Quine and Variables

Jaroslav Peregrin

Abstract. In *Word and Object*, Quine presents his project of the regimentation of natural language, one step of regimentation being the introduction of variables. This is important as it allows him to articulate his famous slogan: *to be is to be the value of a variable*. It looks like a paradox that Quine, who takes such pains to convince us about the usefulness and indispensability of variables, shows us at the same time how to dispense with variables. Specifically, in a paper that appeared in the very same year as *Word and Object*, he shows how the language of the first-order predicate logic can be translated into a completely variable-less language. The aim of this paper is to throw some new light on this apparent paradox.

Quine on the indispensability of variables

In *Word and Object*, Quine (1960a) presents his project of the regimentation of natural language, one step of regimentation being the introduction of variables. Quine introduces them as sort of *ad hoc* names. Just like somebody who wants to replicate a novel, the names of the many protagonists of which she has forgotten, will do well if she gives the characters some provisional names, one who wants to describe a complicated situation such as (Quine's example):

(1) A lawyer told a colleague that he thought a client of his more critical of himself than of any of his rivals,

will do well if she regiments the description using variables:

(1') A lawyer *x* told a colleague *y* that *x* [or *y* ?] thought a client *z* of *y* [or *x* ?] more critical of *z* [or *y* ? or *x* ?] than of any of *z*'s [or *y*'s? or *x*'s?] rivals.

Used in this way, variables are anchored to concrete persons or objects and so they function as 'nouns and enhanced pronouns'. If we use a variable without such an anchoring, like

z is more critical of *z* than any of *z*'s rivals,

the result will not be of course a meaningful sentence. But instead of linking a variable to a concrete object, we can let it be bound by a quantifier, linking

(2) there is a *z* such that *z* is more critical of *z* than any of *z*'s rivals,

or, regimented into the form of standard logic,[1]

[1] Where ***rival***(*x*,y) is the regimentation of *x is a rival of y*; while ***more-critical***(*x*,*y*,*z*) is a regimentation of *x is more critical of y than of z*.

(2') $\exists z \forall x(\textbf{\textit{rival}}(x,z) \rightarrow \textbf{\textit{more-critical}}(z,z,x))$.

From this viewpoint, it would seem that variables are indispensable for the process of regimentation.

However, this is not yet the end of the story regarding the indispensability of variables for Quine. For it is regimentation using variables that is, according to Quine, the key to 'ontic commitment'. Quine would reject the question *What is there*? and in its stead moves to the question *What is a language or a theory ontologically committed to*? or *What is there according to the language or the theory*? And the answer, given an adequate regimentation, is quite simple: *to be is to be the value of a variable* (Quine, 1953, pp. 12–13):

> We can very easily involve ourselves in ontological commitments by saying, for example, that there is something (bound variable) which red houses and sunsets have in common; or that there is something which is a prime number larger than a million. But this is, essentially, the only way we can involve ourselves in ontological commitments: by our use of bound variables. The use of alleged names is no criterion, for we can repudiate their namehood at the drop of a hat unless the assumption of a corresponding entity can be spotted in the things we affirm in terms of bound variables. Names are, in fact, altogether immaterial to the ontological issue, for I have shown, in connection with 'Pegasus' and 'pegasize', that names can be converted to descriptions, and Russell has shown that descriptions can be eliminated. Whatever we say with the help of names can be said in a language which shuns names altogether. To be assumed as an entity is, purely and simply, to be reckoned as the value of a variable. In terms of the categories of traditional grammar, this amounts roughly to saying that to be is to be in the range of reference of a pronoun. Pronouns are the basic media of reference; nouns might better have been named propronouns. The variables of quantification, 'something', 'nothing', 'everything', range over our whole ontology, whatever it may be; and we are convicted of a particular ontological presupposition if, and only if, the alleged presupposition has to be reckoned among the entities over which our variables range in order to render one of our affirmations true.

