언어 제7권 제2호 (1982)

A Surface Compositional Categorial Syntax

Roland R. Hausser
(Institut für Deutsche Philologie Ludwig-Maximilians Universität München)

0. *Introductory remarks*

This paper presents a new type of syntax, called orthogonal syntax or ORTAX. ORTAX is a categorial syntax in the tradition of Lesniewski (1929), Ajdukiewicz (1935), Bar-Hillel (1964), and Montague (1974). ORTAX differs from Bar-Hillel's system, however, in that it allows the treatment of discontinuous elements. ORTAX differs from Montague's syntax, furthermore, in that ORTAX is a pure categorial system in the sense that it is strictly context-free (the formal proof is given in Hausser 1980c). In many ways, ORTAX may be regarded as a refinement of Montague's PTQ-syntax, to which it is similar in that it is indirectly-interpreted via translation into intensional logic.

The paper consists of four sections, whereby the first three sections are of a preparatory nature in that they give a general outline of our approach to syntax, semantics, and the lexicon. The fourth section, on the other hand, presents the formal definitions of ORTAX and the systematic derivation of examples, including one with a discontinuous element, one with a relative clause, one with quantifier scope variation, and one with a so-called unbounded dependency.

1. *Surface compositional syntax*

A generative syntax of natural language may be characterized by the way it answers the following two questions:

Q(i) What are the basic units of this generative syntax?

Q(ii) What are the modes of composition employed by this syntax?

There are, of course, many more questions to be asked of a syntax, like "Does it serve as a natural basis for semantic interpretation?", "What kind of category system is it based on?", etc., but for now the questions Q(i) and Q(ii) shall suffice. The characteristic answers to them by a surface compositional syntax are given in A(i) and A(ii), respectively.

A(i) Each *surface word* is a basic unit of a surface compositional syntax, and the only basic units of a surface compositional syntax are surface words.

A(ii) Surface compositional syntax employs only one type of syntactic combination, namely category composition (in the tradition of categorial syntax), which is the syntactic counterpart to the combining of a function with an argument.

But what is a surface word? While the formal definition of the concept 'word' is notoriously difficult, the intuitive notion is quite clear in the vast majority of cases. Let us consider a simple example.

(1) John reads a book.

Obviously, (1) consists of the surface words *John*, *reads*, *a*, and *book*. In addition to these clear instances of surface words in (1) we will for theoretical reasons also count punctuation signs like '.' as a special type of surface word (cf. Hausser (1980b, 1980c), where this move is extensively motivated).

Just by looking at the clear instances of surface words in natural language it becomes obvious that neither of the two main representatives of generative grammar, namely transformational grammar and Montague grammar, are surface compositional. For example, the transformational derivation of a word like *kill* from a syntactic structure like *cause to become not alive* (as proposed by McCawley (1968) is not surface compositional because it assumes basic syntactic units (e.g. *cause*, *become*) below the level of the surface word *kill*. The syncategorematic introduction of complementizer words and relative pronouns, customary both in transformational grammar and Montague grammar, likewise violates the principle A(i) of surface compositional grammar in that these words are not treated as basic syntactic units. I.e., they are not treated as words which combine with other expressions, but instead they are smuggled into the surface by means of special rules without ever attaining a proper lexical status. The same holds of Montague's treatment of determiners like *the*, *a*, and *every* as well as sentence conjunctions like *and* and *or* in PTQ.

Syncategorematic operations are strictly prohibited in surface compositional grammar, because they amount to a violation of the so-called *Fregean principle*:

> The meaning of a complex expression is a function of the meaning of its basic parts (i.e. the surface words) and their mode of composition.

The essence of surface compositional grammar resides in the fact that the Fregean principle is strictly applied to the *natural surface*. It has been shown in Hausser (1978b) that in those grammatical systems which apply the Fregean principle not to the concrete surface, but to some imaginary deep structure level, the restrictive power of the Fregean principle on syntactic, semantic, and pragmatic analysis is lost. I.e., such non-surface compositional systems allow in theory that any surface can be derived from a given deep structure.

Now that we have isolated the basic units of sentence (1) let us turn to the question of their syntactic combination. As a consequence of the Fregean principle, syntactic combinantion must be semantically interpreted. That is, every time two expressions are combined in the course of a syntactic combination, the semantic impact of this operation has to be accounted for. Thus the syntactic composition of forms must be accompanied by the semantic composition of meanings.

The most general principle of semantic composition is the so-called *functional application* (assuming a so-called truth-or denotation-conditional analysis of meaning in the tradition of modern logic, e.g. predicate calculus).

A function is characterized by its domain, its range, and its definition (Zuordnungsbestimmung). Let $f_{A|B}$ be a function-expression denoting a function with a domain of type A and a range of type B. The rule of functional application says, that a function $f_{A|B}$ can only be applied to an argument of type A, e.g. a_A, and will render a value of type B, e.g. $f(a)_B$. What $f(a)_B$ exactly denotes depends on the definition of a_A and the definition of $f_{A|B}$.

The combination of a function expression $f_{A|B}$ with an argument expression a_A, called categorial composition, is part of the logical syntax, while the corresponding application of the function to the argument denoted by the respective expressions, called functional application, is part of logical semantics. According to A(ii), the only mode of syntactic combination in surface compositional grammar is categorial composition. The reason for choosing categorial composition is twofold: (i) it provides a natural basis for an associated semantic interpretation, i.e. functional application, and (ii) it turns out — and this is a major empirical claim of surface compositional syntax that categorial composition completely suffices to describe the combination of words in the surface of any natural language (assuming a type logic with lambda-operator as our auxiliary language — but see section 3 below).

In logic, categorial composition is stated in terms of the following rule:

$$\mathrm{f}_{A|B} \cdot \mathrm{a}_A = \mathrm{f}(\mathrm{a})_B$$

In other words, the convention in logic is that the function expression precedes the argument expression and that the argument expression is enclosed in brackets. In natural language, on the other hand, the expression denoting a function may precede as well as follow the argument expression. Consider the following two examples:

(i) $\textit{red}_{CN|CN} \cdot \textit{house}_{CN} = \textit{red house}_{CN}$

(ii) $\textit{which is red}_{CN|CN} \cdot \textit{house}_{CN} = \textit{house which is red}_{CN}$

In other words, $\textit{red}_{CN|CN}$ as well as $\textit{which is red}_{CN|CN}$ denote functions which take the denotation of the common noun $\textit{house}_{CN}$ as arguments and result in the complex common noun meanings denoted by ***red house***$_{\mathrm{CN}}$ and ***house which is red***$_{\mathrm{CN}}$, respectively. The difference is that in (i) the function expression precedes the argument expression (as in logic), while in (ii) the converse holds. A third type of category composition found in natural languages is based on so-called discontinuous elements, where the function and the argument expressions are not even adjacent to each other.

Let us return now to our example (1) and compare (1) and (2).

(1) John reads a book.
(2) .((reads(a(book)))(John))

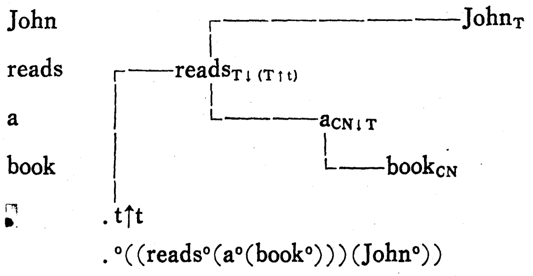
While (1) represents the example in its natural surface order, (2) represents the corresponding logical form in accordance with the above rule of category composition. We conclude that the goal of a surface compositional syntax is quite simply to

(i) determine the function/argument structure of natural surfaces, and
(ii) correlate the natural surface order to the corresponding logical order.

Consider (3), where the surface order and the presumed logical structure of example (1) are jointly represented in form of a novel type of tree structure, called an orthogonal tree.[1]

1) These tree structures were first presented in an informal talk at the Linguistics Department of Stanford University in November 1979. I adopted the name 'orthogonal tree' from a remark made by Ron Kaplan in a discussion after the talk. The formal system of the orthogonal syntax was developed in Hausser (1980c). Hausser (1980b) contains a sample of 25 orthogonally analyzed sentences of English, including interrogatives, imperatives, quantifier scope variation, relative clauses, closed sentential complement like that-complements and infinitival complements, open sentential complements like who-, where-, whether-, etc. clauses, unbounded dependencies, etc.

The manuscript of this paper was read by my colleague Claudia Gerstner. I would like to thank her for comments, which have led to numerous improvements throughout the text.

(3)

John
reads
a
book
.
$John_T$
$reads_{T\downarrow(T\uparrow t)}$
$a_{CN\downarrow T}$
$book_{CN}$
$.t\uparrow t$
$.^o((reads^o(a^o(book^o)))(John^o))$

In (3), each surface word appears in two forms. One is the so-called analyzed surface word, consisting of the surface form and a subscripted refined category, called subcategory, e.g. $a_{CN\downarrow T}$. The other is the so-called unanalyzed translation, consisting of the surface form plus the '^{o}'-diacritic, e.g. a^o. The surface order of the sentence is encoded in the column on the left of the tree (top to bottom), while the logical structure is encoded horizontally (left to right). Thereby, the former consists of analyzed surface words, while the latter consists of unanalyzed translations.

The purpose of the new subcategories is that they encode the surface combinatorial properties of the words they index (in addition to the basic domain/range structure). The word *a*, for example, has the sub-category CN ↓ T. The arrow '↓' indicates that in the orthogonal tree the argument expression (e.g. book_{CN}) stands directly ***below*** the function expression $a_{CN\downarrow T}$. This means that on the surface the argument expression book_{CN} follows the function expression $a_{CN\downarrow T}$. The inverse arrow '↑', as in ***reads a*** $\mathit{book}_{T\uparrow t}$, on the other hand, indicates that the argument expression, e.g. $John_T$, stands directly above the function expression, and thus precedes it on the surface: John_T ***reads a*** $\mathit{book}_{T\uparrow t}$.

