CHAPTER VII

PARTIAL RULES

ABSTRACT

In the framework the syntactic and semantic rules are considered as algebraic operators. As a consequence of the definitions given in the first chapters, the syntactic rules have to be total. This is investigated and compared with linguistic requirements. Partial syntactic rules from the literature are considered and alternatives for them are presented. One of the methods to avoid partial rules is the use of rule schemes. It turns out that the requirement of using total rules is a valuable heuristic tool. Consequences of this requirement are compared to consequences of Partee's well-formedness constraint.
1. RESTRICTIONS OF THE FRAMEWORK

Based upon the principle of compositionality, we have developed an algebraic framework for the description of syntax and semantics. The algebras of the framework have operators: i.e. functions from carriers to carriers. This implicates that an operator can be applied without any further restriction to any element of the sorts required by the operator. In this chapter I will consider consequences of this aspect of the framework, and especially its consequences for the syntactic algebra. Some of these consequences are closely related with those of the 'well-formedness constraint', (PARTEE 1979b), which will be considered in section 5.

In linguistics one often conceives of a grammar as a generating device for producing all and only the expressions of a language. With this conception it is rather natural to think of restrictions on this production process. One might think of restrictions on the order of application of the rules. Two examples are the following. One might have rules of which the applicability depends on the way in which an expression is produced (such conditions are called 'global constraints'). One might have a filter which throws away some of the produced elements (e.g. one which filters out all expressions which contain a certain symbol). The description of the possible sequences of application of the rules constitutes an important component of a transformational grammar (for instance certain rules might be obligatory, others ordered cyclically), and filters are also often used in that field. If one wishes to use the syntactic knowledge from the field of transformational grammar in the field of Montague grammar, then one is tempted to incorporate these restrictions on the generation process in Montague grammars. Would that be possible, and at what price?

In our framework the syntax has to be a many-sorted algebra, i.e. a set of carriers with operations defined on these carriers. An algebra is not a generating device, it rather is the description of a situation. By describing what the syntactic algebra is, it is said what the relevant expressions are, and what the functions are which describe the relations between these expressions. The expressions can be determined in any way one likes, and nothing has to be said about their production. One might for instance define an algebra by mentioning all the elements and describing the effects of all operators (we did so in the beginning of chapter 2). A simpler method is to give a collection of generators, and tell what the operators are. Several choices of generators may be possible, one more
clever than the other. But no matter how the algebra is defined, the elements remain the same elements, the operators remain the same operators, and the algebra remains the same algebra.

The operators of the algebra are mappings from carriers to carriers. The range of an operator (the expressions obtained as results of an application of the rule) consist by definition of elements of a certain carrier. Therefore it is in our framework not possible to have a filter which says that certain outcomes are not acceptable. The domain of an operator (the expressions it operates upon) is some n-tuple of carriers. Now we obtained the information that an expression belongs to a carrier is of no influence. The applicability of an operator cannot depend on the information which rules were applied previously, because there are no 'previously applied rules' for an element of an algebra. For this reason, there cannot be a prescribed ordering on the rules, there cannot be rules that are explicitly required to be used in all derivations, and the derivational history cannot influence the applicability of the rules.

Of course, the generation of expressions is an important aspect of syntax, and therefore we paid special attention to it. The notion of a generated algebra was defined, and theorems were proved about such algebras. In a generated algebra it might be meaningful to speak about filtering, ordering of the application of rules, the influence of derivational history, and obligatory rules. But if we would allow this, we would describe a generation mechanism, and not operators of an algebra: in an algebra there is no place for such aspects. So this discussion brings us to reject certain methods which are customary in the tradition of transformational grammars. But the rejection only concerns the method, not the ideas. It is possible to organize an algebra in such a way that the same effects are obtained in another way. Below I will give some examples.

An explicit ordering of rules is not possible in an algebra. But in a generated algebra there is a certain natural ordering among the operators. If an operator $R$ takes as its argument an expression of category $C$, then the operators which yield expressions of the category $C$ are used before $R$. In this way the categorial system of the algebra has as effect a certain implicit ordering of the operators.

If one wants a certain ordering on the rules, this effect can be obtained by a suitable refinement of the categorial system. Let $R_a$ and $R_b$ be two rules, both operating on sentences and yielding sentences. Suppose that
we want \( R_a \) to be applied precisely one time before \( R_b \). This effect can be obtained by distinguishing among the sentences two categories: \( S_1 \) and \( S_2 \). Here \( S_1 \) is the category of sentences to which \( R_a \) has not yet been applied, and \( S_2 \) of sentences to which \( R_a \) has applied. Then \( R_a \) can be defined as a rule which operates on expressions of category \( S_1 \) and yields expressions of category \( S_2 \), whereas \( R_b \) is defined to operate on expressions of category \( S_2 \), yielding expressions of this category again. The definitions of the other rules have to be adapted for these categories as well. I expect that by means of a refined categorization system the effect of any ordering can be obtained. Since in the field of Montague grammars explicit rule ordering hardly is employed, I will not consider this topic any further.

As explained above, the applicability of a syntactic rule to an expression cannot depend on the derivational history of that expression. Notice that, on another level, we already met a situation where it was important to have derivational histories available. The meaning of an expression may depend on the derivational history of that expression. We did not define the translation homomorphism on the algebraic grammar for a language because in that grammar such histories are not available. The translation homomorphism is defined on the associated term-algebra, i.e., the algebra of derivational histories. This suggests us what to do when derivational histories would be important in syntax: use an algebra in which the elements represent derivational histories. But in the field of Montague grammars I know of only one rule which uses information about the derivational history (rule 3 of THOMASON 1976), so the issue does not seem to be important. Moreover, this aspect of Thomason's rule can probably be avoided by following the proposal of PARTEE (1973) to let a grammar produce not unstructured strings, but labelled bracketings. For these reasons the role of derivational histories in the syntax will not be considered here any further.

Above we have considered some restrictions on the circumstances in which a rule may be used. The conclusion was that such rules violate basic aspects of our framework. Another request from linguistics is to allow restrictions on the expressions to which a rule is applied. In the field of transformational grammars it is standard to put conditions on the possible inputs of a transformation. In the field of Montague grammar many rules are proposed as well which do not apply to all expressions of the category required be the rule, but only to some of them. In the field of semantics one has proposed to use functions which are not defined for all arguments of the required type (see section 2.2). In contrast to the constraints on
applicability discussed above, one might argue that our framework should allow for operators which are not defined for all arguments of the required sort. Such partial operators are known in the theory of universal algebras; the algebras in which they occur are called partial algebras. In the next sections it will be investigated whether we could be more liberal than we have been, and whether we should allow for partial algebras within our framework.

2. PARTIAL ALGEBRAS

2.1. Partial grammars

Contrary to what one might expect, it is not just a minor variation of the system to allow for partial algebras (i.e. algebras with partial operations). Such a step would disturb important parts of theory we have developed so far. I will illustrate this by means of two examples which show that certain theorems we proved concerning properties of the syntax are not valid when partial rules are allowed. In 2.2 it will be shown that certain theorems of intensional logic loose their validity when partial operators are allowed in the logic.

2.1. EXAMPLE.

\[ G = <\{ [A], \{ b \}, \{ c \}, \{ F_a, F_b, F_c \} \}, D> \]

Here \( F_a : A \rightarrow A \) is defined by \( F_a(\alpha) = \alpha \alpha \)

\( F_b : B \rightarrow B \) is defined by \( F_b(\beta) = \beta \beta \)

\( F_c : C \rightarrow C \) is defined by \( F_c(\gamma) = \gamma \gamma \).

So by repeated application of \( F_a \) strings of arbitrary length consisting of \( a \)'s are produced. Analogously for \( F_b \) and \( F_c \). Furthermore the partial rule \( F \) is defined as follows:

\[ F : A \times B \times C \rightarrow D \]

where

\[ F(\alpha, \beta, \gamma) = \begin{cases} \alpha \beta \gamma & \text{if the lengths of } \alpha, \beta \text{ and } \gamma \text{ are equal.} \\ \text{undefined} & \text{otherwise} \end{cases} \]
The language \( L(G) \) generated by \( G \) is \( \{ a^n b^n c^n \mid n \in \mathbb{N} \} \). This is a non-context-free language (see Hopcroft & Ullmann 1979 example 6.1). So when partial operations are allowed in the syntax, theorem 5.6 from chapter 2 does not hold.

