3

Radical Lexicalism

LAURI KARTTUNEN

1. Introduction

The kind of grammar I discuss here is a marriage of Bay Area unification formalism (Kaplan et al. 1986) with a much older, distinguished partner called Categorial Grammar (CG). A number of people are currently exploring categorial unification grammars, but very little has appeared in print so far (Pareschi 1986, Uszkoreit 1986, Whitelock 1986, Wittenburg 1986). This framework seems particularly suited to the description of a language like Finnish, in which the morphology provides a lot of information about the syntactic role of constituents and in which the word order is relatively uninformative in this regard. The aim of this paper is to explore a radical lexicalist view of syntax. In many current approaches to syntax, much of the information about how words are combined into phrases is encoded in the lexicon. What I try to do is to merge techniques for lexical encoding developed in such frameworks as Lexical-Functional Grammar (LFG), HPSG, and the PATR project at SRI with the combinatorial principles of categorial grammar, thus doing away with phrase-structure rules altogether.

Because the framework of this paper is unfamiliar to many linguists, I start with a brief review of categorial grammars and explain how such grammars can be encoded in a unification-based grammar formalism. (For an excellent introduction to unification, see Shieber 1986.) I then present some basic facts about Finnish word order and give an account of the data within a very simple fragment of categorial unification grammar. I review some more problematic data concerning nonlocal dependencies and explain how the simple analysis can be extended to such cases.

This paper, made possible in part by a gift from the Systems Development Foundation, was also supported by a grant from the Nippon Telegraph and Telephone Company. I am grateful to Annie Zaenen and John Bear for their help on the form and content of this paper.
1.1. Classical Categorial Grammar

A characteristic feature of CG is that the lexical entries of words encode virtually all the information about how words are combined into phrases; there is no separate component of syntactic rules, as is found in most other grammatical frameworks. The combinatorial properties of a word are indicated by the name of its lexical category. There are two types of categories: **basic** and **derived** (= **functor**). Basic categories are designated by atomic labels; the names of derived categories are complex expressions. In a prototypical CG, common nouns and noun phrases are regarded as basic categories; the set of functor categories includes determiners and intransitive verbs.

(1) **Basic Categories**  
CN (common noun)  
NP (noun phrase)  

**Derived Categories**  
NP/CN (determiner)  
S\NP (intransitive verb)

The labels of derived categories play the same role in CG as syntactic rules do in a phrase-structure grammar. By assigning determiners to the category NP/CN, the grammarian indicates that determiners combine with common nouns to form noun phrases. In effect, the name of this category encodes the phrase-structure rule NP → Det CN. More generally, the category labels for derived categories are of the form

(2) (result direction argument),

where “result” and “argument” are category labels, either simple or complex, and “direction” is either / or \, depending on whether the functor comes before (/) or after (\) its argument. An example of the latter type is the label for intransitive verbs, S\NP, which indicates that phrases of this category form sentences by combining with a preceding noun phrase. Transitive verbs are commonly assigned to the category (S\NP)/NP, adverbs to (S\NP)/(S\NP). Under this analysis, transitive verbs combine with a noun phrase on their right, while adverbs attach to an intransitive verb phrase on their left. In both cases, the resulting expression is an intransitive verb phrase. A standard CG analysis tree for *John greeted Mary warmly* is:

(3)

```
  S
 / \  
NP   S\NP
    / \   / \ 
  John S\NP (S\NP)/(S\NP)
   / \  /  
(S\NP)/NP NP warmly
  /    /   |
greeted Mary
```
In CG, expressions that belong to nonbasic categories, such as S\NP, are commonly called *functors*. In this context, the term simply means that their category label encodes a combinatorial rule; there is no necessary connection between this technical notion of functor and some semantic or syntactic distinction between functors and arguments. All that distinguishes functors from arguments in the terminology we are using here is that, when a functor is combined with an argument, it is the former that determines what the constraints on the combination are and what the result will be. From the point of view of a parser, functors are syntactically "active," in the sense that they enable the parser to form new constituents, whereas basic expressions are syntactically "inert."

In a prototypical CG, the only combinatorial principle is that of *functional application*: an expression of a derived category can combine with an adjacent phrase, provided that the latter qualifies as the argument. A categorial grammar of this kind is equivalent to a standard phrase-structure grammar. In recent work on CG, Steedman (1988), Dowty (1988), and others have made use of more powerful combinatorial principles. Binary rules of *functional composition* allow functors to combine with other functors; unary rules of *type-raising* convert basic expressions to functors; or functors of one type are converted to functors of another type. Functional composition and type-raising are used in Dowty's (1988) treatment of nonconstituent coordination and in Steedman's (1988) account of wh-questions and relative clauses. Such rules can be expressed easily in our formalism (Uszkoreit 1986). Although we do not use them here, we consider them briefly in §5. The phenomena covered in our grammar—word order variation and long-distance dependencies—can be described simply in terms of functional application. We take advantage of the fact that our unification-based formalism enables us to construct functors that cannot be expressed in the original CG notation consisting of atomic category labels and slashes. Because type-raising and functional composition, in their most general form, are difficult to constrain, it is a welcome result not to have to invoke them in this case.

