CHAPTER VI

VARIANTS AND DEVIATIONS

ABSTRACT

In this chapter the impact of the algebraic framework on the design of grammars, is illustrated by considering several proposals from the literature. Most of these proposals contain details which are not in accordance with the framework. It will be shown that these proposals can be improved by adopting an approach which is in accordance with the framework, without losing the semantic effect the proposal was designed for. Other proposals present acceptable variants for certain details of the framework.
1. INTRODUCTION

On the basis of several proposals from the literature, I will illustrate in this chapter what the practical consequences are of the framework we have developed in chapters 1 and 2. Some of the proposals were already discussed in JANSSEN 1978a. The rules from the proposals will not be adapted to the way of presentation used up till now, but they are quoted in the way they were formulated in the original papers. I expect that this will cause no problems. Only the formulas of IL are sometimes adapted to our notations (e.g. $\lambda$ instead of $\Lambda$). Some of the proposals concern variants which are in accordance with the framework, but most are not. The objections against these proposals, however, concern in most cases only a minor detail of the paper, and my criticism should not be taken as a criticism on the paper as a whole. On the contrary, most of the papers I like very much, and that was a reason for studying them in detail. I will not consider proposals which are presented in such a way that it is evident that they are intended as a non-compositional component of the system (e.g. the indexing component for variables of COOPER & PARSONS 1976, and the interpretation strategy for pronouns of BARTSCH 1979). Rather I will discuss aspects of proposals which seem at first glance to be in accordance with the framework, but which at closer investigation appear not to be. Such examples exhibit that the practical consequences of the framework are sometimes not well understood. These examples are collected here to provide as illustrations of the framework: non-examples too can be very instructive. I hope that the examples give the reader an improved understanding of what it means to design a Montague grammar. As a matter of fact, my personal experience with the examples discussed here, was a great stimulan for the research presented in this book: discovering the foundations of Montague grammar, and investigating the practical consequences of these fundamental properties.

The structure of our framework, as developed in chapters 1 and 2, is presented in figure 1. The arrows 2, 5, and 7, are homomorphisms, and the arrows 3 and 6 are derivers. The examples we will consider are grouped according to the arrow representing the component where the deviation from the framework can be located. The number of the arrow indicates the section where that group of examples will be considered.
\[ \begin{align*}
T_E & : \text{Term algebra of grammar for English} \\
& \downarrow \quad 2 \\
IL & : \text{Intensional Logic} \\
& \downarrow \\
IL' & : \text{Translation of English into algebra derived from intensional logic} \\
& \downarrow \\
M & : \text{Meanings for intensional logic} \\
& \downarrow \\
M' & : \text{Meanings for English}
\end{align*} \]

**Figure 1.** The framework

The framework of Montague grammar constitutes a framework which guarantees that one is working in accordance with the principle of compositionality. Deviations from this framework are not just deviations from some arbitrary mathematical system, but from a framework that is designed with the purpose of both obeying the principle, and being at the same time as general as possible. If one violates this framework, then there is a great risk that one does not only disturb the framework, but also the underlying principle of compositionality. The ultimate consequence may be that one does not describe a semantics at all. In the discussion it will turn out that the practical examples of violations of the framework in most cases indeed yield an incorrect (i.e. unintended) semantics, or no semantics at all. In such cases the framework guides us toward a correct solution. In other cases, where the proposal did not give rise to an incorrect semantics, the principle suggests another kind of solution that is simpler than the original proposal. These aspects exhibit the value of (the formalization of) the principle of compositionality as a heuristic tool.

In the light of the above remarks, it is useful to give a characterization of what are harmful deviations of Montague's framework, and what are harmless variants. This characterization can be given at the hand of figure 1. It is harmless to change the language of which the semantics is given; to change the kind of logic used as auxiliary language, or to change the kind of meanings obtained. All algebras in the figure may be replaced by other algebras. But the algebraic relations between them should not be changed; the algebraic properties of the arrows should not be disturbed. Homomorphisms should remain homomorphisms, and derivers should remain derivers. These are the properties which guarantee that the principle of compositionality is obeyed.
2. THE USE OF SYNTACTIC INFORMATION

2.1. Introduction

Some proposals from the literature contain a translation rule which depends on the actual expression on which the syntactic rule operates. This means that there are different semantic operations for the various syntactic possibilities. Hence there is a one-many correspondence between the syntactic operations and the semantic operations. Then the mapping from the syntactic algebra to the semantic algebra cannot be a homomorphism. Consequently the framework is not obeyed: the relation indicated in figure 1 by arrow 2 has to be a homomorphism. But also the principle of compositionality itself is violated. In this situation the meaning of the compound expression is not determined by the information which syntactic rule is used and what the meanings of the parts of the expression are, but also information about the actual expressions operated upon is needed. This situation is not a source of great practical problems, since, at least in the examples considered below, the rule can easily be reformulated in such a way that the framework is obeyed.

2.2. Easy to please

This example concerns a variant of Montague grammar proposed in Partee 1973. The expressions generated by the grammar contain labelled brackets which indicate the syntactic structure of the expressions. Partee wants to account for the occurrence of verb phrases in conjunctions and infinitives. Examples are given in (1) and (2)

(1) Few rules are both explicit and easy to read.
(2) John wishes to see himself.

For the production of these sentences a rule called 'derived verb phrase rule' is used. The rule is so close to a correct formulation that I would not like to call it a violation of the framework. It is rather an illustrative slip of the pen.

Derived verb phrase rule (Partee 1973)

If \( \phi \in P_t \) and \( \phi \) has the form \( \mathcal{L}_T[A\_he_{\tilde{\mathcal{L}}}A]\_IV[\_a] \), then \( F_{104}(\phi) \in P_{IV} \), where \( F_{104}(\phi) = a' \), and \( a' \) comes from \( a \) by replacing each occurrence of \( he_{\tilde{\mathcal{L}}} \), \( him_{\tilde{\mathcal{L}}} \), \( him_{self} \) by \( he^* \), \( him^* \), \( him_{self}^* \) respectively.
Examples:
\[ F_{104}(he_i \text{ sees } him_{i \text{self}}) = see \text{ him}_{i \text{self}} \]
\[ F_{104}(he_j \text{ is easy to please}) = be \text{ easy to please}. \]

Translation rule

If \( \phi \in P \) and \( \phi \) translates into \( \phi' \), then \( F_{104}(\phi) \) translates into \( \lambda x_1 \phi' \).

