SUPPLEMENTARY INFORMATION FOR THE ARTICLE "COMPUTATIONAL ANALYSIS OF FINNISH NONFINITE CLAUSES" PUBLISHED IN *NORDIC JOURNAL OF LINGUISTICS*

2023

Pauli Brattico

Abstract. This document provides supplementary information for the article "Computational analysis of Finnish nonfinite clauses" published in *Nordic Journal of Linguistics*. It contains a detailed discussion of the computational experiment, results and the technical details of the underlying Python algorithm.

1 INTRODUCTION

The purpose of this study[1] was to test an analysis of Finnish nonfinite clauses which were decomposed into a verb part and an infinitival part under [$_{\alpha P}$ Spec [α VP]] where α is one of the thirteen infinitival heads and Spec is the position of the phrasal subject, if present. Most of their

[1] doi:10.1017/S0332586523000082

syntactic properties were controlled by the lexical feature composition of α. The model was formalized in Python and tested against a dataset.

Finnish nonfinite clauses pose an interesting puzzle for linguistics due to their mixed nominal and verbal behavior. They are all derived from verbal stems, project subject and object arguments, describe events, and license adverbs, while at the same time they also exhibit nominal properties such as possessive suffixes, syntactic and semantic case features, and the distribution of regular arguments. Thus, several previous analyses have assumed that the nonfinite clauses are composed out of verbal and nominal projections, or from projections that are hybrid combinations of nominal and verbal features (Kiparsky 2019; Koskinen 1998; Vainikka 1989; Ylinärä 2018). The intuition pursued in the present study is different. Instead of assuming that the nonfinite clauses are verbal or nominal, or both, we assume that they are regular clauses, and that it is the finite clause that is exceptional. The parallelism between finite and nonfinite clauses is illustrated in (1).

(1) a. Nonfinite clause $[_{\alpha P}\ \alpha_{inf}^{0}\ [_{VP}\ V^{0}\ DP]]$ $(\alpha_{inf}^{0}$ = VA/inf, A/inf, TUA/inf, ...)

b. Finite clause $[_{\alpha P}\ \alpha_{fin}^{0}\ [_{VP}\ V^{0}\ DP]]$ $(\alpha_{fin}^{0}$ = finite T, Neg, Aux,...)

Both finite and nonfinite clauses are headed by α which represents among other things the TAM-properties, full in the case of finite clauses, reduced in the case of infinitives. In both cases the head may have features such as the EPP, which is generalized from the finite domain to the nonfinite domain. Agreement is similarly divided into two syntactically and morphologically distinct blocks, finite and nonfinite: finite agreement occurs at α_{fin}^{0} (e.g. at T, Neg), nonfinite agreement at α_{inf}^{0}.

Use of rigorous computational methods in theoretical linguistics tends still to raise scepticism and is frequently interpreted in a misleading way. This methodology is designed to provide the researcher an unambiguous deductive justification for a hypothesis in a research context where the relationship between the data and theory tends to be computationally complex. To show that the data follows from the theory, and especially that no unattested data follows, requires a method in which all consequences of the theory are explored mechanically in an error-free way and in which the results can be compared in an unambiguous way with an explicit dataset we wish to explain. This is where the computational methodology serves a useful purpose. Most linguistic theories posit combinatorial mechanisms, hence their logical consequences are so numerous that the enterprise falls out of any conceivable realm of practical possibility if attempted by the traditional paper and pencil methodology.[2]

In the early days of generative grammar, it seems, the above was taken for granted. The whole point of the enterprise was to "construct a formalized general theory of linguistic structure" and to "search for rigorous formulation" of linguistic phenomena in order to "gain a deeper understanding of the linguistic data" (Chomsky 1957 p. 5). This was contrasted with "obscure and intuition-bound notions" (ibid.). Indeed, the term "generative grammar" itself refers to a grammar that literally generates a set of expressions; it was not meant to 'generate' them in the imagination of the researcher. It was also clear that the whole point of formalization was to calculate data, not to pursue "logical niceties" (p. 5). Then much later in (Chomsky 1981) the author seems at first to be claiming that the study and its "leading ideas" were too premature and not serious enough to warrant similar rigor (pp. 1–3) which, although understandable, does not constitute an argument in favour of using "obscure and intuition-bound notions." Indeed, while ideas presented in that work where "only a sketch of a true formalization," there was a concern if "full-scale formalization is a worthwhile endeavor at the moment" (p. 336), to which Chomsky responded positively: "[T]he point has been

[2] For example, showing that some one sentence is ungrammatical within the context of a generative theory typically requires that we execute an exhaustive search of all logical consequences of the axioms of the theory and thus show that the sentence is not within that search domain, a domain that increases exponentially as a function of the number of primitive grammatical items it contains. We can face a wall of millions of calculation steps, in the case of just one nontrivial ungrammatical sentence. Moreover, we have to execute the calculations for all ungrammatical sentences in the dataset and do this every time we propose a change in the theory, no matter how small.

reached where these further steps [i.e. formalization] should be undertaken, that there is a sufficient depth and complexity of argument so that formalization will not merely be a pointless technical exercise but may bring to light errors or gaps and hidden assumptions, and may yield new theoretical insights and suggest new empirical problems for investigation" (ibid.). Yet, it was not attempted there or later, and the whole issue was dropped, for reasons unknown to me but which seem unjustifiable in any imaginable form; here we propose a return to the earlier, more rigorous and in my view also more sound methodology.[3]

2 Hypothesis

2.1 General description

The analysis was tested by a Python-based recognition grammar originally proposed in (Brattico 2019a) and then applied in a number subsequent papers to a variety of linguistic phenomena. The model reads input sentences one phonological word at a time, creates an ordered set of parsing solutions as it consumes the words from left to right and explores the solutions by recursive backtracking. The general idea of the left-to-right assembly comes from (Phillips 1996). The model describes both competence, knowledge of language including grammaticality, and performance, the complete step-by-step process of comprehending natural language expressions.[4]

[3] What is missing from Chomsky's comments above is the fact that these "technical exercises," as they are used in all advanced sciences, do not exist solely for the purposes of discovery; their primary role is justification. In order to justify the "leading ideas" there is no other way than to show in a literal sense that they entail the data. The additional and often irrelevant technical problems that come with full-scale formalizations are there not to distract attention from the leading ideas but to show that the leading ideas have a realization under which the data does indeed follow from theory.

[4] The main article describes the process in terms of information processing and language comprehension. This is indeed the intended interpretation. However, it does not follow that one could not apply the left to right assembly also to speech production or to a traditional enumerative generative grammar. In the latter case, sentences are generated by building them in a left to right order (without ambiguity) instead of the standard bottom-up derivation. We would then use primitive lexical items and not phonological words as input.

Suppose the input sentence is *the horse raced past the barn*. We consider the input sentence to be a linear sequence of phonological words, presupposing segmentation and normalization. Phonological words are matched with lexical entries in the surface lexicon and retrieve corresponding lexical items (Section 2.2), which are forwarded to the syntactic component where they are assembled into a syntactic representation by chunking the words into an asymmetric binary-branching bare phrase structure representations (Section 2.3). In this case, the first solution generated by the parser could be α = [$_{\text{VP}}$ [$_{\text{DP}}$ *the horse*] *raced* [$_{\text{PP}}$ *past* [$_{\text{DP}}$ *the barn*]]]. The solution is then transferred to the syntax-semantic interface ("LF interface") where it is evaluated for well-formedness (Section 2.4). If the solution is accepted, it will be interpreted semantically by the semantic module. If it is rejected, no semantic interpretation arises, and the parser will backtrack. If all solutions generated for the input sentence are rejected, the sentence is judged ungrammatical: it has no legitimate syntactic or semantic interpretation.

Suppose, for example, that the input contains one more word *fell*. There is no position in α where this verb could be attached to generate a well-formed output, so the parser will reconsider earlier attachment decisions until it finds [$_{\text{VP}}$ [$_{\text{DP}}$ *the horse* [*raced past the barn*]] *fell*] that is accepted at the syntax-semantics interface with the meaning 'the horse, which was raced past the barn, fell'. As a whole, the algorithm implements a characteristic function for the set of grammatical sentences in any language (depending on what is available in the lexicon), provides each input sentence with a derivation (step-by-step description of the whole process), and furthermore provides each grammatical sentence with a syntactic structure and semantic interpretation. It therefore implements a *recognition grammar*.

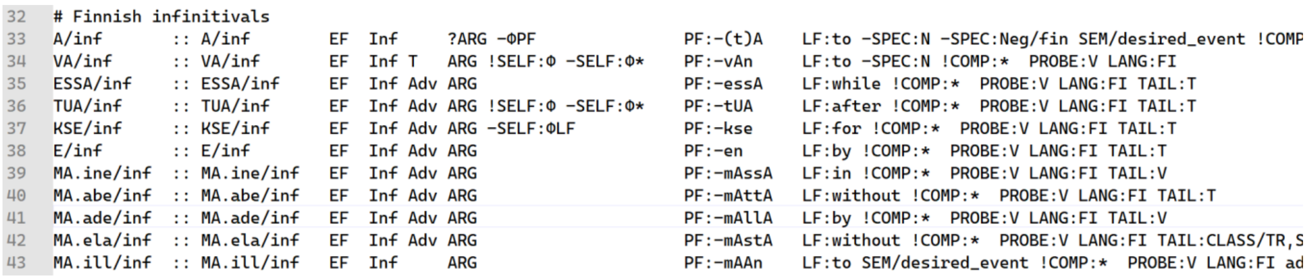
2.2 Lexicon and the lexico-morphological component

The surface lexicon maps phonological words into *lexical entries*. A lexical entry contains lexical information used to prepare lexical items for further processing in the syntactic and semantic components. The simplest type of lexical item is a *primitive lexical item*, which consists of a set of lexical features. The lexical entry for the English definite article *the* is a primitive lexical item and pairs the phonological word with a set of features. Lexical features are string patterns recognized by finite-state models (i.e. feature patterns are regular expressions). How the features are interpreted and what they signify is represented implicitly in the algorithm that processes them.

The second lexical type is constituted by *morphologically complex words* or morphological chunks which are linear sequences L_1#L_2#...#L_n of further items in the same lexicon, usually (currently exclusively) primitive lexical items. Finnish infinitival predicates are morphologically complex entries in this sense. The infinitival predicate *juokse-vAn* 'run-VA/INF' consists of a verb stem and the infinitival suffix. Both items point to further lexical entries: the primitive lexical item for the verb *juokse-* 'run' and the primitive lexical item for the infinitival suffix *-vAn*. We think of morphologically complex lexical entries as literal morphological chunks, complex representations that have been chunked to speed up processing. It is also possible to provide morphological complex words in their decomposed form in the input to simulate morphological parsing and to test the model with unattested but possible and unattested and impossible words.

A third type is constituted by *inflectional suffixes*, which map into sets of features but are inserted inside other lexical items and do not project grammatical heads of their own. Inflectional features can only be inserted into an adjacent head inside its own morphological chunk.

The lexicon is provided as a separate text file that the algorithm reads in the beginning of the simulation trial and is therefore easy to modify. The lexicon is stored into two files for convenience, the first which contains language-specific lexical items and the second hosting the (universal) functional morpheme repertoire. The Finnish infinitival heads (which we assume are ultimately derived from a universal set) are stored in the latter. They are shown below.



```
# Finnish infinitivals
A/inf       :: A/inf       EF  Inf      ?ARG -ΦPF                  PF:-(t)A    LF:to -SPEC:N -SPEC:Neg/fin SEM/desired_event !COMP
VA/inf      :: VA/inf      EF  Inf T    ARG !SELF:Φ -SELF:Φ*       PF:-vAn     LF:to -SPEC:N !COMP:*  PROBE:V LANG:FI
ESSA/inf    :: ESSA/inf    EF  Inf Adv ARG                         PF:-essA    LF:while !COMP:*  PROBE:V LANG:FI TAIL:T
TUA/inf     :: TUA/inf     EF  Inf Adv ARG !SELF:Φ -SELF:Φ*        PF:-tUA     LF:after !COMP:*  PROBE:V LANG:FI TAIL:T
KSE/inf     :: KSE/inf     EF  Inf Adv ARG -SELF:ΦLF               PF:-kse     LF:for !COMP:*  PROBE:V LANG:FI TAIL:T
E/inf       :: E/inf       EF  Inf Adv ARG                         PF:-en      LF:by !COMP:*  PROBE:V LANG:FI TAIL:T
MA.ine/inf  :: MA.ine/inf  EF  Inf Adv ARG                         PF:-mAssA   LF:in !COMP:*  PROBE:V LANG:FI TAIL:V
MA.abe/inf  :: MA.abe/inf  EF  Inf Adv ARG                         PF:-mAttA   LF:without !COMP:*  PROBE:V LANG:FI TAIL:T
MA.ade/inf  :: MA.ade/inf  EF  Inf Adv ARG                         PF:-mAllA   LF:by !COMP:*  PROBE:V LANG:FI TAIL:V
MA.ela/inf  :: MA.ela/inf  EF  Inf Adv ARG                         PF:-mAstA   LF:without !COMP:*  PROBE:V LANG:FI TAIL:CLASS/TR,S
MA.ill/inf  :: MA.ill/inf  EF  Inf      ARG                        PF:-mAAn    LF:to SEM/desired_event !COMP:*  PROBE:V LANG:FI ad
```

The left column contains the keys, symbols that retrieve the corresponding lexical items. What the keys are does not matter as long as all phonological words that occur in the sentences in the test corpus are mapped to a key (or several, if ambiguous). The key is followed by the list of lexical features. The first feature is the infinitival label targeted by selection and other grammatical operations. It also represents the idiosyncratic meaning of that particular infinitival type (e.g., ESSA/inf 'while doing', mapped into semantic types at the postsyntactic phase). This is followed by EF which corresponds roughly to the EPP in the main article and will be commented further below (EF comes from "edge feature").[5] Inf categorizes the head as an infinitive and [Adv] as an adjoinable adverb, licensing adjunct attachment. ARG makes the head a predicate that must be paired with an argument; –ΦPF blocks agreement. The feature TAIL licenses adjunction sites for adjuncts and regulates subject and object control in the present study. TAIL:T requires that the head is adjoined inside a TP, TAIL:V that it is adjoined inside a VP. PF provides a phonological form for the word, used only for output generation, and LF provides (links to) conceptual meaning. Features COMP:L and SPEC:L denote selection features in the usual sense, but come in three modalities:

[5] The EPP terminology was used in the main article due to its familiarity. The edge feature nomenclature comes from (Chomsky 2008).

!COMP:L/!SPEC:L denote obligatory selection, −COMP:L/−SPEC:L denote illicit selection, while COMP:L/SPEC:L denote licensing. See (Brattico 2019a: §4.9.4). The order of presentation of the features in the lexical files does not matter, since they are read into unordered sets.

Almost all adpositions license DP-complements. Repeating this information in connection with every adposition in every language introduces redundancy. This is eliminated by *lexical redundancy rules*. Lexical redundancy rules are defined by mapping feature sets into feature sets. The first set, called the *antecedent set*, determines the conditions under which the rule is triggered. If the lexical item retrieved from the lexicon contains the antecedent features, the rule is triggered. The features on the second set are then added to the lexical item.[6] If the features of the lexical redundancy rules and the specific lexical item conflict, the latter wins. Thus, the lexical redundancy rules represents default lexical values that can be overridden by specific lexical items. This makes it possible to model exceptions. The lexical redundancy rules are stored in a separate text file.

Languages differ from each other at least in terms of their lexicons. Lexical items can be provided with a language feature. The language feature can enter into the lexical redundancy rules, which allows the model to change feature content of the functional lexicon based on the language of the input sentence. For example, while many adpositions in Finnish can exhibit agreement, no adposition in English does. This can be represented by modifying adpositions during lexical retrieval by relying on a language-specific lexical redundancy rules. Before the simulation trial begins, the program constructs an idealized speaker model at runtime for the speaker of each language present in the lexicon, and these models then differ from each other in terms of their lexicons as defined by the language-specific lexical redundancy rules.

[6] Thus, the rule ‘all Ps select for DPs’ is implemented by a lexical redundancy rule ‘P $\rightarrow$!COMP:D’ which reads ‘for any lexical item x retrieved that has feature P, add feature !COMP:D’.

Each word, whether primitive or complex, will ultimately get decomposed into a linear sequence of primitive lexical items (including inflection and clitics). The order of the sequence is reversed, after which it is forwarded to the syntactic component.[7] The forwarding operation is called *lexical stream*.

The syntactic processing pathway can only manipulate special Python data-structures called *constituents*. A constituent is a linguistic object that falls under the domain of syntactic rules. Currently the lexical feature sets are wrapped inside constituents in the lexicon-morphological component. We can perhaps imagine that the lexicon itself stores constituents, although the interface is implemented by text files to make editing as easy as possible. Whatever the ultimate implementation, the result is that the syntactic component will receive lexical items as constituents. These are called *grammatical* or *lexical heads*, denoted by X^0. Each grammatical head sent by the lexico-morphological system into the syntactic system is a list of features wrapped inside a constituent.

2.3 Merge^{-1}

Primitive lexical items, now conceived as primitive constituents notated by X^0, arrive into the syntactic component as a linear stream. The syntactic working memory maintains a phrase structure representation containing all linguistic information received up to any given point during the processing. The incoming lexical items are attached to it by an operation Merge^{-1}. Superscript to -1 connotes ‘inverse Merge’ since the operation differs slightly from the standard bottom-up Merge proposed in (Chomsky 2008).[8] The underlying phrase structure is a variant of the bare phrase

[7] We could reverse the order inside the lexical entries by writing the morphological decompositions in reverse order, but currently the reversal is done algorithmically. The assumption is that it could potentially involve more complex operations.

[8] Merge^{-1} consists of four separate operations: (i) selection of the target right edge node A in the phrase structure in the active syntactic working memory, (ii) detachment of A from its host [β ∅], ∅ being the resulting null right constituent, (iii) creation of [A B] by bottom-up Merge, and (iv) substitution of ∅ by [A

structure model. Each *complex constituent* [A B] is an asymmetric binary-branching object containing the *left daughter constituent* A and the *right daughter constituent* B. We depict [A B] as a syntactic chunk. If the constituent is not complex, it is *primitive*. The label of the complex constituent is determined by a labelling algorithm. The labelling algorithm works (ignoring adjuncts) as follows: if A is primitive, then A will be the label; otherwise if B is primitive, it will be label; otherwise we apply the algorithm recursively to B. The resulting labels are marked in all phrase structure images created by the algorithm.

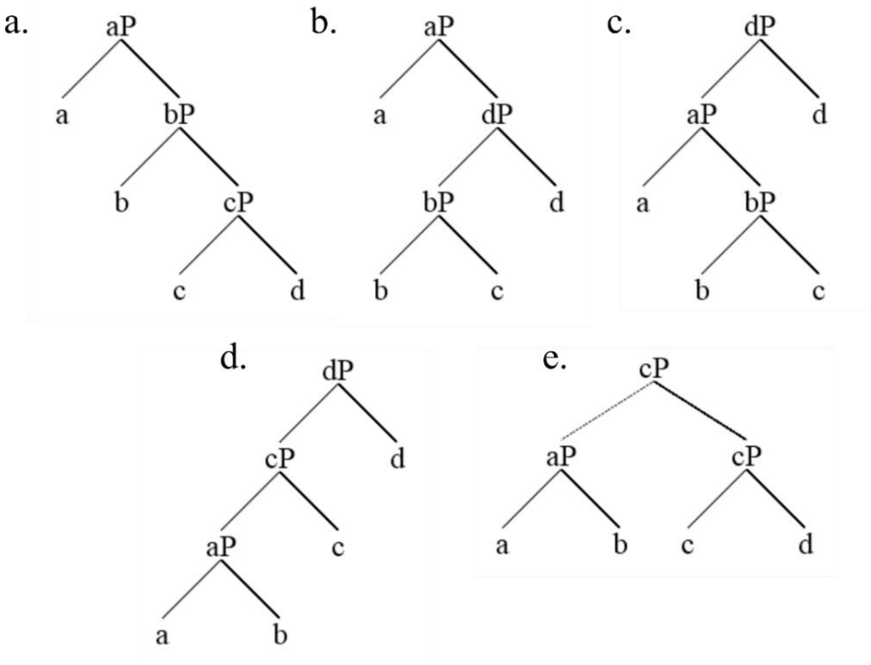
New lexical items (primitive lexical constituents) are always merged to the right edge of the phrase structure representation in the active syntactic working memory, where the right edge of [A B] denotes the right constituent B and applies recursively to B. Since the right edge may contain several nodes, a choice must often be made. It is made by filtering and ranking the possible attachment sites, which creates a parsing tree the algorithm explores by backtracking. Ranking and filtering are regulated by psycholinguistic plausibility rules which determine how well the behavior of the model matches with the behavior or native speakers in terms of processing speed and errors; when correctly implemented they have no bearing on grammatical judgments. Suppose the algorithm makes the decisions illustrated in α = [[*the horse*] *raced* [*past* [*the barn*]]]. Let us call the first solution the *first-pass parse*. If no more words arrive from the lexico-morphological component, this solution will be accepted and forwarded to the semantic component. Since we want to discover ambiguities, the model backtracks by using filtering and rankings established during the first-pass parse. It might next consider a solution where *barn* is merged to *[[*the horse*] *raced* [*past* [*the*]] + *barn*]]. Suppose, however, that another word *fell* arrives. All solutions where this word is attached to the right edge of

B]. Operations (i, iii) are part of the standard bottom-up Merge. Substitution (iv) has been used in several generative models and should not be considered a novelty. Detachment means that there is a brief moment during the derivation when the phrase structure is broken into two independent objects.

α fail to generate well-formed outputs. In this case, then, the correct solution can only be found by considerable backtracking, predicting an extra resource cost when processing *the horse raced past the barn fell*.

If we use the algorithm to process an ad hoc string a * b * c * d containing empty lexical items, it returns (2).

(2)



Filtering removes solutions from this set, while ranking defines the order at which the remaining solutions are explored. The default order, which emerges if there is no relevant lexical information, is illustrated in (2).

The question of how to handle morphologically complex words is nontrivial. The system described thus far decomposes all morphologically complex words inside the lexico-morphological component and then merges the results into the phrase structure as primitive lexical constituents. As a consequence, a single-word d#c#b#a will generate (2). Of this set, representations (b‒e) are implausible. Morphological constituents of phonological words do not scatter; only (a) is plausible.[9] Solutions (b‒e) are therefore filtered out. The consequence of this is that all morphological decompositions are performed during the pre-syntactic phase. In other words, head movement/reconstruction becomes non-syntactic. This works quite well, and was used in one published study (Brattico & Chesi 2020), but it does not capture all data. Even standard C-to-T head movement in English creates difficulties, but the model falls apart when we consider Finnish long head movement constructions (3).

(3)	Myy-dä-kö	Pekka	aiko-o	__	koko	omaisuutensa?
	sell-A/INF-Q	Pekka.NOM	plan-PRS.3SG		all	possessions

‘Does Pekka want to sell all his possessions?’ (*to sell* emphasized by prosody)

‘Is it selling that Pekka wants to do with all his possessions?’

Calculating (3) requires that we have complex words in the domain of syntactic rules. This problem was handled in (Brattico 2022a) by assuming that primitive constituents with one daughter constituent represent complex words in syntax, which means that the class of primitive constituents is partitioned into *primitive* and *complex heads*. Complex heads are reconstructed into syntactic head chains by reverse-engineered head movement algorithm (Section 2.4). For example, the English verb *admires* V#v#T is expanded into ‘...T...v...V...’. In order to calculate (3), and other examples of uncanonical head ordering, the mapping principle A#B (word) ~ $B^0 + A^0$ (lexical decomposition) ~ $[B^0...[...A^0...]]$ (syntax) has an additional step where the polymorphemic phonological word is represented as a complex head $B(A)^0$ in syntax.

[9] This is still an empirical assumption. It seems plausible, however, since the scattering algorithm if not restricted in any way treats the morphological constituents of phonological words exactly as it treats whole phonological words.

2.4 Transfer and the syntax-semantics interface

Once the parser generates a solution, it is *transferred* to the syntax-semantics interface (“LF-interface”) for evaluation. If the solution passes and is accepted, it is forwarded to the semantic component. If the solution is rejected, it is not forward to the semantic component and will not appear in the results. Accordingly, the semantic interpretation is generated on the basis of the syntax-semantics interface objects. Because the semantic interpretation is usually assumed to be universal, meaning that all humans (pathologies and pathological sensory deprivation aside) are assumed to be able to entertain the same thoughts independent of their language or culture,[10] the syntax-semantic interface must be language-independent. Since the input sentences are *not* language-independent, some type of normalization must occur prior semantic interpretation. Transfer accomplishes it.

Technically transfer can be described as a reverse-engineered transformational component of the standard generative theory, which reconstructs head movement (Brattico 2022a), adjunct movement (Brattico 2020), Ā-movement (Brattico *et al.* 2023; Brattico & Chesi 2020) and A-movement (Brattico 2019a, 2023a, 2023b). The result is a normalized syntax-semantics interface representation where (almost) all language-specific information has been eliminated. For example, complex infinitival predicates in Finnish such as *lähte-vän* ‘leave-VA/INF’ are expanded into independent grammatical heads, so that their nature as complex phonological words is eliminated and the output (4)a is aligned with the one engineered for English (b).

