1. Syntax

In the present study, syntax will only play the marginal rôle of a background which has to be assumed in order to discuss questions of semantics. Consequently, the following introduction and discussion of the most central notions of Montague's general theory of syntax will be very brief.

According to Montague, syntax has two parts, *deep* and *surface syntax*, and it is the former one which is of primary interest to semantics. For syntactic deep structures will serve as the input to the rules of semantics. This holds for both artificial languages of formal logic and natural languages such as English or fragments thereof. There is, however, a principal difference between these two types of languages: whereas the deep structures of, e.g., modal predicate logic can be thought of as identical to the corresponding surface structures (i.e. the formulae and terms of that language), any natural language allows for too many syntactic ambiguities to be interpreted in such a direct way, and hence there will usually be a long way from depth to surface.

The *deep syntax* of a given language is a system of rules each of which has the following form:

(1.1) If $X_1,...,X_n$ are deep structures of the corresponding syntactic categories $\kappa_1,...,\kappa_n$, then the result of applying the (n-place) syntactic operation F to $<X_1,...,X_n>$ will be a deep structure of category κ_{n+1}.

In the case of a natural language, the *categories* mentioned in (1.1) are the traditional ones like 'sentence', 'noun phrase', etc. In predicate logic one would instead have 'individual variable', 'formula', 'quantifier', and the like. Similarly, the operation F could, e.g., be something like relative clause formation (out of a sentential deep structure and a relative pronoun), in which case (1.1) would be the relative clause rule; or (1.1) might be the rule of quantification which says that a quantifier combines with a variable and a formula (or 'sentence') in order to produce a new formula (or 'sentence') of predicate logic. Since the categories and operations are the only variable items in the above scheme, a syntactic rule might as well be construed as a sequence $<F, \kappa_1,..., \kappa_n, , \kappa_{n+1}>$, which is (more or less) Montague's format of a rule of

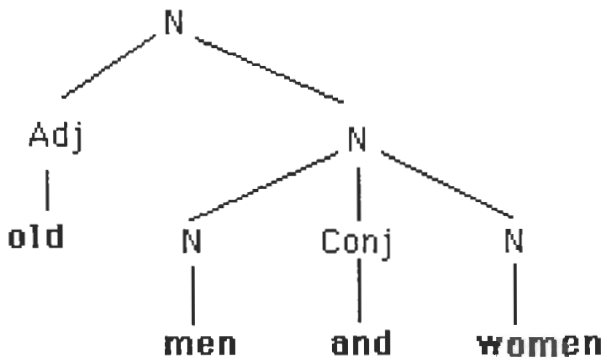
(deep) syntax. Then the information that such a sequence must be interpreted in the sense of (1.1), of course, belongs to the general theory of syntax.

Deep structures can only serve as the input to semantics if they are in some sense unambiguous, i.e. if it is possible to tell from a given deep structure in what way it has been built up. This is the essential difference between a structurally ambiguous expression like

(1.2) **old men and women**

and a simple phrase structure tree or an equivalent labelled bracketing structure:

(1.3)



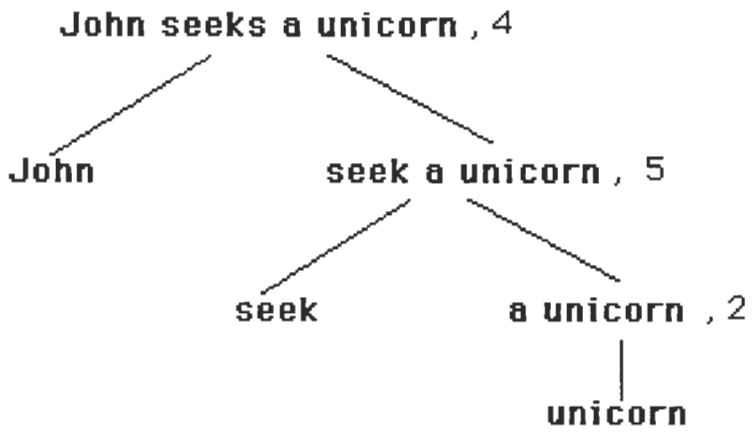
(1.4) $(_N\ (_{Adj}$ **old** $)_{Adj}\ (_N\ (_N$ **men** $)_N\ (_{Conj}$ **and** $)_{Conj}\ (_N$ **women** $)_N\ )_N\ )_N$

Structures like (1.3) or (1.4) do, of course, not provide *all* the information about how they have been construed, because they are not equivalent to their corresponding derivations. On the other hand, it seems to be intuitively clear that the question of whether (1.2) has been built up by first developping the embedded N's and then inserting the lexical item **old**, or whether this has been done the other way round, is immaterial to the interpretation of the structure: although neither (1.3) nor (1.4) reveals the complete history of how the terminal string **old men and women** has been derived (from some given phrase structure grammar), they both do say which parts of the sentence have been concatenated with which, and they also say by which rules this has been done: the first bit of information is given by the 'naked' tree structure (or the bracketing), the rest can be gathered from the category labels. If one thinks of deep syntax as a device for building up structrures like (1.3) or (1.4), a rewrite rule like $N \rightarrow Adj\ \hat{}\ N$ (with '$\hat{}$' for concatenation) would have to be interpreted