From the formulation of the translation rule it might not be evident that the translation rule uses syntactic information. But this becomes clear if one realizes that in order to decide what the actual translation is (\( \lambda x_1 \phi \) or \( \lambda x_2 \phi \) or \( \ldots \)), one needs to know the index of the first word of \( \phi \). So syntactic information is used. The correction of this rule is rather simple, in analogy of term-substitution in PT, we give the syntactic operation an index as parameter: so \( F_{104} \) is replaced by \( F_{104,i} \). In a later paper (PARTEE 1977a) the rule is corrected in this way.

2.3. The horse Cannonero

DELACRUZ (1976) considers expressions like the horse Cannonero. Such expressions belong to a category \( \tilde{T} \) and they are generated by the following rule:

S3.1 If \( \alpha \in B_T \) and \( \zeta \in B_{CN} \), then \( F_{21}(\zeta, \alpha) \in P_{\tilde{T}} \) provided that whenever \( \alpha \) is of the form \( he_{i \tilde{N}} \), \( F_{21}(\zeta, \alpha) = \alpha \); otherwise \( F_{21}(\zeta, \alpha) = the \; \zeta \; \alpha \).

Examples:
\[ F_{21}(\text{horse,Cannonero}) = \text{ the horse Cannonero} \]
\[ F_{21}(\text{horse,he}_j) = he_j. \]

Translation rule:

T3.1 If \( \alpha \in B_T, \zeta \in B_{CN} \) and \( \alpha, \zeta \) translate into \( \alpha', \zeta' \) respectively, then \( F_{21}(\zeta, \alpha) \) translates into \( \alpha' \) if \( \alpha \) is of the form \( he_{i \tilde{N}} \); otherwise \( F_{21}(\zeta, \alpha) \) translates into
\[ \lambda P \exists y [\forall x [\zeta'(x) \wedge \lambda P \lambda z [\tilde{v} P] (\lambda x [\tilde{v} x ](\lambda \alpha'(x)))](\lambda x) (x) ] \leftrightarrow x = y \lambda [\tilde{v} P](y). \]

Translation rule T3.1 depends on the form of the input expressions of the syntactic rule, so it violates the framework. An attempt to formulate the translation rule as a single polynomial in which no syntactic information is used, would require an extension of IL with an if-then-else construction, and with a predicate which discriminates on semantic grounds between variables and constants. I doubt whether the latter is possible. But a simple solution can be found in the syntax. The construction
described by Delacruz provides evidence that we should distinguish among the terms the (sub)categores Proper Names and Indexed Pronouns. Rule 83.1 applies only to the category of Proper Names, or alternatively, rule 83.1 is a partial rule which only applies to the subcategory of Proper Names. This approach describes more clearly what the situation is, than the original rule does, or a semantic reformulation would do. A final remark about the formula (3) given by Delacruz. It is no: the simplest polynomial expressing the intended semantic operation. I would use instead:

\[
\lambda P \exists y \forall x \left[ \zeta (x) \land a' \left( \lambda z [x = z] \leftrightarrow x \equiv y \right) \right].
\]

3. NON-POLYNOMIALLY DEFINED OPERATORS

3.1. Introduction

The algebra of formulas into which we translate, is obtained from the algebra of IL-expressions by means of restructuring this algebra. This means that new operations may be added, another type structure may be put on the elements, and old operations may be omitted. Following MONTAGUE 1970b, we require that in this process of restructuring, all operations are polynomial operations over IL. This restriction ensures that the interpretation homomorphism for IL determines a unique interpretation for the derived algebra. If one uses operations on IL expressions which are not defined as a polynomial, then there is a great risk of disturbing the homomorphic interpretation. This would mean that we have no interpretation for the derived algebra, thus we are not doing semantics at all! Therefore it is advisable to use only polynomially defined operators.

When we consider examples of operators which are not polynomially defined, it will turn out that in all cases the operator can be replaced by a polynomially defined operator which has the desired properties. Replacement of a non-polynomially defined operator by a polynomially defined one, is (in all these cases at least) a simplification. Thus the requirement of working in accordance with the framework guides us toward a simpler treatment than originally was proposed. This consequence illustrates the heuristic value of the principle of compositionality. So there is, from a practical point of view, no reason to use nonpolynomially defined operators. Theoretical aspects of non-polynomially defined operators will be discussed in section 4.
3.2. John who runs

The approach to natural language followed in BARTSCH 1979 is closely related to the approach followed in the field of Montague grammar. The differences which appear in this and the next example are that the treatment of intensions is different, and that the generated language is somewhat more abstract since it contains brackets and other auxiliary symbols. These differences do not influence the aspect I wish to discuss. Bartsch presents a rule for the formation of term phrases containing non-restrictive relative clauses. Such expressions are formed from a term and a relative sentence by the following rule (BARTSCH 1979, p.45).

S4. If α is a term and β a relative sentence, then β(α) is a term. [...].

The corresponding translation rule reads

T4. If α' is the translation of the term α and RELT(λx β'(x)) is the translation of the relative clause β from S*, then (RELT(λx β'(x)))(α') is the translation of β(α), and for all terms α with α' = λP(...P(ν)...)
we have: (RELT(λx β'(x)))(λP(...P(ν)...)) = λP(...β'(ν) & P(ν)...).

The translation rule evidently is no polynomial over IL. The rule works well for the translation one usually obtains for term phrases. For every man the standard translation is (5), and for this case the rule is perfect.

(5) \[ λP ∀u[man'(u) → P(u)]. \]

In case an alphabetical variant of formula (5) is used, the situation changes. Consider (6).

(6) \[ λQ ∀u[man'(u) → Q(u)]. \]

Translation rule T4 as formulated above does not apply: it is not defined for this representation. Probably we have to be more liberal and consider T4 to be defined for all expression of the indicated form. But there are also formulas which are equivalent to (5) and which are certainly not of the same form. Let R be a variable of the same type as the translation of terms, and consider (7)

(7) \[ λQ ∀u[λR[R(man')] → R(0)](λP[P(ν)]). \]

Rule T4 is not defined for this representation. Moreover, application of
the rule to the subexpression $\lambda P[P(v)]$ would yield a semantically incorrect result.