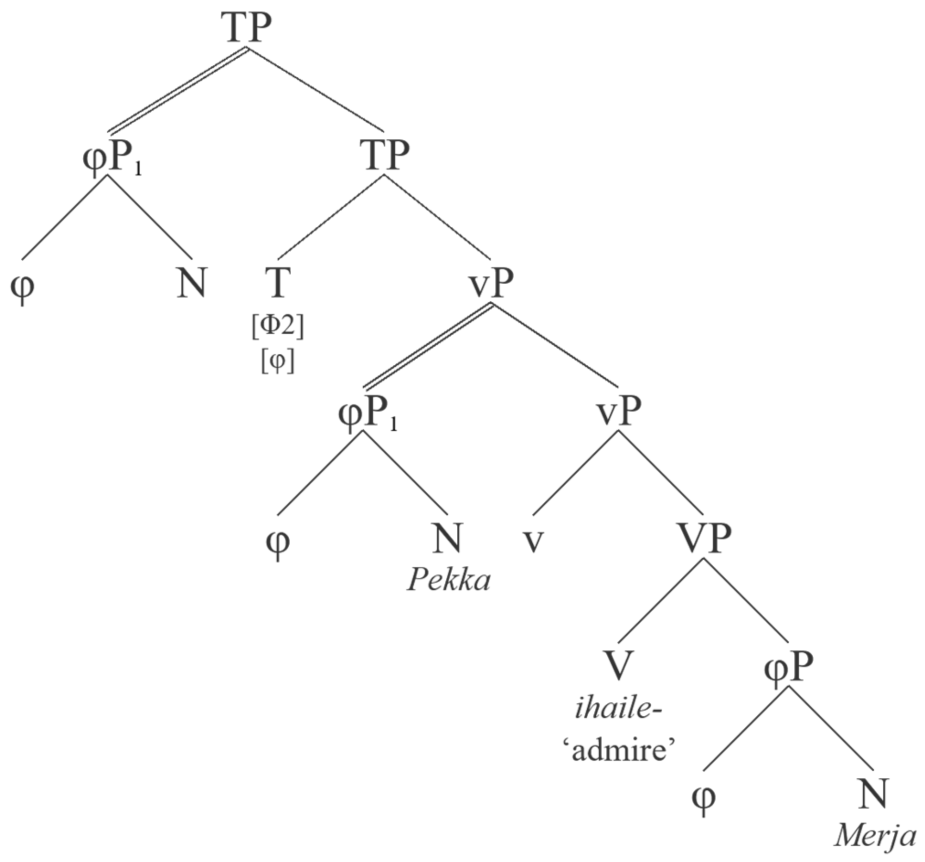
(4)	a.	Pekka	pyysi	Merjaa	lähte-mään.
		Pekka	ask	Merja	leave-MA/INF
		[DP	T	[DP	[$_{\text{InfP}}$ to$_{\text{ma/inf}}$ VP]]]

[10] Showing the contrary requires that we demonstrate that there are thoughts (or cognitive problems, plans) that only the speakers of some language can entertain (solve, formulate), supporting some form of linguistic supremacy. There is little evidence that anything like this is true.

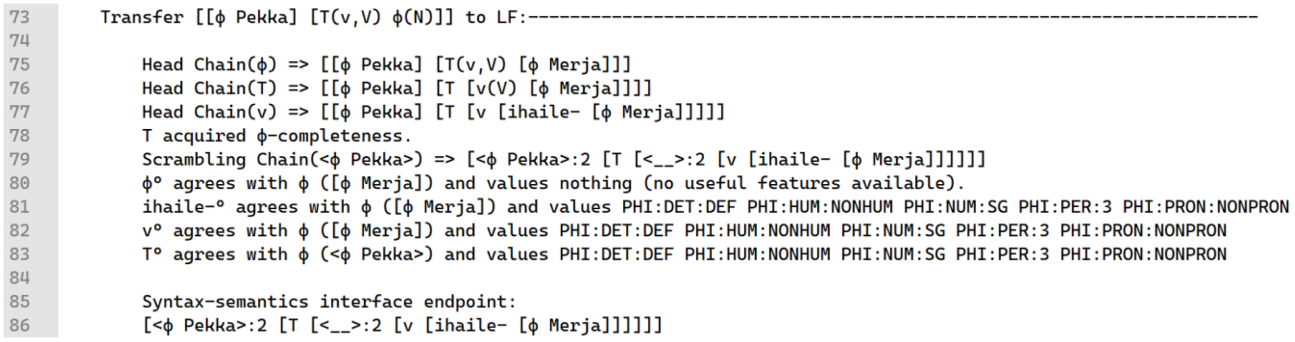
b.	John	asked	Mary	to leave.
	[DP	T	[DP	[$_{\text{InfP}}$ to VP]]]

Having arranged the elements in this way, language-independent syntactic processing and semantic interpretation can be applied.

The nature of transfer is controversial and technically nontrivial to implement. The nature of these operations depend on what's in the dataset and how far the successful operations can be simplified. This component plays a relatively small role in the present study, because the dataset does not have nontrivial chains. Two points merit a comment. Consider the phrase structure representation calculated for the sentence *Pekka ihaile-e Merja-a* 'Pekka.NOM admire-PRS.3SG Merja-PAR'.

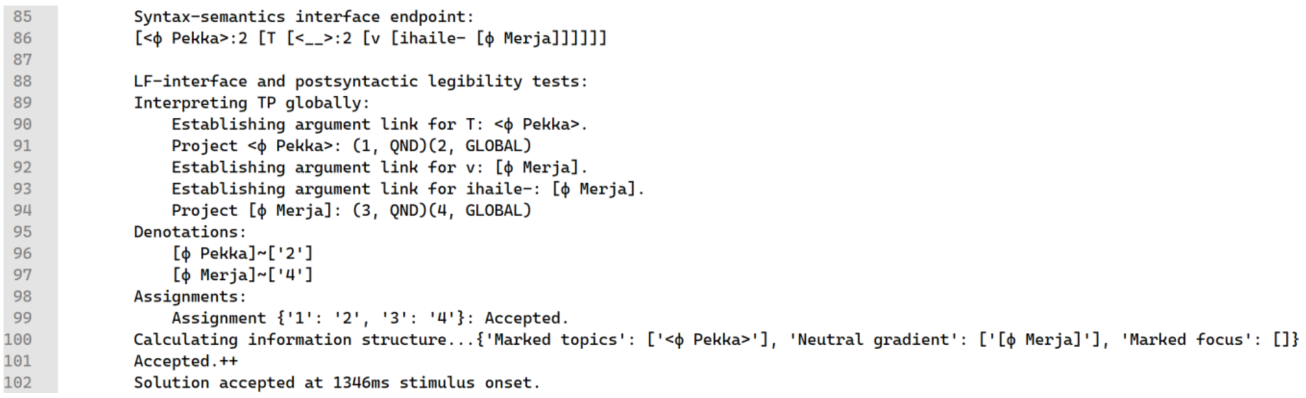


First, the infinitival predicate is expanded into the head chain T-v-V by the head reconstruction algorithm during transfer. Second, Finnish topicalization requires that all preverbal phrases at the topic position SpecTP are reconstructed into thematic positions in order to pair topics with thematic roles. In the above example, this is visible in the subject chain 1, where the preverbal subject *Pekka*, the topic of the sentence, is reconstructed to SpecvP where it receives the thematic role of agent. This operation takes place during transfer. We can see it in the derivational log file, shown below. The operation is on line 79.



```
73    Transfer [[ϕ Pekka] [T(v,V) ϕ(N)]] to LF:----------------------------------------------------------------------
74
75        Head Chain(ϕ) => [[ϕ Pekka] [T(v,V) [ϕ Merja]]]
76        Head Chain(T) => [[ϕ Pekka] [T [v(V) [ϕ Merja]]]]
77        Head Chain(v) => [[ϕ Pekka] [T [v [ihaile- [ϕ Merja]]]]]
78        T acquired ϕ-completeness.
79        Scrambling Chain(<ϕ Pekka>) => [<ϕ Pekka>:2 [T [<__>:2 [v [ihaile- [ϕ Merja]]]]]]
80        ϕ° agrees with ϕ ([ϕ Merja]) and values nothing (no useful features available).
81        ihaile-° agrees with ϕ ([ϕ Merja]) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON
82        v° agrees with ϕ ([ϕ Merja]) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON
83        T° agrees with ϕ (<ϕ Pekka>) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON
84
85        Syntax-semantics interface endpoint:
86        [<ϕ Pekka>:2 [T [<__>:2 [v [ihaile- [ϕ Merja]]]]]]
```

Other information visible here are as follows: (i) the first-pass parse solution (line 73), (ii) head reconstruction (lines 75-77), (iii) agreement reconstruction (lines 80-83) and (iv) the LF interface representation (line 86). Once transfer is complete, the output is evaluated for grammatical well-formedness at the syntax-semantics interface. The tests are called LF-legibility tests, and the result is sent out for semantic interpretation if the tests pass; otherwise the solution is rejected. An acceptable solution appears in the output of the algorithm, together with semantic interpretation created in the semantic component. These steps are visible just below the transfer in the derivational log file:



```
Syntax-semantics interface endpoint:
[<ϕ Pekka>:2 [T [<__>:2 [v [ihaile- [ϕ Merja]]]]]]

LF-interface and postsyntactic legibility tests:
Interpreting TP globally:
    Establishing argument link for T: <ϕ Pekka>.
    Project <ϕ Pekka>: (1, QND)(2, GLOBAL)
    Establishing argument link for v: [ϕ Merja].
    Establishing argument link for ihaile-: [ϕ Merja].
    Project [ϕ Merja]: (3, QND)(4, GLOBAL)
Denotations:
    [ϕ Pekka]~['2']
    [ϕ Merja]~['4']
Assignments:
    Assignment {'1': '2', '3': '4'}: Accepted.
Calculating information structure...{'Marked topics': ['<ϕ Pekka>'], 'Neutral gradient': ['[ϕ Merja]'], 'Marked focus': []}
Accepted.++
Solution accepted at 1346ms stimulus onset.
```

The field "LF-interface and postsyntactic legibility tests" is empty when the construction passes the tests; if there are problems, they appear here.

2.5 Control

The algorithm generates control dependencies when a predicate cannot be paired with a local argument at the syntax-semantics interface (Section 2.4), where the notion of local argument covers both phrasal arguments such as subject and objects and pro-elements generated on the basis of agreement suffixes in the input.[11] If neither is available, a nonlocal antecedent is targeted, resulting in subject and object control depending on the position of the predicate in the overall structure. The basic mechanism was proposed in (Brattico 2021b).

Suppose that a grammatical head H, such as an infinitival head, is a predicate and must be paired with an argument. First, the model examines if the head itself contains a conflict-free phi-set (number, person features) that could represent the argument. A VA-infinitival exemplifies this behaviour. If the condition is satisfied, as in (5), no control dependencies emerge.

[11] Whether a lexical item constitutes a predicate is specified in its lexical entry. A predicate refers to a lexical item that is unsaturated in Fregean sense: it must be paired with an argument in order to generate a well-formed semantic interpretation.

(5)	Pekka	luul-i	pes-see-**ni**	kaikki	ikkuna-t.
	Pekka.NOM	think-PST.3SG	wash-PST-1SG	all	window-PL.ACC(T)

‘Pekka thought that I (=argument) had washed all the windows.’

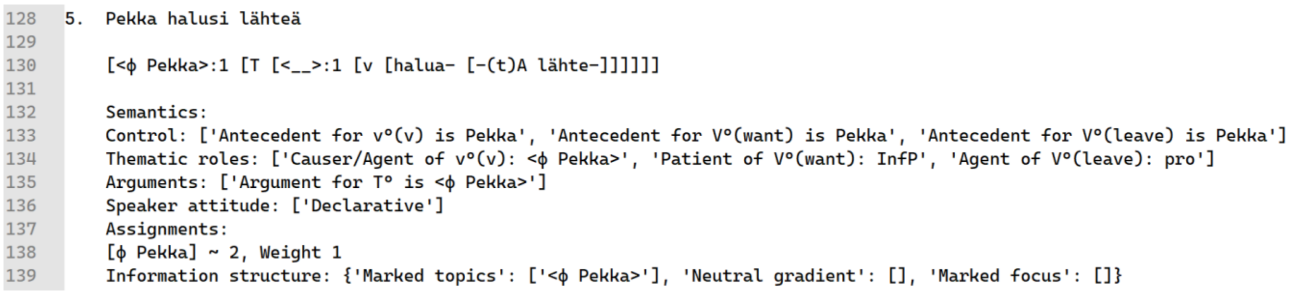
Here the predicate ‘to wash’ is paired with the first person (speaker).[12] If the head itself has no such features, an argument is first searched from the complement. This pairs verbs or prepositions with their object arguments (e.g., *kohti taloa* ‘towards house’, *pesin ikkunat* ‘washed.1SG windows’). If there is no complement, or if the complement cannot be interpreted as an argument, the algorithm examines all referential phrases which it can find from the active working memory. These correspond loosely to phrases that c-command the predicate, though technically the mechanism is different.[13] In some cases, the argument can be found locally from SpecHP (e.g., SpecvP, SpecPP, SpecAP), but if not, scanning continues upwards. This results is control, understood simply as an extension of the regular mechanism pairing predicates with arguments. Example (6) shows how the pairing works in the case of simple control construction in Finnish. There are no null arguments such as PRO in the underlying phrase structure representation.

(6)	Pekka$_1$	halus-i	lähte-ä$_{\text{arg=1}}$
	Pekka.NOM	want-PST.3SG	wash-A/INF

‘Pekka wanted to leave.’

Control dependencies are visible in the results file, where the relevant information is on line 133.

[12] If both a phrasal argument and agreement suffixes are present, an operation Agree checks that there is no conflict (i.e., agreement error).
[13] The current implementation uses a notion of path and connectedness, which was modelled on the basis of (Kayne 1983, 1984). It was also applied to Finnish structural case assignment in (Brattico 2022b).



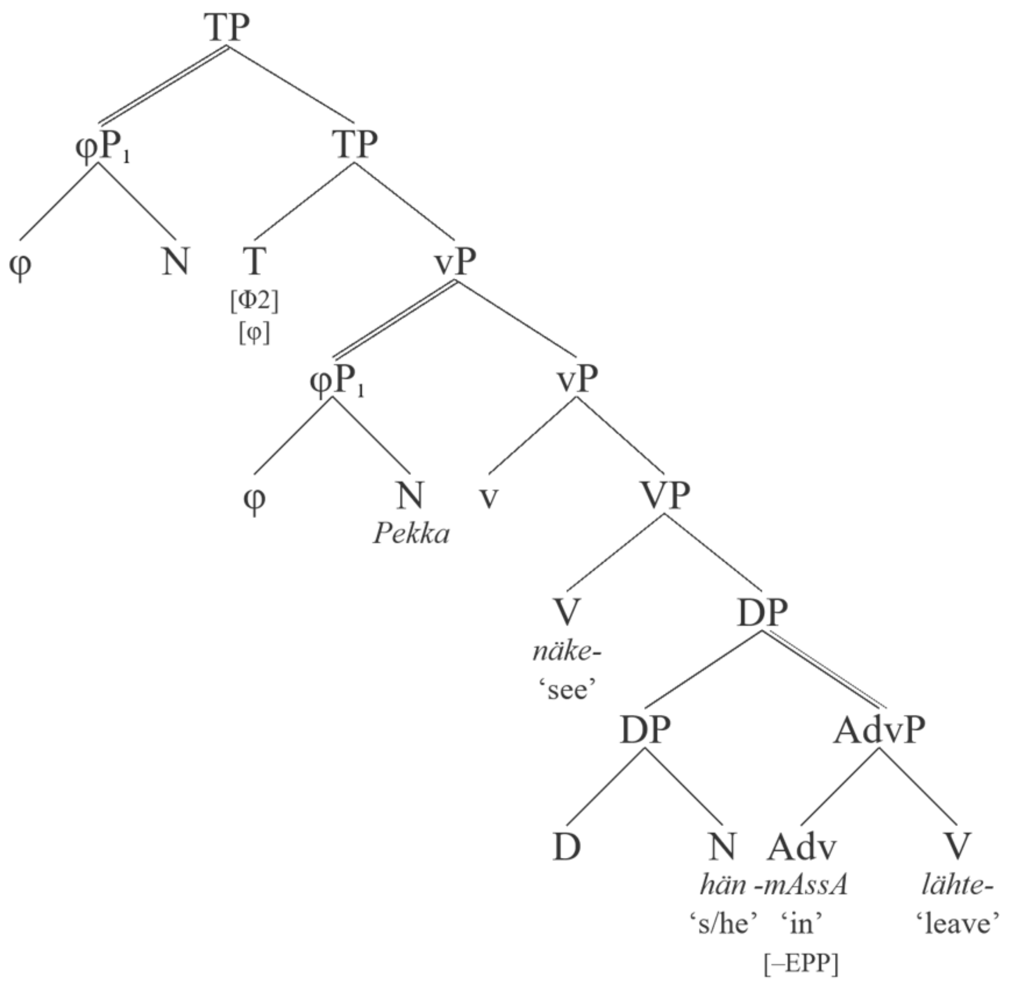
```
5.  Pekka halusi lähteä

    [<ϕ Pekka>:1 [T [<__>:1 [v [halua- [-(t)A lähte-]]]]]]

    Semantics:
    Control: ['Antecedent for v°(v) is Pekka', 'Antecedent for V°(want) is Pekka', 'Antecedent for V°(leave) is Pekka']
    Thematic roles: ['Causer/Agent of v°(v): <ϕ Pekka>', 'Patient of V°(want): InfP', 'Agent of V°(leave): pro']
    Arguments: ['Argument for T° is <ϕ Pekka>']
    Speaker attitude: ['Declarative']
    Assignments:
    [ϕ Pekka] ~ 2, Weight 1
    Information structure: {'Marked topics': ['<ϕ Pekka>'], 'Neutral gradient': [], 'Marked focus': []}
```

As a consequence, the position of the predicate and its hosting phrase determine the antecedent possibilities. For example, if the infinitival clause is right-merged above the VP, it cannot take object arguments as antecedents; if it is merged inside the VP, it can. Consider the two sentences from the dataset shown in (7), where the subscripts represent the predicate-argument pairings.

(7)	a.	Pekka$_1$	näk-i	hän-et$_2$	lähte-mä-ssä$_{*1,2}$. (#125)
		Pekka.NOM	see-PST.3SG	he-ACC(T)	leave-MA-INE
		‘Pekka saw him (not himself) leaving.’			
	b.	Pekka$_1$	näk-i(/suututti)	hän-et$_2$	lähte-mä-llä$_{1,*2}$. (#131)
		Pekka.NOM	see-PST.3SG(/angered)	he-ACC(T)	leave-MA-ILL
		‘Pekka saw(/angered) him by (himself) leaving.’			

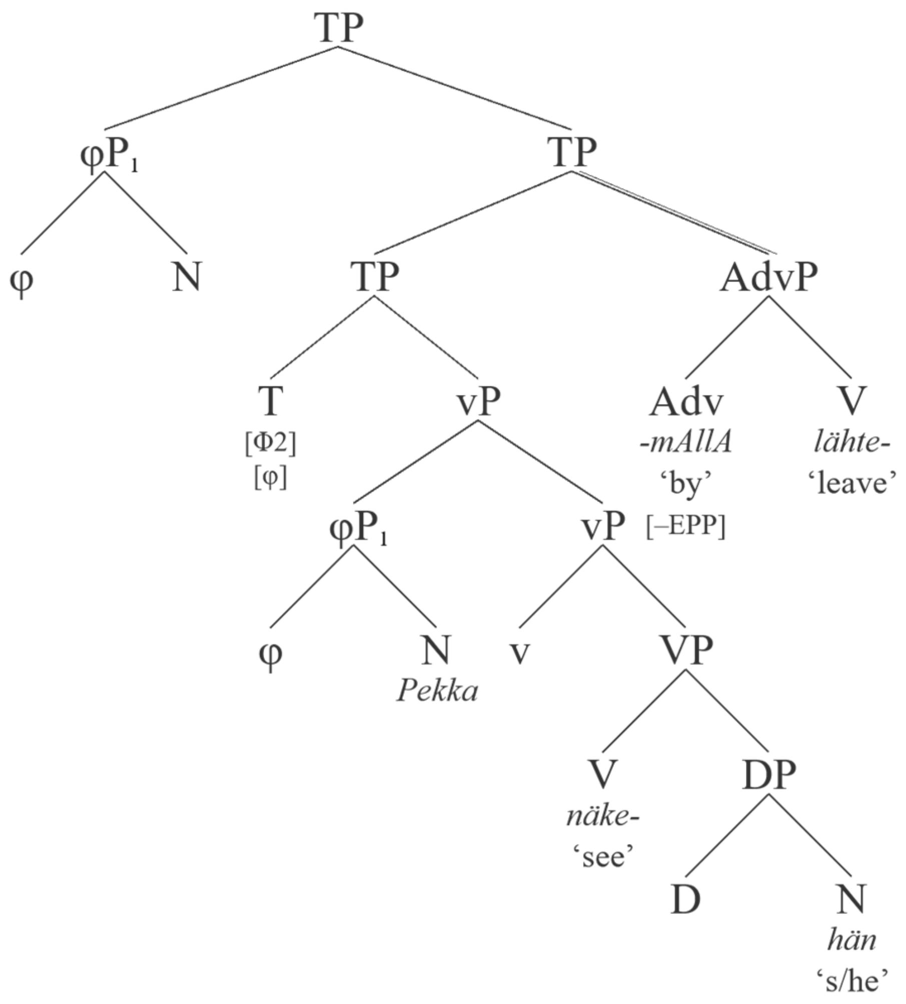
Let us consider the syntactic analyses. The model analyses (7)a as below, where the infinitival adjunct adverb clause is merged into a low position.

(8)



From this position the algorithm will find 's/he' as a possible and the most local antecedent for the infinitival predicate, and interprets the clause correctly as stating that Pekka saw the thirty party, not himself, leaving. Sentence (7)b is analysed as follows.

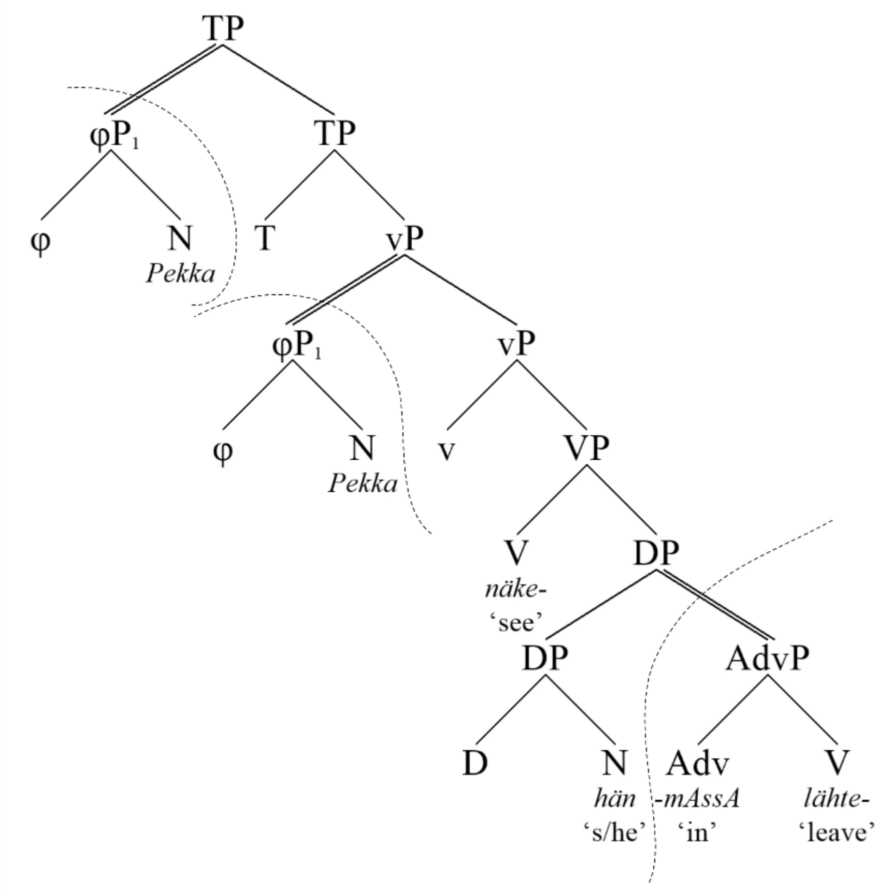
(9)



The infinitive is merged into a higher position, so the pronoun no longer appears in the list of possible antecedents. The antecedent will be the pro-element at T and/or the main clause subject. The position at which the infinitival adjunct is merged is controlled by a lexical feature which separates different types of adjunct adverbials from each other: the *mAssA* infinitive is right-merged inside the VP, whereas the *mAllA* adverbial inside the TP.

2.6 Adjunction and adjuncts

Many Finnish nonfinite clauses occur in an adjunct position either optionally or obligatorily. Adjuncts are created by merging the constituent into the phrase structure as a geometrical constituent and then by moving it out into a different syntactic working space. Once there, it retains its position as a geometrical constituent of the hosting phrase structure while remaining invisible to many grammatical dependencies and operations (labelling, sisterhood, minimal search, reconstruction) inside the host structure. As a consequence, adjunction increases the dimensionality of the phrase structure by generating parallel syntactic planes. The analysis is illustrated below.