1.2. *Categorial Unification Grammar*

In the last few years, the notion of unification has emerged as a common descriptive device in many linguistic theories (LFG, Functional Unification Grammar (FUG), Head-Driven-Phrase-Structure Grammar (HPSG). Unification is an operation that merges partial information as long as it is consistent; it fails when the information becomes incompatible. A simple example of unification involves the merging of features. In (4), A and B are feature sets containing information about agreement and case. The unifica-
tion of $A$ and $B$ is a feature set that results from merging $A$ and $B$ into a single set of features.

\[
\begin{align*}
A & = \begin{bmatrix} \text{agreement} \ [\text{number plural}] \\ \text{case} \ & \text{nominative} \end{bmatrix} \\
B & = \begin{bmatrix} \text{agreement} \ [\text{person 3d}] \\ \text{case} \ & \text{nominative} \end{bmatrix} \\
A \cup B & = \begin{bmatrix} \text{agreement} \ [\text{number plural}] \\ \text{case} \ & \text{nominative} \end{bmatrix}
\end{align*}
\]

In (4), the three feature sets are displayed as matrices. They can also be represented as directed graphs, as we shall do from now on. The graph (5) represents the same information as the last of the three matrices in (4):

\[
\begin{align*}
\text{agreement} & \leftarrow \text{number} \rightarrow \text{plural} \\
\text{case} & \leftarrow \text{person} \rightarrow \text{3d}
\end{align*}
\]

A sequence of attributes in a graph is called a path. In the case at hand, the path (agreement number), for example, has the value "plural."

One advantage of unification is that it gives a precise meaning to rules stated in terms of attributes and values. A unification-based formalism leads to declarative descriptions of syntactic structure. It is not compatible with the notion of a derivational process that changes structure by inserting, deleting, or reordering constituents. Unification-based descriptions are monotonic in the sense that whatever is true of a constituent at some stage of the analysis remains true when new information has been accumulated.

In a unification grammar, it is convenient to express syntactic rules in the form of feature sets. In the case of phrase-structure rules, the left-hand side of the rule and the constituents on the right-hand side are each represented by an attribute whose value is a feature set for that constituent (Shieber 1986, Karttunen 1986). To express the principle of functional application in this format, we introduce five attributes: cat, left, right, argument, and result. For all basic expressions, the value of argument is none. This ensures that they cannot act as functionals. The attribute cat is used to record the category label. The values for left, right, and result are relevant only for nonbasic expressions. The attributes left and right distinguish between backward- and forward-looking functionals. A forward-oriented functor, such as a determiner in English, has an indeterminate value (indicated by []) for right, and none as the value of left; a backward-looking functor has the features left: [], right: none. The attributes argument and result are used to encode what kind of expression the functor combines with and
what the result will be. The value of argument is unified with the feature set of the expression the functor combines with. If this unification fails, the two expressions cannot be combined; if it succeeds, the feature set of the consequent expression is the same as the value of result.

In our framework, the forward-oriented version of functional application is analogous to the phrase-structure rule

(6) Result → Functor Argument.

A detailed statement of the rule is given in graph (7). The paths ⟨Functor⟩ and ⟨Argument⟩ lead to the feature sets of the two expressions that are combined, ⟨Result⟩ points to the feature set of the concatenated phrase, while the converging lines denote unifications.

(7)

The backward-oriented rule is the same except that the paths ⟨Argument⟩ and ⟨Functor argument⟩ are unified with ⟨Functor left⟩ rather than ⟨Functor right⟩.

To see how this version of functional application works, consider the following simple example of a determiner combining with a common noun to form a noun phrase. We use here the standard CG analysis of determiners, shown in 8a. The feature cat in 8a is actually redundant here because its content is spelled out in the values of the other features; it is included only for its mnemonic value. The feature set for the argument—a common noun—is shown in the graph 8b, while the result of the combination appears in the graph 8c:

(8)

c. Result: argument — none

cat — NP

cat — NP/CN

argument — — cat — CN

result — < cat — NP

argument — none

left — none
cat — CN

a. Functor: right — []
b. Argument: argument — none
When the determiner and the common noun are combined by functional application, the determiner's values for the paths \(<\text{argument}>\) and \(<\text{right}>\) are successfully unified with the features of the common noun. The value of the determiner's \textit{result} attribute becomes the feature set of the noun phrase so formed. This example is oversimplified, as it ignores all the features that are specific to particular lexical items—nor have we indicated how the semantics of the noun phrase is determined by its composition. In a real grammar, all three feature sets have more content than is shown here. As the later examples show, parts of the feature sets of the argument and the functor may be incorporated in the result through unification. On the other hand, in one respect the example could be even simpler, since it is possible to encode the same categorial information by using fewer features. For example, instead of \textit{left} and \textit{right} we could just have a single feature, \textit{direction}, with \textit{left}, \textit{right} as possible values. The advantage of using more features is that we can also introduce two-sided functors that combine to both left and right, possibly with different constraints on their arguments. In this respect, the notation used here is a generalization of the standard CG convention.