This discussion shows the consequence of T4 that it is no longer allowed to exchange logically equivalent formulas. The rule defines a partial function between IL formulas; it is an instruction for formula manipulation, not for compositional semantics. A reaction to these objections against a rule like T4 might be that one adds to the rule a clause stating that in case a formula is not of the mentioned form, it must be reduced to that format. This instruction obscures a lot of problems since it does not say how such a reduction is to be performed. A discussion of the problems arising with this attempt to correct in this way a non-polynomial rule, will be given in section 4.

Can the basic idea of the operation be described in a polynomial way? The desired effect is the replacement of $P(v)$ by $\beta'(v) \land P(v)$. This can be obtained giving $\lambda z[\beta'(z) \land P(z)]$ as argument of $\lambda P[...P(v)...]$. We must take care furthermore of the binding of the variable $P$. Thus we come to a version of T4 which is in accordance with our formalisation of the semantic compositionality principle: T4'. Let $\alpha'$ be the translation of the term $\alpha$ and $\gamma'$ the translation of the relative clause $\gamma$. Then the translation of the compound expression $\gamma(\alpha)$ is:

$$\lambda Q(\alpha'(\lambda z[\gamma'(z) \land Q(z)])) \tag{8}$$

One observes that it is not needed to follow the method hinted at above: to define the intended semantic operator by defining an operator on specially selected representations. The formulation of T4' uses the polynomial expression (8). It is more exact and simpler than the original formulation, and it works well for all formulas equivalent with $\alpha'$ or $\gamma'$.

RODMAN (1976) also considers the formation of terms containing a non-restrictive relative clause. He presents a rule which produces such terms out of a term and a sentence, where the sentence contains a suitable variable. His translation rule reads:

If $\alpha \in P_{T}$, $\phi \in P_{T}$ and $\alpha, \phi$ translate into $\alpha', \phi'$ respectively, then $P_{3, n}(\alpha, \phi)$ translates into $\lambda P(\lambda x_{n}[\phi' \land \gamma'(x_{n})])^{(\lambda \alpha')}$.

This is not the simplest formulation of the polynomial. By one time $\lambda$-reduction one obtains
\[ \lambda \in \alpha' \left( \lambda x_\eta \left[ \phi' \land \psi(x_\eta) \right] \right) \].

One observes that this rule is almost identical with the version given above of Bartsch's rule. The differences are due to a different treatment of intensions, and the fact that Bartsch uses the intermediate stage of a relative sentence. Concerning the meaning of the relative clause construction the two solutions are essentially the same. This shows another advantage of the restriction to use only polynomials. It gives us a uniform representation of meanings, and different polynomials can be compared with each other by using known methods.

3.3. Das Mädchen gibt den Apfel dem Vater

BARTSCH (1979) represents a rule which combines an n-place verb-phrase with a term to produce an (n-1)-place verb-phrase. The rule does not in advance fix the order in which the term positions should be filled in: a rule has as parameter a number indicating which position is to be filled. By varying the sequence of 'filling in' one can generate the German versions of The girl gives the father the apple, The girl the apple the father gives, etc. (the German versions all seem to be parts of correct sentences). The result of substituting the term a on the i-th place of a verb \( \beta \) is indicated by \((a,\xi)\beta\). The syntactic rule reads (BARTSCH 1979, p.27)

\[(S1)\] If \( \beta' \) is a \( \mathcal{V}^i(n\text{-place verb}) \) with the set of \( n \) term-places, \( k_\eta \epsilon \mathcal{K} \), and if \( a' \) is a \( \mathcal{T}(\text{term}) \), then \((a',\xi)\beta'\) is a \( \mathcal{V}^{n-1} \) with the set of term-places \( \mathcal{K} - \{\xi\} \).

For this we write \((a',\xi)\beta'\).

\[(T1)\] If \( \alpha'' \) is the translation of \( \alpha' \) as a \( \mathcal{T} \), and \( \lambda x_j,\ldots,x_m \beta''(x_j,\ldots,x_m) \) with \( n \) places, the translation of \( \beta' \) as a \( \mathcal{V}^i \), then the translation of \( \alpha''(a',\xi)\beta' \) is

\[ \lambda x_j,\ldots,x_m \left( \alpha''(\lambda x_{\eta}(\beta''(x_j,\ldots,x_m))) \right) \]

with \( x_{\eta} \) as the variable that precedes \( x_j \) and ' \( x_{\eta} \) as the variable that

This rule is defined only for special representations of the meaning of the term, and, for reasons related to the ones mentioned in the previous example, it is not acceptable as a rule for compositional semantics. Again the idea behind the formulation of the rule can be formulated by means of a polynomial, thus becoming an acceptable rule and obtaining a shorter formulation:
If \( a'' \) is the translation of \( a' \) as a T and \( \gamma'' \) is the translation of \( \gamma' \) as a \( V^n \), then the translation of \( (a', \varepsilon') \delta' \) is

\[
\lambda y_1', \ldots, y_i', y_1, \ldots, y_m (a''(\lambda y_1' y'_{i-1} y_{i+1} \ldots y_m ))
\]

with \( y_i' \) as the variable that precedes \( y_i \) and \( y_i' \varepsilon \) as the variable that follows \( y_i \).

3.4. Woman such that she loves him

Below we have the rule for the formation of restrictive relative clauses from PTQ (MONTAGUE (1973)). This rule reads as follows (notation adapted)

\( S3,n : CN \times S \rightarrow CN \)

\( F3,n : \) Replace \( he_n \) in \( \beta \) by \( he/she/it \) and \( him_n \) by \( him/her/it \), according to the gender of the first CN in \( a \);

Concatenate \( \langle a, such that, \beta \rangle \)

\( T3,n : \lambda x_n [a'(x_n) \wedge \beta'] \).

This rule gives rise to an incorrect translation in case \( \beta' \) contains an occurrence of \( x_n \) which should not be bound by the \( \lambda \)-operator which is introduced by the translation rule. An example is the production presented in figure 2.

The translation obtained reduces to (10).

\[
(10) \quad \lambda x_i'[woman(x_i') \wedge love_s(x_i', x_i') \wedge run(x_i')].
\]

The produced CN-phrase may be used in the production of some sentence, and
in this process John might be substituted for him. Then the sentence contains a CN-phrase woman who loves John. But the translation contains (10), expressing that the woman loves herself.

