Fachgruppe Sprachwissenschaft

Universität Konstanz

Arbeitspapier 56

Syntax and Semantics of Construction

Anne Malchow

Contents

Preface

In the course of my work on the many different aspects of linguistics and logics that contributed in one way or the other to the development and justification of the approach presented here, I came across numerous crossroads with many paths that would have been interesting to explore, if it had not been for the limitations of time. Hence, in order to make this paper a useful starting point for further work I decided to include in the bibliography not only the literature I studied in detail, but also to provide references for some related subjects I can not cover in any depth and only mention briefly in the text.

The basic ideas about the Syntax and Semantics of Construction presented here have been developed by Urs Egli. Also, my own linguistic background is deeply influenced by his thoughts and the present version of the theory has grown in many personal discussions with him. It therefore seems impossible to do justice to his part in this piece of work. Instead of plastering the paper with references to his writings, lectures and personal comments I would like to thank him at this point for his support and for knowledge generously imparted to me. From the references given in the text it should still be possible, of course, to find the crucial concepts in their original place.

I'm grateful to Klaus von Heusinger for many helpful comments on the original version of my thesis[1], and to Uta Schwertel and Aditi Lahiri for lots of encouragement and practical support. Special thanks are due to Peter Perryman, who really read “all this flipping linguistics stuff” and made valuable corrections to my English.

1 This paper is a revised version of my MA thesis, accepted in November 1992.

1. Introduction

The aim of this work is to formalize the functional relation between the predicate and its arguments within a ‘Syntax and Semantics of Construction’ that incorporates an extended notion of dependency into phrase structure grammar. Building up a syntax and semantics that take verbs as anadic predicates the system proposed here allows for all verb complements and adjuncts to be handled in the same way as members of a unique class of anadic predicate functors.

The starting point, even though it will be addressed rather late in this work, is the work of Montague. Taking Frege's principle of compositionality as serious as the claim of true generative grammar, he devised a system of precise translation rules that make it possible to systematically build up a semantic interpretation in which the contribution of every single syntactic constituent is recognizable. The syntactic component that Montague's semantics is working on is a modified categorial grammar.

In our approach it will be shown that semantic types can be built up, as well, to parallel a phrase structure grammar with complex categories. I'm going to follow the idea of ‘semantic syntax’, where syntactic categories are interpreted by semantic categories that reflect their ontological status, and syntactic constructions are interpreted by logical-semantical functions.

The Syntax of Construction, although it is designed as a fairly unrestricted, ‘semantic driven’ syntax, is nevertheless the basic component and will be given the most attention. The use of complex categories allows us to integrate the notion of valence into a phrase structure grammar which, in a second step, is going to be extended to all dependency relations, thus introducing Egli's idea of a Syntax of Construction. The semantic interpretation language will be a higher order predicate functor logic with anadic predicates. Extending Montague's technique of type raising I will suggest predicate functors as corresponding to obligatory as well as optional arguments of a verb, which will lead to a theory of adverbials that makes their structural similarity to noun phrases and prepositional phrases obvious and fruitful in a generalized semantic interpretation.

The (probably too ambitious) philosophy behind this approach is that of ‘German as a formal language’; a full-fledged system of the proposed formalism would inherit the rules of inference from type theory and thus might be a good candidate for a “lingua characterica” (Frege 1897:vi), that is, a language to express all the known truths and make deductions in a strictly formalized way, as has already been intended by Leibniz.

However, the system outlined in this paper is anything but fully fledged; the aim, for the time being, is rather to sketch the basic ideas and try to capture only a few of the many natural language phenomena. As for the semantics they are indeed very few. Nevertheless I consider them worth writing down - if only as a starting point for further thought.

In order to avoid losing sight of the basic ideas of this paper, it was necessary to restrict attention to what seems to play a central role in their development; many other subjects that would require more detailed discussion are touched upon only in a very brief and simplified way. The Syntax and Semantics of Construction as proposed in this paper is the result of tying up together several different strands of linguistic theory. As a consequence of this there is no linear structure to be followed throughout the whole paper. Rather we shall be starting out from a number of different points whose relevance for the overall idea will, hopefully, become clear in the end.

The grammar to be developed is a grammar of German and in particular allows the German case system to be dealt with in an elegant way. German examples will be given with a free translation, and a morpheme-by-morpheme gloss where relevant. However, I will often use English examples where the point at stake can be shown without reference to German constructions.

2. The Development of Phrase Structure Syntax

This chapter will give a brief survey of the historical background that the theory presented in this work has grown upon. Most of it follows closely the account given by Egli and Egli (1991). After a short discussion of generative grammar as following from the apparent shortcomings in the structuralist approach, the problems inherent to simple Phrase Structure Grammars will be expounded. The techniques developed to solve these problems, without giving up the major advantages of a monostratal syntactic analysis, are introduced and finally combined with a theory of valency, leading to a first sketch of what is known as Feature Based Grammars.

2.1 From Structuralism to Generative Grammar

The major interest of structuralist linguistics was purely descriptive. Especially in American structuralism, as most prominently represented by Leonard Bloomfield, Rulon Wells and Zellig Harris, the emphasis was on the question of how to find a grammar. Under the impression of problems faced in the work on Red Indian languages traditional means of syntactic description were questioned and so-called discovery procedures were developed to allow a systematic search for constituents and the specification of their syntagmatic and paradigmatic relations. Segmentation and classification, resulting in an immediate constituent analysis (IC analysis), formed the chief techniques that linguists in the structuralist tradition concentrated upon.

However, the consequential questions (often referred to as the projection problem) of how to formulate a grammar and how to proceed from the description of a representative corpus to a theory that leads to reliable predictions, did not play a role in the structuralist view. This shortcoming, next to objections against

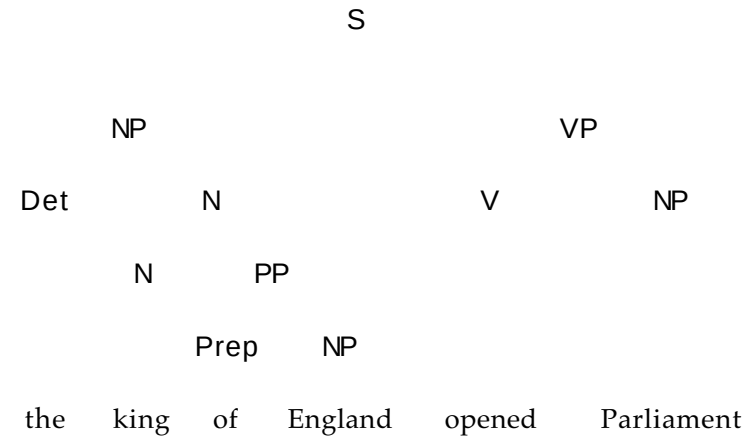
the method of corpus analysis, was to be the main object of criticism by the young Noam Chomsky. Being a student of Zellig Harris as well as of Nelson Goodman and Willard von Orman Quine, he was familiar not only with structural linguistics but also with the methods of analytical philosophy which lead him to realize the importance of clearly defined concepts within a (grammatical) theory. Chomsky's first formulation of a phrase structure grammar, most comprehensively expounded in his “Syntactic Structures” (1957), hence takes a structuralist IC grammar as its basis and refines it by introducing a new standard of mathematical precision.

The crucial step is the introduction of labelled tree diagrams (phrase markers) to show the hierarchy implicitly present in structuralist IC analysis in a more straightforward way. The following example, though still a rather uncomplicated one, should make the practical advantage of this innovation obvious:

(1) a. The king of England opened Parliament.

b. the // king /// of //// England / open- /// ed // Parliament

c.



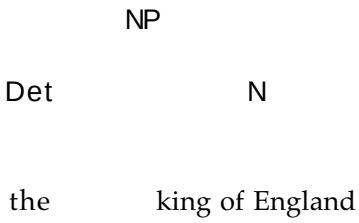
The analysis in (1b) is given in the form used by Rulon Wells (1947), where a single oblique is used to signify a stronger division than a double and so forth. The diagram in (1c) is not only a clearer representation of the structure, but also allows for additional information to be added by labeling the nodes with their respective categories.

The analysis provided by structuralist techniques can be represented in a phrase structure tree and, as Chomsky pointed out, if the analysis of a sentence is correct a set of rules can be extracted from it that can be combined to form new sentences and thus make predictions beyond any given corpus[2]. The diagrams in (2) pick out

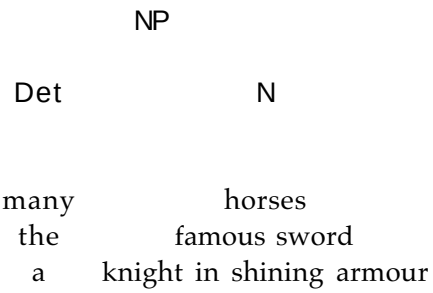
[2] However, concerning the sources of linguistic material Chomsky also departed from structuralist tradition, rejecting corpus analysis as only reflecting performance, prone to mistakes and inaccuracy, instead of describing the real human language competence. Introducing the notion of the ‘ideal speaker’ (Chomsky, 1965) whose performance is not affected by irrelevant interferences or limitations Chomsky contrasts the behaviouristic approach of the structuralists with extreme intuitionism.

one branching from the tree given as (1c) above and shows how it systematically stands for a rule that can be stated in a generalized form for all sentences containing corresponding structures.

(2) a.



b.

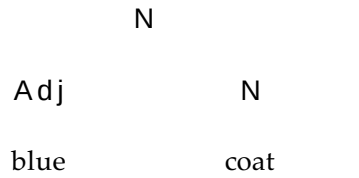


c. A noun phrase consists of a determiner and a noun.

d.[3] Harris: NP = Det + N
Chomsky: NP á Det N

The final step from the generalization formulated in (2c) towards a truly generative formalism was taken when Chomsky took the equations used by Harris (1951) and transformed them into directed rules, known in the theory of automata as rewriting rules or production rules (2d). The new view of grammar as a term rewriting system allows the generation of indefinitely many sentences (now considered mathematically as sequences of words) by a definite set of production rules, and thus solves the structuralist projection problem. The rule and examples given in (3) show how even a single production rule may licence an indefinite number of expressions:

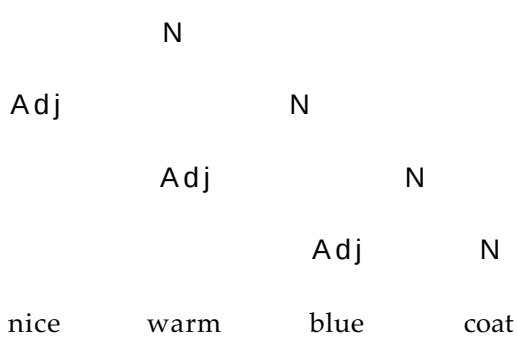
(3) a.



[3] The examples given in (2d) are only meant to show the general format of equations vs. rewrite rules employed by Harris and Chomsky respectively.

(3) b. N á Adj N

c.



The rule (3b) as it can be extracted from a construction like (3a) is a special one, in that the left-hand symbol that is to be expanded (N) occurs again on the right-hand side where it might be expanded again into an adjective followed by a noun, and so on. Rules of this general form are called recursive rules, and their recursive application may yield expressions like (3c) or indefinitely longer ones. Actually, it was the theory of recursion, developed within analytical philosophy, that the idea of generating language originally came from. The new paradigm of generativism is characterized by a view on language that is mathematically orientated: sentences and phrases are seen as linear sequences, a language as a set of such sequences. The benefit of this view was not only that it introduced a new standard of precision into linguistic analysis, but also the possibility to abstract away from the cumbersome facts of any natural language and to gain new insights about the meta-theory of grammar through the study of formal language systems.

2.2 Problems with Simple Phrase Structure Grammars

The idea of phrase structure grammar, however, was threatened from the beginning by problems that had already been noticed by Harris: there are certain linguistic phenomena that cannot be represented adequately in a linear description. Very pertinently, one chapter of Chomsky's "Syntactic Structures" (1957) is called "Limitations of Phrase Structure Description", where he shows how phenomena like conjunction, agreement or relations between sentences seem impossible to put in simple phrase structure rules. For example, the relation between the active and the passive form of a sentence as given in (4):

(4) a. Paul loves Maria.
b. Maria is loved by Paul.

The obvious connection between these sentences cannot be described in terms of local rules (that is, rules that only licence one particular node in the structure tree, as shown in (2)). A pure phrase structure grammar would have to build up two completely different analysis trees and thus need two different sets of rules, which seems a way neither efficient nor elegant to deal with sentences that patently have a lot in common. Chomsky's solution to this kind of problems are

rules that take sentences as their input and form new sentences. For this particular case there could be a rule like (5), expressing the generalizations that can be made about the relation between active and passive:

(5) If NP_1 V NP_2 is a grammatical sentence, then
NP_2 Aux $V_{past.part.}$ "by" NP_1 is also a grammatical sentence.

Following Harris, who had worked out similar procedures (1957), Chomsky called this sort of rules 'transformations', and in his "Aspects of the Theory of Syntax" (1965) he developed a comprehensive model of grammar, known as the 'Standard Theory'. In this grammar only a small kernel grammar of phrase structure rules is left to build up the so-called deep structure which is then subject to transformations for the derivation of different surface structures. Under this view the sentences (4a) and (4b) above are derived from the same underlying deep structure by application of different transformations.

However, we will not go into the transformational approach, since it clearly departs from the idea of phrase structure as a prime principle of language analysis. While for some time transformational grammar (TG) seemed the one and foremost paradigm within linguistic research, the general enthusiasm about it was subdued in the 1970s by some results about the computational complexity of existing TG formalisms. Stanley Peters and Robert Ritchie (1973) showed that their generative power was too great; a grammar encoded in a TG formalism is equivalent to a Turing machine. For a language to be accepted by a Turing machine, however, means nothing more than that it is a recursively enumerable language, i.e. it belongs to the languages which can be characterized mathematically. Since natural languages are known to be at least context sensitive (that is, their characterization can be accomplished by a rewriting system where each rule is of the form ABC á ADEC, intuitively describing the situation that in the context A_C a category B may be expanded into DE) transformational grammars are obviously not restrictive enough to lead to any theoretically meaningful conclusions. Within the TG community the Peters-Ritchie result induced the imposition of radical restrictions to the transformational apparatus, ending up with nothing more than the minimal rule "move α" in the theory of government and binding, the modern descendant of transformational grammar (Chomsky 1981). However, for at least a decade the generative idea was abandoned as sour grapes, or, as Gerald Gazdar in his account of that time puts it:

> "The fact is that generative grammar, in the original sense of the term, virtually died out in linguistics in the 1970s. [...] For the most part, theoretical syntacticians in the early 1970s had no interest in mathematical models of language. [...] nor did those working in the [transformational] frameworks see any merit or interest in matters of formal detail."
>
> (Gazdar 1987:125)

On the other hand, the negative results concerning transformational grammar lead to a new interest in monostratal phrase structure grammar, that is, a grammar that uses only one level of syntactic description without reference to any sort of deep structure.

2.3 Monostratal Techniques in Phrase Structure Grammar

An outstanding exception to the tendency criticized by Gazdar was the work of the logician Richard Montague: “he showed that it was possible to do generative grammar” (Gazdar 1987:126) and “reintroduced [...] the goal of writing completely explicit grammars” (ibid.: 127). Montague had formalized fragments of English in a modified version of categorial grammar and, moreover, provided it with an explicit semantics. Since his influence has been on the theory of semantic interpretation for the most part, we will have a closer look at his ideas in the second part of this work, in chapter 7.

The revival of ‘pure’ phrase structure grammar in order to avoid the computationaly too costly transformations received a major impetus in the beginning of the 1980s through the works of Gerald Gazdar. He showed how by means of complex categories[4] and the use of so-called gaps transformations could be ‘simulated’ within a monostratal system. In this section a simplified account of both techniques will be given that is to be extended in 2.5 to the degree of detail necessary for the present purpose.

2.3.1 The Notion of Complex Categories

One of the problems that lead to the idea of transformations was the fact that certain morpho-syntactic features can be spread over a whole sentence, manifesting themselves in more than one constituent (therefore they are also called ‘suprasegmental’ or ‘long components’). For German nominals they are case, person, number and gender; for verbals, person, number, tense, mood and voice. While each word can be categorized as ‘noun’, ‘adjective’ or ‘verb’ etc., each noun, for example, belongs again to a certain secondary category (or sub-category) depending on its case, person, number and gender. Example (6) shows the difficulties arising with the phenomenon of number agreement:

(6) *This dog* with two tails never *likes* to see *himself* in the mirror.

The words printed in italics all have to appear in third person singular, since it is only one dog being talked about. This cannot be formulated in ordinary phrase structure rules, unless we would accept a multitude of special rules stating, for example, that a singular noun phrase consists of a singular determiner plus a singular noun, while a plural noun phrase consists of a plural determiner and a plural noun, etc. In a grammar puffed up like that however, the structural regularities peculiar to all noun phrases would be hidden under a heap of specialized rules.

A number of different ways have been taken to avoid this loss of generality. In a transformational approach, as we have seen, a basic phrase structure component generates the deep structure of a sentence which subsequently has to go through one or more transformations to acquire its surface form. A transformation rule for number agreement would have to state a structural change copying the features ‘person’ and ‘number’ from the noun (where they are generated) to all the other

[4] The idea of complex categories has in fact been formulated much earlier by Klaus Brockhaus (1971) and was further developed by Angelika Kratzer, Eberhard Pause and Arnim von Stechow (1973).

constituents that have to agree with the noun. An alternative way to represent agreement phenomena without giving up the idea of one level of representation is the use of complex categories, developed by Gerald Gazdar (1982; Gazdar et al. 1985). The concept behind this is the combination of primary and secondary categories in bundles of features. Feature bundles have been used in phonology to describe a phonem by the presence, or absence, of certain distinctive features. Similarly a word or a phrase can also be endowed with a set of features, as for example the German word for 'dog' in (7):

(7) Hund: ⎡ **cat: N** ⎤
⎢ pers: 3. ⎥
⎢ num: sg ⎥
⎣ gen: masc ⎦

The major category, however, still takes a prominent position amongst the features, since it plays a special role in the formulation of the rules. The first conceivable way to state a rule now would be something like this: 'A noun, if it is in the plural form and masculine, combines with a determiner that is plural and masculine, to form a plural masculine noun phrase'. However, we are going to run into problems with this sort of rule as it is no longer context free. In the format introduced in (2d) (repeated here slightly changed as (8a)) a rule is always applicable unconditionally. Compare the two abstract rules in (8) below:

(8) a. A á B C
b. x A y á x B C y

The rule in (8b) is to be read as: 'in the context x_y an A may be rewritten by the sequence B C'. Because their application depends on a fixed context, rules of this general form as well as the grammars containing them, are called context sensitive. In a hierarchy of grammar types concerning their generative power, context sensitive grammars appear as less restricted than context free grammars. That is, the set of languages that can possibly be generated by a context free grammar (= context free languages) is a proper subset of the context sensitive languages (obviously, a rule of the shape depicted in (8a) can always be adapted to the context sensitive form of (8b) by assuming an empty context x_y, but not the other way round). Although it is a controversal question whether all natural languages can be described context free[5] it is still desirable to chose a formalism as restricted as possible and to narrow down the number of possible forms that a phrase structure rule may take.

The solution to the dilemma between loosing generality and expressiveness or loosing the desired restrictedness is the use of rule schemata. A rule schema is a general rule which stands for a number of specialized rules. For example the rule schema for a German noun phrase in (9) is to be understood as standing for six rules, spelling out the six possible combinations of 'singular' and 'plural' with 'masculine', 'feminine' and 'neuter':

(9) NP á Det N

[5] Cp. Shieber (1985) for a proof that some Swiss German constructions are not context free. The introduction by Partee et al. (1990) gives a good survey of so-called 'mildly context sensitive' grammars like indexed grammars or tree adjoining grammars which stand between context free and context sensitive grammars and are capable of dealing with the Swiss German data.

⌈ num: α ⌉	⌈ num: α ⌉	⌈ num: α ⌉
⌊ gen: β ⌋	⌊ gen: β ⌋	⌊ gen: β ⌋

The crucial ‘trick’, introduced by Gazdar, is that although rules like (9) look as if they were context sensitive - by taking them as mere abbreviations for all the individual rules summarized in the schema they can be kept context free. Or the other way round, we can have a grammar that actually contains, for example, a special rule for each sort of noun phrase without loosing all generality in a mess of idiosyncratic rules, by combining the linguistic ‘essence’ of a class of rules in a single rule schema. We will come back to the topic of rule schemata later when for the discussion of dependencies and their implementation into phrase structure, Gazdar's Generalized Phrase Structure Grammar and other formalisms making use of complex categories will be looked at in some more detail.

2.3.2 The Use of Gaps

Another technique that, next to the idea of complex categories, has revolutionized modern conceptions of phrase structure grammar, is the use of gaps or so-called SLASH features. A class of phenomena for which transformations were thought to be indispensible are the various movements within sentences responsible for the grammaticality of different word orders. The transformational conception of a constituent being moved from one position to another is an image that might be helpful to be kept in mind when looking at the SLASH theory; thus, as in the previous chapters we will have a quick look at the transformational approach first, before we proceed to introduce methods for a monostratal description.

Transformations in the Standard Theory were specific rules stating a structural description that had to be matched by the input sentence and a structural change which was to be brought about by a series of elementary transformations on the phrase structure, namely deletion, substitution and adjunction (Winograd 1983:159). In the version put forward by Chomsky in 1975, called the Revised Extended Standard Theory (REST), all structure dependent transformations were abandoned in favour of one generally applicable rule “move-α”, where α stands for anything. We will not have to go into the independent principles that constraint this generalized movement rule, but want to point out that according to the aim formulated in REST, constituents moved from one place to another leave a trace in their original position, so that the resulting surface structure can be used as the basis of semantic interpretation.

Influenced by Chomskian trace theory, Gazdar's idea was to simulate movement within a single tree diagram (that is, by a set of rules that could be extracted from this one tree). Since any phrase structure rule can only describe one particular node and its daughters, the movement of a constituent has to be expressed in terms of local movements. To achieve this, a new set of categories was introduced that allows us to distinguish between a category X and a category X/Y (read: ‘X slash Y’) which is exactly like X except that where X contains a constituent Y, this category Y is missing in X/Y. Instead of having a derived category X/Y, it is now customary to take missing elements as the value of a secondary feature, called SLASH feature[6]. (10) shows X/Y in a feature matrix as used above:

(10) $\begin{bmatrix} \textbf{cat:} & \textbf{X} \\ \text{SLASH:} & Y \end{bmatrix}$

Where it seems more convenient we will go on using the familiar form X/Y, taking it, however, as denoting an X with the value Y on the feature 'SLASH'. A typical example for movement is topicalization, where a constituent is moved from its normal position to the beginning of the sentence to express a special emphasis. For example[7] the sentence in (11a) can be transformed into (11c) to put the emphasis on 'doughnuts' as perhaps one would do in a reply to (11b):

(11) a. Susan likes to buy doughnuts.

b. Susan doesn't like to go shopping.

c. Doughnuts, Susan likes to buy.

d. doughnuts$_i$ Susan likes to buy t_i

Thinking of the object 'doughnuts' as being moved to the top position, it is assumed that it leaves a visible gap or a trace behind as is shown in (11). For the introduction of such a trace into the phrase structure the SLASH theory provides a rule schema of the following form:

(12) X/X á t

A trace, thus, always belongs to a category X/X; in the case of (11c) this is the category NP/NP, a noun phrase with a missing noun phrase. When the phrase structure builds up, the information that a noun phrase is missing has to be handed up the tree, to make sure no incomplete sentence like (13) would be accepted.

(13) Susan likes to buy

At this point the benefit of regarding NP/NP not as a special category but as an NP with the secondary feature SLASH: NP becomes apparent: Early versions of the gap mechanism (e.g. Gazdar 1981) used a set of rules with SLASH categories that were systematically derived from the set of basic rules. If there was, for example, a rule (14a) then the rules in (14b) could be derived from it (Bear 1982:1).

(14) a. S á NP VP

b. S/NP á NP/NP VP
S/NP á NP VP/NP
S/PP á NP/PP VP
S/PP á NP VP/PP

Thinking of gaps as features, the role of these rules can be taken by a general principle which Gazdar called foot feature principle (FFP), stipulating that it

6 The use of "gaps as syntactic features" was first suggested by John Bear (1982).

7 Gazdar et al. (1985:145) provide two examples showing how the analysis with SLASH-categories allows to solve ambiguities, also. These are repeated by Egli and Egli (1991:62) in a simplified form. As the example given here is meant solely to demonstrate the general idea of gaps, it is again simplified in that it does not contain any ambiguities. Otherwise I will follow the account of Egli and Egli.

should apply to all features marked as so-called FOOT features. “The simplest formulation of the foot feature principle [...] would be to require that any FOOT feature specification instantiated on a daughter category in a tree must also be instantiated on its mother category.” (Horrocks 1987:186) So obviously we want our SLASH feature to be a FOOT feature to ensure the propagation of its values up the tree.

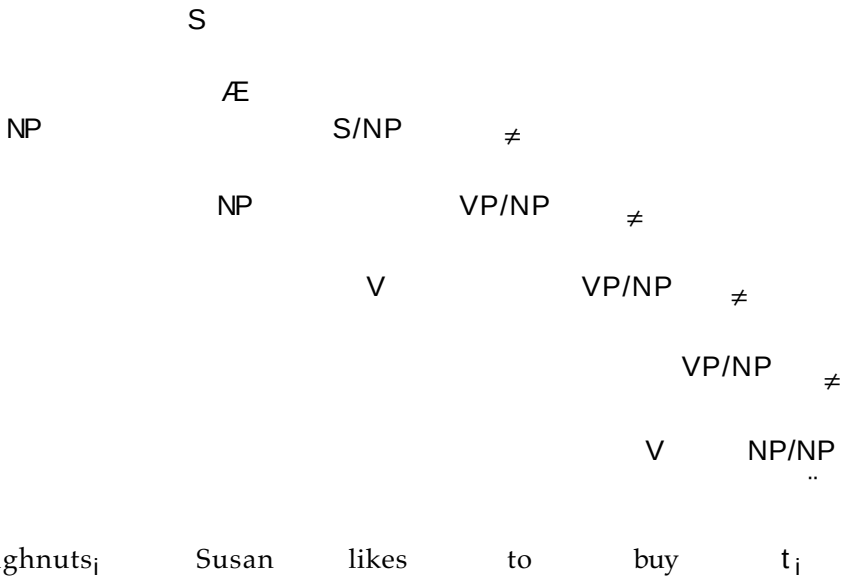
The last problem then, is how to stop the SLASH information from being handed over to the next dominating node when indeed the constituent reported missing has been found. Since constituents cannot be moved just anywhere in the sentence, this is the most complicated problem. To allow for the possible, and only the possible movements, it is necessary to formulate a number of idiosyncratic termination rules. The one we need for the sentence in (11b) (repeated as (16a) below) is shown in (15):

(15) S á NP S/NP

To sum up the theory of gaps, we are going to have a look at the three steps pertaining to it and see how they work in the case of our example (16):

1. Instantiation of the SLASH feature by the general rule X/X á t
2. Feature propagation up the tree by the foot feature principle
3. Elimination of SLASH feature values by idiosyncratic rules

(16) Doughnuts, Susan likes to buy.



The use of SLASH features as well as other features bundled up in a complex category requires a few additional techniques, to arrange the information in an efficient way and control the percolation of features through a tree. These will be touched upon again briefly in 2.5 on the combination of dependency and constituency. For the time being we will take a step back and follow a different strand of linguistic work that contributed an important aspect to the current

theory of grammar.

2.4 Tesnière's Dependency Grammar and the Theory of Valency

While American structuralism strongly emphasizes the notion of constituency, the French linguist Lucien Tesnière concentrates on the logical relations between different parts of speech. In his book, Eléments de syntaxe structurale (1959), Tesnière takes a perspective on the structure of language that is primarily concerned with the dependencies within linguistic constructions. His theoretical work on a dependency grammar became very influential on the theory of valency.

