses and semantic interpretations[15]; *observational adequacy error file* which contains a list of expressions in which the native speaker and predicted grammaticality judgments did not match; *descriptive adequacy error file* which contains a list of expressions in which the binding properties provided by native speakers and the model did not match. For a detailed description of the raw output files generated by the software, see Brattico 2019: §6. The model can also create phrase structure images for all expressions judged grammatical by the model; see Section 2.4.

2.3 Dataset

The *dataset* is a set of input expressions (usually sentences) the model processes one at the time in the order they are provided in the *dataset file* (Brattico 2019: §6.2.2).[16] The dataset file contains the expressions themselves together with comments such as classification headers which help to organize the data, and glossing. Each expression is marked for grammaticality

[14]`https://github.com/pajubrat/parser-grammar/blob/master/language%20data%20working%20directory/study-10-binding-theory/derivations.zip`

[15]`https://github.com/pajubrat/parser-grammar/blob/master/language%20data%20working%20directory/study-10-binding-theory/binding_theory_corpus_results_FINAL.txt`

[16]`https://github.com/pajubrat/parser-grammar/blob/master/language%20data%20working%20directory/study-10-binding-theory/binding_theory_corpus.txt`

in the dataset, which the model compares with its own predictions to assess observational adequacy.

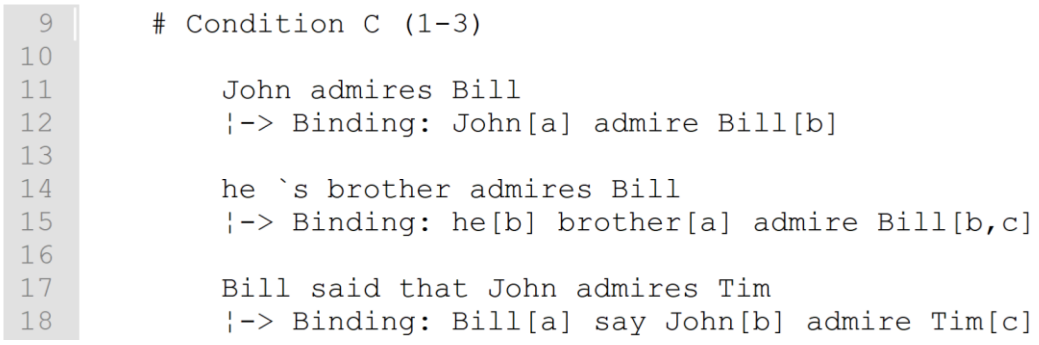
The input expressions are linear sequences of items defined as string literals approximating their phonological properties and other properties, such as morpheme boundaries, as interpreted and defined by the model. Lower-level processing is ignored; segmentation, phonological encoding and in some cases morphological parsing are therefore presupposed. Conversations are represented by semicolons (";") which positions separate sentences inside the same conversation, in which case they will share the global discourse inventory. The inventory is created in the order referential expressions are presented in the input sentences. Other special symbols were as follows: `#` morpheme boundary (separates possessive suffixes from stems in the present study); = clitic boundary (separates Finnish left peripheral clitics from stems in the present study); white space is word boundary; `¦-> Binding` refers to gold standard/native speaker binding properties. The contents of the dataset are summarized in Table 1.

Table 1. Summary of the dataset.

#	Test target
1-3	Condition C in English
4-6	Condition A in English
7-9	Condition B in English
10-15	Conversations and Binding Condition C in English
16-21	Conversations and Binding Condition A in English
22-27	Conversations and Binding Condition B in English
28-37	Conversations and sequences of pronouns in English
38-43	Conversations and intra-clausal coherence in Finnish
44-46	Condition A in Finnish
47-49	Condition B in Finnish
50-52	Condition C in Finnish
53-54	Null subjects in Finnish (first person)
55-61	Null subjects in Finnish (third person, special properties, finite control)
62-96	Phi-feature restrictions on denotations in English and Finnish, regular pronouns and reflexives
97-100	Above in the context of conversations
101-103	Binding reconstruction for A-chains in Finnish, r-expressions, pronouns, reflexives
104-109	Binding reconstruction for scrambling in Finnish, r-expressions, pronouns and reflexives
110	Binding and control in A-chains in Finnish
111-118	Binding in English nonfinite clauses

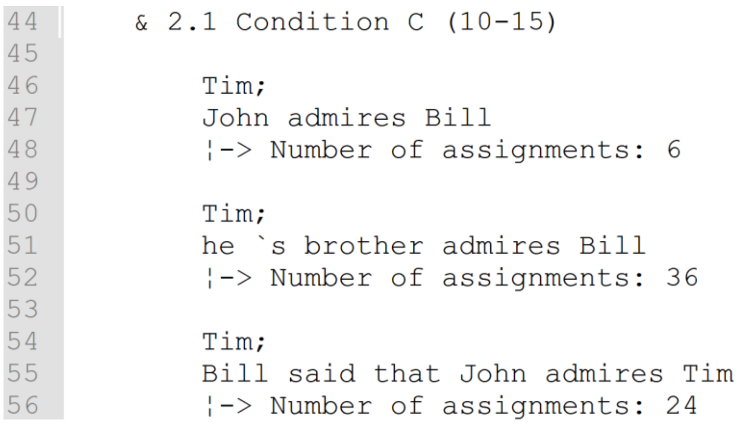
119-132	Binding in Finnish A-infinitive clauses
133-138	DP-internal binding in Finnish and English

Each sentence may be associated with a gold standard/native speaker binding properties by using prefix `¦-> Binding`, as shown in the screenshot from the dataset file below:



```
 9   # Condition C (1-3)
10
11       John admires Bill
12       ¦-> Binding: John[a] admire Bill[b]
13
14       he `s brother admires Bill
15       ¦-> Binding: he[b] brother[a] admire Bill[b,c]
16
17       Bill said that John admires Tim
18       ¦-> Binding: Bill[a] say John[b] admire Tim[c]
```

The model matches the predicted binding properties with these attributes. This information is read in the following way: when two expressions have different index letters, they are necessarily disjoint in denotation (as in line 12); when the indices overlap, they can be coreferential (as in line 15), when the indices are the same, the expressions are necessarily coreferential. Any attribute calculated by the model and reported in the results file can be used in this way. Thus, it is possible to match the number of assignments predicted by the model by using symbol `¦-> Number of assignments: 36` as in

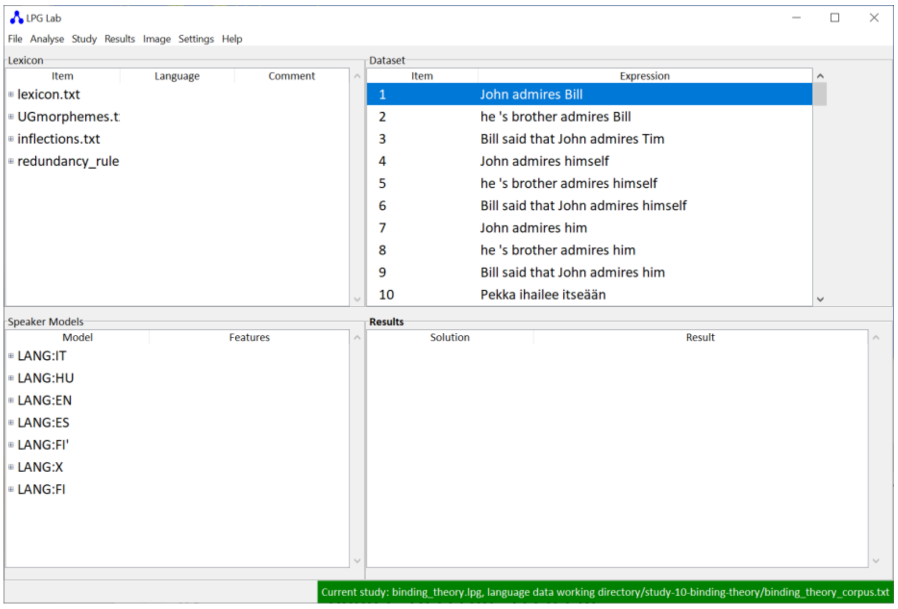


```
& 2.1 Condition C (10-15)

    Tim;
    John admires Bill
    ¦-> Number of assignments: 6

    Tim;
    he `s brother admires Bill
    ¦-> Number of assignments: 36

    Tim;
    Bill said that John admires Tim
    ¦-> Number of assignments: 24
```

This will match the number of assignments predicted by the model with the number provided here.[17]

2.4 Exploring the results by using the graphical user interface

Due to the combinatorial nature of linguistic data and theory, the algorithm produces considerable amount of raw output data. The derivational log file generated in this study contains 85.000 lines of calculations. To mitigate this issue, the results can also be examined by using a graphical user interface. The interface can be launched by writing `python lpparse -app` into the command prompt. It produces the following main window:



Item	Language	Comment
lexicon.txt		
UGmorphemes.t		
inflections.txt		
redundancy_rule		

Item	Expression
1	John admires Bill
2	he 's brother admires Bill
3	Bill said that John admires Tim
4	John admires himself
5	he 's brother admires himself
6	Bill said that John admires himself
7	John admires him
8	he 's brother admires him
9	Bill said that John admires him
10	Pekka ihailee itseään

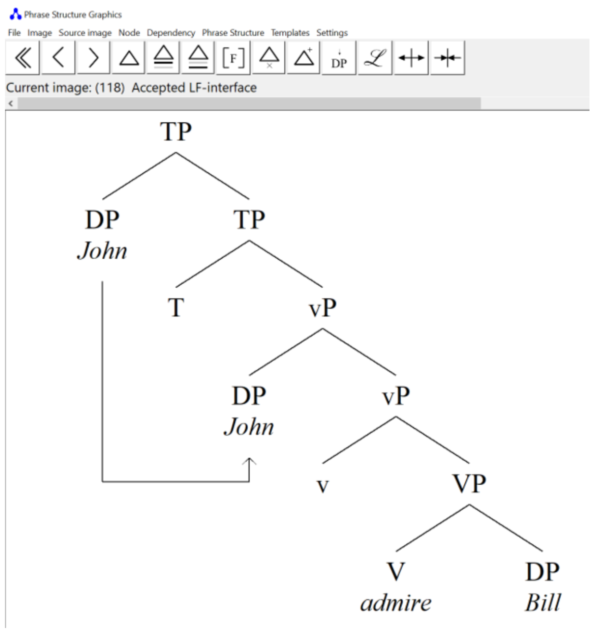
Model	Features
LANG:IT	
LANG:HU	
LANG:EN	
LANG:ES	
LANG:FI'	
LANG:X	
LANG:FI	

Solution	Result

Current study: binding_theory.lpg, language data working directory/study-10-binding-theory/binding_theory_corpus.txt