In order to avoid this collision of variables, Thomason has presented the following translation rule (footnote to PFO, THOMASON 1974, p.261, presentation adapted)

\[ T3', n: \lambda x_m [\alpha'(x_m) \land \psi] \]

where \( \psi \) is the result of replacing all occurrences of \( x_n \) in \( \beta' \) by occurrences of \( x_m \), where \( m \) is the least even number such that \( x_m \) has no occurrence in either \( \alpha' \) or \( \beta' \).

One observes that \( T3' \) uses an operation on expressions which is not an operation of IL: the replacement of certain variables by a variable with a special index. We might try to describe the required change by means of IL operators. It is easy to obtain the replacement: \( \lambda \)-conversion does the job:

\[ T3'' \quad \lambda x_m [\alpha'(x_m) \land \lambda x_n [\beta'](x_m)]. \]

Where \( m \) is as in \( T3' \)

It is, however, impossible to extend IL with a operator \( Gr \) which yields the greatest non-used even index. This can be shown by providing two equivalent formulas for which this operator would yield non-equivalent results. Let \( \phi \) be an arbitrary formula. \( Gr(\phi \land x_2 = x_2') \) would be \( x_4 \), whereas \( Gr(\phi \land x_4 = x_4') \) would be \( x_6 \), what contradicts the law concerning substitution of equivalents.

We just observed that Thomason's rule contains instructions which essentially use the particular IL representation of the meaning of the relative clause. Nevertheless the rule as a whole is correct in the sense that it corresponds with an operation on the set of meanings. If the translation of the common noun or of the sentence is replaced by an equivalent formula, the application of \( T3' \) (or \( T3'' \)) gives for all cases an equivalent result. This is due to the fact that the syntactic operation we called \( Gr \) is used only in the context of renaming bound variables. So \( T3' \), although violating the framework, does not violate the compositionality principle.

But there is a more direct solution to the problem raised by Montague's rule. A translation rule for relative clause formation which does obey the restriction of using polynomially defined operators is

\[ T3''' \quad \lambda P \forall x_n [P(x_n) \land \beta'](\alpha'). \]

This translation rule yields a correct result for the problematic case presented in figure 2, due to the conditions for \( \lambda \)-conversion. In case \( \alpha' \) does not contain free occurrences of \( x_n \), then \( T3''' \) reduces to \( T3 \), otherwise
\( \lambda \)-conversion evokes change of bound variables. One observes that the formulation of \( T_3'' \) is simpler and much more elegant than the formulation of \( T_3' \) (or \( T_3'' \)). Moreover \( T_3'' \) is in accordance with the variable principle, whereas \( T_3'' \) and \( T_3' \) are not (see chapter 8). The simple formulation of \( T_3' \) is possible because the syntactic problem of collision of variables is not dealt with in the translation rule, but on a more appropriate level: in the rules for \( \lambda \)-conversion which, by their nature, are syntactic operations on IL expressions.

4. OPERATORS DEFINED ON REPRESENTANTS

In all examples from section 3, a rule was defined which works well in the situations one usually meets in practice. In two of the examples the rule does not work well in unusual situations. Often one is tempted to design rules in such a way that as a consequence they have this character. One defines a rule for the formulas one is familiar with, using well known properties of these formulas. Then one hopes that an operation defined on these special formulas in fact defines an operation on the associated meanings. In the present section it will be investigated under which circumstances this hope will become reality. It will be shown that it is not easy to create such circumstances.

Besides the practical motivation given above for considering non-polynomially defined operators, there is a theoretical argument. In the introduction of section 3 I mentioned that an operator which is not defined by means of a polynomial over IL violates the framework, and bears the risk of violating the compositionality principle itself as well. But not all non-polynomial operators do so. In 3.4 we have met a non-polynomially defined operator which could be replaced by a polynomially defined one. From chapter 2, section 7, we know that an operator which is defined on the language IL, and which respects all homomorphic interpretations, can be described by means of a polynomial. But this does not imply that all non-polynomially defined operators which respect the interpretation of IL, indeed can be described by means of a polynomial. This is due to the rather strong condition of the theorem that all homomorphic interpretations are respected. We are not interested in all meaning algebras we might associate with IL, but only in some of them. We want to interpret \( P(x) \) as the application of a function to an argument, we want the interpretations of \( \phi \wedge \psi \) and of \( \psi \wedge \phi \) to be the same, and we want several meaning postulates to be
satisfied. Therefore the theorem does not give us the guarantee that every operation on IL which respects the usual interpretations of IL indeed can be defined by means of a polynomial. These considerations constitute a theoretical argument for considering non-polynomially defined operators. But the practical argument given above is, in my opinion a more interesting reason for doing so.

The definition of an operation on IL formulas is acceptable if (and only if) it does not disturb the compositionality principle, i.e. if with the operation on formulas we can associate an operation on meanings. This can only be done if for logically equivalent formulas the operation yields equivalent results. So when defining an operation by defining it for special formulas, every special formula $\phi$ has to be considered as a representant of the class of formulas equivalent to $\phi$.

A mathematical description of the situation is as follows. The set of formulas (of a certain type) is divided into equivalence classes. A class consists of formulas which have the same meaning in all models. Remember that we defined the meaning of an IL formula of type $\tau$ as a function which assigns to an index and a variable assignment some element in $D_\tau$ (so expressions in the same class represent the same function). In each class representants are defined, and we wish to define an operation on the whole class by defining an operation for the representants. If in each class there is only one representant, we are in the situation presented in figure 2b, if there are more, then we are in the situation of figure 2a. We want to know when a definition on a representant defines an operation on the whole class.

![Figure 2a Several representants](image1)

![Figure 2b One representant](image2)

When defining an operation on an equivalence class by a definition on its representant, two aspects can be distinguished.

A) the definition of an operation on the representants
B) A proof that this defines an operation on the whole class.

As for A) we have to fulfill the following two requirements.

A1) One has to describe exactly the set of formulas for which the operation is defined, i.e. one has to define a *recursive* set of representants.

A2) One has to define for all representants what the effect of the operation is, i.e. we have to define an *effective* operation on the set of representants.

This kind of requirements we have met before: define a set and operations on this set (e.g. define a language and a translation of this language). Therefore it seems attractive, in the present context, to define the set of representants by means of a grammar generating the expressions in the subset. In order to be sure that the operation is defined for all formulas in the subset, one might formulate the clauses of the operation parallel to the grammatical rules generating the subset. This will probably be more complicated than a polynomial definition. But other techniques for defining the set of representants and the operation are possible as well.