as an instruction to surround the result of concatenating some structures, Adj and N, by a pair of brackets labelled with 'N', i.e. as an operation which yields the result '$(_N X \hat{} Y)_N$' whenever it is applied to a structure X of the form '$(_{Adj} \underline{X})_{Adj}$' and some Y of the form '$(_N \underline{Y})$'. (I use *Quinean corners* , $\ulcorner\ \urcorner$ and $\ulcorner\urcorner$ whenever linguistic levels get mixed up; $\ulcorner(S_1$ **and** $S_2)\urcorner$ reads: 'the string that begins with "(", whose second element is S_1' etc.) The resulting structure will be unambiguous in the sense that the operation which has been aplied to produce it can be read off from the indices of the brackets surrounding it: since this applies to all sub-structures as well - they, too, have been built up by corresponding operations - the whole structure will carry the information of how it has ben built up from its ultimate constituents. This is, in fact, Montague's only requirement on the operations of deep syntax: whenever a structure is complex, there is only one way of getting it by the syntactic operations or, more precisely,: whenever a structure X can be built up by either applying some (n -place) operation F to $<Y_1,...,Y_n>$ or by applying an m -place G to $<Z_1,...,Z_m>$, then not only the results are identical but the inputs must have been identical, too, i.e. $F = G$, and $<Y_1,...,Y_n> = <Z_1,...,Z_m>$ (and hence $m = n$, $Y_1 = Z_1$, etc.). In this sense Montague's general concept of deep syntax can be regarded as a drastic generalization of the idea of phrase structure. In practice, Montagovian deep structures may, however, look somewhat different from (1.3) and (1.4). Here are two typical examples:

(1.5) {Jones { seeks a {horse such v_7 that
{ it v_7 speaks }}}}

(1.6)



The dashed brackets in (1.5) roughly serve the same purpose as the labelled brackets in (1.4). The superficially invisible variable v_7 is an additional disambiguating device. (1.6) has been taken from a different sample syntax ('PTQ'). This

sort of structure is sometimes referred to as a *Montague tree* . It shows how the sentence at the top node has been gradually built up from less complex expressions. The numbers that occur at some of the nodes indicate which syntactic operations have been applied.

The schematic letter '*F*' in (1.1) stands for an operation, i.e. a function in the set-theoretic sense of the term. The collection of all deep structures of a language thus constitutes - together with the syntactic operations - a *partial algebra* $\mathcal{A}$, i.e. a non-empty set (or *domain*) *A* together with some functions each of which is defined on a subset of *A* and has its values in *A* . I will follow Montague in assuming that all operations are *total*, i.e. defined on *A* itself. This has the possibly unintuitive consequence that, e.g., relative clause formation will have to be defined on pairs of structures none of which corresponds to a relative pronoun or a sentence. In practice, this means that one must sometimes be very careful in defining the results of syntactic operations - even in cases where they do not get assigned a syntactic category by any of the rules. But, as I said, syntax is not my business and in what follows these problems will not arise.

The central part of deep syntax, then, is an *algebra* (= a partial algebra with total functions only), and I will refer to this algebra as the *syntactic algebra* of whatever language will be discussed. A tacit assumption in the above discussion was that all complex deep structures can be built up from the lexical items with the help of the syntactic operations. In view of the unambiguity of syntactic structures this assumption has the consequence the syntactic algebras will be of a very special kind: they will be so-called *free* algebras. Roughly speaking, the elements of a free algebra are either basic, i.e. they cannot be obtained by applying any of the operations, or complex, in which case the way they can be obtained is uniquely determined. This fact implies that the whole algebra is (in an obvious sense) *isomorphic* to the 'language of names' of its elements: for any free algebra $\mathcal{A}$ there is a 'structure-preserving' one-one correspondence between $\mathcal{A}$ and the system $\overline{\mathcal{A}}$ that contains exactly one name for each basic element of $\mathcal{A}$ and one symbol $\overline{F}$ for each operation F of $\mathcal{A}$, and is closed under *term formation* (i.e. the sequence $<\overline{F}, x_1,...,x_n>$ is in $\mathcal{A}$'s domain whenever F is an n-place operation and $x_1,...,x_n$ are already in there.) $\overline{\mathcal{A}}$ itself can, of course, be conceived of as an algebra, the *word algebra* of A , and it is fairly easy to see that free algebras are exactly those which are isomorphic to word algebras. One can thus think of the syntactic deep structures as ('polynomial') terms in the indicated sense.

The basic elements of a syntactic algebra intuitively correspond to lexical items, morphemes, or whatever one takes to be the ultimate building blocks of syntax. The set of all basic elements of a syntactic algebra will be referred to as the *lexicon* of that (deep) syntax. In order to make sense of the syntactic rules, the lexicon must have some structure, i.e. its elements must be syntactically categorized. Properly speaking, then, the lexicon is not just a set but rather a family $(X_\kappa)_{\kappa \in K}$ of sets, where K is the set of syntactic categories and for any $\kappa \in K$, X_κ is a subset of the domain of the syntactic algebra. One should note that this does not exclude some possible overlap between the sets of lexical items of various categories; it is also possible that some (or even all) X_κ are empty. The only restriction on the lexicon $(X_\kappa)_{\kappa \in K}$ will be that its union, i.e. the set of all elements that are in some X_κ, constitutes the set of basic elements of the syntactic algebra. (In case all X_κ are empty, the algebra will have to contain zero-place functions which 'create' the basic elements; I will, however, ignore this pathological case.) In what follows the term 'lexicon' will sometimes refer to the family $(X_\kappa)_{\kappa \in K}$, sometimes it will refer to its union, and sometimes I will be a little bit vague about which of the two I mean. I am positive that no confusion will arise out of this terminological sloppiness.