Since the infinitival clause is merged to the hosting phrase as an adjunct, it does not affect labelling inside the host: thus notice that the combination of a DP and the adjunct is still a DP. From the point of selection, the complement of V is an ordinary DP. Adjunction is marked in the phrase structure tree by double lines, <XP> in the symbolic representations. Left adjunction (φPs in Figure 9) is relatively inconsequential, because left phrases are already ignored by the labelling algorithm.

2.7 Selection and feature checking

The analysis depends on compatibility checks between lexical items in local and nonlocal configurations. A lexical item can impose negative or positive selection for its own local complement or specifier or both. They are defined by lexical selection features which specify the type of selection (specific, complement) and the (one) selected feature.[14] The selected feature can be anything; in the typical case it is a major category label such as N or V. Thus, if we want to specify that V *usko-* 'believe-' cannot take an A-infinitival complement (**usko-n* [*Merja-n lähte-ä*] 'believe.1SG Merja-GEN leave-A/INF), we can provide feature –COMP:A/INF where A/INF is the feature that is selected, COMP designates complement selection and the minus sign means that the configuration is rejected. Complement selection relies on sisterhood; specifier selection, which is implied in subject identification in this study, is slightly more complex but can at first approximation be taken to refer to left nonadjunct phrases inside the head's projection. See Section 2.8. Since complex constituents do not have labels, selection targets the head/label X of the phrase XP as determined by the labelling algorithm (Section 2.3). The tests are performed at the LF-interface.

A mandatory positive complement selection is denoted by !COMP:F. This forces the head to have a complement of specific type, i.e. it creates serial head chains. For example, we might want to say that v^0 has !COMP:V and P^0 has !COMP:D. A positive selection feature without the exclamation mark licenses the selection. Licensing is used during parsing to guide the parser towards plausible solutions.

Many selection features express general patterns rather than idiosyncratic, random lexical properties. General selection features can be determined by the lexical redundancy rules. For example, the fact

[14] Conjunctive selection can be implemented by having two separate features; mandatory disjunctive selection (i.e. 'complement must be either A or B') is not currently possible.

that all infinitival heads license v/V complements can be determined by a lexical redundancy rule ‘INF→ COMP:V’ which puts these features inside any head with feature INF.

2.8 EPP

The EPP (from Extended Projection Principle) originally denoted a condition that English clauses must have a subject (Chomsky 1981, 1982). When working with Finnish, we must accommodate for the extra fact that null subject sentences, both finite and nonfinite, are grammatical if there is either a phrasal subject or sufficient verbal agreement (Holmberg 2005; Vainikka 1989; Vainikka & Levy 1999). Thus, the underlying algorithm assumes that both phrasal subjects and rich agreement can check the EPP feature, both in finite and nonfinite domains (Alexiadou & Anagnostopoulou 1998; Brattico 2021b; Brattico *et al.* 2023). Moreover, to capture the properties of all nonfinite clauses in Finnish we must posit three EPP profiles: obligatory EPP, which makes the subject mandatory, corresponding to the original definition; negative EPP, which blocks all kinds of subjects, creating heads which can be combined only with complements (e.g., MA-infinitivals); and heads without any specification that can occur with or without subjects (e.g., ESSA-infinitival).

The phrasal subject must be in the specifier position to check EPP. This specifier of H is a left phrase inside HP, with adjuncts and nonadjuncts included. The distinction is made in the manner explained in Section 2.6, but is ignored by the EPP mechanism. The definition allows heads to have several specifiers if other grammatical principles and lexical features license them. Left sisters are specifiers, but they are also complements which is defined by relying on sisterhood. Thus, XP in [$_{HP}$ XP H] is both a specifier and complement. This is not a contradiction: it means that XP can check both specifier and complement selection features.

Finnish EPP profile differs from English in that the preverbal subject position does not need to be the subject. It may be an adverbial, adposition phrase, expletive, subject argument, object argument, indirect argument and perhaps even a fronted VP. The preverbal constituent has many topic properties or is the topic (Brattico 2019b, 2021a; Holmberg & Nikanne 2002; Huhmarniemi 2019; Vainikka 1989, 1995). I will ignore this debate here. The relevance of this is that we cannot model Finnish by assuming that the edge feature of T is checked by a grammatical subject or even by a phrase with φ-features. What the mechanism is remains controversial. Notice that there are situations when nothing seems to appear in the preverbal subject position yet the clause is grammatical (10).

(10)	Pekka	sanoi	että	[__	aiko-o	tul-la	myöhemmin].
	Pekka.NOM	said	that		plans-PRS.3SG	come-A/INF	later

‘Pekka said that he (=Pekka) plans to come later.’

The bracketed finite clause is ungrammatical if it occurs alone, but grammatical if the superordinary clause as a suitable antecedent. See (Brattico 2021b).

The EPP mechanism presupposed in the current study requires minimally three behavioural profiles: mandatory EPP, represented by +EPP in the main article; illicit EPP, represented by –EPP; and optional EPP which was represented by having no EPP feature. This was the description provided in the main article, since it suffices for the present dataset. The implementation has undergone several iterations of development in recent research (Brattico 2023b, 2023a; Brattico *et al.* 2023). In the rest of this section, I will describe the implementation used in the present study together with some remarks towards the development of these assumptions. The term “implementation” refers to the actual technical system that was used to accomplish the three-way behavioural EPP-profile.

The first possibility is to posit a primitive EPP feature with the two diacritics +/–. This is possible but ultimately uninteresting, because it only replicates the attested behavior and moreover fails to link the phenomenon to anything else. In the earlier versions of the algorithm, +EPP and –EPP were modelled by unspecified specifier selection [!SPEC:*] and [–SPEC:*] which required or banned the occurrence of specifier of any type, respectively (Brattico & Chesi 2020). Optional EPP could then be represented by [SPEC:*] which licenses what is today referred to as the nonthematic second-merge option. Under this system, EPP become a form of nonthematic specifier selection, much like the thematic specifier selection of v. In order to account for the Finnish pro-drop phenomenon, this required that the notion of specifier was extended such that it applied also to a pro-element, induced by strong agreement at the finite verb, inside the head itself (Brattico 2021b). Thus, proceeding in this way, we would have EPP = SPEC:*.

An immediate problem is that this theory overgeneralizes: only in some languages like Icelandic (Holmberg 2000) can we perhaps assume that anything (represented by *) can satisfy the rule. Modelling the various restrictions on what can and cannot satisfy the EPP requires that we provide a list of selection features, for example, in English SPEC:D while in Finnish perhaps something like SPEC:TOPIC. The problem is that this would again just replicate the attested behavior. In other words, any kind of selection features can be posited in the lexicon, hence the theory make very few predictions. The system is more general and cannot be represented by independent selection features that are tied to specific lexical items.

The empirical data discussed in the present article shows that while agreement and phrasal specifiers are the two mechanisms to satisfy the EPP, in some cases redundant satisfaction is ungrammatical, thus we had to introduce the distinction between Φ1 (either agreement or a phrasal specifier but not both) and Φ2 heads (minimally agreement or a phrasal specifier). This distinction is not part of the original EPP mechanism and cannot be captured by the ( )SPEC:* theory; these features were simple added on the topic of the EPP.

Another distinction not touched upon in the main article is that we must distinguish between heads which project (i) mandatory specifier-subjects (+EPP), (ii) heads which project nonthematic specifiers that could be filled by nonsubjects (e.g., Finnish adpositions), and (iii) heads which project neither (English prepositions)(Brattico 2023b). The distinction between (i–ii) and (iii) was modelled, here as well as in more recent studies, by assuming that the former have feature EF (from "edge feature", in the sense of (Chomsky 2004, 2008)), and that the lack of this feature generates grammatical heads that behave like English adpositions and do not project specifier positions of any kind. This generates a stronger –EPP profile which blocks not only subjects but any kind of specifiers. Most grammatical heads in English or Italian belong to this type, but not so in Finnish, where nonsubject specifier generation is much more widely available and sometimes mandatory (e.g., Huhmarniemi 2012). Thus, in the lexicon and in the redundancy rules of the present study feature EF refers to this behavior: EF-heads project nonthematic specifiers that can be filled by subject and nonsubjects alike, while the lack of this feature (that can be notate it by –EF) does not project such positions. This feature is always *presupposed* when a nonthematic subject is possible (+EPP).

A third issue is that while the standard generative theory available today assumes that the EPP-feature (in whichever way it is implemented) literally causes ("triggers") movement to Spec, in the present model we have to check that the specifier is filled at the spellout structure. If the operation leaves a trace of its existence in the derivation, such as a copy of itself or a feature at the head whose specifier it was, we can create an appropriate filter at the LF-interface. This was implemented by relying on self-selection, i.e. features ( )SELF:F, which check that F is part of the head at the LF-interface, where F is a feature that represents some event that took place during the derivation and thus correspond to what in the standard theory "triggers" operations. We can then model the standard Finnish EPP by features !SELF:Φ1 or !SELF:Φ2 which require that either phrasal subject or agreement occurred.

–EPP could then be modelled by –SELF:Φ1. This misses an important generalization, however: in Finnish it holds quite generally (excluding finite C, T) that a head that cannot exhibit overt agreement cannot have phrasal subjects at Spec either (Brattico 2023b; Brattico *et al.* 2023). This generalization, argued for in (Brattico *et al.* 2023), was used in the present study, thus feature –ΦPF (banning overt agreement) by definition implied –EPP. Since also the lack of EF implied the same behavior, we arrive at the present implementation where –EPP was implemented by –ΦPF, EF (e.g., Finnish A-infinitives, MA-infinitives) or by having no EF (e.g., English prepositions, Finnish high complementizer), the difference being that the former allows nonsubject fronting while the latter does not and blocks all fronting operations.

3 Methods

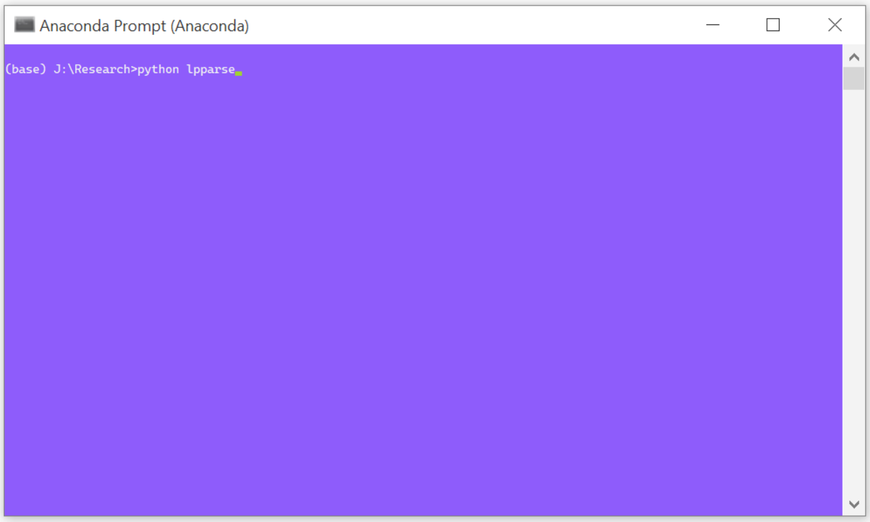
3.1 Design and stimuli

The computational experiment was performed by letting the algorithm implementing the analysis to process a dataset of Finnish nonfinite sentences. The dataset was generated by crossing the syntactic variables targeted for analysis in the main article and then by adding a few additional test cases such as the participle adjective constructions. The independent syntactic variables were the infinitival type (11, i.e. all infinitivals) x selecting verb (4, OC, non-OC, believe class, intransitives) x presence of infinitival agreement (2, yes or no) x presence of phrasal subject (2, yes or no) x syntactic position (2, complement, adjunct). The two participles were tested independently. All input sentences were processed without human intervention.

3.2 Configuring and replicating the study

The source code can be downloaded from the source code repository *http://www.github.com/pajubrat/parser-grammar*. For the exact replication the branch *Finnish-*

infinitivals-(Study-13c) should be used, for all other purposes the master branch which contains the latest version. The simulation is executed by writing *python lpparse* into the command prompt inside the root directly where the package was installed. The user must have Python installed on the local machine.



```
(base) J:\Research>python lpparse
```

This will execute the simulation as specified in the configuration file. The researcher must configure the simulation by providing values for several input parameters. These parameters are defined in the separate file *config_study.txt*, which is located in the root directory (containing the installation of the software on the local machine). The parameters are set into their correct values if the replication package is downloaded. First, the user can set the locations of the various input and output files:

```
2   author= Anon
3   year= 2023
4   date= March
5   study_id= 15
6   study_folder=           language data working directory/study-13-c-infinitivals
7   lexicon_folder=         language data working directory/lexicons
8   test_corpus_folder=     language data working directory/study-13-c-infinitivals
9   test_corpus_file=       infinitivals_corpus.txt
```

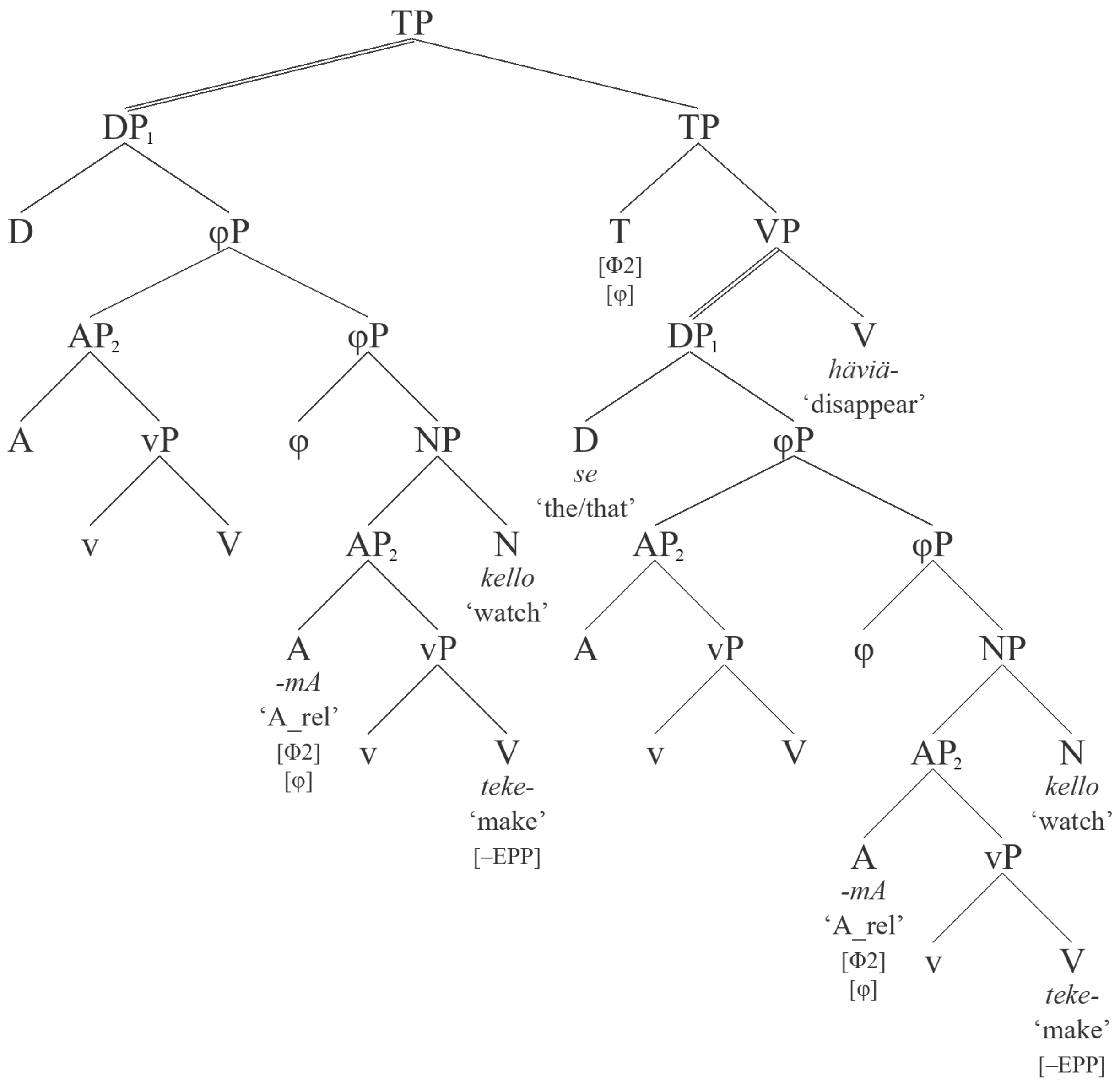
The locations for the input and output files are provided on lines 6-9. This is followed by general simulation parameters as follows:

```
11  # GENERAL SIMULATION PARAMETERS
12  only_first_solution = True
13  ignore_ungrammatical_sentences = False
14  console_output = Full
15  stop_at_unknown_lexical_item = False
16  logging = True
17  check_output = True
```

The parameter *only_first_solution* (line 12), when true, ignores ambiguities and produces only the first solution. This speeds up the simulation and makes the output easier to process, but may leave spurious grammatical solutions undiscovered. The field *check_output* compares the model judgments with the labels “grammatical” and “ungrammatical” provided by the user in the dataset file and provides a list of mismatches into a separate file. If the file is empty, then the model is observationally adequate. This is followed by parameters controlling image generation:

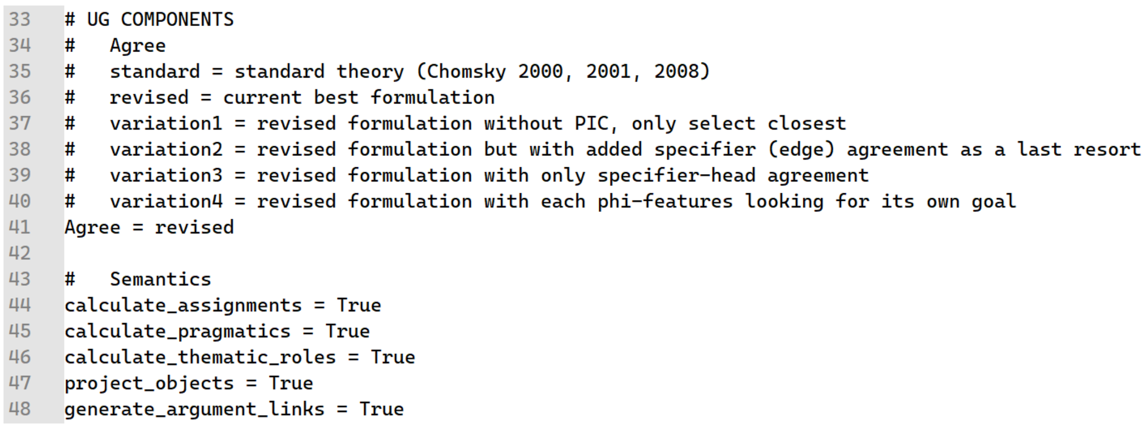
```
22  # IMAGE PARAMETERS
23  datatake_images = False
24  show_features = !SELF:Φ1,!SELF:Φ12,!SEF,ΦPF,-ΦPF,-SELF:ΦLF
25  image_parameter_show_words = True
26  image_parameter_nolabels = False
27  image_parameter_spellout = False
28  image_parameter_show_sentences = False
29  image_parameter_show_glosses = True
30  image_parameter_mark_adjuncts = True
31  image_parameter_stop_after_image = False
```

Image generation requires that the user has the Python pyglet package installed, hence it is set to *False* by default (line 23). The features printed into the phrase structure images are listed on line 24. The currently look as follows:



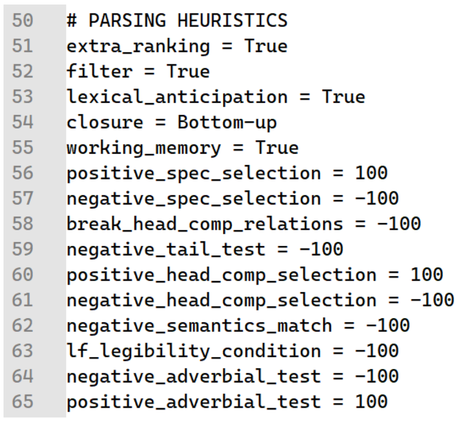
There is additional layer of complexity here due to the fact that the same source code may be used for several studies that may differ in their assumptions while being prepared for publication and/or submission or being under revision. Using different software branches for different studies was found impractical and confusing. This sometimes leads into a situation where the underlying implementation features differ from the ones discussed in the manuscript, for example because the final manuscript was prepared many months or years after the actual study was done. For this purpose the image drawing algorithm allows the user to map the symbols specified in the configurational file to other symbols used in the manuscript when producing the images.

The user must set the parameters of the general model of the UG used in the simulation.



```
33  # UG COMPONENTS
34  #   Agree
35  #   standard = standard theory (Chomsky 2000, 2001, 2008)
36  #   revised = current best formulation
37  #   variation1 = revised formulation without PIC, only select closest
38  #   variation2 = revised formulation but with added specifier (edge) agreement as a last resort
39  #   variation3 = revised formulation with only specifier-head agreement
40  #   variation4 = revised formulation with each phi-features looking for its own goal
41  Agree = revised
42
43  #   Semantics
44  calculate_assignments = True
45  calculate_pragmatics = True
46  calculate_thematic_roles = True
47  project_objects = True
48  generate_argument_links = True
```

Currently, it is only possible to select the model for Agree (lines 34–41). The user should use *revised*, which calculates the Finnish agreement profile correctly. The fields under *Semantics* allows the user to control the type of semantic interpretation calculated by the model. When set to False, the calculations do not take place (speeding up processing) and do not appear in the output. Finally, the file contains a list of cognitive parsing heuristics:

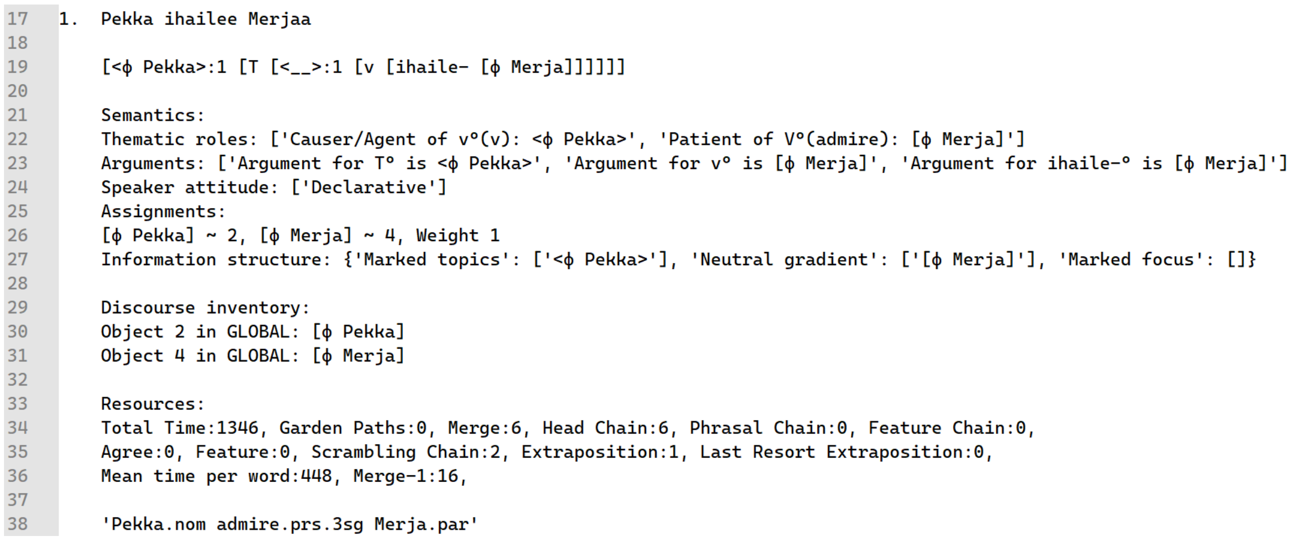


```
# PARSING HEURISTICS
extra_ranking = True
filter = True
lexical_anticipation = True
closure = Bottom-up
working_memory = True
positive_spec_selection = 100
negative_spec_selection = -100
break_head_comp_relations = -100
negative_tail_test = -100
positive_head_comp_selection = 100
negative_head_comp_selection = -100
negative_semantics_match = -100
lf_legibility_condition = -100
negative_adverbial_test = -100
positive_adverbial_test = 100
```

These are irrelevant for the present study.