The term 'valence' or 'valency' is borrowed from chemistry and describes the ability of some words to open up places around themselves that are to be filled with arguments of a certain kind, similar to the chemical processes of molecular binding where, for example, an oxygen atom may combine with two hydrogen atoms in order to reach a stable state[8]. A word with such an ability to take arguments may be a verb (17), an adjective (18), a preposition (19) or a noun (20), and their respective arguments are said to depend on them because they could not occur without them. The examples given as (17-20) make this dependency obvious for each class:

(17) a. Er aß einen Apfel.
he ate a-acc apple-acc
[He ate an apple.]
b. Er aß.
[He ate.]
c. *Er einen Apfel.
[*He an apple]

(18) a. Er war des Lesens müde.
he was the-gen reading-gen tired
[He was tired of reading.]
b. Er war müde.
[He was tired.]
c. *Er war des Lesens
[*He was of reading.]

(19) a. Er wohnte mir gegenüber.
he lived I-dat opposite
[He lived opposite me.]
b. Er wohnte gegenüber.
[He lived opposite.]
c. *Er wohnte mir
[*He lived me.]

(20) a. Er verlor alle Hoffnung auf eine bessere Zukunft.

8 Similarly, it was a chemical concept that Frege had in mind when he introduced the notion of 'saturated' and 'unsaturated' predicates.

he lost all hope of a-acc better-acc future(-acc)[9]
[He lost all hope of a better future.]

b. Er verlor alle Hoffnung.
[He lost all hope.]

c. *Er verlor auf eine bessere Zukunft.
[*He lost of a better future.]

In the traditional grammars of scholasticism these relations were known as 'gubernare' ['to govern'], and the first observations about their nature date back to ancient Greek philosophy (cp. Egli and Egli 1991:85): As early as in the period of Stoic philosophy of language it was noticed that subjects as well as predicates were both endowed with certain secondary features, those which today we know as morpho-syntactic categories. In the second century AD Apollonios Dyscolus wrote, "a predicate is combinable (syntaktôn) with an object of a particular case", employing exactly the locution used earlier by the Stoics, the word which was to become the specialist term for grammatical structure.[10]

As Apollonios Dyscolus describes, the valence, not only of verbs, may also be stated in terms of cases. A word is said to govern its dependants because it determines their morphological form, in particular their case. This is most relevant, of course, for inflectional and agglutinating languages[11]: for example, the German verb 'geben' ['give'] governs the nominative, dative and accusative as is shown in (21):

(21) Der Mann gibt dem Jungen den Schlüssel.
the-nom man-nom gives the-dat boy-dat the-acc key-acc
[The man gives the key to the boy.]

Similarly, adjectives, prepositions and nouns may govern different cases. However, most prominent of all dependency relations is the verb's valence[12] as it functions to pre-determine the structure of a sentence and is commonly used as a criterion to divide verbs into subclasses (or subcategories).

On the first level, verbs are categorized according to the number of arguments they take: a verb like 'sleep' can be combined with a subject alone and is therefore called monovalent (or intransitive), while others, like 'love' or 'give', take also a direct object, or a direct and an indirect object and hence are called bivalent (or transitive) and trivalent (or bitransitive) respectively. The alternative names given in parenthesis indicate that it is a controversial question whether the subject is to be regarded as belonging to a verb's valence or not. In accordance

---

9 The case is given in parentheses because the word 'Zukunft' has no visible inflection.

10 Quoted after Egli and Egli (1991). For more information see Householder's translation of Apollonios Dyscolus' syntax (1981).

11 Although in English, too, a rudimental case system is still extant in the use of pronominals.

12 In the narrow sense the notion of 'valency' is restricted to the relation between a verb and its arguments while, for example, the dependence of an attributive adjective on a noun is called 'determination'.

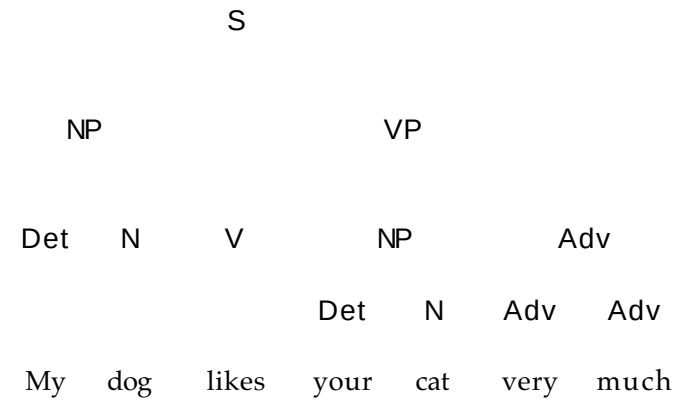
with Tesnière's original approach and for reasons to be amplified later on, valency will be taken to comprise the subject as well as the objects, even though the syntactic properties of the two are rather different in that a subject agrees with the verb while objects do not.

A second level of looking at the argument structure given by a verb concerns not the number but the grammatical case of the arguments required: for example, while the German verb 'lieben' ['love'] takes an accusative object ('Sie liebt *ihn*' ['She loves him']), there are others, like 'helfen' ['help'] which have to go with a dative object ('Sie hilft *ihm*' ['She helps him']).

Tesnière's idea was a hierarchical order of dependencies where one governing element may in turn be governed by another one. The resulting tree diagram, called 'stemma', shows the dependencies between elements of a sentence, but not its constituent structure. For the sentence given in (22a) the diagram in (22b) provides an immediate constituent analysis and (22c) a schema of dependencies:

(22) a. My dog likes your cat very much.

b.



c.

The node at the root of (22c) above is the 'absolute governor', the main verb, and governs two nominal elements and one adverbial. Each noun, in turn, governs a determiner and the adverb 'much' governs the adverb 'very'. The corresponding rules may look something like the following, where the left hand category is the controller or governor and the asterisk marks its position in the sentence:[13]

[13] This is only a simplified set of rules to give a rough idea about how dependence rules may be stated. For a more detailed description of Tesnière's dependency grammar, as well as further developments based on his approach see Heringer et al. (1980, ch. 4).

(23) a. V (N, *, N, Adv)
b. N (Det, *)
c. Adv (Adv, *)

The rule stated as (23a), for example, is to say that a verb can govern a preceding noun as well as a following noun and adverb.

Tesnière introduced a distinction between necessary arguments (complements) and optional arguments giving additional but not necessary information (adjuncts)[14]. However, in the current discussion this traditional complement/adjunct dichotomy is no longer considered sufficient to describe the facts. There is now a preference to assume three classes of arguments, distinguishing in particular between obligatory and optional complements which both depend on the verb, and adjuncts which are optional and in no way pre-defined by the verb's valence. The main candidate for an obligatory_complement is, of course, the subject, because a sentence without a subject is ungrammatical (note that even where there seems to be no natural 'agent' to fill the subject role, a so-called grammatical subject is usually inserted, as in: '*It* is raining'). A typical member of the new class of optional complements is, for example, the direct object of 'eat' which is clearly governed by the verb since it is assigned the case accusative ('The shark ate *him*'), but may as well be totally omitted ('We are eating'). Pure adjuncts, in contrast, may occur in any number or not occur at all, and are independent of the verb's case assignment ('They were eating with them at seven o'clock in the dining room'). However, the distinction between adjuncts and optional complements is not a clearly defined one and in asking whether, for example, the valence of 'come' includes something like a 'where-to' or 'whence' role, we enter the vagueness peculiar to the interface between syntax and semantics. We will have to address this question again in section 4.1 when we will deal with the definition of construction frames.

2.5 From Dependence Schemata to Subcategorization

We have now seen two different notions of syntactic hierarchy, one of which may be formulated in some sort of phrase structure grammar and the other as a dependency grammar. However, the expressive power of the two is not the same: a dependency grammar may not always allow the proper reconstruction of constituency and, on the other hand, the tree diagram showing a phrase structure analysis will conceal all the information about dependencies. In other words, even though both grammars are capable of generating the same set of sentences, they don't give identical descriptions of their structure. It is said therefore that the two systems are weakly equivalent. Two grammars that are inter-translatable in that they produce the same set of sentences and assign the same structural descriptions to them are called 'strongly equivalent'.

14 Tesnière calls them 'actants' and 'circonstants'.

2.5.1 The Amenability of Valency to Phrase Structure

To combine the respective advantages of the two approaches described above the information about dependencies should be integrated into phrase structure rules. Matthews (1981:88ff) introduces a combined representation for dependence and constituent structure which, for sentence (24a), would look like (24b)[15]:

(24) a. My dog sleeps.

b. $_{S}[\ _{NP}[\ _{Det}[my]_{Det}\ _{N}[dog]_{N}\ ]_{NP}\ _{VP}[sleeps]_{VP}\ ]_{S}$

Matthews' amended phrase structure rules thus embody the arrows used to show the dependence relations:

(25) a. NP á Det N

b. S á NP VP

In the following section we shall have a look at a more elegant way to import Tesnière's notion of valency into phrase structure grammar that has become an important feature of many modern approaches to grammar. To encode the information given by Matthews' arrows it turns out advantageous to make use of complex categories. Egli and Egli (1991:101ff) supply an intuitively accessible description of the ideas providing the background for dealing with dependency in this way. As an intermediate step we can think of labelled arrows showing not only the fact of some dependency holding between two constituents, but stating the case assignment made by the governor and thus specifying the kind of dependency. Picture (26) shows the dependence arrow amended with a category label.

(26)

S

NP VP

Hans sleeps

nom

(Egli and Egli 1991:103)

To state dependency relations within complex categories we have to analyse dependence as an interaction between the valence of a verb and the arguments filling the positions provided. In this view valence may be regarded as a verb's demand for arguments of a certain kind; the appropriate arguments then are to

15 Matthews' examples and rules are slightly different. But as we are interested in his representation only as an intermediate step towards the integration of dependency information into phrase structure, I will not go into the details of his method.

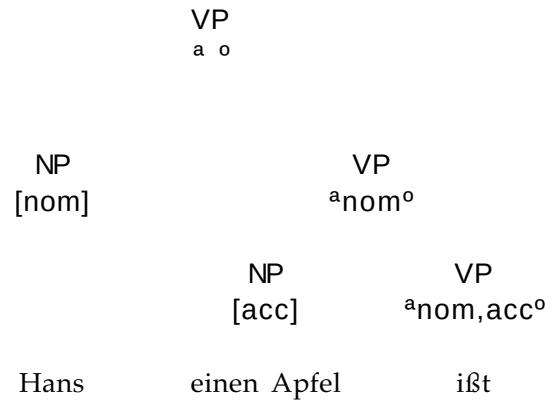
meet the demand. Thus split into two different but corresponding aspects, the information about a dependence relation can be encoded locally in terms of features demanded by the governor and features displayed by the dependant. These features will be taken to be features in a complex category as described above in section 2.3.1. One of the secondary features comprised by the complex description of a noun phrase is its case. A verb governing one or more nominal constituents is endowed with a list of cases demanded of, or assigned to, its arguments. Egli and Egli (1991:86) call such a list a case frame, using the word introduced by Fillmore (1968) in connection with the deep structure of his case grammar.

Before we look at an example to make this idea clearer, we have to introduce a simplification that has become widely accepted in German linguistics: according to arguments of Manfred Bierwisch (1963), which will be presented in more detail in chapter 3, it is customary to assume the word order of subordinate sentences as the underlying basic word order of all German sentences. This is a convenient simplification as it leaves us with the complete set of nominal complements and adjuncts standing in front of the verb. (27b) shows the sentence (27a) in the form we will be looking at in (27c):

(27) a. Hans ißt einen Apfel.
[Hans eats an apple.]

b. (Lotte weiß, daß) Hans einen Apfel ißt.
[(Lotte knows that) Hans eats an apple.]

c.



In (27c) dependency is finally integrated into phrase structure in the form of a verbal demand and its fulfilment by the arguments. We introduce the convention that any relevant features of a constituent are included in a pair of square brackets. In this example we are only interested in the case of each argument, therefore information about number or gender etc. is omitted. The verb's case frame, indicated by pointed brackets, contains the cases of the required complements and shrinks to the degree that arguments of the appropriate case are absorbed into the verb phrase. This corresponds to the intuitive conception that 'eat an apple' has the form of an intransitive verb, just as 'sleep', requiring only to be combined with a nominative as its subject. It also reflects the method used by Montague where a verb phrase is built up gradually by adding one argument at a time. However, the Chomskyan conception that a verb is combined in one step with

all complements under its government, is compatible, too, with the idea of implementing dependence into constituent structure by means of case frames (Egli and Egli 1991:87). The conception that a verb phrase with empty case frame stands for the category sentence (S = $V^{a\ o}$) has its origin in the logical view of a sentence as a predicate of degree 0. A predicate taking no arguments corresponds to a verb with empty valence, or in Frege's terminology, a saturated predicate (Ibid.:89). The generative rules that can be extracted from the tree in (27c) are given as (28a) and (28b). The rule in (28c) shows how they can be formulated in a generalized way, the variable C standing for a list (possibly empty) of cases and c for one single case:

(28) a. $VP^{a\ o}$ á NP[nom] $VP^{a}nom^{o}$

b. $VP^{a}nom^{o}$ á NP[acc] $VP^{a}nom,acc^{o}$

c. $VP^{a}C^{o}$ á NP[c] $V^{a}C,c^{o}$

For a system as the one to be introduced in this work, generalization can be driven even further. However, before the 'Syntax of Construction' devised by Egli and Egli (1991) will be introduced, there are a few remarks to be made on the young tradition of feature based grammars within which many of the techniques employed here have been developed.

2.5.2 Feature based grammars[16]

As mentioned earlier in the chapter on the development of phrase structure grammar, the crisis of transformational grammar and a new set of techniques proposed by Gerald Gazdar in the late 1970's and beginning of the 80's revived the interest in monostratal approaches to grammar and quickly lead to a surge of new theories, the common basis of which was their crucial use of complex categories. According to this central strategy, the decomposition of atomic categories into a set of characteristic features, these grammars were often subsumed under the heading of 'feature based grammars'. Most prominent and influential among the class of feature based grammars are the following[17]:

- **F**unctional **U**nification **G**rammar (Kay 1979)
- **G**eneralized **P**hrase **S**tructure **G**rammar (Gazdar)[18]
- **L**exical **F**unctional **G**rammar (Bresnan, Kaplan)[19]
- **C**ategorial **U**nification **G**rammar (Karttunen)[20]
- **H**eaddriven **P**hrase **S**tructure **G**rammar (Pollard, Sag)[21]

16 This chapter has been written on the basis of my article on Mark Johnson's attribute-value formalism (Malchow 1992) and is an abridged version of the introductory chapters included there.

17 Compare the survey given by Shieber (1986).

18 First outlined by Gazdar (1981), a "fairly complete exposition" of the theory is given in Gazdar et al. 1985.

19 Developed during the late 1970s, it is most comprehensively described by Joan Bresnan (1982).

20 Karttunen (1986), Uszkoreit (1986)

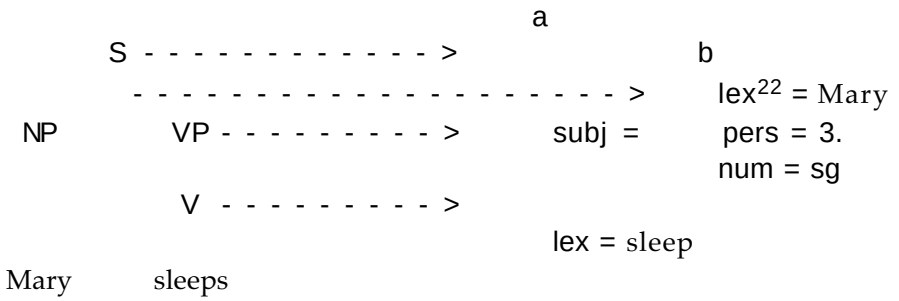
21 HPSG was first introduced through some papers published by Carl Pollard since 1984, subsequently it was developed by him and Ivan Sag (Pollard 1984; Pollard and Sag 1988).

There are quite a number of different names used for feature based grammars; ‘constraint based’ or ‘information based grammars’, ‘attribute-value grammars’ (Johnson 1988) and, after a term used by Martin Kay (1979), ‘unification grammars’. The last one of these names puts the emphasis on one of the crucial techniques used in all feature based approaches to grammar.

The use of features turned out attractive since a set of such features can be seen formally as one complex symbol and thus “does not increase the power of the grammar” (Winograd 1983:168). There are three different ways to structure complex categories the choice of which is basically a matter of taste: tree diagramms (the so-called ‘dags’ (dyrected acyclic graphs)), terms or matrices. The grammar fragment provided at the end of this work is written in a format that builds up feature matrices, and I will introduce the basics of that formalism in this chapter.

The formalism to be used has been designed by Mark Johnson (1988) in order to provide a versatile ‘grammar writing language’ independent of any particular linguistic theory (within the feature based framework). The general idea is that each non-terminal node in a phrase structure tree is annotated with a feature matrix specifying all known details about the syntactic element represented by the node. Figure (29) shows an example of an annotated phrase structure tree:

(29)



(Johnson 1988:5)

In this simple example, there are two so-called attribute-value elements that specify the syntactic properties of the constituents. The verbal nodes (including S) are assigned the element ‘a’, containing the information that the lexical form (lex) of the verb is ‘sleep’ and that its subject (subj) is the constituent specified in the attribute-value element ‘b’, which is in turn connected with the NP. The entries in the lexicon of an attribute-value grammar would look like (30) below:

(30) a. NP á Mary
m(num) = sg
m(pers) = 3.
m(lex) = Mary

22 Johnson calls this attribute ‘pred’.

(30) b. V á sleeps
m(subj)(num) = sg
m(subj)(pers) = 3.
m(lex) = sleep

In these entries ‘m’ stands for the mother node that is licensed. The attributes in parenthesis specify a sort of ‘path’ to the required information. For example, ‘m(subj)(num) = sg’ means that the number of the subject of the verb is singular. In (31) below, the two syntactic rules that are applied in (29) are given. The ‘d_i’ stand for daughter nodes.

(31) a. S á NP VP
m = d_2
m(subj) = d_1

b. VP á V
m = d_1

Syntactic rules in a feature based grammar are endowed with certain conditions of well-formedness. For example (31a) demands that the VP and the NP agree in number and person. This is hidden in the rather terse formulation, of the fact that the VP has all the properties of the S, and the value of the subject-attribute of S (therefore also of VP) has to meet exactly the qualities of the NP. The technique of comparing two sets of features and to merge them together unless they contain incompatible information is called unification. For example, if the number of the verb was plural, the unification of m(subj) (= d_2(subj)) with d_1 would fail since the value of the NP for number in our example is singular. The rule in (31b) does not contain any conditions, but only an instruction how to build up the attribute-value element of the VP, it simply inherits all information from the V.

The information about valency is encoded by means of a so-called ‘subcategorization list’. Reflecting the fact that not all verbs are equal in their argument structure, the idea is to divide them into ‘subcategories’, for example the subcategory of those that take a nominative and an accusative as their complements. The subcategorization list is the value of an attribute (subcat) and particular parts of the list can be picked out by another attribute: the value of (subcat)(first) is the first element of the list, while (subcat)(rest) gives the remainder of the list after removing one element.

Another technique that has been used is to view even the major category as just one of the features and, in particular, to catch the generalities between different categories by defining them in terms of two categorial features, where a verb is defined as +VERB -NOUN, a noun as -VERB +NOUN, an adjective as +VERB +NOUN and a preposition as -VERB -NOUN. This approach should be easily compatible with the rules given here. However, in order to show the effects of the proposed mechanisms as clear as possible, we will cling to the traditionally known category names and not explore the possibilities of feature grammars any further.

3. German Word Order

Before we start looking at the Syntax of Construction in chapter 4 there is something to say about German word order. Though not to be counted amongst the non-configurational languages, German exhibits a relatively free order of constituents compared, for example, to English or French. There is no fixed order for subject, predicate and object. Multiple object sentences may appear in different orderings, and adverbial adjuncts can precede or follow almost any of the other sentence constituents (Bierwisch 1963:31). This chapter will introduce some of the basic German sentence patterns and point out the major grammatical processes involved in their derivation.

Restricting our attention, for the time being, to just one aspect of word order, namely the position of the verb, there are three possible orderings to be considered. The following three examples illustrate the different structures of main clauses (32), subordinate clauses (33) and yes-no-questions (34):

(32) a. Der Hund jagt die Katze.
[The dog chases the cat.]
b. Der Hund hat die Katze gejagt.
[The dog has chased the cat.]

(33) a. … weil der Hund die Katze jagt.
[… because the dog chases the cat.]
b. …weil der Hund die Katze gejagt hat.
[…because the dog has chased the cat.]

(34) a. Jagt der Hund die Katze?
[Does the dog chase the cat?]
b. Hat der Hund die Katze gejagt?
[Has the dog chased the cat?]

The position of the verb varies depending on the type of the sentence: In a main clause or in a wh-question the inflected form of the verb always occurs as the second sentence constituent, a pattern often referred to as V-second. Subordinate clauses typically have the inflected part of the verb at their end (V-end)[23], and yes-no-questions as well as imperatives usually display them as their first constituent (V-first).

The structural differences between types of sentences, as illustrated above in examples (32-34), need to be accounted for. As it would be very unsatisfactory to stipulate three different sentence rules, thus concealing the intuitively apparent parallels between the three, we want to distinguish one basic word order from which other structures can be derived systematically. The question which of the three different sentence patterns presents this underlying word order has been

[23] As typical for the continental westgermanic languages German displays a so-called brace construction or sentence bracing. So, more precisely, the verb takes the position of the right sentence brace which does not really prevent other constituents from being moved behind the verb. (Besten/Edmondson 1983) Also, in spoken German violations of the verb-end pattern in causative subordinate clauses seem to become more and more acceptable (e.g. Günthner 1992). However, for the present purposes it will suffice to assume that the verb does indeed stand in the end of the sentence.

answered in different ways: In traditional typological classifications German is often found among SVO-languages, that is, those languages with the basic ordering: S(ubject) V(erb) O(bject). The implicit assumption that the independent main clause is to be the normal form seems plausible. However, in 1963 Manfred Bierwisch suggested that it is the structure of subordinate clauses that is most suitable to function as the basic word order. His argument follows a transformational approach, assuming that underlying structures are produced by a relatively small set of base grammar rules and go through a number of transformations to attain their final surface form. The following three criteria are put forward to be considered: the form to be chosen as the basic structure should (i) make the syntactic relations transparent, (ii) represent speakers' intuitions about the connections between the elements, and (iii) allow the necessary transformations to be formulated in the simplest possible way. (Bierwisch 1963:34) To justify the idea of German as a SOV-language, first hinted at by Fourquet (1957/8), Bierwisch shows how the order displayed by subordinate clauses meets these conditions:

First, the use of V-end constructions like (35a) (instead of (35b)) for isolated verb phrases as, for example, in a lexicon entry reflects the intuitive conception of the basic order:

(35) a. jemandem etwas geben
somebody-dat something-acc give
[to give something to somebody]

(35) b. *geben jemandem etwas

Secondly, the subordinate clause leaves the verbal complex as one contiguous unit while in other possible orderings the finite form of the composite verb is separated from the main verb that bears the semantic content. Also, having all the arguments of the verb en block allows our intuitions about some arguments being more closely connected to the verb than others to be paralleled by syntactic nearness in the hierarchy of constituents.

The third aspect we will not have to pursue in this work, since transformations as such have turned out dispensable and, as described in the previous chapters, are replaced in many modern theories by mechanisms to show the underlying structure and the surface within one tree. However the argument of simplicity still applies and the following chapters will show how a strictly right-branching construction of sentences together with the use of predicate functors facilitates a very elegant treatment not only of syntactic but also of semantic structure.

Bierwisch's proposal, corroborated by independent findings of Emmon Bach (1962), is now widely accepted (though not unanimously[24]) and will be adopted in this work. Having established V-end as the underlying structure of German sentences we are still facing a great deal of variety licensed by another source of word order 'confusion': while the position of the verb depends on whether the sentence is a question or a statement, independent or embedded, the position of the other parts

[24] See for example Kohrt (1978). On the other hand, Grewendorf et al. (1987:219f) point out that there is also external evidence coming from results about children's acquisition of German, especially (Clahsen 1982).

of speech can vary to express topicalization or stressing. The following list of sentences reflects all the combinatory possibilities for a main sentence with a three-place predicate[25]:

(36) a. Der Ritter gibt dem Jungen den Schlüssel.
the-nom knight-nom gives the-dat boy-dat the- acc key- acc
[The knight gives the boy the key.]

b. Dem Jungen gibt der Ritter den Schlüssel.
c. Den Schlüssel gibt der Ritter dem Jungen.
d. Der Ritter gibt den Schlüssel dem Jungen.
e. Dem Jungen gibt den Schlüssel der Ritter.
f. Den Schlüssel gibt dem Jungen der Ritter.

The sentences given in these examples are by no means equally acceptable or suitable to answer a general question. Some of them seem to be a more 'normal' way of saying it then others. In the tradition of the Prague school of linguistics such normal, or default forms are often called unmarked, as opposed to marked forms whose deviance from the normal form has to be specified or marked. Some of the sentences sound odd, unless they go with some sort of contrastive emphasis or a context creating a particular topicalization. For example, the structures (36b) and (36c) seem more acceptable in the contexts given in (37) and (38):

(37) Die Puppen sind für die Mädchen.
[The dolls are for the girls.
Dem Jungen gibt der Vater den Werkzeugkasten.
the-dat boy-dat gives the-nom father-nom the-acc tool kit-acc
To the boy the father gives the tool kit.]

(38) Was machen wir mit dem Schlüssel?
[What shall we do with the key?
Den Schlüssel gibt der Vater dem Jungen.
the-akk key-akk gives the-nom father-nom the-dat boy-dat
The father gives the key to the boy.]

Certainly acceptability of marked combinations depends to a certain degree on their success in communicating a proposition unambiguously. Where case markers are equivocal, as feminine and neuter articles in German can be, or in the absence of case markers, as in proper names, there is a high preference to use unmarked word order. The following examples show possible ambiguities:

(39) a. (..., daß) die Katze das Mäuschen jagt.
(... that) the-nom cat-nom the-acc mouse-acc chases
[(…that) the cat chases the mouse.]

---

25 Egli and Egli (1991:106) use a similar set of sentences. I have changed all constituents into masculine words to ensure unambiguous case marking.

(39) b. (..., daß) die Katze das Mäuschen jagt.
(... that) the-acc cat-acc the-nom mouse-nom chases
[(…that) the mouse chases the cat.]

(40) a. (..., daß) Jerry Tom jagt.
[(…that) Jerry chases Tom.]

b. ?(..., daß) Jerry Tom jagt.
[(…that) Tom chases Jerry.]

Concerning example (40) it seems that there is only the reading (40a), and no emphasis could contrive to give a different interpretation to the sentence, whereas example (39) does allow for both interpretations a) and b), and would even more so if there were no semantic hints to favour one or the other possibility, since in this case the fact that mice do not hunt cats gives a strong bias towards version a). Also, as Bierwisch points out, definiteness or indefiniteness as well as pronominalization of constituents may play a part in the acceptability of certain sentence patterns (Bierwisch 1963:32). In addition, the length of constituents often determines the order of their occurrence.

The discussion so far should have made clear that word order in German involves many different factors and is a rather intricate problem to handle. For some of the sentences listed above in figure (37) it might indeed be difficult to imagine a possible situation for them to be uttered in, and some may even exceed the bounds of grammaticality[26]; however, it doesn't matter to us at the moment which combinations are or are not grammatical and whether there is one singularly appropriate candidate for the underlying order - the point to be noted is that there is considerable but not complete freedom in German word order and any attempt to describe German sentences needs to account for this fact.