The final ingredient of deep syntax is a distinguished category, *viz.* that of *sentences*. (This will, by the way, guarantee that the set of categories can never be empty.) The reason for giving sentences a special status is not so much that they are thought of as the primary objects of testing syntactic theories - as seems implicit in those accounts of syntax according to which languages are defined as sets of sentences. In the present study and, indeed, in the whole tradition of Montague grammar, sentences rather play a central rôle in defining semantic structures and in evaluating semantic hypotheses.

To sum up: deep syntax consists of a syntactic algebra the elements of which are structurally unambiguous and are built by successively applying syntactic operations to lexical items. The lexicon itself is divided into several sub-sections corresponding to the syntactic categories. Moreover, there is a set of syntactic rules each of which provides information about the syntactic categorization of complex expressions. Finally, one catrregory is singled out as the category of sentences.

As I have already said, deep syntax is the only part of syntax that is of any direct relevance to semantics. It is therefore not too surprising that Montague, whose prime

interest lay in semantics, did not bother about the general theory of surface syntax. His theory, in fact, only says that deep structures have to be somehow related to actual expresssions: there must be some relation R the domain of which is the set of deep structures or a subset thereof. The image of R can then be thought of as the set of expressions of the language under description. (The *domain* of a binary relation R is the set of x such that $x\,R\,y$, or $\langle x, y \rangle \in R$, for some y; the *image* of R is the set of y such that xRy for some x.) Part of the structure of deep syntax then naturally carries over (via R) to R's image, but I will not go into these matters here, since they are completely straightforward and not too exciting anyway.

In the above sketch of Montague's general syntactic theory I have deliberately ignored certain formal details and complications which are not of any interest to my present concerns. In particular, I did not mention what the range of the variable 'n' in the locution 'n-place function' is, thus suggesting that n will always be a natural number. In Montague's original formulation, syntactic operations may even take infinite sequences of arguments, so that the theory even covers certain *infinitary* languages, i.e. logical languages that allow for conjunctions, disjunctions, etc. of infinitely many formulae. Such languages will, however, not play any rôle in the investigations to follow; moreover, it wil sometimes be much more convenient to concentrate on finitary languages. I will thus assume that the number of places of any syntactic operation is finite.

Although my sketch of Montague's general theory of syntax has not followed the original formulation in each and every detail, I have refrained from suggesting any substantial changes. The main reason for this is my own laziness. Since syntax is not what is at stake, almost any general syntactic theory could serve as a point of departure, the only restriction being that the theory contains some 'plug' that makes it possible to link it up to the general semantic theory to be discussed in the following chapter. Before I turn to this, I will, however, briefly comment on two obvious weaknesses of Montague's theory. (i) The first is that the idea of letting the syntactic algebra be a total one is not only fairly unintuitive but in practice sometimes leads to problems that could be easily avoided by a slight complication in the general theory. What seems to be much more attractive than the above notion of a syntactic algebra is some concept of a partial and sorted algebra that incorporates the rules of syntax. Such notions have actually been defined and used by various authors. - (ii) The second obvious weakness is the vague characterization of the 'deep/surface relation'. At least two sugestions to further specify this relation have been made. (ii-a) One is

that surface syntax should itself be an algebra and that the algebra of deep syntax is just its word algebra. The two are then naturally related by a (certain) *homomorphism*, a notion to be introduced in the next chapter. This conception of the deep/surface relation does not bear any consequences on the question of what surface syntax might look like - as long as it can be formulated as some kind of algebra - and it is thus in some sense vacuous. But it has the clear advantage of re-establishing the natural order of research: surface syntax will be formulated according to actual observations made by linguists whereas deep syntax is just some 'super-structure' which can be imposed on it once one wants to cross the boundary from syntax to semantics. - (ii-b) Another, presumably more substantial (if less precise) idea of specifying the deep/surface relation is Barbara Partee's *well-formedness constraint* . In essence, the constraint says that deep structures must be expressions of the actual language, possibly enriched by a few disambiguating symbols such as brackets or subscrispts. According to this constraint, the relation between deep and surface syntax should thus be very simple, like that between a context-free language and a corresponding set of phrase structures.

Sources

The primary source is section 2 of Montague (1970b). The best introductory text that I know is the third chapter of Löbner (1976). The precise formal features of the theory do not seem to be very well investigated; some results can, however, be found in Mönnich (1975), chapter II.1, and Engesser (1980), part II.

Example (1.2) is due to Chomsky (1957), p. 87; so is, of course, the whole 'deep structure = phrase structure' approach. The observation that Montague's conception of syntax is a sort of generalization of that approach has been made independently by various authors including Egli (1974), chapter II. A precise study of tree structures in Montagovian syntax is chapter VII in Janssen (1983). The notation "$\ulcorner$" and "$\urcorner$" for mixed quotations (so-called 'quasi-quotations') is due to Quine (1937), 146; see Quine (1951), §6, for an exposition, and Kaplan (1969), 214, for a nice historical explanation of that notation.