As for B) I will consider first the situation described in figure 2a: one representant in each class. Here the definition of an operation on the representant automatically determines a semantic operation on the whole class: the interpretation of the operation on the representant is the semantic operation on the interpretations of all expressions in the class.

But how can we be sure that we are in the situation of figure 2a? Proving that we are in such a situation means that we have to prove the existence and unicity of a representant for each class. I do not know whether there exists for each type a recursive set of unique representants. Assuming the possibility of such a set, it remains the question how to prove the existence and unicity of such representants. It seems attractive to do this by providing an algorithm which transforms a given formula into the representant of its equivalence class. This expresses the idea we met in section 3.2:

if a formula is not in the required form, it should be transformed into the required form. This approach is, however, in the relevant cases impossible, as follows from the following theorem.

4.1. **Theorem.** Let $s, t \in \text{Ty}$. Define the equivalence relation $\sim$ on $\text{ME}_{<s, t,t_1>}$ by $\phi \sim \psi \iff \models \phi = \psi$. Let $R \subseteq \text{ME}_{<s, t,t_1>}$ be a (recursive) set of representants such that for each equivalence class there is one element in $R$. Let $f: \text{ME}_{<s, t,t_1>} \to \text{ME}_{<s, t,t_1>}$ be a function which assigns to each formula the representant of its equivalence class. Then $f$ is not recursive. The
same result holds if the expressions are from \( \text{ME}_{<t,t>} \).

**Proof.** Let \( \phi \in \text{ME}_{t} \), \( P \in \text{VAR}_{Q} \), \( Q \in \text{VAR}_{t} \). Then the following statements are equivalent

\[
(11) \quad \vDash \phi \quad (\phi \text{ is logically true})
\]
\[
(12) \quad \vDash \lambda P \alpha Q(\phi) = \lambda P \alpha Q(\forall x[x=x])
\]
\[
(13) \quad \vDash f(\lambda P \alpha Q(\phi)) = f(\lambda P \alpha Q(\forall x[x=x])).
\]

Note that in (13) semantic equality of the formulas is expressed. Since for each class there is one representant, (13) is equivalent with (14)

\[
(14) \quad f(\lambda P \alpha Q(\phi)) \equiv f(\lambda P \alpha Q(\forall x[x=x])).
\]

Note that in (14) syntactic identity of the by \( f \) obtained formula is expressed. So, if \( f \) is recursive, the question whether \( \phi \) is logically true is decidable: calculate \( f(\lambda P \alpha Q(\phi)) \) and \( f(\lambda P \alpha Q(\forall x[x=x])) \), and see whether they are identical. Since \( \text{IL} \) is undecidable, \( f \) cannot be recursive. For \( \text{ME}_{<t,t>} \) analogously. Note that we did not use the recursiveness of the set of representants.

4.1. END

The translations of expressions of the \( \text{PTQ} \) fragment are all of the form \( <s,<t,t>> \) or \( <t,t> \). The same holds for the expressions of the fragments considered in all examples. The theorem says that there is no algorithm which transforms a formula into its representant. If one tries to define an operation on a class by an operation defined on its representants, then one has to find some other way of proving the existence and uniqueness of a representant.

Next we will consider the situation described in figure 2b: a multiple set of representants is allowed for. There is no doubt that such a set exists: \( \text{ME}_{t} \) itself is a recursive set of multiple representants of \( \text{ME}_{t} \). But also here a complication arises.

4.2. **Theorem.** Define \( \sim \) as in the previous theorem. Let \( R \) be a (recursive) set such that for every equivalence class there is at least one equivalent element in \( R \). Let \( f: \text{ME}_{<o,<t,t>>} \rightarrow \text{ME}_{<o,<t,t>>} \) be a recursive function that
assigns to every formula an equivalent formula in \( R \). Then for \( r_1, r_2 \in R \) it is undecidable whether \( \models r_1 = r_2 \).

**Proof.** As observed in the proof of the previous theorem \( \models \phi \) is equivalent with \( \models f(\lambda \varphi \psi(\phi)) = f(\lambda \varphi \psi(\forall x[x = x])) \). If equality is decidable for elements of \( R \), then the equality is decidable for these two formulas, so it is decidable whether \( \phi \) holds. This gives rise to a contradiction since IL is undecidable. Note that we did not use the recursiveness of the set of representatives.

**4.2. END**

This result means that we have to prove that an operation defined for representants yields for equivalent formulas an equivalent result, without knowing what the equivalent formulas look like.

The above discussion shows that it is, generally spoken, a complicated and extensive task to define a function by defining a function on specially selected representations. Probably this can only be done in practice, if one considers a situation with a special structure which has the effect that all proofs become drastically simplified. But if the situation is such a special one, it may be expected that the same effect can be obtained in a more direct way: by using polynomials. This is illustrated in the examples considered in section 3. So far there is no evidence that there is any advantage in defining operations in a non-polynomial way.

**5. NEW SYMBOLS IN IL**

**5.1. Introduction**

Some authors extend the language of IL with new symbols. These symbols should obtain an interpretation by means of an extension of the interpretation homomorphism for IL. For each point of reference some semantic object of the right type has to be associated with the new symbol. If the new symbol is an operator, its interpretation has to be a function operating on the interpretation of its argument. If these requirements are not met, then the interpretation homomorphism of IL cannot be extended to an interpretation homomorphism for the extension of IL. Consequently arrow 5 in figure 1 is no longer a homomorphism. Hence arrow 7 is not a homomorphism either. Then the translation homomorphism 2 cannot be continued to an interpretation homomorphism, and this means that the principle of compositionality
is violated. Below we will consider two examples of new symbols.

5.2. Shake John awake

Dowty (1976) treats, among others, the semantics of factive constructions such as shake John awake. In order to do so, he extends the language of intensional logic with two operators: CAUSE and BECOME. Interesting for our discussion is his treatment of CAUSE. In order to define its interpretation Dowty adds "to the semantic apparatus of PTQ a selection function f that assigns to each wff φ and each i ∈ I a member f(φ, i) of I. [Intuitively f(φ, i) is to be that i' most like i with the (possible) exception that φ is the case [..]]". Then the interpretation of CAUSE reads: "If φ, ψ ∈ NL then (φ CAUSE ψ)A1,i,j,S is 1 if and only if [φ ∧ ψ]A1,i,j,S is 1 and [ψ]A1,f(φ, i),j,S is 1."

