1. Second-order logic without variables

KOSTA DOŠEN

In papers published in 1960, 1971 and 1981 Quine has given a formulation of first-order logic without variables. In this formulation propositional connectives and first-order quantifiers are replaced by predicate functors, i.e. functors which transform predicates into predicates. Some additional predicate functors of a combinatorial nature are added to make up for the loss of variables. These combinatory predicate functors exhibit what combinatory role bound variables have in first-order logic. (In addition to Quine's papers cited above, a reader interested in Quine's variable-free formulation of logic should consult Kuhn 1983, which deals with the axiomatization of Quine's formulation. In Kuhn 1983 it is also possible to find references to works with formulations of logic similar to Quine's.)

The aim of the present paper is to extend Quine's variable-free formulation to second-order logic. As before, propositional connectives and first-order quantifiers will be replaced by predicate functors, whereas second-order quantifiers will be replaced by functors which transform predicate functors into predicate functors. In addition to Quine's combinatory predicate functors we shall have combinatory functors which transform predicate functors into predicate functors. Most of these new combinatory functors will play a role analogous to the role of the old combinatory predicate functors. However, some of these new functors will play a completely new combinatory role, which essentially corresponds to *functional composition*. From this we should conclude that the combinatory role of bound variables in second-order logic is more complex than the combinatory role of bound variables in first-order logic. Both Quine's and our variable-free formulation of logic differ from the variable-free formulations of Schönfinkel and Curry in that they do not assume full-fledged type-free combinators (cf. Quine 1971).

Quine has no sympathy with second-order logic, and this might be a sufficient reason for somebody who is of his mind not to be interested in the subject of our paper. However, those who think second-order logic is a legitimate subject would probably like to know what is the combinatory role of bound variables in this logic. In this paper we shall not only try to exhibit this combinatory role, but we shall also make some remarks which deal with questions of categorial grammar.

Quine's variable-free formulation of first-order logic is meant to be a systematization of the idea, advocated also by Geach (1970; Ajdukiewicz 1967, p. 635, fn. **), that the quantifier prefix $\exists x$ and the bound variable x form a single whole which is a predicate functor reducing the number of argument places of the predicate to which it is applied, and analogously with $\forall x$. In the same way, our variable-free formulation of second-order logic is meant to be a systematization of such an understanding of quantifiers, including both first-order and second-order quantifiers. As the first-order quantifier prefix $\exists x$ and the bound variable x join into a single whole to produce the quantifier expression $\exists x \ldots x \ldots$, so the second-order quantifier prefix $\exists P$ and the bound variable P join into a single whole to produce the quantifier expression $\exists P \ldots P \ldots$, which is again a functor reducing the number of argument places of the functor to which it is applied. Of course, both Quine's and our formulation of logic are motivated by theoretical considerations and not by practical convenience.

We shall describe in terms of categorial grammar the languages we need for Quine's and our formulation of logic. For this description we shall first use a rigid categorial apparatus: a categorial apparatus is here called *rigid* if every primitive expression of our languages is of a single category; otherwise, it is called *flexible*. In the last section of this paper we shall find that it might be more natural if we introduced some flexibility into our categorial apparatus. This applies in particular to the language of our variable-free formulation of second-order logic. For example, we have said that for second-order quantifiers we have functors which transform predicate functors into predicate functors. Such is clearly the category of the second-order quantifier expression $\exists P \ldots P \ldots$ in the formula $\exists P(\exists x(Px))$, where it is applied to $\exists x \ldots x \ldots$; on the other hand, $\exists P \ldots P \ldots$ in the formula $\exists P(Px)$ seems to be simply of the category of predicates. We shall argue that a commutative variant of Lambek's calculus of syntactic categories (see Lambek 1958) enables us to account for this shifting of categories: with it we can construct a flexible categorial apparatus for our languages. This flexibility

which brings in the commutativity of concatentation for our languages, need not be desirable for logic, but it might be important when we want to analyze natural languages with our logical languages. We shall also show that if we do not want to bring in commutativity, the original Lambek calculus can still give us some flexibility.

In the first section of this paper we present a variant of Quine's variable-free formulation of first-order logic without identity. In the second section we present our variable-free formulation of second-order logic, which is an extension of the variant of Quine's formulation presented in the first section. Finally, in the third section we consider questions of categorial grammar.

1. Variable-free formulation of first-order logic

In describing the languages we need for our variable-free formulation of logic we shall use the following simple calculus of syntactic categories, which we shall call M. The language of M has finitely, or denumerably, many primitive category terms, the operations on category terms / and $\circ$, and the relation $\rightarrow$ between category terms. In applying M in the sequel we assume that the only primitive category terms are s, which stands for the category of formulae, and n, which stands for the category of terms. If a and b are category terms, then the category term a/b is understood intuitively as denoting the category of a functor which transforms an expression of the category b concatenated immediately to the right of the functor into an expression of the category a. The category term $a \circ b$ stands for the category of an expression obtained by concatenating an expression of the category a and an expression of the category b in that order. The intuitive reading of the formula $a \rightarrow b$ is "every expression of the category a is also of the category b", or more simply "a reduces to b". Accordingly, we shall sometimes call $a \rightarrow b$ a *reduction principle*. We use $a \rightleftarrows b$ as an abbreviation for $a \rightarrow b$ *and* $b \rightarrow a$. Using a,b,c,a',b', ... as schematic letters for category terms, we assume for M the following axiom-schemata and rules:

(i) $a \rightarrow a$

(ii) $(a \circ b) \circ c \rightleftarrows a \circ (b \circ c)$

(iii) $(a/b) \circ b \rightarrow a$

(iv) $$\frac{a \rightarrow a' \qquad b \rightarrow b'}{a \circ b \rightarrow a' \circ b'}$$

$$\text{(v)}\quad \frac{a \to b \qquad b \to c}{a \to c}$$

It is easy to see that the following gives a rough desription of a decision procedure for M. To verify whether $a \to a'$ is provable in M start from a and apply the following two rules: rearrange parentheses using the associativity of $\circ$, and if a_i is $(b/c)\circ c$ replace a_i in the term $a_1\circ\ldots\circ a_k$ by b; it is clear that this way we can reach only a finite number of terms. If a' is among these terms, $a \to a'$ is provable; otherwise, it is not. It is also easy to see that if $d(a)$ is the number of occurrences of $/$ and $\circ$ in a, then for any formula $a \to b$ provable in M we have $d(a) \geqslant d(b)$. A reduction principle $a \to b$ is said to be *category raising* if $d(a) < d(b)$. So, no reduction principle of M is category raising.

