Semantic presuppositions in logical syntax

Yaroslav Kokhan

Institute of Philosophy
National Academy of Sciences of Ukraine
Triokhsviatytelska str. 4, 01001, Kyiv, Ukraine
yarkaen@gmail.com

ABSTRACT. *There are two implicit semantic postulates underlying modern predicate logic. Hence predicate logic is not semantically neutral. The author proposes to take semantically neutral languages, which have no predicate categorial structure but replace the notion of predicate with general notion of function. Some function calculi for different semantics are demonstrated.*

KEYWORDS: *the standard semantics, non-standard semantics, predicate logic, function logic, choice function.*

DOI:10.3166/JANCL.22.41–56

The point of view on logical syntax shared by many logicians consists in that logical syntax could be entirely separated from semantics so that it is possible to build and consider any of syntactical objects — formulas, inferences, calculi, theories — without any referring to semantics. To refute the standpoint described is the aim of the article.

1. Presuppositions and the standard semantics

As a matter of fact, falsity of the view described is obvious. Indeed, all contemporary formal languages contain individual designators — constants and variables — *i.e*. designators for particular individuals. These symbols are introduced immediately as linguistic expressions, which have one and only one semantical value-denotatum. Thus, introducing them in syntax referring to semantics is already presupposed. At present moment the author has revealed two semantic postulates implicitly accepted in modern logical syntax. Here they are:

(I) *All proper names are single-valued,* i.e. *each of them could have only one semantical value-interpretant (no more and no less);*

(II) *Interpretants of linguistic expressions are denotata of these expressions, not their senses.*

Postulate (II) could be formulated another way: interpretation relation or relation between expression **t** and its interpretant τ is semantic relation of denoting (**t** denotes an object τ), but not relation of expressing (**t** expresses a sense τ).

Important consequences follow from these two postulates. Particularly, from (I) follows

(III) *Equality is meaningful only for single-valued names.*

Indeed, contraposition to (I) says that there are no non-single valued names of objects. But where equality asserts that two names of objects have the same denotation, this exactly means that non-single valued expressions cannot appear in equality expression.

The consequence of (II) is no less principal:

(IV) *Existence is a quantifier, not a predicate.*

Indeed, as Frege shows in "Dialogue with Pünjer", to be and to exist for denotata is the same. So when we reject a universal proposition "all x have property $F(x)$", thereby we say that there is an individual without this property or, what is the same, that such an individual exists *i.e*. we assert that there is (= exists) x without property $F(x)$.

Semantics based on the postulates (I) and (II), will be called **the standard semantics** below. Thereby, rejection of one or both of these postulates entails different **non-standard semantics**.

2. Restrictions of the standard semantics

Although the standard semantics is implicitly accepted in all contemporary logic, its application area is much more narrow and does not cover even classical mathematical logic. It is evident at least concerning its first postulate. Indeed, this postulate denies existence of non-single valued names for objects, meanwhile such names do exist and regularly appear in logic as functional terms and descriptions. For example, if $\boldsymbol{a}$ is a proper name and $f(x)$ is a partial function expression, then $f(\boldsymbol{a})$ may well be an empty name; from the other hand, all η-terms and ε-terms are non-single valued by definition. Implicit acceptance of the postulate (I) makes operation with such expressions tangibly more difficult. So, for functional terms, if partial functions are admitted, we are forced to introduce in addition to "ordinary" equality so called conditional equality for not more than single-valued names (this does not cover many-valued names), or to introduce in metalanguage an additional condition, which equate all empty names, what is not always acceptable.

Let us consider the next example. Suppose given two models of the same signature that differ only in one aspect: domain of the first model consists of residents of some city in the year k, when domain of another model consists of residents of that very city in the year n with respect to the same calendar, and $n > k$. Suppose then the signature contains the names not only for alive people, but also for dead, and individuals $\boldsymbol{a}$ and $\boldsymbol{b}$ belong to the first model, but in the second model both names '$\boldsymbol{a}$' and '$\boldsymbol{b}$' being empty, since individuals $\boldsymbol{a}$ and $\boldsymbol{b}$ have died in the year m, where $k < m < n$. In this case, in the first model we'll have true sentence '$\boldsymbol{a} \neq \boldsymbol{b}$' and false sentence '$\boldsymbol{a} = \boldsymbol{b}$'. But trying to define truth values for both sentences in the second model we could come across substantial difficulties. Indeed, one and only one of them should be true, as it follows from the notions of equality and negation. But how a sentence, which includes an empty name, could be true? Evidently, this question goes beyond the limits of the standard semantics, that is why there is no satisfying solution till this moment. Mostly it is stated the sentences that include empty names are false or meaningless. Both of these answers, standard for logicians, are clearly inadequate to our model example: if two individuals die, any statement about them not become meaningless by this reason, particularly, the statements whether they were different people or the same person stay meaningful. Therefore, the sentences considered cannot be meaningless. But they both also cannot be false in the second model, because they contradict each other. So, we have to accept such semantics that would allow us to determine which of these sentences is true. There are at least two ways to resolve the empty names problem suggested in logic. The well-known method by Gottlob Frege, according to which, all empty names in a given formal language are given common denotation, being an arbitrary individual. This method clearly fails with respect to our example: if $\boldsymbol{a}$ and $\boldsymbol{b}$ were different people during their life, they cannot become the same person after their death; so one should not assign one value to all empty names. Another method, proposed by R. Martin, seems not to be widely discussed. It consists in introducing for every model so called "empty individual" singled out among any individuals like null among numbers; the empty individual should be assigned to empty names as

their denotatum. However, this method does not work with our example for the same reasons as in the previous case (different people after their death cannot become equal and became some "null person").