The refined subcategories will be explicitly defined in section 4, where we also present a generative rule system, called ORTAX (from 'orthogonal syntax'). ORTAX generates complete orthogonal trees, like the one illustrated in (3), including structures of so-called discontinuous elements. The two rules of ORTAX needed to generate the tree (3) are given below:

STl. 1 If $\alpha \in P_{A\uparrow B}$ and $\beta \in P_A$ (i.e. if α,β are proper expressions of subcategory A ↑ B and A, respectively), then $F_1(\alpha,\beta) \in P_B$ and

$$\begin{array}{ll} & \ulcorner\!\text{———}\,\beta_A \\ & | \\ F_1(\alpha,\beta) = & \alpha_{A\uparrow B} \\ & \alpha^{o} \quad (\beta^{o}) \end{array}$$

where α^{o} and β^{o} are the unanalyzed translations of α and β, respectively.

STl. 2 If $\alpha \varepsilon P_{A\downarrow B}$ and $\beta \varepsilon P_A$, then $F_2(\alpha,\beta) \varepsilon P_B$, and

$$\begin{array}{ll} F_2(\alpha,\beta) = & \alpha_{A\downarrow B} \\ & | \\ & \llcorner\!\text{———}\,\beta_A \\ & \alpha^{o} \quad (\beta^{o}) \end{array}$$

The rules of ORTAX differ from Montague's PTQ in that ORTAX states the syntactic and the semantic effect of a categorial composition *jointly* in form of an ST-rule, whereas PTQ states the syntactic aspect in terms of S-rules and the semantic effect in terms of T-rules (T for translation). Furthermore, ORTAX differs from PTQ in that the rules of ORTAX describe solely various forms of categorial composition, so that all the syncategorematic operations of PTQ, including variable substitution, are avoided in the surface syntax. ORTAX is based on a lexicon where the words of the language under investigation are indexed by our new subcategories, which completely encode (i) the combinatorial properties of the expressions they index, and (ii) the domain-range structure of the function denoted by the corresponding unanalyzed translation. The second point is due to the so-called category/type correspondence, which will be discussed in the following section.

2. *Surface compositional semantics*

As we have seen, surface compositional syntax is limited to a description of the function-argument structure of the surface expressions of the natural language under description, taking the surface words as the basic units. The correlation of the surface order and the logical structure is systematically described in the format of orthogonal trees. To someone used to 'syntactic generalizations' in the tradition of transformational grammar, this surface compositional syntax may seem extremely sparse. And indeed, the purpose of our type of analysis becomes evident only when we consider the associated surface compositional semantics.

Surface compositional semantics may be characterized by the following two principles:

(a) Surface compositional semantics describes the ***literal meaning*** of expressions.

(b) Surface compositional semantics is naturally divided into two parts, namely so-called ***word semantics*** and so-called ***sentence semantics.***

Note that the term ***literal meaning*** (also called meaning^1) is used in contrast to a second type of meaning (called the ***speaker meaning*** or meaning^2). While meaning^1 is a property of natural surface expressions, meaning^2 is defined as the ***use*** of a surface expression (with its fixed properties such as surface form and compositionally encoded meaning^1) by a speaker relative to an utterance-situation (context). The importance of the distinction between the literal meaning of expressions (meaning^1) and the speaker meaning of utterances (meaning^2) cannot be exaggerated. It has been motivated and discussed from various angles in Hausser (1978a, 1979a, 1980a). It is due to this distinction that the area of semantics and the associated syntax is naturally restricted to a purely structural analysis of concrete surface expressions, leaving enough regularities for a theory of pragmatics to get off the ground.

For the semantic description of surface expressions, characterizing their literal meaning, we use the tehniques of modern logic, more specifically a model-theoretically interpretable type logic with lambda operator. In the semantic analysis of surface expressions we generally distinguish the following three levels:

(i) the unanalyzed translation,
(ii) the analyzed translation, and
(iii) the meaning^1-formula.

These three levels of analysis will be illustrated in the following semantic analysis of sentence (1).

The ***unanalyzed translation*** of sentence (1) is already known. It consists in the bottom line of the orthogonal tree (3) and is repeated in (4) for convenience.

(4) $.^{o}((\text{reads}^{o}(\text{a}^{o}(\text{book}^{o})))(\text{John}^{o}))$

As already mentioned in section 1 above, the symbols $.^{o}$, ***reads***^{o}, ***a***^{o}, etc. are called the unanalyzed translations of the corresponding surface words $._{t\uparrow t}$, ***reads***$_{T\downarrow(T\uparrow t)}$, $a_{CN\downarrow T}$, etc. Unanalyzed word translations are defined as constants of intensional logic. Intensional logic, as defined by Montague in PTQ, is a type language. Since the type of logical expressions plays a similar role on the level of intensional logic as the category of words on the surface level, Montague correlated the categories of the surface language and types of intensional logic by defining a so-called category/type correspondence (Montague 1974, p. 260). This category/type correspondence is an important part of surface compositional grammar, because it leads to a coherent ***categorial frame*** (Hausser 1980c) of the surface and the translation language.

The (sub-)category of a surface expression (which reflects its combinatorial properties) systematically determines the type of the corresponding unanalyzed translation. Thus the types of the constants in (4) need not to be explicitly stated since they can be deduced from the corresponding surface categories. The logical representation in the orthogonal tree consisting of unanalyzed translations represents the complete ***sentence semantic analysis*** of the expression generated. A formula like (4) is a well-formed expression of intensional logic, composed by means of functional application in accordance with the type of each unanalyzed translation.

However, while the domain and range of the function denoted by, e.g. a°, is known in terms of the type of a° (which in turn is specified in terms of the corresponding surface category), the ***definition*** (Zuordnungsbestimmung) of this function is still unknown. In theory, there are two ways to supply the definition for the function denoted by, e.g. a°. One is to define a° ***directly*** either by treating it as an elementary operator of intensional logic (supplying it with a metalanguage definition like that of $\sim$, $\wedge$, Λ, or $\Box$) or by specifying its extension at every index of the model-structure.

The other possibility is to treat a° as an abreviation of a complex formula, e.g.

$$\hat{Q}\hat{P}Vx[Q(x) \wedge P(x)]$$

which consists of the standard operators and variables of intensional logic. In the second case, the meaning a° is composed via the standard definitions associated with the above formula. We call this formula the ***analyzed translation*** of $a_{CN\downarrow T}$. Since it is obviously highly unpractical to treat a°– along with the unanalyzed translations of all other surface words — as an elementary operator or function (something the advocates of 'direct interpretation' manage to overlook), we choose to provide the definitions of unanalyzed translations by equating them with analyzed translations.

The step from unanalyzed to analyzed word translations corresponds to the step from sentence semantics to word semantics in surface compositional grammar. Consider (5), where the unanalyzed word translations of (4) are correlated to corresponding analyzed translations.

$$\begin{array}{lllllll}
 & .^{\circ} & ((& \text{reads}^{\circ} & (& \text{a}^{\circ} & (\text{book}^{\circ})))\\
 & | & & | & & | & |\\
(5) & \hat{p}[p] & ((& \hat{P}_1\hat{P}_0[P_0\hat{x}[P_1\hat{y}[\text{read}'(x,y)]]] & (& \hat{Q}\hat{P}Vz[Q(z) \wedge P(z)] & (\text{book}')))\\
 & (\text{John}^{\circ})) & & & & & \\
 & | & & & & & \\
 & (\hat{P}P(j))) & & & & &
\end{array}$$

The analyzed translations in (5) are built up by means of the lambda operators $\lambda p, \lambda P_0, \lambda P_1, \lambda x, \lambda y, \lambda P, \lambda Q$ (written as $\hat{p}, \hat{P}_0, \hat{P}_1, \hat{x}, \hat{y}, \hat{P}, \hat{Q}$, respectively), functional application, the operator $\wedge$, the existential quantifier Vz, the variables $p, P_0, P_1, x, y, z, P, Q$, and the constants ***read'***, ***book'***, and ***j***.[2]

2) These symbols are of the following types (the corresponding categories are listed in the middle):

symbol	*category*	*type*
p	t	⟨s, t⟩

Each analyzed word translation in (5) is a well-formed expression of intensional logic, the type of which is the same as the category induced type of the respective unanalyzed word translation. Consider for example the analyzed translation of $\mathit{reads}_{T|(T|t)}$ as given in (5). It consists of the logical constant *read′*, which corresponds in type to the category e|(e|t), and the variables x and y, which correspond to category e. Thus *read′* (x,y) (which is equivalent to (*read′*(y))(x), cf. Montague 1974, p. 259) corresponds to category t. The analyzed translation of *reads* consists furthermore of the lambda-operators λx, λy, written as $\hat{x}$,$\hat{y}$, and the variables P_0, P_1, which correspond in type to the category (e|t)|t (abbreviated as T). According to the logical syntax of the lambda-operator, $\hat{y}[\mathit{read}'(x,y)]$ is a well-formed expression corresponding in type to the category e|t. Applying the variable P_1 renders the t-expression $P_1(\hat{y}[\mathit{read}'(x,y)]$. Addition of the lambda-operator $\hat{x}$ renders $\hat{x}[P_1\hat{y}[\mathit{read}'(x,y)]]$, which again corresponds to category e|t. Applying the variable P_0 renders the t-expression $P_0\hat{x}[P_1\hat{y}$ $[\mathit{read}'(x,y)]]$. Finally we apply the two lambda operators $\hat{P}_1$ and $\hat{P}_0$, binding the remaining unbound variables, and rendering the complete analyzed word translation

$$\hat{P}_1\hat{P}_0[P_0\hat{x}P_1\hat{y}[\mathit{read}'(x,y)]]$$

which corresponds in type to the category T|(T|t), as desired.