In the sequel, M, or a calculus like M, will be used to describe a language in the following way. At the beginning we assign to every primitive expression of our language a single category term which is not of the form $a\circ b$. If the expression e_1 has the category term a and the expression e_2 has the category term b, then the expression e_1e_2 has the category term $a\circ b$. We can disregard parentheses in terms obtained by repeated use of $\circ$, because $\circ$ is associative. An expression e is of the category b if, and only if, for the category term a assigned to e we have that $a \to b$ is provable in M. For example, when b is s, the expression e is a formula. It is clear that a language based on M will be in Polish notation; i.e. functors in this language will be written to the left of their arguments. It is not difficult to see that if a is not of the form $a'\circ a''$, then $a \to b$ is provable in M if, and only if, a is equal to b. Since to every primitive expression of our language we assign at the beginning a single category term which is not of the form $a'\circ a''$, it follows that every primitive expression of a language based on M is of a single category. In other words, the categorial apparatus provided by M is rigid.

We shall now present a variant of Quine's variable-free formulation of first-order logic without identity (essentially, this variant is based on Quine 1960). For that purpose we introduce the language L^1 of first-order logic, which has the following primitive expressions:

(0) *individual variables*, of the category n; as schemata for individual variables we use $x,y,z,x_1,y_1,z_1,x_2,\ldots$;
primitive predicates, of the categories p_k, where p_k is an abbreviation for $s/(n\circ\ldots\circ n)$ with $k \geqslant 0$ occurrences of n (p_0 stands for s); as schemata for primitive predicates we use $P,P_1,P_2,\ldots$;

(1) the *conjunction connective* $\wedge$, of the category $s/(s \circ s)$;
the *negation connective* $\neg$, of the category s/s;
the *existential quantifier prefixes* $\exists x$, of the category s/s;

(2) the *logical predicate functors*:
$\wedge_{k,m}$, of the categories $p_{k+m}/(p_k \circ p_m)$;
$\neg_k$, of the categories p_k/p_k;
$\exists_k$, of the categories p_k/p_{k+1};
the *combinatory predicate functors*:
Inv_k and inv_k, of the categories p_{k+2}/p_{k+2};
Ref_k, of the categories p_{k+1}/p_{k+2}.

As schemata for predicates, primitive or obtained by applying predicate functors, we use $F, F_1, F_2, \ldots$

The full language L^1 is obtained by using M. If we omit the expressions of (2) from L^1, we obtain an ordinary first-order language, whereas if we omit the expressions of (1) and the individual variables, we obtain the language of Quine's variable-free formulation. The first of the resulting languages will be called "L^1 without (2)". Analogously, "L^1 without (1)" will denote the language obtained by omitting from L^1 the expressions of (1). Note that there is a certain overlap between (1) and (2), viz. $\wedge_{0,0}$ and $\neg_0$ amount to $\wedge$ and $\neg$ respectively.

Next we formulate in L^1 a system S^1 which will enable us to achieve a reduction to Quine's variable-free formulation. First, let the equivalence connective $\leftrightarrow$ be defined in the usual way in terms of $\wedge$ and $\neg$; for easier reading we shall write $\leftrightarrow$ as an *infix*, and not as in Polish notation. Then we assume for S^1 the following *Rule of Replacement of Equivalents*:

$$\frac{A \leftrightarrow B}{C \leftrightarrow C[A/B]}$$

where the formula $C[A/B]$ is obtained from the formula C by replacing zero or more occurrences of the formula A by the formula B. Next, we assume for S^1 the axioms given by the following equivalences:

$\exists x A \leftrightarrow A$, for every formula A in which x does not occur free

$(\wedge_{k,m}(F_1 F_2))(x_1 \ldots x_k y_1 \ldots y_m) \leftrightarrow \wedge((F_1(x_1 \ldots x_k))(F_2(y_1 \ldots y_m)))$

$(\neg_k F)(x_1 \ldots x_k) \leftrightarrow \neg(F(x_1 \ldots x_k))$

$(\exists_k F)(x_1 \ldots x_k) \leftrightarrow \exists x(F(x_1 \ldots x_k x))$

$(\mathrm{Inv}_k F)(x_2 \ldots x_{k+2} x_1) \leftrightarrow F(x_1 \ldots x_{k+2})$

$$(\mathrm{inv}_k F)(x_2 x_1 x_3 \ldots x_{k+2}) \leftrightarrow F(x_1 \ldots x_{k+2})$$
$$(\mathrm{Ref}_k F)(x_1 \ldots x_{k+1}) \leftrightarrow F(x_1 \ldots x_{k+1} x_{k+1}).$$

Then we can prove the following lemmata:

1.1. **Lemma**: For every F of a category p_{k+2} there is an F_1 such that in S^1 we can prove

$$F(x_1 \ldots x_i x_{i+1} \ldots x_{k+2}) \leftrightarrow F_1(x_1 \ldots x_{i-1} x_{i+1} x_i x_{i+2} \ldots x_{k+2}).$$

Proof: It is easy to check that the required F_1 is of the form

$$\underbrace{(\mathrm{Inv}_k \ldots (\mathrm{Inv}_k}_{k+2-(i-1)}(\mathrm{inv}_k(\underbrace{\mathrm{Inv}_k \ldots (\mathrm{Inv}_k}_{i-1} F)\ldots)))\ldots).$$

1.2. **Lemma**: For every F there is an F_1 such that in S^1 we can prove

$$F(x_1 \ldots x_k) \leftrightarrow F_1(y_1 \ldots y_m)$$

where $y_1, \ldots, y_m$ are the variables $x_1, \ldots, x_k$ with repetitions among these variables eliminated.

Proof: We have in S^1

$$F(x_1 \ldots x_i y x_{i+2} \ldots x_j y x_{j+2} \ldots x_{k+2})$$
$$\leftrightarrow F_2(x_1 \ldots x_i x_{i+2} \ldots x_j x_{j+2} \ldots x_{k+2} yy), \text{ using Lemma 1.1}$$
$$\leftrightarrow (\mathrm{Ref}_k F_2)(x_1 \ldots x_i x_{i+2} \ldots x_j x_{j+2} \ldots x_{k+2} y).$$

This way we eliminate all repetitions and find F_1.

1.3. **Lemma**: For every formula A of L^1 without (2) there is a formula of L^1 without (1) of the form $F(x_1 \ldots x_k)$, where $x_1, \ldots, x_k$ are all the free individual variables of A without repetitions, such that in S^1 we can prove $A \leftrightarrow F(x_1 \ldots x_k)$.

