JÜN-TIN WANG

# ON LOGICAL FORMULATION OF THE COMPUTATION PROCESS IN SEMANTICAL SYSTEMS

In the recent years there is an intensive search for a rigorous and explicit semantics in the theory of natural languages (and in the field of computer languages). In this paper we want to present a general framework in which all these ideas and results can be stated and related in a unified and systematic way, and at the same time, by the using of the language of predicate logic, eventually of higher order, to reformulate and improve the different semantical systems: the system discussed by Scott and Strachey in their development of the theory of mathematical semantics for computer languages, the set-theoretic semantics for context-free fragments of natural languages proposed by Suppes and, finally, Lewis's intensional semantics for a categorial grammar. In § 1, mainly by the example of context-free grammar, we show that the language of predicate logic of first-order can be used to describe the syntax of an object language. We obtain then in § 2 a logical characterization of structural descriptions associated with each reformulated syntactic rule. This leads to the development of the recursive mechanism in the form of definition of semantic functions by cases. This general principle will be then applied in § 3-5 to treat the concrete semantical systems mentioned above.

## 1. GRAMMAR (SYNTAX) AND FIRST-ORDER THEORY

Let us begin with the study of the representation of a grammar as a first-order theory, namely as a theory formalized in the language of predicate logic of first-order. We consider especially the context-free grammar, since it builds up on the one hand the basis for the development of the transformation grammar, on the other hand all the

This work is supported by Deutsche Forschungsgemeinschaft.

semantical systems to be discussed below are referred essentially to context-free languages.

In general, a context-free grammar consists of a set of rewriting rules each of which is of the form

$$A \to A_{i_1} a_{i_1} A_{i_2} a_{i_2} \ldots A_{i_n} a_{i_n} \tag{1.1}$$

where $A$, $A_{i_1}$, ..., $A_{i_n}$ stand for nonterminal symbols (category symbols) and $a_{i_1}$, ..., $a_{i_n}$ for terminal symbols and some of $A_i$ and $a_i$ may be null. We can describe this form of context-free rule by the following logical formula (here and in the sequel we omit the universal quantifications):

$$A_{i_1}(x_1) \wedge A_{i_2}(x_2) \wedge \ldots \wedge A_{i_n}(x_n) \supset A(x_1^\frown a_{i_1}^\frown x_2^\frown a_{i_2}^\frown \ldots {}^\frown x_n^\frown a_{i_n}), \tag{1.2}$$

where we take $A$, $A_{i_1}$, ..., $A_{i_n}$ as unary predicate symbols and $a_{i_1}$, ..., $a_{i_n}$ as individual constants; the symbols "$\wedge$" and "$\supset$" are the conjunction and implication sign and the function symbol "$^\frown$" stands for concatenation. The set of logical formulae thus obtained are called the nonlogical axioms of the first-order theory corresponding to the given context-free grammar.

As an example, we consider the following context-free grammar given by P. SUPPES (1971):

$$\begin{aligned} S &\to NP\ VP \\ VP &\to TV\ NP \\ NP &\to PN \\ PN &\to John \\ TV &\to hit \\ PN &\to Mary \end{aligned}$$

The nonterminal symbols are $S$, $NP$, $VP$, $TV$ and $PN$; the terminal symbols are *John*, *hit* and *Mary*.

Corresponding to this set of context-free rules we can state the following nonlogical axioms:

$$\begin{aligned} NP(x) \wedge VP(y) &\supset S(x^\frown y) \\ TV(x) \wedge NP(y) &\supset VP(x^\frown y) \\ PN(x) &\supset NP(x) \\ PN(John) & \\ TV(hit) & \\ PN(Mary). & \end{aligned}$$

It is obvious that a context-free grammar and its corresponding first-order theory enumerate the same set of terminal strings. More precisely, for each $K$-derivation of a terminal string $w$ in a given context-free grammar in the sense of Chomsky, where $K$ stands for any non-terminal symbol in this grammar, the formal sentence

$$K(w)$$

is in the corresponding first-order theory logically derivable (J. T. WANG, 1973).