The function f is defined on IL-expressions and not on the interpretations of these expressions. As a consequence CAUSE is an operator on IL-expressions and not on the meanings they represent. This is illustrated as follows. The definition of f allows that for some φ, ψ, i holds that f(φ ∧ ψ, i) ≠ f(ψ, i). This may have as a consequence that [(ψ ∧ φ)CAUSE ψ]A1,i,j,S = 1 whereas [(ψ ∧ φ)CAUSE φ]A1,i,j,S = 0. The main features of an example of such a situation are as follows. Let [(ψ ∧ φ)∧ ψ]A1,i,j,S = 1, so [(ψ ∧ φ)∧ ψ]A1,i,j,S = 1. Suppose that f(ψ ∧ φ, i) = i' and [ψ]A1,i',j,S = 1. Then [(ψ ∧ φ)CAUSE ψ]A1,i,j,S = 1. Suppose moreover that f(ψ ∧ φ, i) = i" and [ψ]A1,i",j,S = 0. Then [(ψ ∧ φ)CAUSE ψ]A1,i,j,S = 0.

In the above example the principle of compositionality is not obeyed: two equivalent formulas cannot be interchanged 'salva veritate'. Moreover the meaning of CAUSE described above is incorrect since, intuitively, [(ψ ∧ φ)CAUSE ψ]A1,i,j,S = [(ψ ∧ φ)CAUSE ψ]A1,i,j,S. A correction is possible by taking as domain for f the intensions of formulas: f assigns to each d ∈ D<φ,ψ> and i ∈ I a member f(d, i) ∈ I. Then a situation as described above is automatically excluded. The interpretation of CAUSE now becomes as follows:

[φ CAUSE ψ]A1,i,j,S = 1 if and only if
[φ ∧ ψ]A1,i,j,S = 1 and [ψ]A1,i,j,S = 1, where
i = f([ψ ∧ φ]A1,i,j,S, i).
This definition has the property that if $\phi$ CAUSE $\psi$, then for all tautologies $\eta$ holds that $(\phi \land \eta)$ CAUSE $\psi$, a problem of the same nature as the problem we met in chapter 4 concerning the complements of belief-sentences.

5.3 I and You

GROENENDIJK & STOKHOF (1976) give a treatment of the pronouns I and You. For this purpose, they extend the model for IL. Usually the denotation of an IL expression is defined with respect to a world $i$ and a time $j$; these $i$ and $j$ are called 'indices'. Groenendijk & Stokhof extend the set of indices with three components: $j_0, s$ and $h$. Here $j_0 \in J$ is the moment 'now', i.e. the moment of utterance, $s \in A^{i \times J}$ is a function which for a point of reference $(i, j)$ yields the speaker at that moment, and $h \in A^{i \times J}$ is a function yielding the hearer. The interpretation of $\alpha$ may depend on $i, j, j_0, s$ and $h$, and we write $s^{A,i,j,s,h,j_0}$ and $h^{A,i,j,s,h,j_0}$ for the interpretation of $\alpha$. The language of IL is extended with the constants $i$ and $y$ of type $<s,e>$; these constants occur in the translations of the pronouns I and you respectively. The goal they wish to reach is described as follows. (op. cit. p. 308). 'What we want our interpretations to express is that the extension of the constants $i, y$ are the possible individuals which are speaking now, spoken to now respectively. This would explain the tautological character of a sentence like (15) and the contingent character of sentences like (16)'.

(15) I am the speaker
(16) I will not be the speaker

Groenendijk & Stokhof define $F(i) = s$ and $F(y) = h$. Furthermore they define

$$s^{A,i,j,s,h,j_0} = F(i)(i,j_0) \quad (= s(i,j_0))$$

and

$$h^{A,i,j,s,h,j_0} = F(y)(i,j_0) \quad (= h(i,j_0)).$$

So for any point of reference the interpretation of $i$ is the speaker now, and the interpretation of $y$ is the hearer now.

The corresponding intensions, however, are separately defined: as $s^{A,i,j,s} = F(i)$ and $h^{A,i,j,s} = F(y)$ respectively. One observes that no longer holds that for all $\alpha$: $(s^{A,i,j,s}) \alpha^{A,i,j,s} = \alpha^{A,i,j,s}$. This combination of the definition of interpretation of $s$ and $h$ with the interpretation of $i$ and $y$ violates the recursive interpretation of the IL, thus
disturbing its homomorphic interpretation. This has drastic consequences: several tautologies become unvalid. It is no longer true that for constants of type \( \langle s, e \rangle \) holds \( \forall c \in c \), nor that \( \alpha = \beta \Rightarrow \alpha = \beta \) is valid (for \( \alpha, \beta \in \mathcal{M}_s \)). The interpretation of the logic is not a homomorphism (since \( \mathcal{h}(\alpha) \neq \lambda i, j[h(a)] \)); therefore the interpretation of the natural language is not a homomorphism either. This means that the principle of compositionality is violated.

Let us consider the first goal of the approach of Groenendijk & Stokhof. Sentence (15) is true when evaluated with respect to the moment 'now', but not with respect to a point of reference where the speaker is someone else. The sentence expresses not a tautology (as a matter of fact, this is not claimed by Groenendijk & Stokhof). What they probably wish to express by the phrase 'tautological character' is that for every choice of the moment 'now', the sentence is true when evaluated with respect to this moment, whereas not the same can be said about the second sentence. This effect can be obtained in a compositional way by stipulating that \( F(i) = \lambda i, j[s(i, j)] \) and \( F(y) = \lambda i, j[h(i, j)] \). Then the translation of (16), being something like \( \forall i = s \), becomes true for the point of reference \( (i, j_0) \), no matter what \( j_0 \) is, whereas this is not the case for the translation of (16), being something like \( \forall w[i = s] \).

6. COUNTING ELEMENTS

6.1. Introduction

In the present section I will consider two examples of counting the number of elements in the model. In the first example this is done in a way which suggests a misunderstanding of the framework. As a contrast I present the second example in which the counting proceeds correctly. These examples illustrate the role of the derived algebra \( \mathcal{M} \) which is obtained from the algebra \( \mathcal{M} \) in which we interpret intensional logic.