The obvious fallacy of both the methods does not depend on whether there is equality in our object language — it is enough for it to have a negation. Indeed, it is always possible to build model like in our example, in the way it will have predicates determinating designated individuals $\boldsymbol{a}$ and $\boldsymbol{b}$. Suppose predicates $F(x)$ and $G(x, y)$ belong to the signature of both models, so that these sentences: $F(\boldsymbol{a})$, $\neg F(\boldsymbol{b})$, $G(\boldsymbol{a}, \boldsymbol{b})$ and $\neg G(\boldsymbol{b}, \boldsymbol{a})$ are true in the first model. In this case their truth values should not change in the second model as well as truth values of negations of all these sentences. For example, $F(x)$ can mean "to have the daughter called Ada", and $G(x, y)$ — "x is older then y"; it is clear, that these facts does not depend on that, whether they are stated about dead or alive people, so $G(\boldsymbol{a}, \boldsymbol{b})$ should be true and its negation $\neg G(\boldsymbol{a}, \boldsymbol{b})$ should be false, regardless to whether names '$\boldsymbol{a}$' and '$\boldsymbol{b}$' have values-interpretants.

As for the descriptions as a kind of names the situation is even worse, because there is no general method of operating with them. R. Carnap in "Meaning and necessity" describes five (though says about three) methods to interpret definite descriptions in case when they prove to be empty. A common feature of all these well-known methods is that they do not follow from any general theory, but are *ad hoc* propositions. Carnap himself frankly recognizes the last fact, stating that no one of these methods is a true or a false theory but they are just more or less convenient instruments. Let us remind all of them.

Frege by his first method (he proposed two one) assigns the special class of individuals to every non-single valued description as its denotatum, namely, the class of individuals such that the predicate being the matrix of description holds for these individuals. Thus, it is proposed to consider the class of all kings of France in 1905 (an empty class in this case) as the value of the description "the king of France in 1905". Formally, this method is invulnerable, but in case of empirical descriptions it is out of synch with intuition. For example, when at the funeral feast one says that "the departed man (or woman) was a merciful person", it means one and only one individual (which does not exist now, so term "the departed" is an empty description) but not the class of all departed men over the Earth to the moment of that funeral feast.

Another Frege's method was described above for the general case of proper names: to assign the same denotatum (it does not matter, what) to all empty descriptions. This method we have already discussed.

Matrin's method, which could be taken as a modification of the second Frege's method, also has been discussed above.

Russell's method provides that any formula that includes a definite description also implicitly includes a statement about the single-valuedness of that description, *i.e.* it looks like a conjunction of the formula itself and such statement. This method has two

disadvantages: firstly, descriptions cease to be a kind of proper names in the sense that the laws of quantification, particularly the axiom

$$\forall x F(x) \to F(a), \tag{1}$$

do not hold for them (compare with section 4); secondly, negation of every formula with description can be interpreted in two different ways, where in general case it is impossible to choose the right one.

Gilbert-Bernaise's method allows only those descriptions for which their single-valuedness is proven. But in this case the formation rules lose their definiteness: we do not know whether any sentence with description is well formed until we prove the single-valuedness of that description.

As we can see, all known methods of operating with empty names do not work in the sphere of empirical descriptions. Similarly there are no plausible method for many-valued names. There is a wide class of ambiguous names, not expressed in formal languages neither by functional terms nor by descriptive terms in their modern form, but they are widespread in natural languages; these are proper names of people, geographical and political placenames. Really, there has been more than one person on this planet, named "Karl Marx", there are more than one inhabited locality named "Odessa", at least two rivers named "Angara" etc. If such names were single-valued, it could be possible to introduce for every one of them the description $\iota_x(x = \boldsymbol{a})$, where $\boldsymbol{a}$ is a required name.

But what to do with many-valued (and also with empty) names? Formally, it is possible to hold, that "Karl Marx" and "Angara" are names of properties and by this reason consider the description has a form $\varepsilon_x F(x)$, but it is counterintuitive. No sane person would agree that "Karl Marx" is not a name of a person but a name of some property of this person. It should be namely the name of an individual. But such names are forbidden in the standard semantics, and, as we have seen above, all the tries to overcome the bounds of this semantics and to create a theory for ambiguous names have not been successful.

3. Non-standard semantics according to the first postulate

As we ascertained above on our model example, the standard semantics sometimes is inadequate and should be replaced by some other semantics. So it is important to conceive, how many alternatives for the standard semantics there are and what they are. Such alternatives are formed by rejection of one or both postulates of the standard semantics.