Proof: By induction on the complexity of A. For the basis we have that if A is atomic, i.e. of the form $P(y_1 \ldots y_n)$, we need eventually apply Lemma 1.2. For the induction step we have the following cases.

Suppose A is of the form $\wedge(A_1 A_2)$, and $A_1 \leftrightarrow F_1(y_1 \ldots y_n)$ and $A_2 \leftrightarrow F_2(z_1 \ldots z_m)$. Then we have in S^1

$$A \leftrightarrow \wedge((F_1(y_1 \ldots y_n))(F_2(z_1 \ldots z_m)))$$
$$\leftrightarrow (\wedge_{n,m}(F_1 F_2))(y_1 \ldots y_n z_1 \ldots z_m)$$

and we use eventually Lemma 1.2 to eliminate repetitions among $y_1, \ldots, y_n, z_1, \ldots, z_m$.

Suppose A is of the form $\neg A_1$ and $A_1 \leftrightarrow F_1(y_1 \ldots y_n)$. Then in S^1 we have $A \leftrightarrow (\neg_n F_1)(y_1 \ldots y_n)$.

Suppose, finally, A is of the form $\exists y_i A_1$ and $A_1 \leftrightarrow F_1(y_1 \ldots y_n)$. If $y_i \notin \{y_1, \ldots, y_n\}$, then in S^1 we have $A \leftrightarrow F_1(y_1 \ldots y_n)$.If $y_i \in \{y_1, \ldots, y_n\}$, then using eventually Lemma 1.1 we have in S^1

$$\begin{aligned} \exists y_i(F_1(y_1 \ldots y_n)) &\leftrightarrow \exists y_i(F_2(y_1 \ldots y_{i-1} y_{i+1} \ldots y_n y_i)) \\ &\leftrightarrow (\exists_{n-1} F_2)(y_1 \ldots y_{i-1} y_{i+1} \ldots y_n). \end{aligned}$$

This concludes the proof of the lemma.

1.4. **Lemma**: For every formula of L^1 without (1) of the form $F(x_1 \ldots x_k)$ there is a formula A of L^1 without (2), whose free individual variables coincide with the individual variables of $F(x_1 \ldots x_k)$, such that in S_1 we can prove $F(x_1 \ldots x_k) \leftrightarrow A$.

Proof: By induction on the complexity of F. For the basis we have that if $F(x_1 \ldots x_k)$ is $P(x_1 \ldots x_k)$, then A is $P(x_1 \ldots x_k)$. For the induction step we use the equivalences concerning the expressions of (2) we have assumed for S^1. This proves the lemma.

In fact, the proofs of Lemmata 1.3 and 1.4 are more informative than the statements of these lemmata. Starting from the proof of Lemma 1.3 we could easily, but rather tediously, devise a procedure yielding for every A a unique $F(x_1 \ldots x_k)$, and analogously with Lemma 1.4.

As corollaries of Lemmata 1.3 and 1.4 we get the following two theorems, which we can take as giving the essence of Quine's variable-free formulation of first-order logic:

1.1. **Theorem**: For every formula A of L^1 without (2), which has no free individual variables, there is a predicate F of the category s, without individual variables, free or bound, such that in S^1 we can prove $A \leftrightarrow F$.

1.2. **Theorem**: For every predicate F of the category s, without individual variables, free or bound, there is a formula A of L^1 without (2), which has no free individual variables, such that in S_1 we can prove $F \leftrightarrow A$.

2. Variable-free formulation of second-order logic

We shall now present our variable-free formulation of second-order logic. The language L^2 of second-order logic we need for this formulation has the following primitive expressions:

(0) as for L^1, save that instead of primitive predicates we now have *predicate variables*, for which we use the schemata $P,R,V,P_1,R_1,V_1,P_2,\ldots$;

(1) as for L_1, save that we also have the *second-order existential quantifier prefixes* $\exists P.$ of the category s/s;

(2) as for L^1, save that we also have:

- the *combinatory predicate functors*:
  Id_k, of the categories p_k/p_k;
- the *logical functors*:
  $\exists_j^2$ (which is an abbreviation for $\exists^2_{k,k_1,\ldots,k_{j+1}}$; we use analogous abbreviations below), of the categories q_j/q_{j+1}, where q_j is an abbreviation for $p_k/(p_{k_1} \circ \ldots \circ p_{k_j})$, i.e. an abbreviation for the categories of predicate functors ($j \geqslant 0$; q_0 is p_k);
- the *combinatory functors*:
  Inv_j^2 and inv_j^2, of the categories q'_{j+2}/q_{j+2}, where q'_{j+2} is $p_k/(p_{k_2} \circ \ldots \circ p_{k_{j+2}} \circ p_{k_1})$ for Inv_j^2, and $p_k/(p_{k_2} \circ p_{k_1} \circ p_{k_3} \circ \ldots \circ p_{k_{j+2}})$ for inv_j^2;
  Ref_j^2, of the categories q_{j+1}/q_{j+2}, where q_{j+2} is $p_k/(p_{k_1} \circ \ldots \circ p_{k_{j+1}} \circ p_{k_{j+1}})$;
  Comp_j^2, of the categories $q_j/(q_1 \circ q'_j)$, where q_j is $p_k/(p_{k_1} \circ \ldots \circ p_{k_j})$, q_1 is p_k/p_m, and q'_j is $p_m/(p_{k_1} \circ \ldots \circ p_{k_j})$;
  $\mathrm{Comp}_{i,j}^2$, of the categories $q_{i+j}/(q_2 \circ q_i \circ q_j)$, where q_{i+j} is $p_{k+m}/(p_{k_1} \circ \ldots \circ p_{k_i} \circ p_{k_{i+1}} \circ \ldots \circ p_{k_{i+j}})$, q_2 is $p_{k+m}/(p_k \circ p_m)$, q_i is $p_k/(p_{k_1} \circ \ldots \circ p_{k_i})$, and q_j is $p_m/(p_{k_{i+1}} \circ \ldots \circ p_{k_{i+j}})$.

As schemata for predicate functors, primitive or obtained by applying some of the new functors of (2) of L^2, we use $Q,Q_1,Q_2,\ldots$

The full language L^2 is obtained by using the calculus M of Section 1. The expressions of (0) and (1) of L^2 constitute an ordinary second-order

language, whereas the expressions of (2) of L^2 constitute the language of our variable-free formulation of second-order logic.

