# DO WE NEED FUNCTIONAL ABSTRACTION?

Achille C. Varzi
IRST – Istituto per la Ricerca Scientifica e Tecnologica
I-38100 Trento, Italy

[Published in Johannes Czermak (ed.), *Philosophy of Mathematics. Proceedings of the 15th International Wittgenstein Symposium*, Part 1, Vienna: Hölder-Pichler-Tempsky, 1993, pp. 407–415.]

## 1. Introduction

There is a rather dominant view according to which functional application alone is too poor a paradigm for the analysis of complex linguistic structures: we also need functional abstraction. For instance, variable-binding operators such as quantifiers, integrals, etc. seem to run afoul of the basic functor/argument scheme, though they can be accounted for easily with the help of some abstraction device [1]. More generally, there is some evidence that (i) pure categorial grammars, based exclusively on the operation of functional application, are essentially equivalent to context-free phrase-structure grammars, hence subject to the same severe limitations [2]; and (ii) there is a strong connection between the principles of functional abstraction – most notably lambda-conversion – and those transformation-like rules that seem so necessary to bring out the relations between different levels of linguistic analysis, e.g. between deep logical structure and surface realizations [3]. As a result, lambda-equipped languages are frequently presented as an extension of pure categorial languages — a necessary extension obtained by adding a new primitive operation to increase combinatorial force and descriptive power.

In the following I argue against that view by presenting a way of interpreting functional abstraction within a pure categorial framework. Roughly, the suggestion is that variable binders can be treated as intensional operators (they make the value of an expression under an interpretation depend on its value under other interpretations), and intensional operators admit a natural treatment within the functor/argument scheme. Part of this is meant as a technical point; but my concern is also with some more general philosophical issues. Particularly, I shall consider a strong form of semanticism that seems to emerge from the proposed account: the view that logic is essentially a theory in the model-theoretic sense, i.e. a set of sentences and inference patterns that come out valid as a result of selecting a certain class of models as the only admissible ones (relative to a given language).

## 2. Functional application

My notion of a categorial framework may not be entirely standard, so I shall first outline it briefly.

The basic idea is that the expressions of a language may all be classified into two types[4]: *individual* (or *primitive*) and *functional* (or *derived*). The individual types correspond to those categories of expressions whose syntactic status is not analyzed in terms of other categories: declarative sentences, proper nouns, and presumably not many others. By contrast, functional types are defined in terms of simpler types in a way that fixes the combinatorial properties of the corresponding categories: for each pair of types $t$ and $t'$, primitive or functional, a new derived type $t'/t$ can be formed, corresponding to the category of those functors that combine with expressions of type $t'$ to produce expressions of type $t$. Thus, for instance, if *0* is the type of sentences and *1* the type of nouns, then *0/0* and *1/0* will be the types of monadic connectives and predicates respectively, (*0/0*)/(*0/0*) and (*1/0*)/(*1/0*) the types of the relevant adverbial modifiers, and so on[5].

Using $T$ to denote the set of all types, i.e. the closure of a suitable set of individual types under the slash operation /, the idea is then that the expressions of a language can be recursively specified on the basis of some type assignment to its symbols: for each $t \in T$, the corresponding category of expressions comprises all $t$-typed symbols (if any) plus all those expressions that can be obtained by applying a given structural operation to expressions of type $t'/t$ and $t'$ for some suitable $t' \in T$.

A *language* is a triple $L=(s,g,E)$ satisfying the following conditions:

(a) $s$ is a sequence of symbols, each associated with an ordinal index $\xi$ and a type index $t$;

(b) $g$ is a binary structural operation well-grounded on $s$;

(c) $E$ is the $T$-termed system of smallest categories that, for all $\xi,t \in Ds$ and all $t,t' \in T$, (i) $s_{\xi,t} \in E_t$, and (ii) if $x \in E_{t'/t}$ and $y \in E_{t'}$, then $g(x,y) \in E_t$[6].

