PREDICATE-FUNCTOR LOGIC WITH OPERATION SYMBOLS (1)

Teo GRÜNBERG

This paper is a sequel to the present author's *A tableau system of proof for predicate-functor logic with identity* (2), or [TPF] for short, and should be read together for ready reference. Given that predicate-functor logic aims at "algebrizing quantification... with all the clarity of the discrete and blocklike terms and simple substitutions characteristic of algebra" (3), it would be interesting to build also a *predicate-functor logic with operation symbols* which retrieves all the occurrences of free singular terms. The purpose of this paper is to construct such a system, by extending Quine's method of eliminating the bound variables to first-order predicate languages with operation symbols, and by applying then, our tableau method of proof.

1. Predicate-Functor Language with Operation Symbols: $\mathbf{L_P}$

Consider a first-order language L_{SX} whose vocabulary is the union of a set of extralogical predicate- and operation-symbols, the set S = {'∼', '•', '∃', '='} of standard logical constants, and the set $X = \{x_1, x_2, \ldots, x_i, \ldots\}$ of alphabetically ordered (individual) variables. We use 'φ', 'ψ' as metavariables for formulas of L_{SX}. The *predicate-functor language with operation symbols* corresponding to L_{SX} is the language L_P whose vocabulary is the union of the set of predicate- and operation-symbols of L_{SX} and the set P = {'∩', '–', 'p', '⊊', '⊒', 'I'} of (logical) predicate functors. We use as metavariables (for each $n \geqslant 0$) 'Π^n' for n-place predicate symbols, 'ω^n' for n-place operation

(1) I acknowledge my great gratitude to Quine who has inspired and encouraged my work on predicate-functor logic, and who has read (in August 1981) a first draft of this paper with helpful comments.

(2) **Journal of Symbolic Logic**, 48 (1983) 1140-1144.

(3) See W.V. QUINE, *Algebraic Logic and Predicate Functors*, in **The Way of Paradox and Other Essays** (Harvard Press paperback, enlarged edition, 1976), p. 284.

symbols, and 'π^n' for (possibly complex) predicates of degree n. A *predicate* π^n *of degree* n is defined as in [TPF] by adding only a clause stipulating that for any $m \geqslant 0$, $\omega^m \pi^n$ is a predicate of degree $m+n-1$ if $n > 0$, and of degree 0 if $n = 0$. Thus ω^m operates here on a predicate and is thus itself a 1-place (extralogical) predicate functor([4]). We see that the set of predicates of L_P includes those of the predicate-functor language L in [TPF] so that L_P is an extension of L. We define the *height* hg (π^n) of a predicate π^n of L_P as being equal to the total number of occurrences in π^n of the predicate functors '$\cap$', '$-$', 'p', '$\sqsubset$', '$\searrow$' and of operation symbols 'ω^n'.

A structure $\mathfrak{U}$ with universe $|\mathfrak{U}|$ for L_P is the same as a structure for L_{SX}. Instead of '$|\mathfrak{U}|$' we write 'U' when no confusion results. $\mathfrak{U}(\Pi^n) \subseteq U^n$ and $\mathfrak{U}(\omega^n)$ is a function from U^n to U.

$(\pi^n)^{\mathfrak{U}}$ is defined as in [TPF] by adding only the following clause:

$$(\omega^m \pi^n)^{\mathfrak{U}} = \begin{cases} \{<u_1,\ldots, u_{m+n-1}> : <\mathfrak{U}(\omega^m)(u_1,\ldots, u_m), u_{m+1,\ldots,} \\ u_{m+n-1}> \in (\pi^n)^{\mathfrak{U}}\}, \text{ if } n \geqslant 1; \\ (\pi^0)^{\mathfrak{U}}, \text{ if } n = 0 \end{cases}$$

A sequence $\mathbf{u} = (u_1, u_2,\ldots, u_i,\ldots)$ of elements of U *satisfies-in-*$\mathfrak{U}$ the predicate π^n, or $\mathbf{u} \models_{\mathfrak{U}} \pi^n$ for short, in case

$$<u_1,\ldots,u_n> \in (\pi^n)^{\mathfrak{U}}.$$

A predicate π^n is called *satisfiable* in case it is satisfied in some structure by some sequence, and it is called *universal* in case $-\pi^n$ is unsatisfiable.