2. A Fragment of a Categorial Unification Grammar for Finnish

In this section, we develop a grammar for a subset of Finnish. The purpose is to give a concrete example of a categorial grammar in unification-based formalism with enough coverage to illustrate the problems of word order and long-range dependencies. In particular, we cover simple noun phrases functioning as subjects and objects, auxiliary verbs, and some types of main verbs and infinitival complements. In constructing this grammar, we concentrate on developing an account for the following simple generalizations about Finnish word order:

(9) a. In declarative sentences, subjects and objects may occur in any order with respect to the verb and to one another.
   b. In yes/no questions and imperatives, the finite verb comes first.
   c. The negative auxiliary (\textit{e-}) precedes the temporal one (\textit{ole-}), and both precede the main verb, but the three types of verbs need not be adjacent.

After accomplishing this task, we move on to confront a much harder problem:

(10) Elements of participial and infinitival clauses can be interspersed among the constituents of a superordinate clause.
This problem is similar to the case of English topicalization, in that the clause to which a dislocated constituent syntactically and semantically belongs can be arbitrarily distant. Yet it differs in that the migrant phrase is not in a special syntactic position, as a topicalized phrase is, that identifies it as possibly being a dependent of some distant subordinate clause. Another difference is that, while there is only one topic position per clause, any number of constituents from an embedded clause can occur in a subordinate clause. In §5, we discuss three possible solutions to this type of nonlocal dependency.

2.1. Noun Phrases

In a major departure from CG tradition, we make verbs serve as basic expressions and define noun phrases as functors. This is a natural choice for languages in which word order is relatively free and case-marking provides information about the syntactic role played by noun phrases with respect to verbs. (Whitelock 1986 suggests a similar analysis for Japanese.) To keep our examples as simple as possible, we pretend here that the syntactic roles of noun phrases are determined unambiguously by case. In particular, let us assume that a noun phrase in the nominative case always functions as a subject with respect to some verb and that accusative NPs are always objects. In a real grammar for Finnish, the mapping from case to syntactic roles is of course many-to-many, rather than the one-to-one relationship we are stipulating here. For now, we define a nominative NP in Finnish as an expression that combines with an adjacent verb phrase to yield a verb phrase in which it plays the role of the subject. In terms of a feature set, this combinatorial property is expressed as follows:

(11) semantics
    morphology
    left — []

    argument
    right — semantics — type — declarative
      cat — V

    result
      syntax — Subject — semantics — []
      morphology — []

    argument — NONE

As graph (11) shows, according to our definition subjects are two-sided functors; they can combine either to the left or to the right, but under different conditions. The graph for a partitive NP is the same as above except that it has the object role. Because the value of ⟨left⟩ is indeterminate, there is no constraint on leftward combination; however, the rightward combina-
tion requires that the verb be declarative. This constraint is designed to account for the pattern shown in (12). The ungrammaticality of (12d) is due to the constraint that prohibits the subject from combining to the right with an interrogative verb.

(12) a. Liisa nukkui.
   'Lisa slept.'

b. Nukkui Liisa.
   'Lisa did sleep.'

c. Nukkuiko Liisa?
   'Did Lisa sleep?'

d. *Liisa nukkuiko?
   'Lisa slept?'

Because the value of argument is unified with the value of result in the feature sets of nominative NPs, the phrase Liisa nukkui in our grammar is of Category V just as nukkui ‘slept’ is; in this respect the analysis is reminiscent of Dependency Grammar (Tesnière 1959).

2.2. Main Verbs

All types of main verbs belong to the same basic category but have different valences. In this fragment, verbs are subcategorized with respect to three possible syntactic roles: subject, object, and vcomp (verbal complement). The valence of a simple intransitive verb, such as nukkui, is indicated in the following manner:

(13) syntax — ← subject — []
     object — NONE
     vcomp — NONE

In this graph, the value of the path (syntax subject) is indeterminate; consequently, any subject NP can contribute its content to this position. For example, when the subject NP Liisa is combined with nukkui by functional application, the above graph is unified with

(14) syntax — — subject — ← semantics — Lisa
     morphology — — case — nominative

On the other hand, because the value of (syntax object) is NONE for intransitive verbs, such verbs cannot combine with an object NP. The atomic value NONE does not unify with any nonempty set of features.

As far as semantics is concerned, we follow tradition in assigning to each verb a relation that links the semantic values of those constituents that
make it syntactically complete. For example, the feature set for the transitive verb *rakasta* ‘love’ assigns the subject to the first and the object to the second position in the “love” relation:

(15)

\[
\begin{align*}
\text{semantics} & \rightarrow \text{issue} \\
& \rightarrow \text{arg1} \\
& \rightarrow \text{arg2} \\
\text{subject} & \rightarrow \text{semantics} \\
\text{syntax} & \rightarrow \text{object} \\
& \rightarrow \text{semantics} \\
\text{vcomp} & \rightarrow \text{NONE}
\end{align*}
\]

