CHAPTER VIII

CONSTITUENT STRUCTURES

ABSTRACT

Some proposals from the literature for assigning constituent structures
to the expressions produced by a Montague grammar are shown to violate the
framework. A treatment of the syntax of the PTQ fragment is presented which
assigns constituent structures to the produced expressions and which meets
the requirements of the framework. Furthermore a restricted set of syntactic
operations is introduced for the description of the syntactic rules.
1. STRUCTURE - WHY?

The syntactic rules of PTQ make a primitive impression in comparison to the kind of rules used in transformational grammars. A first point of difference is that the syntactic operations make no reference to the constituent structure of the involved expressions. A second one is that the syntactic operations are described without any formalism: the desired effects are described by English sentences. On the one hand English is a rather poor tool since in this way the description of the syntactic operation can hardly use any abstract syntactic information. At the other hand it is a very unrestricted tool, since it allows any operation that can be described in the English language. Since the earliest times of Montague grammar, it has been tried to bring the syntax of Montague grammar closer to that of transformational grammar. This would open the possibility to incorporate syntactic knowledge from transformational grammars in Montague grammar, and to discuss the differences. In this chapter I will present the first steps of an approach which makes the syntax of Montague grammar less primitive: by developing a formalism for the formulation of the syntactic rules, and by introducing constituent structures in the syntax.

An example of the kind of structure used in transformational grammars is given in figure 1. The tree is not taken from any proposal in that field (then several details would be different), but it can be used to illustrate what kind of information is provided by such trees. The words attached to the end nodes of the tree yield, when read in the given order, the sentence of which the tree represents the constituent analysis. Constituents are groups of words which have a certain coherence. This appears for instance from the fact that it is rather easy to replace a constituent of a sentence by another group of words, whereas this is not the case for arbitrary groups of words from the sentence. The tree in figure 1 indicates what the constituents of the sentence are: all words of a certain constituent are connected to the same node in the tree. This node is labelled by a symbol: the name of the category of the constituent. Thus the tree gives the information that each word is a constituent, and that e.g. a unicorn is a constituent, whereas seeks a is not.
A first argument for the introduction of constituent structures in the syntax of Montague grammars is that it would make it possible to incorporate ideas, or even particular rules, from transformational grammars into Montague grammar. I will not try to sketch the role such structures play in the syntax of transformational grammars; the reader should accept that constituent structures have proven their usefulness. A second argument is that, even without the aim of incorporating ideas from transformational grammar, it is useful to have structural information available about the expressions dealt with. An example, based upon a phenomenon from the PTQ grammar, is the following (ParEe 1973).

Rule $S_{11a}$ from PTQ, the rule for verb-phrase conjunction, produces the IV-phrase

1. *walk and talk.*

Rule $S_8$ produces from (1) and the verb *try* to the IV-phrase

2. *try to walk and talk.*

From the term *John* and the IV-phrase (2) we can produce, according to rule $S_4$, the sentence

3. *John tries to walk and talk.*

Another derivation is to produce first (using $S_8$)

4. *try to walk.*

Next we produce (5), using $S_{11}$.

5. *try to walk and talk.*
Application of $S_4$ to (5) yields (3), but the correct form of a sentence with the intended conjunction would be

(6) John *tries to walk and talks.*

In order to correct rule $S_4$ for this, it is useful to distinguish the conjunction of *try to walk* and *talk* form the IV phrase *try to walk and talk*. So it is useful to assign structure to the strings (5) and (6).

This second argument shows that it is useful to have some kind of structure available, not that it has to be the kind of structures used in transformational grammars. As has been shown by Friedman (1979), the kind of problems mentioned above can be dealt with by un-labelled trees. A completely different kind of syntactic structures is used in Janssen 1981b, where the present framework is combined with the structures used in Dik's 'functional grammar' (Dik 1978, 1980). However, the kind of structures I will consider in this chapter are constituent structures of the kind described above.

2. THEORETICAL ASPECTS

2.1. Trees in Montague grammar

The rules of a Montague grammar determine how basic syntactic units are combined to larger ones. Such production processes can be represented by a tree. The tree for the de-dicto reading of John *seeks a unicorn* is given in figure 2.

```
      John seeks a unicorn
        |      /
      John  seek a unicorn
        |      |
      seek  a unicorn
        |      |
      a     unicorn
```

Figure 2 tree from Montague grammar

Such trees are representations of derivational histories. For this reason Partee (1975) compares them with the T-markers from transformational grammar, and not with the produced trees themselves. In transformational grammars trees are produced, and if one wishes to compare the approach of Montague grammar
to the approach of transformational grammar, then one has to compare trees. Trees like the one in figure 2 are the only trees one finds in publications of Montague. Therefore one is tempted to compare such trees with the trees obtained in transformational grammars.