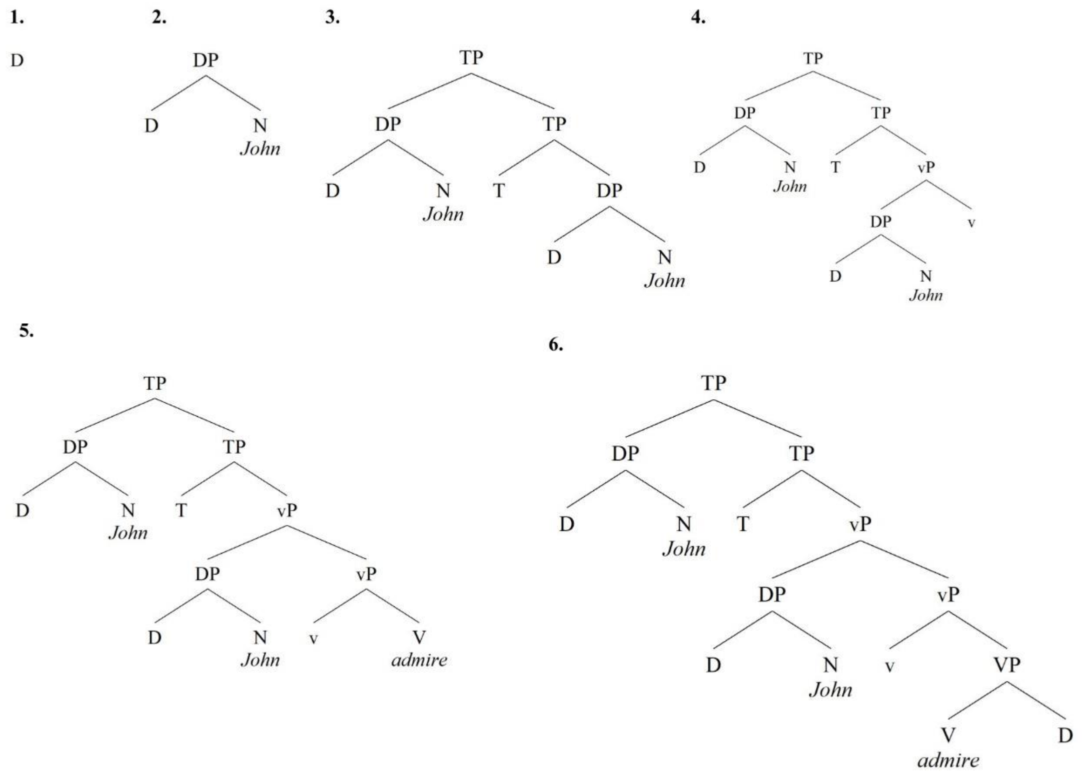
The upper right panel contains the dataset. Double-clicking an item causes the model to try to find a derivation for the selected expression. If a derivation is found, the results are shown in a separate window (the whole study can be executed by selecting `Study > Run Study` from the main menu). For example, by double-clicking *John admires Bill* opens up the following window (the actual contents depend on the settings for the tree drawing algorithm):

[17] The model matches string literals. Care must be taken to ensure that the data format matches with what is produced by the model. If there is a mismatch, the program will judge that the two items diverged and records a prediction error.

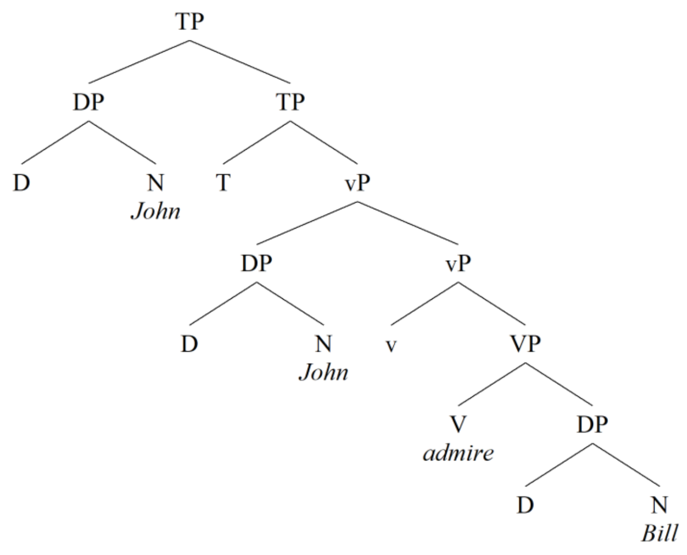

Phrase Structure Graphics
File Image Source image Node Dependency Phrase Structure Templates Settings
TP
DP
John
TP
T
vP
DP
John
vP
v
VP
V
admire
DP
Bill

Current image: (118) Accepted LF-interface

This shows the first acceptable LF-interface representation found for this input expression by the underlying grammar. The user can scroll the derivation backwards and forward by using to examine how the derivation progressed in time. In this case, the derivation contains steps 1–6



with the final representation at LF being



By placing the mouse over any constituent and pressing (or selecting an equivalent item in the main menu) opens up a list of properties associated with that element at the derivational step shown. This interface can make the exploration and verification of the results easier than reading the raw output. There is separate documentation for how to use the graphical user interface to explore the results.[18] The graphical user interface can also be used to draw standard phrase structure trees from scratch.

3 Results

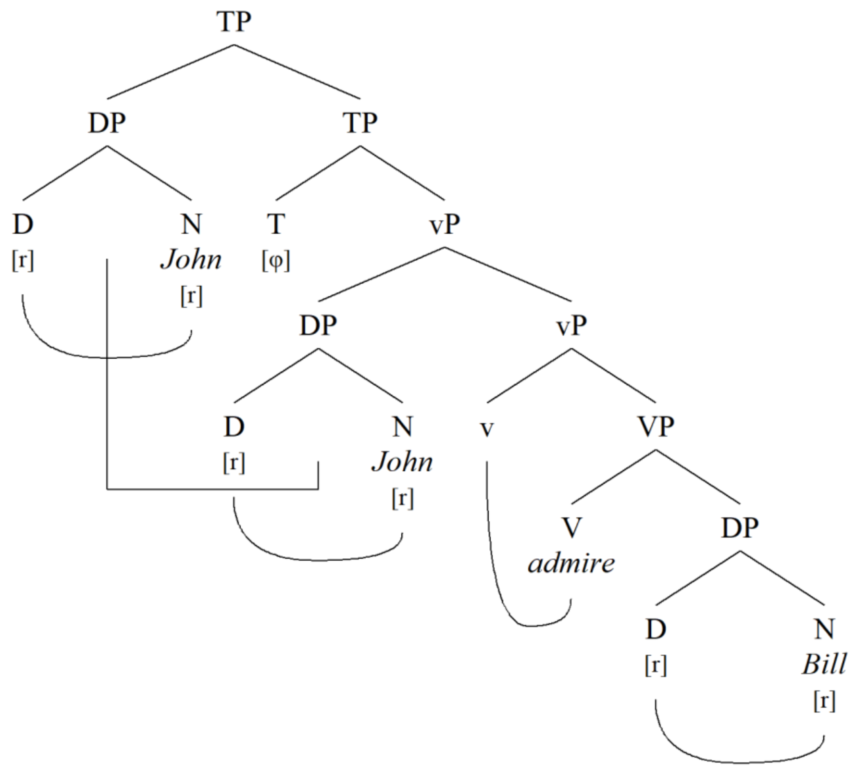
3.1 Baseline model

First we examine the derivation of simple finite clauses by the underlying LPG model without addressing binding. The discussion illustrates the empirical assumptions of the model. We consider the sentence *John admires Bill* (#1) in the dataset. In the phrase structure images presented here (and created by the tree drawing algorithm of the model) syntactic head chains are marked by curved lines, phrasal chains by non-curved lines; symbol [φ] means that the head contains agreement features extracted from the input (overt agreement, including the Finnish

[18]`https://github.com/pajubrat/parser-grammar/blob/master/docs/LPGlab_manual.pdf`

possessive suffix); [r] (r-expression, Condition C), [rflx] (reflexive, Condition A) and [pron] (regular pronoun, Condition B) are the three lexical features which control binding as explained in the main article. The input expression #1 creates the following LF-interface representation:

(2)

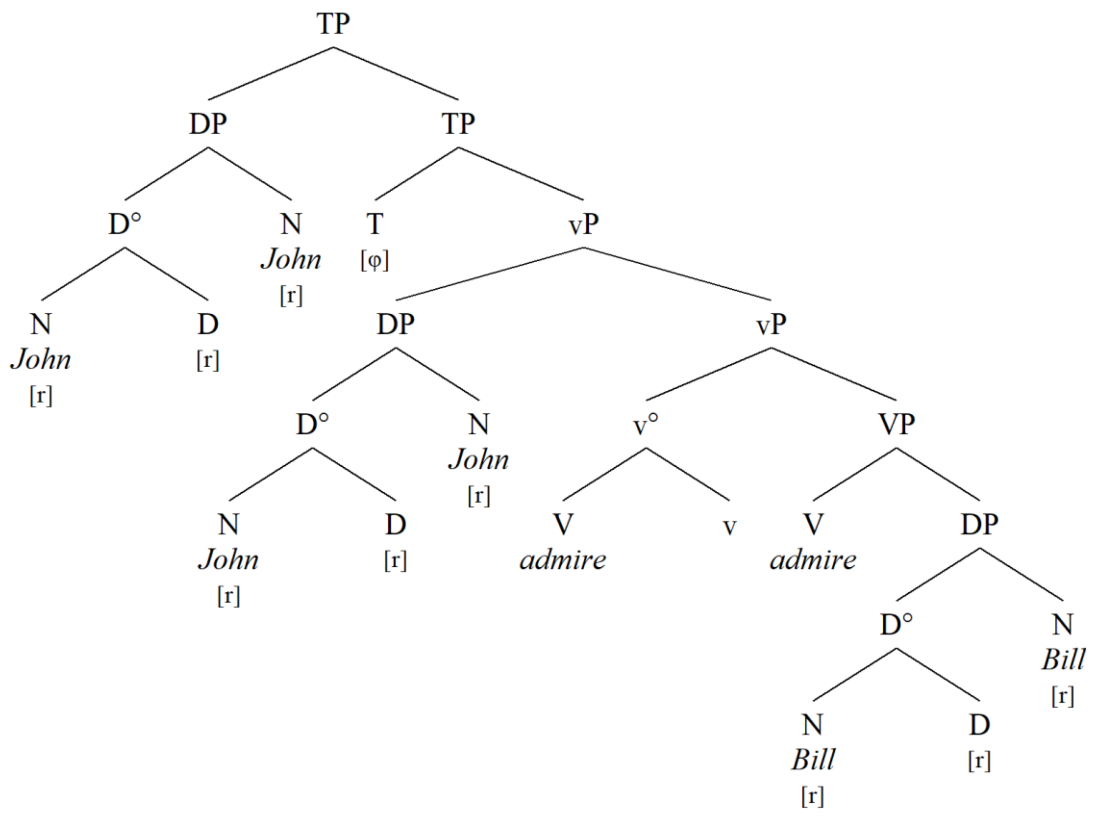


For the underlining phrase structure formalism, see Brattico 2019: §7.1. Monomorphemic full arguments such as *John* and *Bill* are created by merging D and N and combining the two heads by means of a syntactic head chain, by the PF-LF mapping (Brattico, 2022). Trivial transformations are not visible in the images that follow.[19] The chain (SpecTP, SpecvP) is a "standard" A-chain (as far as we can call them standard in the present framework, Brattico 2019: §4.8.5).[20] Explicit representations created by head chains are not visible in these images

[19] What is presented in the images can be controlled by the study configuration file, as explained in the software documentation.
[20] Notice that tense T is not combined with the complex head v + V (for the notion of complex head, see below) by a syntactic head chain; this amalgamation operation is performed during the postsyntactic SM-PF mapping (hence in "phonology") in the current model for reasons that are irrelevant here; the binding analysis would be the same if it were performed during PF-LF mapping.

but can be turned on from the study configuration file. This creates the following, more explicit representation:

(3)



These complex heads exist at the LF-interface, but are not usually visualized, as shown in (2), since the same information can be read from the head chains. Symbol X° highlights the fact that the complex item is regarded as a primitive constituent from the point of view of phrasal syntax.

As described in Section 1, the model derives LF-interface representations by applying cyclic and noncyclic operations (Brattico 2019: §4.8.3, 7.2) to the linear input expressions organized as a left-to-right sequence. The whole sequence is recorded into the derivational log file, but it can be inspected by using the graphical interface tool (Section 2.4, see steps 1-6). In the GUI interface the user can click buttons « ‹ › to scroll the derivation both backwards and forward in time or select the corresponding menu items from the main menu (e.g., `Source image > First LF-interface`). We look at some aspects of the derivation at the end of this section.