P_0, P_1	(e|t)|t	⟨s, ⟪s, ⟪s, e⟩, t⟫, t⟫
x, y	e	⟨s, e⟩
P, Q	e|t	⟨s⟪, e⟩, t⟫
read′	e|(e|t)	⟨s, ⟪s, e⟩, ⟪s, e⟩, t⟫⟩
book′	e|t	⟨s, ⟪s, e⟩, t⟫
j	e	⟨s, e⟩

Note that for reasons of simplicity, we define the domain/range structure of categories like that of types, namely D|R (Montague defines categories as R/D, Ajdukiewicz and Bar-Hillel as R/D; see section 4, (13), for exact definitions). Due to the systematic category/type correspondence in our system, we may say that, e.g. *read′* is of a type corresponding to e|(e|t), rather than specifying the type directly as ⟨s, ⟪s, e⟩, ⟪s, e⟩, t⟫⟩ which would be equivalent but more cumbersome. See section 4 for the definition of the category/type correspondence in our strictly intensional system.

This brief discussion of the built-up of an analyzed translation shows the importance of the type structure in connection with functional abstraction by means of lambda-operators and variables. Readers not familiar with these features of intensional logic should consult standard text-books on Montague grammar, such as Loebner (1976), Link (1979), or Dowty *et al.* (1981). Our particular version of strictly intensional logic, which is used in our examples, is explicitly defined in Hausser(1978b, 1979b)[3]

The analyzed translation of a surface sentence is a rather complicated logical formula. The reason for the degree of complication is that in the analyzed translation the logic is built up to mirror the surface structure word by word (the reason for this is explained in Hausser 1978b). The corresponding ***meaning*¹*-formula***, on the other hand, represents the logically most simple way to represent the meaning in question. The meaning¹-formula is logically equivalent to respective analyzed translation, and derived from the analyzed translation by means of so-called ***lambda reduction.*** Consider (6), where the meaning¹-formula of (1) is derived step by step from the analyzed translation of (1) given in (5).

(6) $$\hat{p}[p]((\hat{P}_1\hat{P}_0[P_0\hat{x}[P_1\hat{y}[read'(x,y)]]](\hat{Q}\hat{P}Vz[Q(z) \wedge P(z)](book')))(\hat{P}P(j)))$$

$$\hat{p}[p]((\hat{P}_1\hat{P}_0[P_0\hat{x}[P_1\hat{y}[read'(x,y)]]](\hat{P}Vz[book'(z) \wedge P(z)]))(\hat{P}P(j)))$$

$$\hat{p}[p]((\hat{P}_0[P_0\hat{x}[\hat{P}Vz[book'(z) \wedge P(z)](\hat{y}[read'(x,y)])]])(\hat{P}P(j))))$$

$$\hat{p}[p]((\hat{P}_0[P_0\hat{x}(Vz[book'(z) \wedge \hat{y}[read'(x,y)](z)])]])(\hat{P}P(j)))$$

$$\hat{p}[p]((\hat{P}_0[P_0\hat{x}(Vz[book'(z) \wedge read'(x,z)])])(\hat{P}P(j)))$$

$$\hat{p}[p]((\hat{P}P(j))(\hat{x}(Vz[book'(z) \wedge read'(x,z)])))$$

3) The version of strictly intensional logic defined in Hausser (1979b) differs from that of (1978b) in that in (1979b) intensions are defined as functions from speaker-world-time triples (rather than world-time pairs) into extensions and in that a believe-operator is defined. The two systems have in common, however, that the recursion of the logic is defined uniformely on the intensional level, so that no intension operator is needed.

$$\hat{p}[p](\hat{x}(\bigvee z[book'(z) \wedge read'(\overset{\uparrow}{x},z)])(\underset{\sqcup}{j}))$$

$$\hat{p}[\overset{\uparrow}{p}](\bigvee z[book'(z) \wedge read'(j,z)])$$

$$\bigvee z[book'(z) \wedge read'(j,z)]$$

The process of lambda-reduction is again tightly constrained by the type-structure of intensional logic: the argument of a lambda-expression must agree in its type with the variable it is to replace. In order to facilitate understanding of the lambda-reductions in (6), the variable and the argument are correlated ~~in each~~ line by means of means of the ‘$\uparrow \sqcup$’-marker.

Comparison of the first and the last line of (6) shows that all lambda-operators and correlated variables of the analyzed translation formula (first line) have disappeared in the resulting meaning^1-formula (last line). What then is the purpose of building up complex logical structures in the analyzed translations of surface compositional grammar, which nothing but evaporate in the process of lambda-reducing the analyzed translation formula to the equivalent meaning^1-formula? Consider the following three reasons. The function of the lambda build-up in the analyzed word translations is

(a) to isolate the translation of each surface word (compare for example the analyzed translation of $a_{CN\backslash T}$ and its distributed remainders in the meaning^1-formula),[4]

4) Russell(1905) claimed that natural language is illogical in the sense that it fails to mirror the proper logical structure. For example, a definite description is treated by Russell as something constituted by two underlying propositions. However, it is not the fault of natural language that it doesn't reflect Russell's postulated underlying form (which is inadequate to begin with, as shown in Hausser 1976). Rather, it is the task of the logician to provide proper logical forms for natural language. A type logic with lambda-calculus is suited to give a coherent translation of the word *the*:

$$\begin{array}{l} the_{CN\backslash T} \text{ (sg.)} \\ \quad | \\ \hat{Q}\hat{P}\wedge x\varepsilon[Q(x)]^1P(x) \end{array}$$

Russell could not arrive at such a solution, because at the time type logic with lambda-calculus had not been invented, let alone been applied to natural language (this was done by Montague in the late sixties).

(b) to allow functors of the lowest possible type on the level of the meaning^1-formula,[5)] while at the same time the analyzed word translations correspond in type to the combinatorially induced surface categories (compare for example to analyzed translation of $\textit{\textbf{reads}}_{T|(T|t)}$ and what remains of it in the meaning^1-formula, namely ***read′***, which corresponds to the category $e|(e|t)$, and

(c) to mediate between the functor/argument structure of the words on the surface and the usually different functor/argument structure of their remainders on the level of the meaning^1-formula.

More generally speaking, the purpose of the analyzed translation formula with its lambda-operators and variables is that of a ***transformer.*** As explained in Hausser (1978b), the analyzed translation must be ***structurally equivalent*** to the natural surface and ***logically equivalent*** to the meaning^1-formula. The structural equivalence between the surface and the analyzed translation, which is achieved by means of the lambda-operators and variables, is essential to guarantee that the step from the surface to the analyzed translation is ***meaning-preserving*** (preservation of meaning in terms of preservation of structure). Once we have translated a surface into a structurally equivalent analyzed translation, the denotation preserving

5) This is a notable feature of our semantic analysis in comparison with Montague (PTQ). Montague used the lowest possible type in his ingenious translation of proper names e.g. $\textit{\textbf{John}}_T$, and $\textit{\textbf{be}}_{T|(e|t)}$, but in most cases he

$$\begin{array}{c} \textit{\textbf{John}}_T \\ | \\ \hat{P}[P(j)] \end{array} \quad \begin{array}{c} \textit{\textbf{be}}_{T|(e|t)} \\ | \\ \hat{P}\hat{x}[P[\hat{y}(x=y)]] \end{array}$$

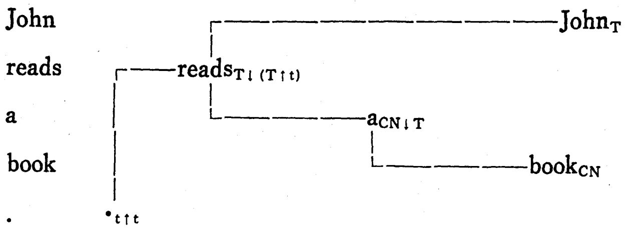
did not. Consider for example a transitive verb like ***reads***:

$$\begin{array}{c} \text{(a) } \textit{\textbf{reads}}_{T|(e|t)} \text{ (Montague)} \\ | \\ \hat{P}\hat{x}[\textit{\textbf{read}}'(x, P)] \\ | \\ T|(e|t) \end{array} \qquad \begin{array}{c} \text{(b) } \textit{\textbf{reads}}_{T|(T|t)} \text{ (Hausser)} \\ | \\ \hat{P}_1\hat{P}_0[P_0\hat{x}[P_1\hat{y}[\textit{\textbf{read}}'(x, y)]]] \\ | \\ e|(e|t) \end{array}$$

(b) uses an unanalyzed constant *read′* of a type corresponding to a lower category than (a), yet the whole translation (b) corresponds to a surface category $T|(T|t)$. Thus (b), in contrast to (a), treats the subject and the object term both as arguments to the transitive surface verb. The indicated method of translation is also used to distinguish transparent (e.g. ***finds***) versus opaque (e.g. ***seeks***) object positions of transitive verbs (cf. Hausser 1979b).

process of lambda-reduction renders the meaning[1]-formula. The meaning[1]-formula is not structurally equivalent to the surface, but presents the meaning in its logically simplest form (preservation of meaning in terms of preservation of denotation).

In summary, let us illustrate the principles of syntactico-semantic analysis in surface compositional grammar in terms of the orthogonal tree (7), which differs from the orthogonal tree (3) only in that it uses analyzed rather than unanalyzed word translations and presents in addition the resulting meaning[1]-formula of sentence (1). An analysis like (7) is also called a *canonical analysis* in our system.

(7)


$\hat{p}[p]((\hat{P}_1\hat{P}_0[P_0\hat{x}[P_1\hat{y}[\text{read}'(x,y)]]](\hat{Q}\hat{P}\bigvee z[Q(z) \wedge P(z)](\text{book}')))$
$(\hat{P}P(j)))$ meaning[1]-formula: $\bigvee z[\text{book}'(z) \wedge \text{read}'(j,z)]$

(7) characterizes the relation between the surface of sentence (1) and its literal meaning. What is notable about this particular surface compositional way to present the surface/meaning-mapping is that

(a) it is completely formal and well-defined—in contrast to the common practice to provide logical formulas which are supposed to characterize the meaning of surface expressions without any formal translation (e.g. Quine (1960) or standard logic books),

(b) no transformations are employed in (7); the only transformation-like process used in surface compositional analysis is lambda-reduction, which is a well-defined, denotation-preserving logical operation,

(c) the analysis is completely compositional, i.e. no syncategorematic operations are employed and the only mode of surface syntactic combination is categorial composition.