The tree in figure 2 is not of the form of the trees of transformational grammars. The main difference is that in transformational grammars the nodes are not labelled with expressions, but with category symbols (except for the end-nodes). Therefore one considers the tree from figure 2 as an unusual representation of the tree given in figure 1. Then the tree from figure 2 is taken as the syntactic structure assigned to the sentence by the PTQ grammar. Proceeding in this way, using the only trees available in Montague grammars, it becomes possible to compare the structures in Montague grammar with the structures in transformational grammars. This view on syntactic structure in Montague grammar can be found in work of several authors. In the next chapter we will see that PARTEE (1973) has compared the relative clause formation in Montague grammar and in transformational grammar by comparing trees like the one in figure 2, with those of transformational grammars. This way of discussion was followed up in by BACH & COOPER (1978). The same approach can be found in COOPER & PARSONS (1976). They describe a transformational grammar that is claimed to be equivalent with the PTQ system. The basic rules of their transformational grammar produce (roughly) the same trees as the derivational histories of PTQ.

If one just compares the trees in the two approaches one soon will find great differences, and problems arise if one wishes to take the trees from Montague grammar as serious proposals for the syntactic structure assigned to a sentence. Consider the tree for the de-re reading of John seeks a unicorn, given in figure 3, or alternatively the one in figure 4.

![Figure 3: de-re reading](image1.png)  ![Figure 4: variant of figure 3](image2.png)
This tree cannot be taken as a serious analysis of the constituent structure of the sentence since it does not even fulfill the weakest requirement: that the lexical material is presented in the correct sequence.

Cooper has developed a variant of Montague grammar in which no quantification rules are used, and which seems to eliminate the problem just mentioned. I give an example from COOPER 1978 (the idea originates from COOPER 1975). Consider the tree in figure 2. The interpretation of this tree follows its structure. The lexical items are interpreted first, and next the interpretations of larger constituents are formed. The usual interpretation yields the de-dicto reading, roughly presented as \( \text{John'}(\text{seek}'(a \text{ unicorn'})) \). The de-re interpretation is obtained by means of a mechanism which gives the translation of the \text{unicorn} wide scope. This mechanism simply is a storage mechanism which allows the translation of the noun phrase \( a \text{ unicorn} \) to be stored, putting a variable placeholder in the regular translation. The stored translation of a unicorn is carried up the tree, until it can be retrieved at a suitable point where quantifying in is allowed. The store is a set of pairs consisting of an interpretation of a term and a variable. The way of processing is as follows.

\[
\begin{align*}
\text{a unicorn} & \rightarrow \langle \lambda P(x_0), \langle a \text{ unicorn'}, x_0 \rangle \rangle \\
\text{seek a unicorn} & \rightarrow \langle \text{seek}'(\lambda P(x_0)), \langle a \text{ unicorn'}, x_0 \rangle \rangle \\
\text{John sees a unicorn} & \rightarrow \langle \text{John}'(\^\text{seek}'(\lambda P(x_0))), \langle a \text{ unicorn'}, x_0 \rangle \rangle \\
& \quad \text{retrieve from store, yielding} \\
& \quad \langle a \text{ unicorn'}(\lambda x_0(\text{John}'(\^\text{seek}(\lambda P(x_0))))), \emptyset \rangle.
\end{align*}
\]

Cooper is not very explicit about the details of his proposal, and therefore it is difficult to evaluate it. Nevertheless, I have serious doubts about the acceptability of his proposal in any approach which accepts the principle of compositionality of meaning. The reason for this is as follows. The phrase \( \text{seek a unicorn} \) has two parts: \( \text{seek} \) and \( a \text{ unicorn} \). The contribution of the latter part to the meaning of the whole phrase consists in three components, one of them being the variable \( x_0 \). We have formalized meanings as abstract functions (intensions), and the symbol \( x_0 \) is not an element in this formalization. I assume that Cooper does not intend to define meanings as something which has the symbol \( x_0 \) as a component. So the mechanism does not build meanings from meanings, and therefore it violates the principle of compositionality of meaning. A more explicit description of a storage mechanism is given in PARTEE & BACH (1981); that
proposal is discussed in LANDMAN & MOERDIJK (1983), where is shown that related objections apply.

