An introduction to predicate-functor logic

Mats Dahllöf
[matsd@stp.ling.uu.se]
Dept of Linguistics, Uppsala University

December, 1997

1 Introduction

This paper[1] addresses the issue of the semantic work done by variables and variable-binding operators in predicate calculus and similar formalisms. Quine has shown that variables can be replaced, without loss or gain in semantic power, by other formal devices. In this way we get a new "picture" of how predicate calculus works. This investigation is motivated, on the one hand, by the fact that variables are very important, and on the other hand by the conceptually problematic "semantics" of variables: They do some kind of semantic work, but do not have an ordinary semantic value.

One way of investigating what a device is good for is to remove it and see what other instruments would be needed to do its work. Variables will be investigated in this way here.

2 Ordinary predicate calculus with variables

Only atomic expressions denote predicates in standard predicate calculus: It lacks a direct compositional semantics for predicates and deserves the name "*predicate* calculus" only indirectly: A formula with n different free variables does, in a sense, stand for an n-place predicate, but not simply by denoting one. Compositionally derived predicates are "present" as formulae which are true (or false) relative to variable-assignment functions. Variables necessitate a notion of truth (of formulae) as relative not only to an interpretation, but also to a variable-assigment function. The notion of variable-assignment-relative truth is necessary as a means of defining the main (only *interpretation*-relative) notion of truth.

From the point of view of linguistics, interpretation-relative denotations like truth values and predicates (extensions) have a clear connection to intuitive se-

[1] **This paper is an unpublished draft. Comments are welcome.**

mantic properties, whereas interpretation-and-variable-assignment-relative truth is a much more exotic notion, at least as used in predicate calculus. A simple statement like "someone sleeps" would, for instance, be rendered as '$\exists x S(x)$'. The introduction of variables here (and in other semantic constructions) shows that predicate calculus is very different from natural language with respect to its modes of composition. (This use of variables and variable-binding operators is due to Frege's genial and ingenious logic *Begriffsschrift*.)

If we try to view open formulae as predicate-denoting expressions, variables complicate the picture considerably: Formula F 1, for instance, may be intended to stand for the three-place relation that holds of three objects when the first (x) is larger than the second (y) and the second is larger than the third (z).

F 1 $\mathsf{larger\text{-}than}(x, y) \wedge \mathsf{larger\text{-}than}(y, z)$

The two subordinate formulae '$\mathsf{larger\text{-}than}(x, y)$' and '$\mathsf{larger\text{-}than}(y, z)$' intuitively stand for the same two-place relation. The important thing is that two different variables occupy the argument places (otherwise we would have defined the property "larger than itself"). The choice of variables is immaterial from the point of view of the subformulae themselves. When they are conjoined, however, the fact that the second argument position of the first subformula is occupied by a variable identical to that found in the first argument position of the second subformula is crucial, as the two argument positions thereby are connected in a semantically significant way. The conjunction of two instances of a two-place predicate in this case defines a three-place predicate. If we take the semantic value (relative to an interpretation) of a formula with free variables to be the predicate it in this way defines (given an ordering of the variables), then we find that predicate logic with variables does not possess a compositional semantics:[2] The semantic value of the conjunction formula F 1 is not possible to determine from the semantic values of the conjuncts, because of the significance of the identity of variables. A formula like formula F 2, below, would define another predicate, but is just like formula F 1 a conjunction whose conjuncts (we have supposed) stand for the relation of being larger than.

F 2 $\mathsf{larger\text{-}than}(x, y) \wedge \mathsf{larger\text{-}than}(x, z)$

[2] It might be objected here that variables are only a syntactic device and that their only function is to define by which "mode of combination" predicates are joined together. This suggestion implies that there is an infinity of such modes of combination, as there is no upper limit to the number of variables that may occur in a formula. (Compositionality is thereby in a sense saved: The semantic value of the whole is determined by the semantic values of its constituents and their modes of combination.) However, having an infinite number of modes of combination is clearly unacceptable and this view of the the syntax is inadequate: Variable cooccurrence is not a matter of syntactic rule application. For instance, the conjunction rule in an ordinary first-order logic syntax like the one given by Dowty, Wall, and Peters [5, p. 56–57] (rule B. 3) or by Allwood, Andersson, and Dahl [1, p. 71–72] (rule (g) (iii)) just says that two formulae joined by a conjunction sign form a new formula. The syntactic combination rules do not mention the variables occurring in the formulae.

This lack of "naive" compositionality is, of course, due to the presence of variables. There are *two* systems of semantically significant structure in ordinary predicate calculus: the constituent structure (as defined by the syntactic modes of combination) and what we could call the variable cooccurrence structure.

A variable does not contribute to the semantic value of an expression by having a certain semantic value itself. Its significance is due to its being bound from the outside, by being identical to another variable token occurring in another expression. So, when we combine the two conjuncts in formula F 2 into a conjunction, the semantic value of the conjunction depends on the semantic values of the conjuncts and on *both* the semantic operation represented by conjunction and the identification of two variable occurrences. The latter factor is reflected neither in the syntactic operation (conjoining) or in the semantic values of the conjuncts (as naively understood). The difficult situation lies in the fact that semantically significant links within a formula are established both by means of constituent syntax (as defined by the syntactic rules) and by means of variable cooccurrence.