6.2. Keenan & Faltz count

KEENAN & FALTZ (1978) present a system for the description of syntax and semantics that is related to Montague's system. An important difference is that they obtain their semantic domain by application of algebraic operations (join, meet, homomorphism) on certain basic elements. One way in which they compare their system with Montague's is by ways of
counting the number of elements in the semantic domain of their system and Montague's sets $D_t$. They base an argument in favour of their system on the fact that a certain domain $D_t$ contains many more elements than the corresponding set in their own system. There are several objections against this comparison. The stage at which Keenan & Faltz carry out their comparison (viz. p. 130) does not do justice to Montague's enterprise. They compare their model for an extensional fragment of English with Montague's domains developed for an intensional fragment. Furthermore they do not take Montague's meaning postulates into account. So the numbers they obtain are not the relevant numbers. I will, however, not correct their calculations, since I am primarily interested in the method of comparison. This method will be discussed below.

Keenan & Faltz have a theory which says e.g. how many verb phrase meanings are possible (for a given domain of individuals): it is the number of homomorphisms between certain sets (which are built from the set of individuals). Keenan and Faltz count in their framework the number of elements in some of such sets, i.e. they count the number of possible denotations of certain types. In Montague's framework they count the number of elements in $D_t$ for the corresponding types. I have fundamental objections against this comparison since in this way sets are compared that are incomparable. The sets $D_t$ in Montague's system are sets in the algebra $M$ (see figure 1). Out of algebra $M$ a derived algebra is defined. This derived algebra $M'$ consists precisely of the elements which are used for the interpretations of expressions produced by the grammar for the fragment. In the process of forming the derived algebra $M'$ elements of $D_t$ may be thrown out; e.g. a set $D_t$ may consist of all functions of a certain type, whereas in $M'$ only the homomorphisms may be left over. If one wants to count the number of possible denotations for the expressions of a certain category, then one has to count the number of elements of the corresponding type in the union of all derived algebras. One should not count the auxiliary set $D_t$ instead. The method of counting of Keenan & Faltz neglects the role of arrow 6 in figure 1.

The number of elements in a derived algebra can easily be counted. The derived algebra $M'$ is the image of the syntactic algebra for the fragment. Therefore the number of elements in $M'$ cannot be larger than the number of expressions in the syntax. Since the latter is denumerable, the former is. And for a given category, say the verb phrases, the number of expressions
of this category gives an upper bound for the number of elements of the corresponding type in the semantic algebra.

6.3. Partee counts

As a contrast to the previous example, I would like to consider Partee (1977b), where the difference between M and M' is taken into account. Partee discusses the psychological plausibility of possible world semantics. She argues that the finiteness of our brains requires that the theory of linguistic information we have should be finitely representable. The possible world semantics, however, gives rise to sets of rather large cardinalities (for instance if |A| = N_0 and |I| = N_0, then |D_{s,e}| = 2^{N_0} and |D_{s,e,s,e}| = 2^{N_0}). These cardinalities make it impossible to assume that we have finite representations of all sets D_τ in our brains. Partee gives a way out of this dilemma: assume that we have finite representations of the form of the sets D_τ, but not of all their elements. Partee (1977b, pp. 317-318) says: 'In the acquisition of any particular language, not all of the in-principle possible denotations need to be considered as a hypothesis about the actual denotation of an actual expression of the language. The intentional logic into which the natural language is translated, will contain at most denumerable many expressions of any given type, and the finite perceptual and cognitive apparatus will make at most denumerable many members of D_a,A,I,J finitely representable, and it will only be correspondences between these two at most denumerable sets that will have to be empirically determined by the language learner'.

It is striking to compare these psychologically motivated opinions with the mathematical properties of the framework. These predict that in an interpretation of a particular language there are only denumerable many meanings because there are denumerable many expressions in the language of which we give the meaning. So for any particular language the number of meanings in the model, and the number found on psychological considerations agree.

In relation with the previous discussion Partee considers the following question (Partee 1977b, p. 318). 'One might ask at this point, if the 'available' members of D_a,A,I,J always going to form a denumerable set, why shouldn't all the sets D_a,A,I,J be constrained to be denumerable by the semantic theory?' The answer Partee would argue for is 'that there is no telling in advance which possible world the native speaker will find her-
self in; [...] her semantic component must equip her for a language in any of them.

An alternative is to consider the psychological arguments as an invitation to change one of the algebras of the framework in figure 1. It seems to be an argument against taking the standard model for intensional logic because of the large cardinality of its sets, in favor of taking as semantic algebra some generalized model with denumerable many elements (generalized model in the sense of HENKIN 1950, see sections 1 and 3 of ch.3). This would have the interesting consequence that the axiomatization of intensional logic (given in chapter 3) would be a complete axiomatization for this class of models. This is a direct consequence of lemma 3.3.1 in GALLIN 1975. But no matter which conclusion is drawn from the psychological arguments, this whole discussion remains an interesting excursion because Montague's system was not designed, as I explained in chapter 1 and 3, to formalize any psychological theory.

7. THE TRANSLATION LANGUAGE

7.1. Introduction

An intensional language is a language of which the denotation of an expression depends on an index, and an extensional language is a language where this is not the case. An example of an extensional language is predicate logic, an example of an intensional language is IL. Our approach makes English an intensional language: a sentence denotes a truth value; which one this is depends on the current index (point of reference). In an extensional approach we would say that a sentence denotes an function from indices to truth values (an intension). Is it possible to give an extensional treatment of English, and what are the consequences? In other words, is it possible to change the relation indicated in figure 1 by arrow 7?

It will turn out that the answer to the above question is positive. This gives us the choice between (at least) two different approaches. When making a choice, we have to realize what the role is of the translation level in the whole framework. We aim at defining a systematic relation between English expressions and their meanings. In order to be able to express this relation conveniently, we use the translation into intensional logic. In chapter 2 is explained how this logic is interpreted.
homomorphically: e.g. an expression of type $t \wedge s$ as its homomorphic
image (has as its meaning) some function in $\{0,1\}^{T \times J \times G}$. Fundamental to this
whole approach is the relation between expressions and meanings. If a trans-
lation into an extensional language gives rise to the same relation, it is
acceptable as well. Another translation is just another tool, and a choice
has to be made on the basis of technical arguments.