Quine on the dispensability of variables

It seems a paradox that Quine, who takes such pains to convince us about the usefulness and indispensability of variables, shows us at the same time how to dispense with variables. In particular, in a paper that appeared in the very same year as *Word and Object*, Quine (1960b, p. 344) writes:

> We were able to avoid the variable as abstractive pronoun in the case of '$x^3 = 3x$' by torturing *'$x^3 = 3x$' into 'x gives the same result when cubed as when trebled.' Once we had coaxed t*he dummy letter '*x*' thus into suitable position and segregated the rest, we

> ceased to need the '*x*.' In this example, the coaxing depended on such auxiliary words as 'gives,' 'result,' and 'when,' along with participial endings of verbs. Now my question is whether a general, finite battery of such auxiliary operators can be assembled that will enable us always to coax variables thus into positions where we can dispense with them. The answer is affirmative, as I shall show.

To resolve this apparent paradox is not difficult. Quine himself writes (*ibid*.):

> As a point now of theory and not of practical convenience, it can be interesting to inquire whether the variable is in principle dispensable. … There is no thought of denying ourselves the continuing convenience of variables in practice.

Hence variables, according to Quine, are practically indispensable, though theoretically dispensable. We can make do without them, but the price of their elimination is such that we are practically unable to do the useful things we can do with them.

The birth of variable[2]

Variables have been, of course, used in mathematics for many centuries (especially as representations of 'unknowns'). However, to understand why variables have become so important for logic, let us return to the end of the 19th century, when our artificial languages of logic came to be born. There we can see that the concept of variable, as nowadays employed within logic, as having risen inseparably connected with the concept of quantifier developed by Gottlob Frege and his followers.[3] When introducing quantifiers, Frege says roughly this: Imagine a sentence decomposed into two parts and imagine one of the parts 'abstracted away', thus turning the sentence into an 'unsaturated', gappy torso. Then think of the gap in this matrix as being filled with various things and consider the truth values of individual cases – the truth value of a corresponding quantificational sentence then can be computed from these values.[4] A variable

[2] This and the following sections incorporate a few paragraphs from my earlier paper (Peregrin, 2000), which is no longer easily accessible.

[3] See Peregrin (2020, §9) for more details.

[4] This consideration has later come to be seen as ambiguously standing between the 'substitutional' and 'objectual' version. According to the 'substitutional' version of the story, we fill the gap literally: we replace the variable by a suitable expression, thus saturating the matrix 'syntactically'. According to the 'objectual' version, on the other hand, we make the variable stand for a suitable object, thus saturating the matrix 'semantically'. If we consider the matrix '*x* conquered Gaul' (which might have arisen, e.g., out of the sentence 'Caesar conquered Gaul' via taking away 'Caesar'), then the 'substitutional' 're-saturation' would consist in replacing *x* by various names ('Caesar', 'Aristotle', 'Cartman', ...) and thus turning the matrix into various sentences ('Caesar conquered Gaul', 'Aristotle conquered Gaul', 'Cartman conquered Gaul', ...), whereas the 'objectual' one would consist in making *x* stand for various individuals (Caesar, Aristotle, Cartman, ...) and thus making the matrix express various propositions (that Caesar conquered Gaul, that Aristotle conquered Gaul, that Cartman conquered Gaul, ...).

is then a symbol that is employed to mark the gap(s). Thus, in his *Begriffsschrift,* Frege (1879, p. 19) writes:

> In the expression for a judgement, the complex symbol to the right of ⊢ may always be regarded as a function of one of the symbols that occur in it. *Let us replace this argument with a Gothic letter, and insert a concavity in the content-stroke, and make this same Gothic letter stand over the concavity, e.g.:*
>
> $$\vdash \overset{\mathfrak{a}}{\smile} \Phi(\mathfrak{a}) \qquad (9)$$
>
> *This signifies the judgement that the function is a fact whatever we take its argument to be.*