A traditional semantics for ordinary predicate calculus deals with this situation and establishes compositionality with the help of certain technical manœuvres, which invalidate the idea that open formulae represent predicates. One method is to define semantic values as relative to an interpretation *and* a **value assignment**, which is a function assigning values to variables.[3] (Other methods have to introduce some similar kind of further relativity.) Open formulae are then taken to denote a truth value relative to an interpretation *and* a value assignment function. The truth value of formulae without free variables will not be sensitive to the value assignment, and *their* truth values are consequently only interpretation-dependent.

This method saves variables and compositionality. The disadvantage is that the system of semantic values becomes more complex and artificial: Instead of having the intuitively simple picture with only true expressions (sentences), and expressions true of extra-logical entities (predicates), we also have to accept expressions true relative to variable assignments. This complication also comes with the *restriction* that predicate expressions cannot be compositional. (Addition of λ-terms may be a way of removing that restriction in a variable-based calculus.)

The problems may be summarised as follows: The only kind of interpretation-relative complex denotations are associated with predicate constants, which are atomic symbols, and the syntactically complex expressions (formulae) are only assigned truth-values, which do not possess any significant structure, by interpretations. *Syntactically simple objects denote complex ones, and syntactically complex objects denote simple ones*, to put it briefly.

Variables (and the difficulties they introduce) are in no way essential to first-

[3] See Dowty, Wall, and Peters [5, p. 59–61] for a semantics for first-order predicate calculus along these lines.

order predicate calculus (or to most other variable-based calculi). Quine has shown this by inventing a variable-free predicate calculus. His calculus uses a small number of functors, whereby new predicates are defined in terms of primitive or already defined one. These functors are responsible for cross-identification and rearrangement of argument places, and quantification. They take over the tasks performed by variables. This **predicate-functor logic** is precisely as powerful as ordinary first-order predicate calculus (with variables and variable-binding quantifiers). This variable-free language is semantically perspicuous: Every expression denotes a semantic value of the intuitively right kind (i.e. a truth value, a predicate extension, or an individual) relative to an interpretation and the semantics is perfectly compositional. Quine's calculus consequently offers an alternative way of organising the compositional semantics of first-order predicate calculus (viewed as a category of calculus defined by its semantic power).

3 Predicate-functor logic

The most important aspect of a predicate-functor logic is its inventory of predicate-functors. Quine has suggested a few alternative ones (in his papers [11], [12], [14], [15], and [16], also cf. [13]). Between four and seven predicate-functors are required. Quine's languages do not contain individual constants, but the calculus presented below will.[4]

I will use Quine's set of six functors from the 1960 paper "Variables Explained Away" [11] . The notation will however be slightly modified to suit my purposes (and taste) better. Quine [14] has shown how to manage with only four functors, but this gain in economy leads to a corresponding complication of expression. I think that the 1960 set is easier to understand and use. The language discussed here is a first-order logic with identity (i.e. an identity predicate is included among the logical constants). There are no other logical constants than these seven (viz. the six functors and the identity predicate).[5]

There are two kinds of terms in the present predicate-functor logic: **predicate terms** and **individual constants**. A predicate term consists of a predicate constant or is a complex expression. Predicate terms are of a certain degree (arity). Formulae may be identified with predicate terms of degree 0. The six predicate functors are to be represented by the symbols '$\imath$', '$\overline{\imath}$', 'Σ', '$\mathfrak{C}$', '$\boxtimes$', and '$\rfloor$'. I will use a predicate-first notation, in order to eliminate the need for parentheses. This means that a predicate will precede its individual constant arguments, but that

[4] For some results on predicate-functor logic, see Noah [9], Grünberg, [7], Kuhn [8], and Bacon [2]. Predicate-functor logic is applied to natural language semantics by Grandy [6], who suggests that "linguists, psychologists and philosophers should pay more attention to algebraic formulations of logic when discussing the logical form of natural language sentences" [p. 398], and by Purdy [10].

[5] Those who are interested in the details of Quine's languages and want to compare them to each other and to the one defined here are referred to his papers.

predicate functors will be placed after the predicate term(s) they operate on. (This notation makes it natural to think of individual constants as being applied to predicate terms rather than vice versa[6].) This syntax implies that complex predicate terms are always formed by strings whose first element is a predicate constant and whose last element is a functor or an individual constant (as complex predicate terms are formed by the application of a functor or an individual constant.) Reading from left to right gives us a bottom-up perspective on how undefined predicates given by the predicate constants are combined into complex ones by means of functor applications.