2. Predicate-Functor Language with Singular Terms: L_{PX}

By adjoining to the vocabulary of L_P the set X of variables we obtain an extension L_{PX} called the *predicate-functor language with (operation symbols and) singular terms*. The predicates of L_{PX} are defined as those of L_P and the singular terms as those of L_{SX}. We use as metavariables 'τ', 'σ' for singular terms. A formula of L_{PX} is an expression of the form $\pi^n \tau_1\ldots\tau_n$, $n \geqslant 0$. In analogy to [TPF], the *height* hg $(\pi^n \tau_1\ldots\tau_n)$ of such a formula is defined as being equal to hg (π^n). We

([4]) We tacitly assume that L_P contains an unwritten operator which (in the context $\omega^m \pi^n$ operates on ω^m to form a 1-place predicate functor.

say also that a string of singular terms $\tau'_1 \ldots \tau'_n$ *is* (σ_1, σ_2) – *related to* string $\tau_1 \ldots \tau_n$ in case $\tau'_1 \ldots \tau'_n$ differs from $\tau_1 \ldots \tau_n$ only by replacing one of the singular terms σ_1, σ_2 by the other.

A structure $\mathfrak{U}$ (with universe U) for L_{PX} is the same as a structure for L_P (or for L_{SX}). For any sequence **u**, define $(x_i)^{\mathbf{u}} = u_i$, $(\omega^n \tau_1 \ldots \tau_n)^{\mathbf{u}} = \mathfrak{U}(\omega^n)(\tau_1{}^{\mathbf{u}}, \ldots, \tau_n{}^{\mathbf{u}})$, and

$$\mathbf{u} \models_{\mathfrak{U}} \pi^n \tau_1 \ldots \tau_n \text{ (u satisfies-in-}\mathfrak{U}\ \pi^n \tau_1 \ldots \tau_n) \text{ iff } \langle \tau_1{}^{\mathbf{u}}, \ldots, \tau_n{}^{\mathbf{u}} \rangle \in (\pi^n)^{\mathbf{u}}.$$

Satisfiability and validity of a formula are defined as usually.

Definition: A *reducible* formula of L_{PX} is one whose predicate begins with '--', 'p', 'ς', 'ω^m', or else, whose predicate has the form '$\supset \pi^0$'. The *reduced transform* $\widehat{\pi^n \tau_1 \ldots \tau_n}$ of a reducible formula is defined by the following conditions:

1. $\widehat{(\text{--}\pi^n)\tau_1 \ldots \tau_n} = \pi^n \tau_1 \ldots \tau_n$

2. $\widehat{p\pi^n \tau_1 \ldots \tau_n} = \begin{cases} \pi^n \tau_1 \tau_n \tau_2 \ldots \tau_{n-1}, & \text{if } n \geqslant 3; \\ \pi^n \tau_1 \ldots \tau_n, & \text{if } n = 0, 1, 2. \end{cases}$

3. $\widehat{(\varsigma\ \pi^n)\tau_1 \ldots \tau_{n+1}} = \pi^n \tau_2 \ldots \tau_{n+1}$.

4. $\widehat{\supset \pi^0} = \pi^0$.

5. $\widehat{(\omega^m \pi^n)\ \tau_1 \ldots \tau_{m+n-1}} = \pi^n \omega^m (\tau_1, \ldots, \tau_m)\ \tau_{m+1} \ldots \tau_{m+n-1}$, if $n \geqslant 1$.

6. $\widehat{\omega^m \pi^0} = \pi^0$.

The height of the reduced transform is below the height of the corresponding reducible formula.

Proposition: Any reducible formula of L_{PX} is equivalent to its reduced transform.