Later, Frege came to see the 'de-saturation' which underlies quantification as only a special case of 'functionalization', i.e. of the process of making a linguistic gap to define a function. Thus in *Funktion und Begriff,* Frege (1891, p. 8) writes:

> We recognize the function in the expression by imagining the latter as split up, and the possibility of thus splitting it up is suggested by its structure. ... For instance, if I say 'the function $2 \cdot x^3 + x$', *x* must not be considered as belonging to the function; this letter only serves to indicate the kind of supplementation that is needed; it enables one to recognize the places where the sign for the argument must go in.

This indicates that for Frege variables played a merely auxiliary role of 'gap-markers': they helped turn expressions into unsaturated torsos which can be seen as indicating functions (not *denoting* functions, for unsaturated expressions are not names and thus do not denote anything)[5] and which can yield saturated expressions (names, especially sentences) not only by their gaps being filled with names, but also by the gaps (i.e. variables) being 'bound' by quantifiers (or 'abstracted away' by Frege's operator **'**, which was the counterpart of the modern λ).

Free variables

Let us now consider free variables in open formulas. We may take them as mere stepping stones on the way to closed formulas. Seen thus, a formula F containing *x* free is only an intermediary to formulas such as ∃*xF* and ∀xF (or, more generally, λ*xF*[6]). Their usefulness consists in the fact that

[5] Frege distinguished between a function and the course of values of the function. His function is something which is almost linguistic, something which we now would probably call *rule*. His course of values is the function in the modern sense – in effect a certain set of ordered pairs of objects. Whereas what he calls a *function* is no object, what he calls the *course of values* of a function is, and hence it can be named.

[6] It is clear that λ-abstraction can be seen as the only variable binding operation – once we see quantifiers as second-order predicates, quantification gets decomposed into λ-abstraction and simple application: '∃xF' turns into a shorthand for '∃(λxF)'. This is also, I think, what made Frege see quantification as based on a special case of 'functionalization'.

as stepping stones they are extremely simple and elegant; and they provide for the simple grammar of the predicate calculus. Seen thus, free variables are simply to-be-bound variables.

However, it may be insisted that open formulas are more than this, that they are needed for something more than for arriving at closed formulas. We can perhaps distinguish two versions of this claim: one is that open formulas are needed as *schemes*, i.e. as means of envisaging *types* of closed formulas; the other is simply that there is no reason for denigrating open formulas as less fully-fledged constituents of logical calculi as closed ones. As what interests us here is natural language, it is crucial to see how this view fares from the point of view of the regimentation of natural language; we should ask whether open formulas are in some sense necessary for the regimentation.

It is clear that open formulas are a very useful tool of specifying classes of formulas. For example, the scheme

$$\forall x(\textbf{\textit{rival}}(x,z) \rightarrow \textbf{\textit{more-critical}}(z,z,x))$$

delimits the class

$$\{\forall x(\textbf{\textit{rival}}(x,\textbf{\textit{Alice}}) \rightarrow \textbf{\textit{more-critical}}(\textbf{\textit{Alice}}, \textbf{\textit{Alice}},x)),$$

$$\forall x(\textbf{\textit{rival}}(x,\textbf{\textit{Bob}}) \rightarrow \textbf{\textit{more-critical}}(\textbf{\textit{Bob}}, \textbf{\textit{Bob}},x)),$$

$$\ldots\}$$

(and thereby the class of people who are more critical to themselves than to any of his/her rivals). As in logic we usually treat kinds of sentences rather than specific sentences, this becomes useful. However, these schemes themselves are regimentations of no natural language sentences.

Also, if we consider an artificial language of logic as a self-contained entity, there is no reason to see variables and open formulas as their less natural or less genuine parts than constants. (True, when it comes to semantics, variables are interpreted differently from constants, but so what?). However, in this case we must renounce any talk of regimentation or about a relationship of the language to another language or to reality. Hence, this is a case that we are not interested in.