The two functors '$\imath$' and '$\bar{\imath}$' are used to permute argument places in predicates. Applied to a predicate term they yield a predicate term of the same degree. The **minor inversion** functor '$\imath$' interchanges the two first argument places, whereas the **major inversion** functor '$\bar{\imath}$' puts the first argument place last, so to speak. (Applied to a two-place predicate the major and minor inversions are identical.) These functors are necessary because the 'Σ' and '$\beth$' functors (see below) operate on the first one or two argument places of a predicate. It is therefore crucial that any argument place can be "moved" to the position where a 'Σ' or '$\beth$' functor can "reach" it. If we use a kind of mixture notation we may describe the effects of the two inversion functors as follows (taking variables to be implicitly universally quantified and P to be an arbitrary n-place predicate).

$$P\imath x_2 x_1 x_3 \ldots x_n \equiv P x_1 x_2 x_3 \ldots x_n$$
$$P\bar{\imath} x_2 \ldots x_n x_1 \equiv P x_1 x_2 \ldots x_n$$

The two kinds of inversion are sufficient to define any argument place permutation. (There are $n!$ such "permutations" of an n-place predicate, including its original argument place ordering.) For instance, the fact that all six argument place permutations of a three-place predicate may be obtained by applications of these two operators is illustrated by the following six equivalent formulae (note that '$((((P\imath)\bar{\imath})a)c)b$' is the syntactic structure of '$P\imath\bar{\imath}acb$'):

$$Pabc \equiv P\imath\bar{\imath}acb \equiv P\imath bac \equiv P\bar{\imath}bca \equiv P\overline{\imath\imath}cab \equiv P\bar{\imath}\imath cba$$

It is easy to verify that the six formulae above are equivalent. Note also that further applications of the '$\imath$' and '$\bar{\imath}$' functors on the predicate terms can only yield a new predicate term whose denotation is identical to the denotation of one of the formulae above. So, if $[\![x]\!]$ is the semantic value (i.e. denotation) of an expression x (relative to a given interpretation), $[\![P\imath\imath]\!] = [\![P\overline{\imath\imath}\imath]\!] = [\![P]\!]$ (provided P is of degree 3). And, if P is a two-place predicate, we have, for instance, $[\![P]\!] = [\![P\imath\imath]\!] = [\![P\bar{\imath}\imath]\!] = [\![P\imath\bar{\imath}]\!]$ and $[\![P\imath]\!] = [\![P\imath\imath\imath]\!] = [\![P\bar{\imath}\imath\bar{\imath}]\!] = [\![P\imath\overline{\imath\imath}]\!]$.

The **reflection** functor 'Σ' is used to identify two argument places (compare a reflexive pronoun). It operates on an at least two-place predicate term, identifies

[6] Singular terms are thereby assimilated to quantifiers in a way reminiscent of Barwise and Cooper's [3] treatment of **generalised quantifiers**.

the first two argument places, and consequently yields a predicate term whose degree is one step lower than the term to which it is applied. If 'L' is a two-place predicate symbol corresponding to the verb *love*, '$\mathsf{L}\Sigma$' corresponds to *love oneself*. Generally it holds:

$$P\Sigma x_1 \ldots x_n \equiv Px_1x_1 \ldots x_n$$

This functor ('Σ') encodes information that is encoded by means of variables in ordinary predicate calculus, where the identification of argument places is achieved by associating them with the same variable. One of the reasons that the inversion functors ('$\imath$' and '$\bar{\imath}$') are of crucial importance is that they allow any two argument places to be "moved" to the front positions where the 'Σ' functor may be used to unite them.

The **complement** functor[7] applies to a predicate term and yields the predicate that holds in all those cases where the first predicate does not hold. So, if 'L' is as above '$\mathsf{L}\mathfrak{C}$' stands for the relation of not loving. Applied to a formula, the complement functor yields one with the reverse truth value. The term obtained by an application of the complement functor is of the same degree as the operand term. The following formula (which again is expressed in mixture notation) clarifies the semantics of '$\mathfrak{C}$':

$$P\mathfrak{C}x_1 \ldots x_n \equiv \neg(Px_1 \ldots x_n)$$

The **Cartesian multiplication** functor '$\boxtimes$' is the only one that operates on two predicate terms. The degree of the Cartesian product is equal to the sum of the degrees of the operand predicate terms. If the operands represent the predicates P and Q, whose degrees are m and n respectively, their Cartesian product is the relation that holds of $m + n$ ordered individuals if and only if P holds of the first m ones and Q holds of the remaining n ones. This is made clearer by the following equivalence:

$$PQ\boxtimes x_1 \ldots x_m y_1 \ldots y_n \equiv (Px_1 \ldots x_m \wedge Qy_1 \ldots y_n)$$

So, if 'L' is as above, and 'H' stands for the predicate of being happy, '$\mathsf{HL}\boxtimes$' stands for the three-place relation holding of three individuals if and only if the first is happy and the second loves the third. An application of '$\boxtimes$' does not connect any argument places. (To do that is the purpose of the 'Σ' functor.) No connection between the three argument places in '$\mathsf{HL}\boxtimes$' example is consequently implied. By "concatenating" argument sequences in this way, the Cartesian multiplication functor makes it possible to apply the other functors in ways that connect argument places. For instance, '$\mathsf{HL}\boxtimes\Sigma$' stands for the relation that holds of two individuals if and only if the first is happy and loves the second.