If no legitimate derivation is found, the input expression is judged *ungrammatical*. If a legitimate derivation is found, as above, it is judged *grammatical*. If the expression can be mapped into several legitimate derivations, then it is grammatical and *ambiguous*, with each solution associated with particular syntactic and semantic properties as defined by its complete derivation. The model searches through all possible legitimate derivations for the input expression as a linear sequence of items (morphemes) and thus implements the *derivational search function* for the LPG recognition grammar (see Brattico, 2024).

The grammatical information contained in each phonological word in the input sentence is retrieved from the *lexicon*. Thus, the lexicon interprets *John* as a morphological chunk containing D and N, which is inserted into the PF-interface representation as a complex head D(N) and reconstructed into a DP by a head chain (thus, the mappings are *John* ~ D#N ~ D(N) ~ [$_{\text{DP}}$ D N]). The feature contents of D and N are synthesized from their lexical features and morphosyntactic features extracted from the input if any (e.g., *admire* ~ *admires*)(4).

(4)	John	admires	Mary	Expression (SM)
	D#N	V#v#T#3sg	D#N	Lexicon, morphological decompositions
	[D(N)	[T$_{[3sg]}$(v, V)	D(N)]	PF-interface
	[$_{\text{TP}}$ [$_{\text{DP}}$ D N] [$_{\text{TP}}$ T [$_{\text{vP}}$ v [$_{\text{VP}}$ V [$_{\text{DP}}$ D N]]]]]			LF-interface

The lexical features posited to account for binding are generated in the same way: they are part of the referential expressions such as proper pronouns, reflexives and r-expressions. The model does not have a separate morphological parser. Morphological decompositions can be provided either directly in the lexicon as memorized morphological chunks or in the input strings by using morpheme boundary symbols, the latter which in effect simulates morphological parsing (Brattico 2019: §4.9.1).

Let us consider the derivation of the sentence *John admires Bill* as it is visible in the derivational log file. The process begins by analyzing the first word *John* which retrieves a morphological chunk from the lexicon:

```
     Next morpheme /John/ retrieves  (1) morphological chunk [JOHN.D]
```

Symbol `JOHN` stands for the noun head N containing the lexicon content of the item, `D` is the standard D-element retrieved as a separate head. Both are retrieved from the lexicon by `/John/`, present in the input. The dot notation `JOHN.D` means that both elements were inside the same phonological word *John*; when the two elements are separated by a morpheme boundary, symbol # is used so that *Pekka-a* 'Pekka-PAR' can be represented as PEKKA#par and not PEKKA.par. These two items are then merged together, forming a complex head D(N) or (D N)^0:

```
⊢    Merge(D, John)
     = D(John)
```

Thus, the word *John* at SM is mapped into complex head D(N)^0 at PF-interface level. This is followed by two cyclic transformations

```
     Cyclic reconstruction:

     Feature inheritance(D(John))
      = D(John)
     IHM(John)
      = [D John]
```

Where `Feature inheritance` captures D-N concord (Brattico 2019: §7.2.3.2) and `IHM` stands for Internal Head Merge and creates cyclically the head chain D(N)^0 ~ $[_{DP}$ D N] (Brattico 2019: §4.8.6, 7.2.3.4, see also Brattico 2022). The verb *admires* decomposes (via morphological chunking, Brattico 2019: §4.9.1) into a sequence of heads T, v and V, the first of which creates

```
⊢    Merge([D John], T)
     = [[D John] T]
```

and is followed by three cyclic transformations

```
     Cyclic reconstruction:

     Feature inheritance(T)
      = [[D John] T]
     A-chain([D John])
      = [[D John]:2 [T __:2]]
     Agree(T) values [NUM:SG][PER:3] from goal __:2.
      = [[D John]:2 [T __:2]]
```

which implement the standard local A-chain (59-60)(Brattico 2019: §4.8.5) and agreement (Agree) between T and the grammatical subject (61-62)(Brattico 2019: §4.8.9).[21] Cyclicity explains why the operations are local: no further structure is present when the operations are triggered (Brattico 2019: §4.8.3). The derivation continues in the same way until we reach the end, at which point the PF-interface representation

```
122      = [[D John]:2 [T [__:2 [v [admire [D Bill]]]]]]
```

is subjected to noncyclic Agree transformations

```
134      Agree(admire) values [NUM:SG][DET:DEF][PER:3] from goal [D Bill].
135       = [[D John]:2 [T [__:2 [v [admire [D Bill]]]]]]
136      Agree(v) values [NUM:SG][DET:DEF][PER:3] from goal [D Bill].
137       = [[D John]:2 [T [__:2 [v [admire [D Bill]]]]]]
138      Agree(T) values nothing from goal __:2.
139       = [[D John]:2 [T [__:2 [v [admire [D Bill]]]]]]
```

resulting in the LF-interface representation

```
143      = LF-interface [[D John]:2 [T [__:2 [v [admire [D Bill]]]]]]
```

This representation feeds semantic interpretation reported next in the derivational log file:

[21] The operation Agree in LPG is discussed in detail in a currently unpublished manuscript Brattico, P. "Across the board agreement in Finnish."

```
146 ----Semantic interpretation-----------------------------
147
148     Object projections:
149         Project object (1, QND) for [D John]
150         Project object (1, GLOBAL) for [D John]
151         Project object (2, QND) for [D Bill]
152         Project object (2, GLOBAL) for [D Bill]
153         Project T-event (1, PRE)
154         Project object (3, GLOBAL) for T-Event
155         Project N-concept (2, PRE)
156         Project object (4, GLOBAL) for Concept John
157         Project V-concept (3, PRE)
158         Project object (5, GLOBAL) for Concept admire
159         Project N-concept (4, PRE)
160         Project object (6, GLOBAL) for Concept Bill
161     Argument for T°: [D John], indexed to [D John]
162     Argument for v°: [D Bill], indexed to [D Bill]
163     Argument for admire°: [D Bill], indexed to [D Bill]
164     Denotations:
165         [D John]~['1', '2']
166         [D Bill]~['1', '2']
167     Assignments:
168         Assignment [D John] ~ 1, [D Bill] ~ 1 -Illegitin
169         Assignment [D John] ~ 1, [D Bill] ~ 2 +
170         Assignment [D John] ~ 2, [D Bill] ~ 1 +
171         Assignment [D John] ~ 2, [D Bill] ~ 2 -Illegitin
172         Summary:  John[a] admire Bill[b]
```

Lines 148-160 project semantic objects into global discourse inventory (GLOBAL) and into narrow semantics (PRE, QND); lines 161-163 find arguments for predicates (ignored in this study); lines 164-166 report possible denotations; lines (167-172) show the assignment calculations. This information is followed by a complete list of lexical items (heads) and their features plus a complete listing of what is in the semantic inventory when the derivation was completed (Brattico 2019: §4.10).

Notice that in most cases when new heads are merged into a partial PF-interface representation held in the syntactic working memory there are several possible merge solutions depending on the attachment site. All these possibilities are registered and evaluated cyclically and inserted into a recursive stack which is revisited by backtracking. This information is recorded under `Ranking` in the derivational log file. For example, when v is merged to the PF-interface representation [[$_{DP}$ D N]$_1$ [T __$_1$]], we have two options:

```
    Ranking:

        1.__:2 + v (1)
        2.John + v (-398)
```

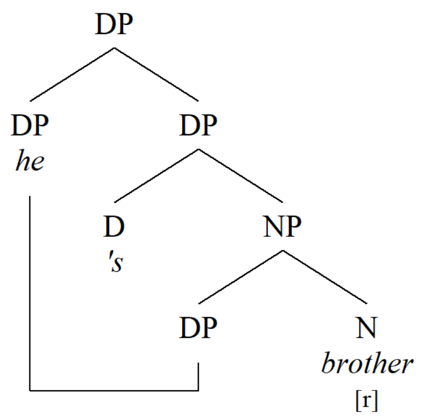
which correspond to [[$_{\text{DP}}$ D N]$_1$ [T [~~[$_{\text{DP}}$ D N]~~ v]]] (1) and [[[[$_{\text{DP}}$ D N]$_1$ [T [$_{\text{DP}}$ ~~D [N~~ v]]]] (2). The latter has a small ranking score due to being implausible.[22] All solutions that are possible in principle are treated in the same way. They must be treated in this way so that the model can find ambiguities and judge ungrammatical expressions correctly as ungrammatical.

3.2 DP-internal syntax in English and Finnish

3.2.1 DP-internal syntax of English

There are currently no systematic computational generative grammar studies of DP-internal syntax of English based on the LPG analysis and which we could rely on here. It was assumed that *`s* represents D (not particularly controversial) and that the possessor argument is located at its specifier position as indicated by the surface order and then reconstructed to SpecNP which is assumed to be a noun-internal thematic position (more debatable). Thus, expressions such as *his brother* (present in the dataset) are analyzed as (5).

(5)



In short, D behaves like a standard non-thematic EPP head hosting formal phrasal specifiers while N is a thematic head. This behavior can be controlled in the lexicon, and is easy to change if we want to eliminate reconstruction and have the possessor at SpecDP at LF. The fact that the possessor is reconstructed to SpecNP, or that the clitic is represented as D, are largely inessential for the purposes of binding. The binding model only requires that the pronoun be

[22] Solution [$_{\text{vP}}$ [$_{\text{TP}}$ DP$_1$ [$_{\text{TP}}$ T __$_1$]] v] is filtered out as impossible, is therefore not visible in the ranking, and is not evaluated by backtracking. The fact that the filtering operation is not recorded in the derivational log file is an accidental feature of the implementation.

inside the DP and is as such not visible in the syntactic working memory from the direct objects of the same clause, ruling out assignments such as **his$_a$ brother admires himself$_a$*. For the formalization of the notion of syntactic working memory, see Brattico 2019: §7.1.3.

3.2.2 DP-internal syntax of Finnish

Finnish does not have obligatory articles, therefore the input sequence may contain bare nouns which should nevertheless function as full phrasal arguments. This was captured by assuming that bare nouns are analysed morphologically as φ + N constituents, as provided in the lexicon, projecting [$_{\varphi P}$ φ N], where φ is a minimal head able to create a referential argument (thus it contains [Ref]). As a consequence, any prenominal heads or modifiers will get generated above φN (e.g., *tämä sukka* 'this sock' = [$_{DP}$ D [$_{\varphi P}$ φ N]]). Possessors are generated to SpecφP and A-reconstruct to SpecNP for thematic reason, SpecNP being a thematic position inside the noun phrase. Detailed computational generative grammar studies of Finnish DPs do not exist at the present writing, but see Brattico & Leinonen (2009) for an analysis compatible with the system proposed here.

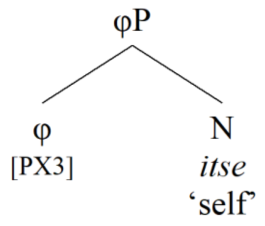
3.2.3 Analysis of the Finnish possessive suffix

Finnish reflexive pronouns are created by combining the stem *itse* 'self' with an optional third person possessive suffix (6). Notice that the gender feature of the English translations is absent from the Finnish examples.

(6) a. Pekka ihailee itse-ä.
Pekka admires self-PAR
'Pekka admires himself.'
b. Pekka ihailee itse-ä-*än*.
Pekka admires self-PAR-PX3
'Pekka admires himself.'