Bound variables

Let us now concentrate on *bound* variables, i.e. variables which are within the scope of a quantifier. The important thing to notice is that *they are not essential*: we can do logic (predicate calculus) wholly without them. What does this mean?

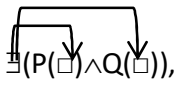
It is well known that we can (as Polish logicians demonstrated) do predicate calculus wholly without parentheses; and this fact is usually taken to show that parentheses are idiosyncratic to *one specific* way of articulating the syntax of the calculus, and they are not essential for the calculus as such. It is less well known that precisely the same holds for (bound) variables.

Why are bound variables not essential? Informally speaking, the reason is that their role is merely auxiliary, it is a role that can also be played by other things. The analogy with parentheses may be helpful: parentheses are dispensable for their task is to specify the order in which logical operators apply, and they are no longer needed if we use prefix (instead of the infix) notation – for then the order is unique. Likewise, variables are inessential as their role is simply that of a clamp (they connect quantifiers with the appropriate places within the matrix), and this can be provably accomplished using other means.

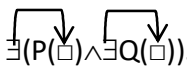
To illustrate *how* it is possible to rid ourselves of variables, first think about quantifiers attached only to atomic formulas with unary predicate constants. Then we obviously need no variables at all: we can treat quantifiers syntactically on par with ordinary individual constants and write, e.g., P(∃) instead of ∃xP(x) (if we want to make explicit the essential *semantic* difference between quantifiers and individual constants, we can make predicates the arguments of quantifiers rather than vice versa and thus write ∃(P)). If we now allow for binary predicate constants, the situation changes: we cannot write R(∃,∀) for this would be ambiguous between ∃x∀yR(x,y) and ∀y∃xR(x,y). Another problem arises when quantifiers are attached to non-atomic formulas (even when these contain only unary predicate constants): P(∃)∧Q(∃) would be ambiguous between ∃x(P(x)∧Q(x)) and (∃xP(x))∧(∃xQ(x)). However, the needed disambiguation can be carried in ways which *avoid* variables: we can, e.g., take inspiration from Bourbaki (1958) and mark the linear order of the quantifiers by means of using clamps, so that ∃x∀yR(x,y) will be

whereas ∀y∃xR(x,y) will be

and ∃x(P(x)∧Q(x)) will be



while ∃xP(x)∧∃xQ(x) will be



An obvious objection is that such clamp notation would not lead to a syntax that could support a reasonable compositional semantics. This may be right, but what Quine and others have demonstrated is precisely that there *does* exist a syntax which underlies compositional semantics and which manages without variables. Let us sketch the idea of these proposals.

First, for the sake of simplicity, we shall consider monadic first-order predicate calculus (i.e. first-order predicate calculus involving no predicate constants of arity greater than one). The syntax of this calculus is based on the following rules:

(i) A (unary) predicate plus a term (where a term is an individual constant or a variable) gives a formula.

(ii) $\neg$ plus a formula gives a formula.

(iii) $\wedge$ ($\vee$, $\rightarrow$) plus two formulas gives a formula.

(iv) $\exists$ ($\forall$) plus a variable plus a formula gives a formula.

The basic idea for dispensing with variables is to treat a quantifier as something which can yield a formula together with a *predicate*. This is fine, we have seen, with formulas like $\exists$xP(x), which can be seen directly as the combination of a quantifier with a predicate constant ($\exists$ with P). Problems, though, arise with formulas like $\exists$x(P(x)$\wedge$Q(x)) – for such formulas need to be seen as the combination of a quantifier with a *complex* predicate ($\exists$ with the conjunction of P and Q), and we lack the way to form a conjunction of predicates. The remedy, however, is to add such a way to our syntax; one of the possibilities to do this is to replace (iv) with (iv') and to add (v) and (vi):

(iv') $\exists$ ($\forall$) plus a predicate gives a formula.