The above discussion shows that we cannot get rid of trees like the one given in figure 3 by using Cooper storage. This has the following consequence. If one takes the tree representing the derivational history of a sentence in a Montague grammar to be the syntactic structure assigned to that sentence, then one has to conclude that in certain cases they are unacceptable as constituent structures. This is a practical reason against identifying the derivational histories with constituent structures. As will be explained below, there are also algebraic reasons against it.

2.2. Algebraic considerations

In our framework the syntax is an algebra, i.e. a collection of carriers with operations defined on them. An algebra can be defined in many ways. For instance, one can enumerate all the elements of each carrier, and state what the operators are. But we have developed a more efficient way of defining an algebra: state what the generators and the operators are. In this way with each element of the algebra (at least) one derivational history can be associated. Such derivational histories are important for the semantics, because this process is mirrored when building the corresponding meanings. We have met several examples where the choice of a certain generated algebra was determined by semantic considerations. If we consider only the syntactic side of the situation, the generation process is just some method to define the algebra. If we would replace a given definition by another definition of the same algebra, the elements and the operators would remain the same. More in particular, an element of an algebra by itself does not have a derivational history. Only if one has additional information concerning the way in which the algebra is defined, it becomes possible to associate with an element some derivational history, and with the algebra itself an algebra of derivational histories (a term algebra). The operators of a syntactic algebra are functions defined on the elements of that algebra, and since the information how the algebra is defined, cannot be read off from these elements, the operators of the syntactic algebra cannot interfere with derivational histories. In section 2 I argued that we need syntactic structures in order to design more sophisticated rules. As argued above, the syntactic rules are completely independent of such histories. Hence we cannot consider derivational histories to be the
structures we are looking for. This means that the only available trees cannot be used as a kind of syntactic structures. So the conclusion has to be that the PTQ grammar assigns no structure at all to the expressions it deals with.

If one wants the elements of an algebra to have a structure, then these elements should be structures! So in order to obtain a syntactic structure for the expression of a Montague grammar, this grammar should produce structures: trees, or, equivalently, labelled bracketings. This brings us to an approach dating from the first years of Montague grammar: PARTEE 1973. That proposal follows the sound approach to structure in Montague grammar. It distinguishes between the structure of the derivational history and the structure of the produced element itself. A remark about the relevance of distinguishing these two levels in a grammar for natural language can already be found in CURRY (1961), who calls the level of history 'tectogrammatics', and the level of produced expressions 'phenogrammatics'. DOWTY 1982 claims that rather different languages (such as Japanese and English) may have the same tectogrammatic structures, whereas the differences between the languages are due to phenogrammatical differences. This idea can also be found in LANDSBERGEN 1982, where it constitutes the basic idea for a computer program for automatic translation. In figure 5 the two kinds of structure are presented for the de-re reading of John seeks a unicorn: the trees within the squares are the constituent structures produced by the grammar, and the tree consisting of double lines with squares as nodes is the tree representing the derivational history.

2.3. Practical differences

Above I argued on algebraic grounds for distinguishing the structure an element has, from the derivational history assigned to it in some generative definition of the algebra. A practical aspect of this distinction is that there are completely different criteria for the design of these two kinds of structures. The derivational history is mapped homomorphically to the semantic algebra and determines the meaning of the expression. Semantic considerations play a decisive role in the design of the operators, and considerations concerning efficiency of definition determine the choice of the generators. The inherent constituent structure of the expressions is determined by syntactic considerations, e.g. the role such a structure has to play in the description of the syntactic effect of an operation. These
Figure 5  One derivational history containing many constituent structures
two different kinds of arguments may yield different kinds of structures. Below I will give some examples which show that the derivational history may sometimes differ considerably from what an acceptable constituent structure might be.

a) The PTQ rule $S_{16}$ produces e.g. John runs out of John and He$_1$ runs. This is not an acceptable syntactic structure since it contains at an end node a word that does not occur in the sentence (cf. the discussion concerning figure 3).

b) In the grammar for questions by Groenendijk & Stokhof (1981), there is a rule which substitutes a common noun into phrases of a certain kind. Thus which man walks is produced out of man and which one walks. Here the same argument applies as for the quantification rule of PTQ: it contains at an end node an argument that does not occur in the sentence.

c) In Hauser (1979b) another variant is presented of the substitution of a common noun for an occurrence of one in some phrase. Here the same conclusion holds.