2.2. **Example.** Let \( L \) be some recursively enumerable language over alphabet \( A \). According to theorem 3.7 from chapter 2, there is an algebraic grammar \( G \) such that \( L(G) = L \). Suppose that \( G = \langle \langle B \rangle_{s \in S}, \langle F \rangle_{Y \in T}, s_0 \rangle \).

Let \( \sigma \in A^* \) be arbitrary, and define the algebraic grammar \( H_{\sigma} \) by

\[
H_{\sigma} = \langle \langle B \rangle_{s \in S}, \langle F \rangle_{Y \in T}, v[f], s_1 \rangle
\]

where \( s_1 \) is a new sort \( (s_1, v(s_0)) \), and where \( f \) is a partial operation defined by

\[
f: s_0 \rightarrow s_1 \text{ where } f(\alpha) = \begin{cases} 
\sigma & \text{if } \alpha = \sigma \\
\text{undefined} & \text{otherwise.}
\end{cases}
\]

Note that \( H_{\sigma} \) produces a language which is either empty (if \( \sigma \notin L(G) \)) or consists of \( \sigma \) (if \( \sigma \in L(G) \)). So \( L(H_{\sigma}) \neq \emptyset \) iff \( \sigma \in L(G) \).

Suppose now that it was decidable whether \( L(H_{\sigma}) = \emptyset \); then it was decidable as well whether \( \sigma \in L(G) \). Since \( L(G) \) is an arbitrary recursively enumerable language, the latter is not decidable, and consequently it is not decidable whether \( L(H_{\sigma}) = \emptyset \). This means that theorem 5.5 from chapter 2 (which states the decidability of the emptiness of \( L(G) \)) is not valid if we allow for partial operations.

2.2. Partial models

The following example concerns the use of partially defined operations in the semantics. They arise, for instance, if one wishes to deal with sortal incorrectness: certain combinations of a verb phrase with a subject do not fit well together, although most expressions of their categories give no rise to problems. An example (Thomason 1972) is (1).

(1) The velocity of light is shiny.

It is not attractive to say of such a sentence that it is 'false', since then its negation would be 'true'. Either, one should consider (1) as being syntactically incorrect, or the strangeness should be dealt with in the semantics. Thomason (1972) followed the latter approach and has proposed
to assign to such sentences no truth values. This idea is worked out in the framework of Montague grammar by WALDO (1979). In his proposal several semantic domains contain partial functions, and the function corresponding with *shiny* is not defined for arguments such as 'the velocity of light'. So (1) is not associated with a truth value.

Waldo's approach gives rise to strange consequences. Formulas which one might expect to be equivalent, are not. I will discuss two examples, and indicate how the problems could be solved by using total functions in the model.

The first example concerns formula (2), where \( \phi \in \mathfrak{M}_\mathcal{E} \).

(2) \[ \phi = \phi. \]

Suppose that the interpretation of \( \phi \) is undefined (e.g. because it is the translation of (1)). Then, due to the interpretation of =, also (2) is undefined. However, due to the interpretation of connectives (which uses 'extended interpretations'), formula (3) gets the interpretation true:

This difference in interpretation is, in my opinion, a strange result.

(3) \[ \phi = \phi \land \phi = \phi. \]

The second example is based upon a suggestion of R. Scha (pers.comm.). It concerns formula (4), where \( z \in \var{\mathcal{E}} \), and where \( \phi \in \mathfrak{M}_\mathcal{E} \) is as in (2).

(4) \[ \lambda z[z=z](\phi). \]

The possible assignments to \( z \) are, in Waldo's partial model, the truth values true and false. Therefore the expression \( z = z \) is equivalent with some tautology not containing \( z \), for instance \( \forall w[w=w] \). Hence (4) is equivalent to (5)

(5) \[ \lambda z[\forall w[w=w]](\phi). \]

According to the standard conditions for \( \lambda \)-conversion, formula (5) can be reduced to (6), which clearly gets the interpretation true.

(6) \[ \forall w[w=w]. \]
Also in (4) \( \lambda \)-conversion is, according to the standard conditions, an allowed operation. Then (2) is obtained, but the interpretation of that formula is undefined. So the formulas (6) and (2), obtained by reduction of (4) are not equivalent, an unacceptable result. We have to conclude that one of the reductions steps is not allowed. This problem is, in my opinion due to the fact that for the variable \( z \) in (4), there are two possibilities (true and false), whereas for the arguments \( \phi \) there are three possibilities (true, false, and undefined). Note that the variable \( z \) cannot be undefined, because its range consists of all elements in the model of the correct type, and \textit{undefined is no value in the model.}

The above examples show that the laws of logic we have met before, cannot be used in this system without further investigations. In any case the conditions for \( \lambda \)-conversion have to be changed. Unfortunately, Waldo does not provide laws for his system. This causes a difficulty in the study of his proposal. He presents several examples, and each consists of a sentence accompanied by its reduced translation. Since I do not know which reduction rules hold in an approach with partial functions in the semantics, I cannot check the correctness of the examples. Also other authors who describe a fragment with the use of partial functions in the semantic domains, do not present reduction rules (HAUSSER (1976), COOPER (1975)). The last author mentions at least that not all standard reductions remain valid. I expect that it will be very difficult to reformulate the reduction rules. An obvious attempt to improve the conditions for \( \lambda \)-conversion would be to require that the reduction of \( \lambda z[a](\beta) \) is allowed only if \( \beta \) is defined. This is, however, not a syntactic condition on \( \beta \), and I doubt whether it is possible to give a syntactic characterization of 'undefined'.

I already explained that the problem is due to the fact that a variable cannot take the value \textit{undefined}, whereas an argument \( \phi \) (which might be substituted for that variable) can be undefined. Therefore I expect that the problems will be solved when a third truth value is introduced, say a value \textit{error}. In any case, the two problems mentioned above disappear. If the value \textit{error} is assigned to \( z \), then the interpretation of \( z = z \) is always the same as the interpretation of \( \phi = \phi \), even in case \( \phi \) is undefined. Now \( \lambda \)-conversion is allowed both in (4) and in (5), and furthermore, all formulas (i.e. (2)-(6)) get the same interpretation for all values of \( \phi \). Note that this plea for using a third value is not an argument for using some of the existing tables for three valued logic. Waldo uses super-valuations (Van FRAASSEN 1969), and one might try to reformulate super-
valuations for an approach with a third truth value.

The idea of using a third truth value is not new; it goes back to EUCKASIEWIC (1920), who gives tables for the connectives in a three-valued system. In the theory of topos one introduces a value representing 'undefined' (GOLDBLATT 1979, p.268). In the theory of semantics of programming languages the problems of working with 'undefined' are well-known. Undefinedness arises, for instance, when a process is defined for calculating the value of a function, whereas the process does not terminate normally because not enough memory capacity is available. The standard approach in this field is not to use partial functions, but to make the functions total by means of the introduction of an extra element in the semantic domain, called 'errorvalue' or 'bottom' (SCOTT & STRACHHEY 1971, GOQUIN 1978). In the field of Montague grammars the situation is as follows. A model for intensional logic with 'undefined' as possible value, is presented in Von KUTSCHERA (1975). It is however common practice to consider undefinedness not as a value (see KAMP 1975, COOPER 1975, HAUSER 1976, WALDO 1979, KLEIN 1981). I know of only one author who presents a treatment of a certain fragment and uses a model with 'undefined' as value: Ter NEULEN (1980).

2.3. Discussion

The examples given in sections 2.1 and 2.2 show that it will be a considerable change of the framework to allow for algebras with partial operations. Moreover, it is not obvious in which way we have to proceed. GRAETZER (1968,p.80) says the following. 'For algebras there is only one reasonable way to define the concepts of subalgebra, homomorphism, and congruence relation. For partial algebras we will define three different types of subalgebra, three types of homomorphism, and two types of congruence relation [...] all these concepts have their merits and drawbacks'. This situation constitutes an argument for my expectation that it will be a considerable task to develop a framework based upon the use of partial algebras. What I have seen of the literature concerning partial algebras did not give me the confidence that an elegant framework can be built using this notion (e.g. ANDREKA & NEMETI (1982), MIKENBERG (1977), ANDREKA, BURMEISTER & NEMETI (1980)). The example concerning partial functions in the semantics gives me the conviction that it is not a good idea to base a semantic theory on partial functions. For these three reasons I do not sympathize with the idea of basing the framework on partial algebras.
As for the introduction of partial rules in the syntax only, the situation seems to be different. It is just a minor change of the framework because the homomorphic relations between the algebras of the framework are hardly disturbed. An argument in favor of the introduction of partial rules is that such rules are used frequently in practice. But there also are arguments against the introduction of partial rules in the syntax. Below I will mention some of them, thereafter they will be discussed.