(v) $\neg$ plus a predicate gives a predicate.

(vi) $\wedge$ ($\vee$, $\rightarrow$) plus two predicates gives a predicate.

Then, writing predicates as arguments of quantifiers,[7] we can clearly reflect the difference between $\exists$x(P(x)$\wedge$Q(x)) and ($\exists$xP(x))$\wedge$($\exists$xQ(x)) as that between $\exists$(P$\wedge$Q) and $\exists$(P)$\wedge\exists$(Q). The situation complicates itself, of course, when we consider full (i.e. non-monadic) first-order predicate calculus; for the presence of predicate constants of arities greater than one necessitates further rules for predicates. Imagine a 'predicate modifier' Ex which exchanges the slots of a binary predicate (that is, it maps a binary predicate *R* on a predicate Ex(*R*) such that Ex(*R*)(*a*,*b*) is *R*(*b*,*a*)), then $\forall x\exists y\, R(x,y)$ can be simply $\exists\forall R$ while $\exists y\forall x\, R(x,y)$ is $\forall\exists$Ex(*R*). Or a modifier Refl such that Refl(R)(x) is R(x,x) provides for what could be called the 'reflexivization' of the binary predicate R.

All of this indicates that *(bound) variables may not be essential elements of the predicate calculus*; they are better seen as syncategorematic symbols on par with parentheses.

[7] This kind of syntax does justice to the fact that if we treat quantifiers as categorematic terms (like within lambda calculus – see Church, 1940), we usually treat them as second-order predicates (unary predicates predicable of unary predicates), hence as denoting classes of classes of individuals.

Combinators

Probably the only person who contributed to the problem of elimination of variables (thought it may have not been his main goal) in the first quarter of the 20th century was Moses Schönfinkel. A somewhat mysterious Russian, who published, in his life, only two articles, presented what would later become the foundations of the so-called combinatory logic.[8]

We can look at what Schönfinkel (1929) did as specifying a set of predicate modifiers of the kind of Ex and Refl from the previous paragraph. As predicates can be also understood as functions (from *n*-tuples of individuals into truth values), he brought the whole problem into the space of functions, where functions can be arguments of other functions etc. He defines several specific functions to which, in a sense, we can reduce all of standard logic. Of these, three are crucial, because the rest of the functions can be reduced to them

$(Cx)y \equiv_{Def.} x$

$((Sx)y)z \equiv_{Def.} (xy)(xz)$

$(Ux)y \equiv_{Def.}$ T iff $x \cap y = \varnothing$; $\perp$ otherwise

We may build other interesting combinators such as

$I \equiv_{Def.} SKK$,

which gives us the identity functor, for $Ix = SKKx = Kx(Kx) = x$; or the operator

$Z \equiv_{Def.} S(CS)C$,

of composition of functions, as $Zfgx = S(CS)Cfgx = (CSf)(Cf)gx = S(Cf)gx = (Cfx)(gx) = f(gx)$.

Schönfinkel's own example is the statement

(3) For every predicate there exists a predicate incompatible with it

i.e.

(3') For every predicate f there exists a predicate g such that the pro-positional function fx & gx is not true of any object x:

$\forall f \exists g \forall x \neg(fx \wedge gx) =$

$\forall f \exists g\ (fx \mid^x gx)$ (where $(fx \mid^x gx) \equiv_{Def.} \forall x \neg(fx \wedge gx)) =$

$\forall f \neg \forall g \neg(fx \mid^x gx) =$

$\forall f \neg(fx \mid^x gx \mid^g fx \mid^x gx) =$

$\forall f \neg(fx \mid^x gx \mid^g fx \mid^x gx) \wedge (fx \mid^x gx \mid^g fx \mid^x gx)] =$

$((Ufg) \mid^g (Ufg) \mid^f ((Ufg) \mid^g (Ufg)] =$

$(U(Uf)(Uf) \mid^f (U(Uf)(Uf)] =$

[8] See Wolfram (2020).