So far we have concentrated on sentence patterns that are derived by moving around whole constituents, or in other words, deleting a constituent in one position and inserting the same constituent somewhere else. This kind of operation is often called permutation. The following sentences obviously involve a somewhat different derivational process:

(41) a. (…, daß) die Damen den Ritter bewundern.
(..., that) the-nom ladies-nom the-acc knight-acc admire
[(… that) the ladies admire the knight.]

b. (…, daß) der Ritter von den Damen bewundert wird.
(..., that) the-nom knight-nom by the-dat ladies-dat admired is
[(… that) the knight is admired by the ladies.]

The examples show that the subject and object of a verb do not only change their order but also appear in different cases after passivation. The former object takes on the nominative and the former subject turns from an NP[nom] into a NP[von+dat]. However, it would be deceptive to localize the manifestations of voice in the noun

[26] Egli and Egli (1991:106) judge that all of them are acceptable.

phrases. David Dowty (1982:90) points out that “it is not the status of a noun phrase which is changed by a relation-changing rule, but rather it is only the verb or verb phrase which undergoes any change.” What is called ‘relation-change’ by Dowty, corresponds to the notion of ‘recategorization’ in Egli's terminology. The concept behind this notion is that of a verb being shifted from one subcategory to another, in the example above the VP[a]nom,acc[o] ‘admire’ becomes a VP[a]nom,von+dat[o], or, in other words, it is recategorized as a VP[a]nom,von+dat[o].

The difference between changes brought about by permutations and those involving recategorization of the verb, is reflected in the use of two completely different sorts of syntactic mechanisms in their analysis. The technique dealing with permutations is the SLASH-mechanism that has been explained in section 2.3.2. The topic of recategorization as manipulation of case frames will be addressed in 4.3 and, very briefly, in 9.2. Recategorization is indeed to be counted as one of the central ideas of the Syntax of Construction which will be introduced in the following chapter.

4. Syntax of Construction

4.1 From Case Frames to Construction Frames

We have seen how the theory of valence in combination with the idea of complex categories allows a decomposition of the category verb into the main syntactic category and its case frame, the list of arguments required. Following Egli again (Egli and Egli 1991:87), we will now extend the notion of a case frame to the more general notion of a construction frame. Since in the course of this work we will come to interpret case as a semantic entity, a construction frame may not only contain the cases of nominal complements but also the corresponding qualities of prepositional, adverbial or sentential arguments.

Reviving a notion used in traditional grammar Egli and Egli (1991) introduce the term ‘Satzgruppe’ to stand for all major constituents except the predicate, or in other words all possible arguments of the verb regardless of their syntactic category or classification as obligatory or optional.[27] We will use the word ‘sentence prop’[28] in exactly the sense of Egli's ‘Satzgruppe’ to carry the generalization implied by the idea of construction frames. (42) repeats an example given by Egli (Egli and Egli 1991:89) that shows how a German verb may indeed be subcategorized for other than just nominal arguments, in this case, a prepositional object:

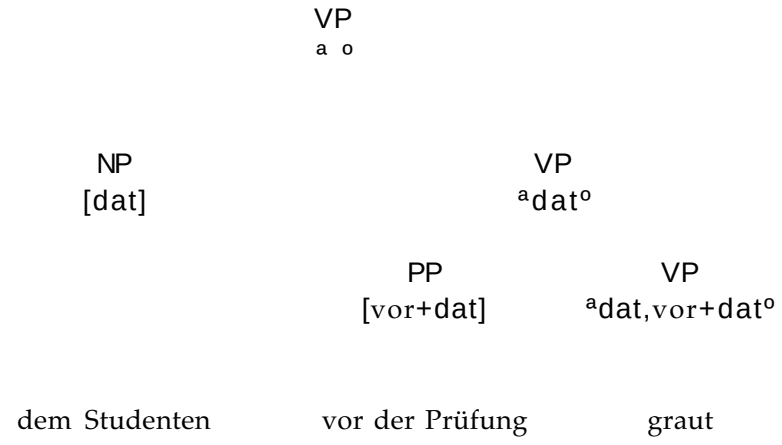
---

[27] Similarly, Bierwisch (1963:61,103) subsumes nounphrases and adverbials under the notion of ‘Satzglied’.

[28] The image behind this name can be thought of as the predicate forming the roof of the sentence under which the arguments, supporting the roof, combine to construct the whole edifice. Just as in a dipteral architecture some pillars are necessary while others may be merely ornamental, the word ‘sentence prop’ is meant to include both, necessary and optional arguments. (Alternatively, if we think of a sentence as a one-act play, sentence props may be understood as the things (and persons) involved in the plot.)

(42) a. (…, daß) dem Studenten vor der Prüfung graut.
(… that) the-dat student-dat of the-dat exam dreads
[(… that) the student dreads the exam.]

b.



(Egli and Egli 1991:89)

An example of a different sort is the subcategorization of adverbials. A verb like 'bleiben' ['stay'] strongly implies the question of location and also tends to occur with an adjunct specifying duration. The example in (43a) seems a typical sentence employing 'bleiben' and we could posit the construction frame of the verb as (43b) containing the semantic roles of the adverbs involved as requirements next to the subject case:

(43) a. (…, daß) Siegfried lange dort bleibt.
(… that) Siegfried-nom long there stays
[(… that) Siegfried stays there for long.]

b. bleiben^{a}nom,dur,loc^{o}

To include the possibility of adverbial or preposition phrases being part of a verb's valence, the notion of sentence prop now allows us to add a further generalization to the rule (28c) given in section 2.5.1 (here repeated as (44a). The new rule (44b) does not only cover the combination of a verb phrase with its nominal arguments, but also licences any sentence prop (abbreviated as SP) in the argument position; that is, SP may stand for any phrasal constituent other than a verbal one.

(44) a. VP^{a}C^{o} á NP[c] V^{a}C,c^{o}

b. VP^{a}C^{o} á SP[c] V^{a}C,c^{o}

As explained earlier in 2.5.1, the variables C and c may now be interpreted as simply being a list of construction features and a single construction feature, respectively. The following table lists the categories that SP may stand for and the kinds of construction features [C] that they may carry.

Subjects:

NP[nominative]
Der Ritter reitet zum Schloß.
[**The-nom knight-nom** rides to the castle.]
S[daß+mood], e.g.: S[daß+indicative]
Daß die Nacht naht, treibt ihn an.
[**That the night draws-ind near** urges him (to hurry).]

Objects:

NP[case], e.g.: NP[dat], NP[acc]
Er bringt **der Jungfrau** eilends **die Kräuter**.
he brings **the-dat virgin-dat** hurriedly **the-acc herbs-acc**
[He brings the herbs to the virgin hurriedly.]
PP[preposition+case], e.g.: PP[an+acc]
Die Prinzessin glaubt **an den Zauber**.
[The princess believes **in the-acc magic-acc**.]
S[daß+mood], e.g.: S[daß+subjunctive]
Die alte Hexe befahl, **daß sie um Mitternacht bereitet würden**.
[The old witch commanded **that they be prepared at midnight**.]

Adverbials:

AP[adverbial quality], e.g.: AP[time]
Er hatte sich **früh** aufgemacht.
[He had set out **early**.]
PP[preposition+case], e.g.: PP[vor+dat]
Er wollte **vor Sonnenuntergang** im Schloß sein.
[He wanted to be in the castle **before sunset-dat**.]
S[conjunction+mood], e.g.: S[sobald+subjunctive]
Das Mittel sollte fertig sein, **sobald die Uhr zwölf schlagen würde**.
[The remedy was to be ready **as soon as the clock would strike twelve**.]

Infinitival complement:

VP[zu+infinitive]
Er hoffte, Dulcineas bleiches Gesicht erblühen **zu sehen**.
[He hoped **to see** Dulcinea's pale face blossom up.]
VP[accusative + infinitive (= AcI)]
Er sah **ihre zarte Hand** aus dem Turmfenster **winken**.
he saw **her-acc dainty-acc hand-acc** from the tower's window **waving**.
[He saw her dainty hand waving from the tower's window.]

Later in section 8.3.2 we will see how for a semantic interpretation of the construction frame it seems useful to abandon the roles commonly used in semantic representation ('location', 'direction', 'instrument', 'time', etc.). As adverbs can be substituted by adverbial prepositional phrases, we will define the abstract adverbial qualities in terms of natural language paraphrases as combinations of the form 'preposition+case'.

In the light of construction frames that specify an argument structure going beyond the traditional notion of transitive or intransitive verbs, the concepts of 'complement' vs. 'adjunct' that we adhered to so far seem to get blurred; indeed, the criteria for the distinction of obligatory vs. optional elements have always been problematic. As mentioned before in the chapter on dependence, it has been suggested that three classes of arguments may be distinguished: the obligatory and the optional complements as well as the pure adjuncts. However, the relation

between a verb's valency as a structural property and its semantic roles and selectional restrictions still seems to be a far more complicated problem. In addition to the two different sorts of complements, we also might wish to differentiate between adjuncts that seem to be semantically implied (e.g. the 'where' and 'whence' of the word 'come') and others that are completely arbitrary. Hence the Syntax of Construction as proposed by Egli and Egli dispenses altogether with a syntactic notion of 'obligatory arguments'. All necessary restrictions are to be provided in form of semantic constraints. Likewise the transitive/intransitive distinction must be modelled within the semantics.

4.2 The Concept of Anadicity

Getting rid of the distinction between obligatory and optional arguments means to give up the traditional classification of verbs concerning the number of complements they take. If there is no substantial syntactic difference between complements and adjuncts, and adverbials may be added in any number, we might have to think of our construction frames as being of arbitrary length. In section 2.4 on the theory of valence we introduced the idea of 'monovalent', 'bivalent' or 'trivalent verbs'. In the terminology of formal logic they are often called 'monadic', 'dyadic' and 'triadic predicates', and in analogy to these expressions a verb that has no fixed adicity will be called 'anadic'. The theoretic background for taking construction frames as argument lists of unspecified length will be introduced in 8.1. We will reconsider the concept of anadicity under a semantic point of view.

Practically, verbs are still endowed with some sort of basic construction frame in our system. However, being viewed as anadic, a verb is not confined to the sort of environment specified in its valency. Some of the complements demanded may be omitted in the syntactic realization, and in building up a phrase more arguments can be added to the construction frame. The way this is done, and the benefit of such a view, will only become fully clear in the light of semantic interpretation, which will be discussed in the final part of this paper. One of the crucial techniques, however, for the alteration of construction frames is the technique of 'recategorization', which will be introduced in the following section.

4.3 Variability of Construction Frames

In the last section we have been talking about changing construction frames. Such a technique of changing does not only help to keep the lexicon tidy in that there is only a relatively small number of basic construction frames, but also to keep track of systematic changes of meaning going along with different argument structures. The idea of dealing with the different environments a verb may occur in by changing its valence goes back to Tesnière (1959). David Dowty (1982:90ff) has formulated a set of so-called 'relation-changing rules' to be applied within the framework of Montague Grammar. The approach presented here is the one outlined by Egli and Egli (1991:94ff) which includes elements of both, Tesnière's and Dowty's theories.

A construction frame can be composed of only a finite number of basic elements

(ibid:93). From the table given on page 37 we can summarize them as follows:

(45) The classes of construction features:

- case
- preposition + case
- conjunction + mood of the subordinate clause
- adverbial quality
- zu+infinitive
- AcI

The idea of recategorization is to formulate a finite set of recategorization rules to allow for indefinite, though not unrestricted, changes to the construction frame. The name indicates that a recategorization rule does not apply to the construction frame directly but to the whole complex category and might also affect the main syntactic category. (For example: we might want to use a rule to recategorize an adverbial phrase into a prepositional phrase if the prepositional phrase takes the role of an adverbial in the sentence.) To get an idea of the sort of phenomena to be captured by use of recategorization, a few examples are provided in (46) to (47) to show how a verb can obviously have more than one construction frame:

(46) a. The king today eats roast beef with a fork to please the queen.
b. The king eats roast beef.
c. The king eats.

(47) a. The queen gives a lot of money to the poor.
b. The queen gives to the poor.
c. The queen gives a lot of money.

(48) a. The queen admires the knight.
b. The knight is admired by the queen.

(49) a. He came.
b. The queen saw him come.

The example in (46a) gives us an inkling of how a construction frame might grow to a considerable length as more and more information is packed into a sentence. Pondering on the possibilities for a moment should suffice for it to become clear that there is an infinite number of potential construction frames (taking into account that the same feature might occur more than once!). The technique suggested by Egli has the power of generating infinitely many construction frames to cope with the variety of natural language sentences.

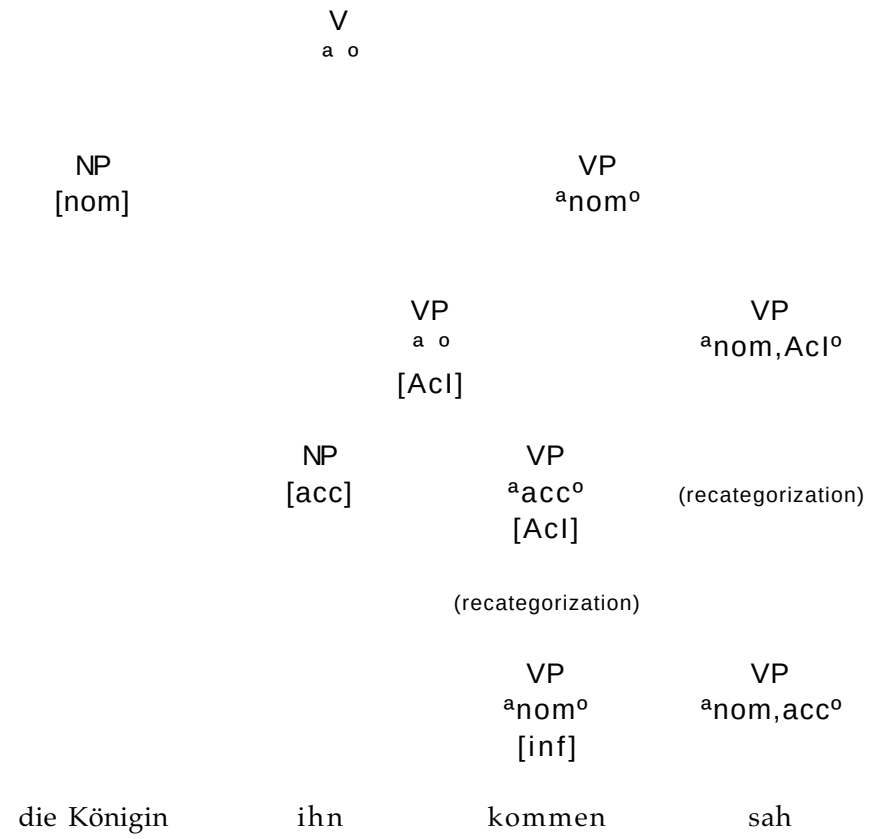
The different construction frames as pertaining to different uses of the same verb are systematically related, just as the meanings of sentences (46b,c), (47b,c) and (48b) are systematically derivable from (46a), (47a) and (48a) respectively. Thus, since the idea of re-categorization is crucially intertwined with semantic interpretation, we will only show one example at this point, to illustrate the syntactic aspect of recategorization. In sentence (49b) both verbs appear in a context different from their basic subcategorization. The example is repeated in German as

(50) below, and (50c)[29] shows the phrase structure tree for (50b):

(50) a. (…, daß) er kam.
[(… that) he came.]

b. (…, daß) die Königin ihn kommen sah.
[(… that) the queen saw him come.]

c.



In the case of the matrix verb 'sah' ['saw'] the recategorization only involves a minor change to the construction frame: the accusative object normally required is replaced by an object sentence of the form commonly called 'accusativus cum infinitivo' (AcI). For the word 'kommen' ['come'] the construction frame is altered to demand an accusative instead of a nominative argument, and at the same time another property of the verb is changed: while in the lexicon the entry for the verbform 'kommen' is marked as [infinitive], its new form is [AcI]. This information will be carried up the tree and indicate the resulting VP as an appropriate filler for the argument place newly created in the construction frame of 'sah'. Formally, the two recategorization rules at work in (50c) can be stated in the familiar format of rewriting rules.

[29] The tree in (50c) is taken from Egli and Egli (1991:105) and slightly modified in the treatment of the matrix verb.

(51) a. $VP^a nom,AcI^o$ á $VP^a nom,acc^o$
b. $VP^a acc^o[AcI]$ á $VP^a nom^o[inf]$

A variety of other recategorizations as well as their semantic implications will be introduced in section 8.3.1. For the moment we only want to emphasize the consequences that the idea of recategorization bears on the concept of construction frames as syntactic units. Egli and Egli (1991:101) summarize this element of the theory of construction in the following way:[30]

> "A verb has one basic case frame[31], a basic valence. All other construction frames are derived by recategorization. This may cause the disappearing of cases, the introduction of new elements or the alteration of the case frame. On the basis of this a complement can be interpreted as part of the basic case frame, while adjuncts can't. A case then, is obligatory in a particular case frame, if it cannot be deleted by any recategorization … ."

This way we are finally in the position to re-introduce the notion of obligatory and optional complements as well as adjuncts into the theory of construction. As will become clear, they are now defined on a much more flexible basis in that the Syntax of Construction does not prescribe any particular argument structure, but leaves it to the semantic component to provide constraints for the wellformedness of construction frames. Throughout this chapter there have been hints referring to the chapter on semantics; this indicates that the Syntax of Construction is indeed designed to be a semantic driven syntax that cannot profitably be taken in its own right. However, as things are going to become more complicated in the following chapters, I thought it useful to introduce some of the more or less syntactic concepts separately before they will be combined with the Semantics of Construction and work in their proper place.

The basic concept of a semantic driven syntax is to tolerate a certain degree of overgeneration of the syntactic rules and make the resulting sentences subject to semantic restrictions. This is why feature based grammars are also often called constraint-based or information-based grammars. In the theory of construction as proposed in this work the syntax component becomes less restrictive (and less decidable) but since construction frames become semantically meaningful entities the semantic component is the part of the grammar where unacceptable sentences may be sorted out.

5. Semantic Interpretation of Natural Language

As has been indicated in the chapters on syntax we are not going to view syntactic structure as an independent system that bears no relation to the question of meaning. To introduce the relevant semantic background and techniques we shall leave syntactic descriptions for a while and take them up again later to show how they combine with a semantic interpretation.

30 Translation: A.M.

31 Although Egli and Egli talk about case frames here, the same is true for the extended class of construction frames of course.

The meaning of sentences in most current approaches to semantics is reduced to what is called their 'truth-conditions'. "Truth-conditional semantics assumes that, in essence, the meaning of a sentence is equivalent to the conditions under which that sentence is true." (Cresswell 1992) This view implies a certain restriction on the subject matter of a semantic theory. Obviously the meaning of imperatives and interrogatives cannot be accounted for in terms of truth or falsehood. Neither can a truth-conditional concept of interpretation do justice to the subtle differences expressible in natural language; distinguishing between sentence meaning and utterance meaning, shades of irony, indications of emotional involvement, metaphorical use of words or allusions to some culturally shared background (e.g. world knowledge and knowledge of stereotypes), are all left aside as belonging to the field of pragmatics. As a reaction to the slightly artificial distinction and the attempts of semanticists to get rid of all cumbersome idiosyncrasies Yehoshua Bar-Hillel coined the expression of "the pragmatic wastebasket" (1971).

However, the tendency to narrow down the field of semantics and accept a limited coverage of meaning in natural language can be justified at least as a temporary simplification that allows us to make use of neat formalizations as a tool for a sound theory of meaning. The requirement formulated by Chomsky for generative grammars and their assignment of structural descriptions remains for any system of semantic description:

> "A generative grammar is a device (or procedure) which assigns structural descriptions to sentences in a perfectly explicit manner [...] A theory of grammar is open for discussion only insofar as it meeets these conditions. To the extent that it fails to provide an explicit characterization of [...] the manner in which grammars assign structural descriptions, it cannot be confronted by evidence, and can receive neither criticism nor support." (Chomsky 1964:995)

The idea of formal semantics, which has become a major branch of modern linguistics today, was not a creation of linguists originally, but has its roots in mathematical logic, where during the first half of our century the semantic structure of artificial languages like the propositional and the predicate calculus were given a precise and consistent formalization. (Lyons 1987:171)

5.1 The Functional View

A crucial principle of truth-conditional semantics goes back to the attempts of Gottfried Wilhelm Leibniz (1646-1716) to construct a universal language as the language of thought and scientific reasoning: his 'calculus ratiocinator' is a calculus for logical inference where propositions are expressed in a structure making use of functions and their arguments. It was the mathematician and logician Gottlob Frege (1848-1925) who introduced this functional structure to natural language sentences and thus laid the foundation of what became a major branch of modern linguistic, the study of logical semantics. Replacing the traditional, syntactically oriented view on sentences as subject, objects and predicate with a notion of functional application he made arithmetical methods and mathematical precision available to the investigation of meaning. In this view, that has now become the common basis to nearly all semantic work, there are two kinds of expressions: closed expressions denoting entities (individuals or truth

values), and open or unsaturated expressions that denote functions. A few simple examples show how natural language verbs express properties or relations with individuals as their arguments:

(52) a. Tristram sleeps.
b. “SLEEP‘(“TRISTRAM‘)

(53) a. Tristram loves Isolt.
b. “LOVE‘(“TRISTRAM‘, “ISOLT‘)

Here it becomes clear that the convention stipulated earlier, to count the subject as part of a verb's valency, despite morpho-syntactic reasons to keep it distinct from the objects (the subject agrees with the verb while objects don't), corresponds to the logico-semantic view where both parts of speech are all the same dealt with as arguments of a predicate.

In (52b) and (53b) the words are printed in capital letters to indicate that they are not the natural language words, but their translations into constants of a semantic representation language. A term of the form “X‘ is to be read as “the meaning of ‘X’,” it stands for the thing in the world that is denoted by ‘X’. We take the meaning of ‘TRISTRAM’ and ‘ISOLT’ to be just the two individuals named Tristram and Isolt. The meaning of ‘SLEEP’ is the function that assigns a truth value to each individual: ‘true’ (or 1) if the individual is in the set of sleepers, and ‘false’ (or 0) if it is not. A truth value may itself be an argument to a function, as for example in (54):

(54) a. Tristram sleeps not.
b. “NOT‘(“SLEEP‘(“TRISTRAM‘))

The meaning of ‘NOT’ is a function that returns the truth value ‘true’ if the value of its argument is ‘false’, and ‘false’ if the argument is true. Functions that take the meaning of one or more propositions (that is, their truth values) as their arguments and return a new truth value, are called truth functions, and the way in which they combine with propositions is defined in statement logic (also called propositional calculus). The system we need to describe the internal structure of a proposition is the logical language of predicate logic. It contains all expressions of statement logic and, in addition, allows us to analyze propositions as composed from predicates and their arguments.

Another important precept of formal semantics is also ascribed to Frege, and often referred to as ‘Frege principle’: the principle of compositionality (also: functionality principle) demands that the meaning of a sentence must be a function of the meanings of its parts and the syntactic relation between them. This includes the necessity of a syntactic analysis prior to the semantic analysis. We will later see how the two are to be combined.

5.2 Quantification

Classical predicate logic includes quantification over individual variables, that is, we can express not only 'Arthur is stupid' as 'STUPID(Arthur)'[32] but, by use of the variable binding universal quantifier ∀, we can also express the quantified proposition 'Everything is stupid' as '∀x STUPID(x)', which can be read as: 'For all x: x is stupid'. Similarly the existence operator or quantifier can be used to formulate propositions such as: 'There is at least one x: such that x is stupid'. Combined with negation, the two operators can also be used to model expressions like 'not all' or 'nothing', but they do not suffice to account for the whole variety of natural language quantifications like 'one', 'a few', 'many', 'most' etc. On the other hand a quantified formula of predicate logic is entirely specific about the skope of each quantifier and thus provides a precise description where natural language tends to be ambiguous. Compare the sentence given in (55), for example:

(55) Everybody loves somebody.

a. for every one there is somebody whom s/he loves
∀x ∃y loves(x,y)

b. there is somebody whom everybody loves
∃y ∀x loves(x,y)

The formulae in this example, however, don't seem to be quite accurate, in that 'everybody' implies something like 'everything that is a person', a restriction that we ignored for the time being, to introduce the notion of skope in its simplest appearance. The following examples show how quantifications can be restricted to a certain subset of the domain:

(56) a. Every man sleeps.
∀x (man(x) μ sleeps(x))

b. A woman sleeps.
∃x (woman(x) ∧ sleeps(x))

c. Every man loves a woman.
∀x (man(x) μ ∃y (woman(y) ∧ loves(x,y)))
∃y (woman(y) ∧ ∀x (man(x) μ loves(x,y)))

Quantification will be addressed again in section 7.2 where Montague's notion of generalized quantifiers is introduced and embedded into the idea of a semantic syntax.

---

[32] We adopt the convenient convention that 'STUPID' and 'Arthur' are constants, interpreted 'stupid' and 'Arthur' respectively.

5.3 Intensionality

The truth or falsity of a sentence, of course, can only be relative to a particular state of affairs. Since in the study of meaning we are not interested in stating what is true or false in the real world, we make use of abstract models to find what the world would have to be like for a given proposition to be true or false. This concept goes back to Tarski (1935, 1954) and allows the recursive formulation of truth conditions for semantic interpretation. A model comprises a set of individuals D (often called the domain of discourse) and a function I that assigns a meaning within this domain to each constant of the language in question. For the examples above: I maps 'TRISTRAM' to an individual in D (e.g. the individual called 'Tristram', provided that it is in D), while I(SLEEP) is some subset of D, namely the set of sleepers in D. To the two-place predicate LOVE the interpretation function I assigns a subset of the Cartesian product D^D, that is, a set of ordered pairs <x,y>, such that x loves y and both are in D.

I have called I a function which assigns a meaning to each expression; more precisely, we are talking about the extension of a constant. The interpretation of 'SLEEP' as the set of sleeping individuals is an extensional characterization of its meaning, one that only specifies the 'extent of sleepyness' in the domain of discourse. Knowing that Tristram and Isolt sleep in Cornwall, however, doesn't enable me to decide whether in some other world, under some other model, the sentence 'Orpheus sleeps' is true or false. To know the truth conditions of 'SLEEP' I have to know something about what it is to be sleeping, so that in any given situation I could determine who of those present is sleeping. Such knowledge might include indicators such as closed eyes, relaxation of consciousness, or what ever else is understood by the word 'sleep'. This level of meaning is called the word's intension. In formal semantics of course, we do not bother about how to identify a sleeping individual, but simply stipulate the intension of the predicate 'SLEEP' as the function that applied to any possible situation returns the set of sleepers in, or in other words, the extension of 'SLEEP' relative to that situation.

To distinguish between extension and intension becomes particularly relevant in certain contexts that we usually call oblique or intensional contexts. Although the Frege principle of compositionality holds for sentences like 'Christopher sleeps', other sentences provide difficulties for a purely extensional view of semantics. Following the very comprehensible summary given in Gallin (1975) I will just briefly let a few examples make the point clear:

1. Example: Modality

'Necessarily the morning star is identical with the evening star'. Given the meaning of 'necessarily' as 'true in all possible worlds' the extension of the sentence is not a function of the extensions of its parts.

2. Example: Compound Phrases

> "Let us suppose that Jones is at this moment a member of the United States Senate, so that the common noun phrase 'colleague of Jones' has the same extension as the phrase 'United States Senator'. The compound phrase 'former colleague of Jones' has as its extension a certain set of

> individuals, among whom is Jones' old law partner, Smith. If we replace the constituent 'colleague of Jones' by the coextensional 'U.S. Senator', however, we obtain the phrase 'former U.S. Senator', the extension of which does not contain Smith. This shows that the extension of 'former colleague of Jones' is not a function merely of the extensions of its parts, so that Frege's principle fails here as well." (Gallin 1975:4)

These problems have been addressed in much of the recent research on semantics. However, since we are not going to deal with any intensional context we will not go deeper into the subject of modal and intensional semantics.