With this result we can state even the principle to formulate each grammatical transformation, which according to Chomsky is a mapping of phrase-markers into phrase-markers, as a nonlogical axiom in the following way: we describe both the applied phrase-markers and the resulting phrase-markers, or parts of them, with formulae in the language of predicate logic of first-order which occur then respectively as premises and conclusion of an implication formula. This very implication formula can be then taken as the representation of the grammatical transformation in question. As an example, for the passive transformation, which has been specified by Chomsky as follows:

structural description: $NP$, $Aux$, $V_t$, $NP$
structural change: $x_1\text{-}x_2\text{-}x_3\text{-}x_4 \rightarrow x_4\text{-}x_2\text{-}be\text{-}x_3\text{-}en\text{-}by\text{-}x_1$

we might state then the following nonlogical axiom:

$$S(x_1x_2x_3x_4) \wedge NP(x_1) \wedge Aux(x_2) \wedge V_t(x_3) \wedge NP(x_4)$$
$$\supset S(x_4 \frown x_2 \frown be \frown x_3 \frown en \frown by \frown x_1) \wedge VP(x_2 \frown be \frown x_3 \frown en \frown by \frown x_1).$$

It seems evident, that the same principle stated above can be applied to the logical formulation of other grammatical transformations as well.

## 2. SYNTACTIC RULE AND STRUCTURAL DESCRIPTION

The results obtained above indicate that, in general, syntactic rules can be described in the language of predicate logic of first-order in the form

$$\Gamma \supset \Delta$$

where $\Gamma$ stands for a conjunction of formulae as premises and $\Delta$ for a conjunction of formulae as conclusion of the rule (nonlogical axiom) corresponding to the given syntactic rule. This suggests, however, that we may take the formula

$$\Gamma \wedge \Delta$$

as a characterization of the structural description of the strings generated by the reformulated syntactic rule in question. For example, we can take the formula

$$A_{i_1}(x_1) \wedge A_{i_2}(x_2) \wedge \ldots \wedge A_{i_n}(x_n) \wedge A(x_1^\frown a_{i_1}^\frown x_2^\frown a_{i_2}^\frown \ldots {}^\frown x_n^\frown a_{i_n})$$

as a characterization of the structural description of the strings of the form $x_1^\frown a_{i_1}^\frown x_2^\frown \ldots {}^\frown x_n^\frown a_{i_n}$, which could be enumerated by the reformulated context-free rule, namely the logical formula (1.2) in § 1.

The set of formulae of structural descriptions thus obtained, one for each reformulated syntactic rule, specify therefore exhaustly and disjointly the syntactic conditions, with respect to which the semantic evaluation function, which assigns a denotation or meaning to each generated terminal string, can be defined recursively. This is the kind of the definition of the (semantic) functions by cases in the usual mathematical sense. With this general characterization of our approach we are going now to treat the concrete semantical systems.

## 3. SEMANTICAL SYSTEM FOR THE LANGUAGE OF NUMERALS

Our general idea of using the language of predicate logic, eventually of higher order, to give a recursive definition of the semantical functions can be well illustrated by reformulating the simple semantical system for the language of numerals, which has been given by D. SCOTT and C. STRACHEY (1971) as an example in their development for a theory of mathematical semantics for computer languages. In the following, we give just our reformulation and do not repeat their specification of syntax and semantics for this simple language.

The numerals are expressions in a certain familiar language; while the numbers are mathematical objects (abstract objects) which provide the intended interpretations of the expressions. Based on the explicit syntax given by Scott and Strachey in the form of context-free grammar, we can describe this syntax for binary numerals by the following logical formulae:

$$\begin{aligned}&\textit{Numeral}\ (\mathbf{0})\\&\textit{Numeral}\ (\mathbf{1})\\&\textit{Numeral}\ (x) \supset \textit{Numeral}\ (x\frown\mathbf{0})\\&\textit{Numeral}\ (x) \supset \textit{Numeral}\ (x\frown\mathbf{1}),\end{aligned}$$

where the symbol *Numeral* is a predicate symbol and the symbols " **0** " and " **1** " are individual constants. Thus, a numeral is either one of the digits " **0** " or " **1** " or is the result of suffixing one of these digits to a previously obtained numeral. Let the set of all numerals be called *Nml*. Semantically speaking each of the numerals is meant to denote a unique number. Let $N$ be the set of numbers. The obvious principle of interpretation provides a function $f$, the evaluation mapping, which has the functional character:

$$f\colon \textit{Nml} \to N.$$

Thus, for each $x \in \textit{Nml}$, the function value $f(x)$ is the number denoted by the numeral $x$.