(S(ZUU)Uf) $|^{f}$ (S(ZUU)Uf) =

U(S(ZUU)U)(S(ZUU)U)

There doesn't seem to have been any significant response to the paper until Haskell Curry (1929), who then built combinatory logic on its foundations.[9] But combinatory logic then amounted to the elimination of variables on a very general level. Arguably, the most general calculus based on variables and their binding is the lambda calculus due to Church (1932). And this calculus is translatable into Curry's combinatory logic and amounts to the proof that we can make do without variables even on this most general level.[10]

Predicate functors

As we have already seen (and as each student of logic knows), $\forall x \exists y R(x,y)$ is something different from $\exists y \forall x R(x,y)$, which renders a sentence such as

Every man reads a book

ambiguous. Also, it underscores the import of logical regimentation because it uncovers this ambiguity. And it seems that variables are an indispensable tool of this disambiguation.

We have also already seen that the problem is this. Given a predicate, we need to know the order of its arguments, *viz*. which of the arguments fills which place in the predicate (for $R(a,b)$ is, of course, something different from $R(b,a)$). But in case the arguments are quantified over, there is also a *different* kind of order, *viz*. the order of the quantifiers (for $\forall x \exists y R(x,y)$ is something different from $\exists y \forall x R(x,y)$). And variables are tools which solve this problem elegantly. But Quine shows that it also works without them.

Quine (1960b) assumes that quantifiers are always connected with the slots of the predicate in the order in which they are listed in front of it and he introduces ways of permuting the slots. Thus, if we have a functor Ex which exchanges the slots of a binary predicate (that is, it maps a binary predicate R on a predicate Ex(R) such that Ex(R)(a,b) is $R(b,a)$), then $\forall x \exists y\, R(x,y)$ can be simply $\exists \forall R$ while $\exists y \forall x\, R(x,y)$ is $\forall \exists \mathrm{Ex}(R)$. (Quine introduces a minimalistic set of such functors with which we can accomplish everything we need.)

Note that Quinean predicate functors are different from combinators. Combinators take, as arguments, other functions, including themselves. Predicate functors take predicates and produce predicates. From the grammatical viewpoint they are predicate modifiers or adverbs. Despite this, predicate functors and combinators may serve a common goal – elimination of variables – they go different ways. Can we have a set of functors such that we will not need any

[9] See Curry, Feys & Craig (1958); Curry, Hindley & Seldin (1972).

[10] See also Cardone and Hindley (2006).

variables whatsoever? Quine (1960b) shows that this is possible. The following enhanced set of functors is taken from Quine (1995);[11]:

$(\exists\ F)x_2 \ldots\ x_n \equiv_{\text{Def.}} \exists x_1\ Fx_1 \ldots\ x_n.$

$(\text{Pad}\ F)x_0 \ldots\ x_n \equiv_{\text{Def.}} Fx_1 \ldots\ x_n.$

$(\text{Refl}\ F)x_2 \ldots\ x_n \equiv_{\text{Def.}} Fx_2\ x_2\ x_3 \ldots\ x_n.$

$(\text{Perm}\ F)x_1\ x_3 \ldots . x_n\ x_2 \equiv_{\text{Def.}} Fx_1 \ldots\ x_n.$

$(-\ F)x_1 \ldots\ x_n \equiv_{\text{Def.}} \neg Fx_1 \ldots\ x_n.$

$(F\ G)x_1 \ldots\ x_n \equiv_{\text{Def.}} Fx_1 \ldots\ x_n \wedge Gx_1 \ldots\ x_n.$

Now let us see how we can use them to do away with the variables in (2'). First, let us transform (2') into a conjunction:

$\exists z \forall \text{x}\ (\textbf{\textit{rival}}(x,\text{z}) \rightarrow \textbf{\textit{more-critical}}(z,z,x))$

$\exists z \forall \text{x}\ (\neg\textbf{\textit{rival}}(x,\text{z}) \vee \textbf{\textit{more-critical}}(z,z,x))$

$\exists z \forall \text{x}\ \neg(\textbf{\textit{rival}}(x,\text{z}) \wedge \neg\textbf{\textit{more-critical}}(z,z,x)))$

(2'') $\exists z \neg \exists \text{x}\ (\textbf{\textit{rival}}(x,\text{z}) \wedge \neg\textbf{\textit{more-critical}}(z,z,x)))$

Now take the second conjunct, $\neg$***more-critical***(z,z,x), and let us transform it using the definitions of the predicate functors:[12]

$\neg$***more-critical***$(z,z,x) \equiv$

$-$ ***more-critical***$(z,z,x) \equiv$

Refl $-$ ***more-critical***$(z,x) \equiv$

Pad Refl $-$ ***more-critical***$(\text{y},z,x) \equiv$

Perm Pad Refl $-$ ***more-critical***$(\text{y},x,z) \equiv$

$\exists$ Perm Pad Refl $-$ ***more-critical***(x,z)

Now let us substitute this into (2'') and let us go on with the transformations:

$\exists z \neg \exists \text{x}\ (\textbf{\textit{rival}}(x,\text{z}) \wedge \exists$ Perm Pad Refl $-$ ***more-critical***$(\text{x},z))) \equiv$

$\exists z \neg \exists \text{x}\ (\textbf{\textit{rival}}(x,\text{z}) \wedge \exists$ Perm Pad Refl $-$ ***more-critical***$(\text{x},z))) \equiv$

$\exists -\ \exists$ ***rival*** $\exists$ Perm Pad Refl $-$ ***more-critical***

[11] There are additional publications that Quine devoted to predicate functors; see Quine (1971; 1981). See also Kuhn (1983).

[12] As predicate functors are applicable only to predicates (and not to functors), we can leave out most of the brackets, without going ambiguous.

In this way we have transformed (2') into a formula containing no variables – being a 0-ary predicate.[13]

The role of variables, then, aside from making the life of the analysts of language easier, is to dispel any air of mystery that might surround variables, special symbols that do not refer to anything and despite this contribute to the meaning of formulas which they co-form (Quine, 1995, p. 35):

> The philosophical importance of the predicate functors is the resolution of the variable's magic into its pedestrian components, which are matters purely of order and recurrence of reference.

And if we embrace predicate functors, what about the 'to be is to be the value of a variable' slogan? Does it vanish into thin air? Not really – it just gets reformulated (*ibid*.):

> In a predicate-functor culture, to be is to be denoted by a one-place predicate.

Conclusion

It is important not to conflate the extreme usefulness of variables for the artificial languages of logic and the regimentation of natural language with their indispensability. If we see them as indispensable, it is easy to see variables not as invented tools but rather as something that was here from the start and that we only discovered and made explicit.[14] It is easy to say that certain constructions of natural language are, 'in fact', variable binding, and we can see them for what they really are only after we unmask them by carrying out the logical analysis. Fortunately, Quine documented that this is not the case and that variables are, despite all their usefulness, only dispensable tools.

References

Bourbaki, N. 1958. *Élements de Mathématique I: Théorie des ensembles*. Paris: Hermann.

Cardone, F., & Hindley, J. R. 2006. 'History of Lambda-calculus and Combinatory Logic'. In D. M. Gabbay and J. Woods (eds.), *Logic from Russell to Church* (*Handbook of the History of Logic*: Volume 5). Amsterdam: Elsevier, pp. 732–817.

Church, A. 1932. 'A Set of Postulates for the Foundation of Logic'. *Annals of Mathematics*, Series 2, 33 (2), pp. 346–366.