3.3 Output

The algorithm pairs each input sentence in the dataset with a *derivation* (explicit step-by-step description of all linguistically and cognitive relevant calculation steps that took place during the processing of each input sentence) and a *grammaticality judgment* (grammatical, ungrammatical). All sentences judged grammatical were further paired with a *syntactic* and *semantic analysis* (or several if the input was ambiguous). The output is recorded into text files and phrase structure images, which constitute part of the raw data of the study. The following screenshot is from the results file and contains an overview of the syntactic and semantic analysis for sentence #1 *Pekka ihailee Merjaa* ‘Pekka admires Merja’.



```
17  1.   Pekka ihailee Merjaa
18
19       [<ϕ Pekka>:1 [T [<__>:1 [v [ihaile- [ϕ Merja]]]]]]
20
21       Semantics:
22       Thematic roles: ['Causer/Agent of v°(v): <ϕ Pekka>', 'Patient of V°(admire): [ϕ Merja]']
23       Arguments: ['Argument for T° is <ϕ Pekka>', 'Argument for v° is [ϕ Merja]', 'Argument for ihaile-° is [ϕ Merja]']
24       Speaker attitude: ['Declarative']
25       Assignments:
26       [ϕ Pekka] ~ 2, [ϕ Merja] ~ 4, Weight 1
27       Information structure: {'Marked topics': ['<ϕ Pekka>'], 'Neutral gradient': ['[ϕ Merja]'], 'Marked focus': []}
28
29       Discourse inventory:
30       Object 2 in GLOBAL: [ϕ Pekka]
31       Object 4 in GLOBAL: [ϕ Merja]
32
33       Resources:
34       Total Time:1346, Garden Paths:0, Merge:6, Head Chain:6, Phrasal Chain:0, Feature Chain:0,
35       Agree:0, Feature:0, Scrambling Chain:2, Extraposition:1, Last Resort Extraposition:0,
36       Mean time per word:448, Merge-1:16,
37
38       'Pekka.nom admire.prs.3sg Merja.par'
```

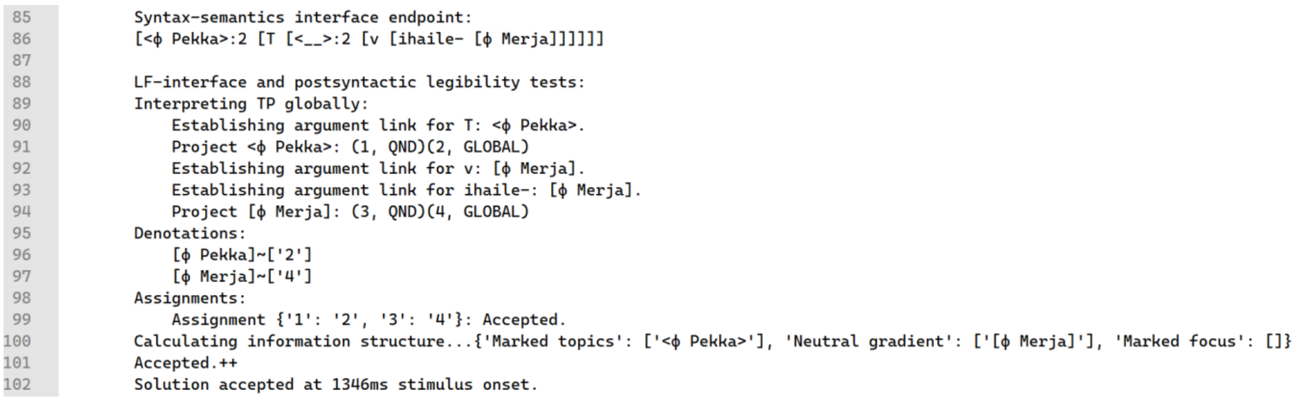
The information appearing in this file are as follows: input sentence (line 17), final syntactic analysis at the LF-interface representation (line 19), thematic roles (line 22), arguments determined by Agree (line 23), speaker attitude (line 24), interpretation of referential expressions (line 26), information structure (line 27), objects projected into existence while processing the sentence (lines 30-31) and resource consumption (lines 33-36). The details of the derivation are available in the derivational log file. It begins by providing details concerning the constructions of the first-pass parse:

```
#1. Pekka ihailee Merjaa
['Pekka', 'ihailee', 'Merjaa']

     1. ['Pekka', 'ihailee', 'Merjaa']

    Morphological decomposition of /Pekka/ ~ ['Pekka-', 'ϕ$', 'sg$', '3p$', 'def$', 'nonhum$', 'proper_name$', '[nom]$']
    Next affix [[nom]]
    Next affix [proper_name]
    Next affix [nonhum]
    Next affix [def]
    Next affix [3p]
    Next affix [sg]
    Next morph [ϕ] ~ ϕ°

    Next morph [Pekka-] ~ N°

        Ranking:
        (1) [ϕ↓+ N°](⇒ #0x1#)
        #0x1#
        Merge(Pekka) => ϕ(N)

    Morphological decomposition of /ihailee/ ~ ['ihaile-', 'v$', 'T/fin$', '[V]$']
    Next affix [[V]]
    Next morph [T/fin] ~ T°

        Filtering and ranking merge sites...{'T/fin'}-Comp selection for ϕ(N)...(-100)
        Ranking:
        (1) [ϕ(N)↓+ T°](⇒ #0x2#)
        (2) [ϕ(N) + T°](=> #0x3#)
        #0x2#
        Merge(T) => [[ϕ Pekka] T]

    Next morph [v] ~ v°

        Ranking:
        (1) [T↓+ v°](⇒ #0x4#)
        #0x4#
        Merge(v) => [[ϕ Pekka] T(v)]
```

This shows the word-by-word left-to-right derivation of the first-pass parse. This is followed by transfer:

```
    Transfer [[ϕ Pekka] [T(v,V) ϕ(N)]] to LF:------------------------------------------------------------------------

        Head Chain(ϕ) => [[ϕ Pekka] [T(v,V) [ϕ Merja]]]
        Head Chain(T) => [[ϕ Pekka] [T [v(V) [ϕ Merja]]]]
        Head Chain(v) => [[ϕ Pekka] [T [v [ihaile- [ϕ Merja]]]]]
        T acquired ϕ-completeness.
        Scrambling Chain(<ϕ Pekka>) => [<ϕ Pekka>:2 [T [<__>:2 [v [ihaile- [ϕ Merja]]]]]]
        ϕ° agrees with ϕ ([ϕ Merja]) and values nothing (no useful features available).
        ihaile-° agrees with ϕ ([ϕ Merja]) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON
        v° agrees with ϕ ([ϕ Merja]) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON
        T° agrees with ϕ (<ϕ Pekka>) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON

        Syntax-semantics interface endpoint:
        [<ϕ Pekka>:2 [T [<__>:2 [v [ihaile- [ϕ Merja]]]]]]
```

The information is as follows: head chain reconstruction (lines 75-77), phrasal chain reconstruction when applicable (line 79), agreement reconstruction (lines 80-83) and the endpoint (line 86). If the solution passes, a postsyntactic semantic interpretation follows:

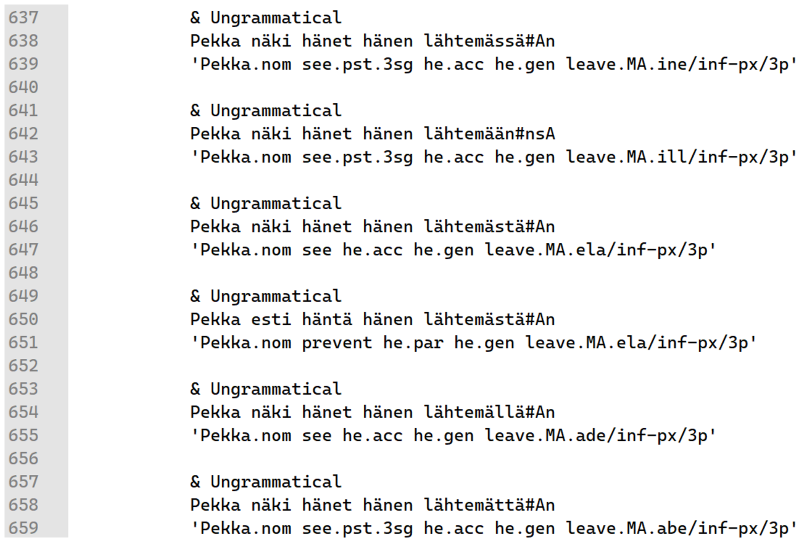


```
85          Syntax-semantics interface endpoint:
86          [<ϕ Pekka>:2 [T [<__>:2 [v [ihaile- [ϕ Merja]]]]]]
87
88          LF-interface and postsyntactic legibility tests:
89          Interpreting TP globally:
90              Establishing argument link for T: <ϕ Pekka>.
91              Project <ϕ Pekka>: (1, QND)(2, GLOBAL)
92              Establishing argument link for v: [ϕ Merja].
93              Establishing argument link for ihaile-: [ϕ Merja].
94              Project [ϕ Merja]: (3, QND)(4, GLOBAL)
95          Denotations:
96              [ϕ Pekka]~['2']
97              [ϕ Merja]~['4']
98          Assignments:
99              Assignment {'1': '2', '3': '4'}: Accepted.
100         Calculating information structure...{'Marked topics': ['<ϕ Pekka>'], 'Neutral gradient': ['[ϕ Merja]'], 'Marked focus': []}
101         Accepted.++
102         Solution accepted at 1346ms stimulus onset.
```

The output must be verified against a gold standard provided by native speakers and/or linguists (here the author and, in few cases, several informants).

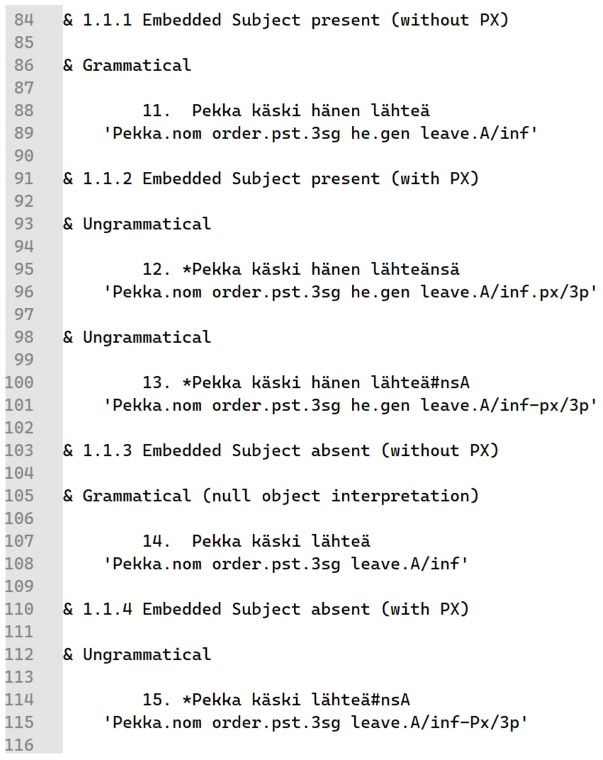
3.4 Stimuli

The test corpus contained a list of Finnish sentences that were crafted by crossing the syntactic variables targeted for examination and then by adding a few additional test cases to test word order variations, binding and the participle adjectives. The final dataset contained 263 constructions. Each input sentence, which were linear sequences of untagged phonological words, was normalized with capitals and punctuation removed. Since one of the syntactic variables tested the presence of infinitival agreement, agreement suffixes were added to the input words as explicit suffixes. This allows the researcher to verify that impossible agreement configurations are correctly judged as ungrammatical by the model. Below is a screenshot from the test corpus file.



```
& Ungrammatical
Pekka näki hänet hänen lähtemässä#An
'Pekka.nom see.pst.3sg he.acc he.gen leave.MA.ine/inf-px/3p'

& Ungrammatical
Pekka näki hänet hänen lähtemään#nsA
'Pekka.nom see.pst.3sg he.acc he.gen leave.MA.ill/inf-px/3p'

& Ungrammatical
Pekka näki hänet hänen lähtemästä#An
'Pekka.nom see he.acc he.gen leave.MA.ela/inf-px/3p'

& Ungrammatical
Pekka esti häntä hänen lähtemästä#An
'Pekka.nom prevent he.par he.gen leave.MA.ela/inf-px/3p'

& Ungrammatical
Pekka näki hänet hänen lähtemällä#An
'Pekka.nom see he.acc he.gen leave.MA.ade/inf-px/3p'

& Ungrammatical
Pekka näki hänet hänen lähtemättä#An
'Pekka.nom see.pst.3sg he.acc he.gen leave.MA.abe/inf-px/3p'
```

Each input sentence is paired with a grammaticality judgment. Below is a screenshot from the output file containing grammaticality judgments calculated by the model. * = the sentence was judged ungrammatical.

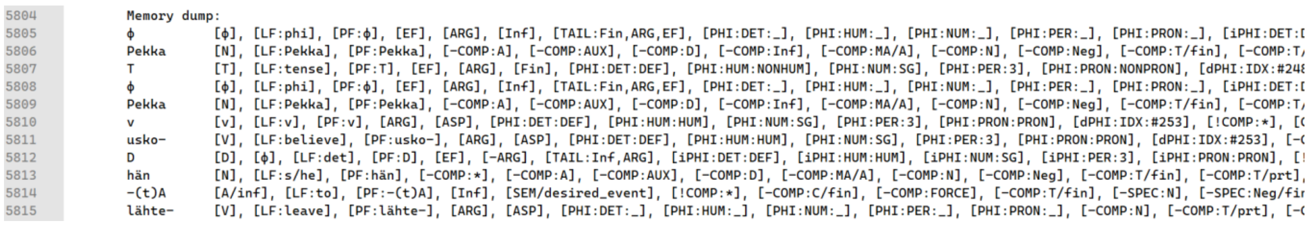


```
& 1.1.1 Embedded Subject present (without PX)

& Grammatical

        11.  Pekka käski hänen lähteä
    'Pekka.nom order.pst.3sg he.gen leave.A/inf'

& 1.1.2 Embedded Subject present (with PX)

& Ungrammatical

        12. *Pekka käski hänen lähteänsä
    'Pekka.nom order.pst.3sg he.gen leave.A/inf.px/3p'

& Ungrammatical

        13. *Pekka käski hänen lähteä#nsA
    'Pekka.nom order.pst.3sg he.gen leave.A/inf-px/3p'

& 1.1.3 Embedded Subject absent (without PX)

& Grammatical (null object interpretation)

        14.  Pekka käski lähteä
    'Pekka.nom order.pst.3sg leave.A/inf'

& 1.1.4 Embedded Subject absent (with PX)

& Ungrammatical

        15. *Pekka käski lähteä#nsA
    'Pekka.nom order.pst.3sg leave.A/inf-Px/3p'
```

The file contains the list of input sentences from the original test corpus, where each sentence is judged to be either grammatical (no prefix) or ungrammatical (prefixed with *). Comment sentences prefixed with & are carried from the test corpus file into this output file to facilitate readability. Labels “grammatical” and “ungrammatical” make the verification easier, especially for non-native speakers. In addition to the grammaticality judgments provided by the model, there is an identical file which contains the grammaticality judgments provided by the native speaker. The results can be verified then by comparing the two files by a mechanical file comparison tool. This can be performed without any knowledge of the target language. Finally, the algorithm matches its own judgments with the labels “grammatical” and “ungrammatical” provided by the user and provides a list of mismatches into a separate file.

The syntactic analyses are also provided in the form of phrase structure trees. Every input sentence the model judges grammatical is provided with an image of the phrase structure representation as it was generated after transfer at the syntax-semantics interface. The phrase structure images are collected into a folder */phrase structure images* and labelled by numeral identifiers provided by the algorithm. If the input sentence is ambiguous, all solutions are provided and are separate by using alphabetical letters. The researcher can control to some extent the information that is present in the phrase structure images. For example, it is possible to control the lexical features printed out in connection with the lexical items.

When interpreting of the phrase structure images it is important to be aware of the way the labels are generated. Labels (representing e.g. major lexical categories) are simple features. When the algorithm draws the image, it uses a specifically designated features as representing the “major lexical labels” that are in turn defined (currently) in the source code. Because any given lexical item may contain several features representing what we would intuitively consider a major lexical category, sometimes a choice must be made. This is currently handled by organizing the major lexical category labels into a list, and the first item from the list that matches with a feature inside a lexical item is selected. Thus, all infinitival heads are labelled as Inf in the images printed out in the present study, because this feature occurs before any of the more specific infinitival labels in that list (i.e., N...V...Inf...VA/inf...). This choice has no impact on anything else, and is made for convenience. Had we inserted the specific infinitival labels before Inf in the list, then those labels would occur in the imaged. To find out what features a given lexical item had at the end of the derivation, we can consult the memory dumb segment in the derivational log file, shown below:



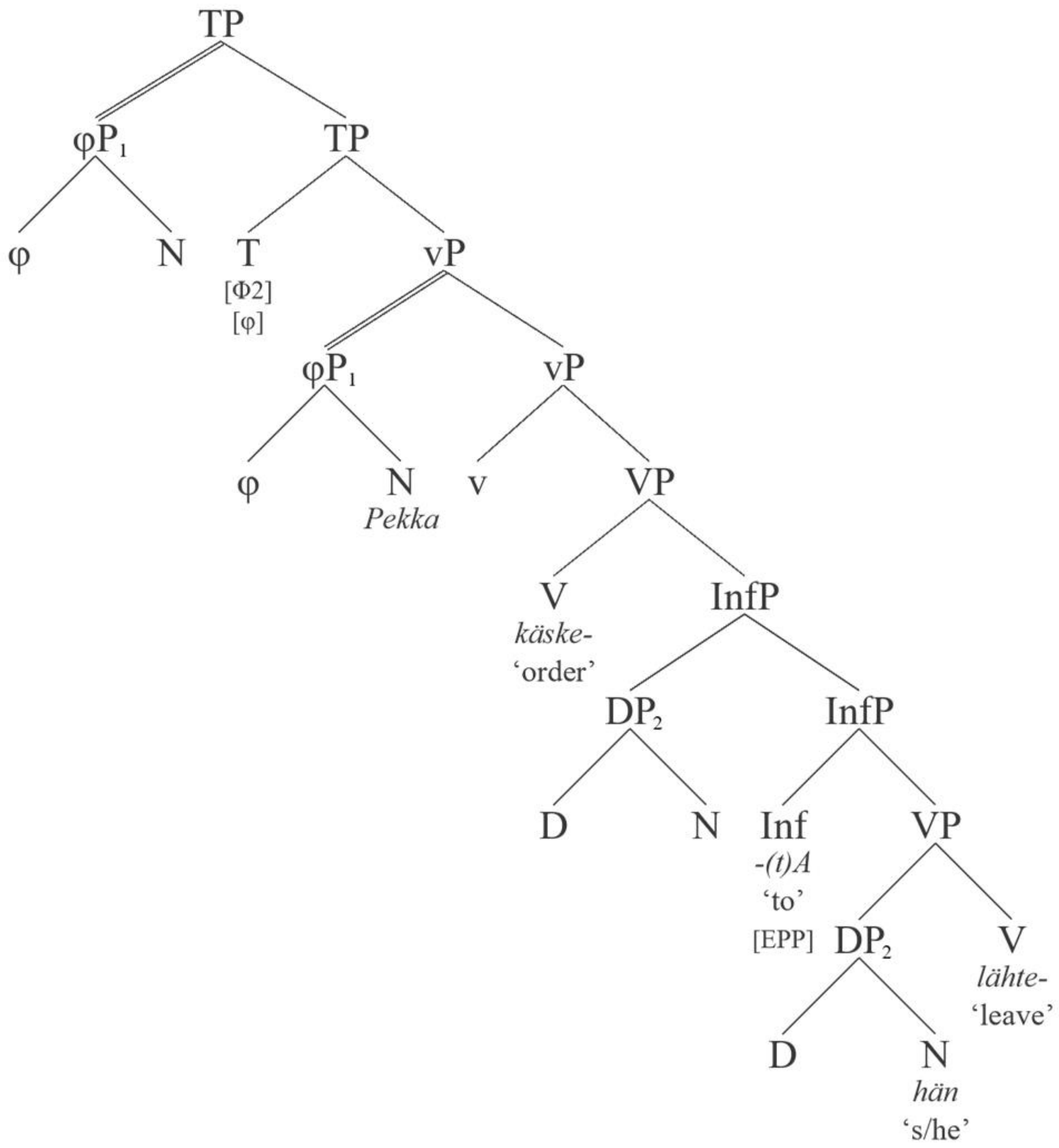
```
Memory dump:
ϕ          [ϕ], [LF:phi], [PF:ϕ], [EF], [ARG], [Inf], [TAIL:Fin,ARG,EF], [PHI:DET:_], [PHI:HUM:_], [PHI:NUM:_], [PHI:PER:_], [PHI:PRON:_], [iPHI:DET:
Pekka      [N], [LF:Pekka], [PF:Pekka], [-COMP:A], [-COMP:AUX], [-COMP:D], [-COMP:Inf], [-COMP:MA/A], [-COMP:N], [-COMP:Neg], [-COMP:T/fin], [-COMP:T
T          [T], [LF:tense], [PF:T], [EF], [ARG], [Fin], [PHI:DET:DEF], [PHI:HUM:NONHUM], [PHI:NUM:SG], [PHI:PER:3], [PHI:PRON:NONPRON], [dPHI:IDX:#24
ϕ          [ϕ], [LF:phi], [PF:ϕ], [EF], [ARG], [Inf], [TAIL:Fin,ARG,EF], [PHI:DET:_], [PHI:HUM:_], [PHI:NUM:_], [PHI:PER:_], [PHI:PRON:_], [iPHI:DET:
Pekka      [N], [LF:Pekka], [PF:Pekka], [-COMP:A], [-COMP:AUX], [-COMP:D], [-COMP:Inf], [-COMP:MA/A], [-COMP:N], [-COMP:Neg], [-COMP:T/fin], [-COMP:T
v          [v], [LF:v], [PF:v], [ARG], [ASP], [PHI:DET:DEF], [PHI:HUM:HUM], [PHI:NUM:SG], [PHI:PER:3], [PHI:PRON:PRON], [dPHI:IDX:#253], [!COMP:*], [
usko-      [V], [LF:believe], [PF:usko-], [ARG], [ASP], [PHI:DET:DEF], [PHI:HUM:HUM], [PHI:NUM:SG], [PHI:PER:3], [PHI:PRON:PRON], [dPHI:IDX:#253], [-
D          [D], [ϕ], [LF:det], [PF:D], [EF], [-ARG], [TAIL:Inf,ARG], [iPHI:DET:DEF], [iPHI:HUM:HUM], [iPHI:NUM:SG], [iPHI:PER:3], [iPHI:PRON:PRON], [!
hän        [N], [LF:s/he], [PF:hän], [-COMP:*], [-COMP:A], [-COMP:AUX], [-COMP:D], [-COMP:MA/A], [-COMP:N], [-COMP:Neg], [-COMP:T/fin], [-COMP:T/prt],
-(t)A      [A/inf], [LF:to], [PF:-(t)A], [Inf], [SEM/desired_event], [!COMP:*], [-COMP:C/fin], [-COMP:FORCE], [-COMP:T/fin], [-SPEC:N], [-SPEC:Neg/fi
lähte-     [V], [LF:leave], [PF:lähte-], [ARG], [ASP], [PHI:DET:_], [PHI:HUM:_], [PHI:NUM:_], [PHI:PER:_], [PHI:PRON:_], [-COMP:N], [-COMP:T/prt], [-
```

This shows each lexical item present in the final syntax-semantics interface object at the LF-interface (left) and their features (right).

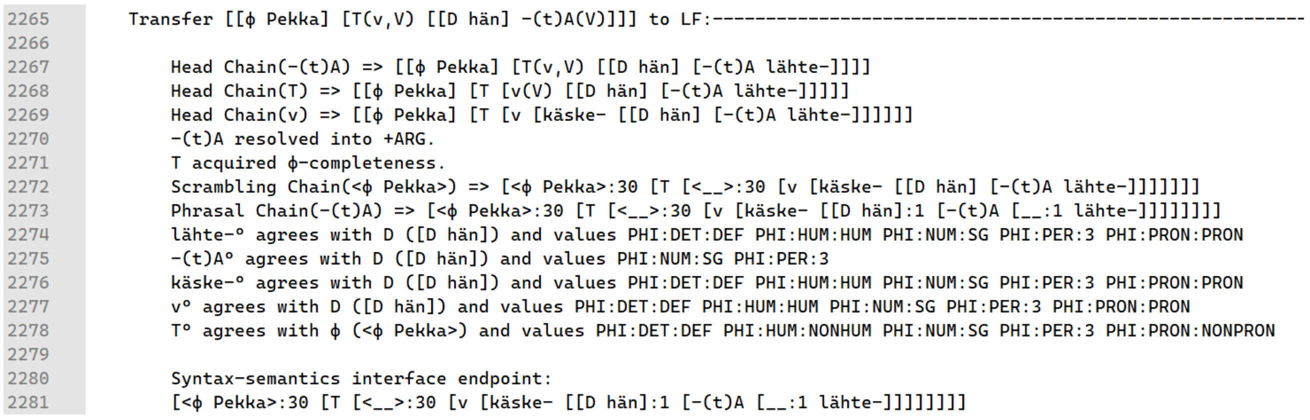
4 Results

4.1 A-infinitive

The A-infinitives are represented by sentences #11-31 in the test corpus. The first sentence in this group is *Pekka käski hänen lähteä* 'Pekka ordered he.GEN leave-A/INF' which the algorithm (correctly) judges as grammatical and analyses as shown below.