The analysis of the Finnish possessive suffix in general is contested. The baseline model analyses it as an inflectional infinitival agreement suffix (thus, not its own head or phrase); for the processing of inflectional information in LPG see Brattico 2019: §4.9.4.[23] Because the stem *itse-ä* 'self-PAR' is analysed as a full φP (Section 3.2.2), the phi-features of the possessive suffix (here third person singular) will be inserted inside φ:

(7)



Once they are inside φ, they will restrict the range of denotations of the whole φP such that the denotations must be compatible with the acquired phi-features. For example, *itse-ä-ni* 'self-PAR-PX1SG' can only refer to first person singular persons. The reflexive without the possessive has fewer restrictions:

(8)	a.	Minä$_a$	ihailen	itse-ä$_{a,*b}$.
		I	admire	self-PAR
		'I admire myself.'		
	b.	Sinä$_a$	ihailet	itse-ä$_{a,*b}$.
		You	admire	self-PAR
		'You admire yourself.'		
	c.	*Minä	ihailen	itse-ä-*si*
		I	admire	self-PAR-PX2SG

For the binding analysis presented in this study it does not matter how the possessive suffix is analysed as long as its phi-features enter into the calculations of denotations to account for the restrictions such as (8).[24]

[23] This hypothesis is discussed in an unpublished paper Brattico, P. "Across the board agreement in Finnish."
[24] For example, the possessive suffix could project its own functional head above φP.

3.3 Binding conditions A-C in English (#1-9)

The hypothesis was checked against basic English binding configurations (#1-9) which test the behavior of r-expressions (proper names), reflexives and regular pronouns against coreferentiality with a clause-mate subject (9)a, an argument embedded inside the clause-mate subject (9)b, and with a subject of a superordinate clause (9)c.

(9) (#1-9)

a. *John*$_{a}$ admires Bill$_{*a,b}$/himself$_{a,*b}$/him$_{*a,b}$.

b. [*His*$_{a}$ brother]$_{b}$ admires Bill$_{a,*b,c}$/himself$_{*a,b,*c}$/him$_{a,*b,c}$.

c. *Bill*$_{a}$ said that John$_{b}$ admires Bill$_{a,*b,c}$/himself$_{*a,b,*c}$/him$_{a,*b,c}$.

Sentence *John admires Bill* (#1, lines 9-335 in the derivational log file) generates two persons [1] and [2] into the global discourse inventory and then provides two possible assignments (from the results file):

```
        8.Assignments:

            [D John] ~ 1, [D Bill] ~ 2
            [D John] ~ 2, [D Bill] ~ 1
```

in which *John* and *Bill* refer to the two persons, respectively, but such that they cannot denote the same person (see further below). Notice that it is possible of course for *Bill* to denote John and *John* Bill. The two objects in the global discourse inventory are defined as follows:

```
        Object [D John](1) ['§Thing'] in GLOBAL
            Class: human
            Constituent: D
            Gender: m
            Number: singular
            Person: third
            Reference: [D John]
            Referring constituent: D
            Semantic space: GLOBAL
            Semantic type: §Thing
        Object [D Bill](2) ['§Thing'] in GLOBAL
            Class: human
            Constituent: D
            Gender: m
            Number: singular
            Person: third
            Reference: [D Bill]
            Referring constituent: D
            Semantic space: GLOBAL
            Semantic type: §Thing
```

These "file-cards" are based on Python dictionary data structures and as such may contain almost any type of information.[25] The denotations and assignments cited above refer to the numbers (here [1] and [2]). These semantic objects or attribute lists are created on the basis of the lexical features of D.[26] The evaluation of possible assignments based on binding theory are visible in the derivational log file:

```
Assignments:
    Assignment [D John] ~ 1, [D Bill] ~ 1 -
    Assignment [D John] ~ 1, [D Bill] ~ 2 +
    Assignment [D John] ~ 2, [D Bill] ~ 1 +
    Assignment [D John] ~ 2, [D Bill] ~ 2 -
    Summary:  John[a] admire Bill[b]
```

+ at the end of the line indicates that the assignment was accepted, – that it was rejected. The last line, which is compared with the binding configuration provided in the input dataset, if any, summarizes the results. Pronouns (#7) produce the same results. Use of the reflexive pronoun (#4) changes the assignment into

```
8.Assignments:

    [D John] ~ 1, [D self] ~ 1
    [D John] ~ 2, [D self] ~ 2
```

which corresponds to the reading *John*$_a$ *admires himself*$_{a,*b}$. The working memory mechanics described in the main document are not transparent in the raw output data, all we see are the consequences and possible violations recorder into the derivational log file (see above).

Locality was tested by embedding the test sentences inside another main clause. The binding dependencies calculated by the model are shown in (10).

[25] Including nested file cards and pointers to other file cards. It is assumed that relations between semantic objects are represented by pointers or links between the file-cards. For example, events are semantic objects which contain pointers to participants (agents, patients); 'thing' objects could contain pointers to properties, relations and events which are predicated of them, the latter which are unsaturated objects that must be predicted of something, and so on. This representation format is obviously too powerful, but it cannot be constrained without evaluation against appropriate datasets.
[26] They were defined by using standard Python dictionary data-structures.

(10) a. Bill$_a$ said that John$_b$ admires Tim$_c$. (#3)

b. Bill$_a$ said that John$_b$ admires himself$_b$. (#6)

c. Bill$_a$ said that John$_b$ admires him$_{a,c}$. (#9)

Let us consider the denotation and assignment calculations for (10)a. Three persons are projected into the global discourse inventory, and each name can refer to each such person:

```
    Denotations:
        [D Bill]~['1', '2', '3']
        [D John]~['1', '2', '3']
        [D Tim]~['1', '2', '3']
```

The assignment calculations are as follows:

```
    Assignments:
        Assignment [D Bill] ~ 1, [D John] ~ 1, [D Tim] ~ 1 -
        Assignment [D Bill] ~ 1, [D John] ~ 1, [D Tim] ~ 2 -
        Assignment [D Bill] ~ 1, [D John] ~ 1, [D Tim] ~ 3 -
        Assignment [D Bill] ~ 1, [D John] ~ 2, [D Tim] ~ 1 -
        Assignment [D Bill] ~ 1, [D John] ~ 2, [D Tim] ~ 2 -
        Assignment [D Bill] ~ 1, [D John] ~ 2, [D Tim] ~ 3 +
        Assignment [D Bill] ~ 1, [D John] ~ 3, [D Tim] ~ 1 -
        Assignment [D Bill] ~ 1, [D John] ~ 3, [D Tim] ~ 2 +
        Assignment [D Bill] ~ 1, [D John] ~ 3, [D Tim] ~ 3 -
        Assignment [D Bill] ~ 2, [D John] ~ 1, [D Tim] ~ 1 -
        Assignment [D Bill] ~ 2, [D John] ~ 1, [D Tim] ~ 2 -
        Assignment [D Bill] ~ 2, [D John] ~ 1, [D Tim] ~ 3 +
        Assignment [D Bill] ~ 2, [D John] ~ 2, [D Tim] ~ 1 -
        Assignment [D Bill] ~ 2, [D John] ~ 2, [D Tim] ~ 2 -
        Assignment [D Bill] ~ 2, [D John] ~ 2, [D Tim] ~ 3 -
        Assignment [D Bill] ~ 2, [D John] ~ 3, [D Tim] ~ 1 +
        Assignment [D Bill] ~ 2, [D John] ~ 3, [D Tim] ~ 2 -
        Assignment [D Bill] ~ 2, [D John] ~ 3, [D Tim] ~ 3 -
        Assignment [D Bill] ~ 3, [D John] ~ 1, [D Tim] ~ 1 -
        Assignment [D Bill] ~ 3, [D John] ~ 1, [D Tim] ~ 2 +
        Assignment [D Bill] ~ 3, [D John] ~ 1, [D Tim] ~ 3 -
        Assignment [D Bill] ~ 3, [D John] ~ 2, [D Tim] ~ 1 +
        Assignment [D Bill] ~ 3, [D John] ~ 2, [D Tim] ~ 2 -
        Assignment [D Bill] ~ 3, [D John] ~ 2, [D Tim] ~ 3 -
        Assignment [D Bill] ~ 3, [D John] ~ 3, [D Tim] ~ 1 -
        Assignment [D Bill] ~ 3, [D John] ~ 3, [D Tim] ~ 2 -
        Assignment [D Bill] ~ 3, [D John] ~ 3, [D Tim] ~ 3 -
        Summary:  Bill[a] say John[b] admire Tim[c]
```

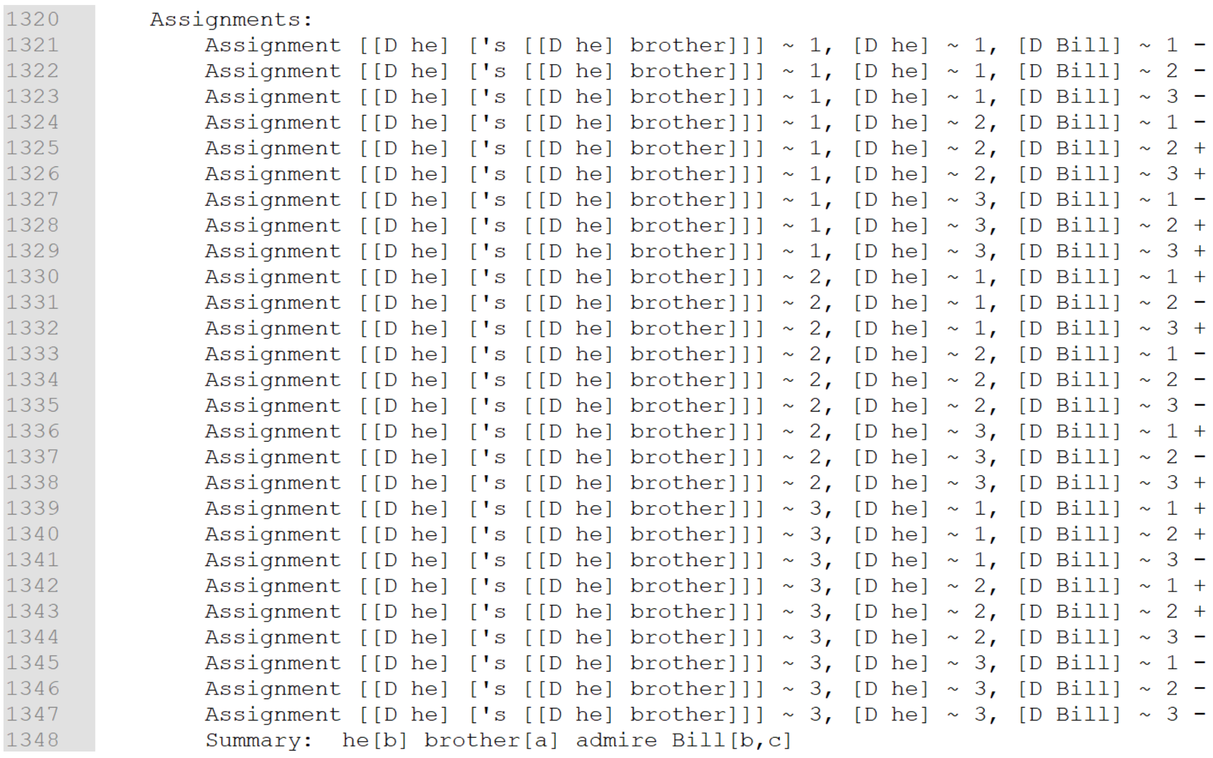
Every assignment in which any of the referential expressions are coreferential is ruled out due to an illegitimate binder, since r-expressions must be free (new in the complete syntactic working memory). This leaves six assignments in which each name is paired with one of the persons. The locality domains for the reflexive and regular pronoun in (10)b-c are restricted to the embedded clause due to the intervening feature [Ref] as shown in (11):

(11) Bill$_{a}$ said that [$_{DP}$ D$_{[Ref]}$ …]$_{b}$ admires himself$_{*a, b}$/him$_{a,*b}$.

The three remaining sentences (#2, 5, 8) verify that the working memory mechanism ignores referential expressions that are "too deep" inside constituents in the working memory (12).

(12) [His$_{a}$ brother]$_{b}$ admires John$_{a,*b,c}$/himself$_{*a,b, *c}$/him$_{a,*b,c}$.