The **cropping** or **derelativization** functor '$\exists$', finally, "removes" the first argument place by introducing an existential quantification. It consequently yields

[7] It is called "negation" in Quine [11], but the term "complement" is used in the later papers.

a predicate term whose degree is one step lower than that of the operand term. The '$\lrcorner$' "symbol is meant to connote excision of the left column" (Quine [14, p. 296]) of a relation. In mixture notation we may characterise its semantics in this way:

$$P\lrcorner x_2 \ldots x_n \equiv \exists x_1[Px_1 \ldots x_n]$$

If 'L' is as before, '$\mathsf{L}\lrcorner$' stands for the predicate of being loved by someone (the "subject" being the first argument of 'L') and '$\mathsf{L}\lrcorner\lrcorner$' for the formula saying that someone loves someone. The '$\lrcorner$' functor only operates on the first argument place of the predicate represented by the predicate term to which it applies. The importance of the inversion functors ('$\imath$' and '$\bar{\imath}$') in this calculus is again illustrated: They can "move" an argument place to the position where the '$\lrcorner$' functor may excise it (by cutting off the first item on each n-tuple in the predicate extension).

The predicate-functor logic based upon the six functors introduced here is as powerful as ordinary predicate logic (cf. Quine's papers on the subject). I will not produce a formal proof of this claim, but the discussion here and the examples in Section 3.3 should make this assertion convincing. A formal and quite detailed definition of the syntax and semantics of the present predicate-functor logic will now be given. We will then return to the question of how this notation relates to that of ordinary predicate calculus with variables. A few examples will illustrate this.

3.1 A formal summary of the syntax

Let us call the present formalism **Predicate-Functor Logic with Individual Constants**, abbreviated to PFLIC. The **lexicon** of a logical formalism contains the non-logical constants of the language. In the case of PFLIC it contains two kinds of constants: individual constants and predicate constants. A predicate constant is of a certain degree (arity) n and $n \geq 0$. (A PFLIC lexicon is just like the one an ordinary predicate calculus is based on.)

The following principles define the syntax of an instance of PFLIC if a lexicon is given. There are two kinds of expressions in an instance of PFLIC, **individual constants** and **predicate terms**. A predicate term is of a certain degree n and $n \geq 0$. Predicate terms of degree 0 are formulae (definition).

In the formulation of the syntax and semantics, I will use 'P' and 'Q' as meta-variables standing for arbitrary predicate terms, 'i' to stand for an arbitrary individual constant, and 'Φ' and 'Ψ' to stand for arbitrary formulae. The six predicate functor symbols and the identity predicate symbol will be used in the meta-language to denote the typographically identical symbols of PFLIC. A pair or triple of juxtaposed meta-variables stands for the expression obtained by concatenating, in the given order, the expressions denoted by the two or three meta-variables.

Individual constants are given directly by the lexicon, but predicate terms may be syntactically complex. The following rules define their syntax.

- A predicate constant of degree n is a predicate term of degree n.
- The identity symbol $\mathbb{I}$ is a predicate term of degree 2.
- If P is a predicate term of degree n, where $n \geq 1$, and i is a individual constant, then Pi is a predicate term of degree $n-1$.
- If P is a predicate term of degree n, where $n \geq 2$, then $P\imath$ (the minor inversion of P) is a predicate term of degree n.
- If P is a predicate term of degree n, where $n \geq 2$, then $P\bar{\imath}$ (the major inversion of P) is a predicate term of degree n.
- If P is a predicate term of degree n, where $n \geq 2$, then $P\Sigma$ (the reflection of P) is a predicate term of degree $n-1$.
- If P is a predicate term of degree n then $P\mathfrak{C}$ (the complement of P) is a predicate term of degree n.
- If P is a predicate term of degree m, and Q is a predicate term of degree n then $PQ\boxtimes$ (the Cartesian product of P and Q) is a predicate term of degree $m+n$.
- If P is a predicate term of degree n, where $n \geq 1$, then $P\rightthreetimes$ (the derelativization of P) is a predicate term of degree $n-1$.

There are no predicate terms except those defined by these principles.

3.2 A formal semantics for PFLIC

The semantics of PFLIC is to be defined with the help of ordinary model-theoretic techniques. An interpretation of an instance of PFLIC assigns semantic values to its non-logical constants and the semantic rules then fixes the semantic values of the composite expressions (all of which are predicate terms). An interpretation is an ordered pair $\langle \mathcal{D}, \mathcal{F} \rangle$, where $\mathcal{D}$ is a domain, i.e. the set of entities over which quantification ranges, and $\mathcal{F}$ an assignment function assigning a semantic value to each item in the lexicon. The two truth values are $\top$ (true) and $-$ (false). The following restrictions hold with respect to any interpretation $\mathcal{M} = \langle \mathcal{D}, \mathcal{F} \rangle$:

- If i is an individual constant, then $\mathcal{F}(i) \in \mathcal{D}$.
- If P is a predicate constant of of degree n, where $n \geq 1$, then $\mathcal{F}(P) \subseteq \mathcal{D}^n$.
- If P is a predicate constant of of degree 0, then $\mathcal{F}(P) \in \{\top, -\}$.