The question in what sense a logical formula like the analyzed translation or the meaning[1]-formula constitutes a reasonable analysis of the literal meaning of the respective natural language expression is discussed in other places, e.g. Hausser (1982).

3. *The structure of the lexicon*

The basic units of surface compositional grammar are ***analyzed surface words***. The analysis of a word consists of three parts:

(i) The description of the surface form of the word. Here we will use the standard orthographic representation for the sake of simplicity.
(ii) The description of the combinatorics (syntax) of the word, encoded in the category.
(iii) The description of the literal meaning (semantics) of the word, specified in terms of a translation into intensional logic.

These three features of a word analysis are jointly represented in the standard form of a so-called ***molecule*** (Hausser 1981). Consider (8), where the by now familiar word ***reads*** is represented as a molecule:

(8)
$$\begin{array}{c} \textbf{\textit{reads}}_{\mathrm{T}\downarrow(\mathrm{T}\uparrow\mathrm{t})} \\ | \\ \hat{P}_1\hat{P}_0[P_0\hat{\mathrm{x}}[P_1\hat{\mathrm{y}}[\mathrm{read}'(\mathrm{x},\mathrm{y})]]] \end{array}$$

Molecules are defined as elementary orthogonal trees and serve as the basic input to the rules of ORTAX (cf. section 4).

The lexicon of surface compositional grammar consists of sets of molecules, whereby each set $\mathrm{M_A}$ contains molecules of the same surface category A. Quite obviously, this type of lexical analysis is in line with the traditional view, according to which a lexicon should characterize the words of a language in terms of their word class (category) and their meaning (translation). We do not share the view prevalent in some linguistic circles (e.g. Dowty 1979), however, that the domain of the lexicon are irregular or non-productive phenomena, while the purpose of the syntax is taken to be the description of regular or productive features of language. Rather, we take a structural approach according to which the domain of the lexicon is the description of words and the domain of the syntax is the composition of words. Thus both the syntax and the lexicon have a share of both regular and irregular phenomena to describe in surface compositional grammar.

Let us consider two cases in point, (i) the treatment of an idiom like *John kicked the bucket* and (ii) the decomposition of a word like $\textit{kills}_{T\downarrow(T\uparrow t)}$. Regarding idiomatic expressions, they are characteristically not irregular on the syntactic or semantic level. Rather, they consist of standard words, composed in a standard fashion, and they have the *standard literal meaning* to be expected. The crucial property of idioms resides in their pragmatics: in addition to the normal use conditions associated with their regularily composed surface they have acquired special, 'frozen' use conditions. It is therefore on the level of the speaker-meaning (meaning^2) that 'John kicked the bucket' is equivalent (or may be equivalent) to 'John died'. On the level of literal meaning (meaning^1), however, 'John kicked the bucket' is equivalent to something like 'John hit the pail with his foot'. Thus, while the frozen use conditions of idioms must be described, it is not in the lexicon, but within a theory of pragmatics, which like any other grammatical component will consist of a set of general principles or rules and lists of exceptions or special cases (the latter being the place for the description of idioms).

Next let us consider the decomposition of a word like *kills*. While the relation between the meaning of *x kills y* and *x CAUSES y to BECOME NOT ALIVE* clearly must be specified, it is not in the syntax but in the lexicon. As mentioned already in section 2, we distinguish three levels of semantic analysis, namely

(i) unanalyzed translations
(ii) analyzed translations, and
(iii) elementary translations,

which correspond to three versions of intensional logic, called

(i) the surface logic,
(ii) the root logic, and
(iii) the universal logic,

respectively. Consider (9a, b, c), where these three levels of semantic analysis are illustrated in form of three alternative molecule-representations of the word $\textit{\textbf{kills}}_{T\downarrow(T\uparrow t)}$.

(9a)
$$\begin{array}{l} \textit{\textbf{kills}}_{T\downarrow(T\uparrow t)} \\ | \\ \textit{\textbf{kills}}^{\circ} \end{array}$$

(9b)
$$\begin{array}{c} \textit{\textbf{kills}}_{T\downarrow(T\uparrow t)} \\ | \\ \hat{P}_1\hat{P}_0[P_0\hat{x}[P_1\hat{y}[\mathrm{kill}'(x, y)]]] \end{array}$$

(9c)
$$\begin{array}{c} \textit{\textbf{kills}}_{T\downarrow(T\uparrow t)} \\ | \\ \hat{P}_1\hat{P}_0[P_0\hat{x}[P_1\hat{y}[(\hat{m}\hat{n}[\textit{CAUSE}(m\hat{Q}\,\textit{Vz}[Q(z)=\textit{\textbf{BECOME}}(n,\hat{P}[P(n) \\ =\sim\textit{ALIVE}(n)])]])(x,y)]]] \end{array}$$

The relation between an unanalyzed translation like (9a) and the corresponding analyzed translation (9b) has already been discussed in the previous section. Logical constants like ***kill'*** or ***read'*** are called roots, which is the reason why the language of analyzed translations is called the root logic. Roots must be distinguished from unanalyzed translations. For instance, the unanalyzed translation ***kills***° corresponds in type to the category $T\downarrow(T\uparrow t)$, whereas the root ***kill'*** corresponds in type to the category $e|(e|t)$.

While in the most basic version of the lexicon of surface compositional grammar presented so far words are stored as complete molecules, a so-called ***horizontally refined lexicon*** (Hausser 1981) constitutes a system where ***molecules are derived from roots*** by means of lexical derivation rules. Consider (10), which illustrates an inflectional lexical derivation rule.

(10) ***Lexical derivation of the past tense of transitive verbs***

If $\alpha'\in RT_{<s,f(e|(e|t))>}$, then $L_2(\alpha')$ is $\begin{array}{c}\Sigma(\alpha)_{T\downarrow(T\uparrow t)} \\ | \\ \hat{P}_1\hat{P}_0[HP_0\hat{x}[P_1\hat{y}[\alpha'(x, y)]]]\end{array}$ if and only

if $\alpha'=\textit{\textbf{kill}}'$, $\textit{\textbf{kiss}}'$, $\textit{\textbf{cook}}'$, etc., whereby $\Sigma(\alpha)=\alpha\textit{\textbf{-ed}}$; or
$\alpha'=\textit{\textbf{find}}'$, whereby $\Sigma(\alpha)=\textit{\textbf{found}}$; or
$\alpha'=\textit{\textbf{break}}'$, whereby $\Sigma(\alpha)=\textit{\textbf{broke}}$; etc.

(otherwise $L_2(\alpha')$ is undefined).

The rule defined in (10) generates a class which contains molecules such as

$$\begin{array}{ccc} \textit{\textbf{killed}}_{T\downarrow(T\uparrow t)} & \text{or} & \textit{\textbf{found}}_{T\downarrow(T\uparrow t)}. \\ | & & | \\ \hat{P}_1\hat{P}_0[HP_0\hat{x}[P_1\hat{y}[\text{kill}'(x,y)]]] & & \hat{P}_1\hat{P}_0[HP_0\hat{x}[P_1\hat{y}[\text{find}'(x,y)]]] \end{array}$$

What the elements of this molecule class have in common is characterized in terms of the schematic molecule in the first line of the rule stated in (10). What distinguishes the molecules generated by the rule, on the other hand, is specified in terms of explicit listing: the different roots in the translations and the corresponding surface forms. ***Regular*** surface forms are thereby characterized by the redundancy rule like formulation in the second line of (10).

Note that the rule type illustrated in (10) accounts also for the so-called derivational phenomena in the lexicon of natural languages. Thus molecules like ***break***, ***breaker***, ***breakable***, ***unbreakable***, etc. (derivation) are derived from the same root ***break'*** as the molecules ***break***, ***breaks***, ***broke***, ***breaking***, etc. (inflection). The purpose of the outlined lexical derivation of molecules from sets of roots RT is (i) to present the information in question in a brief and perspicuous manner, and (ii) to provide analogical models for the formation of new words.

We have seen now two reasons for defining a root logic in addition to the surface logic: (i) the conversion of unanalyzed translation formulas into meaning^1-formulas and (ii) the capturing of word-semantic generalizations in the lexicon. But while translation into the root logic is sufficient for the treatment of all the traditional syntactic generalizations and semantic relations in a given natural language, it does not constitute a ***complete*** semantic analysis. Only the meaning of the operators and variables in the root logic is explicitly defined in terms of meta-language definitions, but the roots themselves are still treated as unanalyzed constants. That is, the roots denote functions the domain/range structure of which is specified in terms of the type of the root-functor, but the definition (Zuordnungs-bestimmung) is left open.

In a complete semantic analysis, however, even the roots have to be defined at some point. As before in the case of the unanalyzed translations of the surface logic, there are two theoretical possibilities. Either we specify the definition of the roots *directly* (in terms of a metalanguage definition as in the case of operators like $\sim, \wedge, \Lambda, \Box$, etc. or in terms of the denotation function F, which specifies the extension of a constant like j for each index of the model structure). Or we treat roots as abbreviations of formulas consisting of even more elementary logical symbols.

Again, we choose the second possibility. The reasons are (i) that roots do not denote universal concepts, but represent semantic concepts which are dependent on and characteristic of the surface language; (ii) that there are far too many roots in a given surface language as to make direct meta-language definitions practical; (iii) that a direct definition of roots would miss certain semantic interrelations between roots (by transposing them into the meta-language). Consider for example the roots ***kill'***, ***assassinate'***, and ***die'***.

The roots ***kill'*** and ***assassinate'*** clearly share some properties of meaning. Yet both roots are necessary in the analysis of English, given the set up of lexical derivation rules in surface compositional grammar. The reason is that the molecules ***kill***, ***kills***, ***killed***, ***killing*** and the molecules ***assassinate***, ***assassinates***, ***assassinated***, ***assassinating*** are generated by the same four lexical derivation rules (assuming that the roots ***kill'*** and ***assassinate'*** are both listed in them). A similar set of rules for intransitive verbs will generate the molecules ***die***, ***dies***, ***died***, ***dying*** and the molecules ***dream***, ***dreams***, ***dreamed***, ***dreaming*** from the roots ***die'*** and ***dream'***, respectively. Thus the degree of semantic decomposition in the analyzed translation that produces the roots as unanalyzed constants is determined by the surface, more specifically the word paradigms of the natural language in question in interaction with certain semantic properties of these words. It is consequently no surprise that the analyzed translations of different natural languages may have to use different roots.