Next we formulate in L^2 a system S^2 which will enable us to achieve a reduction to our variable-free formulation. For S^2 we assume whatever we have assumed for S^1, plus the axioms given by the following equivalences, where $\vec{x}$ is an abbreviation for $(x_1 \dots x_k)$:

$$\exists PA \leftrightarrow A, \text{ for every formula } A \text{ in which } P \text{ does not occur free}$$
$$(\mathrm{Id}_k F)\vec{x} \leftrightarrow F\vec{x}$$
$$((\exists_j^2 Q)(F_1 \dots F_j))\vec{x} \leftrightarrow \exists P((Q(F_1 \dots F_j P))\vec{x})$$
$$((\mathrm{Inv}_j^2 Q)(F_2 \dots F_{j+2} F_1))\vec{x} \leftrightarrow (Q(F_1 \dots F_{j+2}))\ \vec{x}$$
$$((\mathrm{inv}_j^2 Q)(F_2 F_1 F_3 \dots F_{j+2}))\vec{x} \leftrightarrow (Q(F_1 \dots F_{j+2}))\vec{x}$$
$$((\mathrm{Ref}_j^2 Q)(F_1 \dots F_{j+1}))\vec{x} \leftrightarrow (Q(F_1 \dots F_{j+1} F_{j+1}))\vec{x}$$
$$((\mathrm{Comp}_j^2(Q_1 Q_2))(F_1 \dots F_j))\vec{x} \leftrightarrow (Q_1(Q_2(F_1 \dots F_j)))\vec{x}$$
$$((\mathrm{Comp}_{i,j}^2(\bigwedge_{k,m} Q_1 Q_2))(F_1 \dots F_i F_{i+1} \dots F_{i+j}))(x_1 \dots x_{k+m})$$
$$\leftrightarrow (\bigwedge_{k,m}((Q_1(F_1 \dots F_i))(Q_2(F_{i+1} \dots F_{i+j}))))(x_1 \dots x_{k+m}).$$

Now, by imitating the proof of Lemma 1.1 we can prove the following analogue of this lemma:

2.1. Lemma: For every Q of a category q_{j+2} there is a Q_1 such that in S^2 we can prove

$$(Q(F_1 \dots F_i F_{i+1} \dots F_{j+2}))\vec{x} \leftrightarrow (Q_1(F_1 \dots\ F_{i-1} F_{i+1} F_i F_{i+2} \dots F_{j+2}))\vec{x}.$$

Next, by imitating the proof of Lemma 1.2 we can prove the following analogue of this lemma:

2.2. Lemma: For every Q there is a Q_1 such that in S^2 we can prove

$$(Q(P_1 \dots P_j))\vec{x} \leftrightarrow (Q_1(R_1 \dots R_i))\ \vec{x}$$

where $R_1, \dots, R_i$ are the variables $P_1, \dots, P_j$ with repetitions among these variables eliminated.

Now we are ready for the proof of the analogue of Lemma 1.3:

2.3. Lemma: For every formula A of L^2 without (2) there is a formula of L^2 without (1) of the form $(Q(P_1 \dots P_j))(x_1 \dots x_k)$, where $P_1, \dots, P_j, x_1, \dots, x_k$ are all the free variables of A without repetitions, such that in S^2 we can prove $A \leftrightarrow (Q(P_1 \dots P_j))(x_1 \dots x_k)$.

Proof: By induction on the complexity of A. For the basis we have that if A is atomic, i.e. of the form $P(y_1...y_n)$, applying eventually Lemma 1.2, we obtain $P(y_1...y_n) \leftrightarrow F\vec{x}$. Now, if $F\vec{x}$ is of the form $(Q_1...(Q_mP)...)\vec{x}$, where the Q's are instances of Inv_k, inv_k or Ref_k, we have

$$(Q_1(Q_2...(Q_mP)...))\vec{x} \leftrightarrow ((\mathrm{Comp}_1^2(Q_1Q_2))(Q_3...(Q_mP)...))\vec{x}$$

and thus eventually iterating applications of Comp_1^2 we obtain $F\vec{x} \leftrightarrow (QP)\vec{x}$. If $F\vec{x}$ is of the form $P\vec{x}$, then we have $P\vec{x} \leftrightarrow (\mathrm{Id}_kP)\vec{x}$.

For the induction step we have the following cases.

Suppose A is of the form $\wedge(A_1A_2)$, and $A_1 \leftrightarrow (Q_1(R_1...R_r))(y_1...y_n)$ and $A_2 \leftrightarrow (Q_2(V_1...V_s))(z_1...z_m)$. Then we have in S^2

$$\begin{aligned} A &\leftrightarrow (\wedge_{n,m}((Q_1(R_1...R_r))(Q_2(V_1...V_s)))(y_1...y_nz_1...z_m) \\ &\leftrightarrow ((\mathrm{Comp}^2_{r,s}(\wedge_{n,m}Q_1Q_2))(R_1...R_rV_1...V_s))(y_1...y_nz_1...z_m) \end{aligned}$$

and then using eventually Lemma 2.2 to eliminate repetitions among $R_1,...,R_r,V_1,...,V_s$, and Lemma 1.2 to eliminate repetitions among $y_1,...,y_n,z_1,...,z_m$, and applying eventually Comp_i^2 and Comp_j^2 as in the basis of the induction, we obtain $A \leftrightarrow (Q(P_1...P_j))\vec{x}$.

Suppose A is of the form $\neg A_1$ and $A_1 \leftrightarrow (Q_1(P_1...P_j))\vec{x}$. Then in S^2 we have $A \leftrightarrow ((\mathrm{Comp}_j^2(\neg_kQ_1))(P_1...P_j))\vec{x}$.

Suppose A is of the form $\exists y_iA_1$ and $A_1 \leftrightarrow (Q_1(P_1...P_j))(y_1...y_n)$. If $y_i \notin \{y_1,...,y_n\}$, then in S^2 we have $A \leftrightarrow (Q_1(P_1...P_j))(y_1...y_n)$. If $y_i \in \{y_1,...,y_n\}$, then using eventually Lemma 1.1 to push y_i at the end of $(y_1...y_n)$, and applying Comp_j^2 we obtain $A \leftrightarrow ((\mathrm{Comp}_j^2(\exists_{n-1}Q_2))(P_1...P_j))(y_1...y_{i-1}y_{i+1}...y_n)$.