d) Bach (1979a) presents rules which produce persuade Bill to leave out of Bill and persuade to leave. The operation which performs this task, called 'right-wrap', is a kind of substitution operation. It disturbs the sequence of words, and therefore it gives rise to a derivational history in which the order of the words does not correspond with the order of the words in the phrase. Therefore the derivational history is different from any possible syntactic structure.

e) Dowty (1978) gives a very elegant categorial treatment of phenomena which are traditionally treated by transformations. Examples are dative movement and object deletion. His rules shift serve from the category DTV (takes a dative and a term), to the category TTV (takes two terms), and next to TV and IV. This history is presented in figure 6. As far as I know, such a structure has not been proposed in transformational grammars, which is an indication that there is no syntactic motivation for this structural analysis. All steps in this production process are semantically relevant, and I consider it as a prime example of a semantically motivated elegant design of a derivational history.
3. TECHNICAL ASPECTS

3.1. Introduction

In this section I will sketch some tools which are useful in a version of Montague grammar in which the syntax produces structured expressions. The desire to provide handsome tools for a certain limited purpose leads to restricted tools (all-purpose tools are usually not very handsome: I would not like to describe a language by means of a Turing machine). So, whereas I do not have the aim of Partee ('defining as narrowly as possible the class of possible grammars of natural languages' (PARTEE 1979b, p.276)), the practical work is closely related. The tools I will use originate mainly from Partee (ibid); in the details there are some differences. It is not my aim to develop a complete theory about structured syntax, but I will use the opportunity to make some theoretical and practical remarks about the available techniques. For more ambitious proposals which use the same approach to structure, see BACH 1979b, PARTEE 1979a, 1979b, and LANDMAN & MOERDLijk 1981.

The basic change I will make here in comparison with previous chapters, is a change of the algebra on the background, which is always assumed when we define a generated algebra. In the previous chapters this was mostly the algebra consisting of all strings over the alphabet involved with concatenation as operator. In the present chapter this background algebra is replaced by one which consists of all trees, labelled in an appropriate way, and which has the basic operations which will be described in the sequel.
3.2. Operations on trees

It is not very convenient to describe operations on trees by means of English sentences. Following Partee, I describe such operations as the composition of a few basic ones. These are described in the sequel.

The operation root gives a new common root to the members of a list of trees. The new root is labelled with a given category name. Let α and β denote trees, and let ((α, β), IV) denote the list consisting of these two trees. The effect of root(((α, β), IV)) is that the roots of the trees α and β are connected with a new root, labelled IV, see figure 7.

![Figure 7: root ((α, β), IV)](image)

The operation insert substitutes a tree for a given node in some other tree. Let us accept the phrase 'first he_2 in (α)' as a correct description of the node marked with x in tree α, see figure 8. Then the effect of insert (β, first he_2 in (α)) is given in figure 9. A single word is considered as the denotation of a tree consisting of one node, labelled with that word. So the root operation can be applied to it. The effect of root(and, Con) is shown in figure 10.

![Figure 8: situation](image)  ![Figure 9: insert(α, β, x)](image)  ![Figure 10: root(and, Con)](image)
These two operations for tree manipulation, together with operations for feature manipulation and index manipulation, suffice for the treatment of the PTQ fragment. For larger fragments other operations might be required. An example is 'everywhere-substitution', which has the effect of substitution for all occurrences of a variable. This effect cannot be obtained by means of a repetition of the insert operator since one and the same tree cannot be at the same time the daughter of different nodes. So everywhere-substitution requires a copy operation which might be added as a primitive operation. PARTEE 1979b has no copy operation, but considers everywhere-substitution as a basic operation (we do not need everywhere-substitution since we deal with $S_4^{4,n}$ by means of an operation on features).

As an example I present the rule for verb-phrase conjunction. If we stay close to PTQ, it gets the following form, and yields the result given in figure 11.

$$S_{11} : IV \times IV \rightarrow IV$$

$$F_{11} : root((a, and, b), IV).$$

One might prefer to give the connective and a categorical status in the syntactic structure; the status of a connective. Then the operation could read as follows, yielding the result given in figure 12.

$$S_{11} : IV \times IV \rightarrow IV$$

$$F_{11} : root((a, root(and, Con), b), IV)$$

![Figure 11: root((a, and, b), IV)](image)

![Figure 12: root((a, root(and, Con), b), IV)](image)

3.3. Features and lexicon

Rule $S_4$ of PTQ tells that the subject-verb agreement in a sentence is obtained by replacing the first verb by its third person singular present. This is not an explicit formulation of what the effect of the rule is supposed to be. In an explicit form it would say that the verb run is to be replaced by runs and that try to (in PTQ a single word with a space inside)
is to be replaced by *tries to*. Rule $S_4$ can have its short readable form only since it is not explicit about such details. In order to obtain an explicit syntactic rule which is not full with details, we have to abstract from the inflection behaviour of the individual verbs. So it is useful to let the syntactic rules deal with more abstract lexical elements. By incorporating features in Montague grammar, rule $S_4$ may for the PTQ fragment simply attach the features like *present* and *third person singular* to an elementary form of the verb without being concerned with the effect of these details for every individual verb. The information about morphological behaviour of the verb can be given in the lexicon or in a separate morphological component.