Firstly, let us reject the first postulate. In this case we have no limitation on the number of semantic values for proper names, so all the names should be classified somehow. It is natural to classify names in three groups: empty (with no interpretants), single-valued (with only one interpretant) and many-valued (with more than one interpretant). It is easy to classify all the semantics according to this triple di-

vision, in dependence on what kinds of noted types of names they allow. Generally, there are seven different types of semantics:

Table 1. Types of semantics

	1	2	3	4	5	6	7
Empty names	+	+	+	+	–	–	–
Single-valued names	+	+	–	–	+	+	–
Many-valued names	+	–	+	–	+	–	+

The case 6 here is the class of semantics that are standard according to the first postulate (the very standard semantics is among them). The classes represented are unequal: thus, semantics of the class 4 are obviously very specific (they are useful perhaps only for depicting the myths, tales and literary characters), when semantics of the class 1 seems to be the most interesting, as they and only they provide the fully valid operation with descriptions.

Two fundamental questions arise here: what formal languages and what calculi meet the requirements of different classes of semantics from the Table 1? We have already seen that modern logic has no satisfactory theory to describe semantics of all the classes excepting the class 6. Now we are going to show that this is not a historical contingency but a necessary consequence of construction of modern logical systems.

Evidently, from the postulate (I) of the standard semantics it follows that

(V) *Proper names are non-empty;*

This consequence can be expressed by the formula:

$$F(a) \to \exists x F(x). \tag{2}$$

Indeed, it is enough to take a predicate "to have the name $\boldsymbol{c}$" as $F(x)$. But formula (2) easily follows from (1), which is an axiom in all the calculi with quantifiers (for unknown reasons, textbook authors often ignore this consequence taking (2) as an additional axiom, although the required proof was given by Frege in Begriffsschrift). This means that to these days the laws of logic were chosen in a way that proper names remain non-empty, which is a necessary condition for holding the postulate (I). And this means that modern systems of logic are not able to describe semantics of the classes 1–4 from the Table 1). As for description of many-valued names (semantics of the classes 1, 3, 5, 7), in modern formal languages there are only indefinite descriptions, but operating with them meets the non-emptiness problem as it was shown above. Hence, we need logical systems without axiom (1) to describe non-standard semantics. But how it can be substituted if that axiom describes the properties of quantifiers? The author does not know formulation that can be substituted for the axiom (1) without violating the postulate (II) of the standard semantics. It looks like such formulation does not exist at all. But if so, there exist some "forbidden zones" in modern logic within the limits of which it could be described nothing. This is a rather unpleasant concequence, since it reveals non-universality of modern logical

description tools. The reason of this non-universality is the semantical loading: contemporary logical languages and calculi are not **semantically neutral** because they are constructed holding over the semantical postulates (I) and (II) and as a result they cannot describe all semantics.

As a matter of fact, there is a universal method to develop logical languages and calculi that are able to describe semantics of any class. However this method stipulates the changes of the categorical apparatus of logic and building formal languages on principles quite different from the commonly used.

4. Function languages

Categorical system of contemporary logic, on the basis of which formal languages are formulated, is grounded on division of all the non-logical objects of investigation into *objects* (*individuals*) and *predicates*. Respectively, in formal languages there appear notations — as basic — for these two types of objects: individual expressions *i.e*. terms are divided in two groups — individual constants (proper names, individual symbols) and individual variables; and predicate expressions are divided into predicate constants and predicate variables. Sometimes the notion of term needs to be extended if expressions for functions are introduced into language: then functional expressions together with terms constitute a *quasiterm* class. Note that function is more general category than predicate: the predicates are only a kind of functions, namely a kind of truth-value or propositional functions (logical operations also belong to both). Nevertheless, function is not a basic logical category, unlike predicate, and the most of contemporary formal languages can be called **the predicate languages**.

The bounds of this article do not allow to explain completely the connections between described categories and semantics, so we represent only the conclusion (which is clarified below): in order to constitute semantically neutral languages, it is necessary to refuse from predicate as a basic logical category. Predicates should be rejected and replaced with functions. It has not been done until this moment, because the modern notion of function is too narrow with respect to expressive power of formal languages. For every n-ary function $f^{(n)}(x_1, \ldots, x_n)$ there is the $n+1$-ary predicate $F^{(n+1)}(x_0, x_1, \ldots, x_n)$ such that for any $a_0, a_1, \ldots, a_n$ from any model an equation

$$a_0 = f^{(n)}(a_1, \ldots, a_n) \tag{3}$$

implies a proposition

$$F^{(n+1)}(a_0, a_1, \ldots, a_n), \tag{4}$$

but not for every $n+1$-ary predicate there is the n-ary function such that (4) implies (3). The reason of this is quite simple: for any given set $a_1, \ldots, a_n$ of arguments of function f there can be only one value a_0 of this function, meanwhile an arbitrary predicate F can be not single-valued for any of its arguments. The situation that appears can be called *the predicate-function paradox*: whether predicates are just a kind of functions, the expressive abilities of predicate atoms appear more rich than expressive tools of propositional atoms, constituted by functions.