So how should the meaning relations between the roots ***kill'***, ***assassinate'***, and *die′* be formally captured? Consider (11), where these roots are semantically defined in terms of equivalent formulas of Universal Logic.

(11) $\text{kill}' = \hat{m}\hat{n}[\text{CAUSE}(m, \hat{Q}Vz[Q(z) = \text{BECOME}(n, \hat{P}[P(n) = {\sim}\text{ALIVE}(n)])])]$

$\text{assassinate}' = \hat{m}\hat{n}[\text{CAUSE}(m, \hat{Q}Vz[\text{IMPORTANT}(z) \wedge Q(z) = \text{BECOME}(n, \hat{P}[P(n) = {\sim}\text{ALIVE}(n)])])]$

$\text{die}' = \hat{n}[\text{BECOME}(n, \hat{P}[P(n) = {\sim}\text{ALIVE}(n)])]$

The elementary UL-functors CAUSE, BECOME, ALIVE, and IMPORTANT are assumed to be semantically universal. It is hoped that their number can be kept reasonably small, while permitting definition of the different roots from different natural languages in terms of expilicit UL-formulas. The elementary UL-functors will be defined in terms of certain basic parameters of the speaker simulation device (SID), such as geometric structures in three-dimensional space, time-structure, the color spectrum, temperature, etc. The definition of the UL-functors thus brings us to a different topic, namely, the ontology of model-theory, which is discussed in Hausser (1982).

4. *Definition of ORTAX and sample derivations*

What has been established so far? It was argued in section 1 that the syntax of a generative grammar has a very simple task in the description of natural language. This task is (i) to determine the function/argument structure of the surface constituents and (ii) to correlate the surface order with the standard functor/argument notation of standard logic. The basic surface constituents are the surface words, the semantics of which was discussed in section 2 and 3. This excursion to the level of analyzed translations and the structure of the lexicon was intended to dispell any suspicion that traditional syntactic and semantic generalizations are untreated in our system. The fact that the syntax is simple and the analysis of words (molecules) is complex in our theory reflects (a) our adherence to the

Fregean principle which we apply to the surface (and not some underlying level as in, e.g. Cooper 1975), and (b) the assumption that words should be analysed as units with intricate (i.e. sub-categorial) and semantic properties. The goal of surface compositional analysis is to let all syntactico-semantic properties of a sentence originate in the lexically given syntactico-semantic properties of the words and the interaction of these properties.

Below we will take another look at analyzed word translations in order to motivate some of the sample derivations. But for now let us return to the main theme of this paper. namely the definition of our orthogonal syntax or ORTAX. Due to our present focus on syntax, we are interested mainly in two of the three parts of a molecule, namely (i) the surface and (ii) the sub-category, but not (iii) the analyzed or elementary translation. Therefore we will for the moment assume molecules which have only ***unanalyzed*** translations, as illustrated in (12):

(12) $\begin{array}{l} \textbf{\textit{reads}}_{\mathrm{T}\downarrow(\mathrm{T}\uparrow\mathrm{t})} \\ \quad | \\ \textbf{\textit{reads}}^{\circ} \end{array}$

The first task in defining our syntax is to specify the sub-categorial system of molecules, since these sub-categories uniquely specify the combinatorial properties of expressions in our system. The sub-categorial system is based on a system of ***categories***, which are defined in (13) below:

(13) Def. 1 *The categories of ORTAX*

(i) e ε CAT

(ii) t ε CAT

(iii) If A, B ε CAT, then (A|B) ε CAT.

(iv) Nothing is in CAT, except as specified in (i-iii) above.

This definition is like Montague's (PTQ), except that a complex category is written as A|B rather than Montague's B/A. Thus in our system categories (e.g. e|t) and types (e.g. ⟨s,⟨⟨s,e⟩,t⟩⟩) use the same Domain/Range serialization in the notation, namely D|R.

(14) Def. 2 *The sub-categories of ORTAX*

(i) If A ε CAT, then $\overset{m}{A}$ ε SCAT, where $m \geqq 0$.

Convention: If $m=0$, then $\overset{o}{A}=A$.

(ii) If $A \mid B$ ε SCAT, then also $A \downarrow B$, $A \uparrow B$, $\underline{A} \downarrow B$, $\underline{A} \uparrow B$, $\underline{\underline{A}} \downarrow B$, $\underline{\underline{A}} \uparrow B$ ε SCAT.

(iii) Nothing is in SCAT, except as specified in (i,ii) above.

The first part of this definition, i.e. (i), is a generalization of Montague's double slash methoed (e.g. A/B versus A//B) for doubling the categories of a given domain/range structure. The index is superior in that it allows denumerably many sub-categories of a given category, including *e* and *t*. This multiplication of sub-categories will have numerous applications, such as the distinction between different cases, number, and other surface markings which control the surface composition beyond the frame of the functor/argument structure. (A similar proposal to this effect was made in Kiyong Lee (1982).) The second part of the definition, i.e. (ii), is in some respects a variation of Bar Hillel's categories $\overset{R}{\underset{D}{\leftarrow}}$ and $\overset{R}{\underset{D}{\rightarrow}}$, except that we use a horizontal $D \uparrow R$ and $D \downarrow R$, notation. The diacritics '$\underline{\ \ }$' and '$\underline{\underline{\ \ }}$' exhibited by some of the sub-categories control the surface-serialization of so-called discontinuous elements, a phenomenon Bar-Hillel could find no solution for in a pure (i.e. context-free) categorial grammar.

Now that we have defined the sub-categorial system in Def. 2, let us assume a lexicon where the molecules used in our examples are specified in the way illustrated in (15).

(15) $$\begin{array}{lllll} \mathit{John}_{T}, & \mathit{reads}_{T\downarrow(T\uparrow t)}, & \mathit{a}_{CN\downarrow T}, & \mathit{book}_{CN}, & \bullet_{t\uparrow t} \\ | & | & | & | & | \\ \mathrm{John}^{o} & \mathrm{reads}^{o} & \mathrm{a}^{o} & \mathrm{book}^{o} & \bullet^{o} \end{array}$$

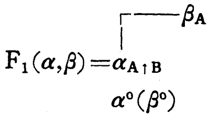
The categorial symbols T and CN are defined as abbreviations of $(e|t)|t$ and $e|t$, respectively (just as in PTQ).

It remains to specify the rules which combine molecules like those illustrated in (15) according to their sub-category into orthogonal trees.

(16) Def. 3 *The rules of ORTAX*

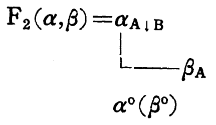
ST0. $M_Z \subseteq P_Z$, for every $Z \in$ SCAT.

ST1.1 If $\alpha \in P_{A\uparrow B}$ and $\beta \in P_A$, then $F_1(\alpha,\beta) \in P_B$ and

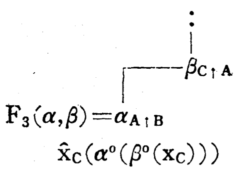


where α^o and β^o are the unanalyzed translations of α and β, respectively.

ST1.2 If $\alpha \in P_{A\downarrow B}$ and $\beta \in P_A$, then $F_2(\alpha,\beta) \in P_B$, and

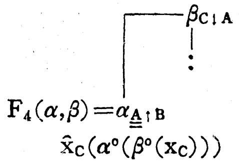


ST2.1 If $\alpha \in P_{\underline{\underline{A}}\uparrow B}$ and $\beta \in P_{C\uparrow A}$, then $F_3(\alpha,\beta) \in P_{C:\uparrow B}$, and

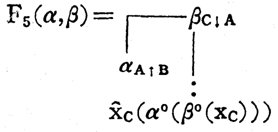

$\beta_{C\uparrow A}$
$F_3(\alpha,\beta)=\alpha_{A\uparrow B}$
$\hat{x}_C(\alpha^o(\beta^o(x_C)))$

where x_C is a variable of type $\langle s, f(C)\rangle$.

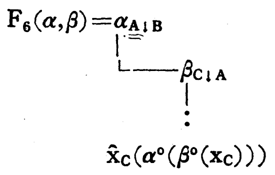
ST2.2 If $\alpha \in P_{\underline{\underline{A}}\uparrow B}$ and $\beta \in P_{C\downarrow A}$, then $F_4(\alpha,\beta) \in P_{C:\downarrow B}$, and


$\beta_{C\downarrow A}$
$F_4(\alpha,\beta)=\alpha_{\underline{\underline{A}}\uparrow B}$
$\hat{x}_C(\alpha^o(\beta^o(x_C)))$

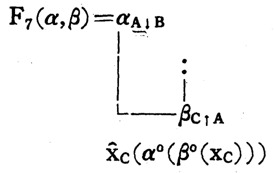
ST2.3 If $\alpha \in P_{\underline{\underline{A}}\uparrow B}$ and $\beta \in P_{\underline{C}\downarrow A}$, then $F_5(\alpha,\beta) \in P_{C:\downarrow B}$, and



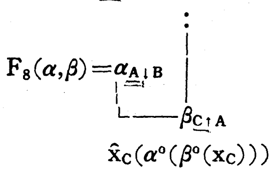
ST2.4 If $\alpha \in P_{\underline{\underline{A}}\downarrow B}$ and $\beta \in P_{C\downarrow A}$, then $F_6(\alpha,\beta) \in P_{C:\downarrow B}$, and


$F_6(\alpha,\beta)=\alpha_{\underline{\underline{A}}\downarrow B}$
$\beta_{C\downarrow A}$
$\hat{x}_C(\alpha^o(\beta^o(x_C)))$

ST2.5 If $\alpha \in P_{\underline{\underline{A}}\downarrow B}$ and $\beta \in P_{C\uparrow A}$, then $F_7(\alpha,\beta) \in P_{C:\uparrow B}$, and