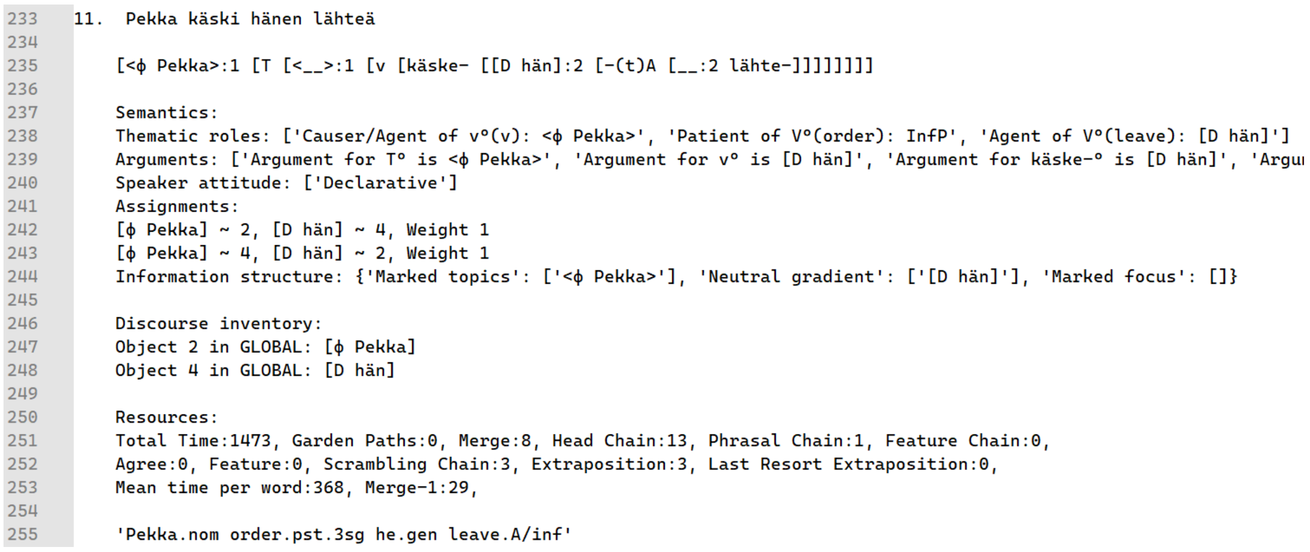


The structure contains a main clause and a nonfinite clause, where the former selects the latter under configuration [$_{VP}$ V^0 [$_{A/infP}$ A/inf^0 VP]] 'order him to leave'. The model merges the embedded subject into a position that precedes the infinitival predicate on the basis of the surface string (*hänen* + *lähteä*) and then reconstructs it to the thematic position inside the VP. Reconstruction takes place during transfer, shown below:



```
2265  Transfer [[ϕ Pekka] [T(v,V) [[D hän] -(t)A(V)]]] to LF:-------------------------------------------------------
2266
2267      Head Chain(-(t)A) => [[ϕ Pekka] [T(v,V) [[D hän] [-(t)A lähte-]]]]
2268      Head Chain(T) => [[ϕ Pekka] [T [v(V) [[D hän] [-(t)A lähte-]]]]]
2269      Head Chain(v) => [[ϕ Pekka] [T [v [käske- [[D hän] [-(t)A lähte-]]]]]]
2270      -(t)A resolved into +ARG.
2271      T acquired ϕ-completeness.
2272      Scrambling Chain(<ϕ Pekka>) => [<ϕ Pekka>:30 [T [<__>:30 [v [käske- [[D hän] [-(t)A lähte-]]]]]]]
2273      Phrasal Chain(-(t)A) => [<ϕ Pekka>:30 [T [<__>:30 [v [käske- [[D hän]:1 [-(t)A [__:1 lähte-]]]]]]]]
2274      lähte-° agrees with D ([D hän]) and values PHI:DET:DEF PHI:HUM:HUM PHI:NUM:SG PHI:PER:3 PHI:PRON:PRON
2275      -(t)A° agrees with D ([D hän]) and values PHI:NUM:SG PHI:PER:3
2276      käske-° agrees with D ([D hän]) and values PHI:DET:DEF PHI:HUM:HUM PHI:NUM:SG PHI:PER:3 PHI:PRON:PRON
2277      v° agrees with D ([D hän]) and values PHI:DET:DEF PHI:HUM:HUM PHI:NUM:SG PHI:PER:3 PHI:PRON:PRON
2278      T° agrees with ϕ (<ϕ Pekka>) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON
2279
2280      Syntax-semantics interface endpoint:
2281      [<ϕ Pekka>:30 [T [<__>:30 [v [käske- [[D hän]:1 [-(t)A [__:1 lähte-]]]]]]]]
```

The first-pass parse is shown on line 2265. The two predicates *käski* ‘order-PST.3SG’ and *lähteä* ‘leave-A/INF’ are decomposed in the lexico-morphological component and expanded into syntactic structure by head reconstruction T(v,V)^0 ~ T...v...V and VA/inf(V)^0 ~ VA/Inf...V (lines 2267-2269). The preverbal subject is reconstructed from SpecTP into SpecvP to pair the topic with the thematic role of agent (Chain 1)(line 2272). The embedded subject of the infinitive is first merged left of the infinitival predicate *hänen lähte-ä* ‘he.GEN leave-A/INF’ and is then reconstructed into SpecVP where it receives the thematic role (chain 2)(line 2273). The predicted thematic roles are available in the results file, line 238, which says that the agent of ‘order’ was Pekka, the agent of leaving was ‘he’ and the patient of ‘order’ was the nonfinite clause InfP ‘he to leave’.

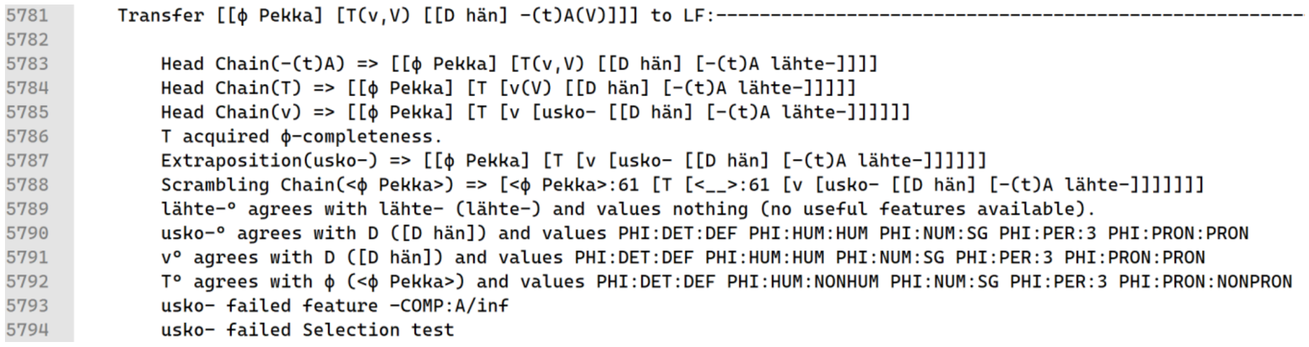


```
11.  Pekka käski hänen lähteä

     [<ϕ Pekka>:1 [T [<__>:1 [v [käske- [[D hän]:2 [-(t)A [__:2 lähte-]]]]]]]

     Semantics:
     Thematic roles: ['Causer/Agent of v°(v): <ϕ Pekka>', 'Patient of V°(order): InfP', 'Agent of V°(leave): [D hän]']
     Arguments: ['Argument for T° is <ϕ Pekka>', 'Argument for v° is [D hän]', 'Argument for käske-° is [D hän]', 'Argu
     Speaker attitude: ['Declarative']
     Assignments:
     [ϕ Pekka] ~ 2, [D hän] ~ 4, Weight 1
     [ϕ Pekka] ~ 4, [D hän] ~ 2, Weight 1
     Information structure: {'Marked topics': ['<ϕ Pekka>'], 'Neutral gradient': ['[D hän]'], 'Marked focus': []}

     Discourse inventory:
     Object 2 in GLOBAL: [ϕ Pekka]
     Object 4 in GLOBAL: [D hän]

     Resources:
     Total Time:1473, Garden Paths:0, Merge:8, Head Chain:13, Phrasal Chain:1, Feature Chain:0,
     Agree:0, Feature:0, Scrambling Chain:3, Extraposition:3, Last Resort Extraposition:0,
     Mean time per word:368, Merge-1:29,

     'Pekka.nom order.pst.3sg he.gen leave.A/inf'
```

The main verb *käske-* 'order' was endowed with a feature which requires that it selects a clause with the semantic property 'desired event' which characterizes (by assumption made in this study) the A-infinitival predicates in Finnish. An alternative is to select the A-infinitival directly as a syntactic class or for some other semantic property. The matter must be decided by evaluation the alternatives against relevant datasets. Verbs in the belief-class are unable to select for the A-infinitival complement clauses due to the negative selection feature –COMP:A/INF.

(11) a.	*Pekka	uskoi	(hänen)	lähte-ä(-nsä). (#20, variations #21-23).
	Pekka	believed	he.GEN	leave-A/INF(-3SG)

The negative selection feature rejects the configuration [$_{VP}$ believe A/infP] at the LF-interface, as shown by lines 5793-5794.



```
Transfer [[ϕ Pekka] [T(v,V) [[D hän] -(t)A(V)]]] to LF:-----------------------------------------------------

    Head Chain(-(t)A) => [[ϕ Pekka] [T(v,V) [[D hän] [-(t)A lähte-]]]]
    Head Chain(T) => [[ϕ Pekka] [T [v(V) [[D hän] [-(t)A lähte-]]]]]
    Head Chain(v) => [[ϕ Pekka] [T [v [usko- [[D hän] [-(t)A lähte-]]]]]]
    T acquired ϕ-completeness.
    Extraposition(usko-) => [[ϕ Pekka] [T [v [usko- [[D hän] [-(t)A lähte-]]]]]]
    Scrambling Chain(<ϕ Pekka>) => [<ϕ Pekka>:61 [T [<__>:61 [v [usko- [[D hän] [-(t)A lähte-]]]]]]]
    lähte-° agrees with lähte- (lähte-) and values nothing (no useful features available).
    usko-° agrees with D ([D hän]) and values PHI:DET:DEF PHI:HUM:HUM PHI:NUM:SG PHI:PER:3 PHI:PRON:PRON
    v° agrees with D ([D hän]) and values PHI:DET:DEF PHI:HUM:HUM PHI:NUM:SG PHI:PER:3 PHI:PRON:PRON
    T° agrees with ϕ (<ϕ Pekka>) and values PHI:DET:DEF PHI:HUM:NONHUM PHI:NUM:SG PHI:PER:3 PHI:PRON:NONPRON
    usko- failed feature -COMP:A/inf
    usko- failed Selection test
```

Selection dependencies are calculated as head-to-head dependencies, since labels are determined dynamically by the labelling algorithm and are not properties of complex constituents (Section 2.3). The A-infinitival head is marked for –ΦPF which causes it to reject agreement. This rules out (12)a, b where the agreement was forced into the predicate by a morphological decomposition in the input. Example (12)b is ruled out because the surface lexicon has no entry for *lähteänsä* ‘leave.A/INF.3SG’.

(12)	a.	*Pekka	käski	hänen	lähte-ä-nsä. (#12)
		Pekka	ordered	he.GEN	leave-A/INF-3SG
	b.	*Pekka	käski	hänen	lähteänsä. (#13)
		Pekka	ordered	he.GEN	leave.A/INF.3SG
	c.	*Pekka	käski	lähte-ä-nsä. (#15)	
		Pekka	ordered	leave-A/INF-3SG	

Examples (12)b–c merit a further comment. This sentence is ruled out because the surface word *lähteänsä* cannot be found from the lexicon. If all agreement errors were rejected on such basis, then the model would rule out (b–c) as a matter of lexical irregularity and, thus, it could be possible for *some* A-infinitives to acquire agreement. The fact that all A-infinitives lacks agreement suggests that the property characterizes the whole infinitival agreement paradigm and is not due to lexical irregularity. This leaves us still with several possible analysis. One possibility is that the combination of the A-infinitive and agreement is ruled out on the basis of an independent morphological rule.

Autonomous morphological rules do not exist in the current implementation, however.[15] The pattern was captured by positing a lexical feature –ΦPF which blocks agreement. The agreement suffixes at the agreeing A-infinitive (12)a are first inserted inside the infinitival head, but the construction is rejected at the LF interface.

The combination of the main verb ‘order’ and the A-infinitival results in ungrammaticality if the phrasal subject is absent (13).

(13)	*Pekka	käski	lähte-ä. (#14)
	Pekka	ordered	leave-A/INF

The requirement that a subject must be present is regulated by EPP. The presence of EPP depends on the selecting verb. If the main verb is ‘want’, then an overt subject is mandatory (14).

(14)	a.	Pekka	halusi	*(hänen)	lähte-ä. (#16, 18, 11)
		Pekka	wanted	he.GEN	leave-A/INF
	b.	*Pekka	halusi	hänen	lähte-ä-nsä. (#17)
		Pekka	wanted	he.GEN	leave-A/INF-3SG
	c.	*Pekka	halusi	lähte-ä-nsä (#18)	
		Pekka	wanted	leave-A/INF-3SG	

[15] Whether autonomous morphological rules exist is nontrivial because many morphological patterns can be derived by syntax. Consider a possible and, in my view, also plausible morphological principle which rules out words such as **vänlähte* ‘VA/INF-leave’ in Finnish. We could rule out the decomposition VA/INF#V in an autonomous morphological component, yet this would be unnecessary since the untested word generates [VP V^0 VA/inf^0] which is ruled out independently by syntactic principles (specifically, selection among the heads).

The A-infinitival head contains a feature ?ARG which, when selected by an anti-OC verbs, is disambiguated during the derivation into a predicate requiring an overt subject.[16]

Let us consider the operation of the EPP which is applied in the case of (13). The EPP is implemented by feature !SELF:Φ1 = EPP (§ 2.8). This condition is violated by (13), because there is neither agreement nor a local phrasal subject present. Since the A-infinitival was marked for –ΦPF, the subject must be identified by a phrasal subject. Control cannot rescue this sentence from the EPP-requirement since it cannot substitute for a local phrasal subject. The negative specification for the EPP explains (14) (#16 in the dataset).

It was assumed that the A-infinitival cannot occur in the adjunct position (15).

(15)	a.	*Pekka	näki	hänet	(hän-en)	lähte-ä(-nsä)(#24-27).
		Pekka	saw	him	he-GEN	leave-A/INF(-3SG)

Adjunction is licensed by +ADV that the A-infinitival head does not have. All parsing solutions generated from the word sequence in (15) lead into ill-formed outcomes. The A-infinitival cannot occur in connection with intransitive verbs (#28-31).

(16)	Pekka	nukkui	lähte-ä. (#30)
	Pekka	slept	leave-A/INF

This follows from the assumption that the A-infinitives cannot be adjoined, preventing an analysis where the intransitive verb has no complement in (16), and from the fact that an intransitive verb

[16] There are several ways to implement this requirement. The first and the easiest is to assume that the lexicon contains two A-infinitival heads, one with the EPP profile and another without. The incorrect form will be ruled out at the LF interface when the selection properties between the head and the infinitival head do not match. Another is to create a special unambiguous feature that is disambiguated during the derivation. The latter was assumed here.

‘sleep’ cannot have an A-infinitival complement. Therefore, both the complement and adjunct solutions are rejected.

4.2 VA-infinitive

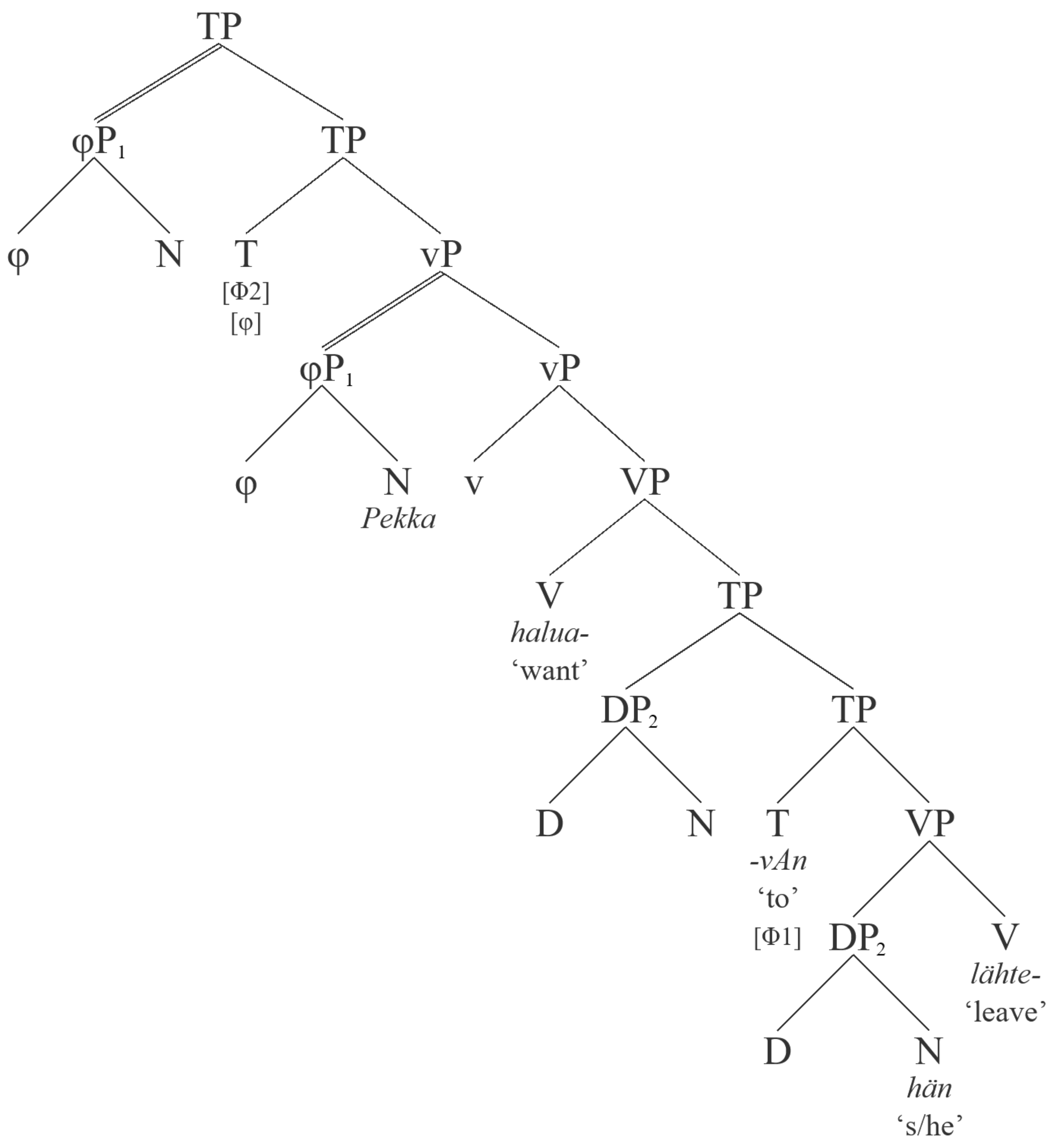
The VA-infinitives are selected complement clauses that do not occur in adjunct positions (17) and are selected by a class of verbs taking propositional complements (18). Selection targets T, which is part of the lexical decomposition of the VA-infinitival predicate.

(17)		*Pekka	näki	hän-et	(hän-en)	lähte-vä(-nsä). (#44, 45, 46, 47)
		Pekka	saw	he-ACC	he.GEN	leave-VA/INF-(3SG)

(18)	a.	*Pekka	käski	(hän-en)	lähtevä(-nsä). (#32-35)
		Pekka	ordered	he-GEN	leave-VA/INF(-3SG)
	b.	Pekka	halusi	hän-en	lähte-vän. (#36)
		Pekka	wanted	he-GEN	leave-VA/INF

Example (18)b is analyzed as shown below, where the agent of the VA-infinitive is the third person ‘s/he’.

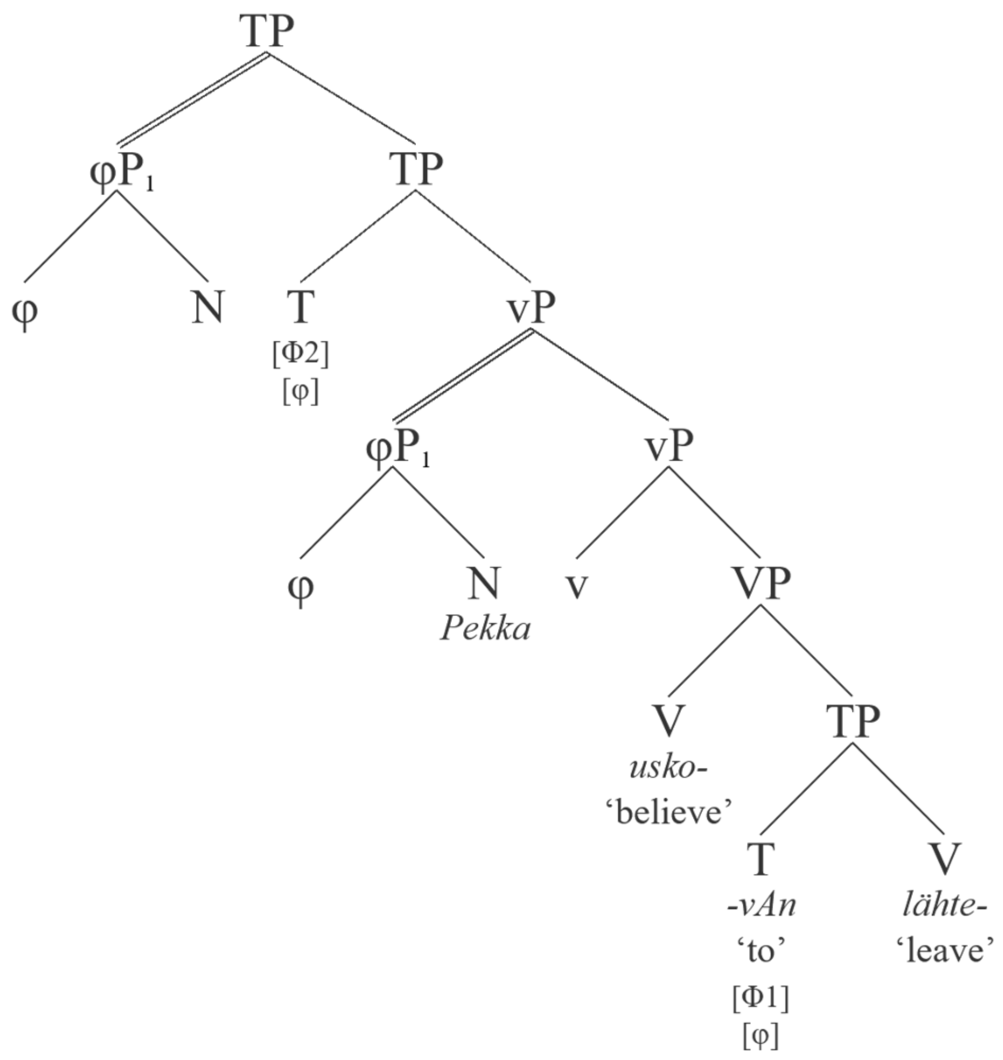
(19)



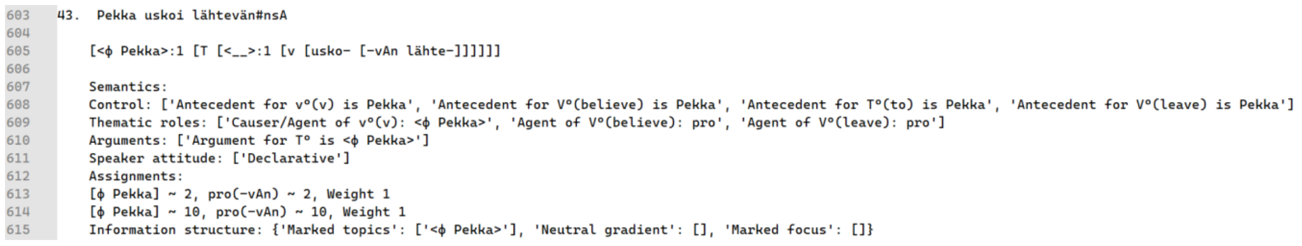
The structure is analogous to the A-infinitival, but differs from it in the sense that while the VA-infinitival has the same EPP property (20)a it can identify the subject both by means of a phrasal subject (b) and agreement (c), but not by both (d).

(20)	a.	*Pekka	halusi	lähte-vän. (#38, 42)	
		Pekka	wanted	leave-VA/INF	
	b.	Pekka	uskoi	hän-en	lähte-vän. (#40)
		Pekka	believed	he-GEN	leave-VA/INF
	c.	Pekka	uskoi	lähte-vä-nsä. (#43)	
		Pekka	believed	leave-VA/INF-3SG	
	d.	*Pekka	halusi	hän-en	lähte-vä-nsä. (#37, 41)
		Pekka	wanted	he-GEN	leave-VA/INF-3SG

Agreement is possible because the VA-infinitival head is not marked for –ΦPF. The co-occurrence of the subject and agreement, sentence (d), is ruled out by Φ1. Sentence (20)c, where the embedded subject is identified by agreement, is analyzed as follows.



There is no subject; rather, the infinitival clause has structure [$_{TP}$ T V] with overt agreement φ at T. The sentence can only mean 'Pekka believed that he (Pekka/*third party) will leave', which is explained by the fact that the third person possessive suffix (-*nsA*) triggers antecedent scanning as shown on line 608 in the results file.



```
603  43.  Pekka uskoi lähtevän#nsA
604
605      [<ϕ Pekka>:1 [T [<__>:1 [v [usko- [-vAn lähte-]]]]]]
606
607      Semantics:
608      Control: ['Antecedent for v°(v) is Pekka', 'Antecedent for V°(believe) is Pekka', 'Antecedent for T°(to) is Pekka', 'Antecedent for V°(leave) is Pekka']
609      Thematic roles: ['Causer/Agent of v°(v): <ϕ Pekka>', 'Agent of V°(believe): pro', 'Agent of V°(leave): pro']
610      Arguments: ['Argument for T° is <ϕ Pekka>']
611      Speaker attitude: ['Declarative']
612      Assignments:
613      [ϕ Pekka] ~ 2, pro(-vAn) ~ 2, Weight 1
614      [ϕ Pekka] ~ 10, pro(-vAn) ~ 10, Weight 1
615      Information structure: {'Marked topics': ['<ϕ Pekka>'], 'Neutral gradient': [], 'Marked focus': []}
```

The VA-infinitival does not occur in connection with intransitive clauses (#48-51).