The following equations define the semantic values of composite predicate terms (given an interpretation $\mathcal{M} = \langle \mathcal{D}, \mathcal{F} \rangle$). Cases involving formulae are treated separately from those involving other predicate terms.

- If c is a non-logical constant, $[\![c]\!] = \mathcal{F}(c)$.
- $[\![\mathbb{I}]\!] = \{\langle x, x \rangle \mid x \in \mathcal{D}\}$.
- If Pi is a predicate term of degree n, where $n \geq 1$ and P is a predicate term of degree $n+1$ and i a individual constant, then $[\![Pi]\!] = \{\langle x_1, \ldots, x_n \rangle \mid \langle [\![i]\!], x_1, \ldots, x_n \rangle \in [\![P]\!]\}$.
- If $P\imath$ is a predicate term, where P is a predicate term of degree n, $n \geq 2$, then $[\![P\imath]\!] = \{\langle x_2, x_1 \ldots, x_n \rangle \mid \langle x_1, x_2, \ldots, x_n \rangle \in [\![P]\!]\}$.
- If $P\bar{\imath}$ is a predicate term, where P is a predicate term of degree n, $n \geq 2$, then $[\![P\bar{\imath}]\!] = \{\langle x_2, \ldots, x_n, x_1 \rangle \mid \langle x_1, x_2, \ldots, x_n \rangle \in [\![P]\!]\}$.
- If $P\Sigma$ is a predicate term, where P is a predicate term of degree n, $n \geq 2$, then $[\![P\Sigma]\!] = \{\langle x_1, \ldots, x_n \rangle \mid \langle x_1, x_1, \ldots, x_n \rangle \in [\![P]\!]\}$.
- If $P\mathfrak{C}$ is a predicate term, where P is a predicate term of degree n, where $n \geq 1$, then $[\![P\mathfrak{C}]\!] = \mathcal{D}^n - [\![P]\!]$.
- If $PQ\boxtimes$ is a predicate term of degree $m+n$, where P is a predicate term of degree m, $m \geq 1$ and Q is a predicate term of degree n, $n \geq 1$, then $[\![PQ\boxtimes]\!] = \{\langle x_1, \ldots, x_m, y_1, \ldots, y_n \rangle \mid \langle x_1, \ldots, x_m \rangle \in [\![P]\!]$ and $\langle y_1, \ldots, y_n \rangle \in [\![Q]\!]\}$.
- If $PQ\boxtimes$ is a predicate term of degree n, where P is a formula (i.e. a predicate term of degree 0), and Q is a predicate term of degree n, $n \geq 1$, then, if $[\![P]\!] = \top$, $[\![PQ\boxtimes]\!] = [\![Q]\!]$, otherwise (i.e. if $[\![P]\!] = -$), $[\![PQ\boxtimes]\!] = \emptyset$.
- If $PQ\boxtimes$ is a predicate term of degree n, where P is a predicate term of degree n, $n \geq 1$, and Q is a formula (i.e. a predicate term of degree 0), then, if $[\![Q]\!] = \top$, $[\![PQ\boxtimes]\!] = [\![P]\!]$, otherwise (i.e. if $[\![Q]\!] = -$), $[\![PQ\boxtimes]\!] = \emptyset$.
- If $P\beth$ is a predicate term, where P is a predicate term of degree n, $n \geq 2$, then $[\![P\beth]\!] = \{\langle x_2, \ldots, x_n \rangle \mid$ there is an x_1 such that $\langle x_1, x_2, \ldots, x_n \rangle \in [\![P]\!]\}$.
- If Pi is a formula, where P is a predicate term of degree 1 and i an individual constant, then $[\![Pi]\!] = \top$ if $[\![i]\!] \in [\![P]\!]$, otherwise $[\![Pi]\!] = -$.
- If $\Phi\mathfrak{C}$ is a formula, then $[\![\Phi\mathfrak{C}]\!] = \top$ if $[\![\Phi]\!] = -$, otherwise $[\![\Phi\mathfrak{C}]\!] = -$.
- If $\Phi\Psi\boxtimes$ is a formula, then $[\![\Phi\Psi\boxtimes]\!] = \top$ if $[\![\Phi]\!] = \top$ and $[\![\Psi]\!] = \top$, otherwise $[\![\Phi\Psi\boxtimes]\!] = -$.

- If $P\beth$ is a formula (where P is a predicate term of degree 1), then $[\![P\beth]\!] = \top$ if $[\![P]\!] \neq \emptyset$, otherwise $[\![P\beth]\!] = -$.

3.3 How the functors work

Ordinary predicate calculus with variables and PFLIC are semantically equivalent in this sense: If they share a lexicon (of non-logical constants), every PFLIC formula corresponds to an "ordinary" formula with precisely the same truth-conditions, and *vice versa* (Interpretations may be shared by an instance of PFLIC and one of ordinary first-order predicate calculus if the two languages contain the same non-logical constants.) Some discussion and examples should make the truth of this claim easier to see and facilitate the understanding of the PFLIC semantics.