It is understood that both $s$ and $g$ need be one-one to avoid ambiguities: combined with the requirement that $g$ be well-grounded on $s$ (i.e. $Rg \cap Rs=\emptyset$), that will secure that each expression be uniquely defined as either a symbol or a compound of the form $g(x,y)$. No further proviso is needed, though one may of course single out a particular class of languages by implementing clauses (a)-(c) as desired. For example, one may require that all functional expressions cancel to individual expressions (in the sense that $E_{t'/t} \neq \emptyset$ always implies that $E_{t'} \neq \emptyset$, hence $E_t \neq \emptyset$), and that $g$ be the operation of concatenation (so that $g(x,y)$ is always a string $xy$). A language satisfying these additional stipulations may be called *standard*.

Turning now to the notion of a model, the general idea is that a model must act as a sort of semantic lexicon: it must determine what kind of things may be assigned to

the basic components of the given language as their semantic counterparts, and it must do so within the limits set by the relevant type distinctions. The definition of a model therefore follows essentially the same format as that of a language:

A *model* for a language $L$ =(*s*,*g*,*E*) is a triple $M$ =(*d*,*h*, *I*) satisfying the following conditions:

(a) *d* is a sequence of denotations, one for each symbol of *s*;
(b) *h* is a binary operation;
(c) *I* is a $T$-termed system of domains so that, for all $\xi,t \in Ds$ and all $t,t' \in T$, (i) $d_{\xi,t} \in I_t$, and (ii) if $x \in I_{t'/t}$ and $y \in I_{t'}$, then $h(x,y) \in I_t$.

Also in this case, the intuitive appeal of a particular model depends on the actual make-up of the basic components *d*, *h*, *I*, but a major feature of the framework is precisely that one can in principle remain neutral with regard to such matters. Just as the notions of sentence, noun, predicate, etc. need not be defined except in relation to some type assignment, likewise their semantic counterparts can be given a purely formal, indirect definition. One may then single out the desired class of models simply by implementing clauses (a)-(c) with the appropriate additional conditions. For instance, a *standard* (Fregean) model would be one in which every functorial domain $I_{t'/t}$ is a set of functions $f: I_{t'} \rightarrow I_t$, and where *h* is the corresponding operation of application (in the sense that $h(x,y)$ always yields $x(y)$).

Finally, the bridge between languages and models is provided by a straightforward notion of a valuation. This is what makes the framework particularly appealing, apart from any considerations of esthetic elegance and conceptual parsimony. For it suffices to note that every language is homomorphic to any of its models. That is, if $L$ =(*s*,*g*,*E*) is a language and $M$ =(*d*,*h*,*I*) a corresponding model, then there exists a unique map $f: \bigcup RE \rightarrow \bigcup RI$ so that (i) $f(s_{\xi,t})=d_{\xi,t}$ for each $\xi,t \in Ds$, and (ii) $f(g(y,z)=h(f(y),f(z))$ for each $y,z \in Dg$. This means that $M$ is sure to provide all the information that is needed in order to evaluate every expression of $L$: *d* assigns a denotation to each basic expression, and *h* says how to compute the value of a compound expression given the values of its component parts [7].

The *valuation* of a language $L$ on a corresponding model $M$ is the unique homomorphism $V$ between $L$ and $M$.

## 3. Quantifiers as intensional functors

The general framework outlined above (not a semantic theory, but a framework wherein each of a variety of semantic theories can be obtained by varying the particular form of the language and the specific constraints on the relevant models) is based essentially on the single principle of functional application:

> The value of the result of applying a functor *y* to an argument *z* is the result of applying the value of *y* to the value of *z*.

In particular, relative to standard models of standard languages, this principle reduces to the simple equation $V(yz)=V(y)(V(z))$.