(21)	*Pekka	nukkui	(hän-en)	lähte-vä(-nsä). (#48, 49, 50)
	Pekka	slept	he-GEN	leave-VA/INF-3SG

This is explained by two factors: intransitive verbs cannot select T by a lexical redundancy rule and the VA-infinitival itself cannot be adjoined. These two conditions rule out all adjunct configurations (#44-47).

4.3 MA-infinitives

Finnish has five different MA-constructions whose morphological form contains the *-mA* morph and a semantic case suffix. They are items #52-159 in the dataset. The MA-infinitives differ from each other in their syntactic and semantic properties, but they also differ from regular PPs. Hence, they were modelled as composites of verb stems and one of the five MA-infinitival heads.

All MA-infinitivals have the negative specification for the EPP property and cannot occur together with a subject irrespective of the selecting verb. Both overt subjects and infinitival agreement were rejected (22), (23).

(22)	a.	*Pekka	käski/halusi/uskoi	hän-en	lähte-mässä(-än). (#52, 57, 72, 77, 93, 98)
		Pekka	ordered/wanted/believed	he-GEN	leave-MA.INE(-3SG)
	b.	*Pekka	käski/halusi/uskoi	hän-en	lähte-mään(-sä). (#53, 58, 73, 78, 94, 99)
		Pekka	ordered/wanted/believed	he-GEN	leave-MA.ILL(-3SG)
	c.	*Pekka	käski/halusi/uskoi	hän-en	lähte-mästä(-än). (#54, 59, 74, 79, 95, 100)
		Pekka	ordered/wanted/believed	he-GEN	leave-MA.ELA(-3SG)
	d.	*Pekka	käski/halusi/uskoi	hän-en	lähte-mällä(-än). (#55, 60, 75, 80, 96, 101)
		Pekka	ordered/wanted/believed	he-GEN	leave-MA.ADE(-3SG)

e. *Pekka käski/halusi/uskoi hän-en lähte-mättä(-än). (#56, 61, 76, 97, 102)
Pekka ordered/wanted/believed he-GEN leave-MA.ABE(-3SG)

(23) a. *Pekka käski/halusi/uskoi lähte-mässä-än. (#67, 88, 108)
Pekka ordered/wanted/believed leave-MA.INE-3SG

b. *Pekka käski/halusi/uskoi lähte-mään-sä. (#68, 89, 109)
Pekka ordered/wanted/believed leave-MA.ILL.3SG

c. *Pekka käski/halusi/uskoi lähte-mästä-än. (#69, 90, 110)
Pekka ordered/wanted/believed leave-MA.INE-3SG

d. *Pekka käski/halusi/uskoi lähte-mällä-än. (#70, 91, 111)
Pekka ordered/wanted/believed leave-MA.ADE-3SG

e. *Pekka käski/halusi/uskoi lähte-mättä-än. (#71, 92, 112)
Pekka ordered/wanted/believed leave-MA.ABE-3SG

The negative EPP specification explains the same pattern when the construction is in an adjunct position (24) and/or is not selected by the superordinate verb (25).

(24) a. *Pekka näki hänet [hän-en lähte-mässä(-än)]. (#113, 119)
Pekka saw him he-GEN leave-MA.INE(-3SG)

b. *Pekka näki hänet [hän-en lähte-mään(-nsä)]. (#114, 120)
Pekka saw him he-GEN leave-MA.ILL(-3SG)

c. *Pekka näki/esti hänet [hän-en lähte-mästä(-än)]. (#115, 116, 121, 122)
Pekka saw/prevent him he-GEN leave-MA.ELA(-3SG)

d. *Pekka näki hänet [hän-en lähte-mällä(-än)]. (#117, 123)
Pekka saw him he-GEN leave-MA.ADE(-3SG)

e. *Pekka näki hänet [hän-en lähte-mättä(-än)]. (#118, 124)
Pekka saw him he-GEN leave-MA.ABE(-3SG)

(25) a. *Pekka nukkui [hän-en lähte-mässä(-än)]. (#139, 144)
Pekka slept he-GEN leave-MA.INE(-3SG)

b. *Pekka nukkui [hän-en lähte-mään(-nsä)]. (#140, 145)
Pekka slept he-GEN leave-MA.ILL(-3SG)

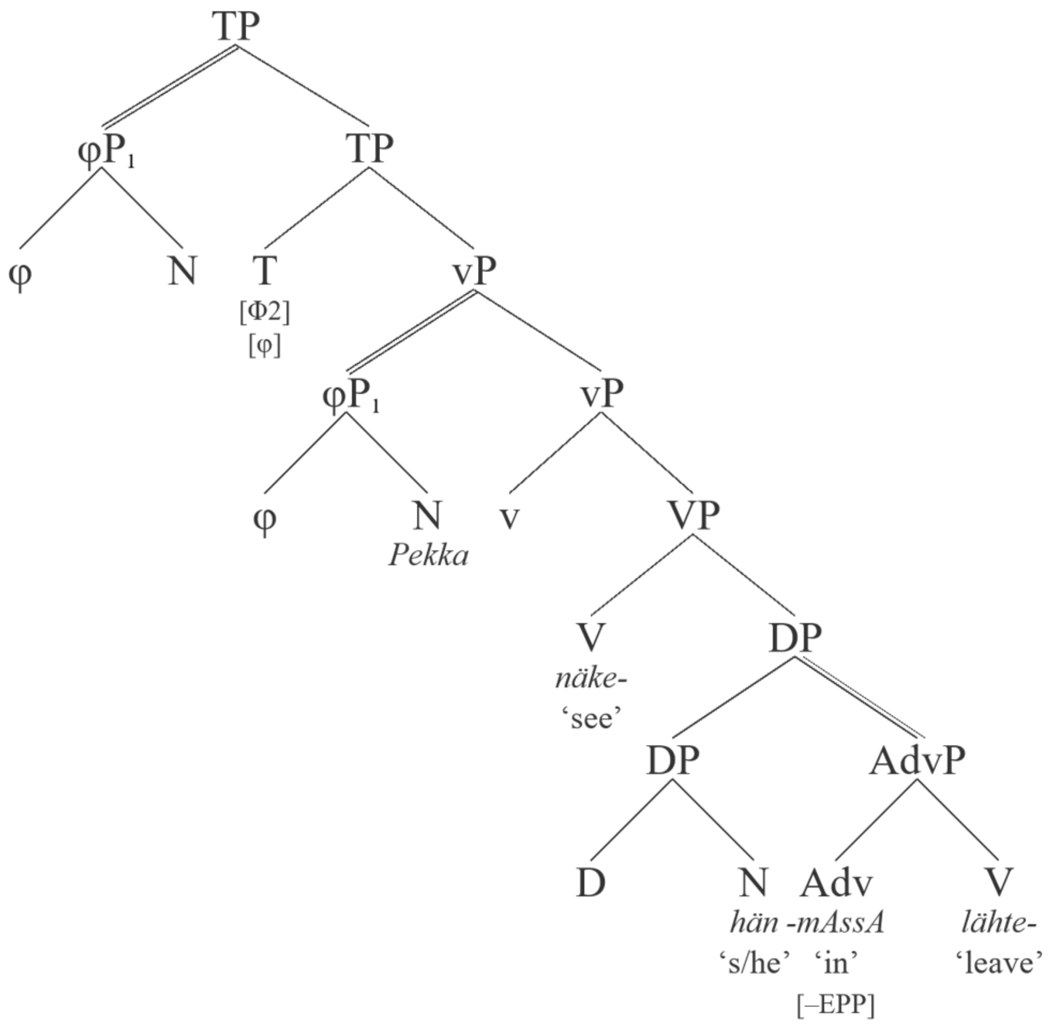
c. *Pekka nukkui hänet [hän-en lähte-mästä(-än)]. (#141, 146)
Pekka slept he-GEN leave-MA.ELA(-3SG)

d. *Pekka nukkui [hän-en lähte-mällä(-än)]. (#141, 146)
Pekka slept he-GEN leave-MA.ADE(-3SG)

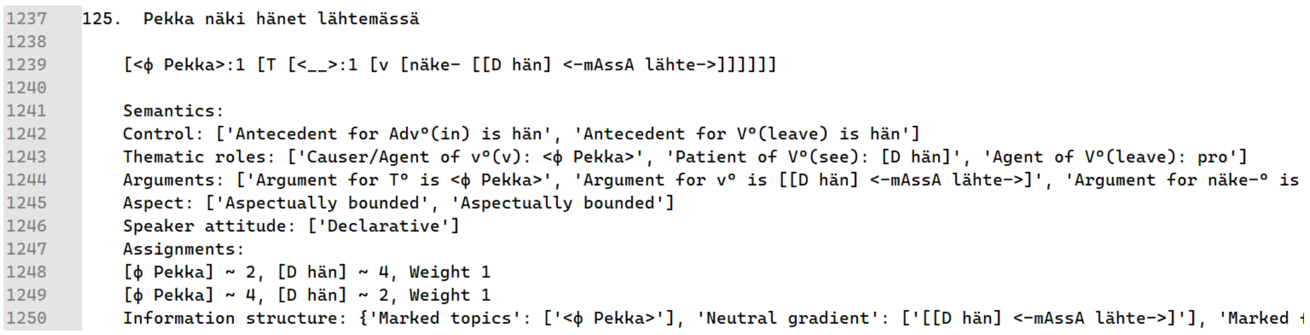
e. *Pekka nukkui [hän-en lähte-mättä(-än)]. (#142, 147)
Pekka slept he-GEN leave-MA.ABE(-3SG)

For the same results for sentences lacking the phrasal subject but with infinitival agreement, see #133-138, #155-160. This leaves us with constructions where the MA-infinitival occurs without subject and agreement. Their subjects are invariable identified by control. Sentence (26) illustrates object control.

(26) Pekka näki hän-et lähte-mässä. (#125)
Pekka saw he-ACC leave-MA.INE/INF
'Pekka saw him leaving.'



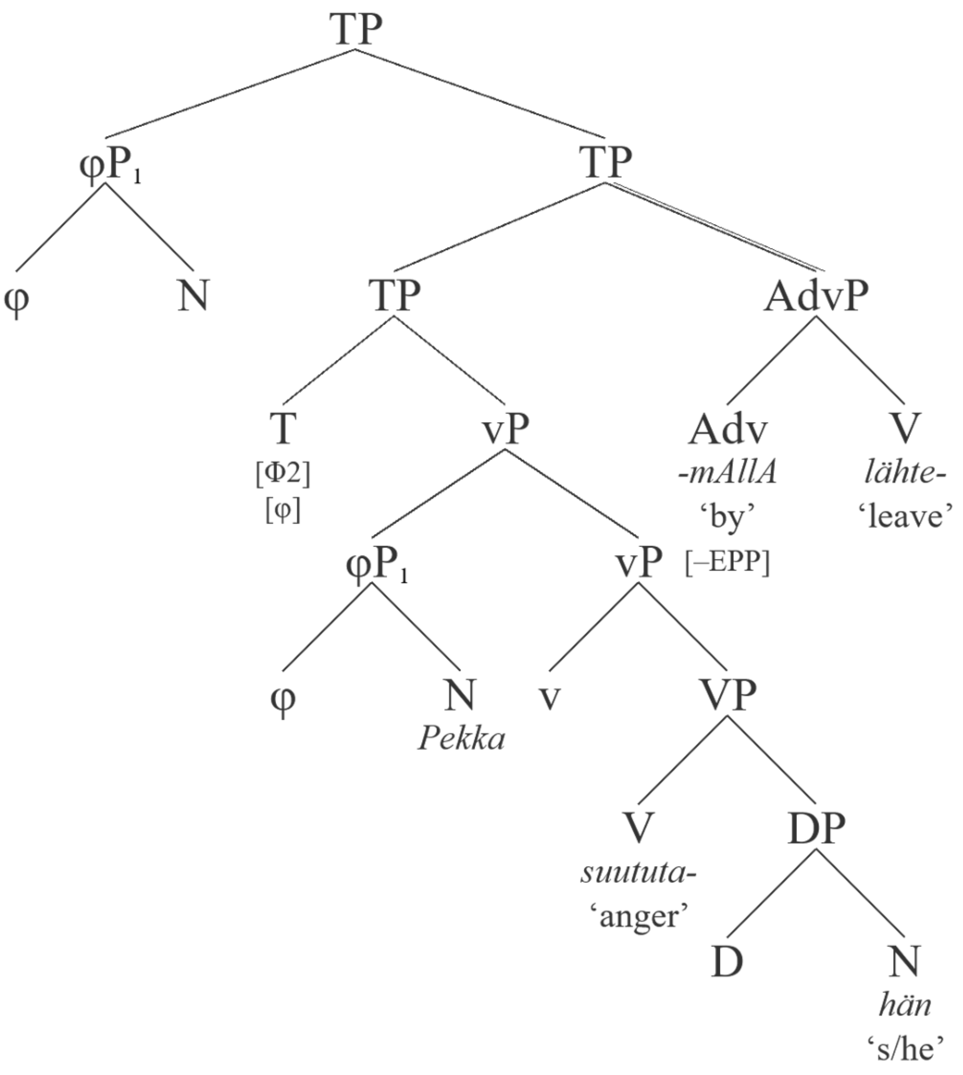
The MA-infinitival is adjoined into a lower position of the clause where the antecedent algorithm finds the object, line 1242:


```
1237  125.  Pekka näki hänet lähtemässä
1238
1239      [<ϕ Pekka>:1 [T [<__>:1 [v [näke- [[D hän] <-mAssA lähte->]]]]]]
1240
1241      Semantics:
1242      Control: ['Antecedent for Adv°(in) is hän', 'Antecedent for V°(leave) is hän']
1243      Thematic roles: ['Causer/Agent of v°(v): <ϕ Pekka>', 'Patient of V°(see): [D hän]', 'Agent of V°(leave): pro']
1244      Arguments: ['Argument for T° is <ϕ Pekka>', 'Argument for v° is [[D hän] <-mAssA lähte->]', 'Argument for näke-° is
1245      Aspect: ['Aspectually bounded', 'Aspectually bounded']
1246      Speaker attitude: ['Declarative']
1247      Assignments:
1248      [ϕ Pekka] ~ 2, [D hän] ~ 4, Weight 1
1249      [ϕ Pekka] ~ 4, [D hän] ~ 2, Weight 1
1250      Information structure: {'Marked topics': ['<ϕ Pekka>'], 'Neutral gradient': ['[[D hän] <-mAssA lähte->]'], 'Marked
```

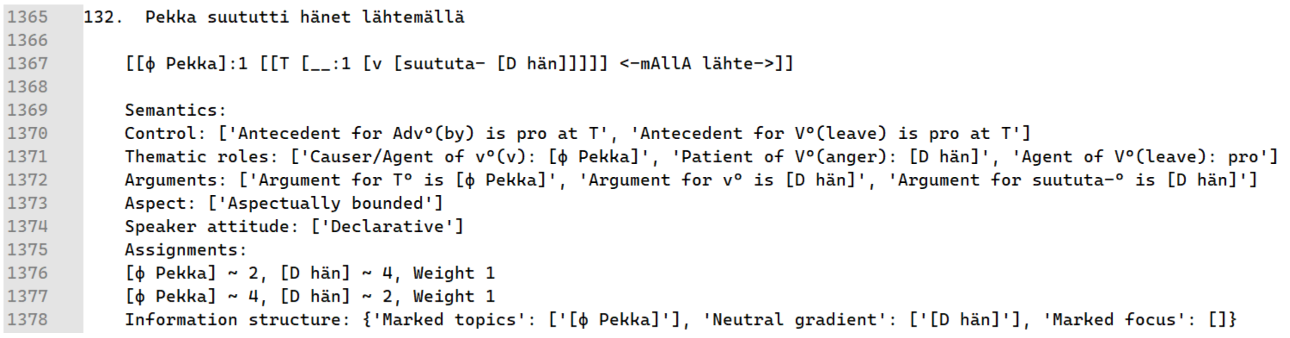
Sentence (27) #132 illustrates subject control.

(27)	Pekka	suututti	hän-et	lähte-mällä.
	Pekka	angered	he-ACC	leave-MA.ADE/INF

‘Pekka angered him by leaving.’



Notice how the MA-infinitive is now adjoined into a higher position, which leads the antecedent algorithm to skip the object, see line 1370. The antecedent is marked as “pro at T,” because we want the same results even if the phrasal subject is absent and because the antecedent algorithm looks only at the head of the phrases it finds by upward search, which is T; the phrasal subject is invisible from the position of the adjoined infinitive. The phrasal subject becomes visible, however, if the infinitive is adjoined to vP inside the parsing path generating the solution.



```
132.  Pekka suututti hänet lähtemällä

    [[ϕ Pekka]:1 [[T [__:1 [v [suututa- [D hän]]]]] <-mAllA lähte->]]

    Semantics:
    Control: ['Antecedent for Adv°(by) is pro at T', 'Antecedent for V°(leave) is pro at T']
    Thematic roles: ['Causer/Agent of v°(v): [ϕ Pekka]', 'Patient of V°(anger): [D hän]', 'Agent of V°(leave): pro']
    Arguments: ['Argument for T° is [ϕ Pekka]', 'Argument for v° is [D hän]', 'Argument for suututa-° is [D hän]']
    Aspect: ['Aspectually bounded']
    Speaker attitude: ['Declarative']
    Assignments:
    [ϕ Pekka] ~ 2, [D hän] ~ 4, Weight 1
    [ϕ Pekka] ~ 4, [D hän] ~ 2, Weight 1
    Information structure: {'Marked topics': ['[ϕ Pekka]'], 'Neutral gradient': ['[D hän]'], 'Marked focus': []}
```

In the following examples, the calculated control dependencies are marked by the subscripts.

(28) a.	Pekka$_1$	näki	hän-et$_2$	lähte-mässä$_{*1,2}$. (#124)	
	Pekka	saw	he-ACC	leave-MA/INF.INE	
b.	Pekka$_1$	pyysi/*näki	häntä$_2$	lähte-mään$_{*1,2}$. (#126, 127)	
	Pekka	asked/saw	he.PAR	leave-MA/INF.ILL	
c.	Pekka$_1$	esti/*näki		häntä$_2$	lähte-mästä$_{*1,2}$. (#128, 129)
	Pekka	prevented/saw		he.PAR	leave-MA/INF.ELA
d.	Pekka$_1$	näki	hänet$_2$	lähte-mällä$_{1, *2}$. (#130)	
	Pekka	saw	he-ACC	leave-MA.INF.ADE	
e.	Pekka$_1$	näki	hänet$_2$	lähte-mättä$_{1, *2}$. (#131)	
	Pekka	saw	he-ACC	leave-MA/INF.ABE	

Some MA-infinitivals, being adjoinable, are also compatible with intransitive verbs. The test sentences are often pragmatically odd; pragmatically felicitous variants are provided in the brackets (29).[17]

[17] These data were originally brought to my attention by an anonymous *NJL* reviewer.

(29) a. ?Pekka nukkui lähte-mässä. (#151) (Koira kuljeskeli kerjäämässä.)

Pekka slept leave-MA/INF.INE (Dog wandered begging)

’Pekka slept while leaving.’

b. ?Pekka nukkui lähte-mällä. (#154) (Pekka lihoi syömällä.)

Pekka slept leave-MA/INF.ADE (Pekka got.weight by.eating)

’Pekka slept by leaving.’

c. ?Pekka nukkui lähte-mättä. (#155) (Pekka lihoi syömättä.)

Pekka slept leave-MA/INF.ABE (Pekka got.weight without.eating)

’Pekka slept without leaving.’

Only the illative variant can occur in a selected position.

(30) a. Pekka käski lähtemään. (#63)

Pekka ordered leave-MA/INF.ILL

‘Pekka ordered people to leave.’

b. *Pekka käski lähtem-ässä/lähte-mästä/lähte-mällä/lähte-mällä. (#62, 64, 65, 66)

Pekka ordered leave-MA/INF.INE/.ELA/.ADE/.ABE

4.4 E-infinitive

The E-infinitive ‘by doing’ creates subjectless subject control adverbials that occur in a high adjunct position (31).

(31) a. Pekka$_1$ näki hänet$_2$ [lähti-en$_{1, *2}$]. (#176)

Pekka saw him leave-E/INF

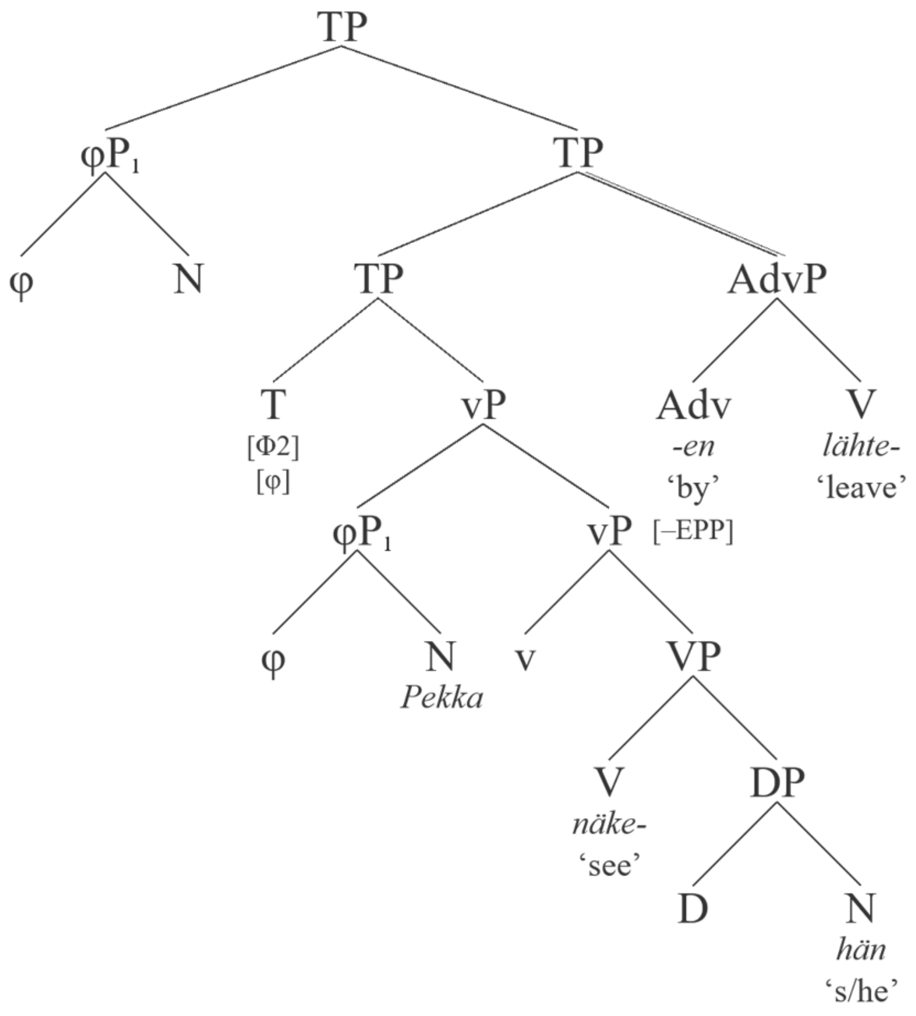
‘Pekka saw him by leaving (/by using a telescope...).’

b. Pekka$_1$ nukkui [lähti-en$_1$]. (#180)

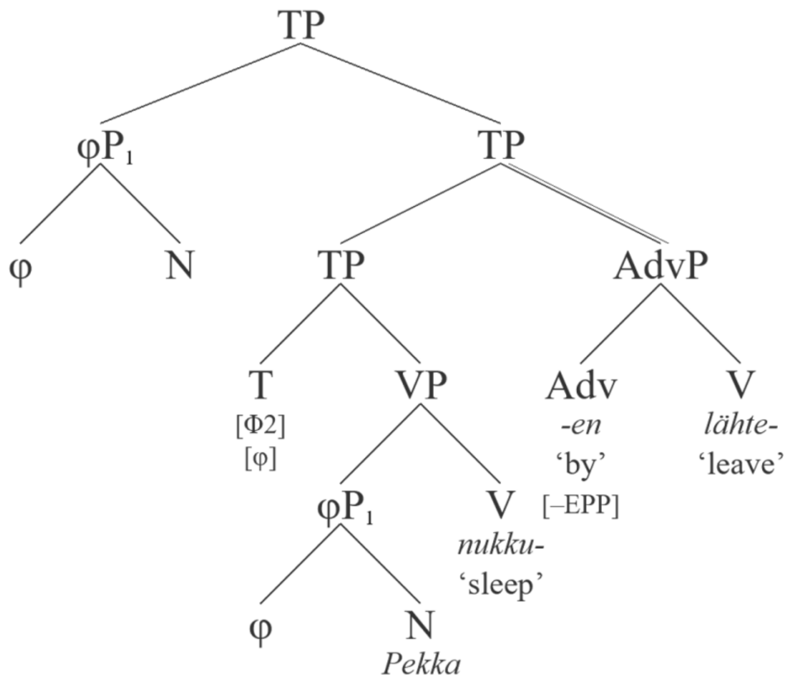
Pekka slept sleep-E/INF

’Pekka slept by leaving (/by snoring...).