The attribute *issue* in the above graph is used to distinguish the basic semantic relation expressed by the verb stem. The complete semantic description of a fully inflected verb contains a number of other semantic properties that are contributed by inflectional suffixes, clitics, and auxiliary verbs. We use the following traditional attributes: *polarity* (positive, negative), *tense* (present, past), *tenseaspect* (simple, perfect, mood (indicative, conditional, potential), and *type* (declarative, imperative, interrogative). For example, in the graph for the word *rakasti* ‘loved’, the value of *(semantics)* is

(16)

\[
\begin{align*}
\text{relation} & \rightarrow \text{love} \\
\text{issue} & \rightarrow \text{arg1} \\
& \rightarrow \text{arg2} \\
\text{mood} & \rightarrow \text{indicative} \\
\text{polarity} & \rightarrow \text{positive} \\
\text{tense} & \rightarrow \text{past} \\
\text{tenseaspect} & \rightarrow \text{simple} \\
\text{type} & \rightarrow \text{declarative}
\end{align*}
\]

2.3. *Auxiliary Verbs*

Unlike main verbs, which are basic expressions, auxiliary verbs are forward-oriented functors in our system. The tense auxiliary combines with the past participle form of a verb, the negative auxiliary with either a past participle or a special negation form. The resulting construction is of the same type as the argument: a basic expression of category V. Its morphological properties and some of its semantic features (*tense, polarity*) come from the functor; the argument contributes the syntactic features as well as the rest of the semantics. For example, the past tense form of the tense auxiliary *oli* ‘had’ combines verb phrases that unify with the following graph:
(17) \text{cat} - V
  \text{semantics} \leftarrow \text{issue} - []
  \quad \text{polarity} - \text{positive}
  \text{morphology} \leftarrow \text{case} - \text{nominative}
  \quad \text{form} - \text{participle}
  \text{syntax} - []
  \text{argument} - \text{NONE}

In other words, the argument must be a participial form of a basic verb in the nominative case. The result of the combination has the feature set shown below:

(18) \text{cat} - V
  \text{semantics} \leftarrow \text{issue} - []
  \quad \text{mood} - \text{indicative}
  \quad \text{polarity} - \text{positive}
  \quad \text{tense} - \text{past}
  \quad \text{tenseaspect} - \text{perfect}
  \quad \text{type} - \text{declarative}
  \text{morphology} \leftarrow \text{number} - \text{sg}
  \quad \text{person} - \text{3}
  \text{syntax} - []
  \text{argument} - \text{NONE}

In this graph, the paths \langle \text{syntax} \rangle and \langle \text{semantics issue} \rangle are actually unified with the corresponding paths in the previous graph; the values of these paths derive from the argument. To show this graphically, we need to view the two subgraphs in the context of the complete graph for the auxiliary. Except for the value of \langle \text{semantics tenseaspect} \rangle, which is a special contribution of the tense auxiliary, all the other features under \langle \text{semantics} \rangle and the value of \langle \text{morphology} \rangle are inherited from the functor directly. Thus \text{oli rakastanut} ‘had loved’ is a declarative singular 3d person verb phrase with past tense because \text{oli} ‘had’ by itself has these features.

Because the negative auxiliary does not have a participial form, it can never follow the tense auxiliary. The fact that both types of auxiliaries are forward-oriented functors means that they must precede the main verb to find their argument. Consequently, the three types of verbs must come in a fixed order:

(19) \text{NegAux} < \text{TenseAux} < \text{MainVerb}.
2.4. Infinitival Complements

Like all other types of main verbs, predicates that take verbal complements (VComp) are basic expressions in our system. There are many types of infinitival and participial complements in Finnish. In this paper, we cover only complements in which the verb is either in the 1st or 3rd infinitive form. In our system, infinitives are backward-oriented functors seeking to combine with a verb of the appropriate type. The resulting combination has the feature set of the argument with the infinitival complement unified in as the value of the path ⟨syntax vcomp⟩. For example, 1st infinitive form of nukkua ‘sleep’ has the following value for the path ⟨argument syntax⟩:

\[
\text{(20)}
\]

As an example of a verb that qualifies as the argument—that is, takes complements of this type—consider the verb halusi ‘wanted’. Its lexical representation specifies the following value for ⟨syntax⟩:

\[
\text{(21)}
\]

This graph indicates that the complement of halusi must be a 1st infinitive and that the subject roles of the two verbs are unified. Consequently, in a sentence such as Liisa halusi nukkua ‘Liisa wanted to sleep’, the subject of the main verb is also the subject of the infinitive. An analysis tree for this sentence and the resulting semantic representation are shown in (22):
3. Word-Order Variation

In terms of classical CG, the main idea in the grammar of Finnish we have sketched in the preceding section is that auxiliary verbs, infinitival complements, subjects, and objects are functors of type V/V or V\V. They combine with a basic verb phrase to yield a verb phrase that is the same as the original except that the content of the functor has been merged with the argument in a manner that is determined by the functor. This aspect of the analysis, of course, can only be expressed in a unification-based formalism. In a more complete fragment, the same treatment would be extended to adverbs. Because all the relevant information is expressed in the lexical entries, a program that uses the grammar for parsing has no need for phrase-structure rules. The parsing algorithm can be extremely simple: Always try to combine a functor with an adjacent argument.