3. Elimination of Bound Variables

Consider the set of expressions consisting of the predicates of L_P and the formulas of L_{SX} and L_{PX}. Any two of such expressions are called *equivalent* in case they are satisfied in every structure by the same sequences. We write '$\models\!\mid$' between two equivalent expressions.

In case the equivalent expressions belong to different languages, we write within parantheses, to the right of the equivalence, the names of the respective languages.

Proposition: There is a (computable) function $\mathscr{F}$ from the set of formulas of L_{PX} to the set of formulas of L_{SX} such that for any $\pi^n\tau_1...\tau_n$:

$$\pi^n\tau_1...\tau_n \models\!\dashv \mathscr{F}(\pi^n\tau_1...\tau_n)\ (L_{PX}, L_{SX}).$$

Proof: Define $\mathscr{F}$ by the following conditions:

1. $\mathscr{F}(\Pi^n\tau_1...\tau_n) = \Pi^n\tau_1...\tau_n, \mathscr{F}(\mathrm{I}\tau_1\tau_2) = \ulcorner\tau_1 = \tau_2\urcorner$.
2. $\mathscr{F}(\ulcorner(\pi_1{}^n \cap \pi_2{}^n)\tau_1...\tau_n\urcorner) = \ulcorner\mathscr{F}(\pi_1{}^n\tau_1...\tau_n) \bullet \mathscr{F}(\pi_2{}^n\tau_1...\tau_n)\urcorner$.
3. $\mathscr{F}(\ulcorner(-\pi_1)^n\tau_1...\tau_n\urcorner) = \ulcorner\sim\mathscr{F}(\pi^n\tau_1...\tau_n)\urcorner$.
4. $\mathscr{F}(\ulcorner(\supset \pi^n)\tau_1...\tau_{n-1}\urcorner) = \ulcorner\exists x_i\mathscr{F}(\pi^n x_i\tau_1...\tau_{n-1})\urcorner$, if $n \geq 1$.

where i is the least positive integer such that x_i does not occur in the string $\tau_1...\tau_{n-1}$.

5. $\mathscr{F}(\pi^n\tau_1...\tau_n) = \widehat{\pi^n\tau_1...\tau_n}$, if $\pi^n\tau_1...\tau_n$ is a reducible formula.

Example: Using 'F', 'G' as predicate symbols, 'f', 'g' as operation symbols, 'a', 'b' as 0-place operation symbols and 'x', 'y' as variables we obtain:

$$\mathscr{F}(\text{`}(f^1F^4 \cap g^1G^4)abxy\text{'}) = \text{`}F^4f^1(a)bxy \bullet G^4g^1(a)bxy\text{'}.$$

Proposition: For any string of singular terms $\tau_1...\tau_n$ and a variable x_i, there is a (complex) predicate functor $\mathscr{R}_{\tau_1...\tau_n}$ (the reductor of string $\tau_1...\tau_n$), and a complex predicate functor $\mathscr{R}_{\tau_1...\tau_n/x_i}$ (the *reductor* of string $\tau_1...\tau_n$ *with respect to* variable x_i), such that

$$\pi^n\tau_1...\tau_n \models\!\dashv (\mathscr{R}_{\tau_1...\tau_n}\pi^n)x_{i_1}...x_{i_r} \quad (r \geq 0),$$

where $x_{i_1},...,x_{i_r}$ – in this order – are all the occurrences of variables in string $\tau_1...\tau_n$; and

$$\pi^n\tau_1...\tau_n \models\!\dashv (\mathscr{R}_{\tau_1...\tau_n/x_i}\pi^n)x_i\,\tau'_1...\tau'_\ell \quad (\ell \geq 0),$$

where $\tau'_1,...,\tau'_\ell$ – in this order – are all the occurrences of maximal x_i-free subterms of $\tau_1,...,\tau_n$. (An x_i-free subterm of τ is a subterm of τ which does not contain x_i.)