1. Consistency of argumentation

The first argument concerns the consistency of our argumentation. In a Montague grammar we distinguish categories, and the rules give the information in which way the expressions of certain categories combine to form expressions of other categories. An argument for distinguishing such categories (given e.g. in chapter 1) was that certain groups of expressions behave differently from other groups in syntactic or semantic respects. Designing partial rules would mean that among a single category we distinguish two subgroups (these expressions of a category to which the rule can be applied, and those to which the rule cannot be applied). A consistent reaction in such a situation would be to conclude that the system of categories was not refined enough, and that the system has to be refined in such a way that the partial rules are no longer partial.

2. Filtering power

A partial rule introduces a kind of filter in the grammar, and filters form a powerful tool which can easily be abused. In a Montague grammar the syntactic rules provide the information which kinds of expressions may be combined to form new expressions. But partial rules would make it possible that the syntactic rules combine rubbish to rubbish, whereas a final partial rule would filter out the undesired expressions. In this way, the other rules would not give information about the combinations which make sense and which not. The filtering power of partial rules in syntax is employed in the first two examples given above.

3. Generation of expressions

Often one wishes to conceive a grammar as a generating device. The rules of the fragment presented in chapter 4 can easily be conceived in this way. A rule like $S_4$ is considered as an instruction stating that if one wants to generate a sentence, one has to generate a term and an
IV-phrase, and next combine them. The rules for term-formation and IV-formation are, in the same way, considered as instructions for a generating process. The processes of generating a term and of generating an IV-phrase may be carried out independently, and every outcome of the processes is acceptable. Details of a computer program based upon these ideas can be found in JANSSEN (1980a). Suppose now that the grammar contains a partial variant of $S_4$, say a rule which imposes restrictions on the possible combinations of a term with an IV-phrase. Then the simple algorithm just sketched cannot be used. One has to design an algorithm that gives the guarantee that after a finite amount of time an acceptable combination is found (provided that there is one). This requirement would make the algorithm rather inefficient: the only possibility I see for such an algorithm is one which tries out all possible combinations of a term with an IV-phrase. So in the perspective of a generation process partial rules are unattractive.

4. Consequences

An important argument in favor of total rules is that this requirement has attractive consequences. On a more theoretical level it gives rise to an interesting restriction on the possibility to incorporate transformations in a Montague grammar (see section 3). On a more practical level the requirement of using total rules turns out to be a valuable heuristic tool. Several partial rules from the literature can be reformulated or eliminated, and the requirement suggests how this can be done. Thus several proposals from the literature can be replaced by a simpler treatment (see section 4).

5. No theory

The introduction of partial rules, even if only in the syntax, constitutes a considerable change of the framework. As the given examples have shown, the theory which we have developed, cannot be used without corrections. Since a theory about partial syntactic algebras is not available, there is no guarantee that all consequences are acceptable.

None of these five arguments is decisive. As for 'consistency', it is indeed more elegant to use the argument for the introduction of categories in all situations with the same conclusion. But with respect to other considerations there might be arguments of elegance in favor of partial rules (e.g. that in that way linguistic generalizations can be captured). That partial rules introduce a powerful filter, is not an impressive theoretical argument since the algebraic grammars have a universal generative capacity
anyhow. As for the argument of 'generation', it is not a primary aim of our grammar to develop an efficient generating device. From a practical point of view, a parser might even be of more importance than a generator. The fact that the practical consequences of using total rules turns out to be attractive in the situations considered, is not a guarantee that in other cases this will be the case as well, and that there is no theory about partial algebraic grammars might be a challenge to develop such a theory.

The arguments against the introduction of partial rules and the arguments in favor of doing so, have to be weighed against each other. The arguments given above show that there are several unattractive aspects related with the introduction of partial rules. I do not know of convincing arguments for the need or attractiveness of partial rules. In the remainder of this chapter I will show that there are several alternatives for the introduction of partial rules. These alternatives are: reformulating as a total rule (section 3), reformulating as a rule operating on another category (section 4) and a refined system of subcategories (section 5). It will turn out that the use of these alternatives gives, in most cases, rise to a simpler treatment than originally proposed: the requirement of using total rules turns out to be a valuable heuristic tool. So the situation can be summarized as follows: there are arguments against the introduction of partial rules, and attractive alternatives are available. Therefore I do not feel enthusiastic about the introduction of partial rules in the syntax. I do not state that I will never use partial rules myself, but I would first try to use total rules.

3. INCORPORATING TRANSFORMATIONS

In the field of transformational grammars, the use of partial rules is standard. As part of their specification the transformations always contain a restriction on the inputs to which they may be applied (a SC: i.e. structural condition). One might wish to incorporate transformations in Montague grammar in order to benefit from the syntactic insights obtained in that field. In this section I will present a general method for the incorporation of a class of transformations in a Montague grammar in which all rules have to be total.
Some characteristics of transformations are as follows

1. Transformations define mappings from trees to trees; these trees represent constituent analyses of sentences.
2. If several transformations can be applied, then their order of application may be prescribed.
3. A transformation is applied to one input tree at a time.
4. A transformation imposes structural conditions determining the possible input trees.

In order to take care of the first point, it is required that a Montague grammar does not produce plain strings, but trees, (or, equivalently, labelled bracketings). Let us assume that Montague's framework is adapted in the way proposed by Paré (1973). So the grammar produces trees. This change of the system turns all rules into rules which operate on trees, so in a certain sense all the rules in the grammar become transformations. In order to avoid confusion of terminology, I will use the name C-transformation ('Chomskyan') for transformations used in transformational grammars. Once they are incorporated in a Montague grammar, they are called M-transformations.

The second characteristic point is not acceptable in our framework. As explained in section I, explicit rule ordering does not fit into the algebraic approach. But an implicit rule ordering which has the same effects might be possible. The third point does not give rise to problems. Although the rules in a Montague grammar mostly take two arguments, there is no objection against rules taking one argument. The fourth point is problematic since it implies that C-transformations are partial rules. This is an important characteristic of C-transformations which makes them very attractive for practical use. It makes it possible to indicate in a simple way what the relevant input trees are, without the need to bother about irrelevant inputs.

I will incorporate a class of C-transformations in a Montague grammar which requires total rules, by reformulating them in a way which makes them total. The reader might be surprised by this reformulation and at first glance consider it as a sneaky trick employed in order to obey the letter of the principle. This is not completely true. The reformulation expresses a different view on transformations than the standard one, and it has interesting consequences.

The reformulation proceeds as follows. Suppose that a C-transformation
is given in the following form.

If the input sentence satisfies structural condition SC, then we may apply transformation T in order to obtain a new sentence, otherwise T cannot be applied.

Its reformulation as a total rule has the following form.

To the input sentence we apply operation T'. Operation T' is defined as follows. If the input sentence satisfies the structural condition SC, then transformation T is applied, and otherwise the 'do nothing' transformation is applied.

By the 'do-nothing' transformation is understood an operation which does not produce another expression, but which gives its input unchanged as output. The reformulation expresses the view that an M-transformation applies to all expressions of the required category, and that its application yields always a result.

Corresponding with a M-transformation T' there has to be a translation rule τ. For the cases that we did 'nothing' in the syntax, we do 'nothing' in the semantics: the input formula is given unchanged as output. This means that for these cases the translation rule τ can be represented as the polynomial xₜ,₁. Since in our framework τ has to be represented by means of a single polynomial expression, τ yields for each input formula, that formula as output. So the M-transformations (obtained with the method described here) do not change meanings. Consequently, if one wants to incorporate C-transformations in this way in a Montague grammar, then these transformations have to be meaning preserving! This requirement is a well-known hypothesis in the so called standard theory of transformational grammars (see PARTER 1971 for a discussion); it is, however, nowadays not generally accepted.