Based on the idea of the semantical equations given by Scott and Strachey and using our general principles described in § 2, we can use the language of predicate logic to state the following semantic rules which define precisely this evaluation function:

$$\begin{aligned}&(3.1)\quad \textit{Numeral}\ (\mathbf{0}) \supset f(\mathbf{0}) = 0,\\&(3.2)\quad \textit{Numeral}\ (\mathbf{1}) \supset f(\mathbf{1}) = 1,\\&(3.3)\quad \textit{Numeral}\ (x) \wedge \textit{Numeral}\ (x\frown\mathbf{0}) \supset f(x\frown\mathbf{0}) = 2.f(x),\\&(3.4)\quad \textit{Numeral}\ (x) \wedge \textit{Numeral}\ (x\frown\mathbf{1}) \supset f(x\frown\mathbf{1}) = 2.f(x) + 1.\end{aligned}$$

This set of logical formulae forms a recursive definition of the semantic evaluation function $f$ with the help of the auxiliary number-theoretic functions of addition and multiplication. It is a definition of function by cases in the usual mathematical sense. Furthermore, this formulation corresponds to the intuitive idea in the linguistic theory, that on the basis of the grammatical properties and the meaning of the basic expressions of the languages the semantic rules enable one to determine the meaning of any well-formed expression in terms of its syntactic structure. In the terminology of J. J. KATZ, J. A. FODOR (1963), (3.1) and (3.2) can be considered as the dictionary component, and (3.3) and (3.4) as the projection rules. However, it is essential to note, that is this kind of logical formulation of syntactic and semantic rules there is no need of any existence of phrase-marker for the semantical system

to compute the denotation or meaning of the expressions. The structural description of a string generated by a syntactic rule is used just as the premises of the semantic rule corresponding to this very syntactic rule.

In this way, we can treat the other complicated semantical systems for computer languages discussed by Scott and Strachey. However, with this simple illustration of our general approach to semantics, we like to continue our study of the semantics for natural languages.

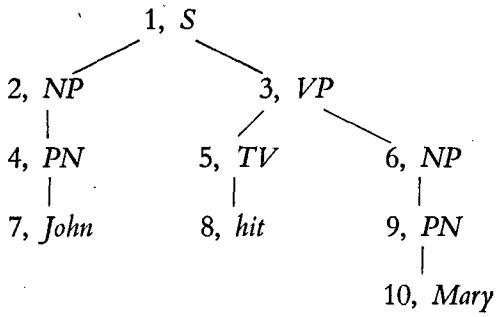
## 4. EXTENSIONAL SEMANTICS OF CONTEXT-FREE FRAGMENTS OF NATURAL LANGUAGES

Within our general framework described in § 1-2 and illustrated in § 3 we want now to treat the semantical system proposed by P. SUPPES (1973). We indicate informally at first his approach and give then our reformulation and improvement of his system.

In his study of the semantics of context-free fragments of natural languages, Suppes intends to give a set-theoretical account of the meaning of a sentence. To do this, a set-theoretical object will be assigned to each basic expression (word). In the case of a noun like *men* the class of men will be assigned; in the case of a proper noun like *John* the class consisting of a certain single member will be assigned; in the case of a transitive verb like *hit* the class consisting of pairs of individuals $x$ and $y$ such that $x$ hit $y$ will be assigned. In this kind of approach, we assume thus, that over the individual words an assignment function $v$ be defined, which assigns a denotation to each word. (We leave aside the discussion about syncategorematic expressions). To give an account of meanings (or denotations) of the various parts of a sentence and their relationships, Suppes introduces set-theoretical functions, just like we have used number-theoretic functions in the semantical system for the language of numerals. With each syntactic rule of the grammar there is associated a semantic set-theoretical function. For example

| syntactic rule | semantic function |
|---|---|
| $S \rightarrow NP \quad VP$ | *truth function* |
| $VP \rightarrow TV \quad NP$ | *image under the converse relation* |
| $NP \rightarrow PN$ | *identity* |
| $NP \rightarrow Adj \quad N$ | *intersection* |
| $PN \rightarrow John$ | *identity* |
| $PN \rightarrow Mary$ | *identity* |
| $TV \rightarrow hit$ | *identity* |