But we could extend the notion of function so that function expressive abilities exceed expressive abilities of predicates and the predicate-function paradox would disappear. Functions, in the modern sense of the term, are maps. To eliminate the predicate-function paradox, they should be reinterpreted as partial multimaps, i.e. functions that (i) can have arbitrary number of values, including different number of values for different sets of arguments, and (ii) can have no arguments at all. Since these functions are ambiguous as functions, the relation between function value and the function itself with given arguments (if there are any) cannot be an equality. However, the required relation is somewhat a generalization of equality relation, because it differs from the last only by that it associates a univocaly given object, which is the function value, with, generally speaking, equivocaly given object, which is the function with given (fixed) arguments (if there are any for this function), because it assigns its every value the same way. On this ground, let us mark the required relation with the sign of approximate equality '$\approx$', because it has no established terminological use in modern logic. This relation $\approx$ we'll call the *representation* proceeding from the fact that an ambiguous function value is not generated but only represented by it.

After the generalization mentioned of the notion of function we immediately notice, that now for any $n+1$-ary predicate $F^{(n+1)}(x_0, x_1, \ldots, x_n)$ there can be found the n-ary function $f^{(n)}(x_1, \ldots, x_n)$ such that for any $a_0, a_1, \ldots, a_n$ from any given model formula (4) implies

$$a_0 \approx f^{(n)}(a_1, \ldots, a_n). \tag{5}$$

Particularly, in any model for every 1-ary predicate $F^{(1)}(x)$ there is the 0-ary function $f^{(0)}$ such that for any a a predicate atom $F^{(1)}(a)$ implies a function atom $a \approx f^{(0)}$. And this means that using functions in this new extended sense makes possible to express everything whatever is possible to express with predicates. Combining formulas of type (5) with propositional and quantification technique, we receive systems that are highly competitive with familiar *predicate systems*. But now they are *function systems*.

Introduction the representation relation as a basic category of logic among category of individual and generalized category of function (predicates we have rejected) changes radically the form of propositional atoms of formal languages: this time there will be formulas not of kind (4), but formulas

$$s \approx t, \tag{6}$$

which will be called *representation formulas* and read from left to right as "s is a value (one of the values, if there are any at all) of a function t" or "s is a t-individual (an individual of kind of t)". We call the formal languages with atoms of form (6) **the function languages** (to distinguish them from modern predicate languages). In formation rules for the function languages it should be noted that an atomic formula is a line '$s \approx t$', when s is a term and t is a quasiterm.

Equality is a partial case of representation (formula (6) expresses equality when t is a term), so it is defined in function languages — unlike predicate languages. But we are forced to define equality and inequality separately. Be

$$s = t \underset{Df}{\equiv} s \approx t \wedge (r \approx t \to (u \approx t \to r \approx u)), \tag{7}$$

$$s \neq t \underset{Df}{\equiv} \neg(s \approx t) \wedge (r \approx t \to (u \approx t \to r \approx u)). \tag{8}$$

Function languages and calculi unlike predicate ones are semantically neutral. So it is possible to develop calculi in function languages for any semantics. Let us start with the standard semantics. We will use first order predicate calculus with equality $\mathrm{P}^{=}\mathrm{F}^{1}$ as exemplar (symbolism is explained in the next section); we assume the $\mathrm{P}^{=}\mathrm{F}^{1}$ contains propositional calculus P, has two lists of variables (free and bound variables differ), axioms (not schemata) and rules of substitution. Now we can build calculus, which is an exact clone of $\mathrm{P}^{=}\mathrm{F}^{1}$ and has the standard semantics, so we will denote it by $\mathrm{I_S F}^{1}$. Alphabets of the both calculi differ only by predicate and function letters (we use F, G, H and f, g, h respectively) and sign for representation. Formation rules are analogous in both calculi except the definition of atomic formulas: in $\mathrm{P}^{=}\mathrm{F}^{1}$ atomic formulas have a form $u = w$ or $P_i(u_1, \ldots, u_n)$, where $P \in \{F, G, H\}$ and $u, u_1, \ldots, u_n, w$ are free individual variables, meantime in $\mathrm{I_S F}^{1}$ atomic formulas have a form $u \approx w$ or $u \approx p_i(u_1, \ldots, u_n)$, where $p \in \{f, g, h\}$; formulas with equality are defined in $\mathrm{I_S F}^{1}$ by (7), (8). Axioms of both mentioned calculi are in Table 2.

Table 2. Logical axioms in the standard semantics

	$\mathrm{P}^{=}\mathrm{F}^{1}$		$\mathrm{I_S F}^{1}$
(a_{P})	$\forall x\, F(x) \to F(a)$	(a_{I})	$\forall x\, (x \approx f) \to a \approx f$
(b_{P})	$a = a$	(b_{I})	$a \approx a$
(c_{P})	$a = b \to (F(a) \to F(b))$	(c_{I})	$a \approx b \to (a \approx f \to b \approx f)$

Formulas (a_{P}), (a_{I}) are axioms of quantifiers, (b_{P}) and (c_{P}) are axioms of equality, (b_{I}) and (c_{I}) are axioms of representation. The existential quantifier is defined in the both calculi.