How do we fit variable binding into this simple scheme? Consider first how the scheme works in an easy case, say a propositional language with sentence symbols and connectives. Let $t^n/t'$ abbreviate a type $t_0/(t_1/(\ldots/(t_{n-1}/t')\ldots))$, where $t_i=t$ for each $i<n$ (for $n=0$, $t^n/t'$ reduces to $t'$). Using this notation, a propositional language can be defined quite simply as a (standard) language $L=(s,g,E)$ whose symbols are associated with types of the form $0^n/0$ for various $n \geq 0$. Correspondingly, a classical propositional model would just be any standard model $M=(d,h,I)$ such that $I_0=2=\{0,1\}$ (*0* and *1* for truth and falsity). In particular, if $L$ includes the usual connectives for negation ~ (of type *0/0*), conjunction (of type $0^2/0$), etc., then $M$ would have to satisfy the additional requirement that these symbols denote the appropriate Boolean operations on *2*:

(~) if $s_{\xi,t}=\sim$, then $d_{\xi,t}(x)=1-x$ for all $x \in 2$

( ) if $s_{\xi,t}=$ , then $d_{\xi,t}(x)(y)=x \cap y$ for all $x,y \in 2$

Likewise for the other connectives. The resulting valuation function would, relative to such models, yield the usual results.

Now this can be generalized in several ways. For instance, one can by the same pattern provide a semantic account conforming to a variety of non-classical propositional logics: all that matters is that the desired domains of interpretation and the intended meaning of each connective be specified accordingly, by setting the relevant constraints on the admissible models. The general format need not change. More interestingly, one can in a similar way account for the semantics of so called "intensional" languages, say the language of a propositional modal calculus. The semantic analysis of such a language is sometimes viewed as inducing a drastic departure from that of purely extensional languages, for the meaning of a modal connective is taken to depend on factors that cannot possibly be captured by an algebraic operation on truth-values. In a categorial framework the treatment is perfectly uniform: to account for the relevant factors one only has to refer to the appropriate class of models, requiring e.g. that the basic domains of interpretation be not just sets of flat, unanalysed entities (truth-values), but sets of functions ranging over those entities and taking as arguments items from an appropriate set of intensional features. Thus, if $L$ is a propositional language with modalities, a suitable model could be any (standard) model $M=(d,h,I)$ where $I_0=2^W$ for some non-empty set $W$ of "possible worlds". The interpretation of ~, , and the other extensional connectives is not disturbed by this shift from truth-values to truth-valued functions. But the shift becomes relevant as we turn to the modal connec-

tives, say the necessity connective □. For the intensional character of such a connective can be accounted for precisely by making its value for a given argument at a given possible world depend on the value of the argument at different possible worlds (e.g. by making it true at that world iff it is true at every world).

(~) if $s_{\xi,t} = \sim$, then $d_{\xi,t}(x)(w) = 1 - x(w)$ for all $x \in 2^W$ and $w \in W$
( ) if $s_{\xi,t} =$ , then $d_{\xi,t}(x)(y)(w) = x(w) \cap y(w)$ for all $x,y \in 2^W$ and $w \in W$
(□) if $s_{\xi,t} = \Box$, then $d_{\xi,t}(x)(w) = \mathrm{I}\{x(w') : w' \in W\}$ for all $x \in 2^W$ and $w \in W$

So much is fairly well known. The examples bear witness to the insightfulness of a categorial framework, which lies in the unity of the syntactic and semantic analyses it provides. What I want to stress is that this very insightfulness has far reaching consequences also with respect to the more complex cases I mentioned at the beginning.