One other obstacle for compositional and model-theoretic semantics is the lexical and structural ambiguity of natural languages. Therefore the interpretation of linguistic words is not done directly, but we use some unambigious intermediate language, a so-called semantic representation language. For example, the English verb 'love' is translated to be the predicate LOVE in our semantic representation language used above[33]. The languages of formal logic lend themselves as very 'well-behaved' representation languages since their properties have been thoroughly investigated. In particular the properties of predicate calculus are very well understood. However, the expressivity of so-called first-order predicate calculus is restricted in so far as no predication may be made about predicates and quantification can only be over individual variables. What this means, and how higher-order logic overcomes this shortcoming will be expounded in the following chapter.

6. The Theory of Types

6.1 First-Order Types

In the chapter on semantic interpretation above we have introduced different kinds of expressions: those denoting entities (individuals or truth values), and others which we called 'unsaturated', since they need to take one or more arguments to obtain the status of a closed term. In the theory of types the distinction between these expressions is made explicit by assigning ontological types to classes of expressions.

The basic atomic types are e (for entity) and t (for truth value). Thus, an expression of the syntactic category proper name, denoting one or more idividuals, can be said to be of type e (the terms 'entity' and 'individual' will be used equivalently). A sentence, denoting a truth value, is of type t. The meaning of a one-place predicate like 'sleep' has been described earlier as a function which when applied to an individual returns a truth value, depending on whether the property denoted by the predicate is true or false for that individual. In other words we could say, a unary predicate is a function that takes an argument of type e and returns a result of type t. The type of such a predicate is therefore $e_{á}t$. The functional application

[33] We will not always explicitly distinguish natural language verbs and logical predicates, but the difference between the two should be kept in mind. (Sometimes we will adopt Montague's notation with a prime, where 'love'' is the predicate denoted by the verb 'love'.)

that allows us to figure out the meaning of a sentence can now be seen as underlying certain restrictions of well-formedness concerning the types involved. The formula in (57) shows how the result of applying a one-place predicate to an individual yields indeed a truth value by simply 'cancelling out' the type of the argument in the type of the function:

(57) $e \acute{a} t\ (e) = t$

Types of the form $a \acute{a} b$ we call functional types, because objects of type $a \acute{a} b$ can be applied to objects of type a and yield an object of type b. In the previous chapter we showed that negation is a truth function which takes a truth value and returns a truth value. Hence its type is $t \acute{a} t$. The examples in (58) show how application of a truth function to an individual or an individual to a predicate causes a clash of types and is therefore semantically ill-formed:

(58) a. $t \acute{a} t\ (e) = ?$
b. $e\ (e \acute{a} t) = ?$

To represent the type of a two-place predicate it is convenient to adopt a technique suggested by Moses Schönfinkel (1924:307f) who conceived polyadic predicates as functional rather than relational. In English it is usually called 'currying', after H. B. Curry who also worked on combinatory logic. Any n-place function may be reduced to a one-place function in the following way: a function $f(x,y)$ can be regarded as a function $f_x(y)$, where f_x is a variable function dependent on x. If we chose f_c to be a function that yields another function when applied to its argument, namely $f_c(x) = f_x$, then $(f_c(x))(y) = f(x,y)$. With the convention that association is to the left we can write:

(59) $f(x,y) = f_c xy$

f_c thus may be called a 'curried' version of f. For a natural language predicate like 'love' the same can be done, as is shown in (60):

(60) love(Tristram,Isolt) = (beloved-of(Tristram))(Isolt)

The unary predicate 'beloved-of' takes 'Tristram' as its argument and gives another unary predicate 'beloved-of-Tristram' which in turn can be applied to the individual 'Isolt' to yield a truth value. Thus the type of 'love' is $e \acute{a} (e \acute{a} t)$ or, since for types association is usually to the right, $e \acute{a} e \acute{a} t$.

The table in (61) summarizes the basic atomic types as well as some of the possible functional types and their realization in natural language:[34]

[34] The literature is full of different notations for types and their combination. The symbols e, t introduced above are those used by Montague, but could be replaced equivalently with ι, o as it is customary in work following the tradition of Russel. Also the notation $a \acute{a} b$ for functional types has been used. In this paper the arrow is always printed at the bottom of the line to avoid confusion with the implication symbol of propositional calculus and the Chomskyan arrow used in rewriting rules. The expressions

(61)

e	individual	e.g. proper name
t	truth value	proposition
$e_{á}t$	one-place predicate	e.g. monovalent verb
$e_{á}e_{á}t$	two-place predicate	e.g. bivalent verb
$e_{á}e_{á}e_{á}t$	three-place predicate	e.g. trivalent verb
$t_{á}t$	one-place truth function	e.g. negation
$t_{á}t_{á}t$	two-place truth function	e.g. conjunction

These type symbols will appear as subscripts on variables and constants to indicate their respective type. In cases where the type is obvious or irrelevant they will usually be omitted. For example, in (62) below α is an object of type $a_{á}b$, that is, it is a function from the domain of entities of type a to the domain of entities of type b, usually represented as $F: D_a á D_b$.

(62) $\alpha_{aább}(\beta_a) = \alpha(\beta)_b$

The objects whose types we have been looking at so far are all objects of ordinary first-order predicate logic. The next section will first give a rough historical overview of the development of type theory and then outline the linguistic reasons to assume higher-order objects.

6.2 Higher-Order Logic

The original reason to think about the ontological type of entities was a contradiction that had emerged in mathematical set theory. In 1901 Bertrand Russell discovered an antinomy in the system of set theory as developed by Georg Cantor and Gottlob Frege that later was to be called ‘Russell's paradox’. The possibility that a set may be an element of another set leads to the following problem: if the set of sets that do not contain themselves did not contain itself as an element, then it would have to be a member of itself; but if it were a member of itself it would contain itself as an element and thus could not be a member. (Kondakow 1975:337) To solve this contradiction Russell introduced a hierarchy of types for logical objects like sets or functions with the stipulation that the elements of a set or the arguments of a function have to be of a lower type than the set or the function itself. In this view, the question of whether a set is a member of itself no longer arises, so neither does the antinomy described above. (Kondakow 1975:489) Russell developed a “doctrine of types” (1903), and finally proposed a “Ramified Theory of Logical Types” (1908) which, incorporated in the ‘Principia Mathematica’ (Whitehead and Russell 1910-13), became very influential. This version was much simplified in later years by different authors (cp. Copi 1971:22) and received its prevailing standard form as “the Simple Theory of

---

$\langle a,b\rangle$, $\langle b,a\rangle$, (ba), (ab), $(a:b)$ found in literature, may all stand for the same functional type $a_{á}b$. However, the 'arrow-notation' seems to provide the clearest way of representing the type structure which will prove advantageous as in the course of this work additional types and rules of type formation will be introduced.

Types" by Alonzo Church (1940). The names 'logic of types', 'higher-order logic' or 'omega-order logic' are used equivalently in the literature on different type theoretical approaches.

The mathematical phenomenon that lead Russell to the assumption of typed objects, can be illustrated by similar problems in natural language. Grelling's paradox of heterological predicates provides a linguistic motivation for the use of a higher-order logic in the analysis of language. Heterological predicates are those predicates that cannot be true for themselves, for example 'German' applied to itself is false, since the word 'German' is not German. Neither is the word 'long' long. As opposed to heterological predicates, autological predicates yield the truth when applied to themselves: 'short' is indeed a short word, and 'English' is English. Now the following paradox arises for heterological predicates:

(63) $\forall F(\mathrm{Het}(F) \leftrightarrow \neg F(F))$ F/Het
$\Rightarrow \mathrm{Het}(\mathrm{Het}) \leftrightarrow \neg \mathrm{Het}(\mathrm{Het})$

We may say that if a predicate is heterological then its application to itself is not true. If this is true for all predicates it must be true for the predicate 'heterological' (Het). However this leads to the obvious contradiction that iff 'heterological' is heterological then it is not true that 'heterological' is heterological. This logical contradiction shows that the predicate 'heterological' is of the wrong type to fill the argument position of 'heterological'. (Egli and Schwertel 1991a:4)

However for an intuitive understanding of how pedicates may be of different types we do not need to employ the intricate notion of heterologicality. There are many other predicates, realized as nouns and verbs in natural language, "which are not properties of individuals, but rather properties of properties of individuals." (Partee et al. 1990:231). The following example makes the idea of higher-order predicates very clear:

> "If this vase is blue and blue is a color, we cannot infer that this vase is a color, but rather that this vase is of a color. The predicate 'is a color' cannot properly be applied to ordinary individuals, but can be applied to properties of them. Properties and relations of these first-order objects are second-order objects, and so on. A logical system is an n-order logic when it contains at least one n-order variable (free or bound)." (ibid.)

In this example 'this vase' denotes an individual of type e and 'blue' a first-order predicate, that is, a predicate applicable to individuals (hence of type $e \rightarrow t$). The predicate '(is a) colour', however, is said to take as its argument a property of individuals, i.e. a first-order predicate. Therefore, since the result of the application is clearly a truth value, the type of 'colour' is $(e \rightarrow t) \rightarrow t$ and it is called a second-order predicate.

Egli (Egli and Schwertel 1991a:3) emphasizes the capacity of higher-order logic to model a theory of grading in natural language. This grading is often realized as nominalization of verbs and can be represented type theoretically as a type change. Consider the examples given in (64):

(64) a. Arthur is bold.

b. Boldness is a virtue.
c. Arthur has the boldness to face the dragon.

Every predicate can be transformed into a form that makes it suitable for the subject position (as 'boldness' in (64b)). Church's formulation of the simple theory of types incorporates certain features of the calculus of ¬-conversion (Church 1940, p. 56). Thus we may use ¬–abstraction to supply a general formulation of nominalization. Let B stand for the first-order predicate 'bold', then the term ¬xBx denotes the set of bold individuals or in other words 'boldness'. The relation between predications like 'Arthur is bold' and 'Arthur has boldness' may then be stated as in (65) (Egli and Schwertel 1991a:3f):

(65) x has ¬xφ(x) iff φ(x)

The small collection of examples given in this chapter may suffice to show how the theory of types provides a valuable descriptive tool for linguistic analysis. The most important work in making a type theoretical approach fruitful in natural language semantics has been done by Rudolf Carnap (1947) and his student Richard Montague. The following chapter will give a brief outline of Montague's approach which will be rather selective and certainly crude in its simplifications; however, since Montague's ideas are too complex to do them justice on a few pages, I do not aim at a comprehensive account, but will concentrate on the aspects of his work that are crucial with respect to the approach presented in this paper.

7. Montague's Approach to Grammar

Around 1970 the philosopher and logician Richard Montague published a number of articles in support of what was his claim in the provocative outset of his paper "Universal Grammar" (1970a:222):

> "There is in my opinion no important theoretical difference between natural languages and the artificial languages of logicians; indeed, I consider it possible to comprehend the syntax and semantics of both kinds of languages within a single natural and mathematically precise theory."

In section 2.3 it has been mentioned that Montague provided a precise syntactic formalization of some substantial fragments of English which in a time when the computational shortcomings of Transformational Grammar had become evident was, as Gazdar observes, a "momentous achievement" (1987:126). It was "generative grammar such as nobody had ever seen before" (ibid.)! Montague's programme, however, comprised much more than the mere description of syntax; his syntax was associated with an explicit semantic theory that allowed the syntactic analysis to be paralleled by compositional semantic interpretation. Thus he proposed a new approach to natural language that formed an intriguing alternative to the pseudo-formality of the transformational framework and has since developed into the foremost paradigm for almost all semantic work, and a major branch within current linguistic research as well as the philosophy of

language. This was possible thanks to an important article of Barbara Partee (1975); since Montague's own papers are formulated mathematically precise but in a very condensed form, it was Partee who first made his ideas accessible to most linguists.

Montague provided a modified categorial grammar for a fragment of English. His aim was to parallel the syntactic rules by some explicit semantic interpretation procedure to allow for a compositional analysis of a sentence's meaning. There are two possibilities to build up a set-theoretic interpretation for natural languages. The first is to parallel each syntactic rule directly with a semantic rule. This was the strategy suggested by Montague in “English as a Formal Language” (1970b) (Knöpfler 1979:2). The second possibility, chosen by Montague in his articles “Universal Grammar” (1970a) (Gallin 1975:v) and “The Proper Treatment of Quantification in Ordinary English” (Bußmann 1990:500), is to translate natural language expressions into a semantic representation language which can then be given a more elegant and accurate interpretation[35]. It has already been mentioned that an important criterion for the choice of a semantic repre-sentation language is the knowledge of its properties. Montague (1970a) introduced an extended version of the theory of types to represent the expressions generated by his grammar. The interpretation of his formal language IL (Intensional Logic) is an extension of Church's functional theory of types extended by intensional, modal and temporal operators.

Montague Semantics is a truth conditional model theoretic semantics. It was Montague who, to provide a sound basis for an intensional semantic interpretation, introduced into linguistic analysis the methods of set-theoretic semantics elaborated by one of his teachers Tarski. The formalized fragments of English were endowed with a possible worlds semantics as used by Carnap and Kripke (Gallin 1975:v). Within this theory it is possible to give an extensional representation to intensional expressions by taking the intension of some α as the function that assigns to every possible world i the extension of α in that world i. The semantic representation language IL satisfies “Frege's functionality principle for intensions - the intension of a compound is a function of the intensions of its constituents - and also the functionality principle for extensions, as amended to take into account oblique contexts” (Gallin 1975:8f). The major innovation is that in IL it is possible to talk explicitly about intensions (functions from possible worlds into extensions) as they have their own type and can be expressed by the intensor ‘å’ (also called ‘cap operator’) which is an abstractor over world (and time) indices (although the index variables are not really visible in IL). For example, in Carnap's system there is no way to make explicit reference to the embedded proposition p in the sentence ‘John believes that p’, whereas in IL it is simply an expression of the type $^{s}\acute{a}^{t}$.

In 1975, Daniel Gallin suggested an amended theory of types to be substituted for Montague's IL that allows us to be even more explicit about possible worlds:

---

35 The two possibilities are equivalent; “As Montague showed, as long as the translation rules are compositional and the semantic interpretation of the intermediate language is compositional, the intermediate language could in principle be eliminated and a compositional model-theoretic semantic interpretation given directly to the source language [...]. But the use of an intermediate language is at least convenient”. (Partee et al. 1990:353)

> "As we observed [...], the cap operator '$\hat{}$' acts as a functional abstractor over indices, although the grammar of IL lacks variables over indices since s alone is not a type. This omission is reasonable, since IL was intended as a formal logic with intensional features close to those of natural language, and in natural language we do not refer explicitly to contexts of use; [...] From a formal point of view, however, it is natural to consider interpreting IL in an extensional theory of types having two sorts of individuals. We call this logic Two-Sorted Type Theory, and denote it by Ty_2." (Gallin 1975:58)

In fact, Gallin's Logic Ty_2 uses three basic types: the two traditional ones, t for truth value and e for entity, and an additional type s of variables and constants denoting possible worlds. (Yet, in the notion of Two-Sorted Type Theory truth values do not count as a sort of types, reflecting their slightly different status (there being only two of them in every model)). In contrast, in IL there is no independent type s, but only a formation rule that allows the construction of intensionalized types, stipulating that if x is a type then $s\hat{}x$ be a type too. Nevertheless, the language of IL is interpretable in Ty_2 (Gallin 1975:61) and the theorem of Zimmermann proves that an analysis given in IL can be formulated equivalently in Ty_2 (Zimmermann 1989).

The treatment of intensionality is indeed a (if not the) central aspect in Montague Grammar and it may sound a strange idea to present Montague's work without addressing the issue of intensional interpretation. However, as the Semantics of Construction that is proposed in the present work shall be introduced in a purely extensional version, it seems reasonable to restrict our attention to those features that have been of some importance in the development of the theory of construction. Although it seems possible to incorporate intensionality into the theory of construction using a version of Ty_2 extended to include anadic predicates and predicate functors as described in the following chapters, I will leave intensional contexts aside to avoid unnecessary complication of the formalism. For a full-fledged semantic system this would be an improper simplification, however, since the time for the present work is too limited to attempt anything but a first sketch of the ideas leading to the Syntax and Semantics of Construction it seems permissable, for the time being, to neglect the question of intensionality. To obtain a semantic representation language suiting our present purposes the extensions mentioned above will be applied to an extensional type logic equivalent to Church's simple theory of types (which we may call Ty_1 to emphasize the fact that it is only meant to be an extensional toy-version of Ty_2). The resulting system L_{cap} will be explained in section 8.4 and defined in a more formal way in appendix A.

The two aspects of Montague's work contributing most to the present approach are the provision of an explicit semantics by building it up in parallel to syntactic structure, and the uniform treatment of noun phrases as sets of properties by a higher-order analysis. These two will be addressed in the following section before we move on to the other 'ingredients' of what is going to be proposed as the 'Semantics of Construction'.

7.1 The Use of a Semantic Representation Language

The basic idea, as described earlier, is to endow syntactic generation rules with a referential semantics, that is, a model-theoretic semantics in the sense of Tarski (1954). Montague achieves this aim for a fragment of English by means of a translation into the symbolism of IL. A translation function can be applied to each expression of English to yield a term of IL which then in turn may be assigned to a domain of possible denotations by the interpretation function I.

The translation of English expressions into IL following strictly formalized rules has to obey three conditions of adequacy:

(a) Correspondence between categories and types:

> "There is a uniform correspondence between the categories of English and the types of IL; that is, if an English expression of category x translates into an IL expression of type a, then all English expressions of category x must translate into expressions of type a."
>
> (Dowty, Wall and Peters 1981:180)

(b) The principle of compositionality:

> "For each syntactic rule of English there is a corresponding translation rule which specifies the translation of the output of the syntactic rule in terms of the translations of the input(s) to it. [...] Thus the correspondence in [(a)] holds for all derived as well as for all basic expressions of English." (Ibid.)

(c) Equivalence to standard formalizations:

> The compositional translation of any expression must yield a result equivalent to a holistic standard formalization of that expression.
>
> (Egli and Schwertel 1991b:3f)

7.2 Generalized Quantifiers

One of the most influential innovations in Montague's semantic approach is his technique to deal with noun phrases in a uniform way (Montague 1973). The traditional formalization of NPs distinguishes between proper names, which denote individuals, and common nouns, denoting properties or relations and combining with quantifiers to form NPs. Consider, for example, the propositions in (66). In a standard semantics where the meaning of 'sleep' is the set of individuals sleeping, the common syntactic structure of the three sentences is hidden because of the different treatment of the proper name.

(66)	a.	Everybody sleeps.	∀x(x é sleep')
	b.	Somebody sleeps.	∃x(x é sleep')
	c.	Arthur sleeps.	Arthur' é sleep'

Only the analysis of sentence (66c) preserves the subject-predicate structure of the original sentence while the other two are translated in a different way. Montague's idea is a type raising that allows us to treat proper names and other

noun phrases in the same way. To achieve this, the denotation of 'Arthur' is no longer viewed as an unanalysed individual, but rather as the set of properties that altogether characterize this particular individual as Arthur. David Lewis (1972) coined the term 'character' for such a set of properties, Dowty, Wall and Peters (1981:220) call them 'individual sublimation'. If we take Arthur as the set of all his properties then the proposition in (66c) can be said to be true if sleeping is among those properties. Now the quantifiers can be seen in the same way: 'everybody' as denoting the set of properties that everybody has ('universal sublimation' (ibid.:221)) and 'somebody' as denoting the set of properties that somebody has ('existential sublimation' (ibid.)). This leads to the semantic analysis shown in (67):

(67) a. Everybody sleeps. sleep' é everybody
b. Somebody sleeps. sleep' é somebody
c. Arthur sleeps. sleep' é Arthur*

While the translation in (66a) says that 'everybody sleeps' is true if all individuals are in the set of sleepers (which defines the property 'sleep'), (67a) puts it this way: 'everybody sleeps' is true if 'sleep' is in the set of properties that everybody has. Montague's *-function that takes an individual and returns a character (a set of properties) can be defined as a ¬-expression in the following way:

(68) * = ¬x¬P(P(x))
Arthur* = ¬P(P(Arthur))

The equal treatment of proper names and other noun phrases is made possible by also raising the type of quantifiers. The ordinary quantifiers of predicate logic, the universal and existential quantifier, are obviously inadequate to express what typically is expressed in natural language quantification. As there are very few things that may be said about all things, usually a quantified expression would be a full noun phrase, "where the noun (CN) and possibly additional relative clauses provide essential restrictions on a quantifier." (Partee et al. 1990:373) For example, a sentence like 'Every child that is under 10 sleeps' is more likely to occur in natural language than an unrestricted universal statement. In (67) above we simply used the semantic representations 'everybody'' and 'somebody'' without defining their actual translation (and, infact, ignoring that they do not really include everything or something but are restricted to refer to certain quantities of people). Montague's solution for the semantic interpretation of noun phrases other than proper names are so-called generalized quantifiers. These are higher-order quantifiers denoting families of subsets of the domain of entities. The determiners 'every' and 'some' are defined as follows:

(69) a. every = ¬Q¬P(∀x(Qx μ Px))
b. some = ¬Q¬P(∃x(Qx ∧ Px))

The Q standing for any common noun, these generalized quantifiers can be applied in a straightforward way to a word like 'man' (interpreted as 'man'') to provide the restriction. For example 'every man' and 'some man/men' are expressed as:

(70) a. every man = $\neg Q \neg P(\forall x(Qx \mu Px))$ (man')
by ¬-conversion:
every man = $\neg P(\forall x(man'(x) \mu Px))$

b. some man = $\neg Q \neg P(\exists x(Qx \wedge Px))$ (man')
= $\neg P(\exists x(man'(x) \wedge Px))$

Intuitively the meaning of 'every man' given in (70a) is the set of properties such that: if some x is a man, then it has got this property, or, in other words, the set of properties which all men have. This interpretation of a quantified noun phrase thus is exactly parallel to what was suggested to be the meaning of 'Arthur*': both are to be understood as sets of properties.

By raising the type of all noun phrases from e (individuals) to $(e_{á}t)_{á}t$ (sets of properties) a new order of application is induced, as shown intuitively in (66) vs. (67). To come to a full semantic translation of our initial sentences given there, the meaning of the subject NP has to be applied to the verb's meaning; Arthur*(sleep') instead of the traditional way: sleep'(Arthur):

(71) a. Every man sleeps. $\neg P(\forall x(man'(x) \mu Px))$ (sleep')
= $\forall x(man'(x) \mu sleep'(x))$

b. Some men sleep. $\neg P(\exists x(man'(x) \wedge Px))$ (sleep')
= $\exists x(man'(x) \wedge sleep'(x))$

c. Arthur sleeps. $\neg P(P(Arthur))$ (sleep')
= sleep'(Arthur)

However, after ¬-conversion, the resulting formulae turn out to have exactly the same form as direct translations of the same sentences into predicate logic. Partee et al. (1990:360) explain:

> "The difference is not in the interpretation but in whether or not that interpretation is arrived at compositionally on the basis of a reasonable natural language syntax. In predicate logic there is no way to view the interpretation of the NP as a constituent; in a more richly typed system there is."

The aim expressed here, that semantic interpretation should be arrived at compositionally, is reflected in a growing tendency to give up on the autonomy of syntax in favour of a semantic driven syntax, which is the idea that also forms the background of this work.

7.3 Semantic Syntax

Facing the obvious interactions between syntactic structure and semantic interpretation Montague developed a system of double-structuring where the syntactic analysis is paralleled on the semantic level. His work started a new direction in linguistics that may be called 'semantic syntax'.

> "There is at present no firm consensus about the form that a generative grammar of English and

> other natural languages should take. But the neatness or goodness of fit between syntax and semantics is generally recognized to be an important criterion in the evaluation of current models of syntax." (Lyons 1987:166)

An important aspect of this idea is implicitly present in the traditional subdivision of nouns into common nouns and proper names. This distinction between the use of common nouns as signifying properties and the individually referring function of proper names has already been named in ancient Greek philosophy of language: 'poiotês' (= property) and 'idiôma' (= idiosyncrasy). The distinction is made explicit in the semantic representation: Syntactic categories are interpreted by semantic categories that reflect their ontological status: common nouns and verbs are interpreted as properties, proper names as individuals. Montague's treatment of noun phrases allows us to represent the semantic difference between the two and yet also preserve the syntactic category of noun phrases as a semantic entity.

The basic idea of semantic syntax is to tolerate a certain degree of overgeneration in the syntax and make the resulting sentences subject to semantic restrictions[36]. However, it is not clear whether a syntax in the form of a categorial grammar, as Montague used in his system, really fulfils the needs imposed by the syntactic variety of natural languages (Abraham 1988:502). An alternative approach is to model the semantic interpretation in parallel to phrase structure rules. This has been done by Gazdar in his Generalized Phrase Structure Grammar (Gazdar et al. 1985). Another attempt to provide a semantic syntax based on a phrase structure grammar is the one suggested by Egli and Egli (1991) which is set out in this paper, and the semantics of which is going to take shape in the following chapters.

8. A Semantics of Construction

The aim of this work is to build up a 'Syntax and Semantics of Construction', a largely semantic driven grammar where verbs are viewed as anadic predicates and thus all verb complements and adjuncts can be handled in the same way as members of a unique class of anadic predicate functors. Typically, in a semantic based grammar the syntax is rather non-restrictive, but semantic categories take the role of constraining possible constructions. In our approach to semantics construction frames become semantically meaningful entities that are to be interpreted in a rather straightforward fashion. In this chapter I will explain the role of construction frames, the semantical impacts of the concept of anadicity and the use of predicate functors.

---

[36] This is why feature-based grammars are often called constraint-based or information-based grammars.

8.1 Anadicity in the Semantics of Construction

In the chapter on syntax it has already been argued that a single verb may have many different construction frames. The examples given in (72) are only a small selection of possible constructions employing the verb 'eat':

(72) a. Arthur eats.
b. Arthur eats apples.
c. Arthur eats apples with appetite.
d. Arthur eats his lunch in the kitchen every day because of the dog.
etc. …

Rather than assuming different predicates 'eat^i' for each arity, a unary predicate 'eat^1', a two-place predicate 'eat^2' etc., to deal with different occurances like those above, in the Semantics of Construction 'eat' is to be treated as anadic predicate, that is, the degree or adicity of 'eat' is not fixed. This not only allows for the intransitive and transitive case, but also for additional adjuncts specifying the circumstances whatsoever. Of course, we do not want every verb to combine freely with every sort of argument. For example, Angelika Kratzer (1988) observed that some predicates in German allow for a spatial or temporal argument while others do not. Distinguishing between stage-level and individual-level predicates she provides the following examples:

(73) stage-level predicate

(…, weil) ihn fast alle Flöhe in diesem Bett gebissen haben.
(… since) him almost all fleas in this bed bitten have
a. [(… since) almost all of the fleas in this bed bit him.]
b. [(… since) almost all the fleas bit him in this bed.]

(74) individual-level predicate

(…, weil) fast alle Schwäne in Australien schwarz sind.
(… since) almost all swans in Australia black are
a. [(… since) almost all swans in Australia are black.]

The German sentence in (73) is ambiguous. The two different readings depend on whether the spatial expression 'in this bed' is analyzed as an adjunct specifying the preceeding noun phrase (leading to the interpretation of the sentence given in (73a)) or as an argument of the verb (leading to (73b)). The idea of anadic predicates allows for the fact illustrated here, that certain verbs may or may not have a spatial argument.

However, for (74) there is no (b)-reading available, since being black is a property inherent in Australian swans that will not change even if they were taken elsewhere. A syntactic analysis of 'in Australia' as argument of the verb phrase 'are black' should therefore be prevented. Since there seems no way to provide a positive syntactic characterization of the construction frame, this approach will rely on semantic restrictions formulated as meaning postulates to distinguish genuine

demands or incompatibilities of some verbs. While there are semantic rules to interpret the omission of certain arguments as existential ellipsis, meaning postulates (of which an example will be given in chapter 10) may specify some constraints as to what a verb can or can not be combined with.

Anadic predicates were first used by Paul Bernays in mathematical logic. Richard Grandy (1976) introduced the notion of anadicity into linguistic context. We will come back to the formal aspects of anadic predicates in the chapter on predicate functors and later in connection with the actual formalization of our logic of constructed anadic predicate functors.