The model analyses (31)a as follows:



(32)



The adverbial is adjoined into a high position in the clause, accounting for subject control. Complement positions are ungrammatical (33), since the main verbs used in this study do not select for these infinitives (#162-173).

(33)	a.	*Pekka	käski/halusi	(hänen)	lähti-en(-sä). (#164-169)
		Pekka	ordered/wanted	he-GEN	leave-E/INF(-3SG)
	b.	*Pekka	uskoi	[(hän-en)	lähti-en(-sä)]. (#170-173)
		Pekka	believed	he-GEN	leave-E/INF(-3SG)

E-infinitivals have a negative specification for the EPP property and are ungrammatical when the subject is present (34), independent of whether the clause is in the selected complement position or in an adjunct position (35).

(34)	a.	*Pekka	käski/halusi/uskoi	hän-en	lähti-en(-sä). (#162, 163, 166, 167, 171, 172)
		Pekka	ordered/wanted/believed	he-GEN	leave-E/INF(-3SG)

b. *Pekka käski lähti-en-sä. (#153)
Pekka ordered leave-E/INF-3SG

(35) a. *Pekka näki hänet [hän-en lähti-en(-sä)]. (#174, 175)
Pekka saw him he-GEN leave-E/INF(-3SG)

b. *Pekka näki hänet [lähti-en-sä]. (#177)
Pekka saw him leave-E/ING-3SG

c. *Pekka nukkui [hän-en lähtien(-sä)]. (#178)
Pekka slept he-GEN leave-E/INF(-3SG)

d. *Pekka nukkui [lähti-en-sä]. (#181)
Pekka slept leave-E/INF-3SG

4.5 ESSA-infinitive

The ESSA-infinitive ‘while doing’ occurs in adjunct positions, triggers subject control and has an optional subject (36)b that may be identified either by phrasal subject (a, d) or by infinitival agreement (c, e).

(36) a. Pekka$_1$ näki hänet$_2$ [hänen$_{*1, 2, 3}$ lähti-essä]. (#195)
Pekka saw him he-gen leave-essa/inf
’Pekka saw him while he (≠Pekka) was leaving.’

b. Pekka$_1$ näki hänet$_2$ [lähti-essä$_{1, *2}$]. (#197)
Pekka saw him leave-essa/inf
’Pekka saw him while leaving.’

c. Pekka$_1$ näki hänet$_2$ [lähti-essä-än$_{*1, 2}$]. (#198)
Pekka saw him leave-ESSA/INF(-3SG)
’Pekka saw him while leaving.’

d. Pekka nukkui [hän-en lähti-essä]. (#199)

Pekka slept he-GEN leave-ESSA/INF

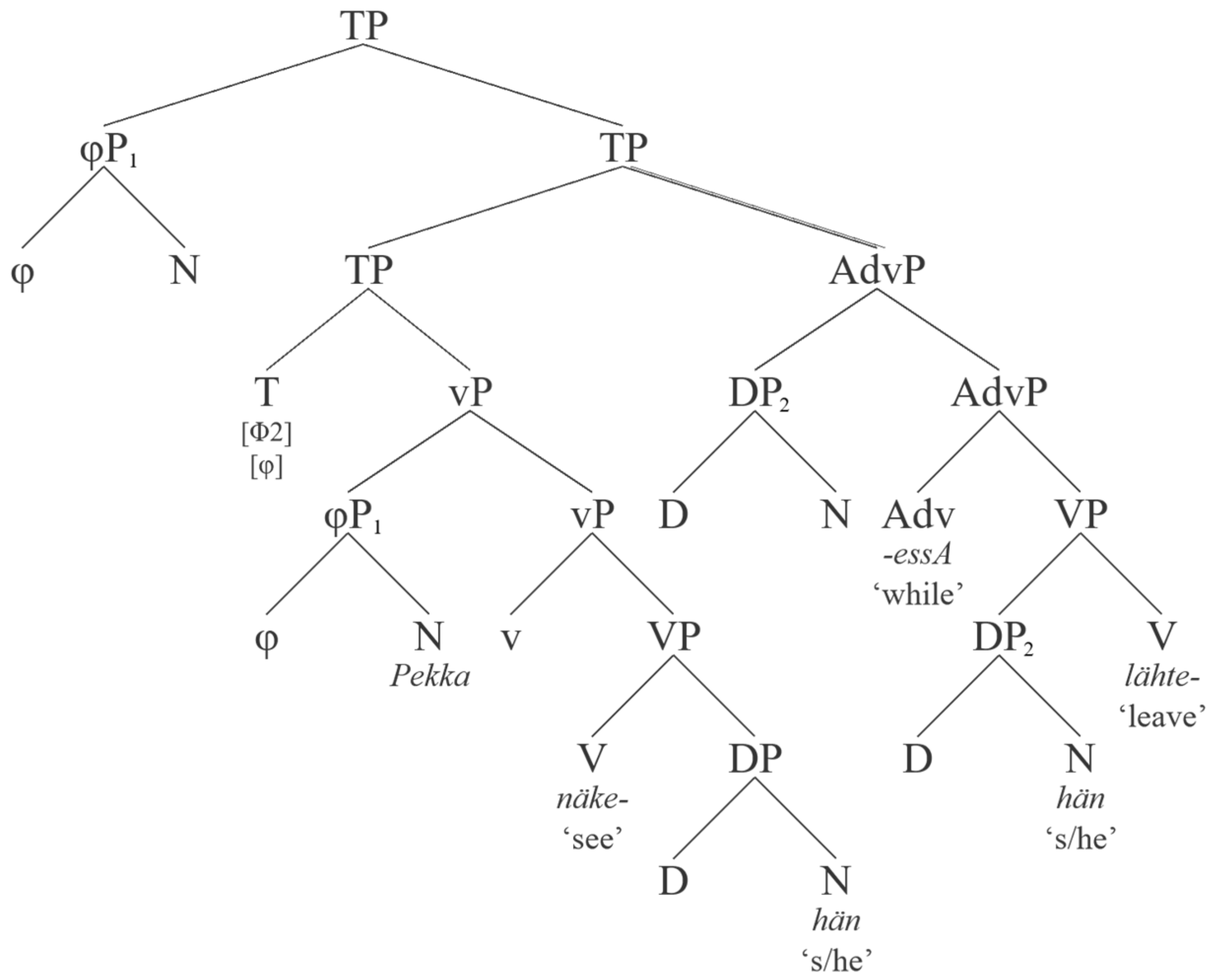
’Pekka slept while he was leaving.’

e. Pekka nukkui [lähti-essä-än]. (#200)

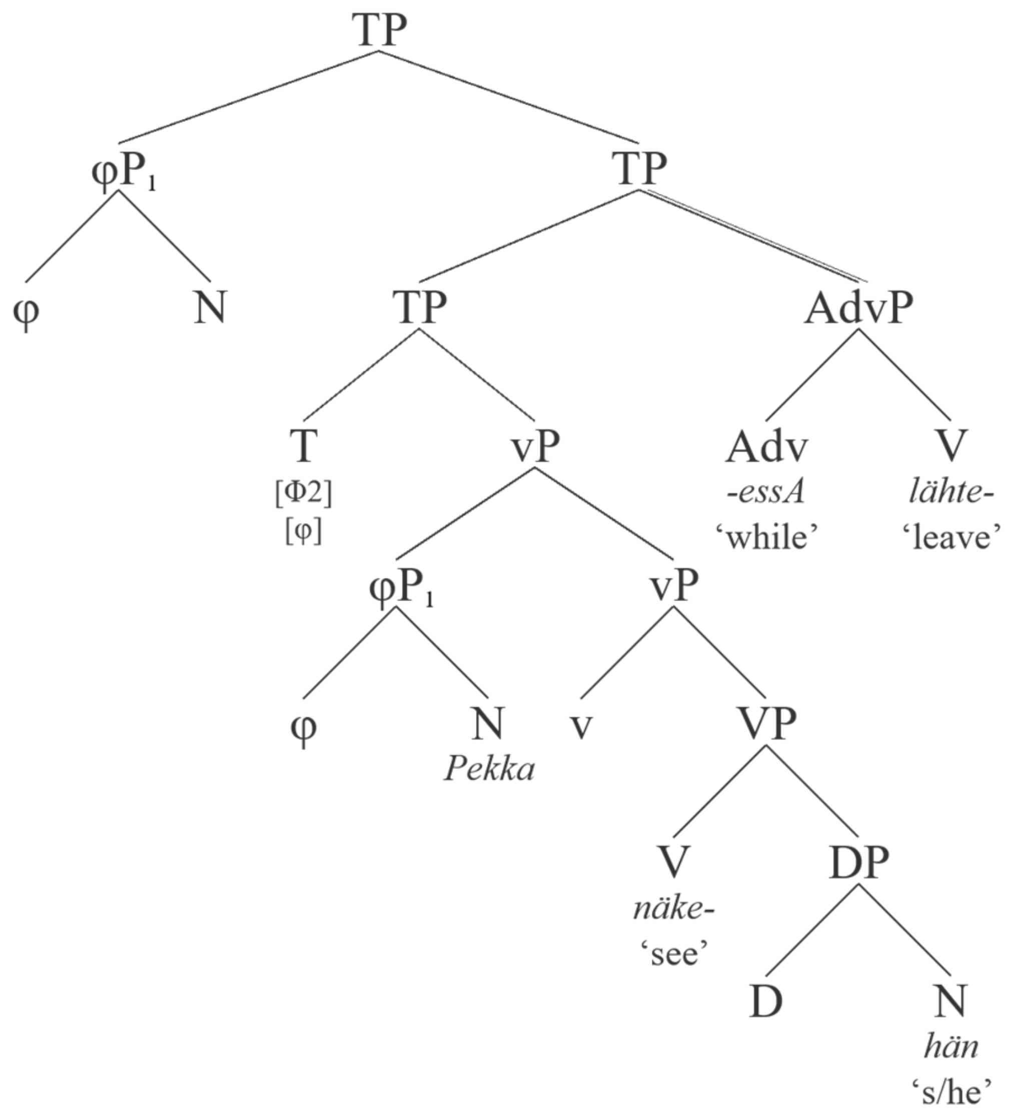
Pekka slept leave-ESSA/INF-3SG

’Pekka slept (/sing...) while he was leaving.’

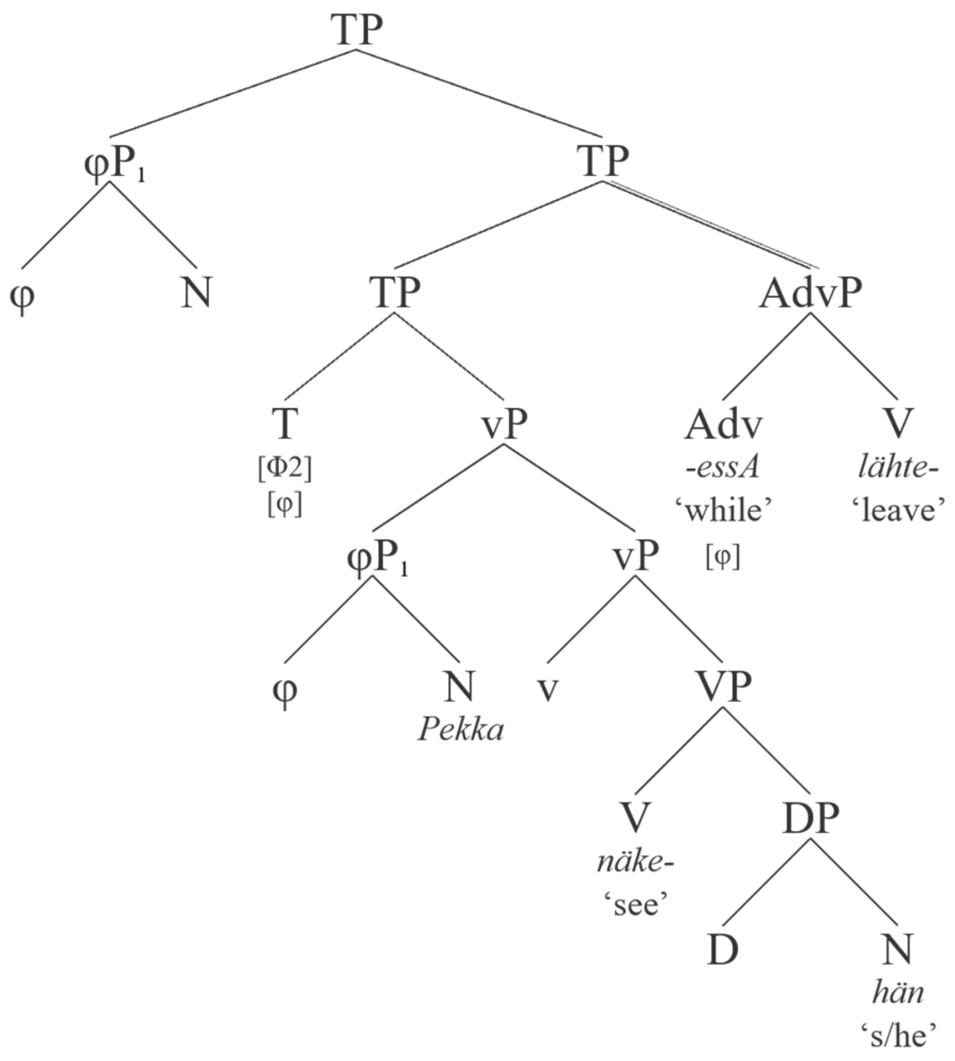
Example (a) is analyzed as shown below.



Removing the overt phrasal subject from the input sentence results in



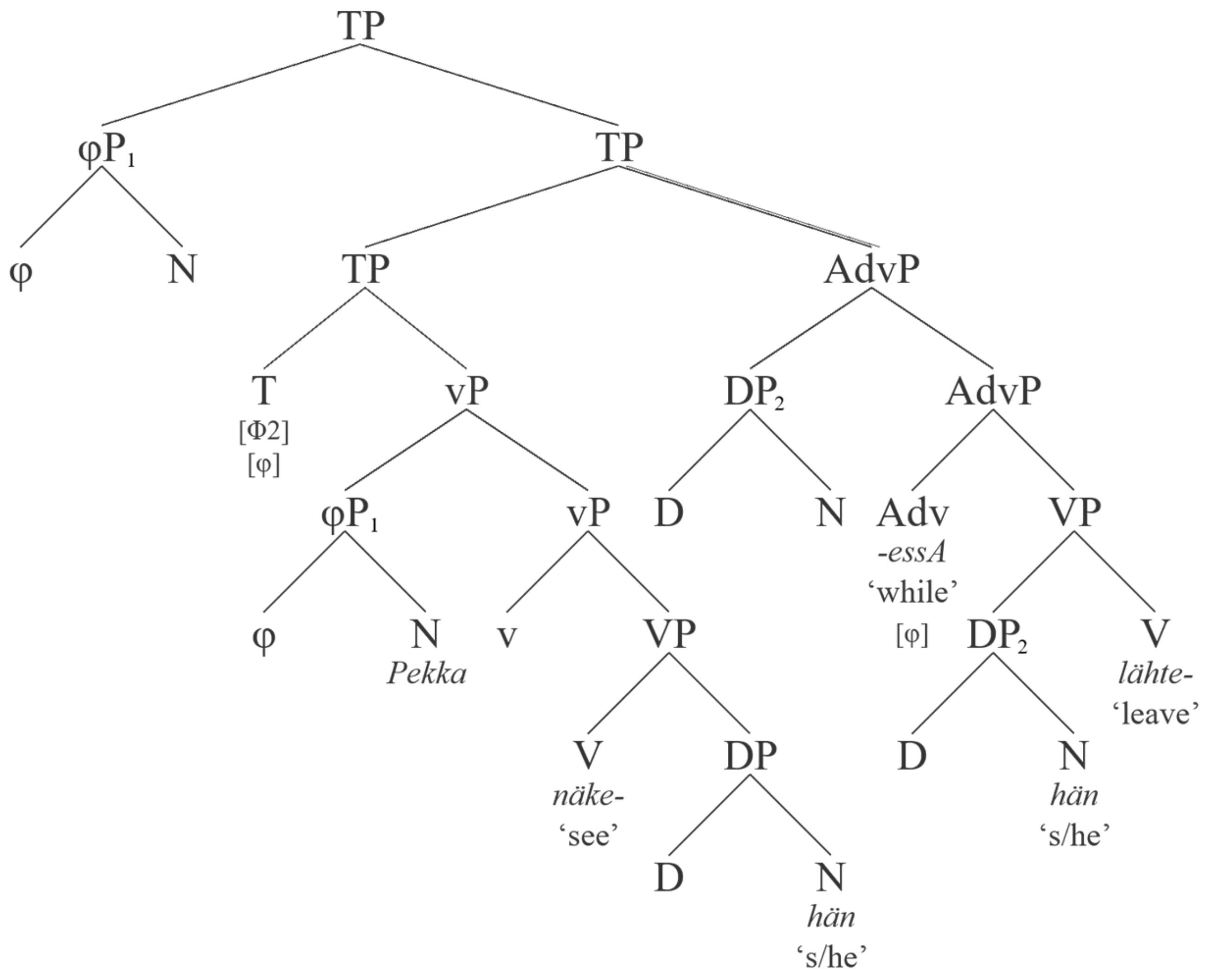
where only [$_{AdvP}$ ESSA/inf V] is projected and the subject is determined by subject control due to the high adjunction site. Adding agreement to the infinitival predicate derives



where agreement is visible as [φ] at the predicate. There is some variation in whether speakers accept the construction where the overt subject co-occurs with infinitival agreement (37).

(37)	??Pekka	näki	hänet	hän-en	lähti-essä-än. (#193)
	Pekka	saw	him	he-GEN	leave-ESSA/INF-3SG

There was considerable variation among speaker in whether this construction could express the subject redundantly. The difference can be represented by features Φ1 and Φ2 where the former judges (37) ungrammatical, the latter grammatical. The speaker model used in the present study adopted the latter lexicon (and hence also the corresponding brain model). Under this analysis, the analysis as follows.



The presence of the subject is optional, and when it is present, both phrasal subjects and infinitival agreement can identify it. Optional behavior does not need to be regulated by designated lexical features. If infinitival agreement is present in the input, the corresponding phi-features will be inserted inside the ESSA-infinitival head. If a phrasal subject occurs, it will be reconstructed into VP. If neither is present, the thematic subject is determined by subject control. All complement configurations are ungrammatical (38).

(38)	Pekka	käski/halusi/uskoi	[(hän-en)	lähti-essä(-än)]. (#183-194)
	Pekka	ordered/wanted/believed	he-GEN	leave-ESSA/INF(-3SG)

4.6 TUA-infinitive

The TUA-infinitive ‘after doing’ is an adjunct adverbial that has Φ1 which requires either a phrasal subject or infinitival agreement (39)–(41) but not both (#216, 220).

(39) a. *Pekka näki hänet [lähdet-tyä]. (#217)
 Pekka saw him leave-TUA/INF
b. *Pekka nukkui [lähde-ttyä]. (#221)
Pekka slept leave-TUA/INF

(40) a. Pekka$_1$ näki hänet$_2$ [hän-en$_{*1, 2, 3}$ lähdet-tyä]. (#215)
 Pekka saw him he-GEN leave-TUA/INF
 ‘Pekka saw him after he left.’
b. Pekka näki hänet [lähdet-tyä-än]. (#218)
Pekka saw him leave-TUA/INF-PX/3P
’Pekka saw him after he (Pekka) left.’

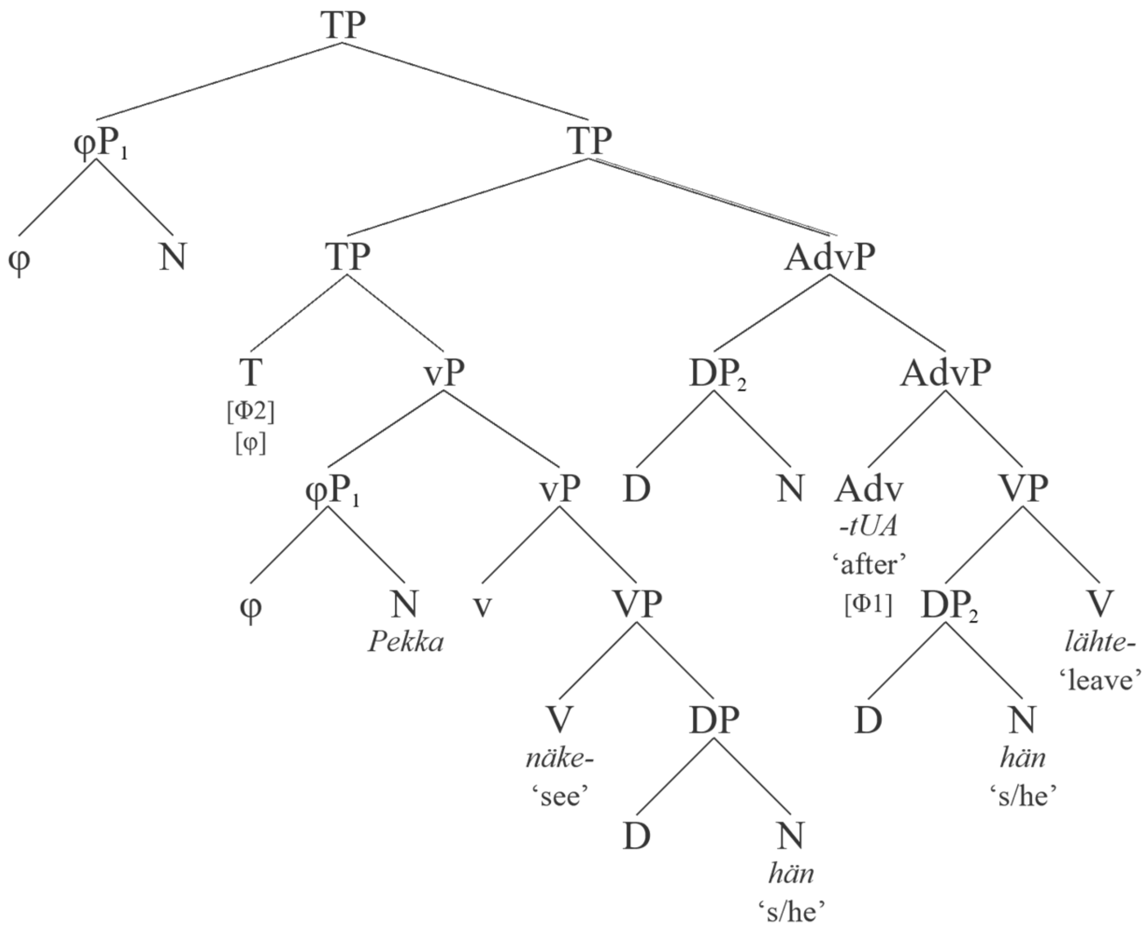
(41) a. Pekka nukkui [hän-en lähdet-tyä]. (#219)
Pekka slept he-GEN leave-TUA/INF
‘Pekka slept after he left.’
b. Pekka nukkui [lähdet-tyä-än]. (#222)
Pekka slept leave-TUA/INF-PX/3P
’Pekka slept after he left.’

It creates subject control constructions when the subject is absent. All complement configurations are ungrammatical (42).

(42)	*Pekka	käski/halusi/uskoi	[(hän-en)	lähdet-tyä(-än)]. (#203-214)
	Pekka	ordered/wanted/believed	he-GEN	leave-TUA/INF(-3SG)

The calculated analysis for (40)a is (41), where the TUA-infinitival is adjoined into a high position in the clause.

(43)



4.7 KSE-infinitive

The KSE-infinitive ('in order to do') is a subject control adjunct adverbial (44) which does not occur in complement configurations (45) and exhibits obligatory agreement (+ΦPF) and cannot, therefore, host an overt subject (46).

(44) a. Pekka$_1$ näki hänet$_2$ [lähteä-kse-en$_{1, *2}$]. (#238)

Pekka saw him leave-KSE/INF-PX/3P

‘Pekka saw him in order to leave.’

b. Pekka$_1$ nukkui [lähteä-kse-en$_1$]. (#242)

Pekka slept leave-KSE/INF-PX/3P

’Pekka slept in order to leave.’