7.2. Hauser translates

As an introductory step of treating English as an extensional language,
I consider an approach which is very close to the PTQ translation: translate
into an IL expression denoting the meaning of that expression. Let the int-
ensional ed translations of two sentences be $A$ and $B$ respectively. Then
the translation of their conjunction has to be $^A(\forall^A \forall^B)$. Most of the trans-
lation rules have the format $\alpha(\forall^B)$. With the new translations this becomes
$^A([\forall^A](\forall^B))$. These examples illustrate that this approach does give rise to
somewhat more complex formulas. A next step is to use a logical language in
which the operators on elements of type $<s,\tau>$ are defined. For instance $\wedge$
where $A \wedge B$ is interpreted as the complex conjunction formula given above,
so as indexwise evaluation of the parts of the conjunction. For function
application is used $\operatorname{A}(\operatorname{B})$; to be interpreted as denoting the same as
$^A([\forall^A](\forall^B))$. Such operators are used in JANSS & VAN EMDEN BOAS 1977 for
dealing with semantics of programming languages.

Following TICHY 1971, HAUSER (1980) argues for an extensional approach
to English. In HAUSER (1979a, 1984) this idea is worked out. He does not
use the standard logical operators (e.g. conjunction on truth values) any
more, and this gives him the opportunity to use the standard symbols with
a new meaning. Now $\phi \wedge \psi$ means indexwise valuation of the parts of the con-
junction, and $\alpha(\beta)$ is the variant of function application which is described
above as $\alpha(\beta)$. In this way one obtains a simplification of the formulation
of the translation rules since no intension symbols have to be used. The
price one has to pay for this, is that the logical symbols obtain a some-
what deviant interpretation, what is the normal price, and what is quite
acceptable. But the presentation of the translation rules is not the only
aspect of a new translation. What happens if one wishes to take the meaning
postulates into account, or if one wishes to simplify the formulas one ob-
tains? To understand the dangers, one should realize that the new trans-
lation causes the intension operators to be invisible whereas semantically
they still are there. Therefore new reduction rules have to be found. In
HAUSSER (1979b, 1984) indeed a simple translation is obtained but not all
meaning postulates are expressed in the translation. Therefore his results
are not convincing, and further investigations are required before it is
clear whether this extensional approach gives a simplification.

7.3. Lewis translates

A simplification I expect from the approach in LEWIS 1974. He discusses
the consequences of another kind of extensionalized translation. He con-
siders using the extra possibilities given by an extensionalized translation:
namely the possibility to relate to an expression a meaning that is not an
intension (i.e. not a function in D\times\text{JxG}). The verb run gets in the PTQ ap-
proach a translation of type \langle<s,e>,t\rangle. In the Hauser approach it is trans-
lated into an expression of type \langle<s,\langle<s,e>,t\rangle\rangle. But in the Lewis approach its
translation would be of type \langle<s,e>,\langle<s,t\rangle\rangle. So the translation of run would be
a function which assigns to an individual concept a proposition. In this
way one gets rid of the remarkable non-constant interpretation of constants
of IL, where run_{\langle i,j \rangle}; s = F(\text{run})(i,j). In Lewis approach it would be just
run_{\langle i,j \rangle}; s = F(\text{run}). The translation of \text{He}_f runo would be run(x_1) being an
expression of type \langle<s,t\rangle. Note that here the function application has its
standard meaning. This illustrates the advantage of Lewis approach, but
further investigations are required in order to decide whether it is a real
simplification. A remarkable aspect of Lewis approach is that it gives a
completely different relation between expressions and meanings than we
considered up till now, so investigating these matters here, might bring
us far from the current work (cf. HAUSSER 1984, p. 82, 83).

7.4. Groenendijk & Stokhof translate

A last version of what might be called an extensionalized translation is
used in GROENENDIJK & STOKHOF 1981. They do not translate into IL, but
into Ty2 (see GALLIN 1975). Such a translation can be called extensional
since the interpretation of a Ty2-expression does not depend on an index,
but only on the variable assignment (including the assignment to 'index'
variables). Also in this translation we get rid of the non-constant inter-
pretation of constants since the index dependency of predicates as run is
made explicit by translating run into an expression containing an index-
variable. The phenomena described by Groenendijk & Stokhof seem to require
the expressive power of Ty2, and it is to be expected that this power will be required for the treatment of other phenomena as well (e.g. VAN BENNEM (1977) argues that explicit reference to moments of time is needed for tense phenomena). Since we will not consider these phenomena, we will not investigate the details of such a translation. From the way in which we introduced IL in chapter 3 (using a translation into Ty2), it is evident that this would cause no problems at all (on the contrary, several aspects would become simplified).

7.5. Keenan & Faltz on translations

In KEENAN & FALTZ 1978, several requirements are given concerning the logical form of a natural language, e.g. criteria concerning the correspondence between a natural language expression and its logical form. They criticize the logical form which is obtained in a Montague Grammar. An example is their comment on the translation of John which is in an extensional fragment λP[P(j)]. They say (p.18) '... this assignment of logical structure fails the Constituent Correspondence Criterion, since it contains three logical elements, namely j, P and λP, none of which corresponds to a constituent of John.' Such criticism plays an important role in the argumentation in favour of their framework.

The argumentation of Keenan & Faltz is, however, based upon a misconception of the framework. I assume that they understand by 'logical form', that level of description at which the meaning of an expression is completely determined. In fact, there is no unique level of description in Montague Grammar for which this holds. The analysis tree of an expression, its immediate, unreduced translation, its reduced translation (and all the stages in between), all determine the meaning of that expression completely. That, in particular, the translation of an expression into IL cannot be claimed to have a special status as the logical structure of that expression, becomes clear if one realizes that this level of representation, in principle, can be dispensed with altogether. Grammar provides a correlation between syntactic structures and meanings. In Montague Grammar this is done by providing a homomorphism from the set of syntactic structures into the set of abstract settheoretical entities, modelling the meanings. In the PTQ-system this homomorphism is defined in two steps. First a homomorphic translation from syntactic structures into logical expressions is provided, second the logical expressions are interpreted, i.e. related in the usual homomorphic
way to the abstract entities defined in the model. These two homomorphisms together determine one homomorphism from the syntactic structures into the meanings, viz. the composition of the two. This two-step approach is chosen for reasons of convenience only, it is not necessary. As a matter of fact, the EFL-system (MONTAGUE 1970a) is an example of a system in which the homomorphism from syntactic structures into abstract meanings is defined in one fell swoop, without an intermediate stage of translation into a logical language. All this means that within the PTQ-framework it is not possible to talk of the logical structure, or the logical form, of an expression. So Keenan & Faltz criticize a non-existing aspect of Montague grammar.