The conclusion that transformations have to be meaning preserving, holds only for the method described above. But I do not know of any other uniform method for incorporating transformations in a Montague grammar with total rules. To illustrate this, I consider one attempt. Instead of requiring that the translation rule corresponding with a do-nothing transformation is the identity operation on formulas, we might require that it is the identity operation on meanings (but not necessarily the identity on formulas). This would make it possible that the polynomial is not the identity when interpreted for the real transformation. Such a rule τ has the following effect:
\[ \tau(\phi) = \begin{cases} 
\rho(\phi) & \text{if } \phi \text{ is the translation of an expression which satisfies the conditions for application of the transformation} \\
\phi & \text{(}\rho \text{ formalizes the semantic effect of the transformation)} \\
\phi & \text{otherwise.} 
\end{cases} \]

The first objection is that this effect cannot be obtained by means of polynomial over IL. In order to obtain the effect of such a choice, IL has to be extended with something like the \textit{if-then-else} construction. There would be, however, no problem in doing so. A more essential objection is that in the description of the translation rule \( \tau \) information about the (syntactic) expressions is used. This has to be replaced by information concerning their meanings. For most transformations there is probably no semantic property corresponding to the condition on the transformation. In any case, we have no uniform method for obtaining a logical condition which is equivalent with the structural condition of the transformation. So a uniform method for finding the polynomial cannot be given.

I described a uniform method for the incorporation of a class of transformations in Montague grammar by means of a do-nothing transformation. This method might be generalized to a method to eliminate certain partial rules from a Montague grammar. For rules with one argument the method can be used if the rule is meaning preserving. For rules with more than one argument the use of a kind of do-nothing transformations implies that (at least) one of the inputs should have the same category as the output. The do-nothing transformation has to correspond to a translation rule which is the identity on formulas. Therefore the translation rule which corresponds with the original partial rule has to be the identity translation for one of its arguments. So this method can be used only for very limited class of the partial rules with more than one argument.

4. DEFINED FOR ANOTHER CATEGORY

4.1. Introduction

In this section I will consider several rules from the literature which are partial, and for which the corresponding translation rule is not meaning preserving. This implicates that the method developed in the previous section cannot be used for them. The method employed in this section is to
reformulate the rule for another category than where it was originally for-
mulated for. It turns out that in all cases the new version of the rule is
simpler than the original formulation of the rule, and sometimes the origin-
al rule was incorrect whereas the new rule is correct. This shows the
heuristic value of the framework, and of the requirement of using total
rules in particular. The examples are presented in the notation of the origi-
nal proposal; most examples were already mentioned in JANSSEN (1978a).

4.2. *He*$_1$ is loved

PARTEE (1973) considers the M-transformation 'Passive Agent Deletion'.
An example is

$$F_{102}(he$_1$ is loved by him$_3$) = he$_1$ is loved.$$

Translation:

If $\phi \in P$, and $\phi$ translates into $\psi'$, then $F_{102}(\phi)$ translates
into $\exists x \phi'$.

On the one hand this transformation applies only to input trees of a
special form, on the other hand the translation rule is not the identity
mapping. This means that we cannot reformulate this transformation as a
total rule, and that Partee's way of dealing with agentless passive is dis-
allowed by the requirement of using total rules. For the example under dis-
cussion, the literature provides an alternative. THOMASON (1976) presents
rules for generating passive directly, i.e. without a passive transforma-
tion and without a passive agent deletion.

4.3. Give John a book

The C-transformation of dative shift changes sentence (7) into (8).

(7) Mary serves the cake to John
(8) Mary serves John the cake.

A refined category system for sentences in which dative-shift would be a
total rule is very difficult to design (since each new subcategory would
require rules producing expressions of that subcategory). Also here the
literature contains an alternative. Dowty (1979a) shows that the partial rule of dative shift on the level of sentences, can be replaced by a rule on the level of lexical elements. That rule changes the category of the verb serve fromDTV (verbs which take a Dative and a Term) to TTV (verbs which take two Terms). By having a sufficiently refined category system, these lexical rules become total rules. Many examples of transformations which are reformulated on the lexical level can be found in Dowty 1978, 1979a, and in Bartsch 1978b, thus they can easily be reformulated as total rules.

4.4. Mary shakes John awake again

In chapter 5, section 5.2, we considered some semantic aspects of the proposals of Dowty (1976) concerning the treatment of factives. Now I will consider some syntactic aspects (of course, it doing this, the semantic aspects cannot be neglected). Dowty produces the factive sentence Mary shakes John awake from the term Mary and the TV-phrase shake John awake. This TV-phrase in its turn is obtained from the TV-phrase shake awake. The first rule Dowty presents for generating this TV-phrase is as follows.

\[ S_{30}: \text{If } \alpha \in P_{TV} \text{ and } \phi \in P_t \text{ and } \phi \text{ has the form } he_n \text{ is } \gamma \text{ then } F_{30,n}(\alpha, \phi) \in P_{TV} \text{ where } F_{30,n}(\alpha, \phi) = \alpha \gamma. \]

An example is:

\[ F_{30,1}(\text{shake, he awake}) = \text{shake awake}. \]

The corresponding translation rule reads:

\[ T_{30}: \text{If } \alpha \text{ translates into } \alpha' \text{ and } \phi \text{ translates into } \phi' \text{ then } F_{30,n}(\alpha, \phi) \text{ translates into:} \]

\[ \lambda x^\gamma P(\lambda x[\alpha'(x, \lambda x^\phi^\gamma P(x)) \text{ cause[become[\phi']]}. \]

This rule is a partial rule which is not meaning preserving, so we have to find another approach. Can the above result be obtained by means of a total rule? For generating expressions like shake awake one only needs an adjective and a TV-phrase. So it lies at hand to try the following rule

\[ S_{601}: \text{If } \alpha \in P_{TV} \text{ and } \beta \in P_{adj} \text{ then } F_{601}(\alpha, \beta) \in P_{TV} \text{ where } F_{601}(\alpha, \beta) = \alpha \beta. \]

The corresponding translation rule would be

\[ T_{601}: \text{If } \alpha \text{ translates into } \alpha' \text{ and } \beta \text{ translates into } \beta' \text{ then } F_{601}(\alpha, \beta) \text{ translates into:} \]

\[ \lambda x^\gamma P(\lambda y[\alpha'(x, \lambda x^\beta^\gamma P(y)) \text{ cause[become[\beta'(y)]]}. \]
Why did Dowty propose a production of *shake awake*, with as intermediate stage the sentence *he, is awake*? This has probably to do with the ambiguity of *Mary shakes John awake* again. On the one reading Mary has done it before, on the other John has been awake before. Dowty treats again as a sentence modifier and he needs two different sentences in the derivation in order to deal with the ambiguity. He starts his investigations along this line probably for historical reasons: it is the way in which such constructions are treated in generative semantics. But, as in the previous examples, we need not to follow the old pattern. By rule R\textsubscript{601} we are guided to another approach to this ambiguity. The one reading can be obtained by combining again with *Mary shakes John awake*, the other by combining it with *shake awake*. I do not go into details of this approach for the following reason. After considering several phenomena concerning factives, Dowty observes that his first approach is not completely adequate. He discusses extensively several alternatives and escapes. Finally he concludes 'there would be no reason why we should not then take the step of simplifying rules S30–S32 drastically by omitting the intermediate stage in which a sentence is produced'. Next he presents as the rule which he considers as the best one, a rule which is identical with S\textsubscript{601}. So the framework has led us immediately to the solution which is the simplest and best one. This example suggests that we might derive from the framework the advice 'when designing a syntactic rule, ask for what you need as input and not for more'.

4.5. See himself

In chapter 5, section 2.1, we considered the derived verb phrase rule of PARTEE (1973). This rule makes verb phrases out of sentences. An example is

$$F_{104}(\textit{he}_{1} \textit{see him}_{2} \textit{self}) = \textit{him}^* \textit{self}.$$  

The syntactic part of this rule reads as follows:

If $\phi$ $\epsilon$ $P_{t}$ and $\phi$ has the form $t_{\textit{IV}} (he_{t}) [a]$, then $F_{104}(\phi)$ $\epsilon$ $P_{IV}$, where $F_{104}(\phi) = \alpha'$, and $\alpha'$ comes from $\alpha$ by replacing each occurrence of $he_{t}$, $him_{t}$, $him_{t} \textit{self}$ by $he^*_{t}$, $him^*_{t}$ $him^*_{t} \textit{self}$ respectively.

At the one hand the derived verb phrase rule is a partial rule, at the other hand its output belongs to a different category than its input. Therefore we cannot reformulate this rule as a total one using a do-nothing transformation. The derived verb phrase rule is disallowed by the requirement
of using total rules, and we have to find another treatment for the cases where Partee uses this rule. Let us, in accordance with the advice given in section 4.4, just ask for what we actually need and not for more. In the above example we only need a TV-phrase. So we might try the following rule.

\[ S_{602} \text{ If } a \in P_{TV} \text{ then } F_{602}(a) \in P_{TV} \text{ where } F_{602}(a) = a \text{ him } ^* \text{self}. \]

The corresponding translation rule reads:

\[ T_{602} \text{ If } a \text{ translates into } a', \text{ then } F_{602}(a) \text{ translates into } \lambda x[a'(x, \lambda P(x))]. \]

Would this rule be an acceptable alternative?