Suppose, finally, A is of the form $\exists P_iA_1$ and $A_1 \leftrightarrow (Q_1(P_1...P_r))\vec{x}$. If $P_i \notin \{P_1,...,P_r\}$, then in S^2 we have $A \leftrightarrow (Q_1(P_1...P_r))\vec{x}$. If $P_i \in \{P_1,..,P_r\}$, then using eventually Lemma 2.1 to push P_i at the end of $(P_1...P_r)$, we obtain $A \leftrightarrow ((\exists^2_{r-1}Q_2)(P_1...P_{i-1}P_{i+1}...P_r))\vec{x}$. This concludes the proof of the lemma.

We can also prove the following analogue of Lemma 1.4:

2.4. Lemma: For every formula of L^2 without (1) of the form $F\vec{x}$ there is a formula A of L^2 without (2), whose free variables coincide with the variables of $F\vec{x}$, such that in S^2 we can prove $F\vec{x} \leftrightarrow A$.

This is proved, in complete analogy with the proof of Lemma 1.4, by induc-

tion on the complexity of F, where complexity is measured by the number of expressions from (2) which occur in F.

As we have already remarked concerning Lemmata 1.3 and 1.4, it would be possible to devise a procedure yielding for every A of Lemma 2.3 a unique $(Q(P_1 \ldots P_j))(x_1 \ldots x_k)$, and analogously with Lemma 2.4.

As corollaries of Lemmata 2.3 and 2.4 we get the following two theorems, which we take as giving the essence of our variable-free formulation of second-order logic:

2.1. **Theorem**: For every formula A of L^2 without (2), which has no free variables, there is a formula A' made only of expressions of (2) of L^2, such that in S^2 we can prove $A \leftrightarrow A'$.

2.2. **Theorem**: For every formula A' made only of expressions of (2) of L^2 there is a formula A of L^2 without (2), which has no free variables, such that in S^2 we can prove $A' \leftrightarrow A$.

The fact that in (0) of L^1 there are no individual constants, and in (0) of L^2 there are no constant expressions at all, does not diminish the generality of our results. If there were such constant expressions of the categories n and p_k in (0), Lemmata 1.3 and 1.4, and Lemmata 2.3 and 2.4, would give enough information of what would happen with these constant expressions in the variable-free formulations.

Our variable-free formulation of second-order logic extends Quine's variable-free formulation of first-order logic by first duplicating at a different level Quine's predicate functors, and next adding some completely new functors. These completely new functors are the combinatory predicate functors Id_k and the combinatory functors Comp^2_j and $\mathrm{Comp}^2_{i,j}$. And among these it is the last two types of functors which represents the essential addition. The functors Id_k are introduced for more technical reasons, as can be seen in the basis of the induction in the proof of Lemma 2.3, and also in Section 3 below. (The functors Id_k for $k \geqslant 2$ are anyway dispensable, since they are easily definable in terms of Inv_{k-2} or inv_{k-2}.) The functors Comp^2_j and $\mathrm{Comp}^2_{i,j}$ are just instances of combinatory functors, corresponding to *functional composition*, of the categories $q_{j_1+\ldots+j_m}/(q_m \circ q_{j_1} \circ \ldots \circ q_{j_m})$, where Comp^2_j and $\mathrm{Comp}^2_{i,j}$ cover the cases when m is 1 and 2.

Functors corresponding to functional composition would be expected for variable-free formulations of a first-order or second-order language with

function symbols. (We shall not enquire here what functors would be needed for such formulations.) But they may seem rather unexpected in a variable-free formulation of a language without function symbols, like our second-order language L^2 without (2). We suppose that their presence in our variable-free formulation of this second-order language shows in what way the combinatory role of bound variables in second-order logic is more complex than the combinatory role of bound variables in first-order logic.

3. Remarks on categorial grammar

This section will be organized as follows. After stating what in L^1 and L^2 might induce us to seek a flexible categorial apparatus, we shall argue that such an apparatus can be provided by a commutative variant of Lambek's calculus of syntactic categories. We shall suggest that this flexibility might be useful when we want to take L^1 and L^2 as tools for the analysis of natural language. We shall also show that though Lambek's original noncommutative calculus cannot give us all that its commutative variant can, with it we can still achieve some flexibility for our languages.

Concerning the categorial apparatus needed to describe L^1 we make the following remarks. If we assumed that $s/s \rightarrow p_k/p_k$ and that $s/(s \circ s) \rightarrow p_{k+m}/(p_k \circ p_m)$, we would have no need for $\neg_k$ and $\wedge_{k,m}$, and $\neg$ and $\wedge$ would suffice. With similar assumptions about category raising we could dispense with the sequences of predicate functors $\exists_k$, Inv_k, inv_k and Ref_k, and introduce only the predicate functors $\exists$ of the category s/p_1, Inv and inv of the category p_2/p_2, and Ref of the category p_1/p_2.

Analogous, though in general more involved, remarks about category raising could be made for the functors of L^2. However, the categorial apparatus connected with L^2 invites also remarks which are more specific to this context. We have said that the functors $\exists^2_j$ are of the categories q_j/q_{j+1}. For example, the functors $\exists^2_0$ would be of the categories q_0/q_1, i.e. $p_k/(p_k/p_{k_1})$. We say "the functors $\exists^2_0$" in plural, because $\exists^2_0$ is just an abbreviation for $\exists^2_{k,k_1}$, and then depending on k and k_1 we get a sequence of functors. So, $\exists^2_0$ in the formulae of L^2 on the left-hand side is of the categories indicated on the right-hand side of the following list:

$(\exists^2_0 \neg_1)x$	$p_1/(p_1/p_1)$
$\exists^2_0 \exists_0$	$p_0/(p_0/p_1)$, i.e. $s/(s/p_1)$
$(\exists^2_0 \mathrm{Inv}_0)(x_1 x_2)$	$p_2/(p_2/p_2)$
$(\exists^2_0 \mathrm{Ref}_0)x$	$p_1/(p_1/p_2)$.

We can either take this situation as it is, or, if we are unhappy about this proliferation of functors $\exists_0^2$, we can require from our categorial apparatus to reduce the number of categories on the right-hand side above.