Features originate from transformational grammar. They were used, as far as I know, for the first time in Montague grammar by Groenendijk & Stokhof 1976 for a phenomenon like the one above. Features are also useful if one incorporates transformations into Montague grammar. Partee 1979b gives several examples of transformations which require features; an example is the Subject–Aux inversion process for questions which requires isolation of the tense morpheme.

As an example of the use of features I give a reformulation of rule $S_4$ using features. Of course, the rule has in other respects the same shortcomings as the original PTQ rule, but it is fully explicit now.

$$S_4 : T \times IV \rightarrow S$$
$$F_4 : \text{add features}((\text{pres}, \text{sing } 3), \text{first verb in } (a));$$
$$\text{root}(a, S), S).$$

The details of the regular formation of the word forms can be given on a separate morphological component, whereas details about irregular word forms can be given in the lexicon. So the function *verbform* in the morphological component will be such that

$$\text{verbform}((\text{pres, sing } 3), a) = \text{as} \quad (\text{e.g. walks})$$
$$\text{verbform}((\text{past, sing } 3), a) = \text{aed} \quad (\text{e.g. walked}).$$

The morphological component also contains a function *pronomen* such that

$$\text{pronomen}(\text{sing } 3, \text{acc, masc}) = \text{him}$$
$$\text{pronomen}(\text{sing } 3, \text{nom, neut}) = \text{it}.$$

Besides morphological details, the lexicon also contains the information which features are inherent to the word (e.g. *John* always bears the feature
and information about kinds of features for which the word may be
specified (e.g. John may not be specified for tense).

On the basis on the above considerations I define a lexical element as
a list consisting of the following five components.
1. a string being the basic form of the word
2. a category symbol, being the category of the lexical element
3. a list of inherent features
4. a list of kinds of features for which the lexical element can be speci-
   fied.
5. a description of the procedure for making derived forms of the word.

The above definition says that a lexical element is denoted by a list
of five elements, of which some are lists themselves. We already introduced
a notation for lists. Let furthermore ( ) denote an empty list. The examples
of lexical elements presented below are somewhat simplified with respect to
PTQ since they only consider past and present tense.

("John",T,(masc,,sing3),(),wordform: "john")
("walk",IV,(),(tense,pres),wordform: verbform((tense,pres)"walk"),)
("run",IV,(),(tense,pres),

if value of tense = past then wordform: "run"
else wordform: verbform((pres,sing3),"run")
).

Up till now we only considered kinds of features which are well known.
But nothing in the feature definition prohibits us to define unusual ones.
We might define a feature kind 'mainverb' with values # and -. The instruc-
tions for introduction or deletion of this feature can be the same as the
instructions for Bennetts # mark which indicates the main verbs of a phrase
(BENNETT 1976). In this way we can use the features as syntactic markers.
Following Partee, I would not like to do so. Features are introduced for
isolating morphological phenomena, not for syntactic marking. So I would
like to restrict features to morphological relevant ones, just as PARTEE
(1979b) proposed. This restriction requires, however, a formal definition
of this notion (Parlee gives no definition).

The notion 'morphologically relevant feature' is clearly word depen-
dent. The case feature is relevant for he but not for John. So we might
call a feature morphologically relevant if it is relevant for at least some
word in the grammar. But what does this notion mean? Something like that
the feature influences the form of the word? It is to be required further-
more that this influence can be observed in real sentences: it is not enough
that it occurs in some morphological rule since this leaves open the possi-
bility of a fake feature which influences the form of some 'external' word
that does not occur in a produced sentence. We want a feature to create an
opposition of wordforms in the produced language. Based upon these consider-
ations I would define the notion as follows.