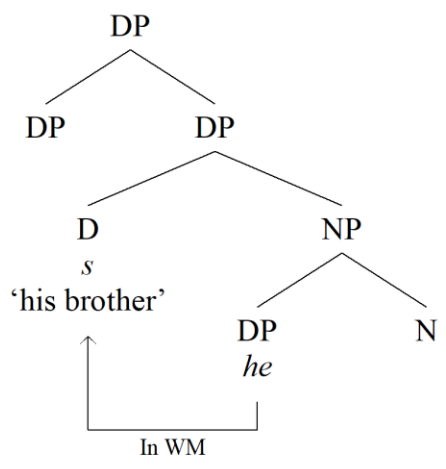
It is uncontroversial that the possessive pronoun *his* is inside the subject DP, while the exact analysis is debatable. The LPG analysis regards the structure as [$_{DP}$ *he*$_{1}$ *\`s* [$_{NP}$ \_\_$_{1}$ *brother*]] where *\`s* is the phonological exponent of D, while the thematic role is assigned by the reconstructed SpecNP position. See Section 3.2. The assignment calculations for #2 are as follows



```
    Assignments:
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 1, [D Bill] ~ 1 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 1, [D Bill] ~ 2 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 1, [D Bill] ~ 3 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 2, [D Bill] ~ 1 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 2, [D Bill] ~ 2 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 2, [D Bill] ~ 3 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 3, [D Bill] ~ 1 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 3, [D Bill] ~ 2 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 1, [D he] ~ 3, [D Bill] ~ 3 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 1, [D Bill] ~ 1 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 1, [D Bill] ~ 2 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 1, [D Bill] ~ 3 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 2, [D Bill] ~ 1 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 2, [D Bill] ~ 2 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 2, [D Bill] ~ 3 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 3, [D Bill] ~ 1 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 3, [D Bill] ~ 2 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 2, [D he] ~ 3, [D Bill] ~ 3 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 1, [D Bill] ~ 1 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 1, [D Bill] ~ 2 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 1, [D Bill] ~ 3 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 2, [D Bill] ~ 1 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 2, [D Bill] ~ 2 +
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 2, [D Bill] ~ 3 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 3, [D Bill] ~ 1 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 3, [D Bill] ~ 2 -
        Assignment [[D he] ['s [[D he] brother]]] ~ 3, [D he] ~ 3, [D Bill] ~ 3 -
        Summary:  he[b] brother[a] admire Bill[b,c]
```

where we have 12 legitimate assignments, those where *his brother* is always disjoint from the two other referential expressions, the latter two which then denote either the same or different persons. Notice that readings in which *he* refers to the same person as *his brother* are ruled out, since *his brother* (or rather its D-head) is inside the syntactic memory calculated from the reconstructed thematic base position of the possessive pronoun (13).

(13)



3.4 Binding conditions A-C in Finnish (#44-52)

The same nine construction types (#44-52) were used to test the operation of the binding conditions in Finnish (14). Notice that Finnish pronouns are not specified for gender.

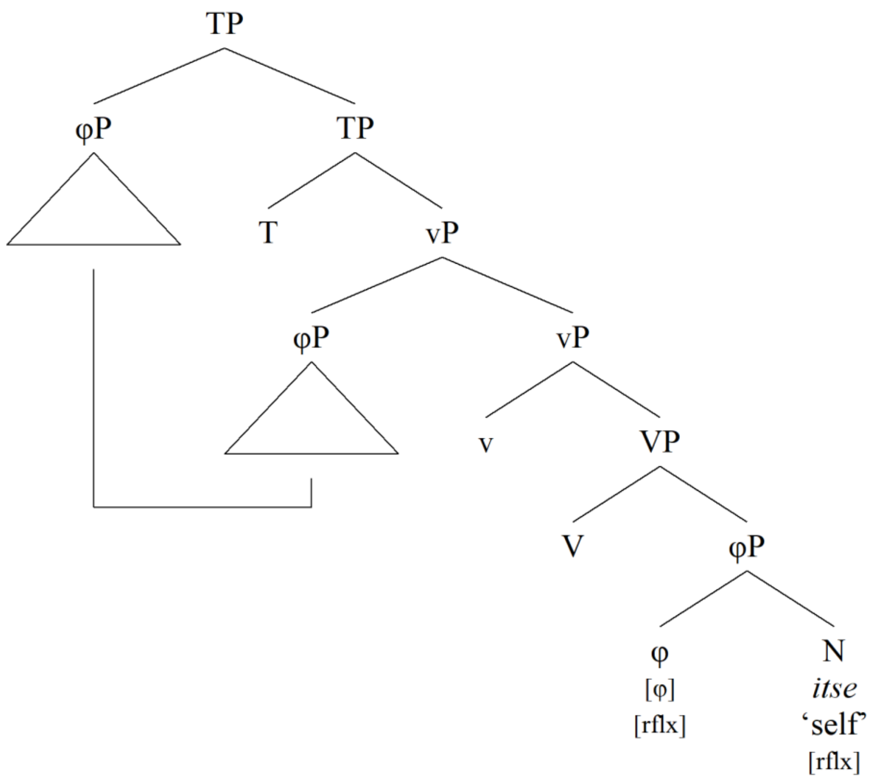
(14)	a.	Pekka$_{a}$	ihailee	itse-ä-än$_{a,*b}$/	hän-tä$_{*a,\ b}$/	Merja-a$_{*a,b}$.	
		Pekka	admires	self-PAR-PX3	he-PAR	Merja-PAR	
		'Pekka admires himself/		him/	Merja.'		
	b.	[Peka-n$_{a}$	sisko]$_{b}$	ihailee	itse-ä-än$_{*a,b}$/	hän-tä$_{a,*b}$/	Merja-a$_{a,*b,c}$.[27]
		Pekka-GEN	sister	admires	self-PAR-PX3	he/she-PAR	Merja-PAR
		'Pekka's sister admires herself/ him/Merja.'					
	c.	Pekka$_{a}$	sanoi	että Merja$_{b}$	ihailee	itse-ä-än$_{*a,b}$/	hän-tä$_{a,*b}$/	Merja-a$_{*a,\ *b,c}$.
		Pekka	said	that Merja	admires	self-PAR-PX3	he/she-PAR	Merja-PAR
		'Pekka said that Merja admires herself.'					

The calculations parallel those of the English examples examined above, with two differences. First, Finnish being a language without obligatory articles all bare noun phrases were analyzed as φPs (Section 3.2.2), φ = a minimal referential head. Second, the reflexive is composed from the reflexive stem *itse* 'self' and the possessive suffix. The possessive suffix was analyzed as an ordinary agreement marker which restricts the denotation of the hosting phrase as described

[27] The coreference reading between Pekka and *Merja* is not possible due to the gender feature mismatch, possible otherwise.

in Section 3.2.3. Given these assumptions the model calculates (15) for (14)a with the reflexive pronoun:

(15)

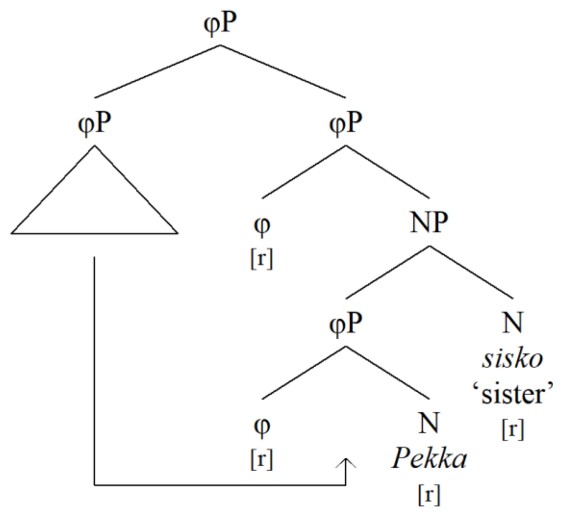


The phi-set marked as [φ] for the φ-head originates from the possessive suffix. The assignments are calculated correctly:

```
        8.Assignments:

            [φ Pekka] ~ 1, [φ itse] ~ 1
            [φ Pekka] ~ 2, [φ itse] ~ 2
```

(14)b with the complex subject *Peka-n sisko* 'Pekka-GEN sister' are analyzed analogously to the English equivalents but with the exception that the noun phrase is φP. The possessor is reconstructed from SpecφP into SpecNP where it checks the genitive case (16).

(16)



The binding possibilities are calculated correctly since the possessor is inside the subject and not visible in the syntactic working memory from the direct object position of the same clause. Here are the legitimate assignments generated for the reflexive sentence:

```
        Assignments:
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 1, [φ itse] ~ 1 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 1, [φ itse] ~ 2 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 1, [φ itse] ~ 3 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 2, [φ itse] ~ 1 +
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 2, [φ itse] ~ 2 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 2, [φ itse] ~ 3 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 3, [φ itse] ~ 1 +
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 3, [φ itse] ~ 2 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 1, [φ Pekka] ~ 3, [φ itse] ~ 3 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 1, [φ itse] ~ 1 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 1, [φ itse] ~ 2 +
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 1, [φ itse] ~ 3 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 2, [φ itse] ~ 1 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 2, [φ itse] ~ 2 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 2, [φ itse] ~ 3 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 3, [φ itse] ~ 1 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 3, [φ itse] ~ 2 +
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 2, [φ Pekka] ~ 3, [φ itse] ~ 3 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 1, [φ itse] ~ 1 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 1, [φ itse] ~ 2 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 1, [φ itse] ~ 3 +
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 2, [φ itse] ~ 1 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 2, [φ itse] ~ 2 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 2, [φ itse] ~ 3 +
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 3, [φ itse] ~ 1 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 3, [φ itse] ~ 2 -
            Assignment [[φ Pekka] [φ [[φ Pekka] sisko]]] ~ 3, [φ Pekka] ~ 3, [φ itse] ~ 3 -
            Summary:  Pekka[b] sisko[a] ihaile- itse[a]
```

Pekka is a male first name whereas *Merja* is a female first name, which restrict possible denotations by ruling out coreference dependencies with mismatching gender features.

3.5 Conversations (#10-43)

Conversations were implemented by allowing several sentences to share the global discourse inventory. In most cases this generates more assignments that what would be possible if the sentences were considered in isolation. In most cases the testing was done by adding one additional person to the context. Consider for example a situation in which the context has mentioned one additional person *Tim*, and we consider the target sentence *John admires Bill* (#10-11). The model will create a semantic object [1] for *Tim*, and then when processing the target sentence that object will be taken into account when creating the set of possible assignments:

```
694          8.Assignments:
695
696              [D John] ~ 1, [D Bill] ~ 3
697              [D John] ~ 1, [D Bill] ~ 4
698              [D John] ~ 3, [D Bill] ~ 1
699              [D John] ~ 3, [D Bill] ~ 4
700              [D John] ~ 4, [D Bill] ~ 1
701              [D John] ~ 4, [D Bill] ~ 3
```

The three persons in the global discourse inventory are [1], [3] and [4], and as shown here each name can denoted each person since there are no phi-feature conflicts (index [2] was reserved for a conceptual object that can be ignored here). If any of these persons were denoted by a female name such as *Mary*, fewer assignments would emerge. When it comes to the binding dependencies in the target sentence, they are *John$_a$ admires Bill$_b$* as the presence of the one extra person in the context does not affect the fact that both proper names must be disjoint in reference.[28] What changes is the list of assignments which now allow both *John* and *Bill* to denote Tim as long as they do not both denote him (lines 696, 697, 698, 700). The three tests for conditions A-C discussed in the two previous sections were repeated in English by using one additional contextual person. The number of assignment increases in most cases (#10-27).[29] For example, *...Tim...Bill said that John admires him* creates 36 possible assignments, for example one in which *Bill* and *him* denote Tim. Conversations were also tested by using a sequence of pronouns such as *...Tim...he admires himself* (#28-37). This creates three possible

[28] Notice that the binding notation does not take objects projected outside of the target sentence into account, i.e. there is no separate index for Tim. Binding is a sentence-internal property.
[29] Notice that there are more sentences since the context was provided by an additional sentence.

assignments depending on the interpretation of *he*: 'Tim admires himself', 'he$_a$(not Tim) admires himself' and 'he$_b$(not Tim) admires himself' where the last sentence is generated (redundantly here) because *himself* is allowed to project its own person into the global discourse inventory,[30] so there are again three persons in the discourse inventory. Examples such as this show that we cannot model binding without taking assignments into account: the denotation of *himself* is always the same as the denotation of *he*, while the latter changes from assignment to assignment. Phi-feature mismatches were also tested in the conversational contexts (#97-100), with sentences such as *...Tim...she admires herself* generating only two assignments for the target sentence due to the gender mismatch between *Tim* and *she*/*herself*.