Using the assignment function $v$ which assigns denotations to basic expressions only and using these semantic set-theoretical functions associated with each syntactic rule, the denotation of each labeled node of phrase-marker of the sentence will be then calculated. In Suppes's explicit formulation, the nodes of the phrase-marker will be numbered, so that the denotation function $f$ is actually defined for pairs $(n, s)$, where $n$ is the number assigned to a node of the phrase-marker and $s$ is a terminal or nonterminal symbol. For example, a numbered phrase-marker looks like this




Let $I$ be the identity function, $\breve{A}$ the converse of the binary relation $A$, i.e.

$$\breve{A} = \{ \langle x, y \rangle \mid \langle y, x \rangle \in A \},$$

and $g''B$ the range of the function $g$ restricted to the domain $B$, and let $T$ be truth and $F$ falsity. Then the denotation of each labeled node of the phrase-marker is calculated by working from the bottom up:

$$\begin{aligned}
f(10, \textit{Mary}) &= v(\textit{Mary}),\\
f(9, \textit{PN}) &= I(v(\textit{Mary})),\\
f(8, \textit{hit}) &= v(\textit{hit}),\\
f(7, \textit{John}) &= v(\textit{John}),\\
f(6, \textit{NP}) &= I(v(\textit{Mary})),\\
f(5, \textit{TV}) &= I(v(\textit{hit})),\\
f(4, \textit{PN}) &= I(v(\textit{John})),\\
f(3, \textit{VP}) &= \breve{I}(v(\textit{hit}))''I(v(\textit{Mary})),\\
f(2, \textit{NP}) &= \breve{I}\,I(v(\textit{John})),\\
f(1, \textit{S}) &= g(f(2, \textit{NP}), f(3, \textit{VP})) = \begin{cases} T & \text{if } f(2, \textit{NP}) \subseteq f(3, \textit{VP}) \\ F & \text{otherwise.} \end{cases}
\end{aligned}$$

The calculation of the denotations of expressions will be thus indirectly achieved by the way of the calculation of the denotations of the nodes of the numbered phrase-marker. This unnecessary complication with the whole matter of numbered phrase-marker, which can be traced back to the specification of context-free grammar in form of rewriting rules, would not arise at all, if the context-free grammar had been formulated in the language of predicate logic since the very beginning of the development of theory of grammar by Chomsky. In fact, the process to compute the denotation of an expression can be specified by the following set of logical formulae directly and precisely:

(4.1) $PN(Mary) \supset f(Mary) = v(Mary)$,
(4.2) $PN(John) \supset f(John) = v(John)$,
(4.3) $TV(hit) \supset f(hit) = v(hit)$,
(4.4) $PN(x) \wedge NP(x) \supset . NP(x) \supset f(x) = z \leftrightarrow PN(x) \supset f(x) = z$,
(4.5) $Adj(x) \wedge N(y) \wedge NP(x^\frown y) \supset . f(x^\frown y) = g_1(f(x), f(y))$,
(4.6) $TV(x) \wedge NP(y) \wedge VP(x^\frown y) \supset . f(x^\frown y) = g_2(f(x), f(y))$,
(4.7) $NP(x) \wedge VP(y) \wedge S(x^\frown y) \supset . f(x^\frown y) = g_3(f(x), f(y))$.

The functions $g_1$, $g_2$ and $g_3$ being used to specify the denotation function $f$ are defined as follows:

$$g_1(A, B) = \{ x \mid x \in A \wedge x \in B \}$$
$$g_2(A, B) = \{ x \mid \forall y \, (\langle x, y \rangle \in A \wedge y \in B) \}$$
$$g_3(A, B) = \begin{cases} T & \text{if } A \subseteq B \\ F & \text{otherwise,} \end{cases}$$

or, formulated in the language of predicate logic:

$g_1(A, B) = C \leftrightarrow \wedge x(x \in A \wedge x \in B \leftrightarrow x \in C)$,
$g_2(A, B) = C \leftrightarrow \wedge x(\forall y(\langle x, y \rangle \in A \wedge y \in B \leftrightarrow x \in C)$,
$\wedge x(x \in A \supset x \in B) \supset . g_3(A, B) = T .\wedge. \neg \wedge x(x \in A \supset x \in B)$
$\quad \supset . g_4(A, B) = F$.