[13] Quine is well-known for repudiating any logic beyond the boundaries of the first-order one. Is it possible to eliminate variables also in other calculi? Yes. In a half-forgotten paper, Došen (1988), for example, shows us that it works in second-order logic.

[14] Cf. Shapiro (1998, p. 139): 'parentheses are artifacts, serving to make the formulas unambiguous, variables, connectives, and quantifiers do have counterparts in natural languages, more or less'.

Church, A. 1940. ‘A Formulation of the Simple Theory of Types’. *The Journal of Symbolic Logic*, *5*(2), pp. 56–68.

Curry, H. B. 1929: ‘An Analysis of Logical Substitution’. *American Journal of Mathematics* 51, pp. 363–384.

Curry, H. B., R. Feys, R., & Craig, W. 1958. *Combinatory Logic*, vol. 1, Amsterdam: North-Holland.

Curry, H. B., Hindley, J. R., & Seldin, J. P. 1972. *Combinatory Logic*, vol. 2, Amsterdam: North-Holland.

Došen, K. 1988. ‘Second-order Logic Without Variables’. In *Categorial Grammar* (ed. W. Buszkowski, W. Marciszewski and J. van Benthem), pp. 245–264. John Benjamins Publishing.

Frege, G. 1879. *Begriffsschrift*. Halle: Nebert.

Frege, G. 1891. *Funktion und Begriff: Vortrag, gehalten in der Sitzung vom 9. Januar 1891, der Jenaischen Gesellschaft für Medizin und Naturwissenschaft*. Jena: Pohle.

Kuhn, S. T. 1983. ‘An Axiomatization of Predicate Functor Logic’. *Notre Dame Journal of Formal Logic*, *24*(2), 233–241.

Peregrin, J. 2000. ‘Variables in Natural Language: Where Do They Come From?’ *Variable-free Semantics* (ed. M. Böttner & W. Thümmel), pp. 46–65. Osnabrück: Secolo.

Peregrin, J. 2020. *Philosophy of Logical Systems*. New York: Routledge.

Quine, W. V. O. 1981. ‘Predicate Functors Revisited’. *Journal of Symbolic Logic*. 46(3):649-652.

Quine, W. V. O. 1953. *From a Logical Point of View*. New York: Harper and Row.

Quine, W. V. O. 1960a. *Word and Object*. Cambridge (Mass.): MIT Press.

Quine, W. V. O. 1960b. ‘Variables Explained Away’. *Proceedings of the American Philosophical Society*, *194*, pp. 343–347.

Quine, W. V. O. 1971. ‘Algebraic Logic and Predicate Functors’, *Logic and Art: Essays in Honor of Nelson Goodman* (ed. R. Rudner and I. Scheffler). Indianapolis: Bobbs-Merrill; reprinted with emendations in *The Ways of Paradox and Other Essay*s, 2nd edition. Cambridge (Mass.): Harvard University Press, 1976.

Quine, W. V. O. 1995. *From Stimulus to Science*. Cambridge (Mass.): Harvard University Press.

Shapiro, S. 1998: ‘Logical Consequence: Models and Modality’. In M. Schirn (ed.), *Philosophy of Mathematics Today*. Oxford: Oxford University Press, pp. 131–156.

Schönfinkel, M. 1924: ‘Über die Bausteine der mathematischen Logik’, Mathematische Annalen 92, 305-306; English translation ‘On the Building Blocks of Mathematical Logic’, in J. van Heijenoort (ed.), *From Frege to Gödel: A Source Book from Mathematical Logic*. Cambridge (Mass.): Harvard University Press, 1967, pp. 357–366.

Wolfram, S. 2020. ‘Where Did Combinators Come From? Hunting the Story of Moses Schönfinkel’. https://writings.stephenwolfram.com/2021/03/a-little-closer-to-finding-what-became-of-moses-schonfinkel-inventor-of-combinators/