A major disadvantage in Montague's work is his treatment of adverbials. Corresponding to the different classes of adverbials concerning their distribution Montague assigns different types to adverbials that modify sentences (de dicto, type $t_{á}t$), transitive or intransitive verbs (de re, type $(e_{á}e_{á}t)_{á}(e_{á}e_{á}t)$ or $(e_{á}t)_{á}(e_{á}t)$). With the concept of anadic predicates we are now able to extend Montague's uniform treatment of proper names and quantifiers to include adverbials. The de dicto / de re distinction is absorbed in the notion of anadicity, and proper names, quantifiers and adverbials are all subsumed under the heading of sentence props.

8.2 Construction Frames as Semantic Entities

As described briefly in 5.1 the meaning of a sentence can be represented in terms of a relation that is specified by the verb and holds between the sentence props. For example the sentence in (75a) may be represented as (75b)

(75) a. Arthur loves Helena.
b. love (Arthur, Helena)

This formalization of meaning is based on a conventional order for the arguments of a predicate. Since the Syntax and Semantics of Construction allow for anadic predicates, a fixed order cannot be stated. Given the anadic predicate 'love', (76) would be uninterpretable in traditional predicate logic:

(76) love (Arthur, Helena, Monday)

As a verb like 'love' may take any number of arguments greater than one, there seems no way to stipulate a fixed role for any particular position, for example, the third argument. The answer to this problem is one of the central points in the Semantics of Construction: Semantic relations are made transparent by the use of construction features as semantic entities. The principles of a semantic syntax as outlined in section 7.3 mark a theory of double-structuring in which the semantic analysis is paralleled on the syntactic level. We can therefore represent a sentence on two levels: (77a) shows the predicate with its n arguments (where the arguments form the semantic counterparts of the sentence props), and (77b) shows the demand of the verb (represented in a construction frame) and its fulfilment by n sentence props that meet this demand.

(77) a. love (Arthur, Helena)
b. <nom,acc> nom acc

To merge these two levels into one we introduce constructed anadic predicates, that is, predicates of unspecified arity taking constructed arguments as the two shown in (78).

(78) love (<Arthur,nom>, <Helena,acc>)

The extension of a constructed anadic predicate is a set of tupels, each consisting of a number of ordered pairs the first element of which is an individual, the second a construction feature. If we recall for a moment the criteria that have been pointed out as essential for choosing the word order of subordinate sentences as the underlying form of all German sentences, it becomes evident that they could easily be adapted to form the conditions for semantic representation: semantic relations should be made transparent and speakers' intuitions about the connections between elements should be reflected. Since the inclusion of construction features provides a very natural way of semantic interpretation, the required transparency and intuitive clarity are achieved in the system of constructed predicates. Egli and Egli summarize the resulting link between the structure of syntactic dependence and the representation of semantic relations as follows:

> "The extension of case frames into construction frames and the recognition that a verb may have different construction frames allows us to represent relations as categories within the conception of constituents and thus integrate them into the theory of constituency."
>
> (Egli and Egli 1991:101)[37]

8.3 Predicate Functor Logic

So far we have specified the properties of our semantic representation language to the effect that it is to be a higher-order language using anadic predicates. We also introduced Montague's technique to operate with characters rather than individuals. The idea of predicate functors, as we will see, is closely connected with this technique of type raising, in that it provides a variable free formulation of individuals as characters.

In choosing an adequate language of representation we are interested not only in mathematical predictability, but also in a formalism that allows us to model natural language in a reasonably natural way, that is, we want the semantic representation to be as close as possible to the surface structure. (Naturalness, however, is to be viewed as a structural feature and does not necessarily stand for readability!) The aspect of naturalness is best illustrated by examples. The first one (Knöpfler 1979:2)[38] is to examine the role of variables:

(79) If the rose adorns itself, it also adorns the garden.

---

37 Translation A.M.

38 Translation A. M.

In (80) a standard formalization of a sentence like (79) is shown, where 'the garden' in which the rose blooms is represented as a one-place function G.

(80) $\forall x(Rx \mu (A(x,x) \mu A(x, G(x))))$

This expression contains a variable x that does not directly correspond to any part of the original sentence (Knöpfler 1979:2). However, in the predicate calculus, variables are, as Grandy observes, "useful devices" (1976:395) that serve important purposes: a variable not only indicates to which of the arguments of a predicate the quantification applies, but also serves to indicate cross-references and express semantic relations like reflexivity or symmetry. Obviously, all three of these functions play a crucial role for the interpretation of our sentence given in (79) above. Thus, if we want to dispense with variables to achieve a more natural representation, we have to see that these functions are taken care of otherwise. Another natural language phenomenon which is not represented adequately in the usual predicate calculus formalization is the active/passive distinction.

(81) a. Darby loves Joan.
b. Joan is loved by Darby.

Both sentences, (81a) and (81b), are assigned the same semantic representation 'loves (Darby, Joan)' from which the voice of the original sentence cannot be retrieved again. As we do not always consider the question of voice irrelevant (e.g. in connection with theme-rheme structure or quantification[39]) we might want our formalism of semantic representation to be capable of expressing the difference between active and passive. (Knöpfler 1979:3)

These problems in the semantic description of natural language form the background of attempts to provide a representation language with predicate functors. The formal representation of predicate functors builds on the idea of combinatory logic, a mathematical theory founded by Schönfinkel (1924) and Curry in the 1920s and developed further by Church (1941) (who also introduced the term 'lambda-calculus'). However, while combinatory logic only operates upon first-order calculus, predicate functor logic is designed to be compatible with a type theoretical system. A combinator as used by Schönfinkel takes a function and returns a new function, whereas a predicate functor takes a predicate and returns a new predicate (predicate = function into truth values). Like combinators predicate functors are lambda-definable.

It was the philosopher Quine who first suggested a predicate functor logic (1971). His system used n–place predicates and therefore ordinary predicate functors of n^n types (taking an i–place predicate to return a j–place predicate). As mentioned in 8.1 Richard Grandy used anadic predicates, and he also introduced anadic predicate functors thus reducing the number of types necessary for predicate functors to one. Before I will set out Egli's system of constructed anadic predicates and constructed anadic predicate functors, I will try to illustrate the use of predicate functors for natural language semantics in more detail by

[39] Compare for example the following two sentences:
a. Everybody in the department speaks two languages.
b. Two languages are spoken by everybody in the department.

following the description given by Siegfried Knöpfler (1979).

8.3.1 Predicate Functors for Natural Language Analysis

Basic functors

To introduce the basic functors defined by Knöpfler (1979:5f) we will use as examples the intransitive, that is, one-place predicate 'smile' and the two-place predicate 'hate', abbreviated by s and h respectively:

- Placing the negation functor N in front of our example predicates yields Ns, the one-place predicate meaning 'smile not', and Nh, the two-place predicate meaning 'hate not'.
- The identity functor I allows us to identify variables. In the case of h the application of I causes reflexivation; Ih is the one-place predicate 'hate oneself', whereas we don't want Is to be a well-formed expression as the corresponding phrase in English (*'smile oneself') is ungrammatical. In fact, I is not merely a functor for reflexivation, but also can be used to express cross references[40].
- A reversal functor R provides a means to represent, for example, passivization by reversal of the arguments: Rh is the two-place predicate with the meaning 'be hated', while Rs again is not well-formed.
- Eh is the one-place predicate 'hate somebody', and Es the 0-place predicate 'somebody smiles'; that is, the predicate functor E binds one argument place existentially. We will thus call it the existence functor E.
- Adopting Montague's treatment of proper names, individuals are no longer considered to be of the type on which a predicate may be applied, but are raised to take themselves predicates as their arguments. They are dealt with as predicate functors called individual functors. To indicate their status as predicate functors we will represent them between obliques; then /Jack/h is the one-place predicate 'hate Jack' and /Jack/s is the 0-place predicate 'Jack smiles'.
- The fusion functor F simply combines the argument frames of two predicates. At first, this seems a bit confusing and violates our intuitive notion of what is a predicate, but it may well prove a useful technique in the course of further analysis. Combining our two sample predicates 'hate' and 'smile' by Fhs yields a sort of 3-place predicate, namely, one with the subject and object of 'hate' as well as the subject of 'smile' vacant.

  /Peter/Fhs then is a two-place predicate meaning something like 'hate and Peter smiles',

  /Jack//Peter/Fhs is the one-place predicate 'hate Jack and Peter smiles',

  /Jill//Jack//Peter/Fhs is the proposition 'Jill hates Jack and Peter smiles'
- The last functor to introduce, though not directly connected to any particular linguistic operation, is a versatile device to manipulate argument frames: the permutation functor P induces the argument places to rotate, such that the list of arguments is changed from $(a_1, a_2, \ldots a_{n-1}, a_n)$ into $(a_n, a_1, a_2, \ldots a_{n-2}, a_{n-1})$.

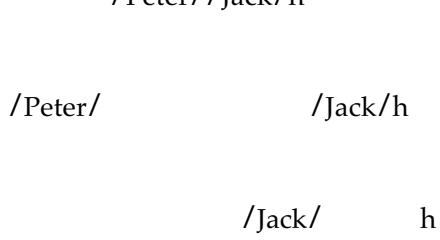
  For example:

[40] For this reason my notation deviates from Knöpfler's where this functor is called R (for 'reflexivation'). Other functors have been renamed as well. To avoid confusion I will introduce all of Knöpfler's system in a notation parallel to the one that will be used later in the formulation of anadic predicate functors.

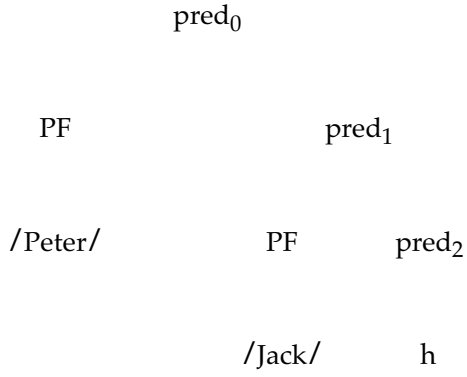
/Peter//Jack/h = 'Peter hates Jack'
/Peter//Jack/Ph = /Jack//Peter/h = 'Jack hates Peter'
/Jill//Jack//Peter/Fhs = 'Jill hates Jack and Peter smiles'
/Jill//Jack//Peter/PFhs = 'Peter hates Jill and Jack smiles'

To understand the structure of a term in predicate functor logic it is often helpful to look at it in a tree diagram. Our example 'Peter hates Jack' can be shown as in (82a) or in a categorial tree as in (82b) (both after Knöpfler 1979:5):

(82) a.



(82) b.



We have now seen intuitively how predicate functors can be used in the translation of natural language expressions into a semantic representation language. I will not introduce the formal language of predicate functor logic as formulated by Knöpfler (1979:3ff), but rather move on to the final step, from anadic predicate functors to constructed anadic predicate functors, which will be taken in the next chapter.

8.3.2 Predicate Functors in the Semantics of Construction

The Syntax of Construction can be viewed as a semantic syntax, where the meaning of a verb is an anadic predicate and the sentence props correspond to predicate functors. As we have seen in the previous chapter, a predicate functor assigns a new anadic predicate to every anadic predicate. We want to specify a higher order predicate functor logic that works on constructed anadic predicates, so as to cover not only the fact that a verb may occur in instances of different degree but to provide an account for the distribution of case as well.
In the example below, the direct object 'einen Apfel' ['an apple'] combines with the

binary predicate 'essen' ['eat'] to form the unary predicate 'einen Apfel essen' ['eat an apple'], or in other words the predicate functor 'einen Apfel' takes a two-place predicate and returns a one-place predicate:

(83) einen Apfel essen á einen Apfel essen
ªnomº acc ªnom, accº

Since all predicate functors are anadic, that is, not specified as to what arity their arguments must have, the same predicate functor 'einen Apfel' may take a three-place predicate and return a two-place predicate:

(84) einen Apfel geben á einen Apfel geben
ªnom, datº [acc] ªnom, dat, accº

Similarly, a dative noun phrase like 'dem Kind' ['(to) the child'] could act as a predicate functor to yield the one-place predicate 'dem Kind einen Apfel geben' ['give an apple to the child'], etc. In section 2.5.1 (28) we have already formulated the general syntactic rule that allows for this unspecified sort of functors. It is repeated here as (85):

(85) VP á NP VP
ªCº [c] ªC,cº

Of course, the construction frames of the verb phrases could well be extended to contain more information, like agreement features, the distinction between pronouns and reflexive pronouns, etc. For the sake of clarity however, we will try to keep the formalism free from those features which are unnecessary to show the principles of construction.

The reason to introduce predicate functors rather than applying anadic predicates to individuals is the same that caused Montague to assume characters of individuals; however, as mentioned in 8.1, he did not make the full possible use of type raising. In the Theory of Construction, as proposed here, Montague's method to unify quantifiers and proper names into one unique semantical class is to be generalized in that we deal in the same way with adverbials and prepositional phrases. The following examples in (86) show that often they can be substituted for each other, and (87) gives an example of how their role in a predicate can be represented exactly parallel to the representation of the direct object as a predicate functor as shown in (83) and (84) above:

(86) tomorrow after one day
here in the city
in into the house

(87) walk into the house á into the house walk
ªnomº [direction] ªnom, directionº

Adverbials as sentence props have been introduced earlier in section 4.1; we can see now that the notion of sentence props is founded on the semantic generalization of anadic predicate functors. The rule given in 4.1 as (44b) is repeated below as (88):

(88) $VP^{a}C^{o}$ á $SP[c]$ $V^{a}C,c^{o}$

We want to formulate the semantical counterpart of this rule, where construction frames contain not merely cases but also adverbial qualities. However, as mentioned briefly in 4.1, we want to dispense with the traditional semantic roles like 'location', 'direction' etc., and define them by preposition+case pairs. The reason for this becomes clear if we consider the following instances of adverbial sentence props. Example (43) given in 4.1 is here repeated as (89).

(89) a. (…, daß) Siegfried lange dort bleibt.
(… that) Siegfried-nom long there stays
[(… that) Siegfried stays there for long.]

b. $\text{bleiben}^{a}\text{nom,dur,loc}^{o}$

The semantic structure in (90a) below is the same as in (89a), but none of the words expressing the 'where' and 'how long' in this sentence could be thought of as containing the features 'duration' or 'location' as part of their meaning in their respective entries in the lexicon. The construction frame for 'bleiben' as derived from this example would thus look a bit different; it is shown in (90b):

(90) a. (…, daß) Siegfried über die Festage im Schloß bleibt.
(… that) Siegfried-nom over the-acc holidays-acc in-the-dat castle-dat stays
[(… that) Siegfried stays in the castle over the holidays.]

b. $\text{bleiben}^{a}\text{nom,über+acc,in+dat}^{o}$

To represent the abstract adverbial qualities in terms of natural language paraphrases would allow for the obviously possible substitution of adverbials and prepositional phrases and thus simplify the semantics. In the section on meaning postulates we will give a few examples of how adverbial qualities are internally translated into prepositional expressions.

Before a possible formalization of the Semantics of Construction is outlined in the following chapter, a reminder may be in place that, being restricted to a purely extensional view, the present system will not be suitable to analyse intensional expressions. In the context of a theory of adverbials this means that intensional adverbs like 'sogenannt' ['so-called'] cannot be considered within the given framework of a one-sorted logic.

8.4 The Logic of Constructed Anadic Predicate Functors

The theory of types that has been expounded in chapter 6 is going to form the logical basis for the formalization of the Semantics of Construction. In this chapter I will introduce some extensions to traditional systems of type logic to incorporate anadicity and construction into a higher-order logic with predicate functors. The resulting system shall be called L_{cap}, 'logic of constructed anadic predicate functors'.

8.4.1 The Non-Standard Types of L_{cap}

In traditional type theory we have only one formation process. (91) shows the rule for the formation of functional types with the operator '$_{á}$' (in a Smullyan-like format, where the top line states one or more conditions and the bottom line gives a conclusion). 'T' standing for the set of types, the rule reads: 'if a and b are types, then $a_{á}b$ is a type'.

(91)
$$\frac{a, b \in T}{a_{á}b \in T}$$

In the familiar format of types we usually deal with atomic types or functional types as defined in (91). For the logic L_{cap} the set of types will be extended to contain lists of types to model the linguistic notion of a subcategorization list and allow for the fact that predicates may combine with an unbounded number of complements. If a predicate is viewed as taking a list of arguments to return a truth value then we don't have to specify the actual number of the arguments, thus preserving the desired generality. If we want to add the possibility of having lists of elements of a certain type, we have to define a list forming operator. List processing is a vital feature of high-level programming languages like Prolog or Lisp, so it is not surprising that a type constructor 'list' has been suggested by the computer scientists Dale Miller and Gopalan Nadathur (1986a:250; 1986b:450).

As ordered pairs form the basis for an inductive definition of n-tuples, lists (= n–tuples) can be represented easily as nested pairs. In most programming languages the list (a, b, c, d) would be formulated as a nested structure of (dotted[41]) pairs, where 'nil' marks the end of the list:

(92) (a. (b. (c. (d. nil))))

Making use of the notion of "head" and "tail" we can now split the list into its two surface elements: If we take the first element a and assign a variable x to the rest of the list (or second element of the pair) (b. (c. (d. nil))), then the whole object can be represented as a dotted pair:

(93) (a. x)

[41] Where the dot is a binary functor to construct a new object from two given objects.

For our purposes it is more convenient to introduce a somewhat different list structure accessible from both sides by breaking the structure up into (94a) or (94b):

(94) a. (head; rest$_i$)
b. (rest$_j$; tail)

This might be thought of as a two-ended stack that can be 'pushed' or 'popped' at both sides. However, the items pushed or popped don't have to be atomic elements. The semicolon serves as a list forming (infix-) operator concatenating two lists, each of them possibly empty (signified by nil) or containing only one element. The table in (95) gives some examples of possible lists:

(95) a
nil
a;b
a;nil
(a;b);(c;d) = a;b;c;d

This conception of a list, as a completely unstructured object, allows for a simplification of list manipulating operations (as will be defined later on in chapter 9) and otherwise for a treatment of lists exactly like tupels. In particular relational expressions may still be transformed into Schönfinkel style functional application, e.g.: $f(a;b;c) = f(a)(b)(c)$. We can now stipulate an additional type formation rule to introduce the new type 'list(a)' for arbitrary lists of elements of type a:

(96) $$\frac{a \text{ é } T}{\text{list}(a) \text{ é } T}$$

The classical formation rule for well-formed terms in the theory of types defines the application of functional types:

(97) $$\frac{\beta \text{ é } ME_a,\ \alpha \text{ é } ME_{a \acute{a} b}}{\alpha(\beta) \text{ é } ME_b}$$

In words: if β is a meaningful expression of type a and α one of type $a \acute{a} b$, then the result of applying α to β is an expression of type b or in a Chomskian rewriting format:

(98) $\alpha(\beta)_b$ á $\alpha_{a \acute{a} b}$ (β_a)

For the newly introduced type list(a) we need three additional term formation rules, to license lists as those given as examples in (95):

(99) a. $$\frac{\alpha \text{ é } ME_a}{\alpha \text{ é } ME_{list(a)}}$$

b. $$\frac{\alpha = nil}{\alpha \text{ é } ME_{list(a)}}$$

c. $$\frac{\beta \text{ é } ME_{list(a)},\ \alpha \text{ é } ME_{list(a)}}{\alpha;\beta \text{ é } ME_{list(a)}}$$

These rules now enable us to build up lists of arbitrary length. For anadic predicates, i.e. predicates taking an arbitrary number of arguments, we can thus assume the type $list(e)_á t$. In traditional type formation we already have a binary structure and a sort of working down a list (cf. the type of a three place predicate, e.g. a standard reading of 'give': $e_á(e_á(e_á t))$ which by iterated functional application can be reduced to two and one place predicates: 'give to somebody' $e_á(e_á t)$, 'give something to somebody' $e_á t$); For anadic predicates, however, we arrange the arguments into one single object, an unstructured list: The type of 'give', $e;e;e_á t$, then can be generalized as $list(e)_á t$ with an unspecified number of arguments (which in the case of a sentence (a sentence being viewed as a VP with empty valence) might be naught). Hence, for examples (72a) to (72d) given in 8.1 (here repeated as (100)) we assume that 'eat' is of the type $list(e)_á t$.

(100) a. Arthur eats.
b. Arthur eats apples.
c. Arthur eats apples with appetite.
d. Arthur eats his lunch in the kitchen every day because of the dog.

An anadic predicate functor that takes an anadic predicate as its argument and returns an anadic predicate again is represented as:

(101) $(list(e)_á t)\ á\ (list(e)_á t)$

We can now see that this way of assuming anadic predicates with unbounded argument lists has the advantage that all predicates are of the same type. And also, all sentence props, semantically interpreted as predicate functors, are of the same type.

Traditional type logic, as we have seen in chapter 6, knows only two basic types, e for entities and t for truth values. In addition to these atomic types we introduce another basic type con, which is the domain of construction features as we have specified them in (45), section 4.3. To represent complex categories we finally have to create the possibility to build up Cartesian types:

(102) $$\frac{\alpha \text{ é } ME_a,\ \beta \text{ é } ME_b}{<\alpha,\beta> \text{ é } ME_{a\hat{}b}}$$

For example, if Arthur is an object of type e and the nominative an object of type con, then <Arthur,nom> is of type e^con. Table (103) gives a summary of the types that our system makes crucial use of:

(103) Table of types:

e = individual
t = truth value
con = construction feature
e^con = individual with its construction feature
list(e^con) = tupel of individuals with their construction features
$e\hat{}con_{á}t$ = one-place predicate[42]
$e\hat{}con_{á}e\hat{}con_{á}t$ = two-place predicate
$e\hat{}con_{á}e\hat{}con_{á}e\hat{}con_{á}t$ = three-place predicate
$list(e\hat{}con)_{á}t$ = anadic predicate
$list(list(e\hat{}con)_{á}t)$ = tuple of anadic predicates
$(list(e\hat{}con)_{á}t)_{á}(list(e\hat{}con)_{á}t)$ = anadic predicate functor

For convenience I will usually omit the type index in the following exposition. However, explicit mention will be made of an expression's type where it seems of particular relevance. In the next chapter I will now introduce how natural language (German) words can be translated into the semantic translation language L_{cap}.

9. The Syntax and Semantics of Construction

The aim of this work is to build up a ‘Syntax and Semantics of Construction’, that is, a largely semantic driven phrase structure syntax and a system of semantic interpretation that is built up to go in parallel with the syntactic rules. Verbs are viewed as constructed anadic predicates and thus all verb complements and adjuncts can be handled in the same way as members of a unique class of anadic predicate functors. So far, the syntactic techniques and the semantic representation language have been presented separately; in this chapter I will show how the two work together.

As we have already seen in chapter 7, the principle method of a semantic syntax using an intermediate semantic representation language is to give a set of rules to translate from the language of the fragment into the semantic representation language. To ensure compositionality these translation rules work in parallel to syntactic structure rules, and the resulting terms of the representation language

[42] As anadic logic is not intended to be an alternative to classical type theory but rather an extension, the predicates of fixed adicity remain in the system but will not be used.

are given a set-theoretical interpretation that follows the principle of compositionality by a recursive definition of meaning. In our case, we will have to define how expressions of German are translated into the representation language L_{cap}. The semantic interpretation of L_{cap} is given formally in appendix A; but first we shall have a look at how the translation is done.

9.1 Translation of German Words into L_{cap}

If the meaning of a sentence should be deductable from the meaning of its constituents, then it seems reasonable to start the translation into our semantic representation language with the basic constituents, the words given in the lexicon. To match the format of lexical entries that has been suggested in section 2.5.2, we think of a word's translation into L_{cap} as one of its features. Each complex category is endowed, therefore, with an attribute 'sem' (for 'semantics'), that may take any expression of L_{cap} as its value. This does not mean, of course, that the translation of each lexical word is completely idiosyncratic; the attribute 'sem' could as well be seen as a translation function that assigns an expression of L_{cap} to every syntactic element, depending on its syntactic category and structure.

This section will give a detailed account of how each basic category is assigned an L_{cap} formula as its semantic representation. Table (120), at the end, summarizes all the translations introduced.

Common Nouns, Adjectives and Verbs:

Common nouns, adjectives and verbs are all translated into anadic predicates of L_{cap}, signified by priming the lexical form. For example the common noun 'Mann' ['man'], the adjective 'schön' ['beautyful'], the intransitive verb 'schlafen' ['sleep'], the transitive verb 'lieben' ['love'] or the bitransitive verb 'geben' ['give']:

(104)
Mann(sem) = Mann(lex) = Mann'
schön(sem) = schön(lex) = schön'
schlafen(sem) = schlafen(lex) = schlafen'
lieben(sem) = lieben(lex) = lieben'
geben(sem) = geben(lex) = geben'

Each of these lines is to be understood as a path-equation to constrain a word's complex category with regard to its feature 'sem', specifying its translation into the semantic representation language L_{cap}. We will see shortly how these translations work together to build up semantic representations of sentences.

Proper Names:

Proper names are syntactically noun phrases and, thus, candidates for a position as an argument of the verb, or, using the terminology introduced in section 4.1, as a sentence prop. Semantically all sentence props correspond to and are translated as

anadic predicate functors. For example, the expression ‘Paul’, denoting an individual can be seen as of type e, or of type $(list(e)_{á}t)_{á}(list(e)_{á}t)$ when used as a predicate functor that takes a predicate like ‘$love_{list(e)_{á}t}$’. This type raising is exactly what is done by Montague's *-function (extensionalized) where a* = ¬F (Fa). Montague turns individuals (e.g.: a) into characters, that is, the set of their properties (e.g.: ¬F (Fa) = a*). However, for the system with constructed anadic predicates as suggested in this work, we have to introduce a non-standard type change function //, that takes into account the fact that in German a proper name never comes without a case (though usually not realized morphologically). In (105) the path description ‘Paul(con)’ stands for the construction feature (con) of ‘Paul’:

(105) Paul(sem) = /<Paul,Paul(con)>/

Generally, the semantic representation of a proper name x appearing in the case c is thus /<x,c>/ and it is defined as a lambda term in the following way:

(106)
$$\begin{aligned} //\ // &=_{def} \neg\langle y,d\rangle\neg P\neg L(P(L;\langle y,d\rangle)) \\ /\langle x,c\rangle/ &= \neg\langle y,d\rangle\neg P\neg L(P(L;\langle y,d\rangle))\ (\langle x,c\rangle) \\ &= \neg P\neg L(P(L;\langle x,c\rangle)) \end{aligned}$$

// is the function that maps the set of proper names into the set of type theoretical expressions of type $(list(e\hat{}con)_{á}t)_{á}(list(e\hat{}con)_{á}t)$, that is, it translates proper names as constructed predicate functors. The resulting translation of ‘Paul’ into L_{cap} is shown in (107):

(107)
$$\begin{aligned} Paul(sem) &= /\langle Paul,Paul(con)\rangle/ \\ &= \neg\langle x,c\rangle\neg P\neg L(P(L;\langle x,c\rangle))\ (\langle Paul,Paul(con)\rangle) \\ &= \neg P\neg L(P(L;\langle Paul,Paul(con)\rangle)) \\ &\text{e.g.: } \neg P\neg L(P(L;\langle Paul,nom\rangle)) \end{aligned}$$

Intuitively, the ¬-expression in (107) represents the set of properties that Paul has as well as the relations that hold between him (in the nominative) and any set of (other) individuals. Such sets of other participants are encoded in the variable L which is of type list(e^con), that is, a list of constructed individuals (ordered pairs of individuals and construction qualities). The variable P stands for an anadic predicate. Strictly speaking // is just an abbreviation for the lambda-expression it is defined by, to make the formulas look a bit tidier.