A minor difference in notation is that in PFLIC predicate terms combine with one individual constant argument into a predicate term of a one step lower degree. So, '$P(a, b)$' in ordinary notation corresponds to 'Pab' in the notation introduced here, 'P' being two-place. In this formula, 'Pa' is a subexpression denoting a one-place predicate (that of being something which a is related to by the P-relation). In the ordinary predicate logic notation no constituent corresponds to this subexpression. (There are no composite expressions corresponding to predicates at all.)

Two functors in PFLIC do the work of the ordinary truth-functional operators. They are the complement functor ('$\complement$') and the Cartesian multiplication functor ('$\boxtimes$'), which correspond to negation and conjunction, respectively, when applied to formulae. These operators do not only operate on formulae, but also on predicate terms (which denote predicates). This means that we can conjoin predicate terms without having to supply the arguments that would combine with them into formulae.[8] So, if we assume that both 'P' and 'Q' represent two-place predicates and that 'a', 'b', 'c', and 'd' are individual constants, the formulae '$Pabc\boxtimes d$' and '$PQ\boxtimes abcd$' are equivalent. Any truth conditional composition of formulae can be defined in terms of negation and conjunction and the absence of disjunction, material implication and other truth-functional operators consequently does not weaken the expressive power of PFLIC.

The derelativization operator ('$\beth$') represents an existential quantification applying to the first argument place of a predicate. Universal quantification is easily defined in terms of the complement functor and existential quantification. (Remember that generally $\forall x\Phi(x) \equiv \neg\exists x\neg\Phi(x)$.) The quantification expressed by the '$\beth$' operator only involves the first argument place. This means that we must be able to connect argument places. This is done by means of the reflection func-

[8] The Cartesian multiplication operator is in this respect closer to natural language conjunction than is the ordinary purely truth-functional conjunction operator found in ordinary propositional and predicate calculus. However, natural language conjunction of (say) two property adjectives does not produce a two-place predicate expression. So, "ugly and expensive" corresponds to '$\mathsf{UE}\boxtimes\Sigma$' (the property of being both ugly and expensive), rather than simply to '$\mathsf{UE}\boxtimes$' (the relation that holds between any ugly thing and any expensive thing).

tor 'Σ', which connects the first two argument places of a predicate. The formula '$P\Sigma\rfloor$', for instance, corresponds to '$\exists x P(x,x)$' in ordinary syntax, where the identification of the two argument places is expressed by associating the same variable with the two places. In PFLIC this identification is performed by means of an application of 'Σ'.

Argument places must often be "moved" to come within the reach of the Σ functor. This need motivates the inversion functors '$\imath$' and '$\bar{\imath}$'. They allow us to obtain a predicate term standing for an arbitrarily permuted version of any predicate defined by a predicate term. In particular, any two argument places that we would like to identify by means of 'Σ' can be "moved" to the two initial positions.

If we consider formulae in a notation mixing elements from ordinary predicate calculus and PFLIC in a way that allows PFLIC predicate terms to occupy the position of ordinary predicate calculus predicate symbols, we can show the stepwise process of conversion from the one notation to the other. (The semantics of this mixture calculus is intuitively obvious and it would not be difficult to produce a formal semantics for it.) It should be noted that the order in which these steps are exhibited here is immaterial. Let us see how '$\forall x[\mathsf{M}(x) \rightarrow \mathsf{L}(x)]$' is recast to conform to the PFLIC syntax. (An expression within a box is a PFLIC predicate term.)

$\forall x[\boxed{\mathsf{M}}(x) \rightarrow \boxed{\mathsf{L}}(x)]$	[formula in ordinary notation]
$\neg\exists x\neg[\boxed{\mathsf{M}}(x) \rightarrow \boxed{\mathsf{L}}(x)]$	[universal quantifier eliminated]
$\neg\exists x\neg\neg[\boxed{\mathsf{M}}(x) \wedge \neg\boxed{\mathsf{L}}(x)]$	[material implication eliminated]
$\neg\exists x[\boxed{\mathsf{M}}(x) \wedge \neg\boxed{\mathsf{L}}(x)]$	[double negation eliminated]
$\neg\exists x[\boxed{\mathsf{M}}(x) \wedge \boxed{\mathsf{L}\mathfrak{C}}(x)]$	[(ordinary) negation eliminated]
$\neg\exists x\boxed{\mathsf{ML}\mathfrak{C}\boxtimes}(x,x)$	[(ordinary) conjunction eliminated]
$\neg\exists x\boxed{\mathsf{ML}\mathfrak{C}\boxtimes\Sigma}(x)$	[two argument places identified]
$\neg\boxed{\mathsf{ML}\mathfrak{C}\boxtimes\Sigma\rfloor}$	[existential quantifier eliminated]
$\boxed{\mathsf{ML}\mathfrak{C}\boxtimes\Sigma\rfloor\mathfrak{C}}$	[(ordinary) negation eliminated]