3.6 Phi-feature restrictions (#62-100)

The model restricts denotations of referential expressions by relying on phi-features. Number, person, human and gender were used as the test features. These restrictions are implemented by a two-step process in which the phi-features are first used to generate semantic attributes for the semantic objects in the global discourse inventory, which are then used to restrict assignments. A pronoun *he*, for example, will generate an object into the global discourse inventory with feature 'gender:male', which blocks assignments in which expressions such as *she* or *herself* denote that object. The operation of these restrictions were tested by #62-100 which compares the amount of assignments for expressions such as *John admires him* (2 assignments) with *John admires her* (1 assignment). Finnish pronouns do not mark gender and provide different results, which are also tested for comparison (#66-69). When the phi-features of a reflexive mismatch with the sole antecedent, the model reports zero assignments (=no interpretation) instead of ungrammaticality (#80-96). This is based on convenience alone; we could judge expressions without assignments ungrammatical. Phi-feature mismatches restrict also the assignments generated in conversations (#97-100).

These mechanisms presuppose that there is a mechanism which converts formal phi-features into semantic attributes. We can think of that mechanism as restricting the assignment space, much like binding. There are nevertheless two ways of modelling the phenomenon. One is to

[30] This assumption was made due to several peripheral cases in which reflexives are seemingly capable for independent reference but which were excluded from the final version of this study due to reviewer comments.

use phi-features to restrict denotations and, as a consequence, blocking *he* from denoting female persons and ruling out assignments that one might still consider to be possible in some marginal or exceptional sense. This system was assumed in the present study, since the phi-features are viewed as the fundamental mechanism by which language creates denotational relations for referential expressions (i.e. the presence of phi-feature(s) = assume a referential expression). The alternative is to reject the offending assignments by an assignment filter. Under this system, an assignment in which *he* refers to Mary emerges into the list of assignments – as an assignment that is in principle possible and visible for global cognition – but is ruled out by a separate semantic or pragmatic filter. This model can be supported by the observation that *he* can indeed denote a female person such as Mary, and even more so under the current cultural model in which gender and the associated pronominal system is considered fluid. However, the same data can also be captured by assuming that *he* is no longer marked for 'gender:male' for the speakers of this dialect, or that Mary (as a person representation) is not marked for binary male gender; it is unclear to the present author what kind of data could be used to distinguish the alternatives and the matter was put aside.

3.7 Null subjects, control and binding in Finnish (#53-61)

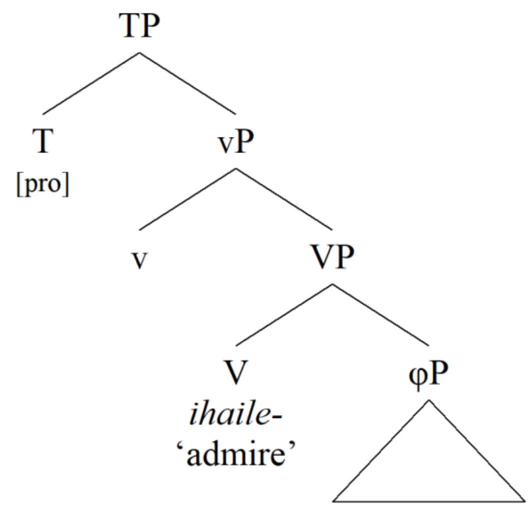
Finnish null subject (NS) profile presents several nontrivial problems for any rigorous analysis of binding. One issue is that if a language allows for null subjects, we need a mechanism for reconstructing them for the purposes of binding. Consider sentences such as (17):

(17)	Miksi	ihail-e-t	itse-ä-si?
	why	admire-PRS-2SG	self-PAR-PX2SG
	'Why do you admire yourself?'		

Native speakers will reconstruct the meaning as shown in the translation even though there is no overt pronominal subject corresponding to 'you'. Instead, the finite verb contains the corresponding second person singular agreement features. Interpreted under the present theory this means that the agreement features alone must be able to trigger both the process which creates the corresponding object 'you' into the global discourse inventory and which serves as an anchor for denotations and assignments. That is, semantic interpretation must treat (17) as a sentence with two participants. We must also make sure that if the subject appears in the sentence, redundantly as it were, the number of participants is not raised to three, that is, that the agreement cluster and the overt subject are in some sense the 'same thing'.

The baseline LPG model analyses sentences like (17) as bare T-vP structures without phonologically covert phrasal subjects but where T (the host of the second position singular agreement cluster) contains an independent pro-subject (18)(Brattico, 2021d):

(18)



This analysis works well in the present context in which the representation is created directly from the linear left-to-right assembly, where the overt subject is indeed absent. Specifically, there are no phonologically null elements such as null pronominal subjects present in the input, hence no such elements are generated.[31] The algorithm which projects objects into the global discourse inventory and created denotations and assignment treats pro-elements as independent referential arguments (example from sentence #53):

```
        8.Assignments:

            pro(-VT) ~ 1, [φ Merja] ~ 2
```

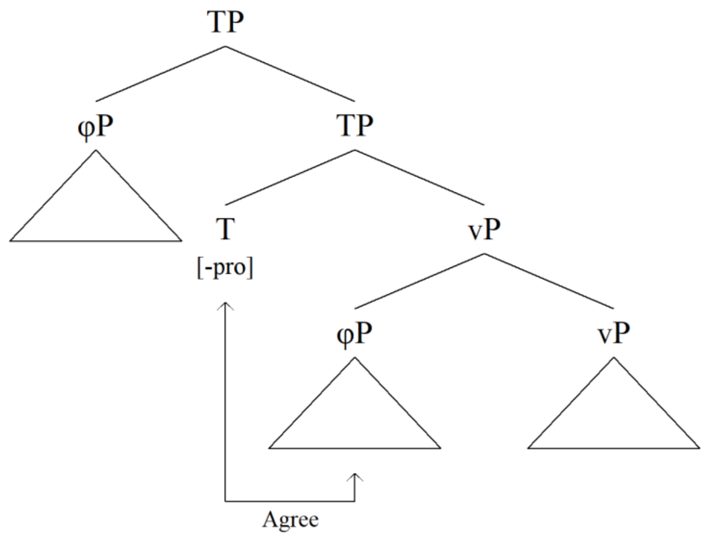
[31] We could develop an algorithm hallucinating syntactic representations for missing subjects (and other phonologically null elements), but this alternative was considered empirically and theoretically problematic (Brattico, 2021).

Symbol `pro(-VT)` refers to the phi-cluster inside T (-V refers to vowel lengthening), created on the basis of the agreement cluster extracted from the finite verb as shown in (19).

(19)	Miksi	ihail-e-t	itseäsi?
	why	admire-PRS-2SG	self.PAR-PX3
	…	V.v#T#2SG	
		T_{2sg}(v, V)	
		[…T_{pro} [v V …] …]	

If the sentence has a separate overt subject and agreement (#54), then Agree (agreement) takes place and pre-empts projection of separate semantic objects for the pro-element:

(20)



The pro-element is still present, however; Agree simply pre-empts the inventory projection. The result is visible in the assignments which ignore the pro-element:

```
          8.Assignments:

               [φ minä] ~ 1, [φ Merja] ~ 2
```

In the current model this condition is stipulated, but it is clear that there must be some connection between agreement, overt and covert arguments and agreement clusters (pro-elements).

Third person NSs have special properties in Finnish, a partial pro-drop language. The basic observation is that third person NSs cannot be dropped from the sentence unless there is a suitable antecedent in a syntactically more prominent position (21).

(21) a. *Ihaile-e Merja-a. (#54)
admire-PRS.3SG Merja-PAR
Intended: 'He admires Merja.'

b. Pekka$_a$ sanoo että ihaile-e$_a$ Merja-a. (#55)
Pekka says that admire-PST.3SG Merja-PAR
'Pekka says that he (=Pekka) admires Merja.'

What 'syntactically more prominent' and 'suitable' mean as properties of the antecedent is subject to controversy that was not addressed in this study, thus only unproblematic sentences such as (21)b were used as test items. Intuitively, however, the data suggests that third person null subjects are not fully independent referential items. They were treated as such and, as a consequence, they do not project objects into the discourse inventory. Sentence (21)a is ruled ungrammatical because T cannot be paired with an argument (another independent assumption needed in the system). This information is visible in the derivational log file:

```
36413      Argument for -VT° not found <== !
36414      Argument for v°: [φ Merja], indexed to [φ Merja]
36415      Argument for ihaile-°: [φ Merja], indexed to [φ Merja]
```

The EPP-profile of Finnish, controversial at present, reduces to the problem of 'argument identification' which fails in this case. The explanation is: third person agreement cluster does not project full referential arguments able to sustain reference → the corresponding third person object is not projected into semantic inventory → T is not paired with an argument and remain unsaturated, hence the sentence is judged ungrammatical. Properties of (21)b follow from the assumption that a (controlling) antecedent is accepted as a possible argument:

```
37410      Argument for T-V°: [φ Pekka], indexed to [φ Pekka]
37411      Argument for sano-°: [φ Pekka]
37412      Argument for -VT°: [φ Pekka]
37413      Argument for v°: [φ Merja], indexed to [φ Merja]
37414      Argument for ihaile-°: [φ Merja], indexed to [φ Merja]
```

The relevant information is on line 37412 which shows the argument calculations for the embedded T: it (correctly) targets the main clause subject. The mechanism is what we would intuitively refer to as control, though here a more appropriate (still problematic) term would be "finite control" since the dependency crosses the finite clause boundary. Control is part of the larger mechanism of argument identification in the LPG (Brattico, 2021d).

Let us then examine how the binding facts follow from these assumptions. Consider (22).

(22)	Pekka$_a$	sanoo	että	ihaile-e	hän-tä$_{*a,b}$. (#57)
	Pekka	says	that	admire-PRS.3SG	he-PAR

‘Pekka says that he (=Pekka) admires him.’