We have now defined the first simple predicate functors. The predicates these functors work upon are formalized as constructed anadic predicates as shown in (104) above and are of the type $list(e\hat{}con)_{á}t$. So, we can have a first go applying a predicate functor to an anadic predicate. In 8.4.1 it was mentioned that the formation rules for well-formed terms, given in a Smullyan table format there, can equivalently be formulated as rewriting rules to parallel the format of a

phrase structure grammar as used in the Syntax of Construction. The type theoretical formation rule for functional application was given in such a format (98), here repeated as (108)):

(108) $\alpha(\beta)_b$ á $\alpha_{a\acute{a}b}$ (β_a)

Instead of looking at $\alpha(\beta)$ as a functional application, Partee et al. take it as "the result of concatenating α, (, β and) in that order" (1990:320). This view makes it very straightforward to draw the parallel to natural language syntax and emphasize the compositionality of interpretation.

(109) S á NP VP
«
$\alpha(\beta)_b$ á $\alpha_{a\acute{a}b}$ (β_a)

We can use this rule for our first example. (110b) repeats the semantic translations of the lexical constituents needed for the sentence in (110a). (110c) gives the syntactic rule that licences (110a) and which is exactly the one in (109). Thus we can also apply the semantic rule (110d) corresponding to it. This is done in (110e):

(110) a. Paul schläft.
[Paul sleeps.]

b. schlafen(sem) = schlafen'
Paul(sem) = /<Paul,nom>/ = ¬P¬L(P(L;<Paul,nom>))

c. S á NP VP

d. $\alpha(\beta)_{list(e\hat{}con)\acute{a}t}$ á $\alpha_{(list(e\hat{}con)\acute{a}t)\acute{a}(list(e\hat{}con)\acute{a}t)}$ $(\beta_{list(e\hat{}con)\acute{a}t})$

e. /<Paul,nom>/(schlafen')
= ¬P¬L(P(L;<Paul,nom>))(schlafen')
= ¬L(schlafen'(L;<Paul,nom>))

The resulting formula obviously is not a closed formula and not of the type truth value, as one would expect the translation of a proposition to be. This is the advantage of anadicity: The semantic represetation reflects the fact that in natural language a sentence like 'Paul sleeps' is still open to additional information ('in the bed', 'since 9 o'clock', etc.) as long as it is not properly brought to a close by a full stop or the appropriate intonational signals. So, ¬L… may be interpreted as the set of circumstances that accompany Paul's sleeping. There will be examples below to show how the argument frame of a predicate can be filled by more participants, necessary or optional ones, and how to finish the sentence off, when no more arguments are expected. However, first we want some more lexical material.

Quantifiers

The idea of generalized quantifiers has been introduced in section 7.2. To fit Montague's technique into our system we have to amend the formulation of generalized quantifiers to go with constructed anadic predicates. (111) shows again ‘every’ and ‘some’, as defined in (69), together with an example of their application to a common noun:

(111) a. every $=_{def}$ $\neg Q \neg P(\forall x(Qx \mu Px))$
b. some $=_{def}$ $\neg Q \neg P(\exists x(Qx \wedge Px))$
c. every man = $\neg P(\forall x(man'(x) \mu Px))$

The same is shown in (112) for quantifiers over anadic predicates, where each of the two predicates involved (the presupposed one that forms the restriction (here: man'), and the one to be attributed to the class of entities denoted by the generalized quantification) is provided with a variable L to indicate possible extensions of the argument frame.

(112) a. every $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;x) \mu P(L_P;x)))$
b. some $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;x) \wedge P(L_P;x)))$
c. every man = $\neg L_Q \neg P \neg L_P(\forall x(man'(L_Q;x) \mu P(L_P;x)))$
d. every man = $\neg L_Q \neg P \neg L_P(\forall x(man'(L_Q;x) \mu P(L_P;x)))$ (nil)
= $\neg P \neg L_P(\forall x(man'(x) \mu P(L_P;x)))$

The formula in (112c) shows that the semantic rule corresponding to the phrase structure rule NP á Det N cannot be a simple application rule, like the one that was to parallel S á NP VP in (109) above. We want the application of a determiner to complete the NP, to the effect that noun expanding rules cannot be productive anymore; for example we want to prevent an analysis of ‘the brother of Paul’ as $[_{NP}[_{NP}$the brother$][_{PP}$of Paul$]]$ instead of $[_{NP}[_{Det}$the$][_{N}[_{N}$brother$][_{PP}$of Paul$]]]$. However, what the $\neg L_Q$ in (112c) does, is exactly what we would like to prohibit, it allows for the common noun ‘man’ to be supplemented. To avoid this, the semantics of noun phrases of the Det N pattern is: NP(sem) = Det(sem) (N(sem)) (nil). The final application to nil leads to the translation given in (112d).

The last step is to amend generalized quantifiers to cope with constructed predicates and predicate functors. As this is the point where case comes in, things are getting a little bit more difficult. The German quantifiers corresponding to ‘every’ and ‘some’ (‘jed-’, ‘ein-’) show a rich case morphology. (113) gives the formalization of some quantifiers for anadic predicates with construction features[43]:

(113) a. jeder(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,nom>)))$
jeden(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,acc>)))$
jedem(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,dat>)))$
jedes(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,gen>)))$
(113) b. ein(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,nom>)))$
einen(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,acc>)))$

[43] To avoid cumbersome notations, like ‘jede/r/s’ etc., all determiners are given in their masculine form, which seems most convenient since the masculine paradigm is unambiguous.

einem(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,dat>)))$
eines(sem) $=_{def}$ $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,gen>)))$

The case assumed for the first half of the formula to be build up is nominative, in all cases, because the restrictive clause that is formed by application of the quantifier to a common noun Q^1 is always of the form $Q^1(x)$, that is, 'x is Q^1', where x must stand in the nominative. The second case, in contrast, varies with the case of the quantifier and is adapted thus to the prospective position of the quantified expression in the sentence turning on the predicate P.

Our treatment of quantification as outlined above is, of course, much simplified. We ignore the fact that natural language quantifiers are not as easy as just $\exists$ and $\forall$. (For example, the German words 'ein' ['a', 'one'] and 'einige' ['some'] actually are very rarely used in the sense of $\exists$ ('at least one')). As another simplification, we will not deal with definite NPs[44], but restrict ourselves to the quantifiers defined so far.

Adverbials

Adverbials, as we have seen before, can be treated in the same way as other sentence props since they take the same position within a sentence (cp. 'Paul drove the car', 'Paul drove to London', 'Paul drove away') and are viewed as arguments of the predicate much the same as other complements or adjuncts. Semantically they are, of course, predicate functors. Thus, we can see now how the raising of proper names and generalized quantifiers allows for a uniform treatment of all sentence props as predicate functors. The translation of adverbials into L_{cap} is achieved in a way parallel to the translation of proper names. The definition of the function // is here repeated as (114), and (115) gives a few examples ('gestern' ['yesterday'], 'drüben' ['(over) there']):

(114) // $=_{def}$ $\neg<x,c>\neg P \neg L(P(L;<x,c>))$

(115) gestern(sem) = /<gestern,time>/ = $\neg P \neg L(P(L;<gestern,time>))$
drüben(sem) = /<drüben,loc>/ = $\neg P \neg L(P(L;<drüben,loc>))$

If we think of 'immer' ['always'] or 'überall' ['everywhere'] as standing for 'jeder-zeit' ['every time'] and 'jeder Ort' ['every location'], respectively, their representation in the following way seems very straightforward:

(116) immer(sem) = $\neg P \neg L Œt(P(L;<t,time>))$
überall(sem) = $\neg P \neg L Œz(P(L;<z,loc>))$

In chapter 10, where we will have a brief look at the possibilities of meaning postulates, it will be argued that for a more substantial theory it would simplify the semantics to replace adverbial qualities such as time, location, etc. by

[44] For the treatment of the definite article cp. Egli and v. Heusinger 1991. In their system the definite article 'der' is translated into the semantic representation /der/ $=_{def}$ ´, which is interpreted as the set theoretic choice function.

prepostion+case pairs of natural language paraphrases, defined in implicatures like 'if x is on <y,dat>, then it is in a certain location'. However, what I wanted to show is how adverbials can be treated as predicate functors, in a way exactly parallel to noun phrases (quantifier and proper names) and prepositional phrases; otherwise I will not go any deeper into the semantics of adverbials.

Prepositions

We have not talked about prepositions so far, although it has been claimed on several occasions that prepositional phrases belong to the class of sentence props, like noun phrases and adverbial phrases, and thus correspond to predicate functors semantically. The reason to put prepositions at the end of our 'semantic lexicon' is that although the syntactic concept of preposition+case pairs as construction features is quite simple, the formal realization looks rather complicated. (117b) shows how we want the semantic representation of a sentence containing a prepositional adjunct to look like:

(117) a. Arthur schläft auf einer Wolke.
[Arthur sleeps on a cloud.]

b. $\exists x(\text{Wolke}'(<x,\text{nom}>) \wedge \text{schlafen}'(<\text{Arthur},\text{nom}>;<x,\text{auf+dat}>))$

The difficulty is that the construction feature of x in the argument frame of 'schlafen' (auf+dat) is built up by combining the construction features of the preposition (auf) and of the noun phrase (dat). To achieve this by means of lambda-conversion it would be possible to use Tarskian abstractions over sequences like <x,dat>, or to define variables of the form <x,dat> where 'dat' is a restriction on the sort of possible substitutes. The solution I choose instead is in effect very similar to the use of sequence variables (as our system contains abstraction over lists); it is, however, a more computational formulation, which makes it intuitively more intelligible than Taski's cylindric algebra (Henkin, Monk and Tarski 1972).

To make single elements embedded in lists of ordered pairs, accessible, we define indexed position functions: The position function POS_n is a function that takes a list as its argument and returns the n^{th} element of this list, counting from the back, that is, the index specifies the position of the chosen element relative to the end of the list. (For a greater variety of applications directed functions $POS>_n$ and $POS<_n$ could be defined, counting from the beginning or the end of a list, respectively.) Correspondingly, the position function $POS_{n.m}$ returns the m^{th} element (again counting backwards) of the list or pair that has been chosen by the function POS_n. As the structures we typically will have to break up are always lists of ordered pairs, this somewhat ad hoc definition will suffice for our purposes and there is no need for a recursively applicable function that digs up elements from any depth of embedding. (118) gives some examples of how the function works on lists of ordered pairs:

(118) $POS_4(<w,a>;<x,b>;<y,c>;<z,d>) = <w,a>$
$POS_{1.1}(<y,c>;<z,d>) = d$

$POS_{3.2}(<w,a>;<x,b>;<y,c>;<z,d>) = x$
$POS_{2.1}(<x,b>;<y,c>;<z,d>) = c$

With the set of position functions $POS_{m.n}$ ($m, n \geq 1$) we have now a tool that enables us to pick out construction features and combine them in a new way. (119) shows how this is done in the case of a prepositional phrase like 'auf einer Wolke' ['on a cloud']45:

(119) auf einer Wolke

einer Wolke (sem) = $\neg P_1 \neg L_1(\exists x(Wolke'(<x,nom>) \wedge P_1(L_1;<x,dat>)))$
auf (sem) = $\neg P_2 \neg L_2 \neg L_3(P_2(L_2;<POS_{1.2}(L_3),auf+POS_{1.1}(L_3)>))$

auf einer Wolke
PP(sem) = $\neg Q \neg L(NP(sem)\ (Prep(sem)(Q)(L)))$
= $\neg Q \neg L(NP(sem)\ (\underline{\neg P_2} \neg L_2 \neg L_3(P_2(L_2;<POS_{1.2}(L_3),auf+POS_{1.1}(L_3)>))[Q]\ (L)))$
= $\neg Q \neg L(NP(sem)\ (\underline{\neg L_2} \neg L_3(Q(L_2;<POS_{1.2}(L_3),auf+POS_{1.1}(L_3)>))[L]))$
= $\neg Q \neg L(NP(sem)\ (\neg L_3(Q(L;<POS_{1.2}(L_3),auf+POS_{1.1}(L_3)>))))$
= $\neg Q \neg L(\underline{\neg P_1} \neg L_1(\exists x(Wolke'(<x,nom>) \wedge P_1(L_1;<x,dat>)))$
$[\neg L_3(Q(L;<POS_{1.2}(L_3),auf+POS_{1.1}(L_3)>))])$
= $\neg Q \neg L(\neg L_1(\exists x(Wolke'(<x,nom>) \wedge \underline{\neg L_3}(Q(L;<POS_{1.2}(L_3),auf+POS_{1.1}(L_3)>))$
$[L_1;<x,dat>])))$
= $\neg Q \neg L(\neg L_1(\exists x(Wolke'(<x,nom>) \wedge$
$Q(L;<\underline{POS_{1.2}}[L_1;<x,dat>],auf+POS_{1.1}(L_1;<x,dat>)>))))$
= $\neg Q \neg L(\neg L_1(\exists x(Wolke'(<x,nom>) \wedge Q(L;<x,auf+\underline{POS_{1.1}}[L_1;<x,dat>]>))))$
= $\neg Q \neg L(\neg L_1(\exists x(Wolke'(<x,nom>) \wedge Q(L;<x,auf+dat>))))$
= $\neg Q \neg L(\exists x(Wolke'(<x,nom>) \wedge Q(L;<x,auf+dat>)))$

Note that the resulting expression has the same type again as its constituent 'einer Wolke', so that both can play the same role of a predicate functor within a sentence.

The technique of position functions as introduced here may also be used to formalize relative pronouns; however, due to the limitations of time I cannot address the matter of relative sentences in this work. I will conclude this chapter by a summary of the translations into L_{cap} assigned to by our semantic lexicon of German, as developed so far (The value of a path X(lex) is the lexical form of the word X, e.g. Paul(lex) = Paul):

45 For all derivations we introduce the convention to underline the function or lambda-expression that is to be applied next, and put the term it is applied to into square brackets.

(120)

Common Nouns:
CN(sem) = CN(lex)'
Adjectives:
Adj(sem) = Adj(lex)'
Verbs:
V(sem) = V(lex)'
Proper Names:
PN(sem) = /<PN(lex),PN(con)>/ = $\neg P \neg L(P(L;<PN(lex),PN(con)>))$
Quantifiers:
jeder(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,nom>)))$
jeden(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,acc>)))$
jedem(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,dat>)))$
jedes(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,gen>)))$
ein(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,nom>)))$
einen(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,acc>)))$
einem(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,dat>)))$
eines(sem) $=_{def} \neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,gen>)))$
etc.
Adverbials:
Adv(sem) = /<Adv(lex),Adv(con)>/ = $\neg P \neg L(P(L;<Adv(lex),Adv(con)>))$
Quantified Adverbials:
immer(sem) = $\neg P \neg L \Œt(P(L;<t,time>))$
überall(sem) = $\neg P \neg L \Œz(P(L;<z,loc>))$
etc.
Prepositions:
Prep(sem) = $\neg P_2 \neg L_2 \neg L_3(P_2(L_2;<POS_{1.2}(L_3),Prep(lex)+POS_{1.1}(L_3)>))$

A more comprehensive lexicon is given in Appendix B, where a syntactic and semantic description of some lexical items is given in the attribute-value format described in 2.5.2. There I will assume seperate entries for a common noun or proper name in each case, even if the case is not morphologically present in the morphology of the word. This leads to the problem that for example a phrase like 'eines Ritters Schwert' ['a knight's sword'], where 'Schwert' can be of three different cases, may be built up as an NP of a different case than the one required by the verb or preposition it is to be combined with later. Such inefficient parsing may make a lot of back tracking necessary. To avoid this problem the construction feature of a word like 'Schwert' could be given as the set of possible cases, e.g. 'Schwert (con) = {nom, acc, dat}', from which by unification with the governor's demand the appropriate one would be chosen. This method would even allow us to have case ambiguities represented within one semantic representation: A sentence like (121a) with the two readings (121b) and (121c) depending on whether 'Maria' is taken as accusative or dative would be given the semantic representation (121d), reflecting the ambiguity:

(121) a. Otto steht auf Maria.
b. [Otto stands on Maria.]
c. [Otto fancies Maria.]
d. stehen'(<Otto,nom>;<Maria,{auf+acc,auf+dat}>)

However, the whole system of feature unification would become rather complicated with this method. Therefore, the grammar given in the appendix dispenses with sets of possible cases and uses the more verbous and less elegant method of stating separate entries for each form.

I have already mentioned a few syntactic rules and how they correspond to semantic operations. The following chapter will show how phrases and sentences are given a compositional translation, that is, a translation that can be derived from the translations of their parts and their syntactic relation.

9.2 Rules for a Compositional Translation

In section 8.3.1 I introduced Knöpfler's system of predicate functors in the analysis of natural language sentences. The idea is that certain recurring operations can be defined as anadic predicate functors allowing us to manipulate construction frames. I will define only two predicate functors here, but it seems important to note the expressive potential in the use of such combinators. For the formulation of the first semantic rule corresponding to the syntactic rule N á Adj N we will define the anadic conjunction functor C[46] that is rather different from Knöpfler's fusion functor F in that it does not combine two construction frames, but identifies them. Intuitively this yields exactly what we think the meaning of a noun with an adjective should be, for example: 'valiant knight' is true for one person if this person is valiant and is a knight[47]. In (122) we have the syntactic rule for the combination of adjectives with nouns and its semantic interpretation. The rule is only an abbreviated version, leaving out all syntactic conditions, which are given in more detail in the grammar of appendix B.

(122) N á Adj N
m(sem) = C d_1(sem) d_2(sem)

As described in 2.5.2, 'm' stands for the mother node and 'd_i' for its daughters. The predicate functor C, used to combine the two properties denoted by the adjective and the noun, is defined as in (123):

(123) $C = \neg P_1 P_2 \neg L(P_1(L) \wedge P_2(L))$

[46] A different fond is used for anadic predicate functors to distinguish them from Knöpfler's predicate functors, but the same letters are used where the same sort of function is denoted.

[47] This is only one possible use of adjective noun combinations, namely, the conjunctive use, assuming that a valiant knight is somebody who is valiant and a knight. For a sensible translation of, for example 'good cobbler', a different rule would be required that allows to make the adjective qualify the meaning of the noun (supposing that by 'good cobbler' we usually do not mean a person who is good and a cobbler, but one who is a cobbler and good at the things pertaining to this profession). The semantic representation built up by rule (122) only covers the first way of interpreting adjective noun combinations.

If we apply the rule in (122) to form the expression 'kühner Ritter' ['valiant knight'], the translation into L_{cap} is derived compositionally from the translations of 'kühner' and 'Ritter' by the semantic translation procedure given as the value of the sem-feature of the resulting noun. The derivation is given in (124):

(124) kühner Ritter(sem) = C kühner(sem) Ritter(sem)
$= \underline{\neg P_1} P_2 \neg L(P_1(L) \wedge P_2(L))$ [kühn'] (Ritter')
$= \underline{\neg P_2} \neg L(kühn'(L) \wedge P_2(L))$ [Ritter']
$= \neg L(kühn'(L) \wedge Ritter'(L))$

The rule for noun phrases has been mentioned in section 9.1 (110) before. Below, in (125a), the rule is stated again in our attribut value format. Obeying this rule, the translation of 'kühner Ritter' now combines nicely with the one of 'ein' ['a'] to yield the translation of 'ein kühner Ritter', as given in (125b):

(125) a. N á Det N
m(sem) = d_1(sem) (d_2(sem)) (nil)

b. $\neg P \neg L_P(\exists x(\neg L(kühn'(<x,nom>) \wedge Ritter'(<x,nom>)) \wedge P(L_P;<x,nom>)))$

A rule tor the construction of prepositional phrases has also been given in 9.2, we can thus take the translation of 'auf einer Wolke' ['on a cloud'] derived in (119) to have a look how adverbial phrases can combine with nouns to form descriptions like 'Mann auf einer Wolke'. The rule for this is given in (126) and the derivation according to the rule in (127) (the adverbial phrase in this case is a prepositional phrase which is licensed by the rule AP á PP).

(126) N á N AP
m(sem) = d_2(sem) d_1(sem)

(127) Mann auf einer Wolke(sem) = auf einer Wolke(sem) Mann(sem)
$= \underline{\neg Q} \neg L(\neg L_1(\exists x(Wolke'(<x,nom>) \wedge Q(L;<x,auf+dat>))))$ [Mann']
$= \neg L(\neg L_1(\exists x(Wolke'(<x,nom>) \wedge Mann'(L;<x,auf+dat>))))$

In the translation instruction of (126) we have the case where the second constituent of a compound is applied to the first one to come to the expression in L_{cap}. This seems to be against our claim, made in section 8.3.1, that a semantic representation with predicate functors is not only a compositional but also a very natural representation largely paralleling phrase structure. However, the formula 'd_2(sem) d_1(sem)' could be formulated equivalently as '$\neg Y(Y(d_1(sem)))$ d_2(sem)' or as '$F(d_1(sem)\ d_2(sem))$, with $F = \neg yz\ (z(y))$' to reverse the order of application. But, in fact, both these alernatives would only make the derivations more complicated, and it seems that even under the aspect of naturalness a post positive functor is acceptable.

In much the same way that nouns can be expanded into noun and adverbial, it is possible to have constructions with noun and genitive NP to form a nominal like 'Sohn eines Bäckers' ['son of a baker']:

(128) N á N NP
d_2(con) = gen
m(sem) = d_2(sem) d_1(sem)

To account for the similar NP construction 'eines Bäckers Sohn' ['a baker's son'] a more complicated semantic rule is necessary.

(129) N á NP N
d_1(con) = gen
m(sem) = ¬Q¬L_Q¬P¬L_P((Q(L_Q;<y,nom>) ∧
P(L_P;<y,d_2(con)>))) (d_1(sem) (d_2(sem))) (nil)

However, I will not dwell too long on examples of every single rule. There are some derivations given in appendix C. The crucial rules to comment on in some more detail are those of functor application. One of the major achievements in the Theory of Construction as suggested by Egli and Egli is the possibility to treat NPs, PPs and APs all the same as sentence props and represent them uniformly as predicate functors. We have seen in the previous chapter that the semantic representations of expressions as different as 'eine Wolke' ['a cloud'], 'auf einer Wolke' ['on a cloud'] and 'immer' ['always'] are all of the same logical type, the type (list(e^con)$_{á}$t)$_{á}$(list(e^con)$_{á}$t). (130) shows the three of them together to make their structural similarity clear:

(130)
auf einer Wolke(sem) = ¬P¬L(∃x(Wolke'(<x,nom>) ∧ P(L;<x,auf+dat>)))
eine Wolke(sem) = ¬P¬L(∃x(Wolke'(<x,nom>) ∧ P(L;<x,acc>)))
immer(sem) = ¬P¬L(Œt(P(L;<t,time>)))

The type theoretical equivalence of these three reflects what we have observed earlier in 4.1: the fact that noun phrases, prepositional phrases and adverbial phrases may all occur as arguments of the verb. The sentences in (131) should make the point clear:

(131) a. Arthur reitet eine Wolke.
[Arthur rides a cloud.]
b. Arthur reitet auf einer Wolke.
[Arthur rides on a cloud.]
c. Arthur reitet immer.
[Arthur rides always.]
d. Arthur reitet im Rennen immer einen Schimmel.
Arthur rides in-the race always a grey
[Arthur always rides a grey at the races.]

With reference to the fact that all sentence props are represented as predicate

functors semantically, Egli and Egli (1991:125) call the rule stated as (44b) in section 4.1 ‘Funktorabarbeitungsregel’, which we may translate as ‘functor application rule’. Here it is repeated as (132) in the attribute-value format and amended with an instruction for the compositional translation into our semantic representation language L_{cap}:

(132) VP á SP VP
$d_1(con) = d_2(subcat)(first)$
$d_2(gap) = «$
$m(subcat) = d_2(subcat)(rest)$
$m(sem) = d_1(sem)\ d_2(sem)$

The semantics of the functor application rule is very straightforward, and the translation of the sentence in (131b) can be derived in a completely deterministic procedure of repeated lambda-conversions:

(133) (..., daß) Arthur im Rennen immer einen Schimmel reitet.

einen Schimmel reitet (sem) = einen Schimmel (sem) (reitet(sem))
$= \underline{\neg P}\neg L(\exists x(Schimmel'(<x,nom>) \wedge P(L;<x,acc>)))\ [reiten']$
$= \neg L(\exists x(Schimmel'(<x,nom>) \wedge reiten'(L;<x,acc>)))$

immer einen Schimmel reitet (sem)
= immer(sem) (einen Schimmel reitet(sem))
$= \underline{\neg P}\neg L(Œt(P(L;<t,time>)))\ [\neg L_1(\exists x(Schimmel'(<x,nom>) \wedge reiten'(L_1;<x,acc>)))]$
$= \neg L(Œt(\underline{\neg L_1}(\exists x(Schimmel'(<x,nom>) \wedge reiten'(L_1;<x,acc>)))\ [L;<t,time>]))$
$= \neg L(Œt(\exists x(Schimmel'(<x,nom>) \wedge reiten'(L;<t,time>;<x,acc>))))$

in einem Rennen immer einen Schimmel reitet(sem)
= in einem Rennen(sem) (immer einen Schimmel reitet(sem))
$= \underline{\neg P}\neg L(\exists x(Rennen'(<x,nom>) \wedge P(L;<x,in+dat>)))\ [\neg L_1(Œt(\exists y(Schimmel'(<y,nom>) \wedge reiten'(L_1;<t,time>;<y,acc>))))]$
$= \neg L(\exists x(Rennen'(<x,nom>) \wedge \underline{\neg L_1}(Œt(\exists y(Schimmel'(<y,nom>) \wedge reiten'(L_1;<t,time>;<y,acc>))))\ [L;<x,in+dat>]))$
$= \neg L(\exists x(Rennen'(<x,nom>) \wedge Œt(\exists y(Schimmel'(<y,nom>) \wedge reiten'(L;<x,in+dat>;<t,time>;<y,acc>)))))$

Arthur im Rennen immer einen Schimmel reitet(sem)
= Arthur(sem) (im Rennen immer einen Schimmel reitet(sem))
$= \underline{\neg P}\neg L_1(P(L_1;<Arthur,nom>))\ [\neg L(\exists x(Rennen'(<x,nom>) \wedge Œt(\exists y(Schimmel'(<y,nom>) \wedge reiten'(L;<x,in+dat>;<t,time>;<y,acc>)))))]$
$= \neg L_1(\underline{\neg L}(\exists x(Rennen'(<x,nom>) \wedge Œt(\exists y(Schimmel'(<y,nom>) \wedge reiten'(L;<x,in+dat>;<t,time>;<y,acc>)))))\ [L_1;<Arthur,nom>])$
$= \neg L_1(\exists x(Rennen'(<x,nom>) \wedge Œt(\exists y(Schimmel'(<y,nom>) \wedge reiten'(L_1;<Arthur,nom>;<x,in+dat>;<t,time>;<y,acc>)))))$

At the end of this long derivation we still have a function instead of a proposition as the meaning of a sentence should be. However, as mentioned earlier in 9.1 this is

only to allow for more arguments that could possibly be added (e.g. '(..., daß) neuerdings Arthur im Rennen immer einen Schimmel reitet' ['(..., that) recently Arthur always rides a grey at the races.']). If there are no further arguments coming (and the construction frame contains no necessary arguments anymore (see chapter 10 below)), a closure rule (134) can be applied:

(134) VP á VP
d_1(subcat) = (nil)
m(sem) = d_1(sem) (nil)

(135) shows how the derivation in (133) is finally finished by applying the closure rule to its result:

(135)

$\underline{\neg L}_1$(∃x(Rennen'(<x,nom>) ∧ Œt(∃y(Schimmel'(<y,nom>) ∧
reiten'(L_1;<Arthur,nom>;<x,in+dat>;<t,time>;<y,acc>))))) [nil]
= ∃x(Rennen'(<x,nom>) ∧ Œt(∃y(Schimmel'(<y,nom>) ∧
reiten'(<Arthur,nom>;<x,in+dat>;<t,time>;<y,acc>))))

At this juncture we will have to ignore that this analysis assigns a skope to the universal quantifier that doesn't seem to express the intended meaning; it is, however, arguable that problems like this can be dealt with by ways of standard methods of quantification logic. In this work, however, I want to concentrate on the merrits of building up an anadic construction frame compositionally and interpreting it with reference to construction features.