Let us consider why one would like to generate \( \text{see himself} \) from the source sentence \( \text{he sees himself} \). There are semantic arguments for doing so. The sentence \( \text{John sees himself} \) is obviously semantically related to the sentences \( \text{John sees John} \) and \( \text{he sees him} \). In transformational grammar this might be an argument for producing these sentences from the same source: no other formal tool is available. The effect of Partee's rules is that such a transformation is split up into several stages; it amounts to the same relations. Montague grammar has a semantic component in which semantic relations can be laid formally. So if we do not have to ask for a sentence as source for syntactic reasons, we are not forced to do so on semantical grounds. So this cannot be an argument against \( S_{602} \).

Partee (1975) provides as an explicit argument for the derived verb phrase rule the treatment of sentence (9)

(9) \( \text{John tries to see himself.} \)

This sentence is generated, using the derived verb phrase rule, from sentence (10)

(10) \( \text{he }_3 \text{ tries to see him}_3 \text{self.} \)

The translation of (9) becomes in this case (11)

(11) \( \text{try to}[^* \text{John}, \lambda x_3[\text{see}(x_3, \lambda P(x_3))]]. \)

Sentence (9) can also be generated according to the rules of PTQ. If we do not change the syntactic details of the rule the following sentence is produced:

(12) \( \text{John tries to see him.} \)

In (12) \( \text{him} \) is coreferential with \( \text{John}. \) The translation is
(13) \( \text{try to}(\lambda \text{John}, \text{see}(\lambda \text{p}[\text{p}(\text{John}))]) \).

Partee provides arguments for her opinion that interpretation (11) might be preferable to (13). Let us assume that her arguments hold and consider whether \( S_{602} \) is compatible with that. The combination of \( \text{try to} \) with the translation of \( \text{see himself} \) (obtained by \( T_{502} \)) yields

(14) \( \text{try to}(\lambda x[\text{see}(x, \lambda \text{p}[\text{p}(x)])]) \).

So the translation of \( \text{John tries to see himself} \) is, as desired, equivalent to (11). As Partee notices, the derived verb phrase rule does not prohibit the unwanted reading (13). Rule \( S_{602} \) is an improvement since it only allows for reading (11). Of course, \( S_{602} \) does not give a complete treatment of reflexives, and I am not sure whether I would like to treat them in this way. For the purpose of the discussion this aspect is irrelevant: I just would like to demonstrate that the requirement of using total rules, and in particular the advice 'ask for what you need', guides us to a better rule than originally proposed.

4.6. Easy to see

Partee (1975) presents another example for the derived verb phrase rule:

\[ F_{104}(\text{he is easy to please}) = \text{be easy to please}. \]

This example may seem somewhat strange since it produces the IV-phrase \( \text{be easy to please} \) from a sentence containing this IV-phrase. The reason is that the sentence is obtained by some transformation from the source

(15) \( \text{To please him, is easy}. \)

This transformation is not sufficient for producing all sentences containing the phrase \( \text{be easy to please} \). Phrases resulting from \( F_{104} \) have to be produced as such, in order to generate (16) and (17).

(16) \text{few rules are both explicit and easy to read}

(17) \text{try to be easy to please}.

In Partee (1977a) other constructions are considered which contain expressions of this kind, such as

(18) \text{John is being hard to please}.

In order to deal with such expressions Partee needs another rule, called
the derived adjective rule, which has the following effect

\[ IV[\text{be easy to please}] \rightarrow \text{ADJ}'[\text{easy to please}]. \]

This is again a partial rule which cannot be brought in accordance with the restriction of total rules. So for (15)-(18) a alternative has to be given.

The advice given in section 3.4 stimulates us to ask just for what we need for generating easy to please. We need an expression like easy and some TV-phrase. Let us, following PARTEE 1977a, assume that we have a special category ADJ which contains easy, tough etc. The resulting expression easy to please will be of the category ADJ'. Then we are guided to the following rule:

\[ S_{603}: \text{If } \alpha \in P_{\text{ADJ}} \text{ and } \beta \in P_{\text{TV}} \text{ then } F_{603}(\alpha, \beta) \in P_{\text{ADJ}}: \text{ where} \]

\[ F_{603}(\alpha, \beta) = \alpha \text{ to } \beta. \]

The translation of (this) easy must be such that it may be combined with an TV-translation in order to obtain an expression of the type of translations of adjectives. Then the translation rule reads

\[ T_{603}: \text{If } \alpha \text{ translates as } \alpha' \text{ and } \beta \text{ as } \beta' \text{ then } F_{603}(\alpha, \beta) \text{ translates into} \]

\[ \lambda x \alpha'((\lambda y \beta')'(y, \lambda \alpha'' P(x))). \]

Rule \( S_{603} \) makes it possible to generate the expressions containing easy to please we mentioned above. Unfortunately, Partee does not provide an explicit semantics for the source of all her constructions (sentence (15)) so we cannot compare it with the semantic consequences of \( S_{603} \); but I expect that she will finally end up with something close to the result of \( T_{603} \). Concerning the syntax, it is demonstrated that our requirement guides us to a much simpler treatment.

In section 3.5 and 3.6 we have considered two examples concerning the derived verb phrase rule. These examples do not cover all possible applications of the rule. But the treatment given here shows that in any case the two kinds of examples for which Partee has used the derived verb phrases rule can be dealt with in a better way by means of total rules.

5. SUBCATEGORIZATION AND RULE SCHEMES

5.1. Hyperrules

An argument for distinguishing categories (given for instance in
chapter 1, section 1.3) is that certain groups of expressions behave (syn-
tactically or semantically) differently than other groups of expressions.
If for some rule it turns out that the rule can only be applied to a subset
of the expressions of its input category, then this can be considered as an
indication that the system of categories is too coarse. A method to avoid
partial rules consists of refining the system of categories. In this sec-
tion we will consider examples of this method, and present tools which are
useful when it is employed.

There are several arguments for distinguishing among the category of
nouns the groups of mass nouns and of count nouns. One of the differences
between the expressions of these two groups is their behaviour with re-
spect to determiners. Let us compare, as an example, the count noun ring
with the mass noun gold. Well-formed are a ring and every ring, whereas
ill-formed are a gold, and every gold. In larger expressions the same dif-
fferences arise: well-formed are a beautiful ring and every ring from China,
whereas ill-formed are a beautiful gold and every gold from China. These
differences constitute an argument for introducing in the grammar the sep-
parate categories 'Mass Noun' and 'Count Noun'.

In many respects, however, mass nouns and count nouns behave analogous-
ly. Expressions of both categories can be combined with relative clauses
and with adjectives. If we treat mass nouns and count nouns as being
two independent categories, then the consequence is that the rules for rela-
tive clause formation and for adjective addition are duplicated. Thus the
grammar will contain a lot of closely related rules. This effect will be
multiplied if more categories are distinguished among the nouns. Therefore
it is useful to have a tool for controlling this proliferation. Such a
tool are rule schemes.

Rule schemes are not new in Montague grammars; recall the rule for
relative clause formation given in chapter 4.

\[ S_{3,n} : CN \times S \rightarrow CN \]
\[ F_{3,n} : \text{replace in } a \text{ all occurrences of } ha_n, \text{ by } him/her/it, \text{ and of } him_n, \]
\[ \text{by } him/her/it, \text{ according to the gender of the first noun or term} \]
\[ \text{in } a; \text{ concatenate } (a, \text{ such that } a). \]

This cannot be considered as a rule because \( F_{3,n} \) deals with occurrences of
\( ha_n \), whereas this expression does not occur in the lexicon of the fragment:
examples of relevant expressions of the fragment for this rule are \( ha_1, ha_2 \).

So we have to consider \( S_{3,n} \) as a rule scheme out of which rules can be obtained.
This can be done by replacing all occurrences of \( n \) in the scheme by some number. Thus \( S_{3,n} \) stands for an infinite collection of actual rules.