Let us consider another question of categorial grammar connected with our language L^2. First, we shall formulate without variables the following two formulae of L_2: $\exists P(\exists x(Px))$ and $\exists x(\exists P(Px))$. For the first formula we have in S^2

$$\begin{aligned}\exists P(\exists x(Px)) &\leftrightarrow \exists P(\exists_0 P)\\ &\leftrightarrow \exists_0^2\exists_0\end{aligned}$$

whereas for the second we have

$$\begin{aligned}\exists x(\exists P(Px)) &\leftrightarrow \exists x(\exists P((\mathrm{Id}_1 P)x))\\ &\leftrightarrow \exists x((\exists_0^2\mathrm{Id}_1)x)\\ &\leftrightarrow \exists_0(\exists_0^2\mathrm{Id}_1).\end{aligned}$$

These translations are not completely symmetric, due to the presence of Id_1 in the second translation; and Id_1 is present because we must take the functor $\exists_0^2$ with an argument from a category q_1. It seems it would be quite natural if we could apply the functor $\exists_0^2$ directly to x in order to produce the formula $\exists_0^2 x$, as the quantifier expression $\exists P...P...$ (i.e. the prefix $\exists P$ and the bound variable P taken as a single whole) seems to be applied in $\exists P(Px)$. This means that we would need a functor $\exists_0^2$ which is also of the category p_1. More formally, we could extend our system S_2 with the equivalence

$$(\exists_0^2\mathrm{Id}_1)x \leftrightarrow \exists_0^2 x$$

and we would have to explain how $\exists_0^2$ which is of a category $p_k/(p_k/p_{k_1})$ is also of the category p_1.

So it seems that in more than one respect our language L^2 invites a flexible categorial apparatus. The cheapest way to achieve this flexibility would be to keep the calculus M of Section 1 and discard the rule by which at the beginning we assign only a single category to the primitive expressions. However, in that case we might have difficulties in verifying whether an expression e is of a certain category b, since e could now have an infinite number of categories. An alternative, which we shall follow, is to keep the rule above, and to leave it to the calculus to provide the reduction principles we need. Of course, our calculus should be decidable if we want to be able to determine in every case whether an expression e is of a certain category b.

We shall now introduce a variant of Lambek's calculus of syntactic categories of Lambek 1958, and we shall argue that this variant enables us to construct the flexible categorial apparatus we want. (For the remainder of this paper we assume a certain familiarity with Lambek 1958.) Our variant of Lambek's calculus is closely related to a system suggested by van Benthem (see Buszkowski 1984, which refers to a paper by van Benthem due to appear in this book). It differs from Lambek's calculus in assuming the commutativity of $\circ$, and in conflating the two bars / and \ into the single / bar. We shall call our variant of Lambek's calculus C, whereas the original Lambek calculus will be called L.

The language of C is identical with the language of the calculus M of Section 1. For C we assume the following axiom-schemata and rules:

(i) $a \to a$

(ii) $(a \circ b) \circ c \rightleftarrows a \circ (b \circ c)$

(v) $\dfrac{a \to b \qquad b \to c}{a \to c}$

(vi) $a \circ b \to b \circ a$

(vii) $\dfrac{a \circ c \to b}{a \to b/c}$ (viii) $\dfrac{a \to b/c}{a \circ c \to b}$.

It is not difficult to show that if in any formula provable in L (see Lambek 1958, p. 163) we replace terms of the form $b\backslash a$ by terms of the form a/b, we obtain a formula provable in C (note that we use the schematic letters a,b,c where Lambek uses x,y,z, and that we do not omit $\circ$ as Lambek does.) So we can take L as a proper subsystem of C: an alternative way to formulate C would be to add to L either the axiom-schema $a \circ b \to b \circ a$, or the axiom-schema $a/b \to b\backslash a$, or the axiom-schema $b\backslash a \to a/b$. It is quite easy to show that M is a proper subsystem of both L and C.

A Gentzen formulation Cg of C is obtained when in the Gentzen formulation of L in Lambek 1958 (p. 165) we discard the rules (2′) and (3′), and add a new (structural) rule:

$$\text{Permutation } \frac{U,a,b,V \to c}{U,b,a,V \to c}.$$

A proof of the admisibility of Cut in Cg can be easily obtained by adapting Lambek's proof (1958, pp. 167-169; it is enough to note that in a proof an application of Permutation need never precede immediately an application of Cut). Using Cg it is not difficult to show that C is a conservative extension of van Benthem's calculus LBC (see Buszkowski 1984), i.e. that LBC

gives the fragment of C with $\circ$ omitted. Again using Cg it is not difficult to show that C is decidable.

The reason for working with C rather than L is that L would not give us all the reduction principles we wanted. To stress that, we shall note when something is provable in L, and hence in both C and L, and when something is provable only in C. We shall not give complete proofs in C, but short sketches at best. In making these sketches we shall use the fact that in both L and C the following rules are derivable (cf. Lambek 1958, p. 164):

(iv) $$\frac{a \to a' \qquad b \to b'}{a \circ b \to a' \circ b'}$$

(ix) $$\frac{a \to a' \qquad b \to b'}{a/b' \to a'/b}\,.$$

Using these rules it is easy to show that the following rule of replacement is also derivable in both L and C:

(x) $$\frac{a \rightleftarrows a' \quad \varphi}{\varphi'}$$

where φ' is obtained from the formula φ of the language of C by replacing zero or more occurrences of a by a'.

Let us now try to see what category raising principles C can give us, and let us first concentrate on L^1. At the beginning of this section we said that it would simplify matters if we had the category raising principle $s/s \to p_k/p_k$. Now, in L we can prove

(xi) $a/b \to (a/c)/(b/c)$

which gives the category raising we needed. Next we said that we might need $s/(s \circ s) \to p_{k+m}/(p_k \circ p_m)$. It is easy to see that in C (but not in L) we have

$$(a/(b \circ c)) \circ (b/b') \circ (c/c') \circ b' \circ c' \to a$$

from which we easily obtain

(xii) $a/(b \circ c) \to (a/(b' \circ c'))/((b/b') \circ (c/c'))$

which is exactly what we needed. Other category raising principles we hinted at are: $s/p_1 \to p_k/p_{k+1}$, $p_2/p_2 \to p_{k+2}/p_{k+2}$ and $p_1/p_2 \to p_{k+1}/p_{k+2}$. Now, in L we can prove

(xiii) $a/(b \circ c) \rightleftarrows (a/c)/b$.

Note that this is not exactly the same as $a/(b \circ c) \rightleftarrows (a/b)/c$, which is provable only in C. From (xi) we obtain easily the following two formulae:

$$a/(b/b') \to (a/c)/((b/b')/c)$$
$$(a/a')/(b/b') \to ((a/a')/c)/((b/b')/c)$$

which with the help of (xiii) and (x) give us

(xiv) $a/(b/b') \to (a/c)/(b/(c \circ b'))$
(xv) $(a/a')/(b/b') \to (a/(c \circ a'))/(b/(c \circ b'))$

taking care in L of the three category raising principles above. So, with the help of C we could dispense with $\neg|_k$ and $\wedge_{k,m}$, and we would have the following assignment of categories in the language L^1:

$\exists$	s/p_1
Inv and inv	p_2/p_2
Ref	p_1/p_2.