By the set of the semantic rules which use these set-theoretic operations, the denotation of any well-formed expressions in this defined language can be then directly calculated. Thus, the denotation of the composed expression *hit Mary* is the set of individuals who hit Mary; the sentence *John hit Mary* is true if and only if the set consisting of the single individual John is properly contained in the set of individuals

who hit Mary. This set of logical formulae forms with respect to the given assignment function $v$ a recursive definition of the denotation function $f$, which assigns to each well-formed expression of the language a set-theoretical object. This function has namely the functional character:

$$f: \textit{ Expressions} \rightarrow \textit{ Set-theoretical Objects.}$$

It is just what Suppes intends to have.

Again, (4.1)-(4.3) can be considered as the dictionary component, and (4.4)-(4.7) as projection rules. The whole set of formulae constitutes a definition of function by cases. The premise of each rule states the syntactic condition, with respect to which the function value is then specified. At the same time, we do not need even the notion of phrase-marker which has played the central role in the concept formation in the theory of generative grammars.

Note there is no general principle to guide us in choosing for each syntactic rule the corresponding semantic set-theoretical function. It may be even possible that no set-theoretical operation can be found for a given syntactic rule at all. It seems likely that in our usual understanding of natural languages we do not use any set-theoretical operation. Anyway, from the standpoint of theory of meaning, the semantics treated by Suppes is only the extensional part of the semantics of a language (A. CHURCH, 1951). With these remarks in mind, let us now consider the intensional semantics discussed by Lewis in his theory of categorial grammar.

## 5. INTENSIONAL SEMANTICS FOR CATEGORIAL GRAMMAR

D. LEWIS (1970) treats a categorial grammar in the sense of Ajdukiewicz as a context-free grammar of the following sort.

First, we have a finite number of basic categories like the categories of name ($N$), common noun ($C$) or sentence ($S$). Second, we have infinitely many derived categories like the categories of intransitive verb ($S/N$), adjective ($C/C$) or article ($(S/(S/N))/C$). In general, whenever $c$, $c_1$, ..., $c_n$, for $n \geqslant 1$, are any categories, either basic or derived, we have a derived category $c/c_1 \ldots c_n$. Third, we have context-free rule of the form

$$c \rightarrow c/c_1 \ldots c_n + c_1 + \ldots + c_n \tag{5.1}$$

corresponding to each derived category, where " + " stands for the concatenation.

One of the main ideas behind the theory of categorial grammar is that the phrases of a derived category are phrases which combine phrases of basic or derived categories to form other phrases of certain category, which may be basic like the category of sentence or derived. Such phrases are called functors by H. B. CURRY (1961). Every functor combines one or more phrases, called its arguments, to form a new phrase called its value. This syntactic property shall be reflected in stating the semantic projection rule. We note here that the context-free rule of the form (5.1) can be described, as before, by the following logical formula as a direct translation:

$$(5.2.) \qquad c/c_1 \ldots c_n\ (\gamma) \wedge c_1\ (x_1) \wedge \ldots \wedge c_n\ (x_n) \supset c\ (\gamma^\frown x_1{}^\frown \ldots\ x_n),$$

where we take $c/c_1 \ldots c_n$, $c_1$, ..., $c_n$ and $c$ again as unary predicate symbols. The word order as specified in the context-free rule (5.1) is sometimes too odd; the phrase of a derived category takes always the left-most position in its combined phrase. To take into account the natural word order, we should actually describe it by the following logical formula:

$$(5.3) \qquad c/c_1 \ldots c_n\ (\gamma) \wedge c_1\ (x_1) \wedge \ldots \wedge c_n\ (x_{1n}) \supset c\ (x_1{}^\frown \ldots {}^\frown x_i{}^\frown \gamma^\frown x_{i+1}{}^\frown \ldots {}^\frown x_n).$$

With these remarks about the syntactic rules in a categorial grammar, we are prepared to treat Lewis's intensional semantics within our general framework.