(45) a. *Pekka käski/halusi/uskoi (hän-en) lähteä-kse-en. (#224, 228, 230, 231, 234)

Pekka ordered/wanted/believed he-GEN leave-KSE/INF-3SG

(46) a. *Pekka käski/uskoi hän-en lähteä-kse-en. (#224, 232)

Pekka ordered/believed he-GEN leave-KSE/INF-3SG

b. *Pekka näki hänet [hän-en lähteä-kse-en]. (#237)

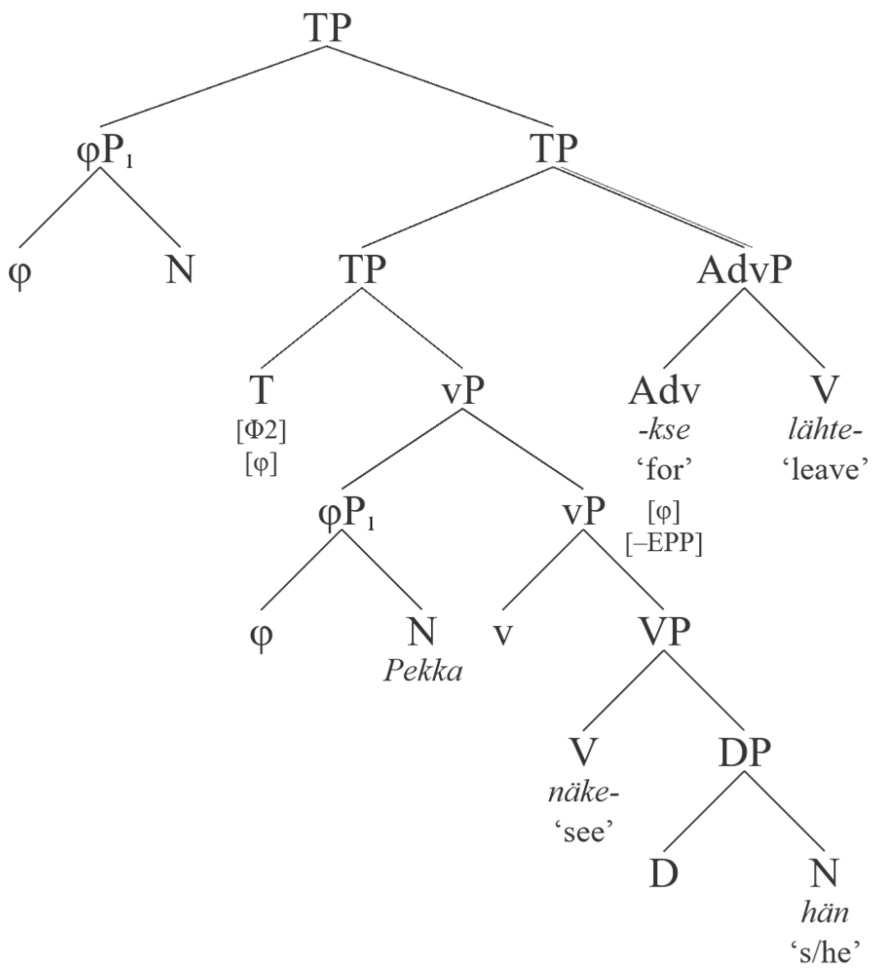
Pekka saw him he-GEN leave-KSE/INF-3SG

c. *Pekka nukkui [hän-en lähteä-kse-en]. (#240)

Pekka slept he-GEN leave-KSE/INF-3SG

The infinitival predicate is composed out of four elements: the verb stem, A-infinitival affix, the ESSA-infinitival affix and obligatory infinitival agreement. The A-infinitival affix was ignored by the morphological decomposition in this study, since, if it is present syntactically, it will get embedded inside the ESSA-infinitival head and does not affect the distribution of the infinitive (=(44)a, #238).

(47)



4.8 Participles

Finnish has two prenominal participle adjectives which serve a relativization function (48). The inanimate pronoun *se* 'the/it' functions as an optional determiner article in Finnish (Laury 1991, 1996, 1997) and is used here to highlight the syntactic position of the participle nonfinite clause inside the noun phrase.

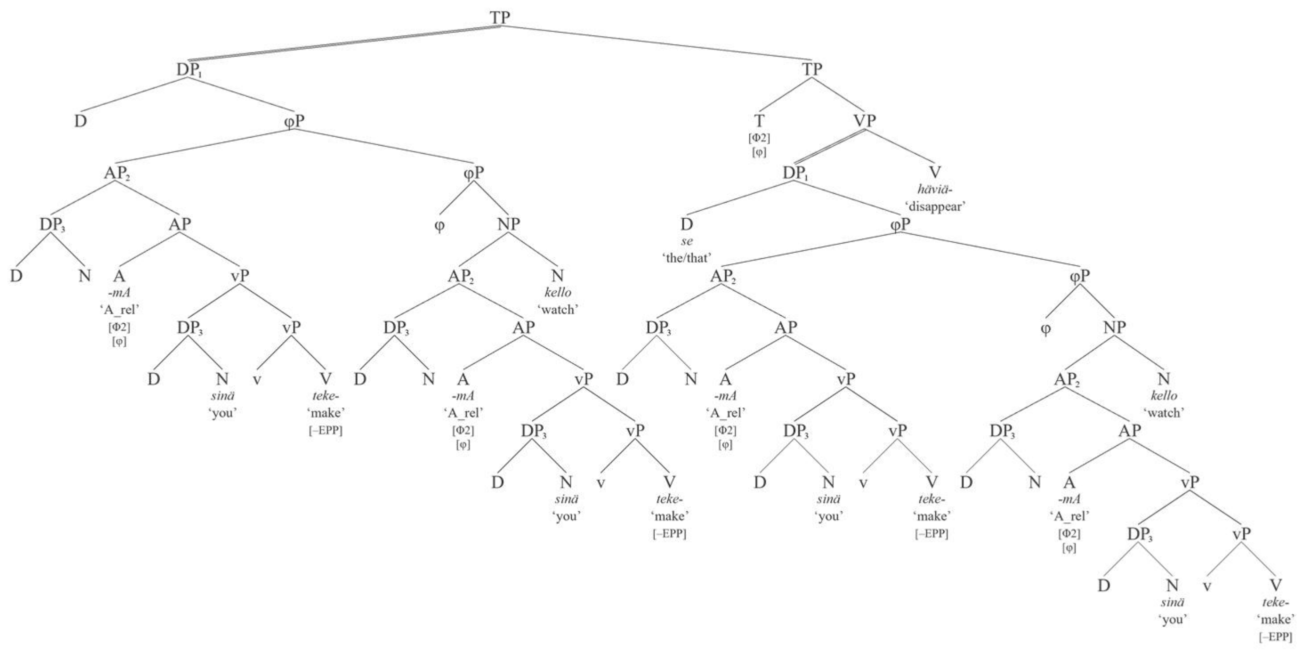
(48)	a.	[DP Se	[AP sinun	teke-mä(-si)]	kello]	hävis-i. (#253, 254)
		The/it	you.GEN	make-MA/A(-1SG)	watch.NOM	disappear-PST-3SG
		'The watch [made by you] disappeared.'				
	b.	[DP Se	[AP sinu-a	ihaile-va]	tyttö]	hävis-i. (#257)

The/it you-PAR admire-VA/A girl.NOM diseappear-PST.3SG

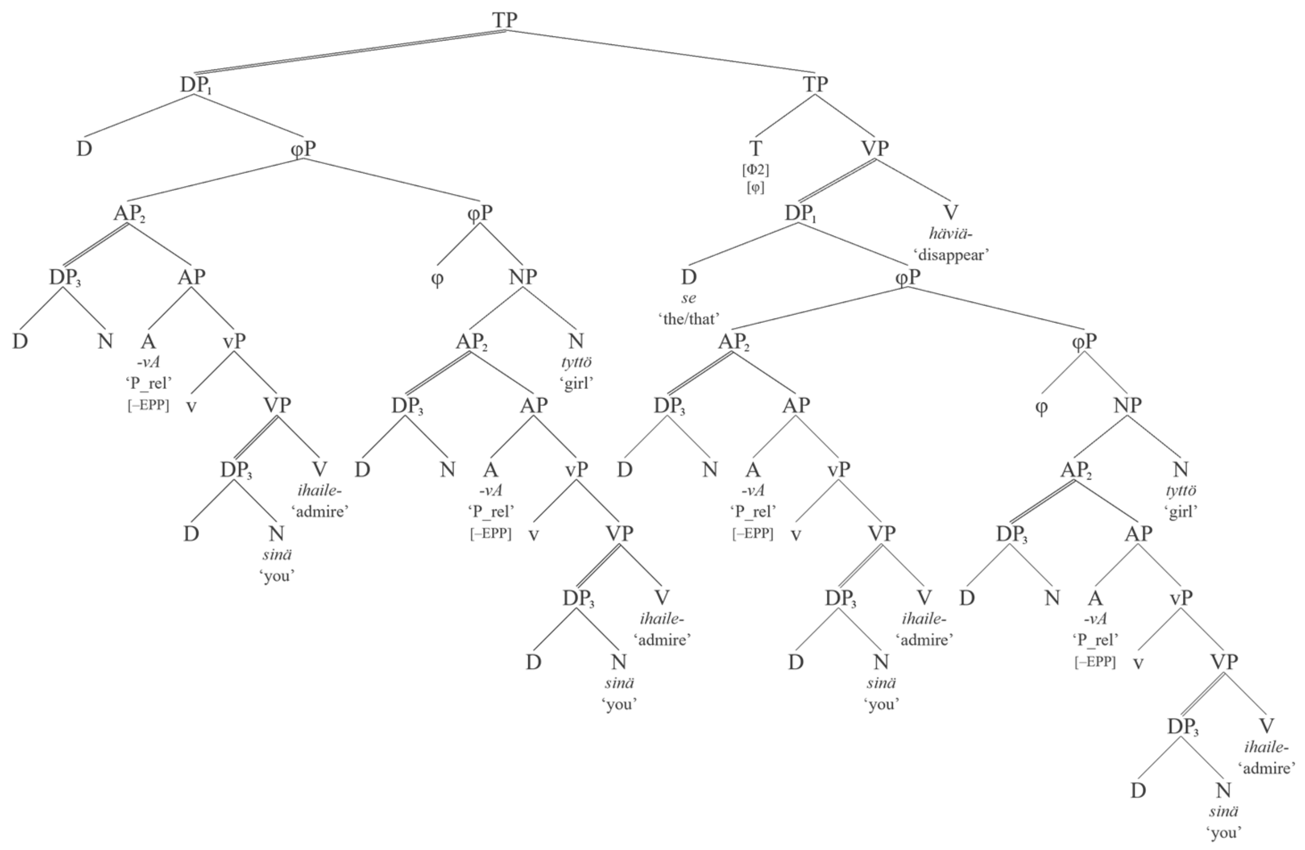
‘The girl [who admires you] disappeared.’

The algorithm analyses them as shown below. A_rel refers to “agent relativization”, P_rel to “aatient relativization.”

(49)



(50)



The MA-infinitival reconstructs the subject into SpecvP and generates the agent relativization 'who made', while the VA-infinitive reconstructs it to CompVP to generate a patient relativization 'admiring you'. The MA-infinitival is marked for the EPP behavior (51), while for the VA-infinitival the presence of an overt object is optional; agreement is not possible (52).

(51)	a.	se	teke-mä-si	kello
		the/it	make-MA/A-2SG	watch
	b.	*se	teke-mä	kello
		the/it	make-MA/A	watch

(52)	a.	se	nukku-va	mies
		the/it	speel-VA/A	man

‘The sleeping man’

b.	*se	nukku-va-si	mies
	the/it	sleep-VA/A-2SG	man

4.9 Infinitives, word order and binding

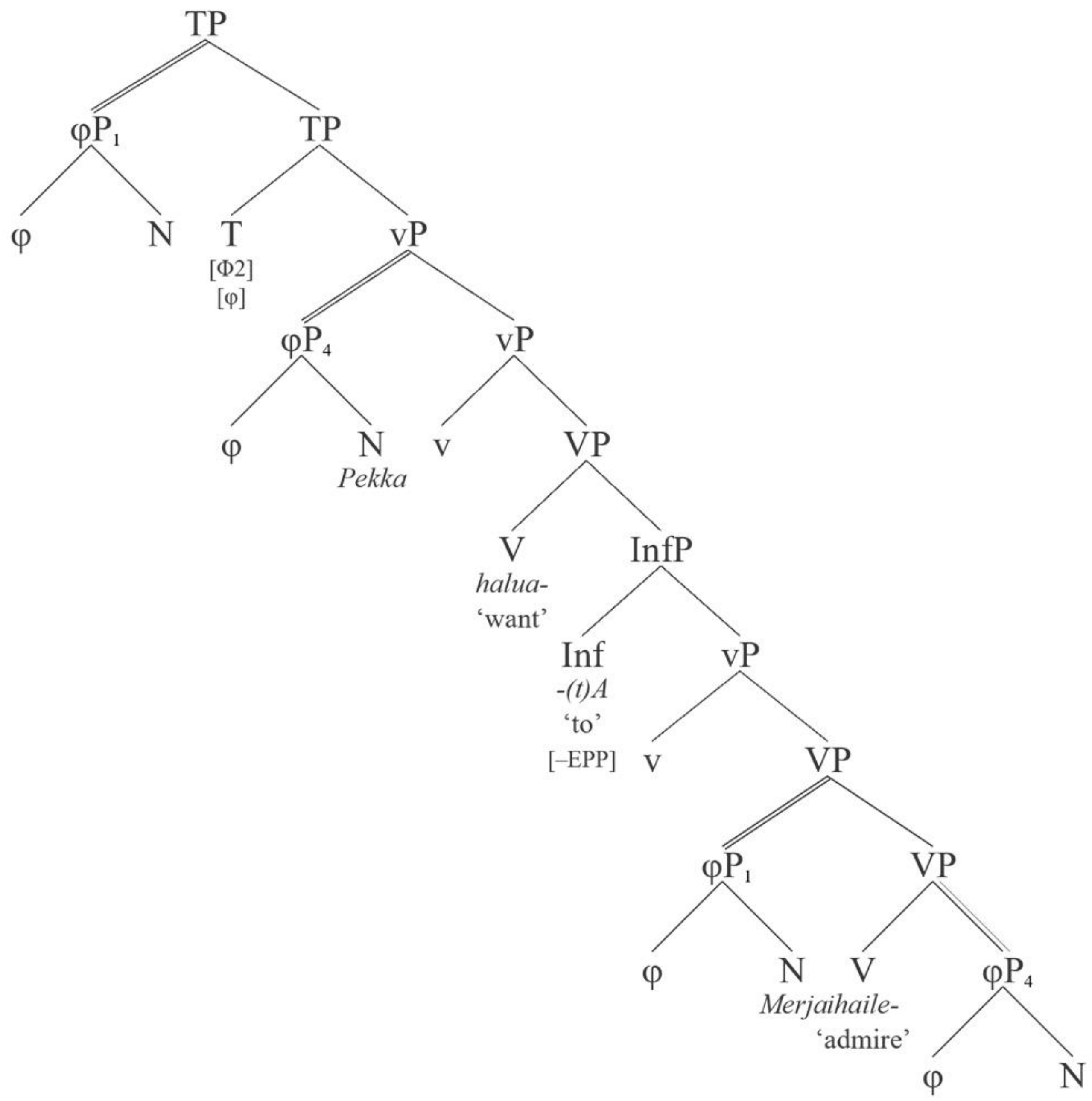
The dataset contained a few sentences testing that the analysis works in connection with a few basic examples of word order permutations (subject-object reversals) (53), (54). They were judged and calculated correctly.

(53) a.	Pekka	halus-i	ihail-la	Merja-a. (#243)
	Pekka.NOM	want-PST.3SG	admire-A/INF	Merja-PAR
	’Pekka wanted to admire Merja.’			
b.	Merja-a	halus-i	ihail-la	Pekka. (#244)
	Merja-PAR	want-pst.3SG	admire-A/INF	Pekka.NOM
	‘When it comes to Merja, it was Pekka who wanted to admire her.’			

(54) a.	Pekka	usko-i	ihaile-va-nsa	Merja-a. (#245)
	Pekka.NOM	believe-PST.3SG	admire-VA/INF-3SG	Merja-PAR
	’Pekka believed that he admires Merja.’			
b.	Merja-a	usko-i	ihaile-va-nsa	Pekka. (#246)
	Merja-PAR	believe-PST.3SG	admire-VA/INF-3SG	Pekka.NOM
	’When it comes to Merja, it was Pekka who believed to admire her.’			

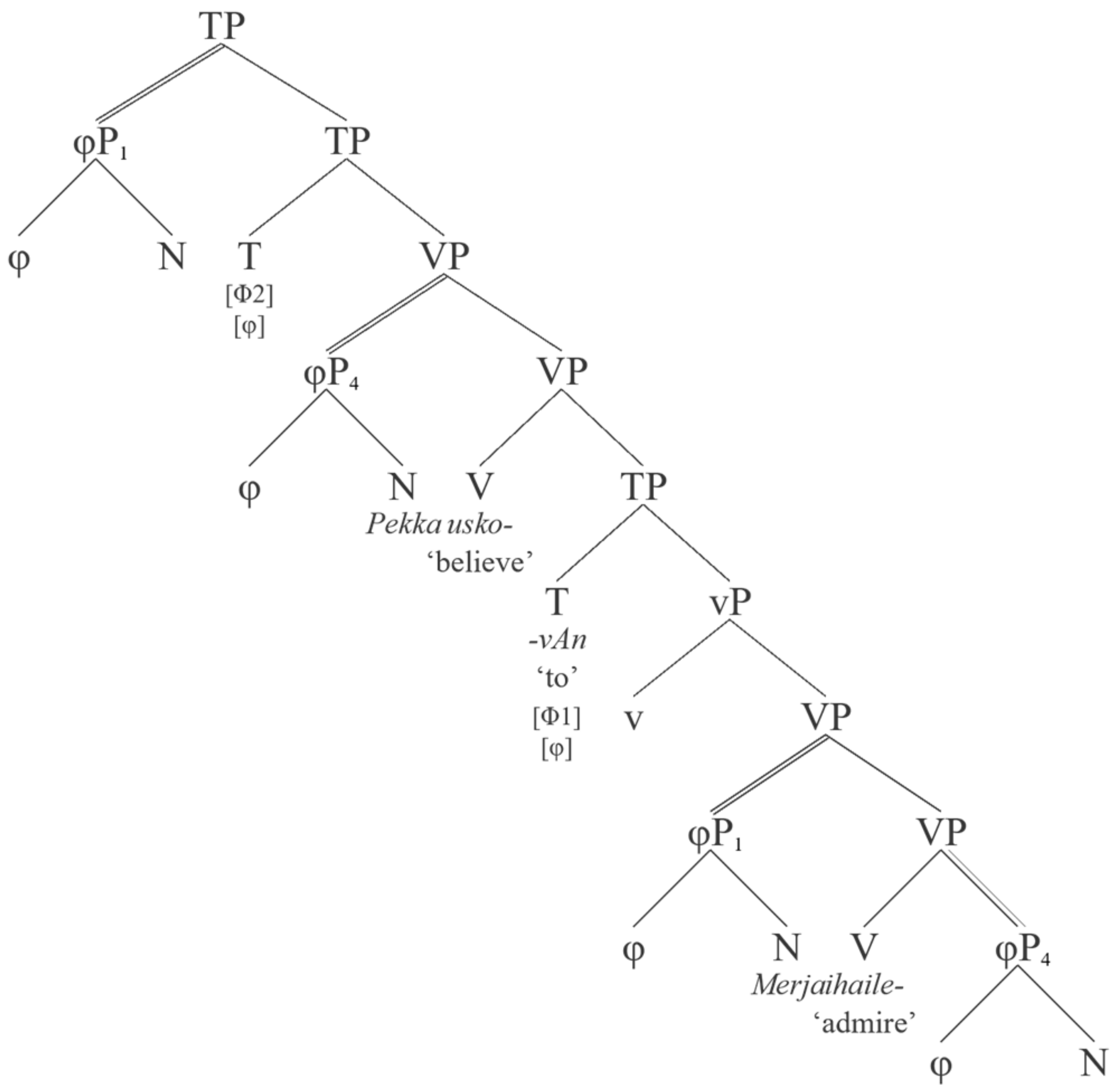
In (53), the algorithm reconstructs both arguments into their canonical positions as shown below.

(55)



Sentence (54) is calculated in the same way.

(56)



Also a battery of basic binding constructions were included. They are shown in (57). The coreference possibilities, shown by the subscripts, were correctly calculated.

(57)	a.	Pekka$_1$	halus-i	ihail-la	itse-ä-än$_1$.	
		Pekka.NOM	want-PST.3SG	admire-A/INF	self-PAR-3SG	
	b.	Pekka$_1$	käsk-i	Merja-n$_2$	ihail-la	itse-ä-än$_{*1, 2}$.

	Pekka.NOM	order-PST.3SG	Merja-GEN	admire-A/INF	self-PAR-3SG
c.	Pekka$_1$	käsk-i	hän-en$_2$	ihail-la	itse-ä-än$_{*1, 2}$.
	Pekka.NOM	order-PST.3SG	he-GEN	admire-A/INF	self-PAR-3SG
d.	Pekka$_1$	usko-i	ihaile-va-nsa	itse-ä-än$_1$.	
	Pekka.NOM	believe-PST.3SG	admire-VA/INF-3SG	self-PAR-3SG	
e.	Pekka$_1$	usko-i	Merja-n$_2$	ihaile-van	itse-ä-än$_{*1, 2}$.
	Pekka.NOM	believe-PST.3SG	Merja-GEN	admire-VA/INF	self-PAR-3SG
f.	Pekka$_1$	usko-i	hän-en$_{*1, 2}$	ihaile-van	itse-ä-än$_{*1, 2}$.
	Pekka.NOM	believe-PST.3SG	he-GEN	admire-VA/INF	self-PAR-3SG

Reference

Alexiadou, A., & Anagnostopoulou, E. (1998). Parametrizing AGR: Word Order, V-movement and EPP Checking. *Natural Language and Linguistic Theory.*, **16**(3), 491–539.

Brattico, P. (2019a). *Computational implementation of a linear phase parser. Framework and technical documentation (version 15.0)*, Pavia.

Brattico, P. (2019b). Subjects, topics and definiteness in Finnish. *Studia Linguistica*, **73**, 1–38.

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

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

Brattico, P. (2021b). Null arguments and the inverse problem. *Glossa: A Journal of General Linguistics*, **6**(1), 1–29.

Brattico, P. (2022a). Predicate clefting and long head movement in Finnish. *Linguistic Inquiry*, **54**(4), 663–692.

Brattico, P. (2022b). Structural case assignment, thematic roles and information structure. *Studia Linguistica*, **76**(3)

Brattico, P. (2023a). Across the board agreement in Finnish. *Manuscript Submitted for Publication*.

Brattico, P. (2023b). Subjects, EPP and agreement. *Ms*.

Brattico, P., & Chesi, C. (2020). A top-down, parser-friendly approach to operator movement and pied-piping. *Lingua*, **233**, 102760.

Brattico, P., Chesi, C., & Suranyi, B. (2023). Agreement and chains. *Ms*.

Chomsky, N. (1957). *Syntactic Structures*, The Hague: Mouton.

Chomsky, N. (1981). *Lectures in Government and Binding: The Pisa Lectures*, Dordrecht: Foris.

Chomsky, N. (1982). *Some Concepts and Consequences of the Theory of Government and Binding*, Cambridge, MA.: MIT Press.

Chomsky, N. (2004). Beyond explanatory adequacy. In A. Belletti, ed., *Structures and Beyond: The Cartography of Syntactic Structures, volume 3*, Oxford: Oxford University Press, , pp. 104–131.

Chomsky, N. (2008). On Phases. In C. Otero, R. Freidin, & M.-L. Zubizarreta, eds., *Foundational Issues in Linguistic Theory: Essays in Honor of Jean-Roger Vergnaud*, Cambridge, MA.: MIT Press, , pp. 133–166.

Holmberg, A. (2000). Scandinavian Stylistic Fronting: How any Category can become an Expletive. *Linguistic Inquiry.*, **31**(3), 445–484.

Holmberg, A. (2005). Is There a Little pro? Evidence from Finnish. *Linguistic Inquiry*, **36**(4), 533–564.

Holmberg, A., & Nikanne, U. (2002). Expletives, subjects and topics in Finnish. In P. Svenonius, ed., *Subjects, Expletives, and the EPP*, Oxford: Oxford University Press, , pp. 71–106.

Huhmarniemi, S. (2012). *Finnish A'-movement: Edges and Islands*, University of Helsinki, Helsinki.

Huhmarniemi, S. (2019). The movement to SpecFinP in Finnish. *Acta Linguistica Academica*, **66**(1), 85–113.

Kayne, R. (1983). Connectedness. *Linguistic Inquiry*, **14.2**, 223–249.

Kayne, R. (1984). *Connectedness and Binary Branching*, Foris.

Kiparsky, P. (2019). Notes on nonfinite clauses in Finnish. In C. Condoravdi & T. King, eds., *Tokens of Meaning: Essays in Honor of Lauri Karttunen*, Stanford, CA: CSLI Publications.

Koskinen, P. (1998). *Features and categories: Non-finite constructions in Finnish* (PhD. dissertation), University of Toronto.

Laury, R. (1991). On the development of the definite article se in spoken Finnish. In *The SKY Yearbook*, , pp. 93–121.

Laury, R. (1996). Sen kategoriasta -- onko suomessa jo artikkeli? *Virittäjä*, **100**, 162–181.

Laury, R. (1997). *Demonstratives in Interaction – The Emergence of a Definite Article in Finnish.*, Amsterdam/Philadelphia: John Benjamins.

Phillips, C. (1996). *Order and structure* (Ph.D. thesis). *MIT*, Cambridge, MA.

Vainikka, A. (1989). *Deriving Syntactic Representations in Finnish* (Ph.D. thesis), University of Massachusetts Amherst.

Vainikka, A. (1995). Functional Projections in Finnish Non-Finite Constructions. *University of Pennsylvania Working Papers in Linguistics (PWPL)*, **2**(1).

Vainikka, A., & Levy, Y. (1999). Empty subjects in Finnish and Hebrew. *Natural Language & Linguistic Theory*, **17**(3), 613–671.

Ylinärä, E. (2018). *Forme infinitive in finlandese,* Munchen: LINCOM.