$F_7(\alpha,\beta)=\alpha_{\underline{\underline{A}}\downarrow B}$
$\beta_{C\uparrow A}$
$\hat{x}_C(\alpha^o(\beta^o(x_C)))$

ST2.6 If $\alpha \in P_{\underline{\underline{A}}\downarrow B}$ and $\beta \in P_{\underline{C}\uparrow A}$, then $F_8(\alpha,\beta) \in P_{C:\uparrow B}$, and


$F_8(\alpha,\beta)=\alpha_{\underline{\underline{A}}\downarrow B}$
$\beta_{\underline{C}\uparrow A}$
$\hat{x}_C(\alpha^o(\beta^o(x_C)))$

ST3. If $\zeta \in P_{C:\uparrow B}$ or $P_{C:\downarrow B}$ and $\gamma \in P_C$, then $F_9(\zeta, \gamma) \in P_B$ and

$$F_9(\zeta, \gamma)=\zeta\text{:}——\gamma$$
$$\zeta^o \quad (\gamma^o)$$

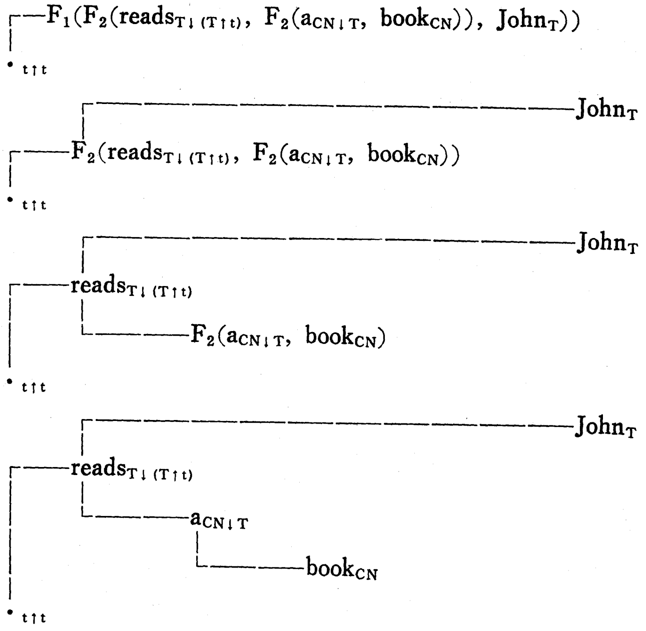
The rules ST 1～3 specify (i) the categories of the two input expressions, (ii) the category of the output expression, (iii) the resulting local orthogonal tree, and (iv) the associated logical structure. As mentioned before, the rules of definition 3 (cf. (16) above) are called ST.-rules because they combine the purposes of Montague's S- and Montague's T-rules (compare PTQ).

The derivation of an orthogonal tree is executed in three stages. The first is the derivation of an indexed bracketing, as exemplified in (17).

(17) $F_1(._{t\uparrow t}$, $F_1(F_2(\text{reads}_{T\downarrow(T\uparrow t)}$, $F_2(\text{a}_{CN\downarrow T}$, $\text{book}_{CN})),\ \text{John}_T))$

The second stage is to 'unfold' the orthogonal tree by applying all orthogonal F-operations of the indexed bracketing. This is done in a step by step manner:

(18)



$F_1(F_2(\text{reads}_{T\downarrow(T\uparrow t)}, F_2(a_{CN\downarrow T}, \text{book}_{CN})), \text{John}_T))$
$\cdot_{t\uparrow t}$
John_T
$F_2(\text{reads}_{T\downarrow(T\uparrow t)}, F_2(a_{CN\downarrow T}, \text{book}_{CN}))$
$\cdot_{t\uparrow t}$
John_T
$\text{reads}_{T\downarrow(T\uparrow t)}$
$F_2(a_{CN\downarrow T}, \text{book}_{CN})$
$\cdot_{t\uparrow t}$
John_T
$\text{reads}_{T\downarrow(T\uparrow t)}$
$a_{CN\downarrow T}$
book_{CN}
$\cdot_{t\uparrow t}$

The third stage is to supply for each of the above steps the corresponding translation as indicated in the respective rule. In the case of the example analyzed in (17) and (18) the resulting logical form is that of (19):

(19) $\cdot^{\circ}((\text{reads}^{\circ}(a^{\circ}(\text{book}^{\circ})))(\text{John}^{\circ}))$

Let us turn now to a more difficult example, namely (19), which contains the discontinuous element *look up*.

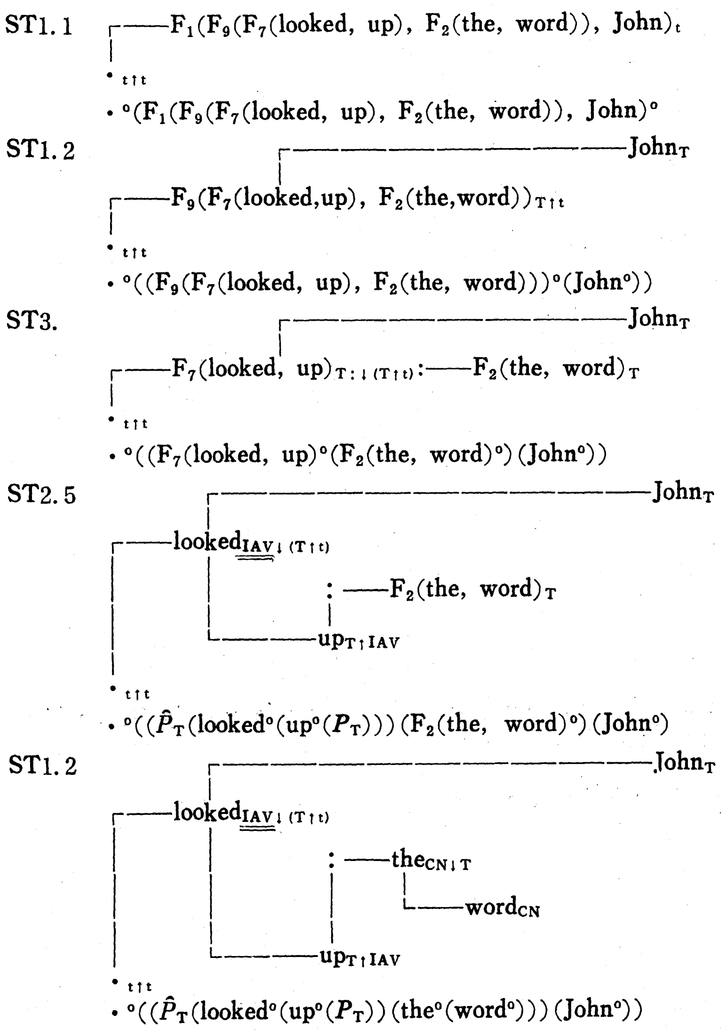
(19) John looked the word up.

Assuming that *looked* and *up* are sub-categorized as *looked*$_{\underline{\underline{IAV}}\downarrow(T\uparrow t)}$ and *up*$_{T\uparrow IAV}$, we may form the following indexed bracketing on the basis of definition 3.

(20) $F_1(\cdot_{t\uparrow t}, F_1(F_9(F_7(\text{looked}_{\underline{\underline{IAV}}\downarrow(T\uparrow t)}, \text{up}_{T\uparrow IAV}), F_2(\text{the}_{CN\downarrow T}, \text{word}_{CN}), \text{John}_T))$

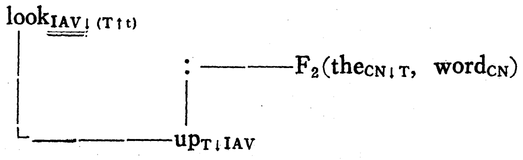
Next we unfold the corresponding orthogonal tree by applying all the orthogonal F-operations specified in (20). Note that in (21) the derivation of the unanalyzed translation is developed in a step by step manner, in accordance with the specifications of the ST rules involved. For complete sub-categorial specifications the molecules see the final tree of derivation (21).

(21)


ST1. 1
$F_1(F_9(F_7(\text{looked, up}), F_2(\text{the, word})), \text{John})_t$
$t\uparrow t$
$^\circ(F_1(F_9(F_7(\text{looked, up}), F_2(\text{the, word})), \text{John})^\circ$
ST1. 2
John_T
$F_9(F_7(\text{looked,up}), F_2(\text{the,word}))_{T\uparrow t}$
$t\uparrow t$
$^\circ((F_9(F_7(\text{looked, up}), F_2(\text{the, word})))^\circ(\text{John}^\circ))$
ST3.
John_T
$F_7(\text{looked, up})_{T:\downarrow(T\uparrow t)}$: $F_2(\text{the, word})_T$
$t\uparrow t$
$^\circ((F_7(\text{looked, up})^\circ(F_2(\text{the, word})^\circ)(\text{John}^\circ))$
ST2. 5
John_T
$\text{looked}_{\underline{\underline{IAV}}\downarrow(T\uparrow t)}$
: $F_2(\text{the, word})_T$
$\text{up}_{T\uparrow IAV}$
$t\uparrow t$
$^\circ((\hat{P}_T(\text{looked}^\circ(\text{up}^\circ(P_T)))(F_2(\text{the, word})^\circ)(\text{John}^\circ)$
ST1. 2
John_T
$\text{looked}_{\underline{\underline{IAV}}\downarrow(T\uparrow t)}$
: $\text{the}_{CN\downarrow T}$
word_{CN}
$\text{up}_{T\uparrow IAV}$
$t\uparrow t$
$^\circ((\hat{P}_T(\text{looked}^\circ(\text{up}^\circ(P_T))(\text{the}^\circ(\text{word}^\circ)))(\text{John}^\circ))$

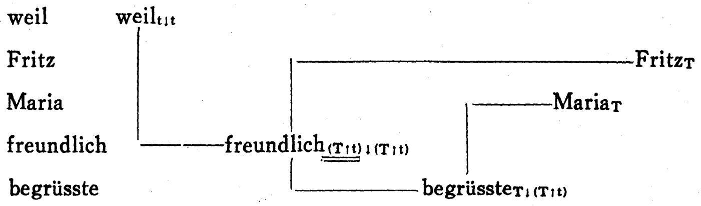
What is the basis to this formal solution to the problem of discontinuous elements? The problem of discontinuous elements is that two constituents which form a semantic unit (e.g. ***look up***) appear in non-adjacent positions on the surface. But what is meant by 'forming a semantic unit'? The most general interpretation one may give to this concept is to assign a kind of function/argument relation to the discontinuous constituents, e.g. (look° (up°)). Once this is done, we can still decide whether (i) to treat ***look up*** as a semantic unit by lexically equating (***look***°(***up***°)) with ***look up***° (this is possible due to the adjacency of the semantic constituents in question), or (ii) to treat e.g., ***look up***, as a ***bona fide*** function/argument structure of a certain kind. In German there are clear instances[6] of discontinuous elements which suggest the function/argument treatment, whereas ***look up*** may be regarded as an instance that is more suitable for the unitary treatment.