Because the third person agreement cluster does not project referential arguments, only two persons are projected into the global discourse inventory:

```
        Project object (1, QND) for [φ Pekka]
        Project object (1, GLOBAL) for [φ Pekka]
        Project object (2, QND) for [φ hän]
        Project object (2, GLOBAL) for [φ hän]
```

The assignments, which are correct, are calculated as follows:

```
        Assignment [φ Pekka] ~ 1, [φ hän] ~ 1 -
        Assignment [φ Pekka] ~ 1, [φ hän] ~ 2 +
        Assignment [φ Pekka] ~ 2, [φ hän] ~ 1 +
        Assignment [φ Pekka] ~ 2, [φ hän] ~ 2 -
        Summary:  Pekka[a] sano- ihaile- hän[b]
```

The regular pronoun and the subject of the superordinate clause must be disjoint, which follows from the fact that since the embedded clause does not have an intervening subject (neither overt phrasal subject nor pro-element), the locality principle sees the main clause subject as too local. The same mechanism explains why an embedded reflexive can take the main clause subject as its antecedent (#58). In other words, completely subjectless sentences, such as (22), extent the locality domains relevant for binding under the assumption that the third person agreement cluster does not project full arguments. If we add an independent first person pro-subject into the embedded clause, the pattern changes such that the embedded subject will now serve as an intervening element for binding (#59-61). Sentence (23)

(23)	Pekka$_a$	sanoo	että	ihail-e-n$_b$	hän-tä$_{a,*b}$. (#60)
	Pekka	says	that	admire-PRS-1SG	he-PAR

‘Pekka says that I admire him.’

projects assignments

```
            8.Assignments:

                [φ Pekka] ~ 1, pro(-VT) ~ 2, [φ hän] ~ 1
                [φ Pekka] ~ 1, pro(-VT) ~ 2, [φ hän] ~ 3
                [φ Pekka] ~ 3, pro(-VT) ~ 2, [φ hän] ~ 1
                [φ Pekka] ~ 3, pro(-VT) ~ 2, [φ hän] ~ 3
```

which are correct: the pronoun can denote the same person as the main clause subject, but not the person denoted by the embedded pro-element (or overt subject, if present).

3.8 Reconstruction (#101-110)

Binding reconstructs for A-bar and scrambling chains, created by reconstruction under the current LPG model. Binding is calculated on the basis of the LF-interface representations where all reconstructed arguments appear at their thematic base positions. A-bar reconstruction was tested by fronting pronominal, reflexive and proper name direct objects (#101-103), scrambling was tested by creating OVS orders with pronominal, reflexive and proper name direct objects (#104-109). In each case the correct binding properties follow because the noncanonical surface (PF-interface) properties are ignored by binding. This of course presupposes that the required operations exist in the baseline model, which is indeed the case (Brattico, 2020, 2023b; Brattico & Chesi, 2020). Reconstruction is part of the PF-LF mapping.

Let us consider sentence (24), #101 in the dataset.

(24)	Itse-ä-än=kö$_1$	Pekka	ihaile-e	__$_1$?
	self-PAR-PX3=Q	Pekka.NOM	admire-PRS.3SG	

‘Is it himself that Pekka admires?’

The left peripheral (second position) clitic =*kO* expresses yes/no questions as shown by the translation, and requires that the host phrase or word (which it is depends on several factors) is moved to the left periphery of the sentence (Brattico, 2021a, 2022). The operation is noncyclic and performed during PF-LF transfer:

```
----Noncyclic derivation-----------------------------------------------

    [[φ-kO itse] [[φ Pekka]:3 [-VT [__:3 [v ihaile-]]]]]

    Noncyclic Ā-chain([φ-kO itse])
     = [[φ-kO itse]:5 [C [[φ Pekka]:3 [-VT [__:3 [v [__:5 ihaile-]]]]]]]
```

Later the yes/no operator is scrambled into the direct object position:

```
Scrambling([φ-kO itse])
 = [[φ-kO itse]:5 [C [[φ Pekka]:3 [-VT [__:3 [v [<__>:5 [ihaile- __:5]]]]]]]]
```

This three-member chain derivation is by no means the correct one, perhaps not even plausible, but this is nevertheless how LPG performed reconstruction in the current study (Brattico 2019: §4.8.4). The question becomes quite complex when different alternatives are tested against nontrivial datasets. Moreover, how to distinguish Ā-reconstruction from scrambling reconstruction in a flexible word order language like Finnish is not trivial. But because the reflexive is reconstructed into the CompVP position – no matter how the result is accomplished – correct binding properties follow:

```
Assignments:
    Assignment [φ Pekka] ~ 1, [φ-kO itse] ~ 1 +
    Assignment [φ Pekka] ~ 1, [φ-kO itse] ~ 2 -
    Assignment [φ Pekka] ~ 2, [φ-kO itse] ~ 1 -
    Assignment [φ Pekka] ~ 2, [φ-kO itse] ~ 2 +
    Summary:  Pekka[a] ihaile- itse[a]
```

Notice that these properties would follow even if the second scrambling step would not occur at all or if the noncyclic Ā-reconstruction would be implemented in one step. The subject *Pekka* is reconstructed to SpecvP cyclically during an earlier stage of the derivation:

```
A-chain([φ Pekka])
 = [[φ-kO itse] [[φ Pekka]:3 [-VT __:3]]]
Agree(-VT) values [NUM:SG][PER:3] from goal __:3.
 = [[φ-kO itse] [[φ Pekka]:3 [T-V __:3]]]
```

These calculations do not affect binding: whether Pekka is at SpecTP or SpecvP and whether the chain is created cyclically or noncyclically has no effect on the assignments reported above.

Scrambling is handled in the same way, with the exception that the scrambling reconstruction, which accounts for the Finnish flexible word order profile (Brattico, 2020, 2023b), works differently (Brattico 2019: §4.8.7, 7.3.4.5). We consider sentence (25) with a marked OVS order.

(25)

Itse-ä-än	ihailee	Pekka.	OVS (#104)
self-PAR-PX3	admire-PRS.3SG	Pekka.NOM	

‘Pekka admires himself.’

The marked OVS order signals the fact that the reflexive represents the topic, Pekka represents new information, as in 'It was Pekka who admires himself' when 'admiring himself' has already been mentioned in the prior context and on such grounds constitutes the topic. The sentence involves two noncyclic scrambling reconstruction operations, lines 57102 and 57108.

```
57101      Scrambling([φ Pekka])
57102       = [[φ itse] [-VT [__:5 [v [ihaile- <φ Pekka>:5]]]]]
57103      Feature inheritance(φ)
57104       = [φ itse]
57105      Agree(φ) did not find suitable goal.
57106       = [φ itse]
57107      Scrambling([φ itse])
57108       = [<φ itse>:6 [-VT [__:5 [v [[ihaile- __:6] <φ Pekka>:5]]]]]
```

The postverbal grammatical subject is first reconstructed to SpecvP by an upward pointing reconstruction (hence, by downward operating movement), which is followed by object reconstruction into CompVP. Both arguments are initially analyzed as being adjoined to the structure (as shown by symbols <, >, see Brattico 2019: §4.7 and Brattico 2020). Current LPG model assumes that the Finnish flexible word order is licensed by case-based argument-adjunction option that allows thematic arguments and not just DP-adverbs to get adjoined to the structure (Brattico, 2018, 2020, 2023b). This model predicts that the freedom of argument ordering should mirror closely the freedom in adjunct ordering, and that the word order shifts in both cases should have the same effects on information structure. Both predictions are borne out. Be that as may, after the arguments have been reconstructed to their base position the correct binding facts follow:

```
57119
57120      = LF-interface [<φ itse>:6 [-VT [__:5 [v [[ihaile- __:6] <φ Pekka>:5]]]]]
57121
```

```
57145         Assignment [φ Pekka] ~ 1, [φ itse] ~ 1 +
57146         Assignment [φ Pekka] ~ 1, [φ itse] ~ 2 -
57147         Assignment [φ Pekka] ~ 2, [φ itse] ~ 1 -
57148         Assignment [φ Pekka] ~ 2, [φ itse] ~ 2 +
57149         Summary:  Pekka[a] ihaile- itse[a]
```

3.9 Nonfinite clauses (#111-133)

Binding properties of nonfinite complement clauses were tested with English *to*-infinitives (#111-119) and Finnish A-infinitives (#120-133), which have very similar LPG-analysis in which the the nonfinite clause segment is built by a α-vP structure where α is a separate infinitival head (corresponding to *to* in English and to the *(t)A*-suffix in Finnish)(Brattico, 2023a). Thus, sentence John wants to admire himself (#111) is provided with the following LF-interface analysis:

```

        = LF-interface [[D John]:2 [T [__:2 [v [want [to [v [admire [D self]]]]]]]]

```

where *to* stands for the infinitival head projecting an infinitival phrase. In Finnish, the structure is the same but the infinitival verb is created by moving and adding v + V to the infinitival head, creating one verb with an infinitival suffix *(t)A*. The basic regularity is that unless the embedded clause contains an intervening subject, the main clause + nonfinite clause complex is treated as one locality domain for binding (26).

(26) a. John$_a$ wants to admire himself$_{a,*b}$. (#111)
     b. John$_a$ wants Mary$_b$ to admire herself$_{*a, b}$. (#112)

Embedded subjects function as interveners for binding. The embedded subject can of course be a reflexive and get bound by the main clause subject (27)(#117).

(27) John$_a$ wants himself$_a$ to …

Again, there is nothing intervening between the embedded reflexive subject and the main clause subject, generating the coreference dependency illustrated in (27). This construction was tested for proper names (#116), reflexives (#117-118) and pronouns (#119). The same tests were replicated successfully in Finnish (#120-133). When reading the results it should be kept in mind that Finnish pronouns (including reflexive pronouns) do not have the gender-feature.

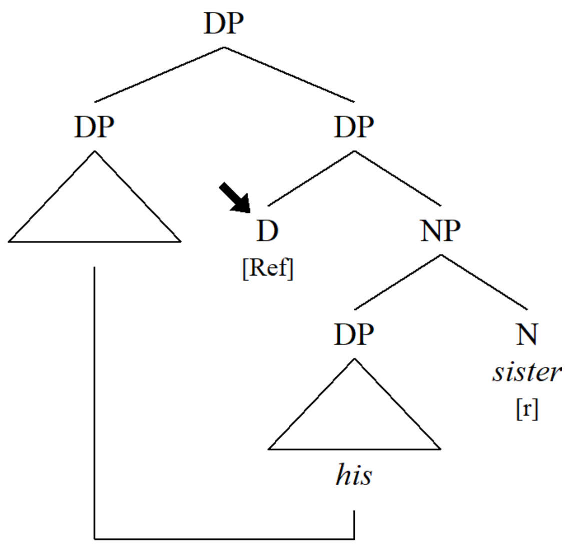
3.10 DP-internal binding (#134-139)

Two special problems of DP-internal binding were examined in this study. One question is how to bind DP-internal pronouns with the subjects of their own clause (28).

(28) John$_a$ admires his$_{a, b}$ sister. (#133-135)

This data follows from the fact that the pronoun is reconstructed from SpecDP into SpecNP (Section 3.2), which then has the consequence that the binding calculations from the pronoun are intervened by the referential feature at D:

(29)



```
        11.LF:

            [[D John]:2 [T [__:2 [v [admire [[D he]:5 ['s [__:5 sister]]]]]]]
```

This generates the correct binding dependencies, both in English and Finnish:

```
        14.Binding:

            John[a] admire he[a,c] sister[b]
```

```
        8.Assignments:

            [D John] ~ 1, [[D he] ['s [[D he] sister]]] ~ 2, [D he] ~ 1
            [D John] ~ 1, [[D he] ['s [[D he] sister]]] ~ 2, [D he] ~ 3
            [D John] ~ 1, [[D he] ['s [[D he] sister]]] ~ 3, [D he] ~ 1
            [D John] ~ 1, [[D he] ['s [[D he] sister]]] ~ 3, [D he] ~ 2
            [D John] ~ 2, [[D he] ['s [[D he] sister]]] ~ 1, [D he] ~ 2
            [D John] ~ 2, [[D he] ['s [[D he] sister]]] ~ 1, [D he] ~ 3
            [D John] ~ 2, [[D he] ['s [[D he] sister]]] ~ 3, [D he] ~ 1
            [D John] ~ 2, [[D he] ['s [[D he] sister]]] ~ 3, [D he] ~ 2
            [D John] ~ 3, [[D he] ['s [[D he] sister]]] ~ 1, [D he] ~ 2
            [D John] ~ 3, [[D he] ['s [[D he] sister]]] ~ 1, [D he] ~ 3
            [D John] ~ 3, [[D he] ['s [[D he] sister]]] ~ 2, [D he] ~ 1
            [D John] ~ 3, [[D he] ['s [[D he] sister]]] ~ 2, [D he] ~ 3
```

Referential expressions therefore create opaque binding domains for referential expressions inside them. The second problem addressed in this study were the picture-nouns (30).