3.1. DEFINITION. A feature $F$ is called morphologically relevant in grammar
$G$ if the following two conditions are satisfied.
1. There is a sentence $S_1 \in L(G)$ containing a lexical element $W$ which bears
feature $F$ and which has wordform $W_1$.
2. There is a sentence $S_2 \in L(G)$ containing an occurrence of $W$ which does
not bear feature $F$ and which has wordform $W_2$ where $W_2$ is different from
$W_1$.
3. END

Note that this definition uses quantification over the sentences in the
language $L(G)$. This quantification makes the notion 'morphologically relev-
ant' to an undecidable notion. Suppose a list of syntactic rules, a lexi-
con containing features, and a list of morphological rules is given. Then
one might try to show that a feature is not morphologically relevant by
producing a lot of sentences and checking the conditions. However, one
never reaches a stage that one can say for sure that such a feature is un-
acceptable. A formal proof is given in the following theorem.

3.2. THEOREM. There exists no algorithm which decides for all grammars $G$
and feature $F$ whether $F$ is morphologically relevant in $G$.

PROOF. Suppose that such an algorithm would exist. Then this would give
rise to a decision procedure for algebraic grammars, as will be shown below.
Let $G$ be an arbitrary algebraic grammar, with distinguished sort $S$, and
suppose $L(G)$ is a language over the alphabet $\Sigma$. Let $a$ be an arbitrary
string over this alphabet, and let $w \in A$ be the first symbol of $a$. Let
$w' \in A$ be a new symbol, and $F$ a new feature not occurring in $G$. Define the
wordform of $w$ when bearing feature $F$ as being $w'$. Extend now grammar $G$ to
$G'$ by adding the following rule:

\begin{align*}
R: & S \rightarrow S \\
R(a) = & a' \text{ where } a' \text{ is obtained from } a \text{ by attaching } F \text{ to } w \\
R(\phi) = & \phi \text{ if } \phi \text{ is not equal to } a.
\end{align*}
The only way to introduce $\omega'$ in some expression of $L(G')$ is by means of this new rule $R$. Hence $F$ is morphologically relevant in $G'$ if and only if $a' \in L(G')$. From the definition of $R$ it follows that $a' \in L(G)$ if and only if $a \in L(G)$. So if it would be decidable whether $F$ is morphologically relevant, it would be decidable whether $a$ is generated by grammar $C$. Since $L(G)$ can be any recursively enumerable language, this question is undecidable.

3.2. END

The undecidability of the notion 'morphologically relevant' has as a consequence that it can not be considered as a formal constraint, and that it cannot be incorporated in the definition of the notion 'grammar'. This does not mean that the property is worthless. It could play about the same role as the well-formedness constraint, being an important practical guideline for designing and evaluating grammars.

3.4. Queries for information

In syntax one often uses information about the grammatical function of words and groups of words. The grammatical tradition has constituted names for most of these functions, e.g. mainverb, subject and object. That the information for determining these functions is present in the syntactic structure assigned to them, has already been stated in CHOMSKY 1965. He defines the subject of a sentence as the NP which is immediately dominated by the main $S$ node. In spite of this approach to grammatical functions, the tradition of transformational grammar never uses such information explicitly. PARTEE 1979b proposes to incorporate this information in Montague grammar and to make explicit use of it in the syntactic rules.

On the question what the main verbs of a sentence are, an answer like $\rho \omega \rho$ is not good enough since that verb might occur more than once. An answer has to consist of a list of occurrences of verbs; or formulated otherwise a list of nodes of the tree which are labelled with a verb. Functions used to obtain syntactic information such as mainverb are functions from trees to lists of nodes of that tree. The first use of functions of this kind in Montague grammar is given in FRIEDMAN 1979. Such functions are called queries by KLEIN (1979); PARTEE (1979b) uses the name properties for a related kind of functions. The different name covers a different approach to such functions. It is a property of each individual occurrence of a verb to be a mainverb or not so; hence a property is a boolean valued
function, but a query is not. Since it is not convenient to use properties in syntactic rules, I use queries.

PARTEE (1979b) defines queries by means of rules parallel to the formation rules of the syntax. This has as a consequence that she in fact performs induction on the trees which represent derivational histories. Thus properties of derivational histories can be defined by means of her queries. It allows, for instance to define a query which tells us what the terms are which are introduced by means of a quantification rule. This query which I call 'substituted terms', can be defined as follows:

1. add to rule $S_{14,n}$ the clause
   \[\text{substituted terms } (S_{14,n}(\alpha,\beta)) = (\alpha) \cup \text{substituted terms } (\beta)\]
2. do the same for the other quantification rules
3. add to the other rules the clause
   \[\text{substituted terms } (S_i(\alpha,\beta)) = \text{substituted terms}(\alpha) \cup \text{substituted terms}(\beta)\]

Since we do not consider the derivational histories as representations of syntactic structures, we do not want information about the derivational history to be available in the syntax. Therefore I will not define queries in this way.