Let us denote the sets of phrases of name, common noun, adjective, intransitive verb, article etc. by *Names*, *Common-Nouns*, *Adjectives*, *Intransitive Verbs* and *Articles* respectively. Let *Things* denote the set of things and *Sets* the set of sets of things. And let $T$ denote the set of truth values, and let $I$ denote the set of indices, each of which is, roughly speaking, a package of the various factors like possible world, time, place and speaker on which the extension of an expression may depend. As used before, given any two domains $D_1$ and $D_2$, we write $[D_1 \to D_2]$ for the set of all functions from $D_1$ into $D_2$. We write

$$h: \ D_1 \to D_2$$

to indicate that $h$ is just such a function.

According to Lewis's theory, an appropriate intension for a name is any function from indices to things; an appropriate intension for a common noun is any function from indices to sets of things; an appropriate intension for an adjective is any function from common-noun intensions to common-noun intensions, etc. In adapting this point of view, we shall consider in the sequel assignment functions, each of which assigns an appropriate meaning to a word of certain category, with the following functional characters:

$v_1$: *Names* $\rightarrow$ [$I \rightarrow$ *Things*]
$v_2$: *Common-Nouns* $\rightarrow$ [$I \rightarrow$ *Sets*]
$v_3$: *Adjectives* $\rightarrow$ [[$I \rightarrow$ *Sets*] $\rightarrow$ [$I \rightarrow$ *Sets*]]
$v_4$: *Intransitive Verbs* $\rightarrow$ [[$I \rightarrow$ *Things*] $\rightarrow$ [$I \rightarrow T$]]
$v_5$: *Articles* $\rightarrow$ [[$I \rightarrow$ *Sets*] $\rightarrow$ [[[$I \rightarrow$ *Things*] $\rightarrow$ [$I \rightarrow T$]] $\rightarrow$ [$I \rightarrow T$]]]
...

Thus, for *man* $\in$ *Common-Nouns*, the function value

$$v_2\ (man)$$

is the assigned meaning of the word *man*, where this meaning itself is a kind of function from indices to sets of things (individuals). In what follows, we assume that such assignment functions specifying an appropriate meaning for each word of certain category are already given or known to us, for example by learning or ostensive definition. In other words, we assume we know the meaning of each individual word. The problem is just how we can know the meaning of any well-formed expression in this language. We ask namely for a semantic evaluation function, which can compute the meaning of any well-formed expression on the basis of its syntactic structure and the meaning of its constituting words. This evaluation function $f$ shall have namely the following functional character:

$$f\colon\ \textit{Expressions} \rightarrow \textit{Meanings}.$$

To define this function, we begin with the specification of its value for the individual words or basic expressions, based on the given assignment functions. As before, we state thus the following semantic rules, here formulated in the language of predicate logic of higher-order due to the functional characters of the assignment functions:

(5.4) $N\ (John) \supset f\ (John) = v_1\ (John)$,
(5.5) $C\ (man) \supset f\ (man) = v_2\ (man)$,
(5.6) $C/C\ (young) \supset f\ (young) = v_3\ (young)$,
(5.7) $S/N\ (sleeps) \supset f\ (sleeps) = v_4\ (sleeps)$,
(5.8) $(S/(S/N))/C\ (the) \supset f\ (the) = v_5\ (the)$,

where we take again the symbols $N$, $C$, $C/C$, $S/N$, $(S/(S/N))/C$ etc. as unary predicate symbols. For the computation of the meaning of composed well-formed phrases, we shall state, corresponding to each syntactic rule, the semantic projection rule. For the syntactic rule of the form (5.2), the semantic projection rule will, in general, be of the form

$$c/c_1 \ldots c_n\ (\gamma) \wedge c_1\ (x_1) \wedge \ldots \wedge c_n\ (x_n) \wedge c\ (\gamma^\frown x_1^\frown x_2^\frown \ldots^\frown x_n)$$
$$\supset . f(\gamma^\frown x_1^\frown x_2^\frown \ldots^\frown x_n) = g(f(\gamma), f(x_1), \ldots, f(x_n)),$$

based on the assumption, that the meaning of an expression depends on the meanings of its parts; it is a function of the meanings of its parts. This function $g$, which uses the meanings of these parts as arguments to compute the meaning of the composed phrase as its value, must be, first of all, specified. In general, as we have seen in the case of Suppes's system, there is no uniform principle to do it. In his semantic theory for categorial grammar, Lewis holds, however, the view that the result of concatenating a phrase of the category $c/c_1 \ldots c_n$ with intension $i_0$, a phrase of the category $c_1$ with intension $i_1$, ..., and a phrase of the category $c_n$ with intension $i_n$ is a phrase of the category $c$ with intension

$$i_0\ (i_1, \ldots, i_n).$$

In accordance with this conception, we can state then, corresponding to each syntactic rule of the form (5.2), the following semantic projection rule:

(5.9) $c/c_1 \ldots c_n\ (\gamma) \wedge c_1\ (x_1) \wedge c_2\ (x_2) \wedge \ldots \wedge c_n\ (x_n) \wedge c\ (\gamma^\frown x_1^\frown x_2^\frown \ldots^\frown x_n)$
$\supset . f\ (\gamma^\frown x_1^\frown x_2^\frown \ldots^\frown x_n) = f\ (\gamma)\ (f(x_1), \ldots, f(x_n))$,

or, taking into account the proper word order:

(5.10) $c/c_1 \ldots c_n\ (\gamma) \wedge c_1\ (x_1) \wedge \ldots \wedge c_n\ (x_n) \wedge c\ (x_1^\frown \ldots^\frown x_i^\frown \gamma^\frown x_{i+1}^\frown \ldots^\frown x_n)$
$. f\ (x_1^\frown \ldots^\frown x_i^\frown \gamma^\frown x_{i+1}^\frown \ldots^\frown x_n) = f\ (\gamma)\ (f\ (x_1), \ldots, f\ (x_n))$.

Thus, we may have semantic projection rule like

$$S/N\ (\gamma) \wedge N\ (x) \supset .f\ (x^{\frown}\gamma) = f\ (\gamma)\ (f\ (x)).$$

One of its instances is as follows:

$$S/N\ (\textit{sleeps}) \wedge N\ (\textit{John}) \supset .f\ (\textit{John}^{\frown}\textit{sleeps}) = f\ (\textit{sleeps})\ (f\ (\textit{John})).$$

In combination with the logical formulae (5.4)-(5.8), such rules enable one to derive logically the meaning of the sentence like *John sleeps*, which can be represented as $\nu_4\ (\textit{sleeps})\ (\nu_1\ (\textit{John}))$. And this representation would correspond to the usual so-called logical form: $SLEEP(j)$. The set of projection rules thus obtained specifies in connection with those semantic rules like (5.4)-(5.8), which constitute the dictionary component, the evaluation function $f$ recursively. This kind of formulation of semantic rules stands very near to our intuition, that on the basis of the syntactic structure and on the knowledge of the meanings of the constituting words we can understand the meaning of an expression.

The semantic projection rule stated above has a nice homogeneous feature in specifying semantic evaluation function. This holds actually for all grammars using the notion of functor which combines one or more phrases as its arguments to form a new phrase. The meaning of the composed phrase can be then always considered as the result of the operation of the meaning of the functor with the meanings of its arguments. According to H. B. CURRY (1961), what Harris and Chomsky call transformations are also functors. From this point of view, we can then state the semantic projection rule for a given transformation rule in the same way as described above.

## REFERENCES


A. CHURCH, *The need for abstract entities in semantic analysis*, in «Proc. Amer. Academy of arts and sci.» LXXX (1951).

H. B. CURRY, *Some logical aspects of grammatical structure*, in R. JAKOBSON (ed.), *Structure of language and its mathematical aspects*, Providence, 1961.

J. J. KATZ, J. A. FODOR, *The structure of a semantic theory*, in « Language », XXXIX (1963), pp. 170-210.

D. LEWIS, *General semantic*, in « Synthese », XXII (1970).

D. SCOTT, C. STRACHEY, *Toward a mathematical semantics for computer languages*, in *Proc. of the symp. on computers and automata*, 1971.

P. SUPPES, *Semantics of context-free fragments of natural languages*, in K. J. J. HINTIKKA, *et al.* (eds.), *Approaches to natural language*, Dordrecht, 1973.

J. T. WANG, *Zum rekursiven Mechanismus im semantischen System*, in A. P. TEN CATE, P. JORDENS (eds.), *Linguistische Perspektiven*, Tübingen, 1973, pp. 205-219.

J. T. WANG, *On the representation of generative grammars as first-order theories*, in R. J. BOGDAN, I. NIINILUOTO (eds.), *Logic, language and probability*, Dordrecht, 1973.