This grammar allows for considerable freedom in word order. For example, because the result of combining a subject with the main verb is a basic verb phrase, the subject NP may occur between an auxiliary and the
main verb. The same is true of objects. Consequently, there are four ways of saying that Lisa has not slept:

(23) a. Liisa ei ole nukkunut.
    b. Ei Liisa ole nukkunut.
    c. Ei ole Liisa nukkunut.
    d. Ei ole nukkunut Liisa.

The resulting set of features is the same regardless of the position of the subject:

(24) \textit{cat} \rightarrow \textit{V}

issue \rightarrow \textit{relation} \rightarrow \textit{sleep}

arg1

mood \rightarrow \textit{indicative}

polarity \rightarrow \textit{negative}

tense \rightarrow \textit{past}

tenseaspect \rightarrow \textit{perfect}

type \rightarrow \textit{declarative}

morphology

\textit{number} \rightarrow \textit{Lisa}

\textit{person}

\textit{case} \rightarrow \textit{nominative}

\textit{number} \rightarrow \text{sg}

\textit{person} \rightarrow \text{3}

syntax

subject \rightarrow \textit{morphology}

\textit{number} \rightarrow \text{NONE}

\textit{person} \rightarrow \text{3}

\textit{vcomp} \rightarrow \text{NONE}

\textit{argument} \rightarrow \text{NONE}

The grammar also places no constraints on the order of the subject with respect to the object. Any of the six possible orderings of the three-word sentence ‘John loved Lisa’ is allowed; the set of features assigned to the sentence remains the same regardless of the order.

    b. Jussi Liisaa rakasti.
    c. Liisaa Jussi rakasti.
    d. Liisaa rakasti Jussi.
    e. Rakasti Jussi Liisaa.
    f. Rakasti Liisaa Jussi.

The only explicit constraint on the ordering of noun phrases is that, when they combine to the right, the argument must be declarative. As we pointed out in §2.1, a verb that is marked with the interrogative enclitic -ko
fails this condition: such a verb cannot be preceded by any of its dependents. Another way of forming interrogative verb phrases in our system is to combine an interrogative noun phrase with an ordinary verb phrase. In Finnish, interrogative noun phrases are of two types: ordinary noun phrases suffixed with the interrogative clitic, e.g. Liisaako, and phrases containing an interrogative pronoun, e.g. ketä ‘who’. In our grammar, both types of interrogative NPs are forward-oriented functors; they combine only to the right and yield an interrogative verb phrase as the result. Because noninterrogative functors in our system combine to the right only if the argument is declarative, this analysis correctly predicts that the constituent that makes the sentence interrogative is always in the initial position:

(26) a.  Rakastiko Jussi Liisaa?
       ‘Did John love Lisa?’
 b.  *Jussi rakastiko Liisaa?
c.  Liisaako Jussi rakasti?
       ‘Was it Lisa that John loved?’
d.  *Jussi Liisaako rakasti?
e.  Ketä Jussi rakasti?
       ‘Who did John love?’
f.  *Jussi ketä rakasti?

By treating subject and object noun phrases syntactically as functors we have accounted for the three generalizations about Finnish word order in (9) by using only function application. In a categorial grammar of a more traditional sort, in which the verb is responsible for linking up with its syntactic dependents, type-raising and functional composition must be introduced to achieve the same result.

4. Multiple Analyses

It is convenient to represent the analysis of a phrase as a tree that shows how the resulting feature set was derived. However, the structure of the tree has no linguistic significance in our system; in this respect our trees are different from phrase-structure trees. All that matters is the resulting feature set. Because no functor has any priority over others with respect to order of application, the same result can often be obtained in more than one way. This is potentially troublesome from a computational point of view. For example, the sentence Jussi rakasti Liisaa has two possible analysis trees in our grammar:
Tree (27a) shows an analysis in which the subject combines with the verb first and the object takes the result as its argument. In (27b), the object is combined with the verb before the subject is joined. A parsing algorithm that always tries to combine adjacent phrases by functional application inevitably produces both analyses for the sentence. From the parser’s point of view, this is a "spurious ambiguity," because the two analyses yield exactly the same set of features. In a more complicated sentence, spurious ambiguities multiply very quickly. For example, there are four possible ways of analyzing Jussi et ole rakastanut Liisaa ‘John has not loved Lisa’, three of which differ only with respect to how much of the verb string has been assembled before it is combined with the object.

Fortunately, it is a simple matter to eliminate such spurious ambiguities. In the D-PATR system (Karttunen 1986), with the help of which this sample grammar was produced, the parser maintains a chart on which it stores all the information about possible analyses of the input phrase and its parts. It is possible to instruct the parser to apply an equivalence test that prevents redundant analyses from being entered on the chart. When an analysis is created, the parser checks whether the chart already contains an equivalent analysis for the same string of words. If one is found, the new analysis is discarded. This policy guarantees that multiple analysis trees for the same string are produced only if there is a genuine syntactic or semantic ambiguity.