FRIEDMAN (1979) defines queries separately from the rules. She defines them for all constituent structures by means of a recursive definition with several clauses; each clause deals with a different configuration at the node under consideration. So Friedman performs induction on the constituent trees, and not on the derivational histories. Consequently, the query 'substituted terms' cannot be defined in Friedman's approach. In principle I will follow Friedman's method, but some modifications are useful.

Friedman's method has a minor practical disadvantage. If one adds a rule to the grammar, then at several places the grammar has to be changed: not only a rule is added, but all query definitions have to be adapted. In order to concentrate all these changes on one place in the grammar, I will mention the clauses of the definitions of a query within the operation which creates a new node, so within the operation root. In this way it is for each node determined how the answer to a query is built up from the answers to the query at its subnodes. If a root operation contains no specifications for a query, this is to be understood as that the answer to the query always consists of an empty list of nodes. As an example, I present rule $S_{11a}$ in which a clause of the query mainverb is defined.
\[ S_{11} : IV \times IV \rightarrow IV \]

\[ F_{11} : \text{root}((a, \text{root}(\text{and}, \text{Con}), b), IV, \text{mainverbs} = \text{mainverbs}(a) \cup \text{mainverbs}(b)) \].

The basis of the recursion for a query is formed by its application to a node only dominating an end node of the tree. Then a query yields as result that end node. So is the query mainverbs is applied to the root of the tree in figure 13, then the result is that occurrence of \textit{run}.

\[ \begin{align*}
\text{IV} \\
\text{run}
\end{align*} \]

\textit{Figure 13: basis of recursion}

4. PTQ SYNTAX

Below I will present a syntax for the PTQ fragment. The purpose of this section is to provide an explicit example of what a Montague grammar in which structures are used, might look like. It is not my aim to improve all syntactic shortcomings of PTQ; only some (concerning conjoined phrases) are corrected. For more ambitious proposals see PARTEE 1979b and BACH 1979.

In the formulation of the rules, several syntactic functions and operations will be used. Below the terminology for them is explained, thereafter they will be described.

1. \textit{Queries}
Functions which have a tree as argument and yield a list of nodes in the tree. They are defined within the root operations.

2. \textit{Primitive functions}
Functions of several types which yield information, but do not change anything.

3. \textit{Primitive operation}
Operations of several types which perform some change of the tree or lists involved.

4. \textit{Composed operations}
Like 3, but now built from other operations and functions.
QUERIES

Mainverbs
Yields a list of those occurrences of verbs in the tree which are the
mainverbs of the construction.

Headnouns
Yields a list of those occurrences of nouns, pronouns and proper names
which are the heads of the construction.

PRIMITIVE FUNCTIONS

\textbf{Index of} \((w)\)
yields the index of the word \(w\) (provided that \(w\) is a variable)

\textbf{First of} \((l)\)
yields the first element of list \(l\).

\textbf{All occurrences of} \((he_N, t)\)
yields a list of all occurrences of \(he_N\) in tree \(t\).

\textbf{Gender of} \((w)\)
yields the gender of word \(w\), and of the first word of \(w\) if \(w\) is a list.

\textbf{Is a variable} \((w)\)
determines whether term \(w\) is a variable (of the form \(he_N\)).

PRIMITIVE OPERATIONS

\textbf{root}((t_1, \ldots, t_n), C, \text{query: ...})

Creates a new node which is the mother of the trees \(t_1, \ldots, t_n\).
This new node is labelled with category symbol \(C\).
For all queries a clause of their recursive definitions is determined:
either explicitly, or implicitly (in case the query yields an empty list).

\textbf{Add features} \((f, l)\)
Attaches to all elements of list \(l\) the features in feature list \(f\).

\textbf{Delete index} \((n, l)\)
Deletes index \(n\) from all elements in list \(l\).

\textbf{Replace index} \((1, m)\)
Replaces the index of all variables in list \(l\) by index \(m\).

\textbf{Insert} \((r, t)\)
Replaces node \(r\) by tree \(t\), thus inserting a tree in another tree.