In \( S_{3,n} \) three characteristic features are illustrated of the kind of rule schemes that I will use. The first one is that a rule scheme differs from a real rule by the occurrence of a parameter. \( S_{3,n} \) contains the parameter \( n \), which stands for a number. Schemes may contain several occurrences of one or more parameters, and I will put no restrictions on where a parameter stands for. The second characteristic feature is that out of a scheme an actual rule can be formed by means of substituting the same expression for all occurrences of a parameter. If it is not required that all occurrences are replaced by the same expression then the occurrences of the parameter will be indexed (e.g. with \( n_1, n_2, \ldots \)), and then occurrences with different indices may be replaced by different expressions. The third feature is that a parameter may stand for a part of a (formally simple) symbol. The expression \( ha_n \) is, formally spoken, a single generator in the syntactic algebra, but in the scheme given above it is treated as a compound symbol with \( ha \) and \( n \) as parts. This does not change the role of \( ha \) in the algebra; it remains a simple generator. One should distinguish the formal position in the algebra, and the presentation of an infinite collection of operators (or generators) by means of schemes.

A rule scheme involving nouns is the following.

\[
S_{604,n}: \text{Adj} \times \text{cm Noun} \rightarrow \text{cm Noun}
\]

\[
F_{604,n}: \text{Concatenate} (a, b).
\]

From this scheme two actual rules can be obtained. If \( \text{cm} \) is replaced by 'Count', then we obtain a rule which says that an adjective in combination with a Count Noun forms a new Count Noun. If \( \text{cm} \) is replaced by 'Mass', then we obtain a rule which says that an adjective in combination with a Mass Noun forms a new Mass Noun. This scheme exhibits again the feature that a compound symbol in the sense of the scheme, can be a single symbol in the algebraic sense. In the algebra 'Count Noun' is a category symbol, whereas in \( S_{604,n} \) it is a compound with 'Count' and 'Noun' as individual parts.

Notice that the above scheme contains two parameters: \( n \) and \( \text{cm} \).

The new formal aspects introduced in this section are the use of compound category symbols and the possibility to use parameters for parts of these symbols. The practical impact of this is that partial rules can be avoided by increasing the number of categories, and that rule schemes can be used for handling these categories.
Now I will introduce some terminology. The parameters in the rule schemes are called *metavariables*. To distinguish the rule schemes of the kind just described, from others, the former are called *hyperrules* (i.e. they are rules containing metavariables). Hyperrules without the occurrence of a variable are considered as a special case; by means of an 'empty' substitution they become actual rules. I will give the hyperrules a 'name' which starts with an H, and its parameters will not be included in the name. So rule $S_{604,n}$ mentioned above will be called $H_{604}$. The distinction between Count Nouns and Mass Nouns is in linguistics called subcategorization. I will use this term with the following formal interpretation. A category $C_1$ is called a subcategory of a category $C_2$ if the carrier of sort $C_1$ is a subset of the carrier of sort $C_2$.

5.2. Metarules

Suppose that we have a hyperrule which contains some metavariable. In the example from section 4.1 concerning nouns, I explicitly listed the two possible substitutions. But often the situation will be more complex. There are arguments for distinguishing among the nouns many more subcategories, and we will meet examples where infinitely many substitutions are possible. Therefore it is useful to have a handy tool for telling what the possible substitutions for a metavariable are. In the sequel we will use rewriting rules for this purpose. Besides the grammar consisting of hyperrules I will give a second grammar, called *metagrammar*. This grammar consists of a collection context-sensitive rewriting rules, and in these rules the metavariables of the grammar occur as auxiliary symbols. If we take some metavariable as start symbol, then the metagrammar determines a language: the set consisting of all strings which can be produced from the metavariable which was taken as start symbol. The possible substitutions for a metavariable in some hyperrule are all strings from the language generated by the metagrammar using that metavariable as starting symbol.

The benefit of using a metagrammar becomes especially clear in cases were there are several levels of subcategorization and crosslinks in the category system. As example I present the metagrammar for the subcategorization system given in CHOMSKY (1965, p.85); it is striking to observe that Chomsky used rewriting rules as well for the presentation of the subcategorization.
\[
\begin{align*}
  \text{common} & \rightarrow \text{sgn} \text{ count} \\
  \text{sgn} & \rightarrow \{ + \} \\
  \text{count} & \rightarrow - \text{Anim} \text{te} \text{ CN} \\
  \text{anim} & \rightarrow \text{sgn} \text{ Human} \text{ CN}
\end{align*}
\]

According to the convention for substitution, this metagrammar implicates that a hyper-rule containing \textit{anim} as metavariable represents two actual rules (for the subcategories \text{+ Human CN} and \text{-Human CN}), and that a hyper-rule containing \textit{common} represents 5 actual rules.

A grammar designed in the way sketched above is a system with two levels in the grammar: the level of metarules and the level of the (hyper)rules. The conception of a grammar with two levels is due to Van Wijngaarden, and was developed for the formal description of the syntax of the programming language ALGOL 68 (see VAN WIJNGAARDEN 1975). He used these notions hyper-rule and metarule with about the same meaning (for a linguistically oriented example see VAN WIJNGAARDEN 1970). The same terminology, although with a somewhat different meaning, is used in GAZDAR & SAG 1981 and GAZDAR 1982.

The concept of a two-levelled grammar gives rise to an elegant method handling a lot of rules, even an infinite number. The method could easily be generalized to multi-level grammars. In Van Wijngaarden's original system the metarules have to be context-free, whereas I allowed for context sensitive rules. This liberty has no consequences since the generative power of system lies in the rules, and not in the metarules. In the example given above (Chomsky's subcategorization) the context sensitive rules turned out to be useful. If we would be more liberal, and allow to use a type-0 grammar as metagrammar instead of a context sensitive grammar, then this would have the consequence that the by the metagrammar produced language would be undecidable. Then it would not be decidable whether a substitution for a metavariable is allowed, and consequently the set of actual rules would not be recursive. Therefore type-0 grammars are in our framework not acceptable in the metagrammar.

5.3. Variables

The use of variables in a Montague grammar gives rise to certain problems. I will consider here two of them. A more extensive discussion will be
given in chapter 8.

1. 'Left over'

According to the PTQ rules we may generate the sentence $Be_3 runa$. This is not a correct English sentence because it contains $he_3$, which is not a correct English word.

2. 'Not there'

One might apply a rule which involves variables in a situation in which such variables are not there. In this way one obtains relative clauses, which do not contain a reflexive pronoun. An example is the man such that Mary seeks a unicorn.

In order to eliminate these two problems, in chapter 8 a restriction will be proposed that contains the following two conditions:

(I) The production of a sentence is only considered as completed if each syntactic variable has been removed by some syntactic rule.

(II) If a syntactic rule is used which contains instructions which have the effect of removing all occurrences of a certain variable from one of its arguments, then there indeed have to be such occurrences.

It is evident that requirement (II) can be guaranteed by means of a partial rule. To this aspect I will return later. Requirement (I) says that all stages of the derivation process have to meet a certain condition. So is appears to be a global filter. Since one can tell from the final result whether the condition is met, it reduces however to a final filter. As I explained in chapter V, filters are not acceptable in our framework. But the effect of (I) can be obtained by means of a partial rule as follows. Replace everywhere in the grammar the category of Sentences by the category of Protosentence (so the grammar produces Protosentences). Then we add an extra rule which produces a sentence out of a protosentence in case requirement (II) is fulfilled, and which is not applicable when this requirement is not fulfilled. Thus only sentences obeying (I) are produced. Since I aim at avoiding partial rules, I have to provide an alternative method for the incorporation of the above two restrictions. This will be given below.

Categories are defined to be complex symbols consisting of two parts: a category name as we used before (e.g. S), and a representation of a set of integers. The set indicates which indices occur in the expressions of
that (complex) category. So he₂ or he₃ is an expressions of the category (T, (2,3)). Other examples are he₄ runs of the category (S, {4}), and John of the category (T, 𝜋). The language generated by the grammar is defined as the set of expressions of the category (S, 𝜋).

The hyprules of the grammar contain variables for integers (n) and variables for sets (set₁, set₂, ...). The following notations are used.

set₁ ∪ set₂ denotes the set obtained as union of the sets set₁ and set₂.
set with n is a compound expression indicating that set contains element n.
set - n is a compound expression denoting the set obtained by removing the element n from set.

The hyprule corresponding with S₄ reads

H₄ : (T₂, set₁) × (IV₂, set₂) → (S, set₁ ∪ set₂)
F₄ : replace the first verb in β by its third person present singular, concatenate (a, β).