Before this will be discussed in chapter 10, there is another set of functor application rules to be introduced, the rules that deal with gaps, as introduced in section 2.3.2. Anadic construction frames have to be seen as containig a necessary arguments in their unmarked order, and although additional arguments may be pushed in at any point (perhaps restricted by some meta-rules stating the regularities inflicted on the word order for pronominal elements, definite vs. indefinite expressions or constituents of different length), the order of necessary complements can only be varied within the possibilities provided by a set of rules which we will call SLASH-rules.

The technique is to change the surface (syntactic) order of constituents without having any difference in the semantic representation. This is possible by type raising and abstraction over predicate functors. At the introduction of a gap a predicate functor variable is applied to the verb and through abstraction over this variable the information about a gap can be 'remembered' until the missing sentence prop occurs. The rules necessary to deal with argument gaps are given in (136-136) below. The example in (139) shows how each node is licensed by one of the rules (showing the gap in the convenient SLASH-notation used in earlier chapters).

(136) a. SLASH-Introduction

SP á t_{SP}
m(gap) = m
m(sem) = ¬F F

b. VP á SP VP
d_1(con) = d_2(subcat)(first)
m(subcat) = d_2(subcat)(rest)
m(gap) = d_1
m(sem) = d_1(sem) (d_2(sem))

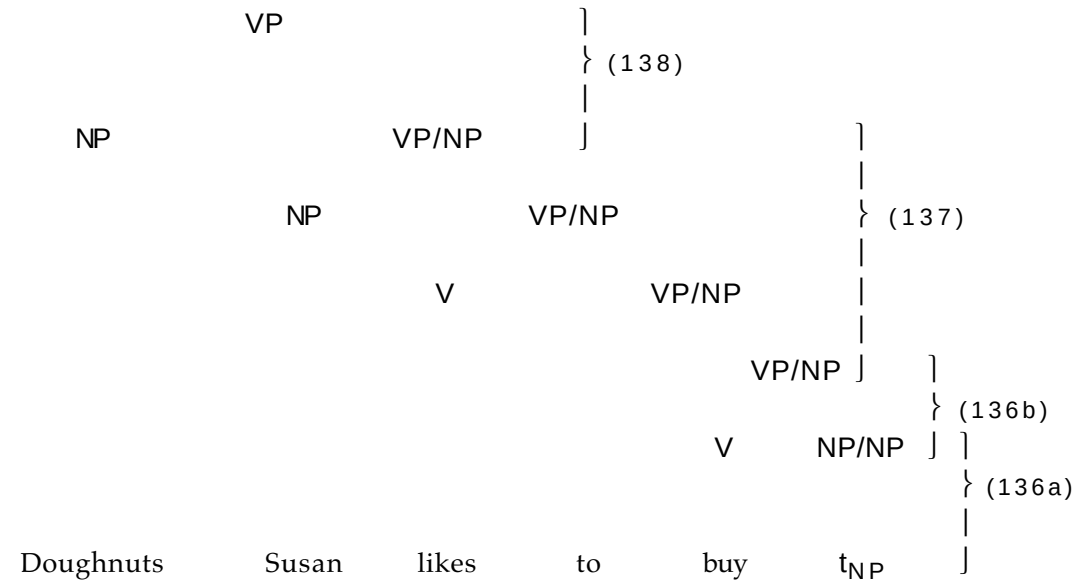
(137) SLASH-Percolation

VP á SP VP
d_1(con) = d_2(subcat)(first)
m(subcat) = d_2(subcat)(rest)
m(gap) = d_2(gap)
m(sem) = ¬F d_1(sem) (d_2(sem)(F))

(138) SLASH-Termination

VP á SP VP
d_1 é d_2(gap)
m(subcat) = d_2(subcat)
m(gap) = d_2(gap) \ d_1
m(sem) = d_2(sem) (d_1(sem))

(139)



The condition m(gap) = d_2(gap) in rule (137) does the work of a foot feature principle as it was mentioned in 2.3.2; it makes sure that the SLASH-feature is carried up the tree. A complete derivation of a sentence analysed as containing a gap is provided as example (2) in appendix C.

It was said above that the construction frame of a verb contains necessary arguments. However, as we have seen in section 4.3, some arguments are classified as 'optional complements', that is, they may be omitted even though they are part of the verb's valence. The example given in (46) is repeated here as (140):

(140) a. The king eats roast beef.
b. The king eats.

Our intuition that if somebody eats, they necessarily eat something, is to be accounted for. In cases like the one in sentence (140) we assume an existential ellipsis which is represented in our formal semantic language by binding the omitted argument existentially. In rule (141) the syntactic recategorization, where the subcategorization frame is cut back by one element, is accompanied by the application of another predicate functor E which is defined in (142):

(141) Existential closure

VP á VP
m(subcat) = d_1(subcat)(rest)
m(sem) = E(d_1(subcat)(first)) (d_1(sem))

(142) E = ¬c¬P¬L∃z(P(L;<z,c>))

The existence functor E is applied to an element of the construction (subcategorization) frame and binds a variable of this construction feature. As I have pointed out on a few occasions, the system proposed in this paper is meant to be a semantic driven syntax. This means that the sentences generated by the syntax are subject to semantic restrictions, formulated in the form of meaning postulates. These meaning postulates are a complicated web of conditions and implicatures that is impossible to outline in any satisfactory way within the limited bounds of this work. However, a few ideas will be put forward in the following chapter to give at least a hint at how a full Theory of Construction could be fashioned.

10. Semantics of Construction as the 'Driving' Component

So far, the description of the Semantics of Construction was mainly concerned with the formal translation into a semantic representation language and very little was said about the interpretation of the resulting intermediate language expressions. Semantic rules for a model theoretic interpretaion of L_{cap}-terms with respect to a Henkin-type model M are given in appendix A. Some more subtle aspects of natural language semantics, however, are much more difficult to get a

grip on. One of these problems, for example, that has received much attention but remains, as yet, unsolved is the definition of selection restrictions to outrule ungrammatical or ontologically deviant sentences. The Syntax and Semantics of Construction has been devised by Egli (1991; personal conversations) to work as a semantic driven grammar. Yet, the problems to design a way of efficient syntax-semantic interaction are so many, that it would go far beyond the original layout of this work to tackle them. Nevertheless, I will briefly summarize some of the thoughts about a possible semantics that are closely connected with the idea of a Theory of Construction.

In chapter 3 we have seen a small selection of possible word order and possible candidates for generalizations. However, the chances of finding a way to characterize all possibilities of word order syntacticly don't seem very encouraging. Egli's approach therefore is to look at the problem in terms of semantic meaningfulness instead of syntactic wellformedness. The idea is to formulate negative constraints in the sense that everything that does not violate them can be accepted as grammatical. The basic concept behind this is to tolerate a certain degree of overgeneration by the grammar and make the resulting sentences subject to semantic restrictions.

One possible technique to state such restrictions could be the definition of predicates that assign a certain semantic subcategory to a verb: just as in set theory certain constraints on relations might be formulated as predicate-predicates, we can use such predicates to express the specific semantic shape of a German verb. For example, the transitivity of the relation ‘greater than’ can be expressed as ‘transitive(>)’ or its property of being a two-place relation as ‘dyadic(>)’. Correspondingly, we want to define predicate-predicates that are true for a particular set of verbs. A verb like ‘love’ that subcategorizes nominative and accusative thus may be described as Nom-Acc(love), where the predicate-predicate Nom-Acc is defined as follows:

(141) $\text{Nom-Acc}(V) =_{def} \forall L(V(L) \mu \exists M,N,P,x,y\ (L=M;<x,nom>;N;<y,acc>;P))$,
where M, N, P are possibly empty lists.

This definition of basic construction frames allows for arbitrarily many adjuncts without giving up the notion of necessary arguments. In section 9.2 we have seen how the permutation of necessary arguments can be dealt with. The structure of construction frames, defined by predicate-predicates, corresponds to the concept of association lists in higher level programming languages. Thus, the possibility of accessing a certain element via its construction feature could be utilized. However, there seem to lie a lot more difficulties in wait than are visable at first sight. For example, the adjunct variables M, N, P need to be restricted to avoid uncontroled insertion of additional accusative arguments or even a second nominative.

Predicate-predicates are a versatile tool in formulating the structural properties of verbs in a precise way. But the crucial objective for any further development in the Theory of Construction would be to define the class of possible construction frames. Also, with predicate-predicates it would be possible to formulate

selectional restrictions in a similar fashion: for example we might say 'liegen' ['lie'] is an 'auf+dat-verb' (formally: auf+dat(liegen)) to posit that it is combinable with auf+dat-adjuncts.

The second component to go with the Syntax of Construction is a set of meaning postulates. The meaning postulates have no part in the generation or analysis of a sentence in the first place. They only will be activated in a separate semantic component to exclude nonsense sentences or to allow for question answering.

With the technique of meaning postulates, adverbials could be defined by paraphrases using prepositions and case. For example <hier,loc> [<here,loc>] could be defined in a more general language as <dieser-Ort,an+dat> [<this-place,at+dat>]. This means that cases are assumed to be genuine semantic entities (which seems to find corroboration in the typological findings of old adverbials being the roots of today's case system). The equivalence of certain prep+case combinations can be stated in implicatures like the following:

(142) a. <Tisch,auf+dat> μ <bestimmter-Ort,an+dat>
[<table,on+dat> μ <specific-place,at+dat>]
b. <Montag,an+dat> μ <ein Zeitpunkt,zu+dat>
[<Monday,on+dat> μ <a time,at+dat>]

In (143) I give a more precise formulation of the general rule to the implicature in (142a) above. It may serve as an example for the sort of rule that could be used to state many other meaning postulates.

(143) $\forall L,M,P,x,y(P(L;<x,\text{auf+dat}>;M) \mu \exists y(P(L;<y,\text{an+dat}>;M) \wedge \text{Ort}(<y,\text{nom}>)))$

Because construction frames are semantically meaningful entities in the Theory of Construction, a close knit network of such postulates could define a major part of the semantics. However, there is no safe method, as yet, to cope with the exeeding ambiguity of almost all preposition+case pairs.

Conclusion

Thinking about summarizing the results of my work, it is the list of issues missing that comes to the mind first: no doubt, a comprehensive theory of type theoretical semantics can not get away with all the simplifications I was forced to impose on the material. Certainly the questions of tense and plurality as well as modal and epistemic contexts cannot be left aside in a substantial theory. And also, I do not think the problems that have been addressed in this work are all solved in a satisfactory way. However, despite of these inadequacies and despite of some formal inaccuracies that I'm aware of, I do think that the steps taken in this paper point into a direction that is worth exploring.

I have argued that existing semantic theories make little use of case information present in languages like German. The relatively free word order, however, suggests that in German this sort of information plays a vital role in the construction of semantic analysis. I has been shown how case and the notion of dependence pertaining to it can be incorporated into Phrase Structure Grammars by use of complex categories. Extending case frames to accommodate adverbial qualities (possibly paraphrased as preposition + case pairs) Egli and Egli (1991) have introduced the idea of construction frames, which proves central to their theory of construction-features (i.e., case and adverbial qualities) as semantically meaningful entities.

Our view of verbs as anadic predicates allows us to dispense with a syntactically defined notion of necessary and optional arguments, facilitating a uniform treatment of noun phrases, prepositional phrases and adverbials as 'sentence props'. This generalization is reflected by the semantic analysis of complements and adjuncts of any category as predicate functors, leading to a very elegant description of syntactic rules in terms of functor application. To achieve this formal unity a Montague-style type raising function is devised, that works on the non-standard types introduced to deal with anadicity and construction.

The use of higher order logic in the formulation of a semantic representation allows for a straightforward treatment of virtually all possible permutations and offers a flexible method of defining combinators for all purposes. Although the merrits of combinatory logic have not been fully incorporated into the present system, the basic principles of Predicate Functor Logic are introduced and a semantic representation language with constructed anadic predicate functors (L_{cap}) is suggested. I have given a small fragment of a grammar in a simplified attribute-value format, endowed with translation functions to assign a semantic representation to any lexical element of the fragment and to build up the semantic representation of strings in a compositional fashion.

Finally the semantic benefit of constructed argument frames though only very briefly touched in this work certainly carries the potential of very explicit and thus more natural semantics for natural language. I have outlined a general format of predicate-predicates for the formulation of structural properties as well as semantic restrictions pertaining to different classes of verbs. To fully explore the possibilities arising with the use of this formal tool, it will be necessary to investigate the number and shape of admissible construction frames. Together with an elaborated network of meaning postulates the Theory of Construction could then provide the key to a truly semantic driven syntax, where word order and argument structure are no longer determined syntactically, but the concept of grammaticality becomes naturally linked to semantic meaningfulness.

Appendix A

The Logic of Constructed Anadic Predicate Functors (L_{cap})[48]

The set T of basic types:

a) e é T
b) t é T
c) con é T

Type formation rules:

a) if a, b é T, then a á b é T
b) if a é T, then list(a) é T
c) if a, b é T, then a^b é T

Primitive vocabulary:[49]

Logical constants:

a) connectives: –, ∨, ∧, á, â
b) quantifiers: Œ, ‰
c) lambda-operator: ¬
d) equality symbol: =

Punctuation symbols:

e) parentheses: (,), [,], <, >

Variables and non-logical constants:

f) For every type a, a denumerably infinite set VAR_a containing variables $v_{n,a}$ for each natural number n.

For convenience we will use the following variable names:

$x_1, x_2, x_3, \ldots, y_1, y_2, y_3, \ldots, z_1, z_2, z_3, \ldots$ é VAR_e

$c_1, c_2, c_3, \ldots$ é VAR_{con}

$L_1, L_2, L_3, \ldots$ é $VAR_{list(e^{\wedge}con)}$

$P_1, P_2, P_3, \ldots, Q_1, Q_2, Q_3, \ldots,$ é $VAR_{list(e^{\wedge}con)á t}$

$F_1, F_2, F_3, \ldots, G_1, G_2, G_3, \ldots$ é $VAR_{(list(e^{\wedge}con)á t)á(list(e^{\wedge}con)á t)}$

g) For every type a, a (possibly empty) set CON_a of (non-logical) constants of type a.

---

48 The formulation of this definition is based on the schema for typed languages given by Partee et al. (1990:343ff).

49 We do not aim at a minimal vocabulary, thus the primitive vocabulary contains redundant elements; for example equality may of course be defined set-theoretically after the principle of Leibniz: a = b â ∀F(Fa â Fb).

Syntactic rules:

The set ME_a of meaningful expressions of type a is defined recursively as follows (where (a) is the basic clause and (b)-(i) are the recursive clauses):

a) For each type a, every variable in VAR_a and every constant in CON_a is in ME_a.

b) For any types a and b, if α é $ME_{a\acute{a}b}$ and β é ME_a, then α(β) é ME_b.

c) For any types a and b, if u is a variable of type a and α é ME_b then (¬uα) é $ME_{a\acute{a}b}$.

d) If ψ and φ are in ME_t (are formulas), then the following are also in ME_t: –ψ, (ψ ∧ φ), (ψ ∨ φ), (ψ á φ), (ψ â φ).

e) For any type a, if ψ é ME_t and u is a variable of type a, then Œuψ and ‰uψ are in ME_t.

f) For any type a, if α and β are both in ME_a, then (α = β) é ME_t.

g) If α é ME_a, then α é $ME_{list(a)}$ and if α = nil, then α é $ME_{list(a)}$.

h) If α é $ME_{list(a)}$ and β é $ME_{list(a)}$, then α;β é $ME_{list(a)}$.

i) If α é ME_a and β é ME_b, then <α,β> é $ME_{a\hat{}b}$.

We define the following predicate functors:

(P is a predicate variable, that is, P é $Var_{list(e\hat{}con)\acute{a}t}$, L stands for a list of arguments (possibly nil), that is L é $Var_{list(e\hat{}con)}$, and c for construction features (é Var_{con}). ¬-expressions of the form ¬x;L F(x;L) are to be seen as abbreviations for ¬y‰x‰L(y = x;L ∧ F(x,L)))

E = ¬c¬P¬L∃z(P(L;<z,c>))

C = $\neg P_1 P_2 \neg L(P_1(L) \wedge P_2(L))$

Semantics:

A frame F is a family of domains, one for each type a:

a) D_e = a non-empty set E
   D_t = {0, 1}

b) D_{con} = a non-empty set C

c) For any a, b é T, $D_{a\hat{}b}$ = a set of ordered pairs from $D_a\hat{}D_b$

d) For any a é T, $D_{list(a)}$ = a set of lists

e) For any a, b é T, $D_{a\acute{a}b}$ = a class of functions from D_a to D_b

(meeting certain closure conditions as specified in Henkin (1950:81f)[50])

[50] Since L_{cap} "incorporates the theory of types, its valid formulas are not recursively enumerable, and therefore no complete axiomatization exists." But it is possible to "prove a generalized completeness

We thus have a general model $M = \langle F, I \rangle$ where I is the Interpretation function that provides the interpretation of the primitive non-logical constants (e.g. maps each individual constant to an individual in D_e).

Semantic rules:

The denotation of an expression α relative to a model M and an assignment function g, symbolized "$\alpha^{M,g}$, is defined recursively, in parallel to the syntactic rules given above, as follows:

a) If α is a non-logical constant in CON_a, then "$\alpha^{M,g} = I(\alpha)$.
If α is a variable in VAR_a, then "$\alpha^{M,g} = g(\alpha)$.

b) If α é $ME_{a á b}$ and β é ME_a, then "$\alpha(\beta)^{M,g}$ = "$\alpha^{M,g}$("$\beta^{M,g}$).

c) If u é ME_a and α é ME_b, then "$(\neg u\alpha)^{M,g}$ is that function f from D_a to D_b such that for any k in D_a, $f(k)$ = "$\alpha^{M,g^{u/k}}$, where $g^{u/k}$ is just like g except that the assignment for u replaced by k.

d) If ψ, φ é ME_t, then:
"$\neg\psi^{M,g}$ = 1 iff "$\psi^{M,g}$ = 0
"$\psi \wedge \varphi^{M,g}$ = 1 iff "$\psi^{M,g}$ = 1 and "$\varphi^{M,g}$ = 1
"$\psi \vee \varphi^{M,g}$ = 1 iff "$\psi^{M,g}$ = 1 or "$\varphi^{M,g}$ = 1
"ψ á $\varphi^{M,g}$ = 1 iff "$\psi^{M,g}$ = 0 or "$\varphi^{M,g}$ = 1
"ψ â $\varphi^{M,g}$ = 1 iff "$\psi^{M,g}$ = "$\varphi^{M,g}$

e) If ϕ é ME_t, u é VAR_a, then
"Œ$u\psi^{M,g}$ = 1 iff for all d é D_a, "$\psi^{M,g^{d/u}}$ = 1
"‰$u\psi^{M,g}$ = 1 iff there is at least one d é D_a such that "$\psi^{M,g^{d/u}}$ = 1

f) If α, β é ME_a, then "$\alpha = \beta^{M,g}$ = 1 iff "$\alpha^{M,g}$ = "$\beta^{M,g}$.

g) If α é ME_a and β é ME_a, then "$\alpha;\beta^{M,g}$ = "$\alpha^{M,g}$;"$\beta^{M,g}$.

h) If α é ME_a and β é ME_b, then "$\langle\alpha,\beta\rangle^{M,g}$ = $\langle$"$\alpha^{M,g}$,"$\beta^{M,g}\rangle$.

---

theorem for an axiomatic formulation IL, based on the corresponding result in Henkin (1950)",that is by using a generalized model instead of a standard model where the domain of functions from M_α into M_β is a subset of all functions from M_α into M_β. (Gallin 1975:17)

Appendix B

The Grammar (a Fragment)

The attributes and their possible values are:

- syntactic category **(cat)**: N, NP, V, VP, P, PP, Adv, AP, Adj, Det;

- lexical form **(lex)**: a German word

- agreement **(agr)**: the agreement features (num), (pers), (gend) - possibly unspecified;
  - number **(num)**: pl, sg;
  - person **(pers)**: 1., 2., 3.;
  - gender **(gend)**: masc, fem, neut;

- construction feature **(con)**: one of the following:
  - one of the cases: nom, acc, dat, gen;
  (- one of the semantic roles: time, loc(ation), dir(ection) ...;)
  - any combination x+y, where x is a preposition with y é (p-subcat), and y is a case;

- definitenes (def): +, -

- construction frame of a verb **(subcat)**: a list of construction features (possibly empty);
  - list processing attributes: **(first)** that picks out the first element of a specified list, and **(rest)** that gives all but the first element;

- construction frame of a preposition **(p-subcat)**: a set of cases;

- verbal form **(form)**: rel(ative), inf(initive) (to be extended ...);

- SLASH feature **(gap)**: a set containing complex categories (possibly empty);

- a feature constraining the application of N á N NP[gen][51]: **(gen–combinable)**: yes, no (default: yes);

- semantic interpretation **(sem)**: an expression of the semantic representation language (association is to the left);

---

[51] All N need an entry '(gen-combinable) = yes' in their attribute-value matrix to allow for the application of the rule N[cas] á N[cas] NP[gen]. Certain other rules, to prevent a later application of this rule, change the value of '(gen-combinable)' to 'no' at their application. This is, for example, to allow for 'der Sohn des Bäckers mit dem Klumpfuß' but not for 'der Sohn mit dem Klumpfuß des Bäckers' (in the same sense).

LEXICON

Where two entries differ only in one or two features, only one entry is printed and the features given with an oblique. The rule is then marked in the title line as standing for x rules by the subscript $^{(x)}$.

Common Nouns

N á Ritter$^{(2)}$ [knight]
d_1(con) = nom/acc
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(lex) = Ritter
d_1(gen-combinable) = yes
d_1(sem) = Ritter'
m = d_1

N á Ritter$^{(2)}$ [knight]
d_1(con) = dat/gen
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(lex) = Ritters
d_1(gen-combinable) = yes
d_1(sem) = Ritter'
m = d_1

N á Edelfrau$^{(4)}$ [noblewoman]
d_1(con) = nom/gen/dat/acc
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = fem
d_1(lex) = Edelfrau
d_1(gen-combinable) = yes
d_1(sem) = Edelfrau'
m = d_1

N á Schwert(3) [sword]
d_1(con) = nom/dat/acc
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = neut
d_1(lex) = Schwert
d_1(gen-combinable) = yes
d_1(sem) = Schwert'
m = d_1

N á Schwertes [sword]
d_1(con) = gen
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = neut
d_1(lex) = Schwertes
d_1(gen-combinable) = yes
d_1(sem) = Schwert'
m = d_1

N á Roland(3)
d_1(con) = nom/acc/dat
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(lex) = Roland
d_1(gen-combinable) = no
d_1(sem) = ¬<i,c>¬P¬L(P(L;<i,c>)) (<d_1(lex),d_1(con)>)
= ¬P¬L(P(L;<Roland,nom/acc/dat>))
m = d_1

N á Rolands
d_1(con) = gen
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(lex) = Roland
d_1(gen-combinable) = no
d_1(sem) = ¬<i,c>¬P¬L(P(L;<i,c>)) (<d_1(lex),d_1(con)>)
= ¬P¬L(P(L;<Roland,gen>))
m = d_1

Determiner

Det á jeder [every]
d_1(con) = nom
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = +
d_1(lex) = jeder
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,nom>)))$
m = d_1

Det á jedes [every]
d_1(con) = gen
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = +
d_1(lex) = jedes
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,gen>)))$
m = d_1

Det á jedem [every]
d_1(con) = dat
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = +
d_1(lex) = jedem
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,dat>)))$
m = d_1

Det á jeden [every]
d_1(con) = acc
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = +
d_1(lex) = jeden
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,acc>)))$
m = d_1

Det á ein [a]
d_1(con) = nom
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = -
d_1(lex) = ein
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,nom>)))$
m = d_1

Det á eines [a]
d_1(con) = gen
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = -
d_1(lex) = eines
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,gen>)))$
m = d_1

Det á einem [a]
d_1(con) = dat
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = -
d_1(lex) = einem
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,dat>)))$
m = d_1

Det á einen [a]
d_1(con) = acc
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = -
d_1(lex) = einen
d_1(sem) = $\neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,acc>)))$
m = d_1

Adjectives

Adj á kühne [valiant]
d_1(con) = nom
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = +
d_1(lex) = kühne
d_1(sem) = kühn'
m = d_1

Adj á kühner [valiant]
d_1(con) = nom
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = -
d_1(lex) = kühner
d_1(sem) = kühn'
m = d_1

Adj á kühnen$^{(6)}$ [valiant]
d_1(con) = gen/dat/acc
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(agr)(gend) = masc
d_1(def) = +/-
d_1(lex) = kühne
d_1(sem) = kühn'
m = d_1

Verbs

V á kämpft [fights]
d_1(subcat) = nom
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(lex) = kämpft
d_1(sem) = kämpfen'
m = d_1

V á liebt [loves]
d_1(subcat) = acc;nom
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(lex) = liebt
d_1(sem) = lieben'
m = d_1

V á widmet [devotes]
d_1(subcat) = dat;acc;nom
d_1(agr)(num) = sg
d_1(agr)(pers) = 3.
d_1(lex) = widmet
d_1(sem) = widmen'
m = d_1

Adverbials

Adv á immer [always]
d_1(lex) = immer
d_1(sem) = ¬P¬LŒt(P(L;<t,time>))
m = d_1

Adv á gestern [yesterday]
d_1(lex) = gestern
d_1(sem) = ¬P¬L(P(L;<gestern,time>))
m = d_1

Adv á überall [everywhere]
d_1(lex) = überall
d_1(sem) = ¬P¬LŒz(P(L;<z,loc>))
m = d_1

Adv á dort [there]
d_1(con) = loc
d_1(lex) = dort
d_1(sem) = ¬P¬L(P(L;<dort,loc>))
m = d_1

Prepositions

P á auf [on]
d_1(p-subcat) = {dat,acc}
d_1(lex) = auf
d_1(sem) = $\neg P_n \neg L_n \neg L_1(P_n(L_n;<POS_{1.2}(L_1),auf+POS_{1.1}(L_1)>))$
m = d_1

SYNTACTIC RULES

Nominals

N á Adj N ≠ N[cas] **á** Adj[cas] N[cas]
e.g. reicher Mann, Haus drüben

d_1(con) = d_2(con)
d_1(agr) = d_2(agr)
m(con) = d_2(con)
m(agr) = d_2(agr)
m(def) = d_1(def)
m(sem) = C d_1(sem) d_2(sem)
[with C = $\neg P_1 P_2 \neg L(P_1(L) \wedge P_2(L))$]

N á N AP ≠ N[cas] **á** N[cas] AP
e.g. Mann mit einem Bart

m(con) = d_1(con)
m(agr) = d_1(agr)
m(gen-combinable) = no
m(sem) = d_2(sem) d_1(sem)

N á N NP ≠ N[cas] **á** N[cas] NP[gen]
e.g. Sohn eines Bäckers

d_2(con) = gen
d_1(gen-combinable) = yes
m(con) = d_1(con)
m(agr) = d_1(agr)
m(sem) = d_2(sem) d_1(sem)

Noun phrases

NP á N ≠ NP **á** PN

d_1(sem) = $\neg\langle i,c\rangle\neg P\neg L(P(L;\langle i,c\rangle))$ (<d_1(lex),d_1(con)>)
m(con) = d_1(con)
m(agr) = d_1(agr)
m(sem) = d_1(sem)

NP á Det N ≠ NP[cas] **á** Det[cas] N[cas]

d_1(con) = d_2(con)
d_1(agr) = d_2(agr)
d_1(def) = d_2(def)
m(con) = d_2(con)
m(agr) = d_2(agr)
m(def) = d_1(def)
m(sem) = d_1(sem) (d_2(sem)) (nil)

NP á NP N ≠ NP[cas] **á** NP[gen] N[cas]
d_1(con) = gen e.g. eines Bäckers Sohn
d_2(con) = nom/acc/dat[52]
d_2(def) = -
m(con) = d_2(con)
m(agr) = d_2(agr)
m(sem) = $\neg Q \neg L_Q \neg P \neg L_P((Q(L_Q;<y,nom>) \wedge P(L_P;<y,d_2(con)>)))$ (d_1(sem) (d_2(sem))) (nil)

Adverbial phrases

AP á Adv ≠ AP[con] **á** Adv[con]
m(con) = d_1(con)
m(sem) = d_1(sem)

AP á PP ≠ AP[con] **á** PP[con]
m(con) = d_1(con)
m(sem) = d_1(sem)

Prepositional phrases

PP á P NP ≠ PP[prep+con] **á** P[prep] NP[con]
d_2(con) é d_1(p-subcat) e.g. auf einem Pferd
m(con) = d_1(lex) + d_2(con)
m(sem) = $\neg P \neg L(d_1(sem)\ (d_2(sem)(P)(L)))$

Verb Phrases

VP á V
m(subcat) = d_1(subcat)
m(agr) = d_1(agr)
m(gap) = d_1(gap)
m(sem) = d_1(sem)

Ordinary Functor Application

VP á SP VP ≠ $VP^a x^o/SP$ **á** SP[a] $VP^a x$, a^o/SP
d_1(con) = d_2(subcat)(first) e.g. Herman auf einem Pferd reitet
d_2(gap) = « auf einem Pferd reiten
m(subcat) = d_2(subcat)(rest)
m(gap) = d_1(gap) (≠ FFP)
m(sem) = d_1(sem) d_2(sem)

SLASH-Introduction

[52] This is to prevent double genitive constructions like '*Es war die Tat einer Witwe Sohnes.'.