Examples: [5]

$$`F^3xg^2(y,a)z' \;\vDash\!\dashv\; `(aq_2g^2p_1F^3)xyz' = \ulcorner(\mathscr{R}_{xg^2(y,a)z}F^3)xyz\urcorner.$$
$$`F^3xg^2(y,a)z' \;\vDash\!\dashv\; `(q_2p_1g^2p_1F^3)yxaz' = \ulcorner(\mathscr{R}_{xg^2(y,a)z/y}F^3)yxaz\urcorner$$

Theorem: There is a (computable) function $\mathscr{G}$ from the set of formulas of L_{SX} to the set of formulas of L_{PX}, such that for any formula φ, there is a predicate π^n which satisfies the condition

$$\varphi \;\vDash\!\dashv\; \pi^n\tau_1\ldots\tau_n,$$

where $\tau_1,\ldots,\tau_n$ – in this order – are all the occurrences of maximal free singular terms in φ. [6]

Proof: Define $\mathscr{G}$ by the following conditions:

1. $\mathscr{G}(\Pi^n\tau_1\ldots\tau_n) = \Pi^n\tau_1\ldots\tau_n$, $\mathscr{G}(\ulcorner\tau_1 = \tau_2\urcorner) = \mathrm{I}\tau_1,\tau_2$.
2. $\mathscr{G}(\ulcorner\sim\psi\urcorner) = \ulcorner(-\pi^n)\tau_1\ldots\tau_n\urcorner$, if $\mathscr{G}(\psi) = \pi^n\tau_1\ldots\tau_n$.
3. $\mathscr{G}(\ulcorner\psi_1\bullet\psi_2\urcorner) = \ulcorner(\pi_1{}^m\times\pi_2{}^n)\tau_1\ldots\tau_m\,\tau'_1\ldots\tau'_n\urcorner$, [7] if $\mathscr{G}(\psi_1) = \pi_1{}^m\tau_1\ldots\tau_m$ and $\mathscr{G}(\psi_2) = \pi_2{}^n\,\tau'_1\ldots\tau'_n$.
4. $\mathscr{G}(\ulcorner\exists x_i\psi\urcorner) = \ulcorner(\trianglerighteq\mathscr{R}_{\tau_1\ldots\tau_n/x_i}\pi^n)\tau'_1\ldots\tau'_\ell\urcorner$, if $\mathscr{G}(\psi) = \pi^n\tau_1\ldots\tau_n$, and $\tau'_1,\ldots,\tau'_\ell$ – in this order – are all the occurrences of maximal x_i-free subterms of $\tau_1,\ldots,\tau_n$.

Example: $\mathscr{G}$('$\sim$(Fg(a) $\bullet\sim\exists$xFg(x))') = '–(F$\cap$ ⊊ – ⊵ gF)g(a)'.

4. Tableau System of Proof for Predicate-Functor Logic with Singular Terms

We shall show now that our tableau system of proof in [TPF] can be extended to *predicate-functor logic with singular terms*, i.e. the logic

[5] In these examples we use the predicate functors 'p_i' defined in Quine, *op. cit.* p. 300. We use also 'q_i' as short for 'p_i^i' (*i.e.*, '$p_ip_i\ldots p_i$' to i occurrences). 'p_i' corresponds to Kuhn's 'ϱ_{i+1}^{-1}' and 'q_i' to 'ϱ_{i+1}'. See S.T. Kuhn, *Quantifiers as Modal Operators*, **Studia Logica** Vol. XXXIX (1980) 2/3 p. 150.

[6] It follows from the existence of the functions $\mathscr{F}$ and $\mathscr{G}$ that the languages L_{PX} and L_{SX} are equivalent in Kuhn's sense. See Kuhn, *op. cit.* p. 152.