(1.5) is a structure derivable from the fragmentary English syntax in Montague (1970b), section 7. (1.6) is from Montague (1973), p. 228. The term 'Montague tree' is due to Partee (1973a). More on free algebras, word algebras etc. can be found in any textbook on universal algebra; see, e.g., Cohn (1981), chapter III. A reformulation of Montague's syntactic theory with (i) partial operations on a sorted domain and (ii-a) deep syntax as the word algebra of surface syntax has been given in Janssen (1983), chapter II; something similar to (i) - if not quite as elegant - can be found in Hinckley (1977), chapter 3. The well-formedness constraint is due to Partee (1979).

2. Compositionality

I will now start to give a detailed account of the semantic theory that will later serve as my frame of reference, *viz.* Montague semantics. The present chapter will introduce part of the abstract overall structure of the 'semantic component' of a grammar, i.e. of that part which assigns meanings to deep structures. The question of what kind of things these meanings are will, however, be left open for the moment.

The most fundamental principle underlying Montague's general theory of semantics is the principle of *compositionality* which is sometimes also referred to as *Frege's principle* and which can roughly be stated as follows:

(2.1) The meaning of a complex expression can be determined from the meanings of its parts.

(2.1) is, of course, very vague and, for the most part, the present chapter will consist in a discussion of one idea of making (2.1) more precise. It should, however, first be noted that (2.1) indeed expresses a straightforward (and maybe even obvious) idea about the semantics of languages with infinitely many expressions. One only needs to consider a recursive syntactic rule like (2.2), which allows for the construction of a sentence S_1 from two given sentences S_2 and S_3:

(2.2) If S_1 is a sentence (of some fixed language), then so is: $\ulcorner(S_1$ **and** $S_2)\urcorner$.

(2.2) might be a syntactic rule of a deep syntax formulated according to the general pattern of the preceding chapter. And (2.2) might indeed be a consquence (a so-called *derived rule*) of a syntactic description of the English language. If that syntax defines any sentences at all, then (2.2) obviously allows for the derivation of infinitely many sentences. So whatever semantics might look like, it is clear that it cannot be defined by simply enumerating all expressions or deep structures plus their respective meanings. (At least this cannot be done in both a precise manner and finitely many

steps.) So one will have to employ some kind of recursive mechanism in semantics as in syntax. But then a general strategy of how to find such a recursive mechanism almost suggests itself: just as any application of (2.2) presupposes that one has already construed some sentences,S_1 and S_2, before one gets another one, S_3(= ⌜(S_1 **and** S_2)⌝), so a corresponding 'semantic rule' for determining the meaning of S_3 might well presuppose that the meanings of S_1 and S_2 are already known. This strategy certainly has some intuitive appeal. For how is it possible for speakers of English to understand a sentence like **My neighbour snores and I don't like that**? It seems that their knowledge of what **My neighbour snores**, **and**, and **I don't like that** mean is indeed of some help here.

The strategy of doing semantics along the lines of syntax is obviously one way of interpreting (2.1). In the case of (2.2), the immediate parts of a sentence of the form ⌜(S_1 **and** S_2)⌝ are S_1, S_2, the word **and**, and the disambguating brackets (if they are needed). If one thinks of the resulting expression as a deep structure of a given syntax, then a more precise formulation of (2.2) would have to involve some syntactic operation F which can be defined by:

(2.3) $F(X,Y) =$ ⌜(X **and** Y)⌝,
whenever X and Y are deep structures.

(It should be noted that F can also be used to conjoin expressions that are not sentences, so that the same operation would occur in more than one rule.) From (2.3) one sees that the parts of complex expressions built up by F are of two kinds: they are either the argumets of the operation, i.e. X and Y , or part of the material introduced by F, i.e. the word **and** or the brackets. The resulting structure is, of course, uniquely determined by F and its arguments; this is a consequence of the unambiguous character of deep structures. So, from an algebraic point of view on syntax, the immediate parts or constituents of a complex expression $F(X_1,...,X_n)$ can be identified with $X_1,....,X_n$, and F (or some name or index of F). However, since $X_1,...,X_n$ also satisfy the condition of unambiguity and thus 'encode' all of their immediate parts which themselves can be identified with their immediate parts etc., one can indeed think of the collection of F and its sequence of arguments as encoding all parts of the complex expression $F(X_1,...,X_n)$. (2.1) thus comes down to:

(2.4) If $F(X_1,...,X_n)$ is a deep structure, then its meaning is uniquely determined by the meanings of $X_1,...,X_n$, and F.

(2.4) refers to the possibly obscure notion of 'meaning of a syntactic operation F'. This can, however, be avoided by considering the following simple sonsequence of (2.4):

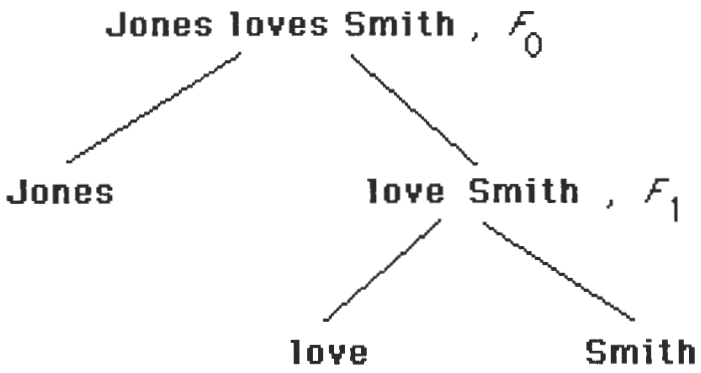
(2.5) For each (n-place) syntactic operation F there is a corresponding (n-place) 'semantic operation' G such that the meaning of any deep structure $F(X_1,...,X_n)$ is $G(b_1,...,b_n)$, where b_1 is X_1's meaning, etc.