Consider for instance an elementary language with quantifiers. Syntactically, there is no difficulty in squeezing quantifiers into the functor/argument scheme. We can e.g. treat them as functors of type $1/(0/0)$, i.e. as "mixed" functors taking nouns and sentences into sentences; or we can treat them as a kind of "structured" connectives of type $0/0$, consisting of a quantifier-marker together with a corresponding bound variable (symbols are atomic relative to the structural operation $g$, but need not necessarily lack internal structure). Take this second alternative. Then our elementary language is simply a (standard) language $L = (s,g,E)$ with symbols of type $t^n/t'$ for various $n \geq 0$ and $t,t' \leq 1$ — i.e. sentence and noun symbols (of type $0$ and $1$ respectively), connectives (of type $0^n/0$), predicates (of type $1^n/0$), and so on. Where Q is any quantifier-marker, e.g. the usual sign for universal quantification , we may then assume that the noun symbols of $L$ include a denumerable subset $V$ so that the string Q$v$ is a monadic connective for each $v \in V$, to be thought of as the Q-quantifier binding $v$.

Now look at this semantically. It is obvious that quantifiers pose a problem if we try to interpret them as operations on truth values, like ordinary truth-functional connectives. The point is, we need not do that. Truth-values are the extensions of sentences, if we like; but quantifiers introduce an intensional element — they make the value of a sentence depend on factors other than just the truth-values of its component parts. And we just saw that this type of dependence can easily be captured within a categorial framework. In the case of a modal connective like □, the intensional shift is from truth-values to truth-valued functions defined on possible worlds. In the case of quantifiers, the shift is due to a different combination of factors, viz. the various values that can be assigned to the corresponding bound variables. We may accordingly define a model for a language with quantifiers simply by requiring individual expressions to be interpreted as functions of such value-assignments [8]. More precisely, where $U$ is any non-empty set, we obtain a standard elementary model $M = (d, h, I)$ by setting $I_0 = 2^{U^V}$ and $I_1 = U^{U^V}$. The appropriate interpretation conditions for classical logic are the following:

(c) if $s_{\xi,t}$ $V$, then $d_{\xi,t}$ is constant, i.e. $d_{\xi,t}(a)=d_{\xi,t}(b)$ for all $a,b$ $U^V$
(v) if $s_{\xi,t}$ $V$ then $d_{\xi,t}$ is $\xi$-variable, i.e. $d_{\xi,t}(a)=a(s_{\xi,t})$ for all $a$ $U^V$
(~) if $s_{\xi,t}=$ ~, then $d_{\xi,t}(x)(a)=1-x(a)$ for all $x$ $I_0$ and $a$ $U^V$
( ) if $s_{\xi,t}=$ , then $d_{\xi,t}(x)(y)(a)= x(a)\cap y(a)$ for all $x,y$ $I_0$ and $a$ $U^V$
( ) if $s_{\xi,t}=$ $v$, then $d_{\xi,t}(x)(a)=\mathrm{I}\{x(a^v_u): u \quad U\}$ for all $x$ $I_0$ and $a$ $U^V$

where in the last clause $a^v_u$ is the function that is exactly like $a$ except that its value at $v$ is $u$ [9].

Of course, if we have both quantifiers *and* modalities, then we need both possible worlds and value-assignments. The generalization is obvious.

## 4. Abstraction

Although quantifiers are but one type of variable-binders, it is evident that the same treatment can be applied to other cases, including mathematical operators such as integrals, products, etc. The general question is whether one can by the same pattern account for all forms of variable binding.

Every mode of variable binding can be reduced to functional abstraction. So in the end the question reduces to whether abstraction can be interpreted as a form of application, using models whose domains of interpretation depend on a suitable package of intensional features and value-assignments.

Some forms of abstraction are immediately captured by the above treatment. For instance, we can enrich an elementary language $L$ with an abstractor $\lambda v$ for each variable $v$ $V$, to be treated as a functor of type $0/(1/0)$. The standard interpretation of this functor is reflected in the reading "is something $v$ such that". And it is easy to verify that within the present framework, this reading translates into the following condition on the admissible elementary models:

($\lambda$) if $s_{\xi,t}=\lambda v$, then $d_{\xi,t}(x)(y)(a)=x(a^v_{y(a)})$ for all $x$ $I_0$, $y$ $I_1$ and $a$ $U^V$.