[7] '×' (Cartesian multiplication) is defined in Quine, *op. cit.*, p. 300 in terms of ' ' and the heterogeneous intersection functor. But the latter is definable in terms of the homogeneous '$\cap$' as pointed out by T.S. Kuhn. See [TPF], n.1.

underlying the language L_{PX}. On the basis of the analogy between the prefixed predicates (of the form $i_1 \ldots i_n \pi^n$) of the auxiliary language L^* in [TPF] and the formulas (of the form $\pi^n \tau_1 \ldots \tau_n$) of L_{PX}, we classify the formulas of L_{PX} also into component-free ones and into α-, β-, γ-, δ-, $(\alpha\beta)$-types. The tabulation of types and components for the formulas L_{PX} results from the tabulation for L^*, by merely substituting 'τ_1',...,'τ_n', 'τ' (and moving them to the left of the predicate) for 'i_1',...,'i_n', 'i' respectively, and also, by adding to the tabulation the following rows:

Type	Form	Form
$(\alpha\beta)$:	$(\pm\omega^m\pi^n)\tau_1\ldots\tau_{m+n-1}$	$\pm\omega^m\pi^0$
$(\alpha\beta)_1$: component	$\pm\pi^n\omega^m(\tau_1,\ldots,\tau_m)\tau_{m+1}\ldots\tau_{m+n-1}$	$\pm\pi^0$
	$n \geqslant 1$	

In the resulting tabulation, the height of any formula of type $\alpha, \beta, \gamma, \delta$, or $(\alpha\beta)$ is always above the height of each of its components.

On the basis of this tabulation, we define a (cut-free, or analytic) *tableau for a formula* $\pi^n\tau_1\ldots\tau_n$ of L_{PX} as a tree with origin $\pi^n\tau_1\ldots\tau_n$, constructed by means of tableau rules for the α-, β-, γ-, δ-, $(\alpha\beta)$-types and a tableau rule for identity(8). The tableau rule for δ-type is the same as the correspondent rule in [TPF], while the other tableau rules result from the correspondent rules by merely substituting metavariables ranging over singular terms for those which are ranging over position markers. A closed tableau for a formula $\pi^n\tau_1\ldots\tau_n$ is defined in the same way as one for a prefixed predicate. A closed tableau for formula $(-\pi^n)\tau_1\ldots\tau_n$ is called a *tableau proof* for $\pi^n\tau_1\ldots\tau_n$, and $\pi^n\tau_1\ldots\tau_n$ is called then *tableau provable*.

(8) The tableau rule for identity has the following form:

If a branch ϑ contains both a node $\pm\Pi^n\tau_1\ldots\tau_n$ and a node $I\sigma_1\sigma_2$, then we may adjoin to ϑ a node $\pm\Pi^n\tau_1'\ldots\tau_n'$, where $\tau_1'\ldots\tau_n'$ is (σ_1, σ_2)-related to $\tau_1\ldots\tau_n$.

This rule is a natural generalization of the corresponding rule in [TPF] which had been suggested to me by the anonymous referee of the Journal of Symbolic Logic. I acknowledge again my gratitude to him.

Example: Let us construct a tableau proof for the valid formula

-(F∩ ς - ⊐ gF)g(a):

1.	--(F∩ ς - ⊐ gF)g(a)	from:
2.	(F∩ ς - ⊐ gF)g(a)	(1)
3.	Fg(a)	(2)
4.	(ς - ⊐ gF)g(a)	(2)
5.	- ⊐ gF	(4)
6.	(-gF)a	(5)
7.	(-F)g(a)	(6)
	(closed by 3, 7)	

Theorem (soundness): Every tableau-provable formula of L_{PX} is valid. The proof is similar to that of the corresponding Theorem 1 in [TPF] and is based on the analogues of Lemmata 1, 2.

Theorem (completeness): Every valid formula of L_{PX} is tableau provable.

The proof is similar to the corresponding Theorem 2 in [TPF] and is based on the analogues of Lemmata 3, 4. In particular a *Hintikka set* of formulas of L_{PX} is defined in analogy to a Hintikka set of prefixed predicates, but the proof of the Lemma that every Hintikka set of formulas of L_{PX} is satisfiable, differs somewhat from the proof of the corresponding Lemma 3 in [TPF]. Therefore we must outline here the proof of that Lemma.