If one writes b(X) for the meaning of a deep structure X, then the essential part of (2.4) can be replaced by the more suggestive formula:

(2.6) $b(F(X_1,...,X_n)) = G(b(X_1),...,b(X_n))$

(2.5) says that the meanings of deep structures form an algebra B which is *similar* to the syntactic algebra (i.e. it has the same number of operations with the same numbers of arguments) and that the notion b of meaning is a *homomorphism* from the syntactic algebra to B, i.e. that it satisfies (2.6). Intuitively, a homomorphism is a function (from one algebra to a similar one) which preserves part - but not necessarily all - of the structure defined by the algebraic operations. In the case of syntactic algebras one can visualize the structure preserving effect of meaning homomorphisms if one represents deep structures by Montague trees:

(2.7)

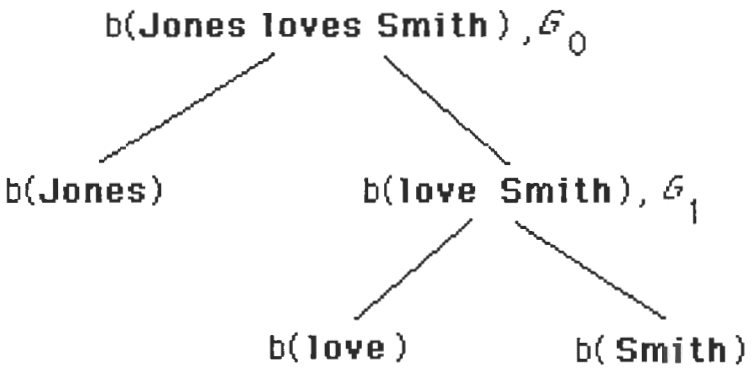


The fact that the meaning assignment function b is a homomorphism implies that the meaning of (2.7) is built in analogy to that deep structure itself, i.e. it has the form:

(2.8) G_0(b(**Jones**),G_1(b(**love**),b(**Smith**))),

where G_0 and G_1 are the semantic operations correspnding to F_0 and F_1, respectively. If one takes a surface-oriented point of view and identifies deep structures with their top nodes - strictly speaking, one may do so only if these top nodes (or, more precisely: their labels) are themselves unambiguous - then (2.8) can be rewritten as the tree:

(2.9)

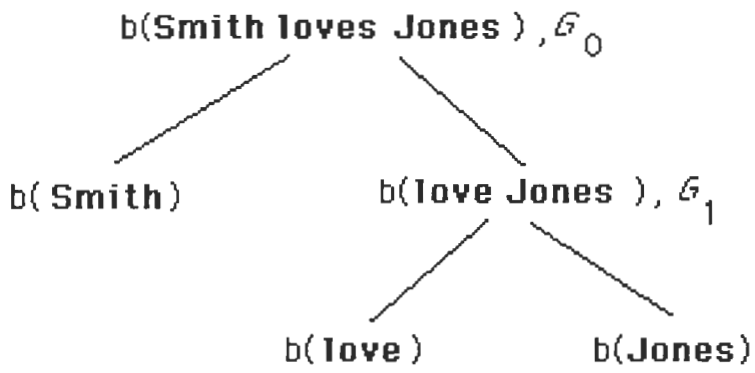


In (2.9) all labels of non-terminal nodes v have the form <b(α),G_n> and can be read: 'b(α) is the result of applying G_n to (the left parts of) the labels of the nodes immediately dominated by v'. (I have, of course, omitted the pair brackets in my

representation of trees.) Then, clearly, (2.9) expresses that (2.8) is the meaning of (2.7). The close analogy betwen (2.7) and (2.9) is quite in the spirit of the compositionality principle (2.1). It must, however, be pointed out that (2.8) is an element of the semantic algebra B whilst (2.9) belongs to the algebra of semantic trees, which can be shown to be isomorphic to B's word algebra and thus (in general) more structured than B. The latter fact easily follows from the observation that two semantic trees correspond to the same meaning *b* whenever *b* occurs in the labels of both top nodes. On the other hand, the algebra of *semantic trees* will in general be less structured than the syntactic algebra. The following argument shows why.

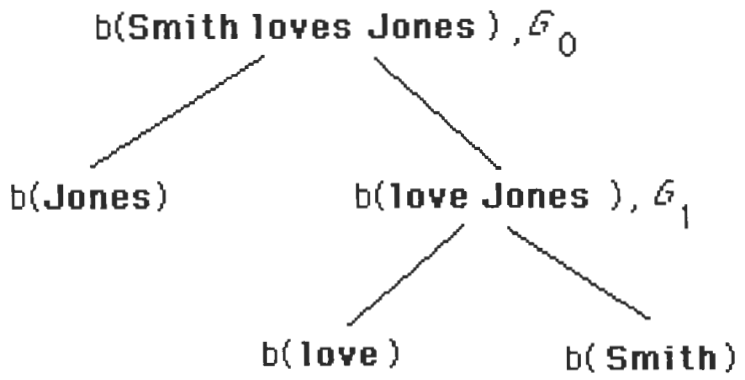
For the sake of this argument one has to imagine that the names **Smith** and **Jones** were *synonymous* (*according to* B), i.e. that b(**Smith**) = b(**Jones**). One should then consider the semantic tree corresponding to the deep structure of **Smith loves Jones**:

(2.10)



Since, by hypothesis, b(**Smith**) and b(**Jones**) are the same object *b*, the two (metalinguistic) names of *b* can be interchanged throughout (2.10). One thus obtains:

(2.11)



In (2.11) the node corresponding to the verb phrase is labeled by an ordered pair consisting of the meaning of **love Jones** (or the deep structure underlying that expression) and the semantic operation G_1. But b(**love Jones**) = G_1(b(**love**),b(**Smith**)), by the above interpretation convention for semantic trees. So b(**love Jones**) = b(**love Smith**), by (2.9), and hence the label on the 'VP node' in (2.11) is: <b(**loves Smith**),G_1>, i.e. the same as in (2.9). A similar argument can be applied to the top node of (2.9) and (2.11). The inevitable conlusion, then, is that (2.9) and (2.11), and thus (2.9) and (2.10), are identical! On the other hand, **Jones** ≠ **Smith**, i.e. the two names are clearly distinct, and hence so are (2.7) and the deep structure of **Smith loves Jones**. The corresponding semantic trees are, however, identical. One should, of course, keep in mind that this conclusion has only been arrived at on the assumption that **Smith** and **Jones** are synonymous, which is clearly not the case. But *if* there are any non-trivial synonymies at all (i.e. synonymies between distinct structures), then something like the above argument could be produced in order to show that distinct deep structures may correspond to a unique semantic tree.

So the algebra of semantic trees in general possesses 'less structure' than the syntactic algebra. *A fortiori* this means that the *semantic algebra* ,i.e. the domain of meanings plus the semantic operations, is also less structured than deep syntax. As has been pointed out above, it will, in fact, usually be even much 'poorer' than the algebra of semantic trees.

Semantic trees are not really part of Montague's core theory, which is only concerned with the relation between syntactic algebras and their corresponding semantic

algebras. I have introduced trees here mainly because they nicely illustrate the compositionality principle, but also because they have been discussed in the literature. Moreover, as a simple argument from universal algebra would show, semantic tree algebras are themselves semantic algebras.

In the above examples I have made no assumptions about the meanings of the lexical items involved; for the principle of compositionality only concerns the meanings of complex expressions. On the other hand, it is easy to see from those examples that the meanings of complex deep structures are completely determined by compositionality once the meanings of the lexical items have been specified. This fact merely reflects an important property of homomorphisms in general, *viz.* that they are uniquely determined by their restrictions to generating sets: if h and h' are homomorphisms from an algebra A to a similar algebra B and, moreover, h and h' coincide on all elements of a generating set G for A (i.e. $h(X) = h'(X)$ whenever $X \in G$), then h = h', i.e. $h(X) = h'(X)$ for any X in A's domain. (As can be gathered from the term, a *generating set* for an algebra A is a subset G of A 's domain such that any element of the domain can be obtained by finitely many applications of A 's operations to elements of G or to what one has already obtained.) Since the lexicon is always a generating set for the syntactic algebra - it will indeed be the smallest such set ('smallest' with respect to set inclusion) - any homomorphism and thus any meaning assignment b can be defined by specifying only its values on the lexical items. The compositionality principle (2.5) then extends b to the whole syntax because it says that b has to be a homomorphism.

The fact that a homomorphism is uniquely determined by its restriction to a generating set does, however, not in itself guarantee that arbitrary functions from a generating set into the domain of another (similar) algebra can actually be extended to a homomorphism: uniqueness does not imply existence. One could, e.g., consider an 'ambiguous syntax' including expressions like **John and Bill or Mary**. Clearly, a function assigning meanings to the lexical items **John**, **Bill**, and **Mary** can only be extended to a homomorphism if structural ambiguities never resulted in semantic ambiguities, i.e. if the way a complex expression has been construed did not have any effect on its meaning. This would, however, put a very severe restriction on the semantic algebra and the assignment of meanings to the lexicon. (With respect to the given example it would, of course, also lead to empirically wrong results.) However,

since syntactic algebras are supposed to be structurally unambiguous, this kind of situation cannot arise: one can prove that any function from a generating set of a free algebra into some similar algebra can be extended to a homomorphism. Given any semantic algebra, one is thus free to choose whatever meanings one wants for the lexical items; the choice will always be consistent with the (algebraic) compositionality principle, i.e. it will be guaranteed that every complex expression gets one and only one meaning. (It is, by the way, this freedom of choice that gives rise to the trem *free algebra* .) These elementary algebraic facts justify the common practice (common among semanticists of the Montegovian tradition) of reducing meaning assignments to their restrictions to the lexicon; in later chapters, I will tacitly follow this practice.