Non-standard semantics involve non-single valued terms, so to build calculi for them, we should introduce into syntax the class of special logical *choice functions*. If we use generalized functions, it is necessary to have a possibility to denote particular values of an ambiguous function via expression for this function. As any function with fixed arguments (or without arguments) does not determinate its values, an additional logical function is required in order to choose one of the values of an ambiguous function and represent it. In other words we have to introduce notations for choice functions into logical syntax; in this case a concrete value (if any exists) of any function f, which is selected by a choice function, is a semantical value (interpretant) of the term formed by application of a choice function expression to the expression for the function f as argument. More precisely, given language L, we treat any choice

function as a partial map ch: t $\times$ I $\rightarrow$ M, where t $\subset$ L is the set of quasiterms of the language L, I is a set of indices, and M is the domain of the principal interpretation of L. Practically we need only denumerable set of indices, so we can take words $'$, $''$, $\ldots, {}^{<n>}, \ldots$ of L as indices (L must contain symbol '$'$') and use naturals to sign them. We denote value of choice function in L by ch(t, i) or by ch^i(t) or even by t^i, where $i \in$ I. If t is $f^{(n)}(x_1, \ldots, x_n)$, let t^i be '$f^{i(n)}(x_1, \ldots, x_n)$'. If function f is a partial map or a map, let us denote it's single value by 'f^0'. We say that any quasiterm of form t^i is a *marked quasiterm*. It is clear, that for every proper name $\boldsymbol{a}$ and every i is true, that the marked term $\boldsymbol{a}^i$ is equal to unmarked term $\boldsymbol{a}$ and $(\boldsymbol{a}^i)^j$ is equal to $\boldsymbol{a}^i$.

Choice functions as well as interpretation maps in the standard model theory are semantic interpretation functions. Thus every context in a fixed language L, for example a calculus or a theory, can be related with only one choice function. Logical laws don't depend on interpretation, so any calculus with language L can be related with an arbitrary choice function that can be used with L; hence we don't need to define choice function, which we use, anyway.

Introduction of the choice functions allows us to describe various objects with the same name within one context. For example, sentence "That dog attacked another dog" can be formalized as 'dog$^1 \approx$ attack (dog^2)'; additionally we can assert in this case that $\neg$(dog$^1 \approx$ dog^2). Using languages with symbolism for a choice function we even do not need to introduce individual designations, because marked quasiterms of a form f^i can play their role, where 'f' is a name for a 0-ary function. In particular, marked quasiterms can be quantified as it happens in natural language, so we can formalize sentences of human speech literally in the simplest cases. For example, sentence "all stars are shining" can be formalized as '$\forall s^2(s^2 \approx sh)$' where '$s$' signs the 0-ary function "star", 'sh' signs the 0-ary function "shining". Thus we have to determinate free and bound marked quasiterms or free and bound occurrences of the marked quasiterms; for this purpose we accept next convention: free occurrences of the marked quasiterms in formal expressions are tagged only by odd indices of the choice functions but bound ones only by even.

Let us build as example the calculus I^{d1}F^1 for semantics of the class 1 with the postulate (II). This is the most interesting and important semantics because it involves names with any number of values. So, the corresponding calculus I^{d1}F^1 can be treated as the most important function calculus.

Alphabet A(I^{d1}F^1) = $\{A, B, C, f, g, h, \neg, \rightarrow, \approx, =, \forall, \exists, ', (,)\}$. Here '$A$', '$B$', '$C$' are *propositional letters*, 'f', 'g', 'h' are *function letters*; other signs treated as usual.

Metaalphabet includes signs 's', 't', 'r' for quasiterms and signs '$\mathfrak{F}$', '$\mathfrak{G}$', '$\mathfrak{H}$' for quasiformulas.

Language L(I^{d1}F^1) is the set of all *words* (*i.e.* lines of letters) in the alphabet A(I^{d1}F^1) including empty word Λ.

Variables: *propositional variables* are the words of a form $U^{<m>}$, where U is a propositional letter and $^{<m>}$ is a word in the alphabet $\{'\}$; function variables are the words of a form $p^{<n>}p^{<m>}$, where p is a function letter and $^{<n>}$, $^{<m>}$ are words in the alphabet $\{'\}$ (n means arity of a function variable). We sign the words in the alphabet $\{'\}$ by naturals and write U_m and $p_m^{(n)}$ instead of $U^{<m>}$ and $p^{<n>}p^{<m>}$; also we can omit an index of arity and write p_m.

Quasiterms: (qt_0) the words of a form $p^{\Lambda}p^{<m>}$ or $p^{<n>}p^{<m>}t_1,\dots,t_n$, where $^{<n>}$ is non-empty, are *unmarked quasiterms*; (qt_1) if s is an unmarked quasiterm containing function letter p, then $p^{<i>}s$ is a *marked quasiterm*, where $^{<i>}$ is a non-empty word in the alphabet $\{'\}$ (i means an index of a choice function); (qt_2) the unmarked and the marked quasiterms are quasiterms. As above we write $p_m^{i(0)}$ or p_m^i and $p_m^{i(n)}(t_1,\dots,t_n)$ or $p_m^i(t_1,\dots,t_n)$ instead of $p^{<i>}p^{\Lambda}p^{<m>}$ and $p^{<i>}p^{<n>}p^{<m>}t_1,\dots,t_n$; also for a quasiterm with index i of a choice function we can sign it as s^i or t^i or r^i.