5. Nonlocal Dependencies

In Finnish, an infinitival and participial clause can blend with the superordinate clause containing it in the sense that some or all of the dependents of the embedded verb can be interspersed among the constituents of the higher clause. To illustrate this phenomenon, we use an example borrowed from Maria Vilkuna’s recent study (1986). Most of the examples in this section are variants of the following simple sentence:
(28) Minä pelaan näissä tennistä.
   I play these-in tennis
   ‘I play tennis in these (clothes).’

To show what happens in infinitival complements, we use two additional verbs: ruveta ‘start’ and aikoa ‘intend’ along with the negative and tense auxiliary verbs. The verb ruveta takes complements in which the verb is the 3d infinitive in the illative case; aikoa requires the 1st infinitive:

(29) En minä ole aikonut ruveta pelaamaan näissä tennistä.
   not I have intend start play these-in tennis
   ‘I did not intend to start to play tennis in these.’

In this case, the infinitival complement, pelaamaan näissä tennistä ‘play tennis in these’, forms a contiguous phrase. However, the adverbial näissä ‘in these’ and the object tennistä ‘tennis’ can be scrambled to any of the six possible positions in the superordinate structure consisting of En minä ole aikonut ruveta ‘I did not intend to start’. Of the 42 possible variants, some are much less acceptable than others, but—because the judgments vary—it is not clear that a grammar should absolutely exclude any of them. The three examples in (30) are acceptable to a great majority of speakers. The discontinuous infinitival phrase is in bold type.

(30) En minä näissä ole tennistä aikonut ruveta pelaamaan.

Not I these-in have tennis intended start play

En minä tennistä näissä ole aikonut ruveta pelaamaan.

Not I tennis these-in have intended start play

En minä tennistä ole aikonut näissä ruveta pelaamaan.

Not I tennis have intended these-in start play

The link from an embedded infinitive to its dependents can in principle extend over an arbitrarily long chain of intervening predicates. In this respect, the phenomenon in Finnish is similar to English topicalization. The difference is that there do not seem to be any structurally defined slots—such as the topic position in English—for the elements of a broken infinitival clause; these elements can be freely interspersed in the host clause.

The grammar that we outlined in §2 cannot handle these cases. According to our definition, a partitive NP needs to find, on one side or the other, a verb that can accept it as its object. The object NP and the verb can be separated only by phrases that are themselves dependents of the same verb. This locality condition is clearly too strong. Some mechanism must be introduced to allow linkage over an unbounded chain of nested complements.

In the next section, we discuss three possible solutions to this problem,
two of which appear to be successful. The first is a replay of our analysis
of auxiliaries in §2.3, the second is based on augmenting functional ap-
lication with a type-changing rule, and the third relies on functional
application but allows functors to have “floating type.” This idea is a vari-
ant of Ronald Kaplan’s notion of “functional uncertainty” (Kaplan and
Zaenen, this volume).

5.1. Clause Union

Before we turn to the successful solutions, let us briefly consider an idea
that fails although it may appear promising in the light of the examples we
have seen so far. We can make verbs like aikoa ‘intend’ and ruveta ‘start’
combine with an infinitival complement in such a way that the resulting
combination inherits its syntactic valence from the complement. Syntac-
tically, the infinitival clause becomes fused with the superordinate clause,
but semantically the two clauses of course remain distinct: the object of
ruveta pelaamaan ‘start to play’ is interpreted as a dependent of the infini-
tive. This is the same solution that we adopted in §2.3 for auxiliary verbs.

This solution works well for auxiliary verbs, but it fails here. Auxiliary
verbs have no syntactic valence of their own and cannot be modified by
adverbial phrases; verbs that take infinitival complements are different in
this respect. For example, consider the verb pyytänyt ‘asked’ in the follow-
ing example:

(31) Liisaa minä en ole pyytänyt Jussia auttamaan.
      Lisa  I not have asked  John help

In a neutral context, most people take this example to mean ‘I did not
ask Lisa to help John’, but the interpretation ‘I did not ask John to help
Lisa’ is also possible. Because pyytänyt ‘asked’ and auttamaan ‘help’ both
require an object, it is not possible, in our framework, for them to enter
into a syntactic clause union. We cannot specify a valency with two distinct
object roles for the same verb phrase.

5.2. Type-Changing

If clause union is not a viable solution, our problem is to find a way for an
NP or an adverbial phrase to combine with a verb phrase so that the functor
acts as a dependent of a verb in a complement clause. Consider the example

(32) En minä tennistä rupea pelaamaan.
      Not I tennis start play
      ‘I will not start to play tennis.’
The desired analysis tree for the sentence is:

(33)

\[
\begin{array}{c}
\text{V} \\
\text{v} \\
\text{not} \\
\text{NP} \\
\text{I-NOM} \\
\text{tennis-PART} \\
\text{start-NEG} \\
\text{play-INF3} \\
\end{array}
\]

At the point where the partitive NP *tennistä* ‘tennis’ is combined with the verb phrase *rupea pelaamaan* ‘start to play’, the feature set of *pelaamaan* ‘play’ has been unified with the value of the path ⟨syntax vcomp⟩ in the representation of *rupea*, the negative form of ‘start’. The value of the path ⟨syntax⟩ in the feature set of *rupea pelaamaan* is thus as follows:

(34) subject — NONE

object — V
cat — V

relation — play

issue — arg1

arg2

mood — indicative

polarity — positive

tenseaspect — simple

type — declarative

case — illative

form — infinitive3

subject — semantics

[[]
syntax — object

< semantics

[[]
vcomp — NONE

If the feature set of the partitive NP *tennistä* ‘tennis’ is of the type in (11), it cannot combine with *rupea pelaamaan*. As the graph above shows, the top-level verb has no place for an object. In order for the operation to succeed, the value of ⟨argument syntax⟩ in the NP’s feature set should be (35a), not (35b), as originally given.
(35) a.  
\[
\text{vcomp} - \text{syntax} - \text{object} \\
\text{morphology} - \text{case - partitive} \\
\text{semantics} - \text{tennis} \\
\text{number} - \text{sg} \\
\text{person} - 3
\]

b.  
\[
\text{semantics} - \text{tennis} \\
\text{object} - \text{morphology} - \text{case - partitive} \\
\text{number} - \text{sg} \\
\text{person} - 3
\]

The type-raising mechanism discussed in Steedman (1986) and Dowty (1986) provides a way out of the problem. We can write a rule that takes as its input an NP of the basic type and yields a new functor by splicing the sequence ‘vcomp syntax’ into the beginning of the path that constrains the argument. This rule must of course be stated in such a way that it can apply to its own output; in the case of

(36) *En minä tennistä aio ruveta pelaamaan.*
   not I tennis intend start play
   ‘I do not intend to start to play tennis.’

the rule must be applied twice in order to make tennistä its argument’s (vcomp syntax vcomp syntax object).

In a parsing system, a recursive type-changing rule leads to an infinite loop unless its application is carefully constrained. In this case, the constraint is easy to find: the rule should be applied only if the NP in question is adjacent to a verb phrase that in fact has a complement at the right level of embedding. In the case at hand, the first application of the type-changing rule is licensed by the fact that *aio ruveta pelaamaan* ‘intend to start to play’ has the complement *ruveta pelaamaan* ‘start to play’; the reapplication in turn is permitted because the latter phrase contains the complement *pelaamaan* ‘to play’. At this point, the rule can no longer be applied because the verb *pelaamaan* ‘play’ does not have a slot for a complement clause. The output of the second round of type-changing is then successfully combined with the adjacent verb phrase *aio ruveta pelaamaan* ‘intend to start to play’.

The constraint needed to prevent infinite recursion of the type-changing rule is difficult to express in a formalism that relies on unification. We have worked out a solution to this problem but we will not discuss it here because the details are rather complicated and because we regard the approach discussed in the next section more attractive.
5.3. Functors with Floating Type

In this volume, Kaplan and Zaenen describe a novel solution for the problem of linking an initial *wh*- or topic phrase to a verb in an embedded clause. This solution is in some respects very similar to the idea of type-changing discussed in the preceding section, but it is conceptually more elegant, even though it requires a substantial change in the way unification is implemented in most current systems (including D-PATR). In this section, we describe the essential features of the proposal within our framework; the original version is stated in terms of LFG.

So far we have assumed that functors have a fixed type, for example, a partitive NP as defined in §2.1 combines only with arguments whose feature set contains a suitable value for the path ⟨syntax object⟩. The type-changing rule discussed in the preceding section generates from it—by repeated applications—a set of functors whose type specifications constitute a series:

(37) ⟨syntax object⟩
    ⟨syntax vcomp syntax object⟩
    ⟨syntax vcomp syntax vcomp syntax object⟩
    ...

In intuitive terms, this is simply a way of stating that a partitive NP is the object of some verb in the adjacent verb phrase that can be reached by way of one or another path in this set. The basic insight in Kaplan’s notion of functional uncertainty is that this kind of open-ended disjunction can be coded directly in the formalism. If we let path specifications be regular expressions, the above set of paths can be represented by a single specification:

(38) ⟨[syntax vcomp]* syntax object⟩.

This is clearly more elegant from a technical point of view than type changing, and Kaplan and Zaenen argue that, in the context of LFG, it is also motivated by linguistic evidence. The use of regular expressions in equations makes it possible to associate initial topic and *wh*- phrases to predicates directly in functional structure without reference to phrase-structure trees. Kaplan and Zaenen discuss several examples that are difficult to handle in terms of constituent structure and show how the data can be described more adequately with the new technique.

In our framework, the proposal eliminates the need to use type-changing to describe nonlocal dependencies. A partitive NP can combine directly with a verb phrase that has a place for an object at some level of embed-
ding. In effect, NPs and adverbial phrases become functors with "floating type." The combination of such a functor with an argument can yield more than one result as all of the alternative ways of successful unification are tried out. This is the case in the following example:

(39) Ketä Liisaa pyydettiin auttamaan?
    who Liisa they-asked help

Although the preferred interpretation is clearly 'Whom did they ask Lisa to help?', the reading 'Whom did they ask to help Lisa?' cannot be excluded. Consequently, when the NP Liisaa is combined with the verb phrase pyydettiin auttamaan 'they asked to help', two instances of Liisaa pyydettiin auttamaan are produced: one for the path (syntax object), the other for the path (syntax vcomp syntax object). The interrogative phrase ketä 'who' will then assume whichever object role remains unfilled as it combines with the two verb phrases.