Our treatment of discontinuous elements as exemplified in (21) is limited to the task of correlating the surface constituents in the form of a functor/argument relation on the level of unanalyzed translations, e.g.


$\text{look}_{\underline{\underline{IAV}}\downarrow(T\uparrow t)}$
$F_2(\text{the}_{CN\downarrow T},\ \text{word}_{CN})$
$\text{up}_{T\downarrow IAV}$

$\hat{x}(\text{look}^\circ(\text{up}^\circ(x_T)))$ $\qquad$ $\text{the}^\circ(\text{word}^\circ)$

6) Consider for example the subordinate clause ***weil Fritz Maria freundlich begrüsste***, the discontinuous structure of which is characterized below:

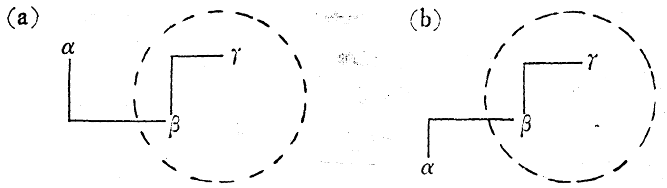

weil
Fritz
Maria
freundlich
begrüsste
$\text{weil}_{t\downarrow t}$
Fritz_T
Maria_T
$\text{freundlich}_{\underline{\underline{(T\uparrow t)}}\downarrow(T\uparrow t)}$
$\text{begrüsste}_{T\downarrow(T\uparrow t)}$

$\text{weil}^\circ(((\hat{x}_T[\text{freundlich}^\circ(\text{begrüsste}^\circ(x_T))](\text{Maria}^\circ))(\text{Fritz}^\circ))$

In this example, ***freundlich*** and ***begrüsste*** are clearly in a composite functional relation. Note that the surface order of this example could not be generated without the use of rule ST 2.5.

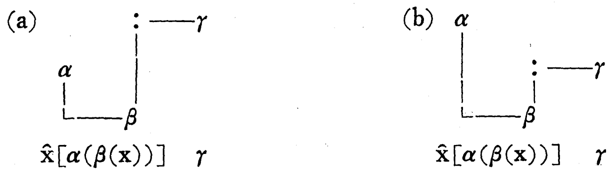
Why did discontinuous elements form such a problem for Bar-Hillel? In Bar-Hillel's system we can only form expressions of the following kind: $\alpha(\beta(\gamma))$. That is, first β is applied to γ and then α is applied to $\beta(\gamma)$. Our treatment of discontinuous elements is based on the possibility to change this order of applying functions to other functions. That is, we may first combine α and β into $\hat{x}[\alpha(\beta(x))]$ and then apply this complex function to γ, rendering $\hat{x}[\alpha(\beta(x))](\gamma)$ which is logically equivalent to $\alpha(\beta(\gamma))$.

What is the effect of this simple and logically perfectly well-defined change in the application of functions to other functions? In Bar-Hillel's system we can build orthogonal trees like (22):



It is a structural fact of the functor/argument relation that the argument expression, e.g. $\beta(\gamma)$, is treated as a unit, the internal structure of which cannot be changed. Therefore the only choice in Bar-Hillel's system is to add the α on the outside of the circled argument constituent in (22). In our system, on the other hand, we are allowed to restructure the functor/argument application. It is for this reason that we can also generate tree structures like (23):

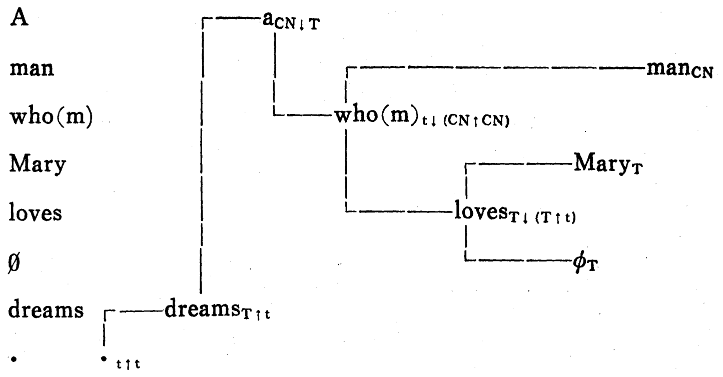
(23)



While the result of (23. b) would be derivable in a Bar-Hillel system (cf. (22.a)), the point of (23. b) is that the two higher functions are combined first and then applied to the argument. Thus (23. b), but not (22. a), is suitable to represent the constituent structure of *x looks y up*. The rules

ST 2.1–2.6 generate all 3! (=1·2·3=6) possible permutations of three expressions on the surface (maintaining a given logical structure). The rules are purely categorial in the sense that they involve no transformations. The proof that ORTAX is a strictly context-free system given in Hausser (1980c).

The enriched surface combinatorics of ORTAX is an important technical innovation leading to a successful surface compositional syntax, but it is not the only one. The other ingredient is the use of so-called Ø-surface molecules (zero-surface molecules) in combination with lambda-abstraction over functions expressed by formulas open in certain variables.[7] This feature of our grammar is best explained on the basis of a relevant example, such as the relative clause construction of (24).

(24)


$\bullet^{\circ}(\text{dreams}^{\circ}(\text{a}^{\circ}(\text{who(m)}^{\circ}((\text{loves}^{\circ}(\emptyset^{\circ}))(\text{Mary}^{\circ})))(\text{man}^{\circ})))$

(24) does not contain any discontinuous structures, i.e. it is derived like (17–19) solely on the basis of ST 1.1 and ST 1.2. The point of analysis (24) becomes transparent when we look at the presumed level of ***analyzed*** translations, specifically the translations of $\text{who(m)}_{t\downarrow(CN\uparrow CN)}$ and $\emptyset_T$, which are given in (25).

7) Zero-surface molecules may, of course, also be used without the special lambda abstraction over functions expressed by formulas open in certain variables. Such a case is for example the plural of the indefinite article discussed below.

(25a) $\underset{\hat{p}_X\hat{Q}\hat{x}[Q(x) \wedge p_X]}{\underset{|}{\boldsymbol{who(m)}_{t\downarrow(CN\uparrow CN)}}}$ $\qquad\qquad$ $\underset{\hat{P}P(x)}{\underset{|}{\emptyset_T}}$

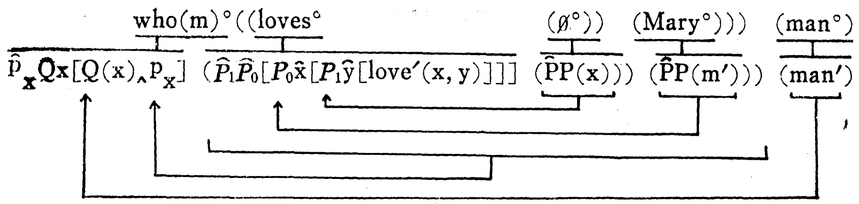
(25a) contains the variable p_X of a type corresponding to t, which is defined to range over ***propositions expressed by formulas open in x***. Thus $p_X[\ldots p_X \ldots]$ is and expression which takes only t-arguments that contain the unbound variable x. Such a suitable argument expression would be for example *love′(m′, x)*. If such an argument is supplied (cf. 26), lambda reduction may apply (cf. 27), whereby the variable x becomes bound by the $\hat{x}$ in the functor.

(26) $\hat{p}_X\hat{Q}\hat{x}[Q(x) \wedge p_X] \qquad (love'(m, x))$

(27) $\hat{Q}\hat{x}[Q(x) \wedge love'(m, x)]$

Consider now the analyzed translation of *man who(m) Mary loves*$_{CN}$ in (28):

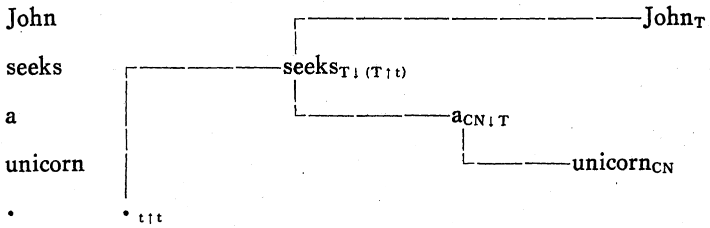
(28)



Systematic lambda-reduction (as indicated above) renders) $\hat{x}[man'(x) \wedge love'(m', x)]$, which is precisely the same result as Montague's, except that our analysis is completely surface compositional. That is, what Montague achieves by means of the special, non-compositional rule of relative clause formation (PTQ, S3), is handled in our system on the basis of the translation of ***who(m)*** and the use of a ∅-surface molecule. Note that ∅-surface molecules correspond to Montague's *he*$_N$-pronouns and are therefore no unfamiliar feature within a Montague-style analysis. The essential feature of our solution is the use of the variable p_X, which is perfectly within the means of standard logical methods.