Terms are quasiterms containing only odd indices of a choice function.

Atomic quasiformulas: (qf_0) propositional variables are atomic quasiformulas; (qf_1) if s is a marked quasiterm and t is a quasiterm, then $s \approx t$ is an atomic quasiformula.

Quasiformulas: (f_0) the atomic quasiformulas are quasiformulas; (f_1) if $\mathfrak{F}$ is a quasiformula, then $\neg\mathfrak{F}$ is a quasiformula; (f_2) if $\mathfrak{F},\mathfrak{G}$ are quasiformulas, then $(\mathfrak{F}\to\mathfrak{G})$ is a quasiformula; (f_3) if $\mathfrak{F}(s^i)$ is a quasiformula containing a term s^i, then $\forall s^{i+1}\mathfrak{F}(s^{i+1})$ is a quasiformula. As in predicate logic we say that in the last quasiformula first occurrence of a quasiterm s^{i+1} is an *occurrence in a quantifier* but second occurrence of s^{i+1} is an *occurrence in the range of a quantifier*; such occurrences of quasiterms are *bound*, all another occurrences are *free*.

Formulas are quasiformulas that contain (occurencies of) any marked quasiterm that is not a term only in some quantifier or in the range of that quantifier (in other words, contain no free occurrences of marked quasiterms that are not terms).

Axioms:

(a^{d1}) $\forall f^2(f^2\approx g)\to f^1\approx g$,

(b^{d1}) $f^1\approx f^1$,

(c^{d1}) $f^1\approx g^1\to(f^1\approx h\to g^1\approx h)$,

(d^{d1}) $f^1\approx g^1\to f^1\approx g$,

(e^{d1}) $\exists g^2(f^1\approx g^2)\to f^1\approx g$,

(f^{d1}) $\exists h^2(h^2\approx f^1)\to(f^1\approx g\to\exists f^2(f^2\approx g))$,

(g^{d1}) $\forall f^2(f^2\approx g)\to\neg\exists f^2\neg(f^2\approx g)$.

In addition, axioms of propositional logic and definitions (7), (8) are introduced. Here (a^{d1}) is the axiom of universal quantifier, (b^{d1}) – (d^{d1}) are axioms of representation,

(e^{d1}), (f^{d1}) are axioms of existential quantifier and axiom (g^{d1}) relates both quantifiers.

Rules of transformation includes one *rule of inference* — modus ponens, two *rules of quantification*:

(α) $\quad \mathfrak{G} \to \mathfrak{F}(f^i) \vdash \mathfrak{G} \to \forall f^{i+1}\mathfrak{F}(f^{i+1})$,

(β) $\quad \exists h^j(h^j \approx f^i) \to (\mathfrak{F}(f^i) \to \mathfrak{G}) \vdash \exists f^{i+1}\mathfrak{F}(f^{i+1}) \to \mathfrak{G}$,

where j is even but i is odd, and three *rules of replacement*:

Propositional substitution $\mathrm{S}^{\mathfrak{G}}_{A}\mathfrak{F}(A) : \mathfrak{F}(A) \vdash \mathfrak{F}(\mathfrak{G})$,

Replacement of an index of a choice function $\mathrm{R}^{j}_{i}\mathfrak{F}(s^i) : \mathfrak{F}(s^i) \vdash \mathfrak{F}(s^j)$, where i, j both are odd or both are even, and

Quasiterm replacement $\mathrm{S}^{t}_{s}\mathfrak{F}(s) : \mathfrak{F}(s) \vdash \mathfrak{F}(t)$, where t is a term if s is a term or an unmarked quasiterm, and t is a marked quasiterm if s is a marked quasiterm.

If we want to add individual designations (*i.e*. single-valued terms) to this calculus, we should add in it individual's letters a, b, c, x, y, z, extent the notions of atomic quasiformula and formula, then add the next two axioms:

(g_{I}) $\quad \neg\forall x(x \approx g) \to \exists x\neg(x \approx g)$,

(h_{I}) $\quad \exists x(x \approx a)$,

and add the rules of substitution and replacement of the individual variables; the author denotes this new calculus by $\mathrm{I}^{d1}_{\mathrm{I}}\mathrm{F}^1$.

Different semantics cause different sets of logical laws. In particular, in semantics of classes 1–4, which involve empty names, quantifiers are not dual to each other, so formula

$$\neg\forall f^2(f^2 \approx g) \leftrightarrow \exists f^2\neg(f^2 \approx g)$$

here is not a logical law (indeed, the set of f-individuals can be empty for some f). Hence quantifiers are not defined one by another, so they must be introduced separately and described with the different axioms (see above axioms of types (*a*) and (*f*)). Additionally we set an axiom of type (*g*) describing relation between both quantifiers; if calculus contains individual notations, we need one more axiom (g_{I}), which together with the axiom of type (*g*) guarantees the duality of quantifiers under individual variables. Because of absence of the duality in semantics of classes 1–4 the rule (β) does not have a form

$$\mathfrak{F}(f^1) \to \mathfrak{G} \vdash \exists f^2\mathfrak{F}(f^2) \to \mathfrak{G},$$

which it have in predicate logic, but get an additional existential antecedent (see above).