I will now further illustrate the algebraic approach to compositionality by discussing a simple example, *viz.* the deep syntax of a certain 'language' of decimal numeric notation. The syntactic categories of that language are the natural numbers 0, 1, 2, etc., where each number intuitively corresponds to the number of digits of the numerals of that category. The lexicon consists of the digits '*0* ', '*1* ',...,'*9* ' each of which is of category 1. (I use italic characters for the digits in order to distinguish them from the numbers they denote.) The syntactic operations form an infinite family of functions F_n , one for each positive integer n . (A *positive integer* is a natural number distinct from 0.) If X and Y are deep structures, then $F_n(X,Y)$ is the result of concatenating X with Y and surrounding the resulting string with a pair of (labelled) brackets: $F_n(X,Y) = \ulcorner[_n\ X\ Y\ ]\urcorner$. (I assume that whenever m and n are distinct positive integers, then the corresponding brackets are distinct, too; moreover, none of these brackets must be a digit and strings are sequences, n -tuples or something like that.) The numeric subscripts on the brackets indicate how long the whole term is: if a syntactically *well-formed* structure (i.e. one belonging to some syntactic catregory) starts with '$[_2$', it contains 3 digits, etc. This multiplication of the single superficial operation of concatenation has been made for semantic reasons to be explained further below. The syntax also contains infinitely many rules all of which follow the general scheme (2.12) which is to be instantiated by replacing n by a positive integer:

(2.12) If X is a structure of category 1 (i.e. a lexical item, a digit) and Y is of category n , then $F_n(X,Y)$ is of category $n+1$.

Finally, the category of sentences is 0. Hence there are no sentences.

In order to see how this syntax works, one first has to observe that (2.12) only allows for 'right-branching' structures. Thus, e.g., '[$_2$*7*[$_1$*12*]]' is of category 3, whereas '[$_2$ [$_1$*71*]*2*]' and '[$_1$ [$_2$*71*]*2*]' do not belong to any category and are thus (*syntactically*) *ill-formed*. The surface structures are, of course, exactly those strings that can be obtained from deep structures by deleting redundant '*0* 's (in the initial position) and the labelled brackets, so that, e.g., '[$_2$ *0* [$_1$*10*]]' and '[$_1$*10*]' both correspond to the surface form '*10*'. The deep/surface relation is thus 'one - many'.

The semantics of this sample language associates with each (well-formed) deep structure the number it denotes. So the domain of the semantic algebra is the set of natural numbers. Moreover, there are certain semantic operations G_n on that domain, where each G_n 'interprets' the corresponding syntactic operatiion F_n. For any natural numbers x and y and any positive integer n one can put:

(2.13) $G_n(x,y) = 10^n x + y$

This finishes the definition of the semantic algebra. I will now turn to the meaning homomorphism b from syntax to semantics. From the general discussion in the first half of the present chapter it has become clear that only b's values for the lexical items will have to be defined; the rest will be done by the compositionality principle. The lexical items are the digits '*0* ', '*1* ', '*2* ', etc., and their interpretation is quite obvious: b('*0*') = 0, b('*1*') = 1, etc. These definitions should now suffice to interpret arbitrary numeric expressions (or, rather, their deep structures). As an example, one might again consider the numeric expression '*712*' with its deep structure '[$_2$*7* [$_1$*12*]] ', i.e. F_2('*7*', F_1('*1* ','*2* ')). One can compute the meaning of this structure in the following way:

(2.14) b (F_2('*7*' , F_1 ('*1*' , '*2*')))
= G_2 (b('*7*') , b (F_1('*1* ','*2*')))
= G_2 (b ('*7*') , G_1 (b ('*1*' , b ('*2*')))
= G_2 (7, G_1 (1, 2))
= $10^2 \times 7 + 10^1 \times 1 + 2$
= 712

The above calculation makes use of the definitions of G_n and b (as restricted to digits) as well as of the fact that b is a homomorphism.

It should be pointed out that a much simpler syntax and semantics could be given by building the numeric terms as left-branching structures: it would then suffice to have a single concatenation function F which could be interpreted by $10x + y$. The corresponding (to my taste more intuitive) left-branching structures can, however, not be interpreted in the intended sense. This example has, indeed, been used by Theo Janssen in order to illustrate Montague's remark "that the aim of syntax can be realized in many different ways, only some of which would provide a suitable basis for semantics". Janssen's observation shows that the use of a whole family $(F_n)_{n \in \mathbb{Z}^+}$ of syntactic operations (where $\mathbb{Z}^+$ are the positive integers) is indeed essential in the above example. It also shows that the principle of compositionality (or, at least, its algebraic version) is not entirely trivial: there are deep syntaxes the structures of which cannot be assigned their (intended) meanings in a compositional way. So compositionality generally imposes a restriction on the relation between syntax and semantics.

These considerations naturally lead to questions about the status of compositionality: should it be regarded as a *methodological principle* or rather as an *empirical hypothesis* ? In practice this question arises if one has succeeded in construing and interpreting some deep syntax without being able to formulate the interpretation in such a way that it fits the compositionality principle. Would that mean that the description was in some sense defective or maybe not as 'deep' as one could wish it to be, so that one should look for a better one? Or would it only cast some doubt on the general validity of the compositionality principle? Different authors have given different answers to these questions and I will not mingle in this dispute here. In the present context, these questions cannot be seperated from the general problem of defining the methodological status of Montague's semiotic theory. To my knowledge, Montague himself did not say anything substantial about these matters; nor will I in the present study. As long as all the examples to be discussed can be captured by the general theory, the problem of determining its precise status is not too urgent anyway.