This hyprule states that set of the syntactic variables in the sentence is the union of the syntactic variables in the T-phrase and the IV-phrase. An example of an actual rule obtained from H₄ is

H₄ : (T₂(1,2)) × (IV₂, 𝜋) → (S, {1,2})
F₄ : see above.

This rule may be used in the production of he₁ or he₂ runs. Corresponding with S₅, S₆, ..., S₁₃, and S₁₄, we have analogous hyprules. The hyprules corresponding with the rules S₁₄ and S₃ are:

H₄₄ : (T₂, set₁) × (S, set₂ with n) → (S, set₁ ∪ [set₂ - n])
F₄₄ : substitute (a, first occurrence of heₙ in β);
replace all occurrences of heₙ in β by he/she/it and of himₙ by him/her/it according to the gender of the first noun or term in a.

H₃ : (CN, set₁) × (S, set₂ with n) → (CN, set₁ ∪ [set₂ - n]).
F₃ : Replace heₙ in β by he/she/it and himₙ by him/her/it, according to the gender of the first CN in a;
concatenate (a, such that, β).

An actual rule obtained from H₄₄ is

H₄₄ : (T₂, 𝜋) × (S, {2,3}) → (S, {3}).

An application of this rule is the production of John loves him₃ from John
and he \(_2\) loves him \(_2\).

A formalist might object to the hyperrules given above since they implicitly assume that the reader knows what sets are, and what is meant by the symbols \(v, with \) and \(-\). This is, however, not knowledge about operations of the grammatical system, but set theoretical knowledge, and the rules should not be dependent on this knowledge. In appendix 3 of this book it will be shown how these notions can be described by means of purely grammatical tools (viz. by rewriting rules).

Clause (I) required that the expressions of the generated language do not contain any occurrences of syntactic variables. In my approach this requirement is not formalized as a filter or as a condition in a partial rule, but within the system of categories. This is theoretically more attractive, and practically somewhat simpler. Clause (II) requires that in case a rule is applied which removes variables, then there are such occurrences. This clause is also dealt with in the categorial system, as one can see from the following. Let us suppose that the categorial information given in the rules corresponds with the syntactic operations performed by these rules (i.e. if the rule removes all occurrences of a variable, its index is removed from the set mentioned in the category of the produced expression). This assumption can easily be checked from the rules. Assuming this correspondence, the condition \(set_n with n \) in \(H_4\) and \(H_3\) guarantee that these rules are applied only to expressions containing the required occurrences of variables. So instead of formalizing (I) as a condition in a partial rule, it is formalized within the categorial system. This is theoretically more attractive, but practically somewhat more complex.

One observes that the requirements concerning variables can be dealt with in accordance with the aim of using total rules. This is made manageable by using a two-level grammar. Within this system the requirements can be handled about as easy as in a system with partial rules. But the introduction of two levels did not make the system simpler. Therefore I would not say that the requirement of using total rules has led us here to a simpler treatment. In order to see practical advantages of using a two-level grammar, one has to consider a much more complicated situation. Such a situation will be described in chapter 9: the interaction of tense scope, and quantifier scope. But in the present situation the advantage is only of theoretical importance. Therefore one might take in practice the following position. It has been shown that the requirements concerning variables can be incorporated within a system with only total rules. This implicates
that in practical cases there is no need to treat the requirements explicitly in this way. One might use requirements (I) and (II) as they are formulated, assuming the present formalization.

5.4. A theoretical result

The method introduced in this section for eliminating partial rules consists in refining the system of categories. For nouns I gave an example with five subcategories, and for the treatment of variables even an infinite number. One might consider the possibility of applying this method up to the very limit (every expression constituting a single category on its own). By proceeding that far, all partial rules are eliminated from the grammar. This simple idea is followed in the proof of the following theorem (LANDSBERGEN 1981).

5.1. **Theorem.** For every enumerable algebraic grammar $G$ with partial rules, there is a general algebraic grammar $G'$ with total rules, such that $L(G) = L(G')$.

**Proof.** We obtain $G'$ as follows. For each category $C$ of $G$ and each expression $\omega$ of this category, we define a new category in $G'$, denoted by the compound symbol $(C, \omega)$. The only expression of this category is $\omega$. Since for each sort of $G$, the expressions are recursively enumerable, the sorts of $G'$ are recursively enumerable as well (but in general not recursive). For each rule $R$ in $G$ there is a collection of rules in $G'$. If according to a rule of $G$ the expression $\omega_0$ (of category $C_0$) is formed out of the expressions $\omega_1, \omega_2, \ldots, \omega_n$ of the categories $C_1, \ldots, C_n$, then there is in $G'$ a rule producing expressions of the category $(C_0, \omega_0)$ out of expressions of the categories $(C_1, \omega_1), \ldots, (C_n, \omega_n)$. Of course, this rule can be used in only one production, but it is a total rule. Since the rules of $G$ and the expressions of $L(G)$ are recursively enumerable, the rules of $G'$ are recursively enumerable as well. Suppose that the distinguished category of $G$ is $S$ (so $L(G) = S_0$). Then we add for each category $(S, \omega)$, where $\omega$ is arbitrary, a new rule which takes as input an expression of category $(S, \omega)$ and yields as output the expression $\omega$ of category $S$. From this construction it is evident that $L(G) = L(G')$.

5.1. END

The theorem states that every language generated by a grammar with
partial rules can be generated by a grammar with total rules. As such the theorem is not surprising: even finite grammars have a universal generating capacity. The merit of the theorem lies in the method used in its proof. The grammars G and G' do not only generate the same language, but they do so in the same way. The derivational history of a given expression has in G and in G' the same structure. Several properties of G are carried over to G'; for instance, if G consists of concatenation rules only (i.e. if the rules correspond with a context free rules), then the same holds for G'. This correspondence between G and G' means that the proof can be used for restricted classes of grammars as well.

One might be tempted to conclude from the theorem that grammars with partial rules are just notational variants of grammars with total rules, and that it constitutes a justification for writing partial rules in a framework that requires total rules. This is however not the case, since an important property of G can be lost by transforming it to G'. If G is a recursive grammar, where its generated language L(G) is not recursive, then G' is not a recursive grammar. In chapter 2 we have restricted our attention to the class of recursive grammars. Hence the method used in the theorem may bring us outside the class of grammars we are working with. For this class the grammars with partial rules cannot be considered as a notational variant of the grammars with total rules. So the requirement to use total rules is a substantial one. It has a consequence that not every condition on applicability is acceptable: only those are acceptable which can be reformulated as total rules in a recursive algebraic grammar. In previous sections it has been demonstrated that such a reformulation gives rise to a simpler, a better grammar.

6. THE WELL-FORMEDNESS CONSTRAINT

In this section I will discuss some aspects of a principle for syntax due to Partee. It is called 'the well-formedness constraint', and it reads as follows (PARTEE 1979b, p.276):

Each syntactic rule operates on well-formed expressions of specified categories to produce a well-formed expression of a specified category.

The motivation for this principle is related with the aim 'to pursue the linguists goal of defining as narrowly as possible the class of possible grammars of natural languages' (op. cit. p.276). Although this is a completely different aim than the theme of the present chapter, it turns out
that the well-formedness constraint has practical consequences which can be compared with consequences of our algebraic approach, in particular with the requirement of using total rules. I will restrict the discussion of the well-formedness constrain to these aspects.

Our investigations started from algebraic considerations, and the requirement of using total rules constitutes a formal restriction. The value of the requirement was its impact on heuristics. What is the position of the well-formedness constraint in this respect? Is it a formal restriction, heuristic guideline, or something in between these two? I will first try to answer this question by considering the constraint as it is formulated; Partee's interpretation will be considered thereafter. In order to answer the question concerning the formal position of the well-formedness constraint, it is important to have a formal interpretation for the phrase 'well-formed expression'. I will consider two options.

One might decide to associate the phrase 'well-formed expression' with the meaning that this phrase has in formal language theory. The rules of a grammar produce strings over some alphabet, and these strings are called the well-formed expressions over this alphabet. The epithet 'well-formed' is used to distinguish these strings from the other strings over this alphabet. Un-well-formed generated expressions do not exist by definition. It is possible to tell what the well-formed formulas of predicate logic are, but it is not possible to give examples of un-well-formed formulas of predicate logic: if a string is not well-formed, it is no formula at all. If we apply this interpretation to the PTQ grammar, then we have to conclude that _him_ is a well-formed expression (of the category IV) because it is a string produced by the grammar, whereas _her_ is not well-formed (because it is not produced as IV-Phrase). With this interpretation the phrase in the constraint stating that the rules produce well-formed expressions is a pleonasm. The same holds for the input: the only possible expressions of specified categories are the expressions generated by the grammar. With this interpretation the well-formedness constraint just describes how the framework operates, and it is no constraint at all.