The use of regular expressions in path specifications raises many formal and computational issues that require further study (Johnson 1986). A computer implementation of unification for this type of grammar formalism has been developed by Kaplan.

6. Function of Word Order

The purpose of the preceding section is to sketch an account of Finnish syntax that reflects the intuition that hierarchical structure and word order have relatively little to do with the syntactic roles of constituents in this language. The analyses that our grammar produces are feature structures in which the syntactic dependents of a verb are labeled in a traditional way; a part of that representation is intended to serve as input to semantic interpretation. However, many important aspects of meaning, such as quantifier scope, have intentionally been left unresolved. Because we have focused on the problem that constituents can appear in many different orders without changing the grammatical relations they express, we have not paid any attention to the fact that the alternative orderings of the same constituents differ with respect to some aspects of meaning, such as quantifier scope and discourse function.

Although the derivation of a categorial analysis for a sentence can be recorded in the form of the tree, the analysis trees do not represent any aspect of sentence structure. As we have developed it so far, the grammar simply assigns to a sentence an unordered set of features that does not even express the linear order of constituents. This does not mean that constituent structure cannot be represented in a categorial analysis. On the contrary,
because we distinguish between forward and backward application of functional composition, it is a simple matter to augment the system in such a way that some designated attribute will have as its value a list that consists of all the constituents in the order in which they appear. It is also possible to construct a more hierarchical structure; in fact, we could easily arrange the feature system in such a manner that a phrase-structure tree of a traditional sort is constructed as a side effect of applying categorial rules.

This state of affairs has certain advantages that we intend to exploit in future work. By not being committed to any constituent structure in advance, we are free to build as little or as much structure as the data about discourse meaning warrant. Although there are many studies of word order in Finnish (Hakulinen 1976, Heinämäki 1980, Karttunen and Kay 1985, Vilkuna 1986), they have produced very little insight into the principles that are involved. What little is known indicates that there is a close connection between word order and intonation. There appear to be no phenomena that support the assumption that hierarchical structure is involved in determining conversational meaning; it is commonly assumed that conversational structures are flat and consist of only a few elements. Vilkuna (1986), for example, describes discourse functions in terms of a simple template

(40) $P_2 P_1 V X$,

where $P_2$ and $P_1$ are positions in front of a finite verb and $X$ covers the rest of the sentence. It is possible that a special role must also be assigned to sentence-final position (Hakulinen 1976). The $P_1$ role has also been called theme or topic; the optional $P_2$ element could be called contrastive focus. As Vilkuna points out, the assignment of constituents to these roles is not constrained by category or syntactic function.

The pivotal role of the finite verb in discourse structure is one of the few facts about Finnish word order on which there is common agreement. Another well-attested phenomenon is the contrastive focus in sentences like

(41) **Tennistä Liisa ei aio pelata.**

<table>
<thead>
<tr>
<th>Tennis</th>
<th>I</th>
<th>not intend play</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_2$</td>
<td>$P_1$</td>
<td>$V$</td>
</tr>
</tbody>
</table>

'Tennis Lisa does not intend to play.'

It is interesting to note that the $P_2 P_1 V$ order may be inverted just in case the finite verb is the negative auxiliary:

(42) **Ei Liisa tennistä aio pelata.**

<table>
<thead>
<tr>
<th>Not Lisa</th>
<th>tennis</th>
<th>intend play</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V$</td>
<td>$P_1$</td>
<td>$P_2$</td>
</tr>
</tbody>
</table>
The initial position of the negative verb implicates that the opposite assumption is being contradicted; in other respects the two variants are very similar. In either case, the normal intonation peak naturally falls on *tennistä* ‘tennis’, which is perceived as the focus of negation.

The purpose of these inconclusive remarks is simply to point out that the kind of constituent structure that is generated by phrase-structure rules of the type \( S \rightarrow NP \ VP, \ VP \rightarrow V \ NP \), and the like appears to be just as irrelevant for the description of conversational meaning in Finnish as it is for the account of grammatical relations. There is no reason to believe that a categorically based description is inferior to a transformational or a phrase-structure grammar with respect to expressing regularities about word order and conversational meaning.

7. Conclusion

This paper is an exploration of unification-based categorial syntax. We have constructed a grammar for a fragment of Finnish that accounts for word-order variation in simple sentences using only functional composition. This is possible because, in departure from CG tradition, we treat subjects, objects, and other dependents of verbs as functors. By adopting Kaplan’s idea of functional uncertainty (floating types), we extend this solution for cases in which the elements of an infinitival clause are interspersed among the constituents of the main sentence. This phenomenon can also be described by introducing a rule of type-changing, but we prefer Kaplan’s technique. In the final section, we point out that standard phrase-structure rules cannot account for either grammatical relations or discourse structure in Finnish. A radical lexicalist approach is a better alternative.