With L instead of C we would obtain all that, except the discarding of $\wedge_{k,m}$.

Let us now consider matters of category raising specific to L^2. Instead of the expressions of (2) of L^2 we could introduce the expressions on the left of the following list with the assignment of categories shown on the right:

Id	s/s
$\exists^2_{k_{j+1}}$	$s/(s/p_{k_{j+1}})$
$\mathrm{Inv}^2_{k_1,k_2}$ and $\mathrm{inv}^2_{k_1,k_2}$	$(s/(p_{k_2} \circ p_{k_1}))/(s/(p_{k_1} \circ p_{k_2}))$
$\mathrm{Ref}^2_{k_{j+1}}$	$(s/p_{k_{j+1}})/(s/(p_{k_{j+1}} \circ p_{k_{j+1}}))$
$\mathrm{Comp}^2_{k,m,k_j}$	$(p_k/p_{k_j})/((p_k/p_m) \circ (p_m/p_{k_j}))$, where p_k/p_m is either s/s, or s/p_1, or p_2/p_2, or p_1/p_2
$\mathrm{Comp}^2_{k,m,n_1,n_2}$	$q_2'/(q_2 \circ q_1' \circ q_1'')$, where q_2' is $p_{k+m}/(p_{n_1} \circ p_{n_2})$, q_2 is $p_{k+m}/(p_k \circ p_m)$, q_1' is p_k/p_{n_1}, and q_1'' is p_m/p_{n_2}.

We shall now justify this replacement. For Id, as for $\neg|$, it is enough to establish $s/s \to p_k/p_k$, which is an instance of (xi), and hence obtainable in L. For $\exists^2_{k_{j+1}}$ we note that using (xiv), (vi), (xiii) and (x) in C (but not in L) we can prove

(xvi) $a/(b/b') \to (a/c)/((b/c)/b')$

which justifies $s/(s/p_{k_{j+1}}) \to p_k/(p_k/p_{k_{j+1}})$. Next, in order to obtain

$p_k/(p_k/p_{k_{j+1}}) \to q_j/q_{j+1}$ we apply (xi), (xiii) and (x). Note that this last reduction is obtainable in L too. For $\mathrm{Inv}^2_{k_1,k_2}$, $\mathrm{inv}^2_{k_1,k_2}$ and $\mathrm{Ref}^2_{k_{j+1}}$ we note that analogously to what we had for (xvi), using (xv), (vi), (xiii) and (x), in C (but not in L) we can prove

(xvii) $(a/a')/(b/b') \to ((a/c)/a')/((b/c)/b')$

which, by applying (xi), (xiii), (vi) and (x), yields that the categories of $\mathrm{Inv}^2_{k_1,k_2}$ and $\mathrm{inv}^2_{k_1,k_2}$ on the list above reduce to q'_{j+2}/q_{j+2}, and that the categories of $\mathrm{Ref}^2_{k_{j+1}}$ on the list above reduce to q_{j+1}/q_{j+2}. Note that in L we can obtain that $(p_k/(p_{k_{j+2}} \circ p_{k_{j+1}}))/(p_k/(p_{k_{j+1}} \circ p_{k_{j+2}}))$ reduces to $(p_k/(P \circ p_{k_{j+2}} \circ p_{k_{j+1}}))/(p_k/(P \circ p_{k_{j+1}} \circ p_{k_{j+2}}))$, where P is $p_{k_1} \circ \ldots \circ p_{k_j}$. This reduction covers an inv^2 working at the end of $p_{k_1} \circ \ldots \circ p_{k_{j+2}}$, rather than at the beginning as inv^2_j did. A similar move is not available for Inv^2, and it seems we must use the commutativity of $\circ$ to achieve some kind of category raising. On the other hand, we have in L that $(p_k/p_{k_{j+1}})/(p_k/(p_{k_{j+1}} \circ p_{k_{j+1}}))$ reduces to q_{j+1}/q_{j+2} which covers the case of Ref^2_j.

For $\mathrm{Comp}^2_{k,m,k_j}$ we note first that in L we can prove

(xviii) $a \circ (b/c) \to (a \circ b)/c$

which with (ix) gives

(xix) $a'/((a \circ b)/c) \to a'/(a \circ (b/c))$.

Now with (xi) we reduce $(p_k/p_{k_j})/((p_k/p_m) \circ (p_m/p_{k_j}))$ to $((p_k/p_{k_j})/c)/(((p_k/p_m) \circ (p_m/p_{k_j}))/c)$, where c is $p_{k_1} \circ \ldots \circ p_{k_{j-1}}$, and then using (xix) and (x) to "push c under" p_m/p_{k_j}, and applying (xiii) and (x), we obtain in L the reduction to $q_j/(q_1 \circ q'_j)$. For $\mathrm{Comp}^2_{k,m,n_1,n_2}$ we proceed analogously, to obtain that in C (but not in L) the categories of these functors reduce to $q_{i+j}/(q_2 \circ q_i \circ q_j)$. (Note that by using (xii), i.e., $s/(s \circ s) \to p_{k+m}/(p_k \circ p_m)$, and also (ix), (xi), (xiii), (vi) and (x), $q_{i+j}/(q_2 \circ q_i \circ q_j)$ reduces to $q_{i+j}/((s/(s \circ s)) \circ q_i \circ q_j)$, but not necessarily the other way round.)

If we are still unhappy with the complexity of the categories assigned to $\mathrm{Comp}^2_{k,m,k_j}$ and $\mathrm{Comp}^2_{k,m,n_1,n_2}$, we can replace these functors by $\mathrm{Comp}^2_{k,m}$ of the categories $(p_k/p_m)/(p_k/p_m)$. In L we can prove:

$$\begin{aligned}(p_k/p_m)/(p_k/p_m) &\to p_k/((p_k/p_m) \circ p_m), \text{ by (xiii)} \\ &\to (p_k/p_{k_j})/((p_k/p_m) \circ (p_m/p_{k_j})), \text{ by (xi), (xix) and (x)}\end{aligned}$$

and we proceed analogously to show that the categories of $\mathrm{Comp}^2_{k,m}$ reduce in C to the categories of $\mathrm{Comp}^2_{k,m,n_1,n_2}$. However, if we make this replacement, we need extra equivalences in S^2, of the type:

$$((\mathrm{Comp}^2_{k,m}Q)F)\vec{x} \leftrightarrow (QF)\,\vec{x}$$

to make for cases where $\mathrm{Comp}^2_{k,m}$ is "wrongly" applied according to the old category assignments.