One might relate the notion well-formedness with the language generated by the grammar. Then the generated language consists of well-formed expressions, and also all substrings of well-formed expressions are considered as well-formed. Following this interpretation, the constraint says that all intermediate stages in the production process have to be substrings of the produced language. So an acceptable grammar for English has not only to be
adequate (i.e. produce the correct language), but also all the intermediate stages arising from the grammar have to be adequate in a certain sense. This mixture of the notions 'possible grammar' and 'adequate grammar' makes the constraint an unusable one. Suppose that a list of rules of a grammar for English is presented, and one is asked whether they conform the constraint. In order to answer this question one may start to produce some strings by means of the rules, and ask for each application of a rule, whether it is applied to well-formed expressions of English. Suppose that this is the case, then one cannot thereby conclude that all rules from the list obey the constraint, since not all possible derivations have been considered. One has to try and try again, but the definite answer 'yes' cannot be given. It may be undecidable whether an arbitrary grammar satisfies the constraint or not. Of course, this is not a mathematical proof. Such a proof cannot be provided, since the set of English sentences is not a mathematically defined set, but related questions in formal language theory are known to be recursively undecidable. Since the constraint is an undecidable constraint, it cannot be accepted as a restriction on the class of possible grammars (otherwise a more attractive, undecidable, constraint would be 'is an adequate grammar for English').

Partee gives no formal definition of the notion 'well-formed expression'. Conclusions about her interpretation have to be based upon the examples she gives concerning the constraint. As an illustration of the constraint she presents a rule which forms adnominal adjectives from relative clauses. (PARTEE 1979b,p.277). Its syntactic function $F_1$ has the effect that:

$$F_1 \text{(immigrant who is recent) = recent immigrant.}$$

The input for this operation is an ill-formed expression (immigrant who is recent), and therefore she judges that this rule is prohibited by the well-formedness constraint. From this example is clear that she does not follow the first interpretation given above, the second one is closer to her intentions. But she would not consider all substrings of well-formed expressions as being well-formed as well (Partee, personal communication). I expect that John and Peter is well-formed, whereas John and is not. Probably the judgement what well-formed expressions are, is to be based upon linguistic intuitions. In any case, Partee does not give a formal interpretation for the notion 'well-formed expression'. If this notion is not formally interpreted, then the constraint itself cannot be a formal
restriction either. Furthermore, both our attempts to give a formal interpretation were not successful.

I conclude that the constraint has to be considered as a guideline for designing rules. As such it might be useful for its heuristic value, but it has not the position of a formal constraint on the possible kinds for grammars. As a guideline it is a very appealing one, since it aims at a natural way of production; in which no artificial expressions occur as intermediate forms. However, following this interpretation is not without problems. As HAUSSER (1978) remarks, the intuitions concerning the well-formedness of incomplete expressions are rather weak. Hauser gives as example and about the seven dwarfs quickly; well-formed or not? Furthermore, the well-formedness constraint does, even in clear cases, not guarantee that only natural production processes are obtained. Hauser gives as example an operation with the following effect.

\[ F_1 (\text{John kissed Mary}) = \text{Bill walks}. \]

This operator \( F_1 \) is according to the well-formedness constraint an acceptable operator: an intuitively well-formed expression is transformed into well-formed expression. In order to give real content to the principle, restrictions have to be put on the possible effects of a rule. PARTEE (1979a) gives some proposals for such constraints, and her ideas will be followed in chapter 8.

A consequence of Partee's interpretation of the well-formedness constraint brings us back to the discussion of this chapter. Her interpretation says that in all stages of the production process only well-formed expressions are formed. So there is no need to filter out some of them. Neither there is a need to have obligatory rules which have in transformational grammars the task to transform ill-formed expressions into well-formed ones. So in a grammar satisfying the constraint obligatory rules and filters are not needed. Partee even goes further and interpretes the constraint in such a way that they are disallowed. As we found in section 1, such requirements are a direct consequence of the algebraic framework.

Partee's proposal deviates in an important aspect from our framework. Following linguistic practice, she allows for partial rules. As explained in the previous sections, I would not like to follow this idea and I would prefer to use total rules. Some effects of the well-formedness constraint can be dealt with by means of the requirement of using total rules, as will be shown below.
Suppose that in a grammar with total rules there is a rule $S_i$ of which the syntactic operation $F_i$ has the following effect.

$F_i(\text{immigrant who is recent}) = \text{recent immigrant}$.

So the rule operates on a common noun phrase which, according to rule $S_{3,n}$ must be constructed from the common noun immigrant and a sentence of the form $he_n$ is recent. This sentence has to come from the IV-phrase be recent. Since we require that the rules are total, we may also combine this IV-phrase with other term-phrases. So the sentence John is recent also is generated by the grammar, and this is not a correct sentence of English.

This example suggests that an adequate grammar for English with total rules cannot contain a rule which generates recent immigrant in the way $S_1$ does, because one cannot get rid of the phrase be recent. But an easy solution for the production of recent immigrant is available. Follow the advice given in section 4.3, and ask for what we need to produce this phrase. This advice suggests us to ask for an adjective (recent) and a noun phrase (immigrant). So the requirement of using total rules has the same practical impact here as the well-formedness constraint: it is a guideline for obtaining a non-artificial production process. (Note that I did not prove that it is impossible to have a rule like $F_i$ in a system with total rules; I expect that a refined subcategorization might make this possible).

PARTEE (1979b) discusses certain aspects of the formation of (13).

(13) Fewer of the women came to the party than of the men.

Following BRESNAN (1973), this sentence is derived from the (ill-formed) sentence (14) by means of an operation called Comparative Ellipsis.

(14) Fewer of the women came to the party than of the men came to the party.

This is in its turn derived from (15) by Comparative Deletion.

(15) Fewer of the women came to the party than $\exists$ many of the men came to the party.

As Partee says, the production of (13) is a difficult case for the well-formedness constraint since it uses the ill-formed source (14). Partee says: 'Unless further analysis of these constructions leads to a different kind of solution, they would seem to require the admissibility of ungrammatical intermediate stages. (Note that the derivations in question give semantically reasonable sources, so any reanalysis has a strong semantic as well as syntactic challenge to meet).' (PARTEE 1979b,p.303,304).
For our requirement of using total rules this production process is problematic as well. It is no problem that the rules of comparative deletion and comparative elipsis are partial rules, since they are meaning preserving. But the production of the ill-formed sentence (14) is problematic since we cannot get rid of this sentence: we cannot filter this sentence out, we may not have it as an expression of the generated language, and we may not use its embeddings (cf. the discussion concerning recent immigrant). But why follow this approach? Maybe one judges that a source like (14) or (15) expresses the semantic content of the comparatives more completely than comparatives. Or one wishes to explain the semantic relations between all variants of comparatives by generating them from the same kind of source.

In transformational grammar this might be valid arguments, no other formal tools than transformations are available. In a Montague grammar there is a semantic component in which such semantic relations can be formally expressed. So if we do not need such a source for syntactic reasons we may try another approach. The requirement of using total rules guides us toward asking what we need. In order to make a sentence of which the kernel consists of two terms and a verb phrase, we need two terms and a verb phrase. Therefore we should introduce a three-place rule

\[ P_{605} (\text{John}, \text{Bill}, \text{see} \, \text{women}) = \text{John sees more women than Bill}. \]

The semantic component has to express what is compared; the syntax needs no to do so.

Another rule might compare of two nouns in which degree they are involved in a certain property.

\[ P_{606} (\text{man}, \text{boy}, \text{come to the party}) = \text{fewer of the men come to the party than of the boys}. \]

One may also compare two terms for two verb phrases

\[ P_{607} (\text{John}, \text{Bill}, \text{see men, meet women}) = \text{John sees more men than Bill meets women}. \]

These examples do not provide for a treatment of the comparative. They just illustrate the kind of solution one might search for in a framework with total rules. Variants are possible: for instance, one might introduce compound quantifier phrases like \textit{fewer of the men than of the boys}, and use instead of \( P_{606} \) a rule with two arguments. Note that all these attempts to find total rules, are in accordance with the well-formedness constraint.