I am going to finish this chapter by briefly considering an alternative to the principle of

compositionality. Although, from an abstract and theoretical point of view, it seems to be very plausible to assume that the assignment of meanings to expressions or their deep structures should proceed in a compositional manner, there are cases where a non-compositional semantics seems to be more natural. Quantifiers in predicate logic are a good example. One way of interpreting an existentially quantified formula like

(2.15) $(\exists x)\ P\ (x)$

is by reducing the problem to that of interpreting all formulae of the form:

(2.16) $P\ (a)$,

where a is a name of an object in the universe of discourse: (2.15) means that one of the instances of (2.16) is true. (One must, of course, assume that every object has a name.) On the other hand, it seems to be extremely awkward (though certainy not impossible) to formulate the deep syntax of predicate logic in such a way that formulae like (2.15) contain all formulae of the form (2.16) as their parts (in an algebraic sense). It is, of course, much more straightforward to construe (2.15) out of the open formula $P\ (x)$ by simply adding the existential quantifier (together with the variable to be bound). But then compositionality clearly fails if one wants to interpret (2.15) substitutionally, i.e. by reducing it to the interpretation of (2.16). Yet although substitutional quantification violates compositionality, there exists another, more general principle of interpretation which it does accept, *viz*. the principle of the *reduction of complexity* :

(2.17) The meaning of a complex expression is determined by the meanings of (certain) less complex expressions.

That (2.17) is a generalization of the compositionality principle (2.1) is obvious, given the fact that parts of exprssions are always less complex than the expressions themselves. As it stands, (2.17) is, however, not only too vague but also slightly incorrect. This can be seen by considering the universally quantified formula:

(2.18) $(\forall x)\ P(x)$

From a substitutional point of view , the meaning of (2.18) depends on the meanings of all formulae of the form (2.16). But so does the meaning of (2.15). Hence the two cannot *solely* depend on the instances of (2.16), for otherwise they would be synonymous. So the meanings of the complex formulae (2.15) and (2.18) are not only determined by certain less complex expressions but also by their 'mode of construction', i.e. their respective quantifiers. Bearing this in mind, it is then quite easy to find a precise version of (2.17) within the present algebraic framework. To achieve this goal, one could introduce the notion of complexity for deep structures by assigning the 'complexity degree' 0 to all lexical items and letting the complexity of a structure $F(X_1,...,X_n)$ be the maximum of the complexity of $X_1,...,X_n$ plus one. The principle (2.17) could then be restated as:

(2.19) There exists a function f which can be applied to pairs consisting of syntactic operations and sets of meaning such that the meaning of a complex structure X of the form $F(Y_1,...Y_n)$ equals the value f (F , b[M]), where M is some set of structures of complexity less than X.

(The notation ' b[M] ' abbreviates '{ b(Z) | $Z \in$ M }'.) As one can easily prove, (2.19) is indeed a generalization of the algebraic version (2.5) of the compositionality principle, and it is non-trivial in the sense that not every conceivable meaning assignment satisfies it. I will, however, not discuss these matters in any detail because the rest of my study will mainly be concerned with compositional semantics. Moreover, it ought to be mentioned that the substitutional semantics sketched above is, of course, not the only way of interpreting predicate logic quantifiers: Tarski's familiar satisfier sequences (or, equivalently, variable assignments) are equally good tools for interpreting predicate logic, and they admit for a straightforward compositional (and algebraic) treatment of variable binding. In the applications of his theory, Montague thus used to take the Tarskian approach.

Sources

The *locus classicus* of the algebraic account of compositionality is the third section of Montague (1970b). Chapter 4 of Löbner (1976) is an excellent detailed introduction. So are the first two chapters of Janssen (1983), which also include the presently

deepest investigation into these matters.

The origins of the compositionality principle are unclear to me. The term 'Frege's principle' seems to be a slight misnomer the origins of which are obscure, too: see Janssen (1983), chapter I, sections 1 and 2, and van Emde Boas & Janssen (1979), section 1, for some historical background.

Semantic trees are a variant of the 'meanings' introduced in Lewis (1972), section V, for quite different purposes. The fact that synonymous lexical items give rise to identical semantic trees has been used in order to refute Lewis's concept of meaning: see Asher (1984), 236, where the argument is traced back to Mates (1952). As has been indicated above, the fact that arbitrary lexical meaning assignments uniquely extend to compositional assignments is a consequence of a standard theorem of universal algebra: see, e.g., Cohn (1981), 120, Theorem 2.6. Numerical notations and their arithmetical interpretations have been used by various authors in order to illustrate Montague's theory of compositionality. See, e.g., Egli (1974), 53f., Löbner *loc. cit.*, and Janssen (1983), chapter II, especially pp. 72-4, where one can find the above-mentioned illustration of the non-triviality of compositionality; Montague's remark (which is also quoted there) is from Montague (1970b), 373f., fn. 2. - Two contributions to the discussion of the status of compositionality are: Heim (1977), 30-7, and especially Kamp (1981), 298, where the principle is regarded as an (untenable) empirical hypothesis. See, however, Zeevat (1984) for a compositional version of Kamp' s own theory. - The substitutional interpretation of quantification ultimately goes back to Wittgenstein (1921) and has been made popular by Ruth Barcan Marcus: see, e.g., Marcus (1962). The fact that it is not compositional has been observed by Alfred Tarski: see Partee (1973b), 532, fn. 3. The interpretation via variable assignments is due to Tarski (1936); the corresponding algebras have been investigated in Halmos (1962) and Henkin *et al.* (1971).