We conclude our consideration of category raising principles for L^2 with the following remark. Though with L we can achieve for Id_k and Comp^2_j what we have achieved with C, and though L enables us to achieve some reductions for $\exists^2_j$, Ref^2_j, and a form of inv^2_j, we cannot obtain with it everything we wanted. In particular, if we were content to assume $\exists^2$ of the categories $p_k/(p_k/p_m)$, we could not achieve any economy in the four categories assigned to $\exists^2_0$ at the beginning of this section, viz. $p_1/(p_1/p_1)$, $s/(s/p_1)$, $p_2/(p_2/p_2)$ and $p_1/(p_1/p_2)$. On the other hand, with the assignment $s/(s/p_m)$, it is enough to have just two categories: $s/(s/p_1)$ for the first two cases, and $s/(s/p_2)$ for the second two cases.

At the beginning of this section we have also said that for $\exists^2_0$ we might need a principle reducing a category $p_k/(p_k/p_1)$ to the category p_1. In this matter C can again help us. First we note that in C we can prove $(s/n)\circ n \rightarrow s$, and hence also

(xx) $\quad n \rightarrow s/(s/n)$.

(In L we cannot prove exactly that, but we can prove $n \rightarrow (s/n)\backslash s$ and $n \rightarrow s/(n\backslash s)$.) The category raising principle (xx) says roughly that names are also first-order quantifiers (of course, this does not mean that first-order quantifiers are also names). This principle could be intuitively justified as follows. Let us articulate the sentence 'John cries' as '$(\exists x$ = John) (x cries)' (instead of ($\exists x$ = John) we could as well use ($\forall x$ = John), or ($\exists x \in$ {John}), or ($\forall x \in$ {John})), and let the name 'John' be an abbreviation for the first-order quantifier expression '$(\exists x$ = John)$\ldots x\ldots$'. So we can take it that 'John' is applied to the predicate 'cries' to produce 'John cries'. This intuitive justification of (xx) could be the basis of an intuitive justification of

(xxi) $\quad s/(s/p_1) \rightarrow p_1$

which is easily derived from (xx) with the help of (ix). (In L we could obtain $s/((s/n)\backslash s) \rightarrow s/n$ or $s/(s/(n\backslash s)) \rightarrow s/n$. Note that in C we also have the converse of (xxi), whereas in L we have the converse of the first reduction of the last sentence.) Now the quantifiers $\exists^2_0$ in $(\exists^2_0\mathrm{Id}_1)x$ and $\exists^2_0 x$ can be taken as a single quantifier $\exists^2_1$ of the category $s/(s/p_1)$.

We can conclude that though L can give a certain flexibility to our

categorial apparatus, it is with C that we obtain all the flexibility we called for. However, this flexibility has a price. In our languages L^1 and L^2, concatenation was not commutative. Working on the categorial apparatus of L^1 and L^2 with C transforms these languages into languages in which concatenation is commutative. For example, with C not only is Px a well-formed formula, but also xP. This is a peculiar situation for logical languages, and we do not wish to suggest that logical languages should be made commutative in this sense. But there are natural languages which are commutative to a significant degree (i.e. they have a "free" word order). If L^1 and L^2 are not considered on their own, but as tools for the analysis of such natural languages, we can perhaps take them as commutative, and thereby flexible. L^1 and L^2 are not considered on their own, but as tools for the analysis of such natural languages, we can perhaps take them as commutative, and thereby flexible.

The languages L^1 and L^2 do not seem to be ill suited for this analysis. Since natural language lacks the full apparatus of bound variables, we should expect that it has found ways to do the work of the functors of (2) of L^1 and L^2. Let us take a brief look at what in natural language could correspond to these functors, after they have been revised by C. The connectives $\wedge$ and $\neg$ correspond to ordinary conjunction and negation, and the quantifier expression $\exists$ corresponds to the quantifiers 'something' and 'somebody'. The role of Inv and inv may be played by the passive voice, which can transform $P(xy)$ into $P'(yx)$ equivalent to $P(xy)$. The role of Ref may be played by reflexive pronouns: if r is such a pronoun, and if it is substituted for y in $P(xy)$, the resulting expresion $P(xr)$ is equivalent to $P(xx)$ (cf. the chapter by Geach (1970)). It is unrealistic to expect that the role of the functors of (2) of L^2 will be mimicked by such simple devices. Second-order quantification seems to be present in natural language more semantically than syntactically. But to the extent that it is present, we might expect to find in natural language traces of the functors of our variable-free formulation of second-order logic.

ACKNOWLEDGEMENT

I would like to thank Wojciech Buszkowski, Leon Kojen, Johan van Benthem and Slobodan Vujošević for their valuable comments and remarks.

REFERENCES

Ajdukiewicz, K. (1967): "On syntactical coherence." *Review of Metaphysics* 20, 635-647 (Part I of K. Ajdukiewicz (1935): "Die syntaktische Konnexität", *Studia Philosophica* 1, 1-27, translated by P.T. Geach from the Polish text in K. Ajdukiewicz (1960): *Język i poznanie*, Warsaw).

Buszkowski, W. (1984): "A note on the Lambek-van Benthem calculus." *Bulletin of the Section of Logic* 13, (1), 31-37.

Geach, P.T. (1970): "A program for syntax." *Synthese* 22, 3-17 (reprinted in Harman and Davidson (Eds.) (1972): *Semantics of Natural Language*, Dordrecht).

Kuhn, S.T. (1983): "An axiomatization of predicate functor logic." *Notre Dame Journal of Formal Logic* 24, 233-241.

Lambek, J. (1958): "The mathematics of sentence structure." *American Mathematical Monthly* 65, 154-170.

Quine, W.V. (1960): "Variables explained away." *Proceedings of the American Philosophical Society* 104, 343-347 (reprinted in W.V. Quine (1966): *Selected Logic Papers*, New York)

Quine, W.V. (1971): "Algebraic logic and predicate functors." In: Rudner and Scheffler (Eds.), *Logic and Art: Essays in Honor of Nelson Goodman*. Indianapolis. (reprinted with emendations in W.V. Quine (1976): *The Ways of Paradox and Other Essays*, 2nd edition, Cambridge, MA).

Quine, W.V. (1981): "Predicate functors revisited." *Journal of Symbolic Logic* 46, 649-652.