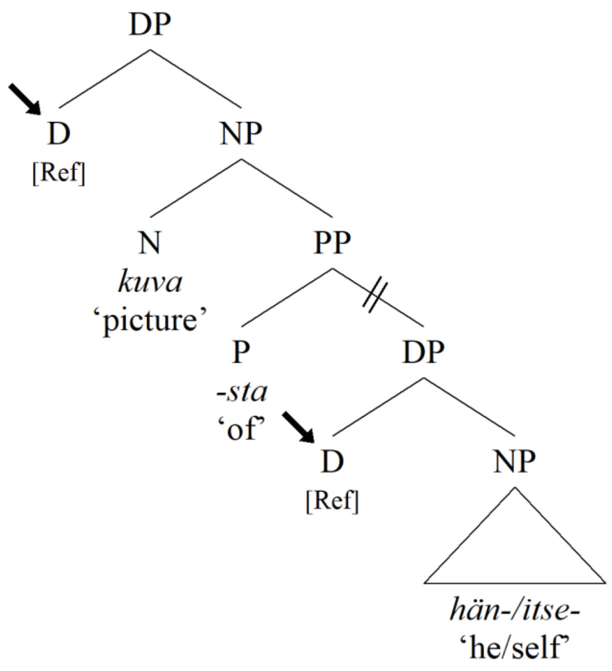
(30) Pekka$_{a}$	otti	kuvan	[$_{PP/DP}$	häne-stä$_{?a,b}$/	itse-stä$_{a,*b}$].
Pekka	took	picture		him-ELA	self-ELA

‘Pekka took a picture of him/himself.’

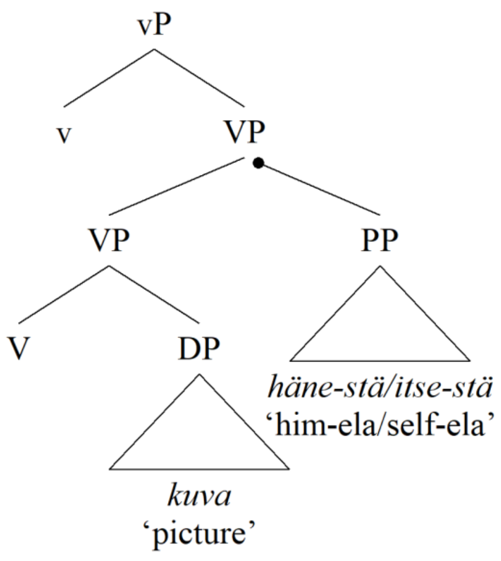
Whether the elative-marked noun is inside a DP or PP does not matter here (the latter was adopted), since in either case the phrase contains a head with [Ref] which should then block the reflexive pattern shown in (30). The data shows that this is not the case; instead, the reflexive reading is possible, while the pronoun behaves as if an intervention did occur.

The baseline model handles these cases correctly, however, since (30) is ambiguous between two readings: one in which the PP/DP is merged as the complement of the noun head (31), another where it is merged as a right-adjunct of the VP (32). The latter analysis circumvents the DP-internal intervention.

(31)



(32)

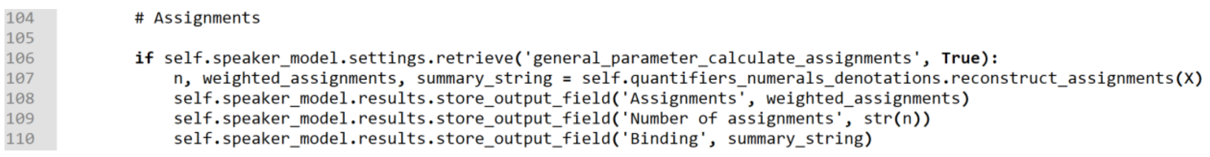


These analyses were tested by items #134-137.

4 Guide to implementation

The hypothesis was implemented in Python by adding the hypothetical mechanisms and representation into an existing LPG model. This section provides a brief guide to the implementation – how it is organized, how to find what from where – but without providing the details which are available in the source code documentation (Brattico, 2019).

Each input expression, provided as a linearly ordered sentence consisting of morphemes, is derived into a set of LF-interface representations, each which is submitted to semantic interpretation. These steps are handled by the speaker model (class `SpeakerModel` in module `speaker_model.py`). The relevant function `evaluated_complete_solution` which processes finished LF representations is in the class `SpeakerModel` (Brattico 2019: §8.1.1). This function calls `postsyntactic_semantic_interpretation` of the `narrow_semantics` module, which handles calculations that have to do with inventory projections, denotations and assignments, among other semantic tasks. The code generating denotations and assignment is inside this function:



```
        # Assignments

        if self.speaker_model.settings.retrieve('general_parameter_calculate_assignments', True):
            n, weighted_assignments, summary_string = self.quantifiers_numerals_denotations.reconstruct_assignments(X)
            self.speaker_model.results.store_output_field('Assignments', weighted_assignments)
            self.speaker_model.results.store_output_field('Number of assignments', str(n))
            self.speaker_model.results.store_output_field('Binding', summary_string)
```

The behavior can be turned on and off by setting the `calculate_assignments` parameter in the study configuration file (line 106, see the software documentation for how to do this). The semantic properties are calculated by `reconstruct_assignments` of the `quantifiers_numerals_denotations` module, the latter which handles semantic properties of 'thing' objects denoted by ordinary referential expressions. The remaining lines (108-110) store the outputs into the results fields, so that they can be shown to the user and compared with the native speaker properties specified and acquired in the dataset. Denotations and assignments are generated by

```
115         self.create_assignments_from_denotations(self.calculate_possible_denotations(X))
```

in the `reconstruct_assignment` function. Internally these functions work so that `calculate_possible_denotations` outputs a list of 4-tuples, where each tuple represents the properties of a referential expression in the input, including a set of possible denotations it may have.[32] For example, *John admires Mary* generates two 4-tuples, one for *John*, another for *Mary*. Function `create_assignments_from_denotations` takes the list as an input and creates the assignments. We consider all possible combinations of assignments from possible denotations (171), provided in the fourth element in the tuple (`tup[3]`, counting from 0), write each into a dictionary (175) and add each dictionary into an assignment list (179).

```
171         for assignment in itertools.product(*[tup[3] for tup in ref_constituents_lst]):
172 
173             #   Create assignment dict
174 
175             assignment_dict = {tup[0]: assignment[i] for i, tup in enumerate(ref_constituents_lst)}
176 
177             #   Calculate assignment weights and add the (assignment, weight) pair into all_assignments
178 
179             self.all_assignments.append(self.calculate_assignment_weight(assignment_dict, ref_constituents_lst))
```

Ontological compatibility, binding and other sematic properties are checked by `calculate_assignment_weight`. For example, binding properties are calculated by

```
191         for expression in ref_constituents_lst:
192 
193             # Violations of binding conditions reduce the weight to 0
194 
195             if not self.binding_conditions(expression, assignment):
196                 weighted_assignment['weight'] = 0
```

[32] The 4-tuple format was used for convenience and is not assumed to be realistic.

which sets `weight = 0` if the binding properties are violated. Function `binding_conditions` first checks if the expression has binding-related features (so-called R-features, from "referential," line 277) and then checks if they are violated (line 283). It calculates these properties on the basis of the expression, assignment and semantic working memory, the latter which is currently constructed inside the same function (line 283).

```
277         for f in self.get_R_features(X):
278             R, rule, intervention = self.parse_R_feature(f)
279 
280             # Check that no binding conditions are not violated
281             # Function construct_semantic_working_memory will create a set of object based on attributes complete, limited
282 
283             if binding_violation(assignment[idx], rule, X.construct_semantic_working_memory(intervention, assignment), X):
284                 return False
```

Binding violations are determined by `binding_conditions`:

```
        def binding_violation(semantic_object, rule_, semantic_working_memory, X):

            # Semantic working memory (whether complete, limited) is constructed by the caller and
            # is here provided as a set

            # RULE 1
            # If the same semantic object denoted by X is inside (complete, limited) semantic working memory but
            # X marked as NEW, raise violation

            if 'NEW' in rule_ and {semantic_object} & semantic_working_memory:
                log(f'-Illegitimate binder for {X.max().illustrate()}({semantic_object}) in WM ')
                return True

            # RULE 2
            # If the same semantic object denoted by X is not inside (complete, limited) semantic working memory
            # but X is marked for OLD, raise violation

            if 'OLD' in rule_ and not {semantic_object} & semantic_working_memory:
                log(f'-Binder missing for {X.max().illustrate()}({semantic_object}) from WM ({semantic_working_memory})')
                return True
```

Both semantic working memory (`semantic_working_memory`, a set) and semantic objects (`semantic_object`) contain numerical indices (referred to as `idx`) of the corresponding semantic objects. These string literals identify the objects: the processing sees only semantic objects, while structure has disappeared (it was still present when calculating the contents of the semantic working memory, the step just before the above computations). Notice that whether the semantic working memory is complete or limited has been determined by the caller. It is constructed by the function `construct_semantic_working_memory` in the phrase structure class which considers all referential expressions that have a denotation in the syntactic working memory.

References

Brattico, P. (2018). *Word Order and Adjunction in Finnish*. Aguila & Celik.

Brattico, P. (2019). *Computational implementation of a linear phase parser. Framework and technical documentation (version 20.1)*. https://github.com/pajubrat/parser-grammar/blob/master/docs/Old%20documentation/Computational%20implementation%20of%20a%20linear%20phase%20parser%20(version%2020.1).pdf

Brattico, P. (2020). Finnish word order: does comprehension matter? *Nordic Journal of Linguistics*, *44*(1), 38–70. https://doi.org/doi:10.1017/S0332586520000098

Brattico, P. (2021a). A dual pathway analysis of information structure. *Lingua*, *103156*.

Brattico, P. (2021b). Computation and the justification of grammars. *Finno-Ugric Languages and Linguistics*, *10*(1–2), 51–74.

Brattico, P. (2021c). Computational linguistics as natural science. *Lingbuzz/005796*. https://ling.auf.net/lingbuzz/005796?_s=MUzSsZPWMdmoX8nI&_k=3e1ty8tPIVI3hVmY

Brattico, P. (2021d). Null arguments and the inverse problem. *Glossa: A Journal of General Linguistics*, *6*(1), 1–29. https://doi.org/10.5334/gjgl.1189

Brattico, P. (2022). Predicate clefting and long head movement in Finnish. *Linguistic Inquiry*, *54*(4), 663–692. https://doi.org/http://dx.doi.org/10.1162/ling_a_00431

Brattico, P. (2023a). Computational analysis of Finnish nonfinite clauses. *Nordic Journal of Linguistics*, 1–40. https://doi.org/10.1017/S0332586523000082

Brattico, P. (2023b). Structural case assignment, thematic roles and information structure. *Studia Linguistica*, *77*(1), 172–217. https://doi.org/10.1111/stul.12206

Brattico, P. (2024). Computational generative grammar and complexity. *Software Documentation for Python Scripts Implementing Computational Generative Grammars*. https://github.com/pajubrat/Templates/blob/main/docs/Computational%20generative%20grammar%20and%20complexity.pdf

Brattico, P., & Chesi, C. (2020). A top-down, parser-friendly approach to operator movement and pied-piping. *Lingua*, *233*, 102760. https://doi.org/10.1016/j.lingua.2019.102760

Brattico, P., & Leinonen, A. (2009). Case Distribution and Nominalization: Evidence from Finnish. *Syntax*, *12*(1), 1–31.

Phillips, C. (1996). Order and structure [Ph.D. thesis]. In *MIT*.