If 'M' and 'L' stand for the predicates to be a man and to be lazy, respectively, '$\mathsf{ML}\mathfrak{C}\boxtimes\Sigma\rfloor\mathfrak{C}$' says that all men are lazy. Let us dissect this formula to examine its parts. The term '$\mathsf{L}\mathfrak{C}$' stands for the predicate of not being lazy, and '$\mathsf{ML}\mathfrak{C}\boxtimes$' for the relation that holds between any man and anyone who is not lazy. '$\mathsf{ML}\mathfrak{C}\boxtimes\Sigma$', then, stands for the property of standing in this relation to oneself, i.e. for the property of being a man without being lazy. '$\mathsf{ML}\mathfrak{C}\boxtimes\Sigma\rfloor$' is a formula saying that there is at least one individual who has this property. '$\mathsf{ML}\mathfrak{C}\boxtimes\Sigma\rfloor\mathfrak{C}$', finally, is the denial of this, i.e. it says that there is no individual who is a man without being lazy. In other words: all men are lazy.

This example illustrates the differences between the ordinary notation for first-order logic and the PFLIC notation. In the former, predicates are represented only by simple expressions. Predicate symbols combine with individual constants and variables into atomic formulae and these formulae are combined into more

complex formulae by means of truth-functional composition, cross-identification of variables, and quantificational binding of variables. In PFLIC, on the other hand, everything is expressed by the modification or joining of predicate terms. In the mixture notation above, this difference is made clear by the way the logical constants are successively incorporated into the predicate terms and the predicate terms conjoined into larger ones as we step by step transform a formula in ordinary notation into one in the PFLIC notation.

Let us consider another example. The logical form of one reading of a sentence like "Every logician has read something written by Frege" may be rendered as '$\forall x[\mathsf{L}(x) \rightarrow \exists y[\mathsf{W}(y, \mathsf{f}) \wedge \mathsf{R}(x, y)]]$' (constants being interpreted in the obvious fashion). We may convert it into PFLIC notation as follows:

$\forall x[\boxed{\mathsf{L}}(x) \rightarrow \exists y[\boxed{\mathsf{W}}(y, \mathsf{f}) \wedge \boxed{\mathsf{R}}(x, y)]]$
$\neg\exists x\neg[\boxed{\mathsf{L}}(x) \rightarrow \exists y[\boxed{\mathsf{W}}(y, \mathsf{f}) \wedge \boxed{\mathsf{R}}(x, y)]]$
$\neg\exists x\neg\neg[\boxed{\mathsf{L}}(x) \wedge \neg\exists y[\boxed{\mathsf{W}}(y, \mathsf{f}) \wedge \boxed{\mathsf{R}}(x, y)]]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\exists y[\boxed{\mathsf{W}}(y, \mathsf{f}) \wedge \boxed{\mathsf{R}}(x, y)]]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\exists y[\boxed{\mathsf{W}\imath}(\mathsf{f}, y) \wedge \boxed{\mathsf{R}}(x, y)]]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\exists y[\boxed{\mathsf{W}\imath\mathsf{f}}(y) \wedge \boxed{\mathsf{R}}(x, y)]]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\exists y\boxed{\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes}(y, x, y)]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\exists y\boxed{\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath}(x, y, y)]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\exists y\boxed{\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}}(y, y, x)]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\exists y\boxed{\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}\Sigma}(y, x)]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \neg\boxed{\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}\Sigma\rfloor}(x)]$
$\neg\exists x[\boxed{\mathsf{L}}(x) \wedge \boxed{\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}\Sigma\rfloor\mathfrak{C}}(x)]$
$\neg\exists x\boxed{\mathsf{L}\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}\Sigma\rfloor\mathfrak{C}\boxtimes}(x, x)$
$\neg\exists x\boxed{\mathsf{L}\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}\Sigma\rfloor\mathfrak{C}\boxtimes\Sigma}(x)$
$\neg\boxed{\mathsf{L}\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}\Sigma\rfloor\mathfrak{C}\boxtimes\Sigma\rfloor}$
$\boxed{\mathsf{L}\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}\Sigma\rfloor\mathfrak{C}\boxtimes\Sigma\rfloor\mathfrak{C}}$

The constituent terms involved here may be interpreted as follows:

'$\mathsf{W}\imath\mathsf{f}$': the property of being written by Frege.

'$\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes$': the relation that holds of three things iff[9] the first is written by Frege and the second has read the third.

'$\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath$': the relation that holds of three things iff the second is written by Frege and the first has read the third.

'$\mathsf{W}\imath\mathsf{f}\mathsf{R}\boxtimes\imath\bar{\imath}$': the relation that holds of three things iff the first is written by Frege and the second has been read by the third.

[9] The conjunction "iff" should be read as shorthand for "if and only if".

'W$\imath$fR$\boxtimes\imath\bar{\imath}\Sigma$': the relation that holds of two things iff the first is written by Frege and has been read by the second.

'W$\imath$fR$\boxtimes\imath\bar{\imath}\Sigma\beth$': the property of having read something written by Frege.

'W$\imath$fR$\boxtimes\imath\bar{\imath}\Sigma\beth\complement$': the property of not having read something written by Frege.