Let then T be the set of singular terms of L_{PX}. Given a Hintikka set Γ, define a relation E on T, as the smallest subset of T^2, such that for every $\tau, \tau_1, \ldots, \tau_n, \tau'_1, \ldots, \tau'_n$:

1. if $I\tau_1\tau_2 \in \Gamma$ then $\tau_1 E \tau_2$,
2. $\tau E \tau$,
3. if $\tau_1 E \tau'_1, \ldots, \tau_n E \tau'_n$ then $\omega^n(\tau_1, \ldots, \tau_n) \; E \; \omega^n(\tau'_1, \ldots, \tau'_n)$.

E exists and is an equivalence relation on T which satisfies the following condition:

If $\pm\Pi^n\tau_1\ldots\tau_n \in \Gamma$, $\sigma_1 E \sigma_2$, and $\tau'_1\ldots\tau'_n$ is (δ_1, δ_2)- related to $\tau_1\ldots\tau_n$, then $\pm\Pi^n\tau'_1\ldots\tau'_n \in \Gamma$.

Define then a structure $\mathfrak{U}$ with universe U for L_{PX} and a sequence $\mathbf{u}$ such that:

1. $U = T/U$ (i.e., the set of equivalence classes $[\tau] = \{\sigma : \sigma E \tau\}$, $\tau \in T$);
2. $\mathfrak{U}(\Pi^n) = \{<[\tau_1],\ldots,[\tau_n]> : \Pi^n \tau_1 \ldots \tau_n \in \Gamma\}$;
3. $\mathfrak{U}(\omega^n) = \{<[\tau_1],\ldots,[\tau_n],[\tau_{n+1}]> : [\omega^n \tau_1 \ldots \tau_n] = [\tau_{n+1}]\}$;
4. $\mathbf{u} = ([x_1],[x_2],\ldots,[x_i],\ldots)$.

We show then:

i. $<[\tau_1],\ldots,[\tau_n]> \in \mathfrak{U}(\pm\Pi^n)$ iff $\pm\Pi^n \tau_1 \ldots \tau_n \in \Gamma$.
ii. $\mathfrak{U}(\omega^n)([\tau_1],\ldots,[\tau_n]) = [\omega^n \tau_1 \ldots \tau_n]$.
iii. $\tau^{\mathbf{u}} = [\tau]$.
iv. $\mathbf{u} \models_{\mathfrak{U}} \Gamma$, i.e., for every formula $\pi^n \tau_1 \ldots \tau_n$ of L_{PX}:
If $\pi^n \tau_1 \ldots \tau_n \in \Gamma$ then $\mathbf{u} \models_{\mathfrak{U}} \pi^n \tau_1 \ldots \tau_n$.

Finally in order to prove the analogue of Lemma 4 in [TPF], we consider an effective enumeration of the set T of singular terms of L_{PX}.

5. Elimination of Free Variables and Tableau System of Proof for Predicate-Functor Logic without Singular Terms

Proposition: There is a (computable) function $\mathscr{F}$ from the set of predicates of L_P to the set of formulas of L_{SX}, such that for every π^n:

$$\pi^n \models\!\dashv \mathscr{F}(\pi^n) \quad (L_P, L_{SX}).$$

Proof: Define $\mathscr{F}(\pi^n) = \pi^n{}_{x_1 \ldots x_n}$.

Proposition: For any finite sequence of positive integers $i_1,\ldots,i_n$, $n \geqslant 0$, there is a (complex) predicate functor $\Theta_{<i_1,\ldots,i_n>}$, (9) such that for every π^n, $\Theta_{<i_1,\ldots,i_n>}\pi^n$ is a predicate of degree $m = max(i_1,\ldots,i_n,n)$, and

$$\Theta_{<i_1,\ldots,i_n>}\pi^n \models\!\dashv \pi^n x_{i_1} \ldots x_{i_n} \quad (L_P, L_{PX}).$$

(9) The predicate functors $\Theta_{<i_1,\ldots,i_n>}$ are defined in Kuhn, *op. cit.*, p. 151.