Nevertheless, one might object to the use of Ø-surface molecules in analyses like (24) as a violation of a certain intuitive notion of surface compositionality. I would like to argue, however, (i) that Ø-surface molecules are independently necessary, due to certain wholes in linguistic paradigms, and (ii) that they will not impede the decidability of unanalyzed surface recognition, provided the recursive application of Ø-surface molecules (e.g. $Ø_{B|A}(Ø_{C|B}(Ø_{A|C}(\ldots)))$) is prevented by a constraint on the lexicon. The second point is discussed and formally proven in Hausser(1981), the first point may be shown by an example. Consider for instance the plural term ***books*** as in ***John reads books.*** In order to obtain such a T-phrase without overt quantifier we must either postulate a basic ***books***$_{T}$, in addition to basic ***books***$_{CN}$, or we assume a Ø-surface quantifier which translates as the plural of the indefinite article. It is clearly more general in this case to postulate a Ø-surface quantifier which combines with ***books***$_{CN}$ to render $Ø_{CN\downarrow T}$ (books$_{CN}$), thus contributing the usual quantificational machinery to the usual translation of books$_{CN}$. There is no motivation for postulating the basic term ***books***$_{T}$ in addition to the noun ***books***$_{CN}$, except for the theoretically inopportune wish to avoid Ø-surface molecules at all costs.

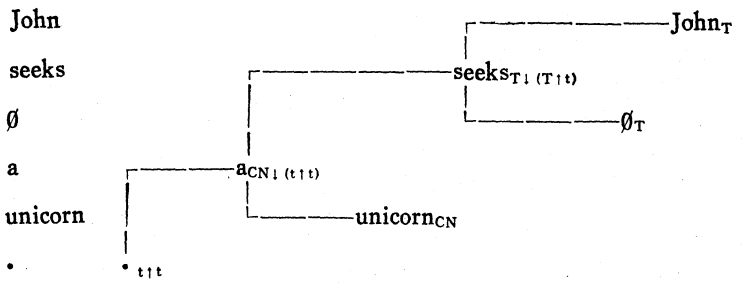
Without any additional requirements, the technique of analysis (24-28) is suitable also for the treatment of quantifier scope variation and unbounded dependencies. The first type of phenomenon is illustrated in (29) and (30).[8]

(29)


$\hat{p}[p]((\hat{P}_1\hat{P}_0[P_0\hat{x}[\text{seek}'_{T|(e|t)}(x, P_1)]](\hat{Q}\hat{P}\vee x[Q(x) \wedge P(x)](\text{unicorn}')))(\hat{P}P(j))$
lambda-reduction: $\text{seek}'\ (j, \hat{P}\vee x[\text{unicorn}'(x) \wedge P(x)])$

8) Since the unanalyzed translations may be deduced from the analyzed surface, they are omitted in these and the following canonical analyses.

(30)



$\hat{p}[p]((\hat{Q}\hat{p}_x \bigvee x[Q(x) \wedge p_x](\text{unicorn}'))(((\hat{P}_1\hat{P}_0[P_0\hat{x}[\text{seek}'(x, P_1)]](\hat{P}P(j))))$
lambda-reduction: $\bigvee x[\text{unicorn}'(x) \wedge \text{seek}'(j, \hat{P}P(x))]$

(29) is the so-called *de dicto* reading (narrow quantifier scope), while (30) represents the *de re* reading (wide quantifier scope, or more precisely: the quantifier of the object term is outside the scope of *seek'*). Disregarding the simplifications of our strictly intensional logic (as defined in Hausser (1978b, 1979b)), we have obtained the same result as Montague (PTQ) in the translation of these two examples. Again, we have achieved this result without the use of a special, non-compositional rule (in this case S14 of PTQ). Note that the wide scope quantifier in (30) corresponds to the category CN↓(t↑t). The two analog derivations of *John finds a unicorn* will render logically equivalent translations (as desired) without the use of any meaning-postulates, provided *finds*$_{T\downarrow(T\uparrow t)}$ is analyzed as in (31a):

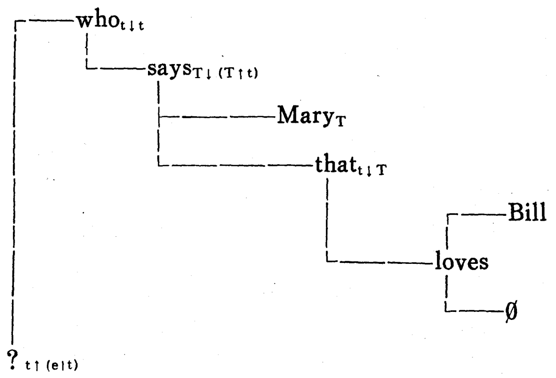
(31a) *finds*$_{T\downarrow(T\uparrow t)}$
$\hat{P}_1\hat{P}_0(P_0\hat{x}[P_1\hat{y}[\text{find}'(x, y)]]]$

(31b) *seeks*$_{T\downarrow(T\uparrow t)}$
$\hat{P}_1\hat{P}_0[P_0\hat{x}[\text{seek}'(x, P_1)]]$

The simple proof is left to the interested reader. Note that (31a) and (31b) exhibit again our principle of using elementary constants (roots) of the lowest possible type. In the case of *find'*, it corresponds to e|(e|t), in the case of seek', it corresponds to T|(e|t). Note that the surface category is nevertheless the same in (31a) and (31b).

(32) below is a canonical analysis of an example containing a so-called unbounded dependency.

(32) Who
says
Mary
that
Bill
loves
Ø
?



$\hat{p}_y\hat{y}[p_y]\hat{p}_x[\hat{x}p_x(y)]((\hat{P}_0\hat{P}_1[P_0\hat{x}[P_1\hat{y}[say'(x, y)]]])(\hat{P}P(m)))(\hat{p}\hat{P}Vx$
$[P(x)=p](love'(b, x)))$

lambda-reduction: $\hat{y}[say'(m, \hat{P}Vx[P(x)=love'(b, y)])]$

Analysis (32) applies the treatment of interrogatives as presented in Hausser (1980b).

The above syntactico-semantic analyses of English sentences show two things: firstly, that examples considered important within transformational grammar (because there they involve intricate constituent-structural considerations), find straightforward solutions in our orthogonal system, and secondly, that in these analyses we cannot avoid to look at the level of analyzed translations, if we want to see the syntactico-semantic generalizations captured. Without exception, the above analyses[9] bear out our desideratum that all the syntactic and semantic properties of a sentence should be a systematic result of the formal properties of the smallest surface constituents (i.e. the molecules), whereby the surface constituents are composed solely by means of functional application.

9) Sub-categorial diacritics introduced by definition 2, (i), cf. (14), section 4), like in $\overset{n}{A}$, are omitted in the orthogonal analyses of this paper. For example, in order to prevent recursive application of $\cdot t\uparrow t$, we have to give this molecule a sub-category like $\overset{1}{t}\uparrow\overset{0}{t}$. Consequently, intransitive verbs would have to be of sub-category $T\uparrow\overset{1}{t}$, etc.

References

Ajdukiewicz, K. 1935. Die syntaktische Konnexität. Studia Philosophica, Lwow 1. 1-27.

Bar-Hillel, Y. 1964. Language and information. Reading, Mass.: Addison-Wesley.

Cooper, R. 1975. Montague's semantic theory and transformational syntax. Unpublished Ph. D. dissertation. University of Massachussetts at Amherst.

Dowty, D.R. 1979. Word meaning and Montague grammar. Dordrecht: Reidel Publishing Company.

Dowty et al. 1981. Introduction to Montague semantics. Dordrecht: Reidel Publishing Company.

Hausser, R.R. 1976. Presuppositions in Montague grammar. Theoretical Linguistics 3. 3.

Hausser, R.R. 1978a. Surface compositionality and the semantics of mood. Amsterdam papers in formal grammar, ed. by J. Groenendijk and M. Stokhof. Vol. 2, University of Amsterdam, 1978; reprinted in Speech act theory and pragmatics, ed. by J.R. Searle, F. Kiefer and M. Bierwisch. Dordrecht: Reidel Publishing Company. 1980.

Hausser, R.R. 1978b. Chapter I, Surface compositional semantics: Methodological deliberations. Pittsburgh 1978, revised: Stanford 1979 and München 1980. Unpublished manuscript, 123 pages.

Hausser, R.R. 1979a. A new approach to context: Do pronouns denote or refer?, to appear in the proceedings of the anaphora conference, held at the Accademia della Crusca, Florence, Italy, December 1978.

Hausser, R.R. 1979b. A constructive approach to intensional contexts. Language Research 18. 2.

Hausser, R.R. 1980a. The place of pragmatics in model-theory. Formal methods in the study of language, ed. by Groenendijk et al Mathematical Center Tracts 135, The University of Amsterdam. 1980.

Hausser, R.R. 1980b. The syntax and semantics of English mood, to appear in Questions and answers, KVAL-series Linguistic Calculus, Dordrecht: Reidel Publishing Company.

Hausser, R.R. 1980c. Chapter III: Categorial surface syntax, München, Unpublished manuscript, 68 pages.

Hausser, R.R. 1981. Chapter IV: The structure of the lexicon, München, Unpublished manuscript, 67 pages.

Hausser, R.R. 1982. The speaker simulation device and the foundations of semantics and pragmatics, to appear in Language Research, Journal of the Language Research Institute, Seoul National University, Korea, 1982.

Lee, Kiyong 1982. On case shifting. Linguistic Journal of Korea 7. 2. 497-517.

Lesniewski, S. 1929. Grundzüge eines neuen Systems der Grundlagen der Mathematik, Fundamenta Mathematicae 14, Warsaw.

Link, G. 1979. Montague Grammatik. München: Wilhelm Fink Verlag.

Loebner, S. 1976. Einführung in die Montague Grammatik. Kronberg.

McCawley, J.D. 1968. Lexical insertion in a transformational grammar without deep structure. CLS 4. 71-80.

Montague, R. 1974. Formal philosophy. The papers of Richard Montague, edited and with an introduction by Richmond Thomason, New Haven: Yale University Press.

Quine, W.V.O. 1960. Word and object. Cambridge, Mass.: MIT Press.

Russell, B. 1905. On denoting. Mind 14.