'LW$\imath$fR$\boxtimes\imath\bar{\imath}\Sigma\beth\complement\boxtimes$': the relation that holds between a logician and anyone who has not read something written by Frege.

'LW$\imath$fR$\boxtimes\imath\bar{\imath}\Sigma\beth\complement\boxtimes\Sigma$': the property of being a logician and not having read something written by Frege.

'LW$\imath$fR$\boxtimes\imath\bar{\imath}\Sigma\beth\complement\boxtimes\Sigma\beth$': the proposition that there is a logician who has not read something written by Frege.

'LW$\imath$fR$\boxtimes\imath\bar{\imath}\Sigma\beth\complement\boxtimes\Sigma\beth\complement$': the proposition that there is no logician who has not read something written by Frege.

These examples show how a formula in standard predicate calculus can be transformed into the PFLIC notation. An algorithm spelling out the general procedure in detail could be provided, but will not be specified here.

4 Concluding remarks

PFLIC is a semantically more parsimonious language than ordinary predicate calculus. The variable-relative aspect of the semantics is eliminated. As usual, the price we have to pay for a step from a semantically richer language to a more parsimonious is a certain complication of expression. At least, PFLIC represents a "way of thinking" that probably feels quite awkward to people used to ordinary predicate calculus.

It is consequently obvious that the variable-based notation has some advantages. Variables give us a kind of graphical overview of logical structure. They indicate each "unfilled" argument position and how they are "unified" (by being filled with the same variable). It may be quite difficult, by contrast, to keep track of the effects of the permutational inversion operators ('$\imath$' and '$\bar{\imath}$') in PFLIC. It is consequently not a good idea to try to convince the logic community to abandon variables in favour of predicate functors. However, variables do complicate semantics, and in some contexts it may be a good idea to replace them by other devices (cf. Dahllöf [4]).

The main point of this paper has been to introduce the reader to predicate-functor logic and to convince him or her that it is a calculus of great theoretical interest. By seeing that and how predicate calculus can be given a "purer" but equally powerful semantics when variables are expelled, we elucidate what variables are good for. This is important, as the Fregean use of variables in the analysis of language has had a profound influence. Twentieth century linguistics and philosophy of language would certainly have looked quite different if some early modern logician had convinced the logic community that the predicate-functor-based way of dealing with quantification is the natural one.

References

[1] Allwood, J., Andersson, L.-G., and Dahl, Ö. (1977) *Logic in Linguistics.* Cambridge: Cambridge University Press.

[2] Bacon, J. (1985) The Completeness of Predicate-Functor Logic, *Journal of Symbolic Logic*, Vol. 50, 903–926.

[3] Barwise, J. and Cooper, R. (1981) Generalized Quantifiers and Natural Language, *Linguistics and Philosophy*, Vol. 4, 159–219.

[4] Dahllöf, M. (1995) *On the Semantics of Propositional Attitude Reports.* Gteborg University, Dept of Linguistics.

[5] Dowty, D. R., Wall, R. E. and Peters, S. (1981) *Introduction to Montague Semantics.* Dordrecht: Reidel.

[6] Grandy, R. E. (1976) Anadic Logic and English, *Synthese*, Vol. 23, 395–402.

[7] Grünberg, T. (1983) A Tableau System of Proof for Predicate-Functor Logic with Identity, *Journal of Symbolic Logic*, Vol. 48, 1140–1144.

[8] Kuhn, S. T. (1983) An Axiomatization of Predicate Functor Logic, *Notre Dame Journal of Formal Logic*, Vol. 24, 233–241.

[9] Noah, A. (1980) Predicate-Functors and the Limits of Decidability in Logic, *Notre Dame Journal of Formal Logic*, Vol. 21, 701-707.

[10] Purdy, W. C. (1991) A Logic for Natural Language, *Notre Dame Journal of Formal Logic*, Vol. 32, 409–425.

[11] Quine, W. V. (1960) Variables Explained Away, *Proceedings of American Philosophical Society*, Vol. 104, 343–347. Also in Quine, W. V. (1960) *Selected Logic Papers*, New York: Random House. 227–235.

[12] Quine, W. V. (1971) Predicate Functor Logic. In Fenstad, J. E. (ed.) *Proceedings of the Second Scandinavian Logic Symposium.* Amsterdam and London: North-Holland Publishing Company. 309–315.

[13] Quine, W. V. (1976) The Variable. In Quine, W. V. (1976) *The Ways of Paradox.* Revised and enlarged edition. Cambridge, Mass.: Harvard University Press. 272–282.

[14] Quine, W. V. (1976) Algebraic Logic and Predicate Functors. In Quine, W. V. (1976) *The Ways of Paradox.* Revised and enlarged edition. Cambridge, Mass.: Harvard University Press. 283–307.

[15] Quine, W. V. (1981) Predicates, Terms, and Classes. In Quine, W. V. (1981) *Theories and Things.* Cambridge, Mass. and London, England: The Belknap Press of Harvard University Press. 164–172.

[16] Quine, W. V. (1981) Predicate Functors Revisited, *Journal of Symbolic Logic*, Vol. 46, 649–652.