\textbf{Union} \((l_1, l_2)\)
Yields one list, being the concatenation of lists \(l_1\) and \(l_2\).
COMPOSED OPERATIONS

Term substitution \( (t, n, r) \)

Substitutes term \( t \) in tree \( r \) for the first occurrence of \( h_n \). The operation is defined by:

if \( t \) is a variable (\( t \))
   then replace index (all occurrences of \( h_n \), index of \( t \))
else insert \( t \), first of (all occurrences of \( h_n \), \( r \))

   define list= all occurrences of \( h_n \), \( r \)
   delete index(list)
   add features ((sing,pers3,gender of (first of (main nouns (t))), list).

End definition.

Below the rules for the FTQ-fragment are given with exception of the rules for tense. Their formulation resembles the formulation they would get in an ALGOL 68 computer program I once thought of.

\( S_2 : \) Det \( \times \) CN \( \rightarrow \) T
  root \((a,\beta), T\),
  head nouns = head nouns (\( \beta \)).

\( S_3, n : \) CN \( \times \) S \( \rightarrow \) CN
  define: list = all occurrences of \( h_n \), \( \beta \);
  delete index (n, list);
  add features (gender of (head nouns (\( a \)), list));
  root \((a, root (\text{such}-that, Rel), \beta), CN\),
  head nouns = head nouns (\( a \)).

\( S_4 : \) T \( \times \) IV \( \rightarrow \) S
  add features ((pres, sing3), main verbs (\( \beta \)));
  add features ((num, head nouns (\( a \)));
  root \((a, \beta), S\),
  main verbs = main verbs (\( \beta \)).

\( S_5 : \) TV \( \times \) T \( \rightarrow \) IV
  add features (acc, head nouns (\( \beta \)));
  root \((a, \beta), IV\),
  main verbs = main verbs (\( a \)).

\( S_6 : \) Prep \( \times \) T \( \rightarrow \) IV
  add features (acc, head nouns (\( \beta \)));
  root \((a, \beta), IV\).
\[ S_7 : IV/S \times S \rightarrow IV \]
\[ \text{root} \ (\alpha, \beta, \ IV, \ \\
\text{main verbs} = \text{main verbs} \ (\alpha)). \]

\[ S_8 : IV/IV \times IV \rightarrow IV \]
\[ \text{root} \ (\alpha, \beta), IV, \ \\
\text{main verbs} = \text{main verbs} \ (\alpha)). \]

\[ S_9 : S/S \times S \rightarrow S \]
\[ \text{root} \ (\alpha, \beta), S, \ \\
\text{main verbs} = \text{main verbs} \ (\beta)). \]

\[ S_{10} : IAV \times IV \rightarrow IV \]
\[ \text{root} \ ((\beta, \alpha), IV, \ \\
\text{main verbs} = \text{main verbs} \ (\beta)) \]

\[ S_{11a} : S \times S \rightarrow S \]
\[ \text{root} \ ((\alpha, \text{root} \ (\text{and}, \text{Con}), \beta), S, \ \\
\text{main verbs} = \text{union} \ (\text{main verbs} \ (\alpha), \text{main verbs} \ (\beta))). \]

\[ S_{11b} : S \times S \rightarrow S \]
\[ \text{root} \ ((\alpha, \text{root} \ (\text{or}, \text{Con}), \beta), S, \ \\
\text{main verbs} = \text{union} \ (\text{main verbs} \ (\alpha), \text{main verbs} \ (\beta))). \]

\[ S_{12a} : IV \times IV \rightarrow IV \]
\[ \text{root} \ ((\alpha, \text{root} \ (\text{and}, \text{Con}), \beta), IV, \ \\
\text{main verbs} = \text{union} \ (\text{main verbs} \ (\alpha), \text{main verbs} \ (\beta))). \]

\[ S_{12b} : IV \times IV \rightarrow IV \]
\[ \text{root} \ ((\alpha, \text{root} \ (\text{or}, \text{Con}), \beta), IV, \ \\
\text{main verbs} = \text{union} \ (\text{main verbs} \ (\alpha), \text{main verbs} \ (\beta))). \]

\[ S_{13} : T \times T \rightarrow T \]
\[ \text{root} \ ((\alpha, \text{root} \ (\text{or}, \text{Con}), \beta), T, \ \\
\text{head noun} = \text{union} \ (\text{head nouns} \ (\alpha), \text{head nouns} \ (\beta))). \]

\[ S_{14,n} : T \times S \rightarrow S \]
\[ \text{termsubstitution} \ (\alpha, n, \beta). \]

\[ S_{15,n} : T \times CN \rightarrow CN \]
\[ \text{termsubstitution} \ (\alpha, n, \beta). \]

\[ S_{16,n} : T \times IV \rightarrow IV \]
\[ \text{termsubstitution} \ (\alpha, n, \beta). \]