In semantics of classes 5–7, where there's no empty names, situation is more much regular: here quantifiers are mutually dual as well as they are dual in predicate calculi.

So, they defined one by another and we don't need an axiom of types (f) and (g) but obtain the theorem

$$f^1 \approx g \to \exists f^2(f^2 \approx g),$$

which is completely analogous to the theorem (2) from predicate logic.

Let us make two important remarks on the functional syntax. The first of these concerns predicates. There is no separate category of predicate in function languages, but the predicates are present here as propositional functions. Just, every n-ary function f, whose first arguments are g_1-individuals, ..., n-th ones are g_n-individuals, and the values are g_0-individuals, defined by expression

$$f^{(n)}(g_1^2, \ldots, g_n^2); \tag{9}$$

analogous expression

$$g_0^2 \approx f^{(n)}(g_1^2, \ldots, g_n^2) \tag{10}$$

define the $n+1$-ary predicate as be a propositional form in the sense of A. Church, hense denotes the propositional function. The case of 0-ary function f is the same. If $g_0, \ldots, g_n$ are the same function g, then (9) and (10) become $f^{(n)}(g^2, \ldots, g^{2n})$ and $g^2 \approx f^{(n)}(g^4, \ldots, g^{2(n+1)})$ respectively.

The second remark concerns descriptions. We have already some of them: these will be function expressions of the form $f^{i(n)}(g_1^1, \ldots, g_n^{2n-1})$ as $n \geqslant 0$. But we can introduce even descriptions of the general form into function languages; these are indefinite descriptions analogous to ε-descriptions from predicate languages. Not discussing details, the result is given. Description from a point of view of the function logical system is an unary function with propositional function as the only argument. In other words description "f-individual such that satisfy constraint $\mathfrak{F}$" will be written as

$$f^1(\mathfrak{F}(f^1)),$$

in particular, "f-individual such that have property g" be written as '$f^1(f^1 \approx g)$'. As far as one can judge, if one extends a function language with descriptive quasiterms, it needs no additional axiom, because descriptions behave themselves as ordinary general names. Thus any problems with treatment of descriptions shouldn't emerge.

5. Non-standard semantics according to the second postulate

If the refusal of the first postulate of the standard semantics generates a whole class of alternatives, then the second postulate has only one alternative:

(VI) *Interpretants of linguistic expressions are their senses, but not their denotata.*

Both of these alternatives are not equal: the postulate (VI) is poorer for the consequences than postulate (I), since linguistic expressions as well as senses may play role of denotata (so far as the category of denotatum is relative), meanwhile the category of sense is absolute. The most natural technique for empirical descriptions, in particular for descriptions that contain empty names, is based on the postulate (VI). This is

conditioned by the point that if the postulate (VI) is accepted, then existence does not become a quantifier but transforms into a predicate speaking about particular individuals. If we denote the latter by '$x \approx E$', the proposition "Julius Caesar exists" could be written down as '$\boldsymbol{c} \approx E$', where $\boldsymbol{c}$ is "Julius Caesar", but "not every town exists" as '$\neg\forall \boldsymbol{t}^2(\boldsymbol{t}^2 \approx E)$', where $\boldsymbol{t}$ is function-property "town". At the same time existential quantifier will change from existential state into *partial*: being connected with him will mean existence no more.

Let us call semantics containing the postulate (II) *denotational semantics* or *d-semantics*, and semantics containing the postulate (VI) — *sense semantics* or *s-semantics*. So, let us use superscripts 'd' and 's' in notation of calculi and theirs axioms for the sake of indicating that fact what kind of semantics lays in the basis of them (index 'd' has used in previous section just in this sense). If calculus contains individual notations, let us use subscript 'I' in it's symbolism. Predicate calculi the author denotes by letters 'PF' (what meaning: "propositional function"), and function calculi — by letters 'IF' (so far as it goes about the functions of individuals); superscript after 'F' means logic's order in the sense of theory of types.

Calculi with sense semantics from classes 1–3 and 5–7, containing notations for the predicate of existence, have the following additional axiom:

$(ex^{s13})\ f^1 \approx E \to \exists h^2(h^2 \approx f^1)$,

from which, in particular, the next theorem can be derived:

$$f^1 \approx E \to \exists f^2(f^2 \approx E);$$

(note that in s-semantics of the classes 1–3 formulas '$\exists f^2(f^2 \approx f^1)$' and '$\exists f^2(f^2 \approx E)$' are mutually independent, in s-semantics of the class 4 they both are identically false, and in s-semantics of the classes 5–7 the second formula is logically stronger than the first one). If we remove the predicate of existence and an axiom of type (*ex*) from any calculus under discussion, we obtain *the system of logic without existential presuppositions*.