In the general case, where we have abstractors acting on variables of any type in expressions of any type, the account is not so straightforward. In fact it is clear that we cannot go very far if we stick to standard Fregean models, for the presence of functor variables prevents us from defining adequate intensional models where each functorial domain is a set of functions $I_{t'/t}=I_t^{I_{t'}}$. However, we need not do that. We only need consider models whose domains are built *upon* sets of functions — and that can be done in the appropriate way to obtain the desired result. This is a rather natural generalization, familiar from intensional logics and Montague grammars. The details are as follows.

To allow for generalized abstractors, we consider a full categorial language $L$ comprising a non-empty set $S_t$ of symbols for all $t$ $T$. Each $S_t$ includes a subset $V_t$ of

variables so that the string $\lambda vt'$ is a symbol of type $t'/(t/t')$ for all $t' \in T$ and all $v \in V_t$. Now let $U_t : t \in T$ be a system of sets so that $U_0 = 2$ and $U_{t'/t} = U_t^{U_{t'}}$ for all $t,t' \in T$ and define $A = \prod U_t^{V_t} : t \in T$. To obtain an adequate model $M$ we simply need require that $I_t = U_t^A$ for all $t \in T$. We can then make sure that each $\lambda vt'$ be interpreted as a $v$-binding abstractor by requiring $M$ to also satisfy the following general condition:

($\lambda$) if $s_{\xi,t} = \lambda vt'$, then $h(d_{\xi,t}, x)(a)(y) = x(a^v_y)$ for all $x \in I_{t'}$, all $y \in I_{\tau(v)}$ and all $a \in A$

where $\tau(v)$ is the type of $v$. Along with the obvious conditions on the interpretation of constant and variable symbols, it can be verified that this clause conforms to the usual principles of the classical $\lambda$-calculus.

## 5. Logics as theories

The treatment above provides an answer to the question of the title. If we think of functional abstraction as a new operation in addition to functional application, then we do not *need* it in order to increase the combinatorial force and descriptive power of a pure categorial framework. For there is a clear sense in which abstraction can be interpreted within such a framework.

I am not sure whether this suffices to justify a claim of generality. But the fact that we can analyze a language with variable-binders by the simple functor/argument schema is not unattractive, also in relation to some fundamental issues in semantics. Often the peculiar behavior of such operators is taken to imply that some crucial aspects of our language lie outside, or perhaps "above", the realm of pure semantics: some symbols *must* receive a fixed interpretation because they are vehicles of crucial logical principles that we need keep in mind when we spell out the semantic framework we intend to use. This is typical, for instance, in a rather customary model-theoretic way of presenting first-order logic, where the meaning of quantifiers and other logical words is characterized only indirectly through the recursive definition of the truth-value of a sentence. Likewise, in $\lambda$-equipped categorial languages the meaning of the abstraction operator is usually fixed during a recursive definition of the value of an expression: it is not specified by a model, like the meaning of the other symbols, and it is not therefore captured by the homomorphism that relates a language with its models; rather, it is imposed upon the entire semantic machinery. To some extent this is just a matter of convenience: if we are not going to consider other ways of interpreting those symbols, there is no need to do otherwise. But the question is whether one could do otherwise, whether one could in principle specify the semantics of a language without at the same time imposing a logic upon it (which is not, of course, a question of whether a semanticist could live without logic: semantics is set theory, and *that* surely embeds quite a bit of logic).

As I see it, there has never been much clarity on this point. My suggestion is that within the approach outlined here the question admits of a clear answer. What is attractive in a categorial approach is that it leads to a general framework fixed in advance, rather than a theory to be worked out afresh for each case: all we need to do is to provide a syntactic category for the symbols we want to study (eventually along with a suitable structural operation) and then specify which, among the indefinitely many structures that give a homomorphic interpretation of the language, are to count as "admissible" models. Insofar as quantifiers and, more generally, functional abstractors can be treated as ordinary functors, one has good grounds to regard these symbols as being *on a par* with all the others — hence good grounds to treat logics as true *theories* in the usual semantic sense.