SP á t_{SP} ≠ SP/SP **á** t_{SP}
m(gap) = m
m(sem) = ¬F F

VP á SP VP ≠ $VP^{a}x^{o}$/SP[a] **á** SP[a]/SP[a] $VP^{a}x$, a^{o}
d_1(con) = d_2(subcat)(first)
m(subcat) = d_2(subcat)(rest)
m(gap) = d_1
m(sem) = d_1(sem) (d_2(sem))

SLASH-Percolation

VP á SP VP ≠ $VP^{a}x^{o}$/SP **á** SP[a] $VP^{a}x$, a^{o}/SP
d_1(con) = d_2(subcat)(first)
m(subcat) = d_2(subcat)(rest)
m(gap) = d_2(gap) (≠ FFP)
m(sem) = ¬F d_1(sem) (d_2(sem)(F))

SLASH-Termination

VP á SP VP ≠ $VP^{a}x^{o}$ **á** SP[a] $VP^{a}x^{o}$/SP[a]
d_1 é d_2(gap)
m(subcat) = d_2(subcat)
m(gap) = d_2(gap) \ d_1
m(sem) = d_2(sem) (d_1(sem))

Closure of a sentence

VP á VP
d_1(subcat) = (nil)
m(sem) = d_1(sem) (nil)

Existential closure

VP á VP
m(subcat) = d_1(subcat)(rest)
m(sem) = E(d_1(subcat)(first)) (d_1(sem))
[with E = ¬c¬P¬L∃z(P(L;<z,c>))]

Appendix C

More Derivations

Example 1:

(..., daß) ein Lächeln eines kühnen Mannes jede Frau bezaubert

kühnen Mannes

N á Mann [man]
$d_1(sem) = Mann'$
$m = d_1$

Adj á kühnen [valiant]
$d_1(sem) = kühn'$
$m = d_1$

N á Adj N
$m(sem) = C\ d_1(sem)\ d_2(sem)$
[with $C = \neg P_1 P_2 \neg L(P_1(L) \wedge P_2(L))$]

$kühnen\ Mannes(sem) = C\ kühn(sem)\ Mann(sem)$
$= \underline{\neg P_1} P_2 \neg L(P_1(L) \wedge P_2(L))\ [kühn']\ Mann'$
$= \underline{\neg P_2} \neg L(kühn'(L) \wedge P_2(L))\ [Mann']$
$= \neg L(kühn'(L) \wedge Mann'(L))$

eines kühnen Mannes

Det á eines [a]
$d_1(sem) = \neg Q \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,gen>)))$

NP á Det N
$m(sem) = d_1(sem)\ (d_2(sem))\ (nil)$

$eines\ kühnen\ Mannes(sem) = eines(sem)\ (kühnen\ Mannes(sem))\ (nil)$
$= \underline{\neg Q} \neg L_Q \neg P \neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,gen>)))\ [\neg L(kühn'(L) \wedge Mann'(L))]\ (nil)$
$= \neg L_Q \neg P \neg L_P(\exists x(\underline{\neg L}(kühn'(L) \wedge Mann'(L))\ [L_Q;<x,nom>] \wedge P(L_P;<x,gen>)))\ (nil)$
$= \neg L_Q \neg P \neg L_P(\exists x(kühn'(L_Q;<x,nom>) \wedge Mann'(L_Q;<x,nom>)) \wedge P(L_P;<x,gen>)))\ (nil)$
$= \neg P \neg L_P(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge P(L_P;<x,gen>)))$

Lächeln eines kühnen Mannes

N á Lächeln [smile]
d_1(sem) = Lächeln'

N á N NP
m(sem) = d_2(sem) d_1(sem)

Lächeln eines kühnen Mannes(sem)
= eines kühnen Mannes (sem) Lächeln(sem)
= $\underline{\neg P}\neg L_P(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge P(L_P;<x,gen>)))$ [Lächeln']
= $\neg L_P(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(L_P;<x,gen>)))$

ein Lächeln eines kühnen Mannes

Det á ein [a]
d_1(sem) = $\neg Q\neg L_Q\neg P\neg L_P(\exists x(Q(L_Q;<x,nom>) \wedge P(L_P;<x,nom>)))$

NP á Det N
m(sem) = d_1(sem) (d_2(sem)) (nil)

ein Lächeln eines kühnen Mannes(sem)
= ein(sem) (Lächeln eines kühnen Mannes(sem)) (nil)
= $\underline{\neg Q}\neg L_Q\neg P\neg L_P(\exists y(Q(L_Q;<y,nom>) \wedge P(L_P;<y,nom>)))$
$[\neg L(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(L;<x,gen>)))]$ (nil)
= $\neg L_Q\neg P\neg L_P(\exists y(\underline{\neg L}(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(L;<x,gen>)))$
$[L_Q;<y,nom>] \wedge P(L_P;<y,nom>)))$ (nil)
= $\underline{\neg L_Q}\neg P\neg L_P(\exists y(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge$
$Lächeln'(L_Q;<y,nom>;<x,gen>))) \wedge P(L_P;<y,nom>)))$ [nil]
= $\neg P\neg L_P(\exists y(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge$
$Lächeln'(<y,nom>;<x,gen>))) \wedge P(L_P;<y,nom>)))$

jede Frau

Det á jede [every]
d_1(sem) = $\neg Q\neg L_Q\neg P\neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,acc>)))$

N á Frau [woman]
d_1(sem) = Frau'

NP á Det N
m(sem) = d_1(sem) (d_2(sem)) (nil)

jede Frau(sem) = jede(sem) (Frau(sem)) (nil)
= $\underline{\neg Q}\neg L_Q\neg P\neg L_P(\forall x(Q(L_Q;<x,nom>) \mu P(L_P;<x,acc>)))$ [Frau'] (nil)
= $\underline{\neg L_Q}\neg P\neg L_P(\forall x(Frau'(L_Q;<x,nom>) \mu P(L_P;<x,acc>)))$ [nil]
= $\neg P\neg L_P(\forall x(Frau'(<x,nom>) \mu P(L_P;<x,acc>)))$

<u>jede Frau bezaubern</u>

V á bezaubern [enchant]
d_1(sem) = bezaubern'

VP á SP VP
m(sem) = d_1(sem) d_2(sem)

jede Frau bezaubern(sem) = jede Frau(sem) bezaubern(sem)
= $\underline{\neg P}\neg L_P(\forall x(Frau'(<x,nom>) \mu P(L_P;<x,acc>)))$ [bezaubern']
= $\neg L_P(\forall x(Frau'(<x,nom>) \mu bezaubern'(L_P;<x,acc>)))$

<u>(..., daß) ein Lächeln eines kühnen Mannes jede Frau bezaubert</u>

VP á SP VP
m(sem) = d_1(sem) d_2(sem)

ein Lächeln eines kühnen Mannes jede Frau bezaubert(sem)
= ein Lächeln eines kühnen Mannes(sem) jede Frau bezaubern(sem)

= $\underline{\neg P}\neg L_P(\exists y(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(<y,nom>;<x,gen>))) \wedge$
$P(L_P;<y,nom>)))$ [$\neg L(\forall z(Frau'(<z,nom>) \mu bezaubern'(L;<z,acc>)))$]
= $\neg L_P(\exists y(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(<y,nom>;<x,gen>))) \wedge$
$\underline{\neg L}(\forall z(Frau'(<z,nom>) \mu bezaubern'(L;<z,acc>)))$ [L_P;<y,nom>]))
= $\neg L_P(\exists y(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(<y,nom>;<x,gen>))) \wedge$
$(\forall z(Frau'(<z,nom>) \mu bezaubern'(L_P;<y,nom>;<z,acc>)))))$

VP á VP
d_1(subcat) = (nil)
m(sem) = d_1(sem) (nil)

= $\neg L_P(\exists y(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(<y,nom>;<x,gen>))) \wedge$
$(\forall z(Frau'(<z,nom>) \mu bezaubern'(L_P;<y,nom>;<z,acc>)))))$ [nil]
= $\exists y(\exists x(kühn'(<x,nom>) \wedge Mann'(<x,nom>)) \wedge Lächeln'(<y,nom>;<x,gen>))) \wedge$
$(\forall z(Frau'(<z,nom>) \mu bezaubern'(<y,nom>;<z,acc>)))))$

Example 2:

(..., daß) eine Edelfrau jeder Ritter liebt

eine Edelfrau (sem) = $\neg P \neg L_P(\exists x(\text{Edelfrau}'(<x,\text{nom}>) \wedge P(L_P;<x,\text{acc}>)))$
jeder Ritter (sem) = $\neg P \neg L_P(\forall x(\text{Ritter}'(<x,\text{nom}>) \mu P(L_P;<x,\text{nom}>)))$
t_{XP}(sem) = ¬F F
liebt(sem) = lieben'

t_{XP} liebt

VP á SP VP
d_1(con) = d_2(subcat)(first)
m(subcat) = d_2(subcat)(rest)
m(gap) = d_1
m(sem) = d_1(sem) (d_2(sem))

t_{XP} liebt (sem) = t_{XP}(sem) (liebt(sem))
= ¬F F(lieben')

jeder Ritter t_{XP} liebt

VP á SP VP
d_1(con) = d_2(subcat)(first)
m(subcat) = d_2(subcat)(rest)
m(gap) = d_2(gap)
m(sem) = ¬F d_1(sem) (d_2(sem)(F))

jeder Ritter t_{XP} liebt (sem) = ¬G jeder Ritter (sem) (t_{XP} liebt(sem)(G))
= ¬G jeder Ritter(sem) (¬F F(lieben')[G])
= ¬G jeder Ritter(sem) (G(lieben'))
= ¬G ¬P¬L_P(∀x(Ritter'(<x,nom>) μ P(L_P;<x,nom>))) [G(lieben')]
= ¬G ¬L_P(∀x(Ritter'(<x,nom>) μ G(lieben')(L_P;<x,nom>)))

eine Edelfrau jeder Ritter t_{XP} liebt

VP á SP VP
d_1 é d_2(gap)
m(subcat) = d_2(subcat)
m(gap) = d_2(gap) \ d_1
m(sem) = d_2(sem) (d_1(sem))

eine Edelfrau jeder Ritter t_{XP} liebt (sem)

= jeder Ritter $t_{\chi P}$ liebt (sem) (eine Edelfrau (sem))
= $\underline{\neg G}\neg L_P(\forall x(Ritter'(<x,nom>) \mu$
$G(lieben')(L_P;<x,nom>))) [\neg P\neg L_E(\exists y(Edelfrau'(<y,nom>) \wedge P(L_E;<y,acc>)))]$
= $\neg L_P(\forall x(Ritter'(<x,nom>) \mu$
$\underline{\neg P}\neg L_E(\exists y(Edelfrau'(<y,nom>) \wedge P(L_E;<y,acc>)))[lieben'] (L_P;<x,nom>)))$
= $\neg L_P(\forall x(Ritter'(<x,nom>) \mu$
$\underline{\neg L_E}(\exists y(Edelfrau'(<y,nom>) \wedge lieben'(L_E;<y,acc>))) [L_P;<x,nom>]))$
= **$\neg L_P(\forall x(Ritter'(<x,nom>)$** μ
$(\exists y(Edelfrau'(<y,nom>) \wedge lieben'(L_P;<x,nom>;<y,acc>)))))$

References

The reference year given in the text is always the year of first publication (with the exception of Abraham (1988) and Bußmann (1990)), while the page numbers may refer to a later edition or translation as stated in the references.

ABRAHAM, Werner: (1988) Terminologie zur neueren Linguistik. Germanistische Arbeitshefte. Tübingen, Niemeyer;
(this volume is a second, completely revised edition of Abraham 1973)

BACH, Emmon: (1962) The Order of Elements in a Transformational Grammar of German. Language 38:3, 263-269.

BAR-HILLEL, Yehoshua (ed.): (1971) Pragmatics of Natural Language. Dordrecht, Reidel.

BEAR, John: (1982) Gaps as Syntactic Features. Bloomington, Indiana University Linguistics Club.

BESTEN, den H. & J. A. EDMONDSON: (1983) The verbal complex in continental Westgermanic. In: Werner Abraham (ed.): On the formal syntax of the Westgermania. (Linguistik Aktuell 3). Amsterdam. 155-216.

BIERWISCH, Manfred: (1963) Grammatik des deutschen Verbs. studia grammatica II. Berlin, Akademie Verlag; (quoted from 1971: Seventh impression).

BRESNAN, Joan: (1982) Control and complementation. In: J. Bresnan (ed.): The mental representation of grammatical relations. Cambridge, 282-390.

BROCKHAUS, Klaus: (1971) Automatische Übersetzung. Untersuchungen am Beispiel der Sprachen Englisch und Deutsch. Braunschweig, Vieweg.

BUSSMANN, Hadumod: (1990) Lexikon der Sprachwissenschaft. Stuttgart, Kröner;
(= second, completely revised edition of Bußmann 1983).

CARNAP, Rudolf: (1947) Meaning and Necessity. Chicago, University of Chicago Press.

CHOMSKY, Noam: (1957) Syntactic Structures. The Hague, Mouton.

CHOMSKY, Noam: (1964) The logical basis of linguistic theory.
In: H. G. Lunt (ed.): Proceeedings of the Nith International Congress of Linguists 1962. The Hague, Mouton, 914-978.

CHOMSKY, Noam: (1965) Aspects of the Theory of Syntax. Cambridge, Mass., M.I.T Press.

CHOMSKY, Noam: (1975) Reflections on language. New York, Random House.

CHOMSKY, Noam: (1981) Lectures on Government and Binding. Dordrecht, Foris (Studies in Generative Grammar 9).

CHURCH, Alonzo: (1940) A Formulation of the Simple Theory of Types. Journal of Symbolic Logic 5, 56-68.

CHURCH, Alonzo: (1941) The Calculi of Lambda-Conversion. Princeton, N.J., Princeton University Press.

CLAHSEN, Harald: (1982) Spracherwerb in der Kindheit. Eine Untersuchung zur Entwicklung der Syntax bei Kleinkindern. Tübinger Beiträge zur Linguistik. Series A: Language Development 4. Tübingen, Gunter Narr.

COPI, Irving M.: (1971) The Theory of Logical Types. London, Routledge & Paul.

CRESSWELL, Maxwell J.: (1992) Truth-Conditional and Model-Theoretic Semantics. In: William Bright (ed.): International Encyclopedia of Linguistics. Vol. 3. New York, Oxford, Oxford University Press. 404–405.

DOWTY, David: (1982) Grammatical Relations and Montague Grammar. In: Pauline Jacobson & Geoffrey K. Pullum (eds). The Nature of Syntactic Representation. Dordrecht, Reidel (Synthese Language Library).

DOWTY, David R. & Robert E. WALL & Stanley PETERS: (1981) Introduction to Montague Semantics. Dordrecht, Reidel.

EGLI, Urs & Renata EGLI-GERBER: (1991) Sprachsysteme - logische und historische Grundlagen der erweiterten Phrasenstrukturgrammatik. Universität Konstanz, Fachgruppe Sprachwissenschaft (Arbeitspapier 28) (quoted from 1992: second, corrected impression).

EGLI, Urs & Klaus v. HEUSINGER: (1991) Epsilon-Operator und E-Typ-Pronomen. In: Egli & v. Heusinger: Zwei Aufsätze zur definiten Kennzeichnung. Universität Konstanz, Fachgruppe Sprachwissenschaft (Arbeitspapier 27).

EGLI, Urs & Uta SCHWERTEL: (1991a) Typentheorie und Semantik. unpublished lecture notes taken by U. Schwertel in a seminar of U. Egli in winter 1990/91 at the University of Konstanz.

EGLI, Urs & Uta SCHWERTEL: (1991b) Montagues Intensional Logik (IL). unpublished lecture notes taken by U. Schwertel in a series of lectures given by U. Egli at the University of Konstanz.

FILLMORE, Charles: (1968) The case for case. In: Emmon Bach & Robert T. Harms (eds): Universals in Linguistic Theory. New York, Holt, Rinehart & Winston.

FOURQUET, Jean: (1957/58) Revision of: Heinz Anstock: Deutsche Syntax, Lehr- und Übungsbuch. Wirkendes Wort 8, 120-122.

FREGE, Gottlob: (1897) Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens. Halle/S.

GALLIN, Daniel: (1975) Intensional and Higher-Order Modal Logic. With Applications to Montague Semantics. Amsterdam, North-Holland.

GAZDAR, Gerald: (1981) Unbounded Dependency and Coordinate Structure. Linguistic Inquiry 12, 2, 155-184.

GAZDAR, Gerald: (1982) Phrase Structure Grammar. In: Pauline Jacobson & Geoffrey K. Pullum (eds). The Nature of Syntactic Representation. Dordrecht, Reidel (Synthese Language Library).

GAZDAR, Gerald: (1987) Generative Grammar. In: Lyons et al. 1987.

GAZDAR, Gerald & Ewan KLEIN & Geoffrey PULLUM & Ivan SAG: (1985) Generalized Phrase Structure Grammar. Cambridge, Mass., Harvard University Press.

GEBAUER, Heiko: (1978) Montague-Grammatik. Eine Einführung mit Anwendung auf das Deutsche. Germanistische Arbeitshefte. Tübingen, Niemeyer.

GRANDY, Richard E.: (1976) Anadic Logic and English. Synthese 32, 359–402.

GREWENDORF, Günther & Fritz HAMM & Wolfgang STERNEFELD: (1987) Sprachliches Wissen. Eine Einführung in moderne Theorien der grammatischen Beschreibung. Frankfurt a. M., Suhrkamp.

GÜNTHNER, Susanne: (1992) "... weil - man kann es ja wissenschaftlich untersuchen" - Diskurspragmatische Aspekte der Wortstellung in WEIL-Sätzen. Universität Konstanz, Fachgruppe Sprachwissenschaft (Arbeitspapier 45).

HARRIS, Zellig S.: (1951) Methods in Structural Linguistics. Chicago, Chicago University Press. (Structural Linguistics. Midway Reprint Edition 1986).

HARRIS, Zellig S.: (1957) Co-occurrence and Transformation in Linguistic Structure. Language 33, 283-341.

HENKIN, Leon: (1950) Completeness in the Theory of Types. Journal of Symbolic Logic 15, 2, 81-91.

HENKIN, Leon & J. D. MONK & Alfred TARSKI: (1972) Cylindric Algebras I. Amsterdam.

HERINGER, Hans Jürgen & Bruno STRECKER & Rainer WIMMER: (1980) Syntax. Fragen - Lösungen - Alternativen. München, Fink Verlag (UTB).

HORROCKS, Geoffrey: (1987) Generative Grammar. New York, Longman.

HOUSEHOLDER, Fred W.: (1981) The Syntax of Apollonios Dyscolus. Translated and commened by F. W. Householder. Amsterdam, John Benjamins (Amsterdam Studies in the Theory and History of Linguistic Science, Studies in the History of Linguistics 23)

JOHNSON, Mark: (1988) Attribute-Value Logic and the Theory of Grammar. Stanford, CSLI (CSLI Lecture Notes 16).

KARTTUNEN, Lauri: (1986) Radical Lexicalism. Artificial Intelligence Center at SRI and Center for the Study of Language and Information: Stanford (presented at the Conference on Alternative Conceptions of Phrase Structure, July 1986, New York).

KAY, Martin: (1979) Functional Grammar. In: Proceedings of the fifth Annual Meeting of the Berkeley Linguistics Society. Berkeley, California.

KNÖPFLER, Siegfried: (1979) Linguistische und formallogische Untersuchung zur Prädikat-Funktor-Logik (PFL). Mit Anhängen von U. Egli und Th. Zimmermann. Universität Konstanz, Sonderforschungsbereich 99 "Linguistik".

KOHRT, Manfred: (1978) The Long and Winding Road to German SOV, Aller-Retour, Taking the Short-Cut. Linguistische Berichte, 57, 69-73.

KONDAKOW, N. I.: (1975) Logiçeskij slovar´spravoçnik. Izdatel´qtvo »Nauka«. Moskva. (quoted from the German translation, edited by E. Albrecht & G. Asser: (1978) Wörterbuch der Logik. West-Berlin, verlag das europäische buch.

KRATZER, Angelika: (1988) Stage-Level and Individual-Level Predicates. In: Manfred Krifka: Genericity in Natural Language. Proceedings of the 1988 Tübingen Conference.

KRATZER, Angelika & Eberhard Peter PAUSE & Arnim von STECHOW: (1973) Einführung in die Theorie und Anwendung der Generativen Syntax. Vol 1. Frankfurt, Athenäum.

LEWIS, David: (1972) General Semantics. In: Donald Davidson & Gilbert Harman (eds.): Semantics of Natural Language. Dordrecht, Reidel.

LYONS, John: (1987) Semantics. In: Lyons et al. 1987.

LYONS, John & Richard COATES & Margaret DEUCHAR & Gerald GAZDAR: (1987) New Horizons in Linguistics 2. London, Penguin.

MALCHOW, Anne: (1992) Der Attribut-Wert Formalismus von Mark Johnson. In: Urs Egli, Klaus v. Heusinger & Anne Malchow: Aufsätze zur Unifikationsgrammatik. Universität Konstanz, Fachgruppe Sprachwissenschaft (Arbeitspapier 42).

MATTHEWS, Peter H.: (1981) Syntax. Cambridge, Cambridge University Press (Cambridge Textbooks in Linguistics).

MILLER, Dale A. & Gopalan NADATHUR: (1986a) Some Uses of Higher-Order Logic in Computational Linguistics. Proceedings of the 24th Annual Meeting of the Association for Computational LInguistics, New York, Columbia University, 247-256.

MILLER, Dale A. & Gopalan NADATHUR: (1986b) Higher-Order Logic Programming. In: E. Shapiro (ed.): Proceedings of the Third International Conference on Logic Pprogramming. Berlin, Springer (LNCS 225), 448-462.

MONTAGUE, Richard: (1970a) Universal Grammar. Theoria, 36, 373-398. (quoted from Montague 1974)

MONTAGUE, Richard: (1970b) English as a Formal Language. In Bruno Visentini et al.(eds.):Linguaggi nella Societa e nella Tecnica. Milan, Edizioni di Comunità, 189-224. (reprinted in Montague 1974)

MONTAGUE, Richard: (1973) The Proper Treatment of Quantification in Ordinary English. In Jaakko Hintikka, Julius Moravcsik & Patrick Suppes (eds.): Approaches to Natural Language: Proceedings of the 1970 Stanford Workshop on Grammar and Semantics. Dordrecht, Reidel, 221-242. (reprinted in Montague 1974)

MONTAGUE, Richard: (1974) Formal Philosophy: Selected Papers of Richard Montague. Edited and with an introduction by Richmond H. Thomason, New Haven and London, Yale University Press; (quoted from 1979: Third impression).

PARTEE, Barbara H.: (1975) Montague Grammar and Transformational grammar. Linguistic Inquiry 6, 203-300.

PARTEE, Barbara H. & Alice TER MEULEN & Robert E. WALL: (1990) Mathematical Methods in Linguistics. Dordrecht, Kluwer.

PETERS, P. Stanley Jr. & Robert W. RITCHIE: (1973) On the Generative Power of Transformational Grammars. Information Science 6, 49-83.

POLLARD, Carl: (1984) Generalized Phrase Structure Grammars, Head Grammars, and Natural Languages. Dissertation, Stanford University.

POLLARD, Carl & Ivan A. SAG: (1987) Information-Based Syntax and Semantics. Vol. 1: Fundamentals. CSLI Lecture Notes 13, Stanford.

QUINE, Willard van Orman: (1971) Predicate Functor Logic. In: J. E. Fenstad (ed.): Proceedings of the 2. Scandinavian Logic Symposium. Amsterdam, North-Holland, 309-315.

QUINE, Willard van Orman: (1972), Algebraic Logic and Predicate Functors. In: R. Rudner & I. Scheffler (eds.): Logic and Art: Essays in Honor of Nelson Goodman. Indianapolis and New York, Bobbs-Merrill.

RUSSELL, Bertrand: (1903) The Principles of Mathematics. Cambridge. (2nd ed., 1938, New York)

RUSSELL, Bertrand: (1908) Mathematical Logic as Based on the Theory of Types. American Journal of Mathematics 30, 222-262.
(Reprinted in R. C. Marsh (ed.): (1956) Logic and Knowledge. London. and in J. van Heijenoort: (1967) From Frege to Gödel. A Source Book in Mathematical Logic. 1879-1931. Cambridge, Mass.)

SCHÖNFINKEL, Moses: (1924) Über die Bausteine der mathematischen Logik. Mathematische Annalen 92, 305-316.

SHIEBER, Stuart: (1985) Evidence against the Context-Freeness of Natural Language. Linguistics and Philosophy 8, 333-343.

SHIEBER, Stuart: (1986) An Introduction to Unification-Based Approaches to Grammar. Stanford, CSLI (CSLI Lecture Notes 4).

SMULLYAN, R. M.: (1968) First Order Logic. Berlin, Springer.

TARSKI, Alfred: (1935) Der Wahrheitsbegriff in den formalisierten Sprachen. Studia Philosophica 1, 261-405.

TARSKI, Alfred: (1954) Contributions to the Theory of Models I. Indagationes Mathematicae 16, 572-581.

TESNIERE, Lucien: (1959) Eléments de syntaxe structurale. Paris, C. Klincksieck; (Translated by Ulrich Engel (ed.): (1980) Grundzüge der strukturalen Syntax. Klett-Cotta: Stuttgart.

USZKOREIT, Hans: (1986) Categorial Unification Grammars. Proceedings of the International Conference on Computational Linguistics (Coling). Bonn, 187-194.

USZKOREIT, Hans: (1987) Word Order and Constituent Structure in German. Stanford, CSLI (CSLI Lecture Notes 8).

WELLS, Rulon S.: (1947) De Saussure's System of Linguistics. Word 3, 1–31.

WHITEHEAD, Alfred North & Bertrand RUSSELL: (1910-13) Principia Mathematica. 3 Vol., Cambridge. (2nd ed. 1927).

WINOGRAD, Terry: (1983) Language as a Cognitive Process. Volume I: Syntax. Reading/Mass., Addison-Wesley.

ZIMMERMANN, Thomas Ede: (1989) Intensional Logic and Two-Sorted Type Theory. Journal of Symbolic Logic 54, 65-77.