6. Notes on model theory

Rejection of the standard semantics leads to changes in model theory. First of all, we want signature maps σ : E $\to$ M be completely defined (here E is the set of all well-formed expressions of a given language L, and M is the domain of a given model $\mathfrak{M}$). If we reject postulate (V), then E may contain empty names, so we need to introduce a fictional analougue of interpretant for every such name. Let us call such nonentities *semantic nulls*; by $\mathbb{N}$ denote set of semantic nulls, by $\mathbb{A}$ denote set of "ordinary" interpretants; then be M $= \mathbb{A}\cup\mathbb{N}$, $\mathbb{A}\cap\mathbb{N} = \varnothing$. Unlike the unique Martin's empty individual semantic nulls can be any number. Indeed, putting semantic nulls is necessary if we want to describe cases such as discussed above in section 2, where $\boldsymbol{a}$ and $\boldsymbol{b}$ on the second model are fictions, although $\boldsymbol{a} \neq \boldsymbol{b}$. It should be emphasized that the idea of semantic nulls is not a hypothesis *ad hoc*, but occurs naturally within the theory of semantic triangle. It will not be discussed here this time.

Further, a model $\mathfrak{M}$ for every calculus C has the single domain M only if C contains individual notations as, for example, calculi $\mathrm{I}_{\mathrm{I}}^{d1}\mathrm{F}^{1}$ and $\mathrm{I}_{\mathrm{S}}\mathrm{F}^{1}$ do. In this case individual variables set a total class of individuals, *i.e.* universe; its presence is significant because all the functional terms behave as variables of different sorts (a sort determined by a function constructing a term), so every term has in model the corresponding set of individuals. If calculus C doesn't contain individual notations, every model $\mathfrak{M}$ of C has not single domain but the set of domains. As we know, in predicate logic many-sorted calculi are trivially reducible to single-sorted ones; in function logic if calculus doesn't contain individual notations, such reducibility is impossible. Thus function calculi without individual notations (for example, $\mathrm{I}^{d1}\mathrm{F}^{1}$) are *calculi with irreducible sorts*. Models for such calculi are not relational systems but have more general form

$$\mathfrak{M} =< \{\mathrm{M}_i\}, \{\mathrm{R}_j\} >_{i\in I, j\in J},$$

where I, J are arbitrary sets of indices. Relations R_j are interpretations for function expressions 'f', *i.e.* if a signature map has a form $\sigma : \mathrm{E} \to \mathrm{M}_i$, predicate (10) corresponds to function f, and $\sigma(g_0) = \mathrm{M}_0, \ldots, \sigma(g_n) = \mathrm{M}_n$, then $\sigma(f(g_1^2, \ldots, g_n^2)) = \mathrm{R} \subseteq \mathrm{M}_1 \times \ldots \times \mathrm{M}_n \times \mathrm{M}_0$; $\sigma(f^i) \in \mathrm{M}_0$ for all i.

In s-semantics, if we introduce the predicate of existence, we have to take pairs $< \mathrm{M}_i,\ \mathrm{M}_i^{\circ} >$ instead of single sets M_i, where M_i are sets of senses (and, possible, semantic nulls-pseudo senses corresponding to them), M_i° are sets of denotata (and, possible, semantic nulls-nonentities corresponding to them; in this case $\mathrm{M}_i^{\circ} = \mathbb{A}_i^{\circ} \cup \mathbb{N}_i^{\circ}$, $\mathbb{A}_i^{\circ} \cap \mathbb{N}_i^{\circ} = \varnothing$). For the sake of interpreting existence formulas we need to introduce partial maps $\mathfrak{s}_i : \mathrm{M}_i \to \mathrm{M}_i^{\circ}$; if we assume the postulate (V), then every $\mathfrak{s}_i$ is completely defined on M_i; if we reject (V), then every $\mathfrak{s}_i$ is completely defined on $\mathbb{A}_i$ and completely undefined on $\mathbb{N}_i$. Models of calculi with semantics under discussion have a form of the following mathematical structures:

$$\mathfrak{M} =< \{\mathrm{M}_i\}, \{\mathrm{M}_i^{\circ}\}, \{\mathrm{R}_j\} >_{i\in I, j\in J},$$

where R defined as above.

7. Summary

There are two implicit semantic postulates (I) i (II), which form the standard semantics, as it was called in this paper; they underlie modern predicate logic. Contexts that are not satisfactorily described in the standard semantics occur in logical applications from time to time. A striking example of semantically non-standard objects are descriptions, for which we still have no satisfactory theory, as all theorists have tried to build it based just on the (implicitly assumed) standard semantics. However, the possibility to build the predicate syntax based on non-standard semantics is not currently confirmed. The author is inclined to believe that the standard semantics implicitly underlies the categorical distinction between individual and predicate, so predicate languages as such are not semantically neutral.

At the same time there is a semantically neutral alternative for predicate languages — function languages. These new languages are always built on the basis

of explicitly specified semantics, and a formal function language could be constructed for each of semantics. Metatheory of such languages — syntax as well as model theory — is generalization of metatheory of predicate languages in the case of arbitrary semantics.