Theorem: There is a (computable) function $\mathscr{G}$ from the set of formulas of L_{SX} to the set of predicates of L_P, such that for every formula φ:

$$\varphi \models\!\dashv \mathscr{G}(\varphi) \quad (L_{SX}, L_P).$$ [10]

Proof: Define for any string $\tau_1 \ldots \tau_n$ the (complex) predicate functor $\mathscr{C}_{\tau_1 \ldots \tau_n}$, such that for every π^n:

$$\mathscr{C}_{\tau_1 \ldots \tau_n}\pi^n = \Theta_{<i_1, \ldots, i_r>} \mathscr{R}_{\tau_1 \ldots \tau_n} \pi^n, \; r \geqslant 0,$$

where $x_{i_1}, \ldots, x_{i_r}$ – in this order – are all the occurrences of variables in $\tau_1 \ldots \tau_n$. We call $\mathscr{C}_{\tau_1 \ldots \tau_n}$ a $\mathscr{C}$*-predicate*, and we can show that:

(1) $\mathscr{C}_{\tau_1 \ldots \tau_n}\pi^n \models\!\dashv \pi^n \tau_1 \ldots \tau_n \quad (L_P, L_{PX})$.

Define then $\mathscr{G}(\varphi) = \mathscr{C}_{\tau_1 \ldots \tau_n}\pi^n$, in case $\mathscr{G}(\varphi) = \pi^n{}_{\tau_1 \ldots \tau_n}$. This concludes the proof.

We see that this last Theorem provides for the elimination of free variables and, in general, of free singular terms. But, interestingly enough, the eliminated free singular terms reappear as indices of the metalinguistic expression '$\mathscr{C}_{\tau_1 \ldots \tau_n}\pi^n$', and these indices are blocklike terms subject to simple substitutions. Exploiting this characteristic, we can obtain an intrinsic, sound and complete proof procedure for the *predicate-functor logic without singular terms* underlying the language L_P. Indeed, on the basis of equivalence (1) which establishes a 1-1 correspondence between the set of all formulas of L_{PX} and the set of $\mathscr{C}$-predicates of L_P, we can transpose to L_P the whole tableau system of proof in L_{PX} (with its tabulation of formulas, tableau rules, and tableau constructions), merely by translating any formula $\pi^n \tau_1 \ldots \tau_n$ into the corresponding equivalent $\mathscr{C}$-predicate $\mathscr{C}_{\tau_1 \ldots \tau_n}\pi^n$. Thus a *tableau for* $\mathscr{C}_{\tau_1 \ldots \tau_n}\pi^n$ is the translation of a tableau for $\pi^n \tau_1 \ldots \tau_n$. A tableau for a predicate π^n, which is not itself a $\mathscr{C}$-predicate, is defined as a tableau for the equivalent $\mathscr{C}$-predicate $\mathscr{C}_{x_1 \ldots x_n}\pi^n$.

[10] The existence of functions $\mathscr{F}$, $\mathscr{G}$ shows that the languages L_P and L_{SX} are equivalent in Kuhn's sense.

Let us conclude with the remark that L_P, when devoid of operation symbols, reduces to the predicate-functor language L in [TPF]. Then, every$\mathscr{C}$-predicate has the form$\mathscr{C}_{x_{i_1}\ldots x_{i_n}}\pi^n$, and this form reduces further to $\Theta_{<i_1,\ldots,i_n>}\pi^n$. We can then construe a prefixed predicate $i_1\ldots i_n\pi^n$ of the auxiliary language L^* in [TPF], as a shorthand for the predicate $\Theta_{<i_1,\ldots,i_n>}\pi^n$ of L. Alternatively we can construe $i_1\ldots i_n\pi^n$ as standing for the formula $\pi^n x_{i_1}\ldots x_{i_n}$ of the language L_{PX} wich corresponds to L.

Orta Doğu Teknik Üniversitesi Teo GRÜNBERG
(Middle East Technical University)
Dept. of Philosophy
Ankara, Turquie

Abstract

We first extend Quine's method of eliminating the bound variables to first-order predicate logic with operation symbols; then we give a device for eliminating also the free variables; and finally we extend our tableau method of proof to the resulting predicate-functor logic with operation symbols, with or without singular terms.