From this point of view, the proposed account justifies a rather strong form of semanticism. To treat something as, say, a symbol with a certain algebraic meaning (e.g. addition) is to select a certain class of models as the only algebraically admissible ones (e.g. only models where $I_1=\omega$ and the addition symbol denotes that function $f: \omega \rightarrow \omega^\omega$ so that $f(x)(y)=x+y$ for all $x,y \quad \omega$). Now a similar characterization can be applied to logic. To treat something as a "logical" word is to select a certain class of models as the only logically admissible ones. This was evident in the case of purely truth-functional connectives; less evident perhaps in the case of intensional connectives such as those of modal logics — even less evident, I think, in the case of such peculiar logical words as quantifiers and abstractors.

## *Notes*

[1] This view can be traced as far back as to Ajdukiewicz [1935].

[2] Early results in this sense were obtained by Bar-Hillel et al. [1960], and have since become part of the standard view on the limits of categorial grammars.

[3] To my knowledge, this connection was first explored by Cresswell [1973], who conjectured that all "semantically significant" transformational derivations can be seen as sequences of λ-conversions.

[4] Terminology is varied here. Husserl [1913] spoke of meaning categories, Leś niewski [1929] and Ajdukiewicz [1935] of semantic categories, later authors of grammatical or syntactic categories, or simply categories, or types. I follow this last option mainly for its intuitive neutrality.

[5] There is no need to consider types of the form $t_1 \ldots t_n/t'$ for $n>1$, corresponding to those categories of $n$-place functors that take expressions of type $t_1,\ldots, t_n$ to make expressions of type $t'$: as it turns out, such functors can always be treated as $1$-place functors of type $t_1 /(t_2 /(\ldots/(t_n/t')\ldots))$. The point is due to Schönfinkel [1924] and reflects the set-theoretic isomorphism between $A^{B_1\times B_2\times \ldots B_n}$ and $(\ldots((A^{B_1})^{B_2})\ldots)^{B_n}$.

[6] Notation: if $f$ is a function, I write $Df$ for the domain and $Rf$ for the range.

[7] It is possible to further generalize the categorial approach so as to allow for incomplete and/or inconsistent models, but in the present context such a departure from standard semantics would lead us too far afield.

[8] This is in the spirit of Lewis [1970].

[9] It is understood that the values of functional application should not depend on value-assignments unless the arguments do, i.e. one should have $x(y)(a_i)=x(y)(a_j)$ whenever $x(a_i)=x(a_j)$ and $y(a_i)=y(a_j)$. Also, such values should behave coherently, so that $x_i(y)(a)=x_j(y)(a)$ if $x_i(a)=x_j(a)$, and $x(y_i)(a)=x(y_j)(a)$ if $y_i(a)=y_j(a)$.

## *References*


Ajdukiewicz, K., 'Die syntaktische Konnexität', *Studia Philosophica* **1** (1935), 1-27.

Bar-Hillel, Y., Gaifman C., Shamir E., 'On Categorial and Phrase-Structure Grammars', *The Bulletin of the Research Council of Israel* **3** (1960), 1-16.

Cresswell, M. J., *Logics and Languages*, London: Methuen, 1973.

Husserl, E., *Logische Untersuchungen*, Halle: Niemeyer, 1900-1901.

Leś niewski, S., 'Grundzüge eines neuen Systems der Grundlagen der Mathematik', *Fundamenta Mathematicae* **14** (1929), 1-81.

Lewis, D. K., 'General Semantics', *Synthese* **22** (1970), 18-67.

Montague, R., 'Universal Grammar', *Theoria* **36** (1970), 373-398.

Schönfinkel, M., 'Über die Bausteine der mathematischen Logik', *Mathematische Annalen* **92** (1924), 305-316.